Co-funded by the Erasmus+ Programme of the European Union

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Output Identification: O3

Output Title: Computational Modeling & Simulations in Laser Matter Interactions

O3 - Computational Modeling & Simulations in Laser Matter Interactions

O3 - Theory

O3 – Theory

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Chapter 1: Fundamentals of Numerical Methods

P. Vachal

1 Fundamentals of Numerical Methods

In this chapter we will mention some features and pitfalls of the numerical methods to be used in the following chapters, and comment on their practical implementation. The objective is definitely not to cover the complete subject of numerical mathematics, but rather to remind the reader on some machinery we will build on in the following chapters. In particular, we will touch the topic of controlling the error introduced in numerical calculations, refer to sources and some typical examples of numerical instability, and review a few methods for solving of ordinary differential equations on a computer.

1.1 Selected Types of Errors and Their Sources

Among the types of errors we can encounter in this course are the errors of input, truncation error and roundoff error.

1.1.1 Errors on Input

Before we even start the calculation, we have to assess the quality of data on input. In our field of study, we often have to initialize the calculation by results of some measurements, which naturally suffer from limited accuracy and precision, and sometimes also by data that have been interpolated earlier due to incomplete measurements or postprocessing (e.g., our variables are calculated from values of other variables).

There is not much to do about this fact, one just has to keep the input error in mind in order not to overestimate the accuracy of the calculated results, and try to choose methods which will not amplify the error.

One also has to remember that input errors can not only decrease the accuracy of the outcome, but also principally change its nature. For example, when trying to solve the quadratic equation $a\,x^2 + b\,x + c = 0$ with the values b^2 being close to $4ac$, a minuscule variation in coefficients can change the sign of the discriminant $D = b^2 - 4ac$ and thus not only perturb the value of the roots, but, more seriously, also change whether there are two roots ($D > 0$), one root ($D = 0$), or no solution in real domain ($D < 0$).

1.1.2 Roundoff Error

The roundoff error is an inevitable consequence of storing the numbers and performing arithmetic operations in floating point arithmetics. That is, by using approximate, finite-length numbers instead of exact values. Each real number is stored in binary form in a piece of memory of selected length, e.g. 64 bits for double precision or 128 bits for quad precision. (Arbitrary-precision arithmetics will not be discussed here.) Out of these, one bit is reserved for the sign. The rest is divided between the significand (sometimes also called mantissa), determining the precision, and exponent, determining the range of numbers that can be represented. For a given number, the nearest member of such discrete set is then stored instead of its exact value. Obviously, nobody would expect to store an exact value of irrational numbers such as π, e or $\sqrt{2}$. However, one also should keep in mind, that since most computers use the binary system, they are unable to store exactly even such "nice" numbers as 0.1.

Closely related to the range and precision of chosen floating point arithmetics is the danger of overflow and underflow. A good programming practice is to prevent these in the actual code rather than relying on the compiler. In contrast, division by zero is not a feature of the floating point arithmetics, it's the programmer's stupidity.

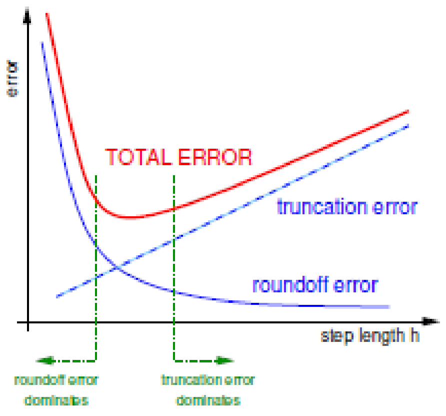


Figure 1: Dependence of error on step size h.

From the brief description of how the numbers are stored in memory it is obvious which arithmetic operations are the most dangerous, that is, which operations tend to increase the relative error. The most notorious one is the subtraction of very similar numbers. Denoting $a(\cdot)$ the relative error of the input and $r(\cdot)$ the absolute error, for addition and subtraction we have

$$a(x \pm y) = a(x) + a(y), \qquad r(x \pm y) = \frac{a(x) + a(y)}{|x \pm y|}. \tag{1.1}$$

Clearly, not only will the relative error dramatically increase for the subtraction of similar x and y because of the small denominator, but also the number of significant (valid) digits in the mantissa (significand) will be substantially reduced. Although this only amplifies the already existing errors, it is very likely that x and y are already approximate values, so we should avoid this situation in practical calculations, where possible. It is also recommended to avoid algorithms which produce intermediate results with values many orders of magnitude bigger than the input and output data.

1.1.3 Truncation Error

The truncation error (also called the error of the method) is an issue principally different from the roundoff error. It arises due to the mathematical approximation of a real, physical problem. And since we translated an analytical mathematical task to a numerical problem, we have to expect the results of an approximation rather than the results of an exact procedure.

For example, in numerical integration or solution of differential equations, we often use techniques which can be derived and/or analyzed with the help of the Taylor expansion and where the infinitesimally short step dx has been replaced by a small but finite step h. This inevitably leads to imprecisions in the calculation.

Unfortunately, the shorter step h we use to reduce the truncation error, the bigger chance we give to the roundoff error to do damage. For example, when choosing a too short step h in integration, we need to perform too many steps where the roundoff error can accumulate, or when choosing a too short step in numerical differentiation, we might run into trouble by subtracting too similar numbers. Therefore, we should always choose the step length in a manner smart enough to maintain the right balance and keep the total error in control as shown schematically in Fig. 1.

1.2 Numerical Stability

There are many definitions of stability in various contexts. For our purpose, we will describe a method or algorithm as unstable, when the accumulation of relatively small errors in particular steps leads to catastrophic loss of accuracy of the numerical solution. In a stable method, the error should increase at most linearly with the number of steps N. (Ideally, but rarely, we can encounter methods where the error evolves proportionally to $\sqrt{N}$.) In unstable methods, the error is amplified superlinearly, sometimes even grows with the geometrical progression q^N with $q > 1$.

There are many examples of methods unstable in this sense. In [1], an example (and its explanation) can be found, where a recursive algorithm to calculate high powers of the "golden mean" $\Phi = (\sqrt{5}-1)/2$ fails due to exponential growth of an initially small roundoff error. The instability can also be caused by the accumulation of the truncation error, typically in the solution of ordinary differential equations (ODE), where the stability often depends on the step length h. For example, solving the ODE $y'(x) = -y,\ y(0) = 1$ by a two-step method

$$y' \approx \frac{y(x+2h) - y(x)}{2h} = -y(x+h)$$

is unstable, unlike even the simplest (but not very accurate) Euler's method $y(x+h) = y(x) - h\,y(x)$. Further examples of instability can involve the approximation of periodic functions by cubic splines with improper choice of conditions (providing the value of the function and its derivative at one point instead of at different points). See the presentation corresponding to this text for details.

Note that in all these cases a hypothetical calculation with exact values in exact arithmetics would provide correct results.

1.3 Solving Ordinary Differential Equations

We will consider a scalar first-order ordinary differential equation (ODE) in one dimension, which can be expressed in the form

$$\frac{\mathrm{d}\,y(x)}{\mathrm{d}x} = f(x, y(x)), \qquad y(x_0) = y_0. \tag{1.2}$$

That is, we want to reconstruct the function $y(x)$, knowing its value y_0 at certain point x_0 (typically the left end of the computational domain) and at any point its derivative, prescribed by the function $f(x, y)$.

A system of first-order ODEs can be written similarly, just with y and f being vectors.

Note that a general scalar ODE of N-th order

$$f\left(x, y, y', y'', \ldots, y^{(N)}\right) = g\left(x\right) \tag{1.3}$$

with N additional conditions can be transformed using the substitutions

$$z_1 \equiv y', \qquad z_2 \equiv y'', \qquad \ldots, \qquad z_{N-1} = y^{(N-1)} \tag{1.4}$$

into a system of N first-order ODEs

$$\begin{aligned} y' &= z_1, \\ z_1' &= z_2, \\ z_2' &= z_3, \\ &\vdots \\ z_{N-2}' &= z_{N-1}, \\ f\left(x, y, z_1, z_2, \ldots, z_{N-1}, z_{N-1}'\right) &= g\left(x\right), \end{aligned} \tag{1.5}$$

where the last one can be typically (but not necessarily) expressed w.r.t. z'_{N-1} as

$$z'_{N-1} = \bar{g}\left(x, y, z_1, z_2, \ldots, z_{N-1}\right). \tag{1.6}$$

Equations (1.5) are then solved consecutively one by one, from the last one up.

For example, for a second-order ODE

$$\frac{\mathrm{d}^2 y(x)}{\mathrm{d}x^2} + r(x)\frac{\mathrm{d}y(x)}{\mathrm{d}x} = q(x) \tag{1.7}$$

with given functions $r(x)$ and $q(x)$, we introduce an additional function $z(x) = \frac{\mathrm{d}y(x)}{\mathrm{d}x}$ to get a system of two first-order ODEs

$$y'(x) = z(x), \qquad\qquad z'(x) = q(x) - r(x)\,z(x). \tag{1.8}$$

As mentioned above, the system (1.5) needs N additional conditions to be fully determined. If all these conditions are given at the same point (e.g., the value of y, its derivatives or any combination of these), we have an *initial value problem*, otherwise we have a *boundary value problem*. Usually in 1D the initial value problem has all conditions specified on the left end of the computational domain, while the conditions for the boundary value problem are imposed on both ends of the domain.

1.4 Methods for Initial Value Problems

1.4.1 Runge-Kutta Methods

A very popular class of methods for the solution of an initial value problem (1.2) (or its vector analog for a system) are the explicit Runge-Kutta methods. To start with, we split the computational domain $[a, b]$ into a set of subintervals

$$a = x_0 < x_1 < \cdots < x_{N-1} < x_N = b \tag{1.9}$$

and then proceed from point x_0 towards x_N, trying to reconstruct the function y. Runge-Kutta methods are one-step methods, meaning that in the $i+1$-st step, to approximate y at point x_{n+1}, we only use directly the information from point x_n (and possibly some virtual points between x_n and x_{n+1}), but not from earlier points. (However, that information is already embedded in the solution from earlier steps of the calculation).

The most obvious approach is to use the Taylor expansion. Denoting the step size (subinterval length) $h = x_{n+1} - x_n$, we have

$$\begin{aligned} y\left(x_{n+1}\right) = y\left(x_n + h\right) &= y(x_n) + hy'\left(x_n\right)\frac{h^2}{2}y''\left(x_n\right) + \cdots = \\ &= y(x_n) + hf\left(x_n, y(x_n)\right) + \frac{h^2}{2}\left.\frac{\mathrm{d}f\left(x,y\right)}{\mathrm{d}x}\right|_{(x_n, y(x_n))} + \ldots, \end{aligned} \tag{1.10}$$

which suggests to approximate the advance of function y in this subinterval by the step

$$y_{n+1} = y_n + h\,f(x_n, y_n). \tag{1.11}$$

This amounts to approximate the function on each subinterval using the value and slope at its beginning, as shown in Fig. 2(a,b). It can be easily proven (and in fact it is immediately visible

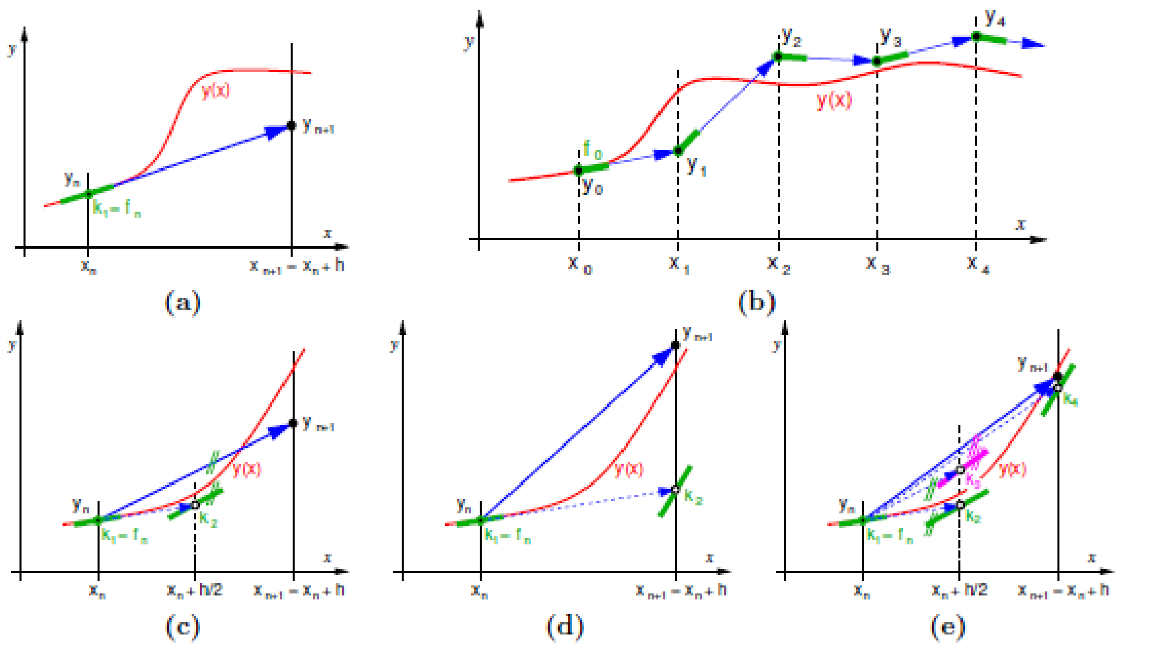


Figure 2: Selected explicit Runge-Kutta methods. (a,b) Euler's method, (c) RK2 midpoint, (d) RK2 Heun/trapezoidal, (d) RK4.

from the figure), that this approach, called *Euler's method*, is only first order accurate, meaning that the error at the final point y_N (i.e., after n steps) is proportional to h.

While it is not very useful to try to employ the second derivative from (1.10), since we would need to approximate the unknown derivatives of f, we can develop more accurate Runge-Kutta methods by improving the approximation in the subinterval, using a combination of slopes f evaluated at different intermediate points $x_n \le x \le x_{n+1}$ and corresponding predicted values of y. For example, we can use the mid-interval value $y(x_n + h/2)$ predicted by Euler's method, evaluate the right-hand side f there, and then use this slope to actually advance from x_n to x_{n+1}, which amounts to replacing the forward finite difference in integration by the central difference. This is the *RK2 midpoint* method shown in Fig. 2(c). Or we can use Euler's method to predict the value $y(x_n + h)$ at the subinterval's endpoint, calculate the slope f there, and then use the average of the slopes at both ends of the subinterval to advance from x_n to x_{n+1}. This is the *RK2 Heun's* method, shown in Fig. 2(d). Both latter approaches are two-stage Runge-Kutta methods of second-order accuracy.

In general, one step of an explicit r-stage Runge-Kutta method can be written as

$$y_{n+1} = y_n + h \sum_{j=1}^{r} b_j \, k_j \tag{1.12}$$

$$\begin{array}{c|c} & \\ \hline & 1 \end{array}$$

(a) Euler

$$\begin{array}{c|cc} 1/2 & 1/2 & \\ \hline & 0 & 1 \end{array}$$

(b) RK2 midpoint

$$\begin{array}{c|cc} 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

(c) RK2 Heun

$$\begin{array}{c|cccc} 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

(d) classical RK4

Figure 3: Butcher tableaux for some popular Runge-Kutta methods.

with

$$\begin{aligned}
k_1 &= f(x_n, y_n), \\
k_2 &= f(x_n + c_2\, h\ ,\ y_n + a_{21}\, k_1\, h), \\
k_3 &= f(x_n + c_3\, h\ ,\ y_n + h(a_{31}\, k_1 + a_{32}\, k_2)), \\
&\dots \\
k_r &= f(x_n + c_r\, h\ ,\ y_n + h \sum\nolimits_{j=1}^{r-1} a_{rj}\, k_j),
\end{aligned}$$

where the constants c_r, a_{rj} and b_j, specifying the particular method, are typically found in literature in the compact form of the Butcher tableau

$$\begin{array}{c|cccccc}
c_2 & a_{21} & & & & & \\
c_3 & a_{31} & a_{32} & & & & \\
\vdots & \vdots & \vdots & & & & \\
c_r & a_{r1} & a_{r2} & a_{r3} & \dots & a_{r,r-1} & \\
\hline
 & b_1 & b_2 & b_3 & \dots & b_{r-1} & b_r
\end{array} .$$

Fig. 3 shows Butcher tableaux for the three already discussed methods and also for the popular fourth-order *RK4* method.

In practical calculations, adaptive step h is used to save computational time, reduce the roundoff error but still capture important features in the solution. That is, in each step the length of the subinterval is reduced or extended, depending on the local smoothness of the solution, evaluated by some error estimator. The particular implementation of the step length control is beyond the scope of this text, the interested reader is referred e.g. to [1].

1.4.2 Multistep and Implicit Methods

As powerful and elegant as the explicit Runge-Kutta methods are in many applications, the maximum length of step h to be used is limited by their region of stability, which is too restrictive in some situations. A typical example where even the adaptive step length control does not help, are the so-called *stiff problems* (or problems with strong damping). Consider for example the initial value problem

$$y'' + 1001\, y' + 1000\, y = 0, \qquad y(0) = 2, \qquad y'(0) = -1001, \tag{1.13}$$

which has the exact solution

$$y(x) = \exp(-1000\, x) + \exp(-x). \tag{1.14}$$

Even though with increasing x the first part of the solution is very soon rendered negligible, its short length scale still forces any explicit method to use a very short time step to stay stable. For a detailed analysis and discussion of stiff problems, refer to [2], [3].

Assume a general method for solving an ODE written in the form

$$y_{n+1} = \sum_{j=1}^{k} a_j\, y_{n+1-j} + h \sum_{j=0}^{k} b_j\, f_{n+1-j}. \tag{1.15}$$

Methods with $b_0 = 0$, that is, methods with the right-hand side of (1.15) not dependent on information at point x_{n+1}, are called *explicit*, while the methods containing $f(x_n, y_n)$ are *implicit*. Methods with the only non-zero coefficients in (1.15) being a_1 and b_1 (and possibly also b_0), that is, methods with the step depending only on values and derivatives at the previous point, are *one-step* methods, while the advance step of *multistep* methods directly uses also values from the previous points x_{n-1}, x_{n-2}, etc. Thus, the Runge-Kutta methods presented so far are one-step explicit methods.

While in the explicit methods each step can be directly written and easily evaluated, relatively short steps h have to be used to maintain stability. On the other hand, with implicit methods, evaluation of the step (1.15) amounts to solving generally nonlinear equations, but they are often stable for much longer steps h, sometimes even unconditionally stable. Even though for sufficient accuracy the step still needs to be reasonably short, implicit methods are powerful tools to overcome difficult situations such as when solving the stiff problem (1.13). Somewhere in between lie the semi-implicit methods, which can be derived by linearizing the implicit methods (neglecting higher-order terms in the Taylor expansion), so that the solution of nonlinear problems is reduced to solving linear systems.

A very popular combination of explicit and implicit methods is the *predictor-corrector* approach, in which the value $\tilde{y}_{n+1}$ at point x_{n+1} is first predicted by an explicit method, then used to evaluate the right-hand side $\tilde{f}_{n+1} = f(x_{n+1}, \tilde{y}_{n+1})$ of the ODE, and this estimated slope is then utilized on the right-hand side of (1.15) in an implicit method. For example, we can combine the explicit Euler's method with a simple implicit method as

- Pred: $$\tilde{y}_{n+1} = y_n + h\, f(x_n, y_n) \tag{1.16a}$$
- Corr: $$y_{n+1} = y_n + \frac{h}{2}\Big(f(x_n, y_n) + f(x_{n+1}, \tilde{y}_{n+1})\Big) \tag{1.16b}$$

or, using higher-order Adams-Bashforth and Adams-Moulton methods as components,

- Pred: $$\tilde{y}_{n+1} = y_n + \frac{h}{12}(23\, f_n - 16 f_{n-1} + 5\, f_{n-2}) \tag{1.17a}$$
- Corr: $$y_{n+1} = y_n + \frac{h}{12}(5\, \tilde{f}_{n+1} + 8\, f_n - f_{n-1}). \tag{1.17b}$$

It can be useful to repeat the corrector step a few times to improve the result. Note that for a fixed number of iterations, the resulting predictor-corrector method as a whole is basically an explicit formula, so we lose the advantage of improved stability gained by employing the implicit corrector. (It is easy to insert (1.16a) into (1.16b) and verify that using one predictor and one corrector, this method is equivalent to Heun's RK2 method discussed in Section 1.4.1!) But in general, predictor-corrector methods with properly controlled iterations of the corrector are proving to be very powerful on difficult problems.

1.5 Methods for Boundary Value Problems

Finally, let us mention some methods for solving the boundary value problem, that is, an ODE with additional conditions given at different points. Without loss of generality, we will consider a 1D problem with one condition imposed at each of the two endpoints of the domain.

1.5.1 The Shooting Method

The shooting method is based on repeated solution of an initial value problem. Starting on one end of the domain, we use all conditions defined there, and to set up a complete initial condition, we fill the missing values by an estimate. Then we solve the initial value problem with this set of conditions and compare the outcome at the other end of the domain with the desired values being the real boundary conditions prescribed there originally. Based on this comparison we modify the initial condition (estimates for missing values) and retry. The problem is solved once sufficient correspondence at the endpoint is reached.

As an example, consider the four ODEs describing shooting in air (hence the name of the method)

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v\,\cos\theta, \qquad \frac{\mathrm{d}v}{\mathrm{d}t} = -\frac{1}{2m}\,c\,\rho\,s\,v^2 - g\,\sin\theta, \tag{1.18a}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = v\,\sin\theta, \qquad \frac{\mathrm{d}\theta}{\mathrm{d}t} = -\frac{g}{v}\,\cos\theta \tag{1.18b}$$

(with c, ρ, s, g being constants) and an artillerist, located at the coordinate origin, trying to hit the target located at point $(x_c, 0)$, that is, with the boundary conditions

$$x(0) = y(0) = 0, \qquad v(0) = v_0, \qquad y(x_c) = 0. \tag{1.19}$$

Assuming the initial velocity v_0 is constant for a given cannon, the artillerist has to choose the correct initial angle $\theta(0)$ w.r.t. ground in order to hit the target. If the shot (solution of the initial value problem) misses the target, the angle is adapted, shot retried, etc. This amounts to solving one nonlinear equation $y(x_c, \theta_0) = 0$ for the unknown $\theta_0 = \theta(0)$. Inside this "wrapper" problem, each particular shot (evaluation of y for given θ_0) consists in solving an initial value problem for (1.18)-(1.19), e.g. by some Runge-Kutta method discussed in the previous section.

The shooting method is a very popular and powerful tool in many fields of physics.

1.5.2 Finite Difference Method

Another approach to solve the boundary value problem consists in discretization of the differential equation to replace the continuous problem by a set of finite difference equations.

First, let us briefly recall what finite differences are. The simplest way to approximate the first derivative of a function on the discrete grid (1.9) is using the forward difference

$$y'(x_i) \approx \tilde{y}'_i = \frac{y_{i+1} - y_i}{\Delta x}, \tag{1.20}$$

with $\Delta x = x_{i+1} - x_i$. This is actually very close to the standard definition of a derivative, except that we are using a finite step here. Using the Taylor expansion

$$\begin{aligned} \tilde{y}'_i = \frac{y_{i+1} - y_i}{\Delta x} &= \frac{y(x_i + \Delta x) - y(x_i)}{\Delta x} \\ &= \frac{y(x_i) + y'(x_i)\Delta x + y''(x_i)\frac{\Delta x^2}{2} + \cdots - y(x_i)}{\Delta x} \\ &= y'(x_i) + \frac{\Delta x}{2}\,y''(x_i) + \ldots \end{aligned}$$

we see, that this approximation of $y'(x_i)$ is of first order of accuracy. On an equidistant grid ($\forall\, i,\ x_{i+1} - x_i = \Delta x$), the central difference

$$y'(x_i) \approx \tilde{y}'_i = \frac{y_{i+1} - y_{i-1}}{2\,\Delta x} \tag{1.21}$$

is second-order accurate at x_i:

$$\begin{aligned}\tilde{y}'_i &= \frac{y(x_i+\Delta x)-y(x_i-\Delta x)}{2\Delta x}\\ &= \frac{\left(y(x_i)+y'(x_i)\Delta x+y''(x_i)\frac{\Delta x^2}{2!}+y'''(x_i)\frac{\Delta x^3}{3!}+\ldots\right)-\left(y(x_i)-y'(x_i)\Delta x+y''(x_i)\frac{\Delta x^2}{2!}-y'''(x_i)\frac{\Delta x^3}{3!}+\ldots\right)}{2\Delta x}\\ &= y'(x_i)+\frac{\Delta x^2}{6}\,y'''(x_i)+\ldots\end{aligned}$$

(From here, one can also see, that the forward difference (1.20) is first-order accurate w.r.t. point x_i, but second-order accurate w.r.t. $x_{i+\frac{1}{2}}$.)

Using recursion, we can easily find the finite difference approximating the second derivative

$$y''(x_i)\approx\tilde{y}''_i=\frac{\frac{y_{i+1}-y_i}{\Delta x}-\frac{y_i-y_{i-1}}{\Delta x}}{\Delta x}=\frac{y_{i+1}-2y_i+y_{i-1}}{\Delta x^2} \qquad (1.22)$$

with second-order accuracy

$$\begin{aligned}\tilde{y}''_i &= \frac{y''(x_i)\frac{\Delta x^2}{2!}+y^{(4)}(x_i)\frac{\Delta x^4}{4!}+\cdots+y''(x_i)\frac{\Delta x^2}{2!}+y^{(4)}(x_i)\frac{\Delta x^4}{4!}+\ldots}{\Delta x^2}\\ &= y''(x_i)+y^{(4)}(x_i)\frac{\Delta x^2}{12}+\ldots\end{aligned}$$

This way, a finite difference approximating any derivative on a selected stencil can be derived. Another approach is to choose the stencil (set of points whose values we want to use), apply the Taylor expansion w.r.t. the master node, and then solve a simple linear system with the desired derivative on the right-hand side. See the presentation corresponding to this text for details.

To demonstrate how to apply the *Finite difference method*, let us consider the second-order ODE

$$a(x)\,v''(x)+b(x)\,v'(x)+c(x)\,v(x)=d(x), \qquad x\in\langle 0,1\rangle, \qquad v(0)=\alpha, \quad v(1)=\beta \qquad (1.23)$$

with a, b, c and d being known functions of x. At each interior point x_i, $i\in\{1,\ldots,N-1\}$ of the mesh, we approximate the ODE by using the central first and second derivatives (1.21), (1.22) to obtain the finite difference equation

$$a_i\,\frac{v_{i+1}-2\,v_i+v_{i-1}}{h^2}+b_i\,\frac{v_{i+1}-v_{i-1}}{2\,h}+c_i\,v_i=d_i.$$

Reordering for each i to the form

$$\underbrace{\left(-a_i+b_i\,\frac{h}{2}\right)}_{r_i}v_{i-1}+\underbrace{\left(2\,a_i-c_i\,h^2\right)}_{p_i}v_i+\underbrace{\left(-a_i-b_i\,\frac{h}{2}\right)}_{q_i}v_{i+1}=-h^2\,d_i, \qquad (1.24)$$

using auxiliary coefficients p_i, q_i and r_i as above and applying the boundary condition for v_0 and v_N, we have a system of linear equations for v_i

$$\begin{pmatrix} p_1 & q_1 & 0 & \ldots & \ldots & \ldots & 0\\ r_2 & p_2 & q_2 & 0 & \ldots & \ldots & 0\\ 0 & \ddots & \ddots & \ddots & & & 0\\ \vdots & & \ddots & \ddots & \ddots & & \vdots\\ \vdots & & & \ddots & \ddots & \ddots & 0\\ 0 & \ldots & \ldots & 0 & r_{N-2} & p_{N-2} & q_{N-2}\\ 0 & \ldots & \ldots & \ldots & 0 & r_{N-1} & p_{N-1}\end{pmatrix}\begin{pmatrix} v_1\\ v_2\\ v_3\\ \vdots\\ v_{N-3}\\ v_{N-2}\\ v_{N-1}\end{pmatrix}=-h^2\begin{pmatrix} d_1+r_1\,\frac{\alpha}{h^2}\\ d_2\\ d_3\\ \vdots\\ d_{N-3}\\ d_{N-2}\\ d_{N-2}+q_{N-1}\,\frac{\beta}{h^2},\end{pmatrix},$$

which is very easy to solve since it has a tridiagonal matrix.

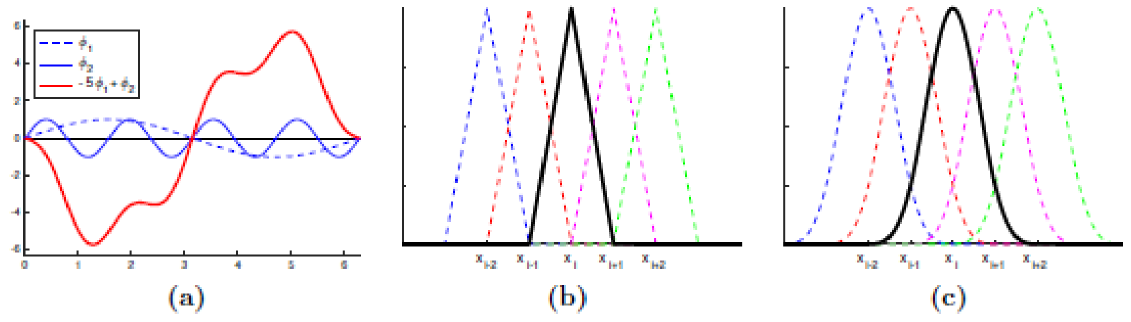


Figure 4: Variational methods. (a) combination of two basis sine functions, (b,c) linear elements and bell splines for the Finite element method.

1.5.3 Variational Methods

In *variational methods*, the boundary value problem is transformed into a variational problem, where the maximum of some functional is being minimized. Instead of looking for the solution on a set of points, we are searching for the solution in a given class of functions. That is, the solution should satisfy

$$y(x) = \sum_{k=1}^{\infty} a_k\, \varphi_k(x), \tag{1.25}$$

where $\{\varphi_k(x)\}$ is a complete system of functions in given Hilbert space. In practice, a finite set of functions is used. See Fig. 4(a) for an illustration of how the basis functions can combine.

The ODE is rewritten using differential operators A, $L_m^{(a)}$, $L_m^{(b)}$ to the form

$$A\, y(x) = f(x), \qquad\qquad x \in [a, b]$$

with homogeneous boundary conditions

$$L_m^{(a)} y = 0, \qquad L_m^{(b)} y = 0, \qquad m = 1, 2, \ldots, k,$$

and a scalar product $(\cdot,\cdot)$ is defined on the Hilbert space.

In Galerkin methods, we are looking for coefficients a_k of the solution

$$y(x) = \sum_{k=1}^{N} a_k\, \varphi_k(x) \tag{1.26}$$

such, that

$$\forall j \in \{1, \ldots, N\}, \qquad (A\, y - f, \varphi_j) = 0, \tag{1.27}$$

which boils down to solving a system of equations.

Another variational approach is the *Finite element method*, which uses special basis functions that are nonzero only on some short subinterval, e.g. linear elements

$$\varphi_i(x) = \begin{cases} 0 & \text{for} \quad x \le x_{i-1} \\ \frac{x - x_{i-1}}{x_i - x_{i-1}} & \text{for} \quad x_{i-1} \le x \le x_i \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & \text{for} \quad x_i \le x \le x_{i+1} \\ 0 & \text{for} \quad x_{i+1} \le x \end{cases} \tag{1.28}$$

shown in Fig. 4(b), or, for ODEs with higher than second derivatives, smoother elements such as bell splines

$$\varphi_i(x) = \begin{cases} 0 & \text{for } q \le 0 \\ q^3 & \text{for } p \le 0 \\ q^3 - 4\,p^3 & \text{for } p \ge 0 \end{cases}, \qquad \text{where} \qquad \begin{aligned} h &= x_{i+1} - x_i,\ \forall i, \\ p &= h - |x - x_i|, \\ q &= 2h - |x - x_i|, \end{aligned} \tag{1.29}$$

see Fig. 4(c).

A closer description of methods in this exciting field of study is unfortunately beyond the scope of this introductory text. However, some of the techniques mentioned here will be applied and detailed in the following chapters.

References

[1] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, New York, NY, USA, 3rd edition, 2007.

[2] E. Hairer, S. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems.* Springer, 2nd edition, 2008.

[3] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems.* Springer, 2nd edition, 2002.

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Chapter 2: Solving partial differential equations (PDEs)

R. Liska

2. Solving partial differential equations (PDEs)

2.1 Overview and types of PDEs

Overview

- finite difference method
- covergence, consistency, well-posedness, stability
- Courant-Friedrichs-Lewy condition, stability by Fourier method, modified equation, numerical diffusion and dispersion
- finite difference schemes for
  - advection equation – hyperbolic
  - heat equation – parabolic
  - advection diffusion equation
- conservation laws
  - integral and differential form
  - difference schemes for conservation laws
  - conservativity
  - Rankine-Hugoniot condition
  - Burgers equation
  - shallow water equation

PDEs

- independent variables t, x, y
- unknown functions $u(t, x), u(x, y)$
- derivatives $u_t = \frac{\partial u}{\partial t}, u_{xx} = \frac{\partial^2 u}{\partial x^2}$
- stationary equations – eliptic equations, e.g. Laplace equation $u_{xx}+u_{yy} = 0$ or Poisson equation $u_{xx}+u_{yy} = f(x, y)$
- evolutionary equations
  - hyperbolic, e.g. advection equation $u_t + au_x = 0$
  - parabolic, e.g. heat equation $u_t + bu_{xx} = 0$
  - combined, e.g. advection-diffusion equation $u_t + au_x + bu_{xx} = 0$
- source terms $s(t, x, u)$ on the right hand side
- conditions
  - initial $u(0, x) = u_0(x)$
  - boundary $u(t, 0) = f_0(t), u(t, 1) = f_1(t)$
- well-posedness of PDEs

Hyperbolic PDEs

- Cauchy initial problem: find $u(t,x)$

$$u_t + au_x = 0, u(0,x) = u_0(x), t \geq 0, x \in R$$

advection equation, one-way wave equation

- solution $u(t,x) = u_0(x - at)$
- with source

$$u_t + au_x = bu, u(0,x) = u_0(x), t \geq 0, x \in R$$

- solution

$$u(t,x) \ = \ u_0(x - at)e^{bt}$$

- usually practical PDEs have no analytical solution, except in special cases
- numerical solution needed

Type of boundary conditions BCs

- Dirichlet BC, $u(t,0) = f_0(t)$
- Neumann BC, $u_x(t,0) = f_n(t)$
  natural BC $u_x(t,0) = 0$
- Robin BC, $\alpha u(t,0) + \beta u_x(t,0) = \gamma, \quad \alpha(t), \beta(t), \gamma(t)$
- periodic $x \in (0,1), u(t,0) = u(t,1)$ compatibility of BC $u_0(0) = u_0(1)$
- reflecting BC
- absorbing BC

Computational mesh

- instead of continuum variables $(t,x) \in [0,\infty) \times R$ we introduce discrete variables, or a discrete computational mesh

$$(t_n, x_j) = (n\Delta t, j\Delta x), n \in N_0, j \in Z,$$

where Δt and Δx are time and space step of the computational mesh

- values of function $u(t,x)$ are repllaced by approximate discrete values $u(t_n, x_j) \approx u_j^n$

2.2 Finite difference method

- replacing derivatives by finite differences
- forward difference

$$u_x \to \frac{u_{j+1} - u_j}{\Delta x}$$

- backward difference

$$u_x \to \frac{u_j - u_{j-1}}{\Delta x}$$

- central difference

$$u_x \to \frac{u_{j+1} - u_{j-1}}{2\Delta x}$$

Naive approach

- advection equation $u_t + au_x = 0$
- forward difference in time, central difference in space

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = 0$$

gives central scheme, which is unstable

Convergence, consistency, well-posedness, stability

- **Definition:** A one-step finite difference scheme aproximating a PDE is **convergent** $\Leftrightarrow$
  $\forall$ solution $u(x,t)$ of the PDE, $\forall u_j^n$ solution of the difference scheme, such that $u_j^0 \to u_o(x)$ when $j\Delta x \to x, \Delta x \to 0$, then
  $u_j^n \to u(t,x)$ for $(n\Delta t, j\Delta x) \to (t,x), \Delta x \to 0, \Delta t \to 0$
- **Definition:** Difference scheme $P_{\Delta t, \Delta x} u_j^n = f_j^n$ is **consistent** with the PDE $Pu = f \Leftrightarrow$
  $\forall$ smooth function $\Phi(t,x)$ holds $P\Phi - P_{\Delta t,\Delta x}\Phi_j^n \to 0$ for $\Delta t \to 0, \Delta x \to 0$
  (point convergence at each mesh point)
- consistency check – Taylor expansion in Δx and Δt, then $\Delta t \to 0, \Delta x \to 0$ give the PDE
- e.g. the central scheme is consistent

- **Definition:** Difference scheme $P_{\Delta t,\Delta x} u_j^n = 0$ for a first order PDE is **stable** in the stability domain $S \Leftrightarrow$

$$\exists J \in N_0, J \ll n, \forall T > 0, \exists c_T \in R$$

$$\sum_{j=-\infty}^{\infty} |u_j^n|^2 \leq c_T \sum_{m=0}^{J} \sum_{j=-\infty}^{\infty} |u_j^m|^2$$

  for $0 \leq n\Delta t \leq T$ a $(\Delta t, \Delta x) \in S$
- **Definition:** The initial problem for a first order PDE $Pu = 0$ is **well-posed** $\Leftrightarrow$

$$\forall T > 0, \exists c_T, \forall u(t,x), Pu(t,x) = 0,$$

$$\int_{-\infty}^{\infty} |u(t,x)|^2 dx \leq c_T \int_{-\infty}^{\infty} |u(0,x)|^2 dx$$

  for $0 \leq t \leq T$
- **Lax-Richtmyer theorem:** A difference scheme consistent with a well-posed PDE is convergent $\Leftrightarrow$ the scheme is stable.

2.3 Basic difference schemes for advection equation

- forward scheme – forward difference in both time and space

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+a\frac{u_{j+1}^n-u_j^n}{\Delta x}=0,$$

we know all values at time level n, we express the only unknown u_j^{n+1}

$$u_j^{n+1}=u_j^n-a\Delta t\frac{u_{j+1}^n-u_j^n}{\Delta x}$$

- during computation the time step is computed as

$$\Delta t=C\frac{\Delta x}{|a|},$$

where $C>0$ is the CFL (Courant-Friedrichs-Levy) number

- forward scheme is first order accurate $O(\Delta x)$ and stable for $a<0$ and $C\le 1$
- backward scheme – forward difference in time, backward difference in space

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+a\frac{u_j^n-u_{j-1}^n}{\Delta x}=0$$

is first order accurate $O(\Delta x)$ and stable for $a>0$ and $C\le 1$

- Lax-Friedrichs scheme

$$\frac{u_j^{n+1}-(u_{j+1}^n+u_{j-1}^n)/2}{\Delta t}+a\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}=0$$

is first order accurate $O(\Delta x)$ and stable for $C\le 1$

- Lax-Wendroff scheme

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+a\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}=\frac{a^2\Delta t}{2}\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{\Delta x^2}$$

is second order accurate $O(\Delta x^2)$ and stable for $C\le 1$

2.4 Courant-Friedrichs-Lewy condition, stability analysis by Fourier method

- **Theorem:** For the explicit difference scheme

  $$u_j^{n+1} = \alpha u_{j-1}^n + \beta u_j^n + \gamma u_{j+1}^n$$

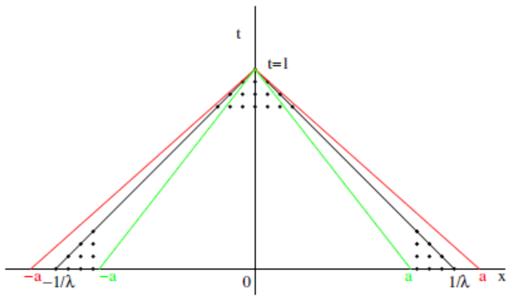
  approximating the advection equation $u_t + au_x = 0$ with $\lambda = \Delta t/\Delta x$ constant the CFL condition $|a|\lambda \leq 1$ is necessary for stability.
- for hyperbolic system $U_t + AU_x = 0$ the CFL condition is $|a_i|\lambda \leq 1, \quad \forall a_i, a_i$ is eigenvalue of the matrix A
- sufficient condition can be different
- limiting the time step $\Delta t \leq \Delta x/|a|$
- **Theorem:** There is no explicit, unconditionally stable, consistent scheme for a system of hyperbolic PDE.
- **Note:** the implicit scheme

  $$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x} = 0$$

  is unconditionally stable

Explanation of the CFL condition

- we take the point $(t, x) = (1, 0) = (n\Delta t, 0)$ and find all points of the mesh on which u_0^n depends; for $t = 1$ these are the points for which $|x| \leq 1/\lambda$; we know $n = 1/\Delta t$, so that $n\Delta x = \Delta x/\Delta t = 1/\lambda$



- characteristics with $|a| < 1/\lambda$ satisfy the CFL condition and with $|a| > 1/\lambda$ do not satisty the CFL condition

Stability analysis by Fourier method

- forward scheme for advection equation $u_t + au_x = 0$

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\frac{u_{j+1}^n - u_j^n}{\Delta x} = 0$$

$$u_j^{n+1} = (1 + a\lambda)u_j^n - a\lambda u_{j+1}^n, \quad \lambda = \frac{\Delta t}{\Delta x}$$

- Fourier transformation $u_j^n \longrightarrow g^n e^{ij\xi\Delta x}\hat{u}^n$, with $\hat{u}$ denoting the discrete Fourier transformation of u_j, gives

$$\hat{u}^{n+1} = [(1 + a\lambda) - a\lambda e^{i\xi\Delta x}]\hat{u}^n(\xi) = g(\xi\Delta x)\hat{u}^n(\xi)$$

- amplification faktor

$$\hat{u}^{n+1} = g(\Theta)\hat{u}^n(\xi), \qquad \Theta = \xi\Delta x$$

for the forward scheme is

$$g(\Theta) = (1 + a\lambda) - a\lambda e^{-i\Theta}$$

- from the initial condition

$$\hat{u}^n = g(\Theta)^n\hat{u}^0(\xi)$$

- using Parseval identity

$$\begin{aligned} ||u^n||_2^2 &= ||\hat{u}^n||_2^2 = ||g(\Theta)^n\hat{u}^0(\xi)||_2^2 \\ &= \int_{-\pi}^{\pi} |g(\Theta)|^{2n}|\hat{u}^0(\xi)|^2 d\Theta \leq c_T||\hat{u}^0(\Theta)||_2^2 \end{aligned}$$

- for a stable scheme $||u^n||_2^2$ has to be bounded, so

$$|g(\Theta)|^2 \leq 1$$

- **Von Neumann theorem:** One step difference scheme, with constant coefficients and with the amplification factor $g(\Theta, \Delta t, \Delta x)$, which does not depend explicitely on Δt and Δx (it can depend on $\lambda = \Delta t/\Delta x$), is stable $\Leftrightarrow$ $\forall\Theta, |g(\Theta)| \leq 1$

- amplification faktor for the forward scheme is

$$g(\Theta) = (1 + a\lambda) - a\lambda e^{-i\Theta}$$

so that

$$|g(\Theta)|^2 - 1 = -2a\lambda(a\lambda + 1)(\cos\Theta - 1)$$

and

$$|g(\Theta)|^2 - 1 \leq 0 \Leftrightarrow a(a\lambda + 1) \leq 0$$

- the forward scheme is stable for $a < 0, a\lambda \geq -1$, i.e. $a < 0, |a|\lambda \leq 1$ or $a < 0, C \leq 1$

2.5 Modified equation of a difference scheme

- Lax-Friedrichs scheme for $u_t + au_x = 0$

$$\frac{u_j^{n+1} - (u_{j+1}^n + u_{j-1}^n)/2}{\Delta t} + a\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = 0$$

- Taylor expansion in t, x

$$u_t + au_x + \frac{1}{2}\left(\Delta t u_{tt} - \frac{\Delta x^2}{\Delta t}u_{xx}\right) = O(\Delta t^2, \Delta x^2)$$

express

$$u_t = -au_x + O(\Delta t, \Delta x)$$

by derivation we get

$$\begin{aligned} u_{tt} &= -au_{tx} + O(\Delta t, \Delta x) \\ u_{tx} &= -au_{xx} + O(\Delta t, \Delta x) \\ u_{tt} &= a^2 u_{xx} + O(\Delta t, \Delta x) \end{aligned}$$

- from Taylor expansion we eliminate all time derivatives except u_t to get the modified equation (ME)

$$u_t + au_x = \frac{\Delta x^2}{2\Delta t}\left(1 - \frac{\Delta t^2}{\Delta x^2}a^2\right) u_{xx} + O(\Delta t^2, \Delta x^2)$$

$$u_t + au_x = \frac{\Delta x^2}{2\Delta t}\left(1 - a^2\lambda^2\right) u_{xx} + O(\Delta t^2, \Delta x^2)$$

- LF scheme is not consistent for $\Delta t = c\Delta x^2$
- from modified eqaution one can check
  - consistency
  - order of approximation
  - stability – heuristic condition
  - numerical diffusion
  - numerical dispersion
- ME for forward scheme is

$$u_t + au_x = -\frac{1}{2}a\Delta x(a\lambda + 1)u_{xx}$$

- scheme is consistent, first order $O(\Delta x)$, stable for $a(a\lambda + 1) \leq 0$, i.e. $a < 0, C \leq 1$, diffusive

- ME for Lax-Wendroff scheme is

$$\begin{aligned} u_t + au_x = &\frac{1}{6}a\Delta x^2 u_{xxx}(a^2\lambda^2 - 1) \\ &+\frac{1}{8}a^2\lambda\Delta x^3 u_{xxxx}(a^2\lambda^2 - 1) \end{aligned}$$

- scheme is consistent, second order $O(\Delta x^2)$, stable for $a^2\lambda^2 - 1 \leq 0$, i.e. $C \leq 1$, dispersive

2.6 Implicit Scheme-Summary for hyperbolic/advection equation schemes

- advection equation $u_t + au_x = 0$
- implicit scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x} = 0,$$

  is consistent, first order $O(\Delta x)$, always stable

- free boundary conditions $u_x = 0$

$$u_2^{n+1} - u_1^{n+1} = 0, \quad u_J^{n+1} - u_{J-1}^{n+1} = 0$$

- rewrite as a system

$$\begin{aligned} u_1^{n+1} - u_2^{n+1} &= 0 \\ -u_{j-1}^{n+1}a\frac{\Delta t}{2\Delta x} + u_j^{n+1} + u_{j+1}^{n+1}a\frac{\Delta t}{2\Delta x} &= u_j^n, \quad j = 2, \cdots, J-1 \\ -u_{J-1}^{n+1} + u_J^{n+1} &= 0 \end{aligned}$$

  in matrix form

$$\begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 & 0 \\ -\frac{a\Delta t}{2\Delta x} & 1 & \frac{a\Delta t}{2\Delta x} & 0 & \cdots & 0 & 0 \\ 0 & -\frac{a\Delta t}{2\Delta x} & 1 & \frac{a\Delta t}{2\Delta x} & \cdots & 0 & 0 \\ & & & \cdots & & & \\ 0 & 0 & 0 & \cdots & -\frac{a\Delta t}{2\Delta x} & 1 & \frac{a\Delta t}{2\Delta x} \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{J-1}^{n+1} \\ u_J^{n+1} \end{pmatrix} = \begin{pmatrix} 0 \\ u_2^n \\ u_3^n \\ \vdots \\ u_{J-1}^n \\ 0 \end{pmatrix}$$

$$\begin{aligned} M \cdot U^{n+1} &= UU^n \\ U^{n+1} &= M^{-1} \cdot UU^n \end{aligned}$$

Summary of schemes

- hyperbolic equations – finite speed of signal propagation
- explicit schemes with time step

$$\Delta t = C\frac{\Delta x}{|a|},$$

  where $C > 0$ – CFL (Courant-Friedrichs-Levy) number

- forward scheme – $O(\Delta x)$, stable for $a < 0$ and $C \leq 1$, diffusive
- backward scheme – $O(\Delta x)$, stable for $a > 0$ and $C \leq 1$, diffusive
- upwind method – combination of forward/backward schemes according to the sign of a
- Lax-Friedrichs scheme – $O(\Delta x)$, stable for $C \leq 1$, diffusive
- Lax-Wendroff scheme – $O(\Delta x^2)$, stable for $C \leq 1$, dispersive

2.7 Parabolic equations

- heat equation $u_t = b u_{xx}$ with initial condition $u(0,x) = u_0(x)$; the initial problem is well posed for $b > 0$
- solution (using Fourier transformation in x)

$$u(t,x) = \frac{1}{2\sqrt{\pi b t}} \int_{-\infty}^{\infty} e^{-(x-y)^2/(4bt)} u_0(y) d\, y$$

- weighted average of u_0
- for small t the weight is a narrow peak around $y = x$; for greater t the weight peak is wider
- $u \in C^{\infty}_{t,x}$ – infinitely differentiable
- positivity of solution – $u_0(x) \geq 0 \wedge \exists y, u_0(y) \neq 0 \implies u(t,x) > 0, t > 0$
- for initial condition

$$u_0(x) = \begin{cases} 1 & \text{pro} \quad x < 0 \\ 0 & \text{pro} \quad x \geq 0 \end{cases}$$

the solution at time $t = \frac{1}{4b}$ is

$$\begin{aligned} u(1/(4b), x) &= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{0} e^{-(x-y)^2} d\, y \\ &= \frac{1}{\sqrt{\pi}} \int_{x}^{\infty} e^{-z^2} d\, z = \frac{1 - erf(x)}{2} \end{aligned}$$

for arbitraty big x is $u(1/(4b), x) > 0$

Explicit scheme for heat equation

- heat equation $u_t = b u_{xx}$
- explicit scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = b \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}$$

is consistent, stable for

$$\Delta t \leq \frac{\Delta x^2}{2b}$$

and second order $O(\Delta x^2)$ accurate when stable

- too severe limitation for $\Delta t \approx \Delta x^2$ – too many time steps needed to reach the final time

Implicit scheme for heat equation

- heat equation $u_t = bu_{xx}$
- implicit scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = b\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2}$$

is consistent, always stable, for $\Delta t \approx \Delta x$ it is $O(\Delta x)$, for $\Delta t \approx \Delta x^2$ it is $O(\Delta x^2)$

- system with natural BCs $u_x = 0$

$$\begin{aligned} u_1^{n+1} - u_2^{n+1} &= 0 \\ -u_{j-1}^{n+1}b\mu + u_j^{n+1}(1+2b\mu) - u_{j+1}^{n+1}b\mu &= u_j^n, \quad j = 2, \cdots, J-1 \\ -u_{J-1}^{n+1} + u_J^{n+1} &= 0 \end{aligned}$$

where $\mu = \frac{\Delta t}{\Delta x^2}$

- in matrix form

$$\begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 & 0 \\ -b\mu & 1+2b\mu & -b\mu & 0 & \cdots & 0 & 0 \\ 0 & -b\mu & 1+2b\mu & -b\mu & 0 & 0 & \\ & & & \cdots & & & \\ 0 & 0 & 0 & \cdots & -b\mu & 1+2b\mu & -b\mu \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{J-1}^{n+1} \\ u_J^{n+1} \end{pmatrix} =$$

$$\begin{pmatrix} 0 \\ u_2^n \\ u_3^n \\ \vdots \\ u_{J-1}^n \\ 0 \end{pmatrix}$$

$$\begin{aligned} M \cdot U^{n+1} &= UU^n \\ U^{n+1} &= M^{-1} \cdot UU^n \end{aligned}$$

- heat equation $u_t = u_{xx}$ with step initial condition

Summary of schemes for parabolic/heat equation

- explicit scheme – $O(\Delta x^2)$, $\Delta t \approx \Delta x^2$ – too many time steps
- implicit scheme – $O(\Delta x)$ for $\Delta t \approx \Delta x$
- Crank-Nicolson implicit scheme – $O(\Delta x^2)$ for $\Delta t \approx \Delta x$, but not dissipative – oscillations

2.8 Advection diffusion equation

- combination of advection and heat equation (mixed hyperbolic – parabolic)

$$u_t + au_x = bu_{xx}$$

is well posed for $b \geq 0$

- transformation of variables $y = x - at$; coordinate systemm moving with velocity a

$$\begin{aligned} w(t,y) &= u(t, y + at) \\ w_t &= u_t + au_x = bu_{xx} \\ w_y &= u_x, \quad w_{yy} = u_{xx} \\ w_t &= bw_{yy} \end{aligned}$$

as $u(t,x) = w(t, x - at)$, so the solution of the advection diffusion equation moves with speed a and in the moving coordinate system is the solution of the heat equation

- explicit difference scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = b\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}$$

stable for

$$\Delta t \leq \frac{\Delta x^2}{2b} \wedge \Delta t \leq \frac{2b}{a^2}$$

and thus $O(\Delta x^2)$

- implicit scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x} = b\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2}$$

is unconditionaly stable and $O(\Delta x)$ for $\Delta t \approx \Delta x$

- splitting hyperbolic and parabolic parts of $u_t + au_x = bu_{xx}$

$$\begin{aligned} u_j^* &= D^a u_j^n & &\longrightarrow\ u_t + au_x = 0 \\ u_j^{n+1} &= D^h u_j^* & &\longrightarrow\ u_t = bu_{xx} \end{aligned}$$

  scheme D^a approximates advection equation and scheme D^h approximates heat equation

- advection diffusion equation $u_t + u_x = u_{xx}$ with step initial condition

2.9 Conservation Laws-Integral and differential form

- 1D tube filled with compressible ideal gas with density $\rho(x,t)$ and velocity $v(x,t)$
- total mass of interval (x_1, x_2)

$$\int_{x_1}^{x_2} \rho(x,t)d\,x$$

- during a small time dt the mass going through the point x is $\rho(x,t)v(x,t)dt$, so the mass flux at point x is $\rho(x,t)v(x,t)$
- time change of the total mass is equal to the difference of fluxes at the end points – integral form of conservation law

$$\frac{\partial}{\partial t}\int_{x_1}^{x_2} \rho(x,t)d\,x = \rho(x_1,t)v(x_1,t) - \rho(x_2,t)v(x_2,t)$$

- by integration over (t_1, t_2) we get the second form of integral conservation law

$$\int_{x_1}^{x_2} \rho(x,t_2)d\,x - \int_{x_1}^{x_2} \rho(x,t_1)d\,x = \int_{t_1}^{t_2} \rho(x_1,t)v(x_1,t)d\,t - \int_{t_1}^{t_2} \rho(x_2,t)v(x_2,t)d\,t$$

- for differentiable functions $\rho(x,t), v(x,t)$

$$\begin{aligned} \rho(x,t_2) - \rho(x,t_1) &= \int_{t_1}^{t_2} \frac{\partial}{\partial t}\rho(x,t)d\,t \\ \rho(x_2,t)v(x_2,t) - \rho(x_1,t)v(x_1,t) &= \int_{x_1}^{x_2} \frac{\partial}{\partial x}(\rho(x,t)v(x,t))d\,x \end{aligned}$$

- so the second form of the integral conservation law can be rewritten as

$$\int_{t_1}^{t_2}\int_{x_1}^{x_2}\left[\frac{\partial\rho(x,t)}{\partial t} + \frac{\partial\rho(x,t)v(x,t)}{\partial x}\right] d\,x\;d\,t = 0$$

this has to hold for any interval (x_1, x_2) and any interval (t_1, t_2), from which the differential form of the conservation law follows

$$\rho_t + (\rho v)_x = 0$$

- conservative form of scalar conservation law

$$u_t + f(u)_x = 0$$

- advection form

$$u_t + f_u u_x = 0$$

advection equation

$$u_t + a u_x = 0$$

f_u is the speed of waves propagation

- system of conservation laws in 1D – conservative form

$$\vec{U}_t + (\vec{F}(\vec{U}))_x = 0$$

- advection form of the system of conservation laws

$$\vec{U}_t + \vec{F}_{\vec{U}} \cdot \vec{U}_x = 0$$

where $\vec{F}_{\vec{U}}$ is Jacobi matrix

$$\vec{F}_{\vec{U}} = \begin{pmatrix} F^1_{U1}, & F^1_{U2}, & \dots \\ F^2_{U1}, & F^2_{U2}, & \dots \\ \vdots & & \end{pmatrix}$$

- conservation laws are hyperbolic, so $\vec{F}_{\vec{U}}$ has real eigenvalues $\forall x, t$
- eigenvalues

$$\det(\vec{F}_{\vec{U}} - \lambda_i \mathcal{I}) = 0, \lambda_i \in R, i = 1, \cdots, n, \vec{U} \in R^n, \vec{F} \in R^n$$

- eigenvectors (rows)

$$\vec{V}_i \cdot \vec{F}_{\vec{U}} = \lambda_i \vec{V}_i$$

- a system is strictly hyperbolic $\Leftrightarrow$ the Jacobi matrix $\vec{F}_{\vec{U}}$ has mutually different eigenvalues and independent eigenvectors
- for hyperbolic system

$$\mathcal{P} = \begin{pmatrix} \vec{V}_1 \\ \vec{V}_2 \\ \vdots \\ \vec{V}_n \end{pmatrix}$$

is non-singular matrix and

$$\mathcal{P}\cdot\vec{F}_{\vec{U}} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}\cdot\mathcal{P} = \Lambda\cdot\mathcal{P},$$

so that

$$\mathcal{P}\cdot\vec{F}_{\vec{U}}\cdot\mathcal{P}^{-1} = \Lambda$$

- we multiply the system

$$\vec{U}_t + \vec{F}_{\vec{U}}\cdot\vec{U}_x = 0$$

by matrix $\mathcal{P}$ from left to get

$$\mathcal{P}\cdot\vec{U}_t + \mathcal{P}\cdot\vec{F}_{\vec{U}}\cdot\vec{U}_x = 0,$$

which we consider localy at point (x,t), assumption of frozen coefficients, i.e. $\mathcal{P}$ does not depend on x,t

$$(\mathcal{P}\cdot\vec{U})_t + \mathcal{P}\cdot\vec{F}_{\vec{U}}\cdot\mathcal{P}^{-1}\cdot\mathcal{P}\cdot\vec{U}_x = 0$$

- characteristic system

$$\vec{W}_t + \Lambda\cdot\vec{W}_x = 0$$

for characteristic variables $\vec{W} = \mathcal{P}\cdot\vec{U}$

$$W^i_t + \lambda_i W^i_x = 0$$

- local expansion of solution into the system of Jacobi matrix $\vec{F}_{\vec{U}}$ eigenvectors
- eigenvalues λ_i are speeds of waves propagation

2.10 Difference schemes for conservation laws

- conservation law in 1D is

  $$u_t + f(u)_x = 0,$$

  where the flux $f(u)$ is in general non-linear function of u
- Lax-Friedrichs (LF) scheme

  $$\frac{u_j^{n+1} - (u_{j+1}^n + u_{j-1}^n)/2}{\Delta t} + \frac{f(u_{j+1}^n) - f(u_{j-1}^n)}{2\Delta x} = 0$$

  is first order $O(\Delta x)$, diffusive scheme; with ME

  $$u_t + f(u)_x = \frac{\Delta x^2}{2\Delta t}(1 - f_u^2 \Delta t^2/\Delta x^2)u_{xx}$$
- time step

  $$\Delta t = C\frac{\Delta x}{\max|f_u|},$$

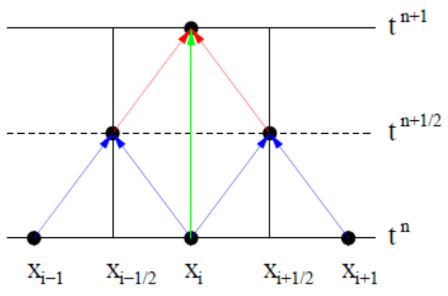
  LF stable for $C \leq 1$

- two-step LF scheme for conservation law has predictor

  $$\frac{u_{j+1/2}^{n+1/2} - (u_{j+1}^n + u_j^n)/2}{\Delta t/2} + \frac{f(u_{j+1}^n) - f(u_j^n)}{\Delta x} = 0,$$

  which computes the solution on the dual (shifted) computational mesh $(n+1/2)\Delta t, (j+1/2)\Delta x$
- corrector of the LF scheme is the predictor shifted by $1/2$ in indices n and j

  $$\frac{u_j^{n+1} - (u_{j+1/2}^{n+1/2} + u_{j-1/2}^{n+1/2})/2}{\Delta t/2} + \frac{f(u_{j+1/2}^{n+1/2}) - f(u_{j-1/2}^{n+1/2})}{\Delta x} = 0$$



- modified equation

  $$u_t + f(u)_x = \frac{\Delta x^2}{4\Delta t}(1 - f_u^2 \Delta t^2/\Delta x^2)u_{xx}$$

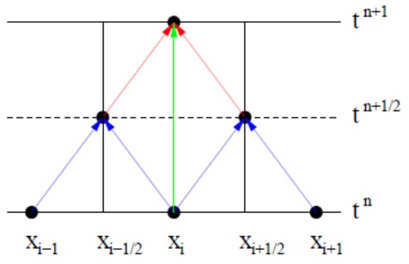
  so two-step LF is $O(\Delta x)$, diffusive scheme – less diffusive than one-step LF, stable for $C \leq 1$

- two-step Lax-Wendroff(LW) scheme uses the same predictor as LF

$$\frac{u_{j+1/2}^{n+1/2}-(u_{j+1}^n+u_j^n)/2}{\Delta t/2}+\frac{f(u_{j+1}^n)-f(u_j^n)}{\Delta x}=0,$$

  and corrector

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+\frac{f(u_{j+1/2}^{n+1/2})-f(u_{j-1/2}^{n+1/2})}{\Delta x}=0$$



- LW for advection equation, i.e. $f(u)=au$, results in

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}+a\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}-\frac{a^2\Delta t}{2}\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{\Delta x^2}=0$$

  which is the known LW scheme for advection equation

- LW is $O(\Delta x^2)$, dispersive scheme, stable for $C\leq 1$
- composite scheme LWLFk – $k-1$ time steps by dispersive LW followed by one step of diffusive LF

Conservativity

- any conservative scheme can be written in the conservative form

$$u_j^{n+1}=u_j^n-\frac{F_{j+1/2}-F_{j-1/2}}{\Delta x},$$

  where $F_{i+1/2}$ is numerical flux

- conservative quantity u on interval $x\in(a,b)$
- by integration of the conservation law $u_t+f(u)_x=0$ over $x\in(a,b)$ we get

$$\frac{\partial}{\partial t}\int_a^b u\,d\,x=f(u(a,t))-f(u(b,t)),$$

  so that time change of the integral of the conservative quantity is equal to the difference of fluxes at the end points of the interval

- similarly for a conservative scheme

$$\sum_{j=1}^{J}u_j^{n+1}\Delta x=\sum_{j=1}^{J}u_j^n\Delta x-F_{J+1/2}+F_{1/2}$$

- LW scheme is directly in conservative form
- for one-step LF scheme

$$F_{j+1/2}=-\Delta x\frac{u_{j+1}^n-u_j^n}{2}+\Delta t\frac{f(u_{j+1}^n)+f(u_j^n)}{2}$$

- for two-step LF scheme

$$F_{j+1/2} = -\Delta x \frac{u^n_{j+1} - u^n_j}{4} + \Delta t \frac{f(u^n_{j+1}) + f(u^n_j)}{4} + \Delta t \frac{f(u^{n+1/2}_{j+1/2})}{2}$$

2.11 Riemann problem for Burgers equation

- the simplest conservation law is Burgers equation

$$u_t + \left(\frac{u^2}{2}\right)_x = 0$$

in advection form

$$u_t + u u_x = 0$$

so that u is speed of waves propagation

- Riemann problem at point x_0 is given by the initial condition

$$u_0(x) = \begin{cases} u_L & \text{pro } x < x_0 \\ u_P & \text{pro } x > x_0 \end{cases}$$

- Riemann problem for Burgers equation
  - $u_L > u_P$ – solution is discontinuous shock wave moving with the speed $s = (u_L + u_P)/2$

$$u(x,t) = \begin{cases} u_L & \text{pro } x - x_0 < st \\ u_P & \text{pro } x - x_0 > st \end{cases}$$

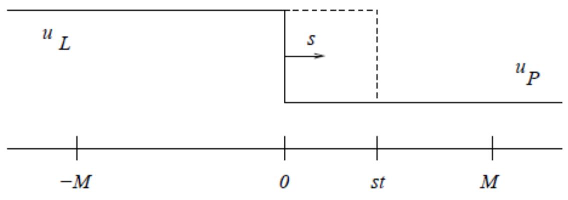
– $u_L < u_P$ – solution is continuous rarefaction wave

- continuous initial condition, $u_t + u u_x = 0$

2.12 Rankine-Hugoniot condition

- shock wave with speed s, left state u_L, right state u_R

$$u_t + f(u)_x = 0$$



$$\int_{-M}^{M} u(x,t)_t \mathrm{d}x + f(u_R) - f(u_L) = 0$$

$$\frac{\partial}{\partial t}\int_{-M}^{M} u(x,t)\mathrm{d}x + f(u_R) - f(u_L) = 0$$

- we express $\int$

$$\int_{-M}^{M} u(x,t)\mathrm{d}x = (M+st)u_L + (M-st)u_R,$$

so that

$$\frac{\partial}{\partial t}\int_{-M}^{M} u(x,t)\mathrm{d}x = s(u_L - u_R),$$

thus

$$s(u_L - u_R) + f(u_R) - f(u_L) = 0$$

- Rankine-Hugoniot condition

$$s = \frac{f(u_L) - f(u_R)}{u_L - u_R}$$

- for Burgers equation $f(u) = u^2/2$

$$s = \frac{u_L^2 - u_R^2}{2(u_L - u_R)} = \frac{u_L + u_R}{2}$$

- for systems of conservation laws: jumps in conservative variables and in fluxes on the shock wave have to be linearly dependent

$$s = \frac{\vec{F}(\vec{U}_L) - \vec{F}(\vec{U}_R)}{\vec{U}_L - \vec{U}_R}$$

or better

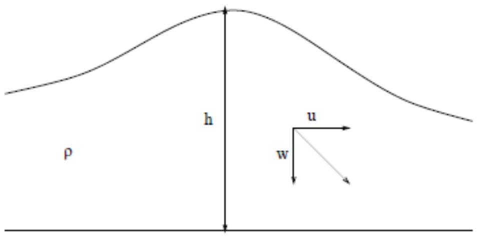
$$s(\vec{U}_L - \vec{U}_R) = \vec{F}(\vec{U}_L) - \vec{F}(\vec{U}_R)$$

2.13 Shallow water equations

- the simplest system of conservation laws are shallow water equations

$$\begin{aligned} h_t + (hu)_x &= 0 \\ (hu)_t + \left(hu^2 + g\frac{h^2}{2}\right)_x &= 0, \end{aligned}$$

  where $h(t,x)$ is thickness of the water layer (depth), $u(t,x)$ is horizontal velocity of water and g is gravitation acceleration



- in conservative variables $\varphi = gh, m = ghu$

$$\begin{aligned} \varphi_t + m_x &= 0 \\ m_t + \left(\frac{m^2}{\varphi} + \frac{\varphi^2}{2}\right)_x &= 0, \end{aligned}$$

- system of conservation laws

$$\vec{U}_t + (\vec{F}(\vec{U}))_x = 0$$

where

$$\vec{U} = \begin{pmatrix} \varphi \\ m \end{pmatrix}, \qquad \vec{F}(\vec{U}) = \begin{pmatrix} m \\ \frac{m^2}{\varphi} + \frac{\varphi^2}{2} \end{pmatrix}$$

- advection form for system

$$\vec{U}_t + \vec{F}_{\vec{U}} \cdot \vec{U}_x = 0$$

where Jacobi matrix $\vec{F}_{\vec{U}}$ is

$$\vec{F}_{\vec{U}} = \begin{pmatrix} 0 & 1 \\ -\frac{m^2}{\varphi^2} + \varphi & \frac{2m}{\varphi} \end{pmatrix}$$

- eigenvalues of Jacobi matrix

$$\det(\vec{F}_{\vec{U}} - \lambda I) = \det \begin{pmatrix} -\lambda & 1 \\ -\frac{m^2}{\varphi^2} + \varphi & \frac{2m}{\varphi} - \lambda \end{pmatrix} =$$

$$= \lambda^2 - \frac{2m}{\varphi}\lambda + \frac{m^2}{\varphi^2} - \varphi = 0$$

$$D = \frac{4m^2}{\varphi^2} - \frac{4m^2}{\varphi^2} + 4\varphi = 4\varphi$$

$$\lambda_{1,2} = \frac{m}{\varphi} \pm \sqrt{\varphi} = u \pm \sqrt{gh}$$

- both LF and LW (as well as LWLFk) can be used for systems

- time step computation – done adaptively after each time step as v_{max} is changing during the computation

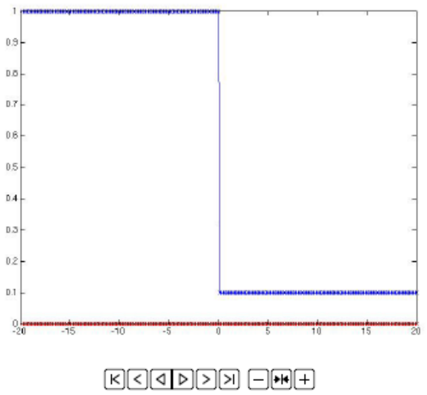
$$\Delta t = C \frac{\Delta x}{v_{max}},$$

where

$$v_{max} = \max_j \left| u_j \pm \sqrt{g h_j} \right|$$

is maximal speed of wave propagation given by eigenvalues of the Jacobi matrix

- dam break problem



- solution of a Riemann problem for shallow water equations are always 2 waves, left one and right one
- left wave is either shock or rarefaction wave, right wave is also either shock or rarefaction wave

Other numerical methods:

- Finite volume methods - 2D and 3D
- Finite element methods (FEM)
  - -Galerkin method
  - -Rayleigh-Ritz method
  - -weighted residual method
  - -discontinous Galerkin (DG) method
  - -spectral methods - Fourier series

Conclusion:

-finite difference method-hyperbolic and parabolic evolution equations conservation laws - hyperbolic non-linear PDEs

-computer laboratory session tomorrow

-lecture and computer laboratory session ”Fluid simulations of laser-produced plasmas" by Milan Kucharik-Euler equations-system of conservation laws for compressible fluid dynamics

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Chapter 3: Fluid Simulations of Laser-Produced Plasmas

M. Kucharik

1 Fluid Simulations of Laser-Produced Plasmas

In this Section, we present basic models and methods for hydrodynamic simulations of laser produced plasmas. After a brief description of Euler equations in Eulerian and Lagrangian reference frames, the main idea and basic steps of an indirect ALE method based on staggered discretization is reviewed, and essential physical models are summarized. Several selected numerical examples demonstrate ability of the approach to treat laser-target interactions in physically-relevant manner.

1.1 Euler Equations

Fluid simulations of laser-produced plasmas represent a useful tool for theoreticians and experimentalists allowing them to investigate processes during laser-plasma interaction, which are often impossible to observe directly in the experiments. Plasma hydrodynamics allows not only interpretation of experimental results, but is also often used for designing of the experimental setup or detailed analysis of particular processes during the experiment.

The simplest model of fluid dynamics is based on the classical Euler equations,

$$\rho_t + \nabla \cdot (\rho\,\vec{w}) = 0\,, \tag{1.1}$$

$$(\rho\,\vec{w})_t + \nabla \cdot (\rho\,\vec{w}^2) + \nabla\, p = \vec{0}\,, \tag{1.2}$$

$$E_t + \nabla \cdot (\vec{w}\,(E+p)) = 0\,, \tag{1.3}$$

representing conservation laws of mass, momenta, and total energy in the static (Eulerian) reference frame. Here, ρ represents fluid density, $\vec{w} = (u, v, w)$ is the velocity vector, p stands for pressure, and $E = \rho\,\varepsilon + 1/2\,\rho\,|\vec{w}|^2$ is the total energy density, with ε being fluid specific internal energy. A particular Equation of State (EOS) closes the system, interconnecting fluid density, pressure, and specific internal energy,

$$p = \mathcal{P}(\rho, \varepsilon)\,. \tag{1.4}$$

For ideal gas, this equation is linear in both density and internal energy,

$$p = (\gamma - 1)\,\rho\,\varepsilon \tag{1.5}$$

with γ representing ratio of gas specific heats, depending on the particular gas, while it can very complicated (and often tabulated) in more realistic plasma models.

The system can be transformed to the moving (Lagrangian) reference frame by introducing the total (material) derivative,

$$\frac{d}{dt} = \frac{\partial}{\partial t} + \frac{\partial x}{\partial t}\frac{\partial}{\partial x} = \frac{\partial}{\partial t} + u\,\frac{\partial}{\partial x}\,, \tag{1.6}$$

or $\frac{d}{dt} = \frac{\partial}{\partial t} + \vec{w} \cdot \nabla$ in multi-dimensional case. The system of Euler equation can be then written in the form

$$\frac{d\,\rho}{d\,t} + \rho\nabla \cdot \vec{w} = 0\,, \tag{1.7}$$

$$\rho\,\frac{d\,\vec{w}}{d\,t} + \nabla\,p = \vec{0}\,, \tag{1.8}$$

$$\rho\,\frac{d\,\varepsilon}{d\,t} + p\nabla \cdot \vec{w} = 0\,. \tag{1.9}$$

On the moving mesh, the velocity of the nodes needs to be defined, which is usually determined by solving the following ODE,

$$\frac{d\,\vec{x}}{d\,t} = \vec{w}\,. \tag{1.10}$$

1.2 Eulerian, Lagrangian, and ALE Methods

The classical Eulerian methods solve the system of Euler equations in the Eulerian formulation (1.1)-(1.3). The computational mesh is static in time and the fluid moves through it in the form of mass fluxes. Due to the static computational mesh, the methods are usually robust, efficient, and simpler to analyze. The exist a large number of classical textbooks related to Eulerian methods, see for example [1, 2, 3].

Unfortunately, in simulations related to laser-target interactions, severe material compressions and expansion are present. The Eulerian mesh is not suitable in this situation – it does not provide high enough resolution at strong shocks, it needs to cover large computational domain when nothing happens for most of the simulation, etc. In this situation, Lagrangian methods solving the system of Euler equations in its Lagrangian formulation (1.7)-(1.9) is much better suited. The computational mesh naturally follows the fluid motion, no mass fluxes are present. This class of method was introduced in [4].

However, due to the motion of the computational mesh, mesh disturbances can occur when shear flows or fluid vortexes are present. Degenerate patches can appear in the mesh, such as non-convex, self-intersecting, or even inverted (negative-volume) cells, resulting in generation of numerical error or even simulation failure. In the pioneering paper [5], movement of the mesh nodes was decoupled from the fluid motion. The class of methods based on this framework is usually called Arbitrary Lagrangian-Eulerian (ALE), and combines the best properties of both approaches – the computational mesh moves with the fluid as in the Lagrangian framework, but the Eulerian part of the algorithm keeps the mesh smooth.

Here, we focus on the description of a typical indirect ALE algorithm in staggered discretization, which is one of the most popular methods used nowadays in hydrodynamic codes [6, 7, 8, 9, 10, 11].

1.2.1 Indirect ALE Algorithm

There exist two basic classes of ALE methods – direct and indirect. In the direct ALE, the Euler equations are formulated with mesh velocity independent of fluid velocity, with mesh validity guaranteed through a certain filtering technique, removing shear or rotational modes from the velocity field. On the other hand, the indirect ALE is more straightforward. It incorporates a traditional Lagrangian solver, advancing the solution and the computational mesh in time. Validity of the mesh is enforced by an explicit mesh rezoning procedure smoothing the computational mesh. The last essential step of an indirect ALE algorithm is remap, transferring all fluid quantities conservatively from the old (Lagrangian) computational mesh to the new (rezoned) one. Rezoning and remapping steps are together called the Eulerian part of the ALE algorithm, as fluid effectively moves between computational cells at this stage.

There exist several strategies for joining the mentioned steps into a single ALE algorithm, basically defining the amount of Eulerian ingredient in the solution. The best strategy cannot be simply specified and usually depends on the particular problem.

The most natural approach employs the ALE algorithm in almost Lagrangian manner. The simulation detects problematic cells and after they appear (or few steps earlier), the mesh is rezoned and quantities remapped. This strategy minimizes the numerical error resulting from the remapping numerical method, and the solution is close to Lagrangian. On the other hand, the mesh quality is usually quite low and due to this facts, various numerical artifacts (such as oscillations, for example) can be present in the solution.

The opposite strategy is based on mesh rezoning and remapping after every single Lagrangian step. The solution is than close to Eulerian and the mesh remains very smooth, preventing any unwanted numerical artifacts. One can even use initial mesh instead of mesh smoothing, resulting in a completely Eulerian numerical method. In general, large amount of Eulerian steps introduces a significant diffusion in the solution, resulting from the remapping numerical error.

There are many possibilities of strategies between these two extrema. As an example, let us mention mesh smoothing and remapping after a fixed number of Lagrangian steps, which is often used in practical simulations. This strategy prevents most of the numerical problems, while keeping the mesh close to the Lagrangian one. One can also switch strategy during a single simulation, for example run in purely Lagrangian regime for a certain period of time, and switch to ALE after first mesh degeneracy appears.

In the following sections, we will describe all three steps in more detail.

1.2.2 Staggered Lagrangian Solver

There exist several options for formulation of the Lagrangian numerical solvers. Most of them can be classified either as cell-centered or staggered method.

The cell-centered methods define all fluid quantities at the same place, in the center of each mesh cell. For an example of a modern cell-centered method, see for example [12]. The main advantage arises when switching to ALE – all fluid quantities can be remapped in the same manner. On the other hand, one need to solve a local Riemann problem at each mesh node to estimate its velocity, and more complicated incorporation of velocity boundary conditions needs to be done.

We mostly focus here on the more classical class of staggered Lagrangian methods, which define the fluid thermodynamic quantities (density, pressure, specific internal energy) in the mesh cells, while the kinematic quantities (mainly the velocity vector) are defined in mesh nodes. This approach allows simple incorporation of velocity boundary conditions, the mesh node motion is directly defined through its nodal velocity, however, staggered discretization requires more complicated remapping approaches. Here, we briefly describe the compatible staggered Lagrangian solver based on mimetic discretization of the Euler equations, for full derivation of the scheme, see [13].

The whole scheme can be summarized as follows:

1. Sum nodal pressure forces $\vec{F}_n^p$ and viscosity forces $\vec{F}_n^q$ [14, 15] (eventually other forces needed [16 17]) to construct the total nodal force $\vec{F}_n$, use it to update nodal velocity,
$$\vec{w}_n^{t^{n+1}} = \vec{w}_n^{t^n} + \frac{\Delta t}{m_n}\vec{F}_n \,. \tag{1.11}$$
2. Update nodal coordinates,
$$\vec{x}_n^{t^{n+1}} = \vec{x}_n^{t^n} + \Delta t\, \vec{w}_n^{t^{n+1/2}} \,, \tag{1.12}$$
where the time-centered velocity $\vec{w}_n^{t^{n+1/2}}$ is computed just as average of old and new values.
3. From the new cell geometry, update cell volumes $V_c^{t^{n+1}}$ and densities $\rho_c^{t^{n+1}} = m_c/V_c^{t^{n+1}}$, where the cell mass m_c is constant in time due to the Lagrangian nature of the scheme.
4. Update the specific internal energy as
$$\varepsilon_c^{t^{n+1}} = \varepsilon_c^{t^n} + \frac{\Delta t}{m_c} W_c \,, \tag{1.13}$$
where heating W_c due to compression/expansion of the cell is computed from the velocities and forces acting on the particular cell.
5. Finally, update new cell pressure from the equation of state, $p_c^{t^{n+1}} = \mathcal{P}(\rho_c^{t^{n+1}}, \varepsilon_c^{t^{n+1}})$.

This scheme is typically used in a two-step predictor-corrector version, guaranteeing its second order of accuracy.

1.2.3 Mesh rezoning

There exist many useful methods for untangling and smoothing of computational meshes in the context of ALE hydrodynamic codes. In practice, fast and simple algorithms are typically used, which is especially important when 3D calculations are performed. Let us name especially the classical and widely used Winslow method [18], performing one iteration of solution of an elliptic PDE. More advanced methods contain for example minimization of the condition number functional [19], representing a measure of the mesh geometric quality, or the Reference Jacobian mesh smoothing technique [20], which is based on a similar idea, while trying to preserve the original mesh features. For mesh untangling, let us name the approach of modified condition number minimization [21], or fully geometric methods based on an explicit construction of the feasible set for each node, see for example [22]. These methods are used in ALE algorithms to guarantee validity of the mesh, while minimizing nodal motion to avoid excessive diffusion of the following remapping step.

1.2.4 Quantity Remapping

The main goal of the remapping step if to transfer all conservative quantities (and their densities) from the distorted Lagrangian mesh to the rezoned one. Let us assume that we know the Lagrangian fluid density ρ_c in the old cell c, which is considered in this context to be the mean value of some unknown underlying density $\rho(\vec{x})$: $\rho_c = m_c/V_c$, where $m_c = \int_c \rho(\vec{x})\, dV$. The goal is to compute new cell $\tilde{c}$ density and mass, $\rho_{\tilde{c}} = m_{\tilde{c}}/V_{\tilde{c}}$, where $m_{\tilde{c}} = \int_{\tilde{c}} \rho(\vec{x})\, dV$. Most remapping methods are based on piece wise linear limited reconstruction [23] of $\rho(\vec{x})$ and its exact or approximate integration, guaranteeing conservation, second-order of accuracy, continuity, and bound-preservation of the remapping scheme.

The remapping methods based on exact integration typically construct intersections of the new cell $\tilde{c}$ with the original mesh. The final new mass is then obtained by summing the contribution from all the intersections, $m_{\tilde{c}} = \sum_{c'} \int_{\tilde{c}\cap c'} \rho(\vec{x})\, dV$, where the density function is approximated by its reconstruction in the particular old cell c' [24]. The approximate models perform the remap typically in a flux form, constructing the fluxes by integrating the reconstructed density function over swept regions, defined by the motion of each cell edge to its new position during mesh rezoning [25]. The exact methods guarantee bound-preservation of the scheme and simple to generalize into multi-material case [26], however, construction of the intersections is computationally expensive and has issues with robustness, especially in 3D. On the other hand, the swept-based methods are faster and more robust, however, bound-preservation can be violated [27] and are difficult to generalize for multi-material simulations. To solve this issue, the hybrid remapping techniques have been developed [28, 29, 30], combining both approaches in different parts of the computational domain. Similar approach can be performed in single-material case [31], choosing the best method according to a local error analysis [32].

Remapping of all fluid quantities needs to be performed consistently, which is simpler in cell-centered approaches. In the staggered methods, different location of thermodynamic and kinematic quantities leads to more complicated approaches, see for example [33, 34] for available methods. Remapping of nodal velocity needs to be done carefully and consistently with the rest of the quantities to avoid increase of the kinetic energy discrepancy [35] or symmetry violation of velocity field [36].

1.3 Physical Models

To approximate laser-generated plasma, standard Euler equations (1.7)-(1.9) need to be enhanced by additional terms representing heat conductivity and laser absorption,

$$\frac{d\,\rho}{d\,t} = -\rho\,\nabla\cdot\vec{w}\,, \tag{1.14}$$

$$\rho\,\frac{d\,\vec{w}}{d\,t} = -\nabla\,p\,, \tag{1.15}$$

$$\rho\,\frac{d\,\varepsilon}{d\,t} = -p\nabla\cdot\vec{w} + \nabla\cdot(\kappa\nabla T) - \nabla\cdot\vec{I}\,, \tag{1.16}$$

where T represents fluid temperature, κ is heat conductivity coefficient (computed by the standard Spitzer-Harm formula [37]), and $\vec{I}$ is the laser beam intensity vector. This modification represents the simplest approximation of the realistic laser plasma model.

The first necessary model is a realistic equation of state. The simple ideal gas EOS is reasonably valid in low density corona, however, in compressed material, more relevant EOS has to be used. As examples, let us mention the often used semi-analytic Quotidian EOS (QEOS) [38] based on the Thomas-Fermi theory electrons and Cowan model for ions, or the tabulated SESAME EOS [39]. Let us also mention the HerEOS library [40], allowing thermodynamically-consistent high-order Hermite interpolation from any tabulated EOS data, improving significantly the computational cost of the EOS evaluation, which is usually the dominant part of a realistic simulation.

Another necessary model is approximation of thermal conductivity, contributing not only to the physical relevance of the simulation results (correct speed of spreading waves), but also to the robustness of the simulation. Thermal conductivity is described by the parabolic term in (1.16), which can be separated from the rest of the system by operator splitting, and the following equation can be formulated fully in temperature,

$$T_t = \frac{1}{\rho\,\varepsilon_T}\nabla\cdot(\kappa\nabla T)\,. \tag{1.17}$$

The derivative of the specific internal energy with respect to temperature ε_T can be computed numerically for general EOS, and the classical Spitzer-Harm formula [37] is used for the evaluation of the heat conductivity coefficient κ. A possible approach for solving this equation is based on the mimetic method using support operators [41]. Discrete operators of gradient and divergence are constructed, mimicking the properties of the continuous operators, which are used for the parabolic term discretization. Temporal derivative is discretized by the standard finite difference, and a fully implicit scheme is constructed to avoid excessive restriction of the time step due to a quadratic term in the CFL condition. The final system is solved by the conjugate gradient method (with pre-conditioning). This method is second order accurate, works well on bad quality meshes, and allows a discontinuous diffusion coefficient. It is also simple to supplement this approach by a heat flux limiter, guaranteeing more realistic plasma energy.

A crucial part of a hydrodynamic code is a suitable model for absorption of a laser beam energy, represented by the source term in (1.16). The simplest approach is based on an explicit location of the critical density surface and full absorption of each laser beam ray in the particular cell intersected by this surface. There exist two main drawbacks of this approach: first, it is necessary to provide the absorption coefficient (which is, in general, dependent on the material properties and laser parameters); and second, all energy of the particular ray is absorbed in a single cell, which leads to a series of “cell explosions”. To avoid these problems, more advanced models are typically used. Let us for example mention the advanced model of raytracing, explicitly tracking each ray in the computational mesh, including its rarefactions at cell boundaries, or a wave-based approach [42] employing stationary solution of Maxwell equations, spreading the absorption region to multiple cells.

To approximate the 3D reality without the need to solve fully 3D equations, cylindrical symmetry of the laser-related problems is often utilized, and 2D $r - z$ geometry is often implemented in hydrodynamic codes. Each 2D cell then represents a ring (rectangular toroid) around $r = 0$ axis, so even its volume and centroid are dependent on its location in the r direction. To switch to cylindrical geometry, all additional r-factor need to be introduced into all integrals. It leads to r-factor added mainly during the force construction in the staggered Lagrangian stage and in integration of function reconstruction in remapping stage, while the mesh rezoning stage is not affected. It is possible to formulate all integrals properly and derive corresponding analytic formulas similarly as in Cartesian case.

There exist several more improvements contributing to more realistic modeling of laser/plasma interactions, let us mention few of them. The first of them is a two-temperature model, splitting a common fluid temperature to separate temperatures of electrons and ions, which is especially important in simulations of non-ideal plasmas. Two distinct energy equations (1.16) need to be solved, with an additional exchange term, representing heat transition from electrons to ions. Another significant issue is a phase transition model, especially important in simulations containing solid target melting and evaporation due to its interaction with a laser beam. It prevents solid target from unrealistic expansion into vacuum, resulting from purely fluid description of the material. The last model which we mention here is the non-local energy transport, representing long-distance energy transfer due to material radiation. Such model can be important in simulations of high-Z materials, for which target preheating is much better modeled.

1.4 Numerical Examples

Most of the models described in the previous sections are implemented in the hydrodynamic Prague ALE (PALE) code, which is being developed at the FNSPE, CTU in Prague. For an overview of the included methods and possible types of simulations, see for example [43]. This code is primarily used for hydrodynamic simulations related the experiments performed at the PALS laser facility. Here, we only briefly show one fluid example and one more realistic laser/plasma example, to demonstrate capabilities of typical hydrodynamic simulations.

1.4.1 Sedov blast wave

The first example is a standard Sedov blast wave [44], which is used here to demonstrate the differences between Eulerian, Lagrangian, and ALE regimes of the simulation. The initial fluid is static everywhere in the $\langle -1.1, 1.1\rangle^2$ square domain covered by an equidistant 90^2 computational mesh. The fluid has a constant density $\rho = 1$ and specific internal energy $\varepsilon = 10^{-8}$, except 4 central cells, where $\varepsilon = 409.7$. Ideal gas EOS with $\gamma = 1.4$ is used. The central high energy region represents an explosion in the fluid, generating a circular shock wave spreading outside from the point of explosion.

In Figure 1, we can see the density profile and the computational mesh at $t = 1$, obtained by an Eulerian, Lagrangian, and ALE20 method (with rezoning and remapping after every 20 Lagrangian steps). As we can see, the Eulerian method keeps the static mesh, while it is moving with the fluid in the Lagrangian simulation. The Lagrangian cells are huge in the center of the domain, where density decreases, while the mesh is naturally refined by the material compression at the shock wave, which much better resolved and the peak density is higher. For the ALE method, the mesh is similar to the Lagrangian one, the shock wave remains nicely resolved as in the Lagrangian case,

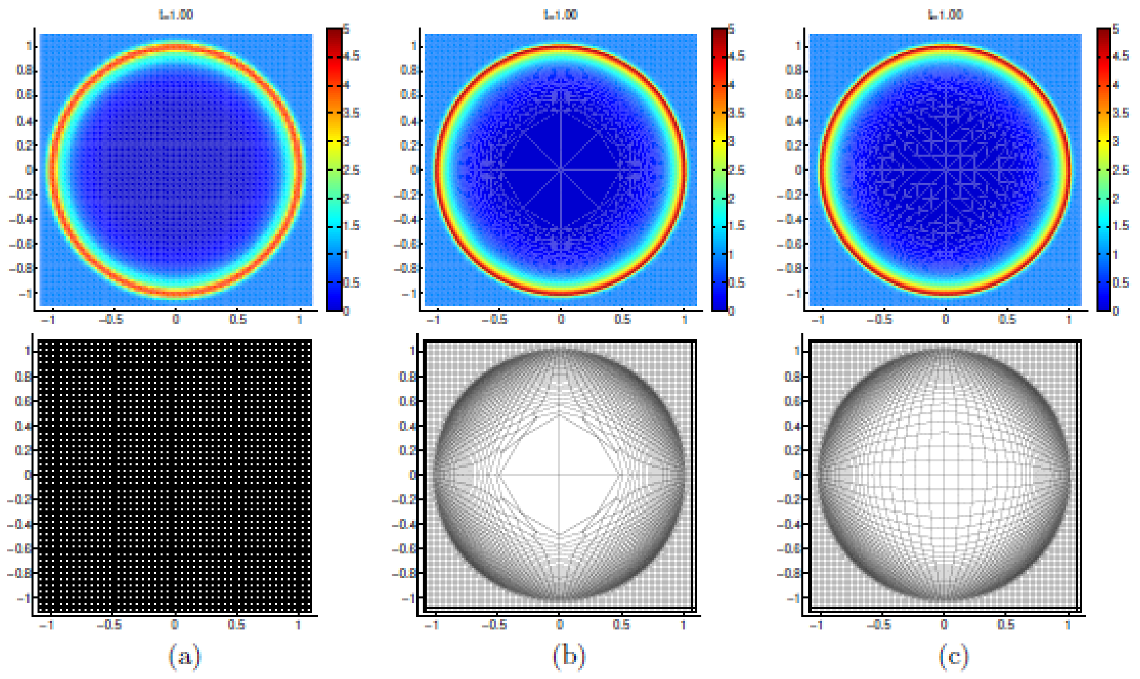


Figure 1: Density profile and computational mesh of Sedov blast wave problem, computed by Eulerian (a), Lagrangian (b), and ALE20 (c) methods on 90^2 mesh.

however, in the center, the mesh is closer to the uniform mesh, avoiding very low resolution and large numerical error in the domain center.

1.4.2 Laser induced cavity pressure acceleration (LICPA) scheme

The second example is inspired by real experiments at the PALS laser facility, see for example [45] for more details. Setup of such experiment is shown in Figure 2. A metal projectile covered by a plastic ablator is situated in a channel covered by a cavity. An intensive laser beam enters the cavity through a small hole, it is absorbed in the ablator and accelerates the projectile to a very high velocity (hundreds of km/s). After it moves through the whole channel, it hits a massive target, generates a shock wave, which moves from the point of impact, melts and evaporates the target material, and creates a crater.

Hydrodynamic simulations were used here for preparation of experimental parameters, and for analysis and interpretation of experimental results. Various modifications have been simulated – different ablator and projectile widths and materials, or different laser energies and frequencies. Various aspects of the experiments have been studied numerically, mostly focusing on hydroefficiency – increasing the portion of laser energy transformed to the shock wave energy. Various simulation data (such as impact velocity, shock wave speed, and crater size) have been compared to the experimental results, and reasonably good agreement was achieved.

In Figure 3, we can see the results of a particular LICPA simulation, performed in two steps – the acceleration and impact processes are simulated separately. A $2.8\,\mu m$ thick Au projectile covered by $5\,\mu m$ thick CH ablator is placed in a channel with radius $150\,\mu m$. The PALS (3rd harmonic, $438\,nm$) laser pulse with $200\,J$ has a super-Gaussian spatial profile with radius $90\,\mu m$ (containing 80% of pulse energy) and Gaussian temporal profile with $\tau_{FWHM} = 300\,ps$. The first

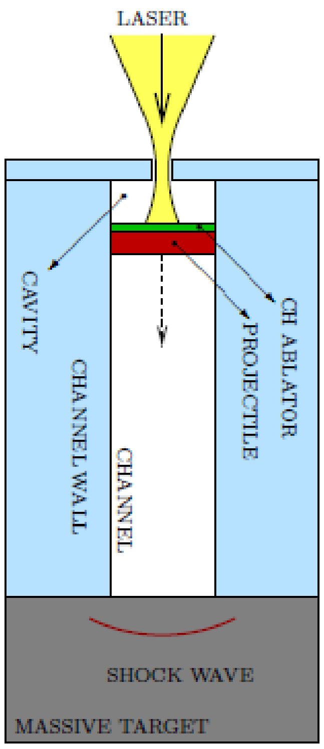
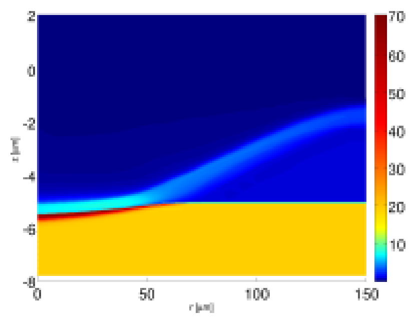


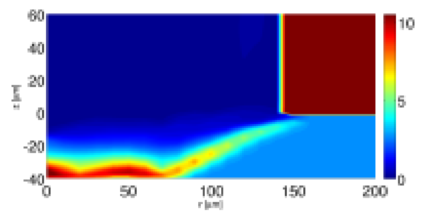
Figure 2: Setup of LICPA experiment.

part (laser absorption and projectile acceleration) is performed in ALE regime, allowing almost Lagrangian mesh motion. Geometrical computational mesh (finer at the upper part of the domain, where laser is being absorbed) covering the projectile and the ablator is used. In Figure 3 (a), we can see the density profile at $t = 250\,ps$ before the pulse maximum. The spatial profile of the pulse forms the shape of the shock wave in the ablator and starts its transition into (still static) higher-density projectile. After the projectile is accelerated by the pulse, it moves through the channel and reaches its end ($2\,mm$ from the initial projectile location) at $t = 800\,ps$ after the maximum, as can be seen in Figure 3 (b). The heavy part of the projectile is still relatively compact before the impact.

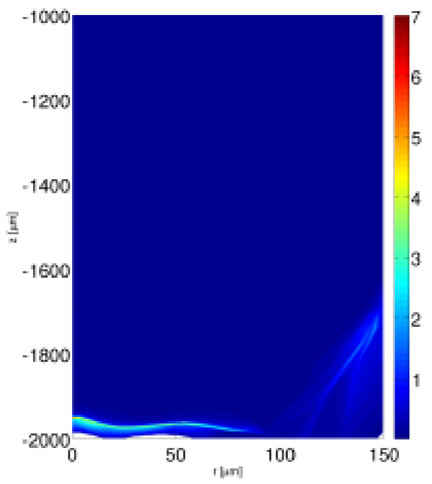
The impact process is modeled by a fully Eulerian simulation to avoid mesh degeneracies at the channel corner short after the impact. The initial data in the channel are taken from the acceleration simulation. The formation of the shock wave in the target material at $t = 0.38\,ns$ after the impact can be seen in Figure 3 (c). We can observe similar shape of the shock wave as the shape of the massive impacting projectile. The material is compressed, the temperature is significantly increased, and a the generated shock wave starts spreading from the point of impact. The temperature profile at $t = 750\,ns$ after the impact can be seen in Figure 3 (d), when the shock wave is already too weak to melt the target material further. The gray colormap shows the solid Aluminum, while the colored region represents the melted and evaporated material, the spherical interface represents the crater boundary. The crater size has been computed and compared to the experimental data, showing relatively good agreement. Many other characteristics can be computed, for example maximum pressure at the shock wave and its temporal development, hydroefficiency, amount of kinetic and internal energy in the target material, speed of crater devel-



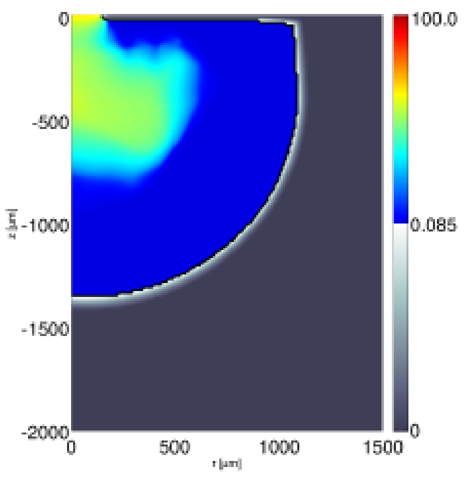
(a)



(c)



(b)



(b)

Figure 3: Density (a-c) and temperature (d) profile at different times of a LICPA simulation: (a) $250\,ps$ before pulse maximum – absorption; (b) $800\,ps$ after pulse maximum – reaching end of channel and impact; (c) $0.38\,ns$ after impact – shock wave formation; (d) $750\,ns$ after impact – developed crater.

opment, profiles along the $z = 0$ axis, and others, allowing deeper insight into the processes during various parts of the experiments. See for example [46] for examples of such simulations and their comparison with experimental data.

1.5 Conclusions

In this short note, we tried to describe the main abilities of typical laser-target simulations via hydrodynamic codes. We have described one particular approach based on ALE technique employing a staggered Lagrangian solver. Several numerical models have been briefly discussed, improving physical relevance for laser-generated plasmas. We have tried to convince the reader that hydrodynamic simulations represent an important tool for studying processes during realistic experiments, which cannot be often measured directly by diagnostic tools. Development of new numerical methods for laser plasma hydrodynamics is an attractive topic, and a vast amount of ongoing research in this field is currently performed in the scientific community.

References

[1] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Birhkauser Verlag, Basel, 1992. ISBN 3-7643-2723-5.

[2] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer Verlag, Berlin, Heidelberg, 1997. ISBN 3-540-61676-4.

[3] J. H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer-Verlag Berlin Heidelberg New York, 2002. ISBN 3-540-65373-2.

[4] R. Courant and K. O. Friedrichs. *Supersonic Flow and Shock Waves*, volume 21 of *Applied Mathematical Sciences*. Springer Verlag, 1999. Reprint of 1st ed. Interscience Publishers, New York 1948. ISBN 0-387-90232-5.

[5] C. W. Hirt, A. A. Amsden, and J. L. Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *J. Comput. Phys.*, 14(3):227–253, 1974.

[6] D. J. Benson. Computational methods in Lagrangian and Eulerian hydrocodes. *Comput. Methods Appl. Mech. Eng.*, 99(2-3):235–394, 1992.

[7] D. E. Burton. Multidimensional discretization of conservation laws for unstructured polyhedral grids. Technical Report UCRL-JC-118306, Lawrence Livermore National Laboratory, 1994.

[8] J. S. Peery and D. E. Carroll. Multi-material ALE methods in unstructured grids. *Comput. Methods Appl. Mech. Eng.*, 187(3-4):591–619, 2000.

[9] A. C. Robinson, T. A. Brunner, S. Carroll, R. Drake, C. J. Garasi, T. Gardiner, T. Haill, H. Hanshaw, D. Hensinger, D. Labreche, R. Lemke, E. Love, C. Luchini, S. Mosso, J. Niederhaus, C. C. Ober, S. Petney, W. J. Rider, G. Scovazzi, O. E. Strack, R. Summers, T. Trucano, V. G. Weirs, M. Wong, and T. Voth. ALEGRA: An arbitrary Lagrangian-Eulerian multimaterial, multiphysics code. In *Proceedings of the 46th AIAA Aerospace Sciences Meeting*, Reno, NV, 2008.

[10] J. W. Murphy and A. Burrows. BETHE-hydro: An arbitrary Lagrangian-Eulerian multidimensional hydrodynamics code for astrophysical simulations. *Astrophys. J. Suppl. Ser.*, 179(1):209–241, 2008.

[11] R. Liska, M. Kucharik, J. Limpouch, O. Renner, P. Vachal, L. Bednarik, and J. Velechovsky. ALE methods for simulations of laser-produced plasmas. In Jaroslav Fořt, Jiří Fürst, Jan Halama, Raphaèl Herbin, and Florence Hubert, editors, *Finite Volumes for Complex Applications VI Problems & Perspectives*, volume 4 of *Springer Proceedings in Mathematics*, pages 857–873. Springer, 2011.

[12] P.-H. Maire, R. Abgrall, J. Breil, and J. Ovadia. A cell-centered Lagrangian scheme for two-dimensional compressible flow problems. *SIAM J. Sci. Comput.*, 29(4):1781–1824, 2007.

[13] E. J. Caramana, D. E. Burton, M. J. Shashkov, and P. P. Whalen. The construction of compatible hydrodynamics algorithms utilizing conservation of total energy. *J. Comput. Phys.*, 146(1):227–262, 1998.

[14] E. J. Caramana, M. J. Shashkov, and P. P. Whalen. Formulations of artificial viscosity for muti-dimensional shock wave computations. *J. Comput. Phys.*, 144(2):70–97, 1998.

[15] J. C. Campbell and M. J. Shashkov. A tensor artificial viscosity using a mimetic finite difference algorithm. *J. Comput. Phys.*, 172(2):739–765, 2001.

[16] E. J. Caramana and M. J. Shashkov. Elimination of artificial grid distortion and hourglass-type motions by means of Lagrangian subzonal masses and pressures. *J. Comput. Phys.*, 142(2):521–561, 1998.

[17] M. Kucharik, G. Scovazzi, M. Shashkov, and R. Loubere. A multi-scale residual-based anti-hourglass control for compatible staggered Lagrangian hydrodynamics. *J. Comput. Phys.*, 354:1–25, 2018.

[18] A. M. Winslow. Equipotential zoning of two-dimensional meshes. Technical Report UCRL-7312, Lawrence Livermore National Laboratory, 1963.

[19] P. M. Knupp. Matrix norms and the condition number: A general framework to improve mesh quality via node-movement. In *Proceedings of 8th International Meshing Roundtable*, 1999.

[20] P. M. Knupp, L. G. Margolin, and M. J. Shashkov. Reference Jacobian optimization-based rezone strategies for arbitrary Lagrangian Eulerian methods. *J. Comput. Phys.*, 176(1):93–128, 2002.

[21] J. M. Escobar, E. Rodriguez, R. Montenegro, G. Montero, and J. M. Gonzalez-Yuste. Simultaneous untangling and smoothing of tetrahedral meshes. *Comput. Methods Appl. Mech. Eng.*, 192(25):2775–2787, 2003.

[22] M. Berndt, M. Kucharik, and M. Shashkov. Using the feasible set method for rezoning in ALE. *Procedia Comput. Sci.*, 1(1):1879–1886, 2010.

[23] T. J. Barth. Numerical methods for gasdynamic systems on unstructured meshes. In C. Rohde D. Kroner, M. Ohlberger, editor, *An introduction to Recent Developments in Theory and Numerics for Conservation Laws, Proceedings of the International School on Theory and Numerics for Conservation Laws*, Berlin, 1997. Lecture Notes in Computational Science and Engineering, Springer. ISBN 3-540-65081-4.

[24] L. G. Margolin and M. Shashkov. Second-order sign-preserving conservative interpolation (remapping) on general grids. *J. Comput. Phys.*, 184(1):266–298, 2003.

[25] J. K. Dukowicz and J. R. Baumgardner. Incremental remapping as a transport/advection algorithm. *J. Comput. Phys.*, 160(1):318–335, 2000.

[26] M. Kucharik, R. V. Garimella, S. P. Schofield, and M. J. Shashkov. A comparative study of interface reconstruction methods for multi-material ALE simulations. *J. Comput. Phys.*, 229(7):2432–2452, 2010.

[27] M. Kucharik, M. Shashkov, and B. Wendroff. An efficient linearity-and-bound-preserving remapping method. *J. Comput. Phys.*, 188(2):462–471, 2003.

[28] M. Kucharik, J. Breil, S. Galera, P.-H. Maire, M. Berndt, and M. Shashkov. Hybrid remap for multi-material ALE. *Comput. Fluids*, 46(1):293–297, 2011.

[29] M. Berndt, J. Breil, S. Galera, M. Kucharik, P.-H. Maire, and M. Shashkov. Two-step hybrid conservative remapping for multimaterial arbitrary Lagrangian-Eulerian methods. *J. Comput. Phys.*, 230(17):6664–6687, 2011.

[30] M. Kucharik and M. Shashkov. One-step hybrid remapping algorithm for multi-material arbitrary Lagrangian-Eulerian methods. *J. Comput. Phys.*, 231(7):2851–2864, 2012.

[31] M. Klima, M. Kucharik, and M. Shashkov. Combined swept region and intersection-based single-material remapping method. *Int. J. Numer. Meth. Fluids*, 85(6):363–382, 2017.

[32] M. Klima, M. Kucharik, and M. Shashkov. Local error analysis and comparison of the swept- and intersection-based remapping methods. *Commun. Comput. Phys.*, 21(2):526–558, 2017.

[33] R. Loubere and M. Shashkov. A subcell remapping method on staggered polygonal grids for arbitrary-Lagrangian-Eulerian methods. *J. Comput. Phys.*, 209(1):105–138, 2005.

[34] M. Kucharik and M. Shashkov. Conservative multi-material remap for staggered multi-material arbitrary Lagrangian-Eulerian methods. *J. Comput. Phys.*, 258:268–304, 2014.

[35] D. Bailey, M. Berndt, M. Kucharik, and M. Shashkov. Reduced-dissipation remapping of velocity in staggered arbitrary Lagrangian-Eulerian methods. *J. Comput. Appl. Math.*, 233(12):3148–3156, 2010.

[36] J. Velechovsky, M. Kucharik, R. Liska, M. Shashkov, and P. Vachal. Symmetry- and essentially-bound-preserving flux-corrected remapping of momentum in staggered ALE hydrodynamics. *J. Comput. Phys.*, 255:590–611, 2013.

[37] L. Spitzer and R. Harm. Transport phenomena in a completely ionized gas. *Phys. Rev.*, 89(5):977–981, 1953.

[38] R. M. More, K. Warren, D. Young, and G. Zimmerman. A new quotidian equation of state (QEOS) for hot dense matter. *Phys. Fluids*, 31(10):3059–3078, 1988.

[39] S. P. Lyon and J. D. Johnson. Sesame: The los alamos national laboratory equation of state database. Technical Report LA-UR-92-3407, Los Alamos National Laboratory, 1992.

[40] M. Zeman, M. Holec, and P. Vachal. HerEOS: A framework for consistent treatment of the equation of state in ALE hydrodynamics. *Comput. Math. Appl.*, 2019. In press.

[41] M. Shashkov and S. Steinberg. Solving diffusion equations with rough coefficients in rough grids. *J. Comput. Phys.*, 129(2):383–405, 1996.

[42] T. Kapin, M. Kucharik, J. Limpouch, and R. Liska. Hydrodynamic simulations of laser interactions with low-density foams. *Czech. J. Phys.*, 56(Supplement 2):B493–B499, 2006.

[43] R. Liska, M. Kuchařík, J. Limpouch, O. Renner, P. Váchal, L. Bednárik, and J. Velechovský. ALE methods for simulations of laser-produced plasmas. Book of Abstracts of FVCA VI, Finite Volumes for Comples Applications, Prague, June 6-10, 2011.

[44] J. R. Kamm. Evaluation of the sedov-von neumann-taylor blast wave solution. Technical Report LA-UR-00-6055, Los Alamos National Laboratory, 2000.

[45] J. Badziak, S. Borodziuk, T. Pisarczyk, T. Chodukowski, E. Krousky, K. Masek, J. Skala, J. Ullschmied, and Y.-J. Rhee. Highly efficient acceleration and collimation of high-density plasma using laser-induced cavity pressure. *Appl. Phys. Lett.*, 96(25):251502, 2010.

[46] J. Badziak, M. Kucharik, and R. Liska. Production of sub-gigabar pressures by a hyper-velocity impact in the collider using laser-induced cavity pressure acceleration. *Laser Part. Beams*, 35:619–630, 2017.

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Chapter 4: Numerical simulations for laser-produced plasmas

J. Limpouch

4. Numerical simulations for laser-produced plasmas

4.1 Introduction to simulations

Numerical simulation is an important tool in scientific research. It is often confused with the theory. However, it is a way of reality exploration additional to experiment and theory (see Fig.11.1). Sometimes, it can be difficult to extract the exact reason for particular effect from numerical simulations. And it is often difficult to find the dependence of results on experimental parameters. So analytical theory is important, even when it has to be simplified and even if analytical solution cannot be obtained.

There are several reasons why computer simulations are carried out. First, they can help scientists to understand consequences of the fundamental physics laws. Thus, conditions interesting either for basic science or for applications can be selected for a detailed exploration.

Second, computer simulations help in interpretation of experimental results. Often, experimental information is incomplete as measurement of some important quantities is too difficult or virtually impossible. For instance, interaction of femtosecond laser pulses with targets is an important field of study but not many information can be obtained with femtosecond resolution. Fastest X ray streak cameras have time resolution of approximately half-picosecond. However, the emitted signal is often too weak for reaching picosecond or even nanosecond temporal resolution, and thus only time integrated X-ray spectra are available in some experiments. Another example is the laser-induced particle acceleration. The properties of laser-accelerated particle beams can be accurately measured, however, the details of acceleration process are very difficult or practically impossible to detect. Thus, computer simulations are carried out and if the resulting beam properties match well with the experiment, the researchers believe that the simulations describe the acceleration process with reasonable accuracy.

Third, computer simulations help to design new experiments and predict their results. Experiments are often expensive and the space of experimental parameters is often too broad for random scanning. Thus, preliminary choice of experimental parameters is must. Results of numerical simulation also help in the selection of suitable diagnostics. The expected signal may be too weak for particular diagnostics to be detected. Or the signal may be too strong leading to the diagnostics saturation or it can even cause damage in the diagnostics equipment.

Numerical simulation consists of writing a computer code and using a computer for performing numerical experiment which shows evolution of some nonlinear system. Analytical solutions exist for most of linear problems, so numerical simulations are usually not necessary.

However, linear problems may serve for a partial validation of numerical codes. Most of problems in laser plasma interactions are highly nonlinear, so numerical simulations are inevitable for their solution, and for design and interpretation of experiments. Both the z-pinch and laser produced plasmas are magnetised. The z-pinch because of the current which flows in it and generates an azimuthal magnetic field and the laser produced plasmas due to currents generated in the blow off plasma. Waves propagating in a magnetised plasma are important because they determine the plasma characteristics and can also be used as diagnostic purposes. In order to understand the propagation of an electromagnetic wave in a magnetised plasma we start from Maxwell's equations:

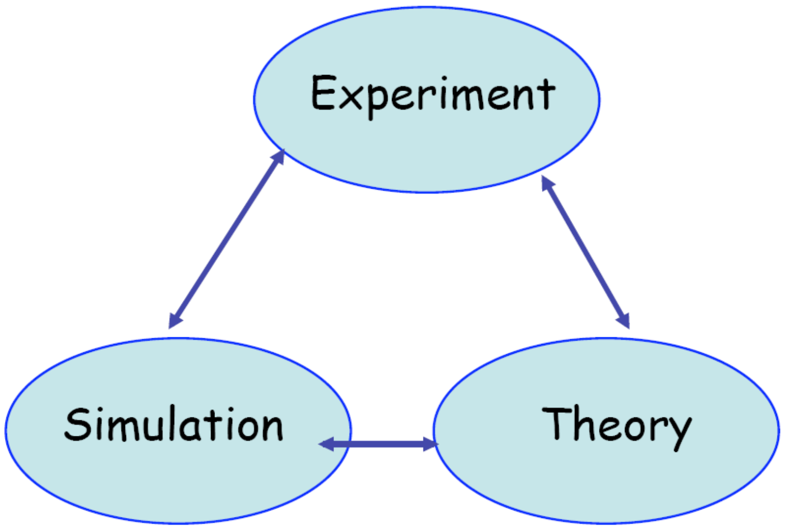


Figure 4.1 Ways of reality explorations – schematics (courtesy C. Ren, presentation [1] at 2009 HEDP Summer School)

4.2 Plasma description and types of numerical simulations

Plasma dynamics is usually described via kinetic or fluid models. Kinetic description is based on an equation for particle distribution function $f(\vec{r},\vec{p},t)$ that is solved either directly or indirectly. Kinetic description is usually used for weakly coupled plasma and it cannot be applied for non-evaporated part of dense (solid or liquid) targets. The advantage of kinetic description is a possibility to treat highly non-linear phenomena in laser-plasma interactions where important differences from Maxwellian distribution occur. However, kinetic description is computationally very demanding. Direct solving a kinetic equation (either collisionless Vlasov equation or collisional Boltzmann or Fokker-Planck equation) generally means solving partial differential equation in 7 variables, which is often above capabilities of present supercomputers. One reason of inefficiency of this approach is the necessity of solving the kinetic equation in the parts of configuration space where particles occur with very low probability (unoccupied phase space). This can be improved by sampling the configuration space with particles or macroparticles. Then, equations of motion in self-consistent electromagnetic fields are solved for each (macro)particle. Easy parallelization is an important advantage of this approach, and thus even 3D problems may be solved using big supercomputers. Sampling of the distribution function by finite particle number inevitably leads to a certain noise that must be kept in acceptable limits. Particle simulations are the main approach for simulations of interactions of intense ultrashort laser pulses with targets while for longer laser pulses they are usually used in limited spatial and temporal intervals for detailed studies of specific nonlinear effects.

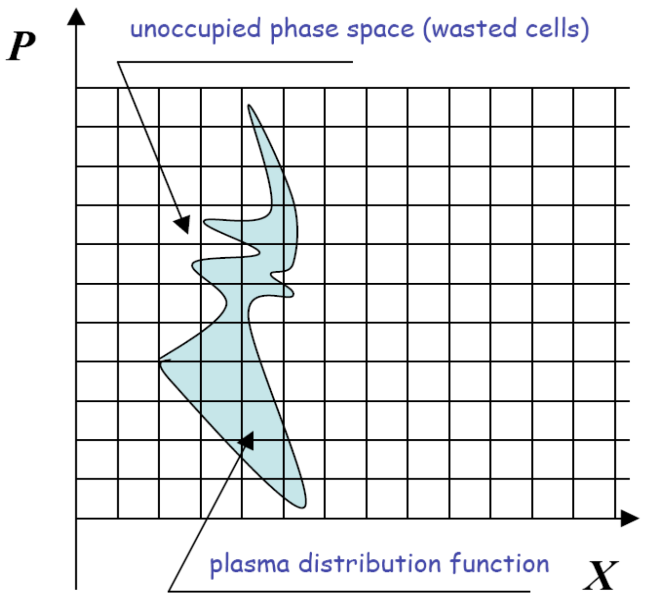

P
unoccupied phase space (wasted cells)
plasma distribution function
X

Figure 4.2 Configuration space with computational grid and occupied space [1]

Fluid approach describes plasma via first few moments of the distribution function. Usually, zeroth, first and second moments are used describing particle density, momentum and energy, so the fluid equations describe conservation of particle number, momentum and energy. The fluid description is incomplete as the details of distribution function are not taken into account. Consequently, some effects may be missing (as e.g. Landau damping) or may be described incorrectly. Fluid equations may be formulated separately for electron and ion fluid but this is rarely used in numerical simulations as it includes fast effects and thus it requires time step comparable with one over plasma frequency p. Fluid approach is usually used for simulations of large scale low frequency plasma processes where quasi-neutrality condition is met and thus, plasma may be described in one-fluid approximation. As the energy equilibration between electrons and ions (electron-ion relaxation) is slow compared to the momentum transfer, two temperature approximation is applied. If magnetic fields are important, magneto¬hydrodynamics (MHD) is employed. However, most MHD codes used in astrophysics cannot be applied directly as they omit magnetic field generating terms (like Biermann battery term due to crossed density and temperature gradients) important for laser-produced plasmas. As most of interaction experiments are carried without external magnetic field and quasi-static magnetic field generated during the interaction reaches the maximum value only after the laser pulse, it may be often omitted and ordinary hydrodynamics may be used. When high atomic number (high Z) materials are involved, radiative transfer must be solved together with hydrodynamics and such approach is called radiation hydrodynamics. Possibility to include cold dense (solid or liquid) material into fluid description is its important advantage and thus global simulations of nanosecond laser-target interactions are conducted via fluid approach. Fluid codes are also used for calculations of initial density profile for kinetic simulations of high intensity femtosecond interactions as the target is usually affected before the arrival of the main pulse due to insufficient contrast of the intense ultrashort laser pulse. Lower computational demands of fluid approach allow modelling of greater temporal intervals and spatial regions than in the kinetic approach. On the other hand, treatment of many non-linear processes can be included only phenomenologically, based on coefficients taken from theory or from kinetic simulations. In the fluid approach, suitable choice of equation of state (EoS) that links plasma pressure and internal energy to temperature and density is important. The mean ion charge must be calculated either via EoS or separately. Detailed atomic model may be included in the fluid code or it may be used as a post-processor.

Certain data for simulation of plasma dynamics may be prepared by pre-processor codes. The results of codes calculating plasma dynamics may be post-processed in order to carry out direct comparison with experimental diagnostics or to assess possibility to use laser-produced plasmas as a radiation or particle source for applications.

Detailed atomic physics of plasmas are usually modelled by specialized codes. The task may be split into 2 parts. First, excitation levels and transition rates of ions are calculated by solving the equations of quantum mechanics (e.g. Hartee-Fock equation with configuration interaction and relativistic corrections). This is very difficult task for ions of heavy elements with many bound electrons and if possible it is corrected with the help of experimental data for the spectral line energies. In the second part, collision-radiative (CR) model is solved to find the populations of the energy states and the emission (or absorption) spectra. The model can be stationary for particular plasma parameters or time-dependent. Plasma may be assumed either optically thin or radiative transfer must be included in a certain approximation (e.g. escape factors).

4.3. Kinetic methods

Methods describing plasma through distribution function may be divided into 2 basic types – particle methods that sample the distribution function and methods that directly solve a partial differential equation for the distribution function. Particle methods are easier to code up and analyse, they are more robust and economical, but they are noisier. There are implemented also in 3D where direct solvers of kinetic equations are impractical. Direct solvers of kinetic equation are sometimes preferred when the studied effect is caused by a small group of particles (e.g. particles at high energy end of the distribution).

4.3.1. Particle methods

Particle simulation techniques attempt to model many-body systems by solving the equations of motion of a set of particles. Tracking particle trajectories enables us to explore physical effects which are inaccessible to other modelling techniques. The method employs the fundamental equations without much approximation, allowing it to retain most of the physics. The algorithm consists of (1) loading of the initial particle positions and velocities, (2) calculating the force on each particle and (3) solving the equation of motion for each particle.

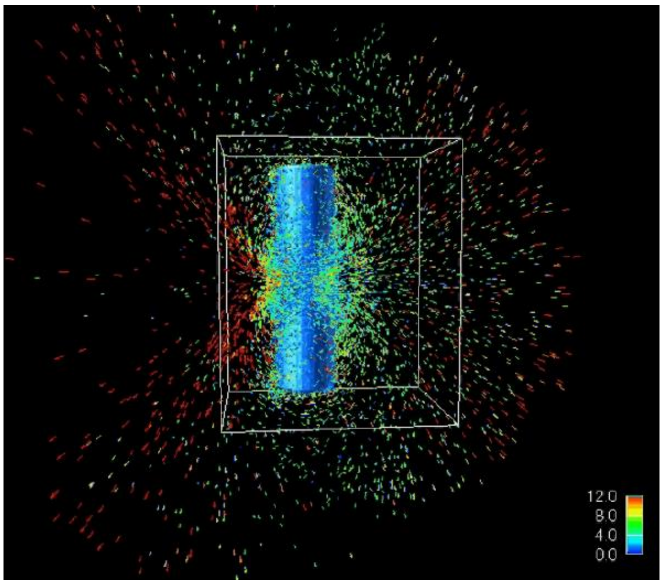


Figure 4.3 Electron distribution 50 fs after laser interaction with 4 m wire calculated via parallel 3D tree code PEPC.(P. Gibbon).

The simplest approach is the particle-particle method. It treats binary interaction between all particle pairs. The basic limitation is the number of arithmetic operations required in the calculation of forces as it scales as N^2 where N is the number of particles. Thus, this approach is viable only for a small number of particles $N < 10^6$ [1]. Thus, only some microscopic processes may be modelled by this approach.

One clearly needs to reduce the scaling of the operation count below N^2. There are two approaches how to decrease the operation count (1) particle-cluster method, so called tree code and (2) particle-mesh method, so called particle-in-cell code. The second method is faster and more popular while tree codes are slightly slower but less noisy.

Tree codes [2]

Tree codes treat near interactions in the same way as particle-particle codes. However, the distant particles are grouped and the potential of a group of distant particles is approximated by a low-order multipole expansion. Operation count in the force evaluation is below $N \times \log(N)$. Tree codes need more CPU time than particle-mesh codes, but they are favoured in systems with large density contrast. They also do better jobs in resolving small scale features of the solution and they are less noisy than the particle-mesh methods. However, most of existing tree codes are electrostatic as it is more difficult to treat laser field in a tree code. Nevertheless, 3D parallel electromagnetic tree code PEPC has been developed by Paul Gibbon and it is capable to perform realistic simulation of ultrashort pulse with mass-limited target as shown in Fig. 11.3.

Particle in-cell codes [3]

In particle-in-cell (PIC) codes, numerical mesh is added in order to compute the forces acting on the model particles. The basic cycle of PIC code consists of 4 steps depicted in Fig.11.4. (1) The particle positions are interpolated to the grid points where charges and currents are computed. (2) Maxwell's equations are solved on the computational grid and (3) the fields are then interpolated to the particle positions to compute forces acting on particles. (4) particle equations of motion are solved to get particle positions and velocities in the new time step.

The particle push must be usually supplemented with boundary conditions for particles. Particle collisions may be added after particle push; one option is to select colliding particles from particles within one cell randomly via a Monte Carlo algorithm [4] and vary their momentum according to the collision differential cross-section. Optical field ionization can be also added as well as other atomic processes. However, the addition of collisions leads to a very significant increase in the computational time. Thus, collisionless PIC simulations are preferred unless the impact of collisions is very significant.

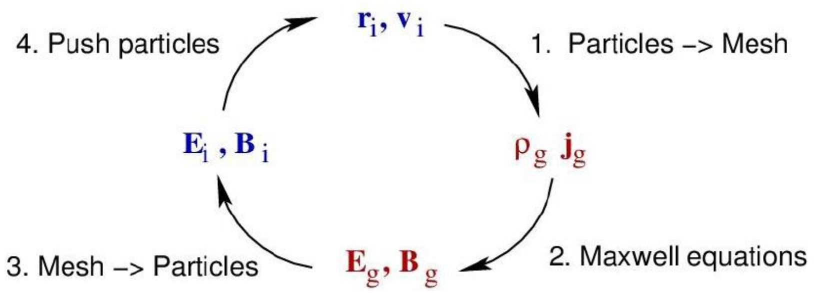


Figure 4.4 Schematics of basic cycle of a PIC code.

The number of floating point operations in PIC codes scales as $\alpha N + \beta N_g \ln N_g + \gamma N_g$, where N_g is the number of grid points and α, β, γ are constants. If the number N of particles is significantly larger than N_g, then most computations are needed for particle push that is easy to parallelize efficiently. As an example, 10000 time steps in simulation with 10^8 particles in 64×64×64 cells can be carried out in 3 seconds on 10 Tflop/s computer [1].

PIC method is basically designated for modelling of systems, where close neighbors contribute little to the force on a particle which is dominated by the sum of its interactions with distant particles, i.e. collective interactions dominate over binary interactions. Such systems are called weakly correlated or weakly collisional and ideal plasma is a typical representative of these systems. Basic PIC model treats these systems in collisionless approximation, while the impact of binary collisions may be taken into account via additional algorithm. PIC method cannot be used in strongly correlated systems like for instance solids. In plasmas, binary interactions are effective only for distances less than Debye length while at larger distances collective interactions dominate. In the PIC method, macroparticles represent a cloud of particles occupying a small area in the configuration space. Spatial dimension of macroparticle is assumed equal to the grid cell. When the grid cell dimensions are taken equal to the Debye length, binary interactions are effectively omitted in the PIC model.

PIC method may be performed in 1, 2 and 3 dimensions, where the macroparticles have shape of slabs, rods and cubes, respectively. Additional velocity components may be taken into account leading to currents in the directions normal to the spatial ones. Thus, 1D2V, 1D3V and 2D3V simulations are also used. While a standard PC is sufficient for typical 1D simulations, powerful workstations or small clusters are typically used for 2D simulations. 3D simulations need massively parallel computing and they generate extensive data sets that are laborious for processing, visualization and interpretation.

Equations of particle motions have to be solved by a fast, reasonably accurate and stable method. The leap-frog method is very popular. In this method, positions and forces are

$$\frac{\mathrm{d}\vec{r}_i}{\mathrm{d}t} = \vec{\mathrm{v}}_i \qquad \frac{\mathrm{d}\vec{\mathrm{v}}_i}{\mathrm{d}t} = \frac{\vec{F}_i}{m_i} \quad \Rightarrow \quad \frac{\vec{r}_i^{\,n} - \vec{r}_i^{\,n-1}}{\Delta t} = \vec{\mathrm{v}}_i^{\,n-1/2} \qquad \frac{\vec{\mathrm{v}}_i^{\,n+1/2} - \vec{\mathrm{v}}_i^{\,n-1/2}}{\Delta t} = \frac{\vec{F}_i^{\,n}}{m_i} \tag{11.1}$$

The error caused by the substitution of the derivatives by the differences is called truncation error. For the leap-frog method, the truncation error is proportional to Δt^2, so the method is of the second order of accuracy. Leap-frog is an explicit method for which the maximum time step is limited by the requirement of stability which means that the total error must not grow in time. The fastest mode limiting the time step is the plasma oscillation when electrons are oscillating with frequency ω_p in respect to ions due to the electrostatic force restoring neutrality. The force acting on electron is $F_i / m_i = -\omega_p^2 x_i$. When restoring force is inserted into (1) and the electron coordinate and velocity in time instant $n\Delta t$ are proportional to $\exp(i\omega n\Delta t)$, the eigenfrequency ω of difference equations is given by the following dispersion equation

$$\sin^2\left(\frac{\omega\Delta t}{2}\right) = \left(\frac{\omega_p \Delta t}{2}\right)^2 .$$

Thus for $\Delta t > 2/\omega_p$, the solution for ω is not real and consequently, the amplitude of plasma oscillations in the numerical scheme grows in time. For shorter time steps $\Delta t < 2/\omega_p$, the leap-frog method is stable and the difference between ω and ω_p leads to a phase error that decreases with decreasing time step Δt. Implicit schemes allow longer time steps, but the computation of the quantities at the next temporal instant is more time consuming and coding is more difficult especially for laser plasma interactions. Thus, most popular codes presently use explicit algorithms.

The deviation of conservation laws caused by the truncation error should be minimized. As the algorithms usually cannot be made fully conservative, unphysical growth of temperature, known as numerical heating, must be kept at negligible values during the whole simulation run.

After the particle push, charge and current must be assigned to a grid point. Once we introduce the grid, we can no longer view the particles as point particles, this leads naturally to the idea of a finite sized particle. Then, instead of charge assignment to the nearest grid point, it is natural to treat macroparticle as a cloud of particles and split the charge to the near grid points (cloud-in-cell). More elaborate smoother charge distributions may be used to suppress the noise. However, such schemes may become complicated and inefficient in more dimensions, and especially in 3D.

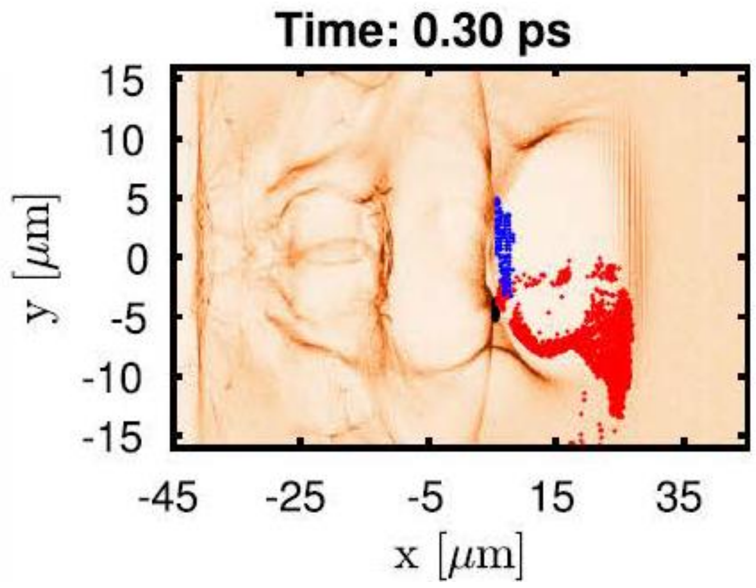


Figure 4.5 2D3V simulation of electron acceleration (optical injection by orthogonally crossing laser pulses) [5]. Electric field plotted in grey, main laser pulse is at x ≈ 25 μm.

Maxwell's equations are usually solved by a finite difference time domain (FTDT) method. In electromagnetic codes, the equations for curls of electric and magnetic field are directly solved. Though the equations for divergences are automatically fulfilled in the differential equations, there has to be an extra care to meet this condition in their differential analogue. Boundary conditions may be either periodic or reflecting or open. Additional perfectly matched layer is used to eliminate the reflection of outgoing electromagnetic waves.

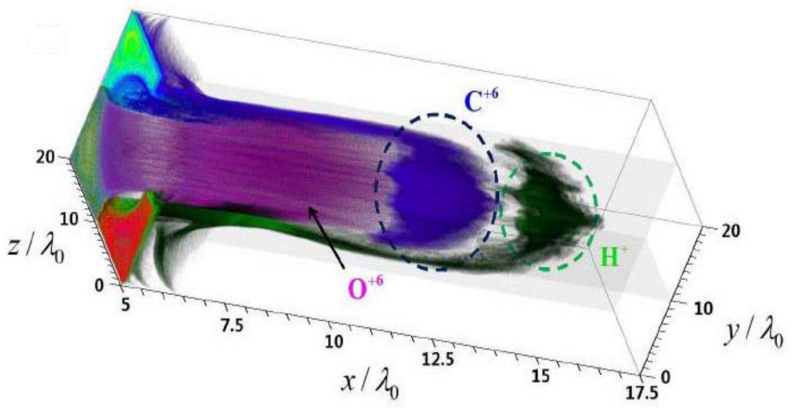


Figure 4.6 2D3V 3D simulation of interaction of circularly polarized laser of a_0=30 and duration 12 period T_0 incident from left on solid plastic foil 0.08 λ_0 thick. Ion density at 35 T_0 [6].

An example of PIC simulation results in Fig.11.5 shows one advantage of particle simulations – the possibility of particle tracking. Electrons denoted by the blue color were injected by the crossing laser beam; electrons pre-accelerated by drive beam are red and electrons induced by self- injection are black. The particle positions are overlaid on grey electric field, main drive laser pulse propagating to the right is at the right side of the figure and laser wake field propagates behind it. In Fig. 11.6, a result of 3D simulation of ion acceleration by an intense ultrashort circularly polarized laser pulde is presented. The density distributions of protons (cyan), carbon ions (blue), and oxygen ions (purple) are plotted after the interaction at t = 35 T_0. The circles show the proton bunch and carbon ion bunch, respectively.

4.3.2 Solving kinetic equations

Vlasov equation

Vlasov simulations solve similar problems as PIC codes. However, they are computationally more demanding. Most often 1D simulations are carried out, though 2D simulations are also performed at present. One advantage is the absence of noise. Vlasov simulations are preferred when a small number of particles at the distribution tail is essential. Vlasov simulations may distinguish small scale structures in the configuration space. The solution of Vlasov equation evolves to the formation of very small structures that are dispersed in plasma even by a small number of collisions. Thus, a small number of artificial or physical collisions is usually introduced to suppress this behavior. As an example, the electron phase space of relativistic KEEN wave obtained in Vlasov simulations of stimulated Raman scattering is plotted in Fig. 11.7.

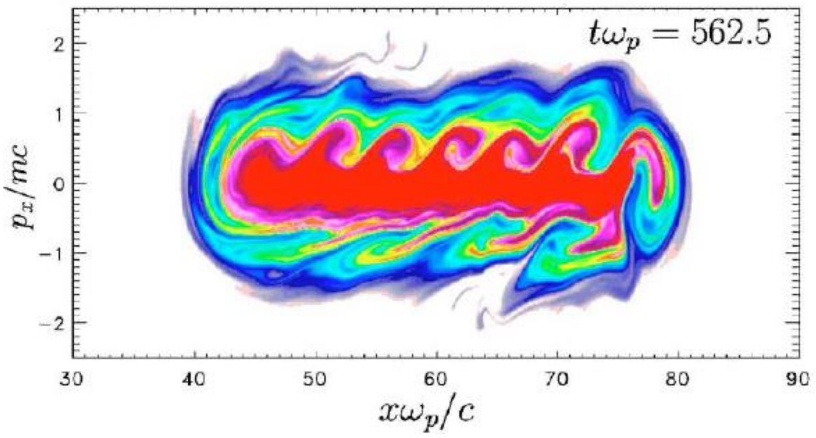


Figure 4.7 Electron phase space of relativistic KEEN wave in Vlasov simulations [7].

Collisional kinetic simulations

Kinetic equation with collisions is solved for the investigation of processes where collisions are essential and the deviations of particle distribution from Maxwellian distribution are significant. Typical studied effects are the electron non-local heat transport between critical and ablation surface, collisional (inverse bremsstrahlung) laser absorption and its impact on the electron distribution and also the impact of collisional atomic processes. Overdense plasma between critical and ablation surface is usually highly collisional and thus, PIC simulation are not efficient there.
Various collisional terms may be used, the most popular is the Fokker-Planck term, which is simpler than Boltzmann and Lenard-Balescu terms that are also sometimes used. Self-generated electric field may be calculated from the Poisson equation (Vlasov-Fokker-Planck code) or it may be obtained from the condition of quasi-neutrality.

The distribution function may be expressed via a series of spherical harmonics that are eigenfunctions of the collision operator, as follows

$$f(\vec{r},\vec{\mathrm{v}},t) = f_0(\vec{r},\mathrm{v},t) + \frac{\vec{\mathrm{v}}}{\mathrm{v}}\vec{f}_1(\vec{r},\mathrm{v},t) + \left(\frac{\mathrm{v}_i \mathrm{v}_j}{\mathrm{v}^2} - \frac{1}{3}\delta_{ij}\right) f_{2ij}(\vec{r},\mathrm{v},t) + ... \overset{1D}{=} \sum_{l=0}^{\infty} f_k(x,\mathrm{v},t)\, P_k(\mu)$$

,

where μ=v$_x$/v. In the simplest case only 2 terms are retained.

4.4 Fluid simulations

Often, one does not need the detailed knowledge of the particle distribution functions. The description is simplified in the fluid (also called hydrodynamic) models where the moments of the distribution functions are used. Fluid description is more accurate in dense plasmas where the collisions decrease deviations from the Maxwellian distribution. The basic moments are the density (0th moment), the average velocity (1st moment) and the internal kinetic energy (2nd moment) that are defined, as follows

$$n_a(\vec{r},t) = \int f_a\, \mathrm{d}\vec{\mathrm{v}} \qquad \vec{w}_a(\vec{r},t) = \frac{1}{n_a}\int f_a \vec{\mathrm{v}}\, \mathrm{d}\vec{\mathrm{v}} \qquad \varepsilon_{ka}(\vec{r},t) = \frac{m_a}{2}\int f_a\, (\mathrm{v} - \vec{w}_a)^2\, \mathrm{d}\vec{\mathrm{v}}$$

In the fluid description, one typically solves a system of conservation laws for the particle number, momentum and energy. These laws are partial differential equations (PDEs) in space and time, but not in velocity. Thus, fluid approach is less computationally demanding than the kinetic description and it can be also used for strongly coupled systems. Consequently, it is applied for global simulations of laser interactions with dense (solid, liquid) targets. On the other hand, non-linear processes in laser-target interactions can be treated only in a phenomenological way, many processes have to be included using either analytical models or an experience from the kinetic simulations.

Fluid models are usually used for simulations of processes slow compared to plasma frequency ω_p. In principle, two fluid (electron and ion) models can describe also fast processes, however, they cannot be used for dense systems and they omit some aspects of particle-wave interactions (e.g. Landau damping), so PIC simulations are more appropriate for these processes. For processes slow compared to ω_p quasi-neutrality may be assumed and one-fluid approximation is sufficient. When a quasi-static magnetic field is important, magnetohydrodynamics (MHD) must be used. In MHD description of laser-target interactions, source terms like source due to crossed gradients of density and temperature (Biermann battery term) have to be retained though they are omitted in most typical astrophysical codes. In most situations, the impact of quasi-static magnetic field is small during the time when laser is interacting with the target and thus, ordinary hydrodynamics is used. As the energy equilibration between electrons and ions is slow due to ratio of ion to electron mass, separate equations are often used for electron and ion temperature. For high-Z targets, a significant part of energy is contained in radiation, and thus, radiation hydrodynamics is used.

Fluid dynamics is described by a system of conservation laws represented by equations of hyperbolic type. It is surprisingly difficult to develop suitable difference scheme for hyper-bolic PDEs. For enlightening this problem, we shall use the simplest hyperbolic equation, the advection equation

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \qquad \text{with initial condition} \quad u(0,x) = u_0(x)$$

The solution is analytical $u(t,x) = u_0(x - a\,t)$. The advection equation is ideal for testing of numerical methods as it is simple (linear) with the known analytic solution. From various combinations of finite differences, the natural scheme is the FTCS (forward time central space)

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^n - u_{i-1}^n}{\Delta x} = 0 \quad \Rightarrow \quad u_i^{n+1} = u_i^n + a\frac{\Delta t}{2\,\Delta x}\left(u_{i+1}^n - u_{i-1}^n\right)$$

The basic requirement for a difference scheme is its stability, the total error should not grow in time. Stability may depend on the Courant-Friedrichs-Levy number *CFL = aΔt/Δx*. Unfortunately, FTCS scheme for advection equation is unstable for any CFL number. Instead, it is possible to use 1st order accurate Lax-Friedrichs scheme or 2nd order accurate Lax-Wendroff scheme that are stable for $-1 \le \mathrm{CFL} \le 1$

$$\frac{u_i^{n+1} - \left(u_{i-1}^n + u_{i+1}^n\right)/2}{\Delta t} + a\frac{u_{i+1}^n - u_{i-1}^n}{\Delta x} = 0 \qquad \text{Lax-Friedrichs scheme}$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^n - u_{i-1}^n}{\Delta x} = \frac{a^2}{2}\frac{\Delta t}{\Delta x^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) \qquad \text{Lax-Wendroff scheme}$$

The solution of propagation with *a* = 1 of initial Gaussian pulse of height 1 and width 5 by the above methods is presented in Fig. 11.8. Unstable FTCS method is unusable, while the stable methods track the pulse motion correctly. Lax-Friedrichs scheme is diffusive, the pulse is broadened and lowered. On the other hand, the dispersive Lax-Wendroff scheme leads to an oscillatory pulse shape. The calculated pulse shape can be partially improved by a suitable alternation of these two methods during successive time steps. The time step may be increased by using implicit schemes, for example FTCS with spatial derivative taken in *n+1* time instant is stable for any CFL number. However, no explicit formula exists for the next step values and non-linear equation system has to solved in each time step.

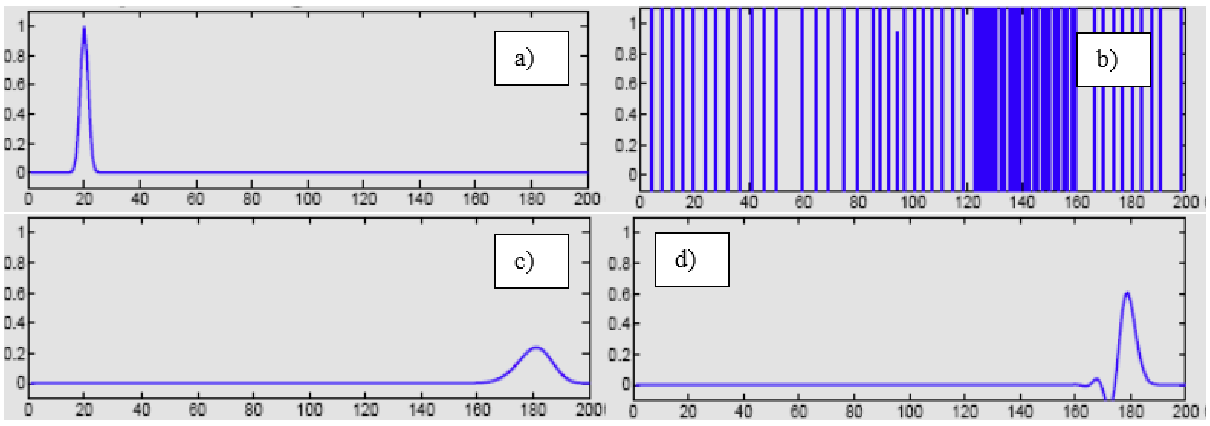


Figure 4.8 Advection equation (a) Initial pulse. Solution at T=160 via (b) FTCS scheme (c) Lax-Friedrichs scheme and (d) Lax-Wendroff scheme.

In the simplest case, the fluid equations are the Euler equations for the conservation of mass, momentum and energy

$$\frac{\partial \rho}{\partial t} + \operatorname{div}\left(\rho \vec{w}\right) = 0$$

$$\frac{\partial\left(\rho \vec{w}\right)}{\partial t} + \operatorname{div}\left(\rho \vec{w} \otimes \vec{w}\right) + \operatorname{grad}\left(p\right) = 0$$

$$\frac{\partial \mathcal{E}}{\partial t} + \operatorname{div}\left[\vec{w}\left(\mathcal{E} + p\right)\right] = 0,$$

where symbol $\otimes$ denotes tensor multiplication, ρ is the density, $\vec{w}$ is the velocity vector, $\mathcal{E}$ is the total energy density and p is the pressure given by the equation of state $p = p(\rho,\varepsilon)$, where the specific internal energy $\varepsilon = \mathcal{E} / \rho - \left|\vec{w}\right|^2 / 2$.

When a computational mesh fixed in space is chosen for solving fluid equations, one speaks about the Eulerian frame. Fluid moves through the static computational frame in the form of fluxes. Methods for solving fluid equations in the Eulerian frame are well understood (Lax-Friedrichs and Lax-Wendroff schemes are typical Eulerian methods) and easy to implement. The theory of their stability and properties is well advanced. However, the fixed computational mesh is not suitable for tasks with enormous expansions (or compressions) of the computational domain, which is typical for laser interactions with solid targets. Also, tracking plasma-vacuum boundary is a problem in the Eulerian frame and a low-density ambient plasma is usually used instead of vacuum.

Alternatively, the Lagrangian frame may be used where the coordinates are attached to the moving fluid. The positions of the nodes move together with the fluid, so there is no mass flux between the adjacent cells and the mass of a cell is constant in time. Time derivative in the Lagrangian frame is the total derivative $\mathrm{d}/\mathrm{d}t = \partial / \partial t + \vec{w}\nabla$. Viscosity has to be added in case of a cell compression (typically due to an incident shock wave). The disadvantage of Lagrangian approach is the possibility of deformations of computational cells, e.g. due to shear. This leads to amplification of the numerical errors and eventually, the simulation may fail when non-convex or negative volume cells appear. A typical result of a Lagrangian

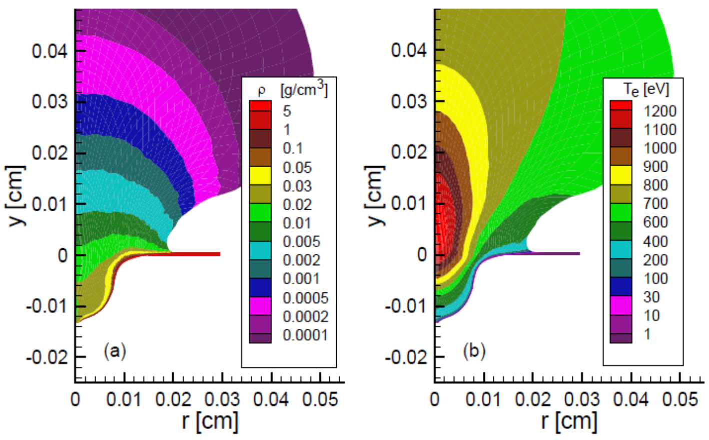


Figure 4.9 Density and temperature profiles 0.3 ns after maximum of 0.4 ns FWHM laser pulse of λ = 439 nm and energy 58 J incident from the top on 5 μm-thick Al foil [8].

simulation is presented in Fig. 11.9. Laser is incident from the above normally on 5 μm-thick Al foil positioned at y=0, laser spot radius is 50 μm. Laser-induced plasma corona expands up to ~700 μm, so the simulation area is increased more than 100× in *y* direction. The critical density ρ_{cr} ~ 0.02 g/cm^3 is more than 100 times lower than the density of solid Al and in Lagrangian approach it is easy to choose cell dimensions that ensure a sufficient resolution in the important area of critical surface neighbourhood.

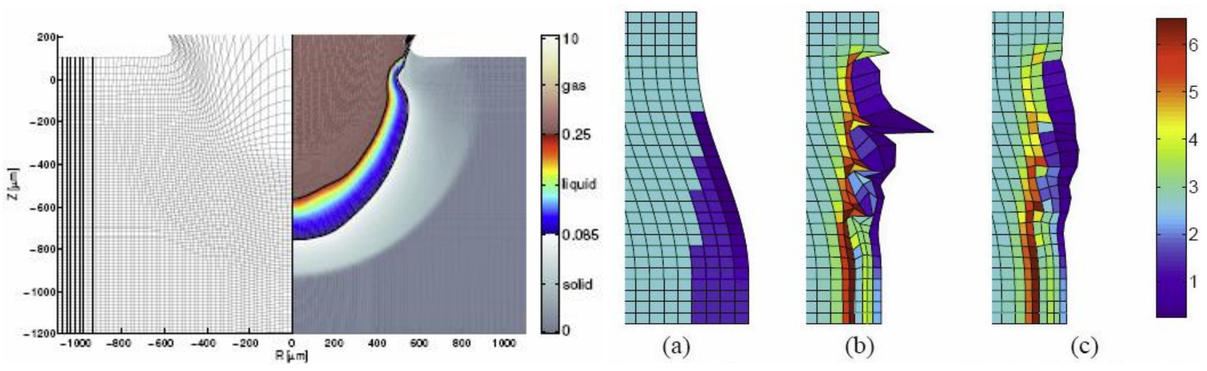


Figure 4.10 (left panel) Computational grid and target temperature in eV 80 ns after the impact of laser accelerated disk on bulk target. (right panel) Density colormap at the critical area (a) initial grid (b) very distorted Lagrangian grid after 0.5 ns including non-convex cells and (c) ALE grid still smooth after 80 ns [9].

The solution for problems of tangling of Lagrangian cells is the Arbitrary Lagrangian-Eulerian (ALE) method that combines Lagrangian and Eulerian approach. First, several or many Lagrangian time steps are computed. After a given number of time steps or when a certain deformation of Lagrangian grid is detected, the Eulerian part is started. In this part, mesh rezoning untangles and improves the computational mesh. Then, remapping conservatively interpolates the conservative quantities from the old to the new mesh. Remapping allows mass fluxes between the computational cells. After remapping, the code goes back to a sequence of Lagrangian steps. The ALE method combines the advantages of Eulerian and Lagrangian approaches. The grid moves together with the fluid, but the Eulerian part keeps it smooth. As an example, the result of 2D cylindrical ALE simulation of the impact of laser-accelerated thin disk on a bulk target is presented in Fig. 11.10. For purely Lagrangian simulation, critical region is at the edge of the impacting disk. The computational grid tangles here and the simulation fails as early as 0.5 ns after the disk impact due to the formation of non-convex cells. As the simulation aim is to compare the volume of the crater created at the bulk surface, the critical region is not very important for the main aim of the simulation. However, Lagrangian simulation cannot proceed any more. On the other hand, the ALE simulation overcomes this problem and the computational mesh is kept smooth for the whole time 80 ns of the simulation.

The system of the Euler equations has to be supplemented by additional terms. Laser propagation and absorption must be calculated. A variant of ray-tracing model is most frequently applied for the laser propagation in an underdense plasma where collisional (inverse bremsstrahlung) absorption is taken into account. Laser absorption and reflection in the critical surface neighbourhood may be calculated by solving Maxwell's equations, however, this is often substituted by crude phenomenological models that use simplified analytical formulas for resonance absorption. As energy absorbed due to resonance absorption and due to nonlinear parametric instabilities goes basically into a relatively small group of fast electrons that transport energy far from the absorption region, simplified model of energy transport by fast electrons is included in some fluid codes. Electron heat conduction from the critical to the ablation surface is mostly non-local due to large temperature gradients. The classical heat flux proportional to minus gradient of electron temperature has been traditionally restricted by the free streaming heat flux with arbitrary coefficient *f* called flux limiter (usually set in range 0.05 – 0.1). Classical Spitzer-Harm thermal conductivity is valid for ideal plasmas, but it severely underestimates heat conductivity at low temperatures, thus an ad hoc interpolation between low and high temperatures is usually used [10]. Recently, heat flux is calculated non-locally using convolution or multi-group approach [11]. Multi-group approach [12] is also used for radiative energy transfer, where a separate difficult problem is the calculation of radiative opacities [13]. An equation of state (EOS) is needed to connect to the pressure and the temperature with the density and the internal energy. Ideal gas EOS is most simple but it is not realistic at high densities and low temperatures, thus either some simplified analytical models [14] or tabulated equation of state [15] must be used.

4.5 Summary

This session has been devoted to an introduction to numerical simulations of laser-target interactions. It has been focused mainly to the description of plasma dynamics. There are two main approaches – kinetic description via particle distribution function, and fluid description via moments of the distribution function.

Kinetic models are performed either by direct solving of kinetic equation or via sampling of the distribution function by macroparticles in particle codes. Particle codes are often preferred as they are faster and they may be parallelized efficiently. Particle codes have to avoid computationally demanding particle-particle interaction. This is performed either by particle-mesh interaction in PIC (particle-in-cell) codes or by particle cluster interaction in tree codes. PIC codes are very popular for modelling of interactions of intense ultrashort laser

pulses where they are able to model strongly non-linear effects in weakly coupled plasmas. 3D simulations of interactions are feasible on supercomputers. Macroparticles in PIC codes are clouds of particles with dimensions equal to one or a few grid cells. PIC approach describes collective interaction of particles through macroscopic electromagnetic fields. Microscopic binary interactions may be added if necessary via additional (e.g. Monte Carlo) algorithm. PIC approach cannot be used for strongly bound systems where interactions with nearby particles dominate. PIC simulations inevitably include certain noise not only due to sampling by finite number of macroparticles, but also due to finite grid cells. The noise can be reduced by using a tree code, but these codes are rare, slower and more difficult to create.
Direct solving of the collisionless Vlasov equation is preferred for effects where a few particles dominate. However, Vlasov simulations are more computationally demanding and cannot be performed in 3D at present supercomputers. Direct solving of collisional kinetic equation is preferred for situations in weakly coupled plasmas when the binary collisions play an important role and deviations from Maxwellian particle distribution are significant. For instance, such codes model accurately the electron heat transport in the highly collisional area between the critical surface and the ablation surface of solid targets.

Fluid codes use simplified description of plasma via density, average velocity and temperature. They are well suited for global modelling of laser-target interactions as they can also include dense cold (e.g. solid) areas. They usually include processes slow compared to the plasma frequency ω_p, and thus quasi-neutrality may be assumed. Consequently, one fluid hydrodynamic or magnetohydrodynamic description is used. On the other hand, non-linear processes can be treated only phenomenologically. Numerical algorithms for simulations in a grid fixed in the space (Eulerian frame) are well developed and understood. However, when enormous expansions (or compressions) occur during simulations, Lagrangian frame is applied, where the nodes are moving together with the fluid. The problem of the Lagrangian approach is possible deformation of computational cells that leads to simulation failure. This is cured in Arbitrary Lagrangian-Eulerian (ALE) codes where mesh rezoning and remapping of conservative quantities is added to the Lagrangian approach.

References

[1] C. Ren, *Introduction to Particle-in-Cell Method in Plasma Simulations*, The 2009 HEDP Summer School, Univ. Rochester, http://hedpschool.lle.rochester.edu/1000_proc2009.php

[2] S. Pfalzner, P. Gibbon, *Many-Body Tree Methods in Physics*, Cambridge University Press, Cambridge 1997

[3] C.K. Birdsall, A.B. Langdon, *Plasma Physics via Computer Simulation*, McGraw-Hill, New York 1985

[4] T. Takizuka, H. Abe, J. Comput. Phys **25** (1977) 205

[5] V. Horny *et al*., Phys. Plasmas **24** (2017) 103125

[6] T. P. Yu *et al*., Opt. Express **21** (2013) 22558

[7] A. Ghizzo *et al*., Phys. Rev. E **74** (2006), 046407

[8] O. Renner *et al*., JQSRT **81** (2003), 385

[9] M. Kucharik *et al*., Czech. J. Phys. **56** (2006) B522

[10] K. Eidman *et al*., Phys. Rev. E **62** (2000) 1202

[11] G. Schurtz *et al*., Phys. Plasmas 7 (2000) 4238

[12] C. Orban *et al.*, *A Radiation-Hydrodynamics Code Comparison for Laser-Produced Plasmas: FLASH versus HYDRA and the Results of Validation Experiments*, Technical Report LLNL-JRNL-636375, 2013

[13] H. A. Scott, JQSRT **71** (2001) 689

[14] R.M. More, Phys. Fluids **31** (1988) 3059

[15] S. P. Lyon, J. D. Johnson, *SESAME: the Los Alamos National Laboratory equation of state database*, Technical Report LA-UR-92-3407, LANL, Los Alamos, 1992.

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Chapter 5: Numerical modeling and simulations Lasers/Plasma

V. Dimitriou

HELLENIC
MEDITERRANEAN
UNIVERSITY

5. Numerical modeling and simulations Lasers/Plasma

Irradiation of solids with pulsed laser beams is a powerful method for materials excitation. Due to the great flexibility of the method, the large and monitored power and the ability of remote manipulation without any contact with the material, this method is rapidly expanding. Common applications of the method include laser surface melting, surface reinforcement by laser shock, surface coating and functionalization, physical and chemical measurements, assembling and dismantling (surface cleaning), nanopowder production and use. Pulsed laser's irradiation ability of broadband signal generation is also used in industrial applications like surface structure detection, compositions, geometry, roughness, plainness and elastic properties of metallic specimens' analysis [1–10].

The dynamic reaction of matter irradiated by a nanosecond laser pulse source depends on its thermo-physical properties, as well as on the laser pulse characteristics. To understand the complex physical phenomenon of this interaction, various analytical and numerical approximations have been developed. For most of the numerical approximations the Finite Element Method (FEM) is chosen to carry out the simulation of the multiphysics thermal-structural problem, because of its ability to predict the achievable temperature gradients and time-dependent displacements and stresses at multiple locations. FEM is not limited by the geometry of the solution domain nor the precision's variance neither the lack of smoothness of solution [11-21].

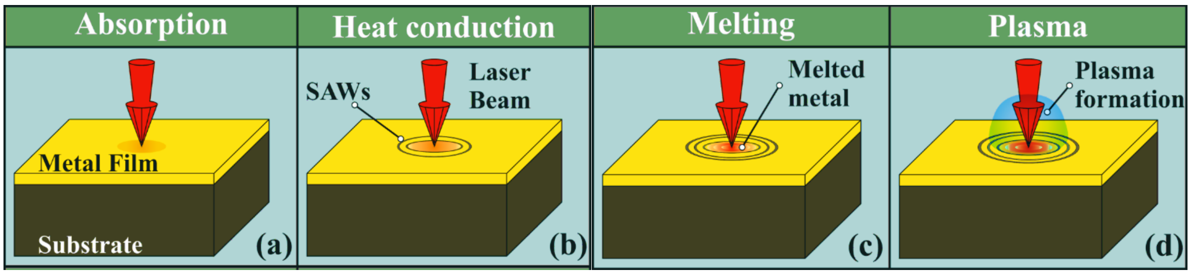


Figure 5.1 Phase change regimes: (a), (b) thermoelastic, (c) melting and (d) plasma, of pulsed laser film-substrate interaction [22].

The main regimes for pulsed laser irradiation of matter are the thermoelastic, melting and ablation (plasma) regime. In the thermoelastic regime, the area of the sample irradiated by the nanosecond laser pulse is rapidly heated by the absorption of the laser energy. The heating rate and the surface temperature are defined by absorption and reflection coefficients, by thermal conductivity, and by the specific heat of the metal. The sole form of heat transport is conduction within the metal (Fig. 9.1). The absorption of energy results in a rapid increase of temperature in the irradiated volume, which in turn causes a local rapid thermal expansion. The localized thermal expansion generates a stress field and ultrasonic surface acoustic waves (SAWs) that propagate in the target's material. SAWs provide valuable elastic information in the vicinity of surface over which the waves propagate, because the surface wave phase velocity is directly dependent on the elastic tensor of the material [23]. Acoustic surface waves are well suited for testing thin films, which are used extensively with applications covering various sectors of industrial activity [24,25]. For greater laser intensities the target surface temperature reaches its melting point (Fig. 1(c)). At this stage, the melted front penetrates the solid phase, while the thermal and optical properties of the target material change. For even greater laser intensities the target surface reaches its boiling point. The process of material removal from the target is called laser ablation. For incident laser intensities greater than the ablation threshold plasma is formed (see Fig. 1(d)) [16,26]. Laser plasma is composed of a large amount of electrons, ions and excited neutrals that absorb the laser light. For all regimes, the generation and propagation of SAWs depends on both the film and substrate thermo-physical (elastic) properties, as well as on the laser pulse characteristics. Studies by the help of FEM have been carried out on the behavior of thin films surface under nanosecond laser pulse excitation [16-21]. Xu et al. [17,18] studied the transient temperature and temperature gradient fields in coating-substrate systems as well as the surface normalized vertical displacements at different source-receiver distances and to the epicenter. Their work is focused on the thermoelastic regime. In [19] the laser ablation of titanium carbide with the aid of a two dimensional (2D) finite element model is simulated, based on the heat conduction equation and on the Hertz-Knudsen equation of vaporization. In [16] a 2D finite element model capable to predict temperature distribution and ablation depth by taking into account the absorption of laser radiation in plasma is developed. The work presented in Ref.'s [16] and [19] is focused on the pulsed laser ablation phenomenon and the ablation depth (crater's formation).

5.1 What is FEM?

5.1.1 A typical problem description

FEM is a numerical method originally developed as: a branch of Solid Mechanics. Nowadays FEM is a commonly used method for MULTIPHYSICS problems. The physical problems that science aims to solve may be categorized to: Structure analysis problems: a cantilever, a building, a bridge, etc.; Solid mechanics problems: a gear, an air wing, etc.; Dynamics: vibration of a bar, of a tower, an earthquake, etc.; Thermal analysis problems: heat conduction, radiation of a surface, etc.; Electrical analysis problems: electrical signal propagation, piezoelectric actuators, etc.; Biomechanics: human organs, bones, tissues, etc.; Fluid mechanics, Magnetic analysis... are some of the basic fields that numerical simulations may work for. The Multiphysics problems may include some or ALL of these branches of problems that may be mathematically described by the help of Differential Equations.

To explain the mathematical modeling philosophy of the FEM let's assume a solid with known Properties: materials & geometry as presented in Fig. 2 having known:

- Boundary: (2D)-The blue line, (3D)-The blue surface that is enclosing the geometry
- Solid: Interior: Surface (2D) or Volume (3D)
- Boundary conditions: Prescribed quantities (e.g. prescribed displacements, prescribed displacements, velocities, stresses, strains etc.)
- Loading conditions: Force, pressure, thermal, electro, magnetic loads

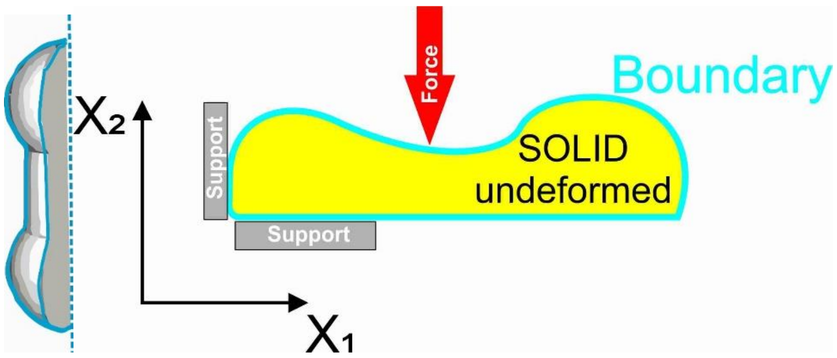


Figure 5.2 A solid imposed to an axial force.

The questions that has to be answered is: What will happen to a solid if a Force is applied to it? and Which will be the values of Displacements, Stresses, Strains? ... at each material point of the solid body [27, 28].

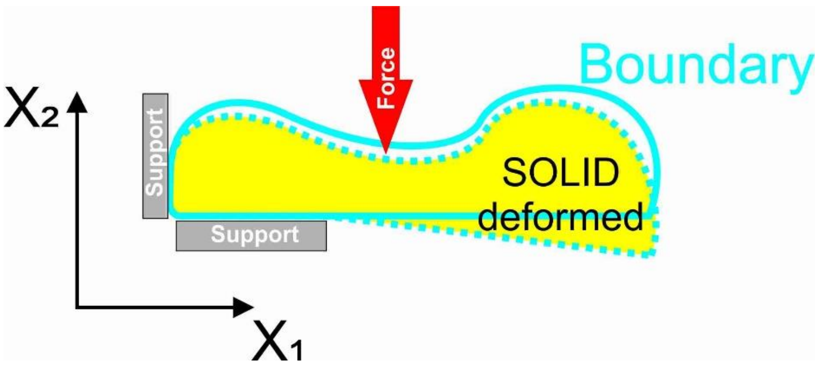


Figure 5.3 Deformed solid.

5.1.2 Mathematical formulation

For the mathematical formulation of the typical problem described in 9.2.1, the Equilibrium the Constitutive and the Strains (Kinematics) equations may be used:

Strains (kinematics):

$$e_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i}\right)$$

Constitutive

$$\sigma_{ij} = 2Ge_{ij} + \lambda e_{kk}\delta_{ij}$$

$$e_{ij} = \frac{1+\nu}{E}\sigma_{ij} - \frac{\nu}{E}\sigma_{kk}\delta_{ij}$$

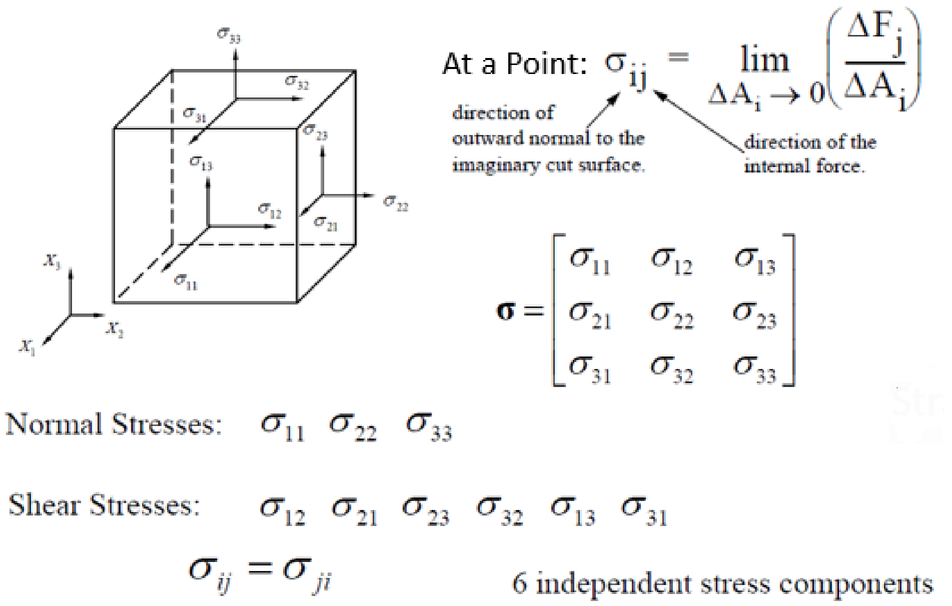
Equilibrium

$$\frac{\partial\sigma_{i1}}{\partial X_1} + \frac{\partial\sigma_{i2}}{\partial X_2} + \frac{\partial\sigma_{i3}}{\partial X_3} + f_i = 0$$

The determination of the values of the unknown variables: Displacements: u_1, u_2, u_3; Stresses: σ_{11}, σ_{22}, σ_{33}, σ_{12}, σ_{21}, σ_{23}; Strains: ε_{11}, ε_{22}, ε_{33}, ε_{12}, ε_{21}, ε_{23} must be provided by the solution of the above 15 equations of which the 9 are Partial Differential Equations (PDE's). At this point we may easily notice that an exact analytical solution is impossible to be found, and this consists the main difference of simulations to pure theory. FEM is a numerical simulation method that may be used to find the best approximate solution to the problem. Simulations are NOT theory.

5.1.3 Basic solid mechanics

Stress: is an internal quantity that has units of force per unit area. At a point Stress needs a magnitude and 2 directions to specify it. The sign of a stress component is determined from the direction of the internal force and the direction of the out-ward normal to the imaginary cut surface and the shear stress components are symmetric (see Fig.9.4).



σ_{33} σ_{32} σ_{31} σ_{23} σ_{13} σ_{12} σ_{21} σ_{22} σ_{11}
x_3 x_2 x_1
At a Point: $\sigma_{ij} = \lim_{\Delta A_i \to 0} \left(\frac{\Delta F_j}{\Delta A_i} \right)$
direction of outward normal to the imaginary cut surface.
direction of the internal force.
$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}$
Normal Stresses: σ_{11} σ_{22} σ_{33}
Shear Stresses: σ_{12} σ_{21} σ_{23} σ_{32} σ_{13} σ_{31}
$\sigma_{ij} = \sigma_{ji}$
6 independent stress components

Figure 5.4 Stresses.

Strain: is a measure of relative movement of two points on the body (deformation). Elongations are positive normal strains. Decrease from right angle results in positive shear strains. Small strains (ε < 0.01) can be calculated using just the deformation in the original direction of the line. Small strain results in a linear theory tensor normal strains equal to engineering normal strains and tensor shear strains equal to: ('engineering shear strains'/ 2), (see Fig. 9.5)

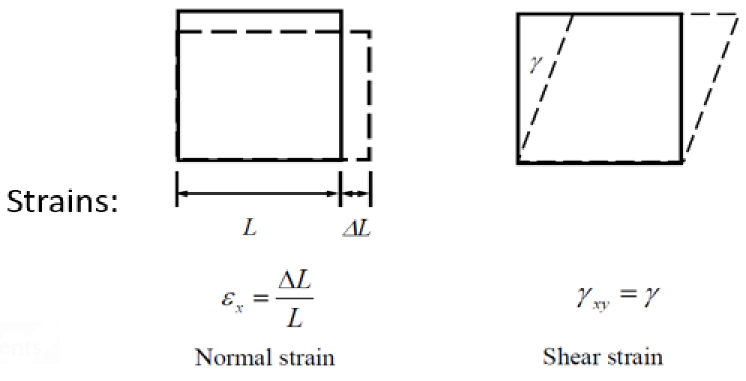


Figure 5.5. Strains.

For linear elasticity:

$$(x_1, x_2, x_3) \Leftrightarrow (x, y, z)$$
$$(u_1, u_2, u_3) \Leftrightarrow (u, v, w)$$

Strains (kinematics) : $\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$

$$i{=}j{=}1 \quad \varepsilon_{11} = \frac{\partial u_1}{\partial x_1} = \frac{\partial u}{\partial x} \qquad i{=}1,\, j{=}2 \quad \varepsilon_{12} = \frac{1}{2}\left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}\right)$$

$$i{=}j{=}2 \quad \varepsilon_{22} = \frac{\partial u_2}{\partial x_2} = \frac{\partial v}{\partial y} \qquad i{=}2,\, j{=}3 \quad \varepsilon_{23} = \frac{1}{2}\left(\frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2}\right)$$

$$i{=}j{=}3 \quad \varepsilon_{33} = \frac{\partial u_3}{\partial x_3} = \frac{\partial w}{\partial z} \qquad i{=}1,\, j{=}3 \quad \varepsilon_{13} = \frac{1}{2}\left(\frac{\partial u_1}{\partial x_3} + \frac{\partial u_3}{\partial x_1}\right)$$

Normal Strains   Shear Strains

$$\varepsilon_{11} = \frac{\partial u_1}{\partial x_1} = \frac{\partial u}{\partial x} \qquad 2\varepsilon_{12} = \gamma_{xy} = \frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$$

$$\varepsilon_{22} = \frac{\partial u_2}{\partial x_2} = \frac{\partial v}{\partial y} \qquad 2\varepsilon_{23} = \gamma_{yz} = \frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}$$

$$\varepsilon_{33} = \frac{\partial u_3}{\partial x_3} = \frac{\partial w}{\partial z} \qquad 2\varepsilon_{13} = \gamma_{xz} = \frac{\partial u_1}{\partial x_3} + \frac{\partial u_3}{\partial x_1} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}$$

5.2 FEM modeling concepts

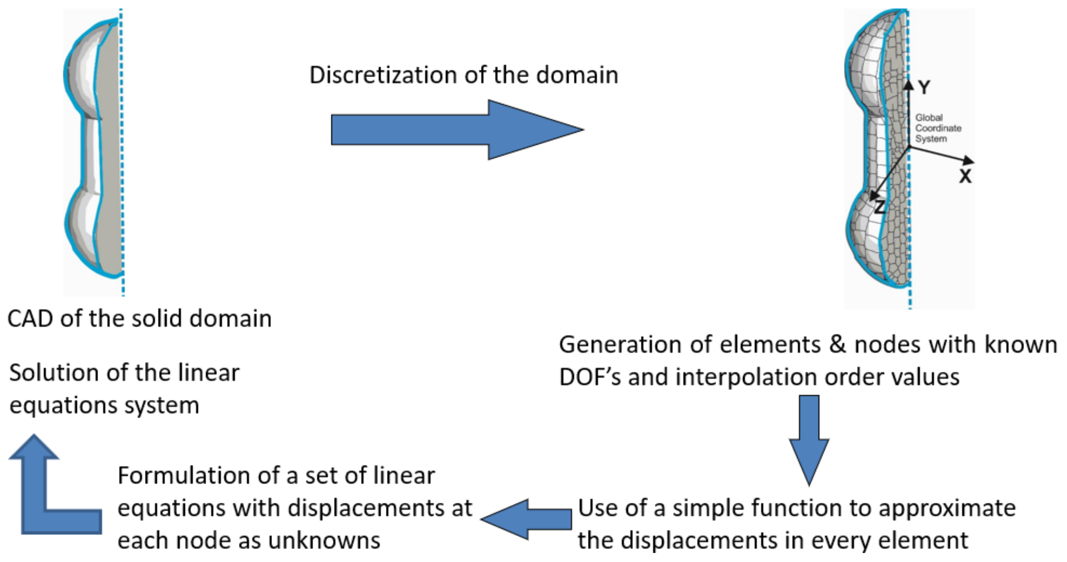
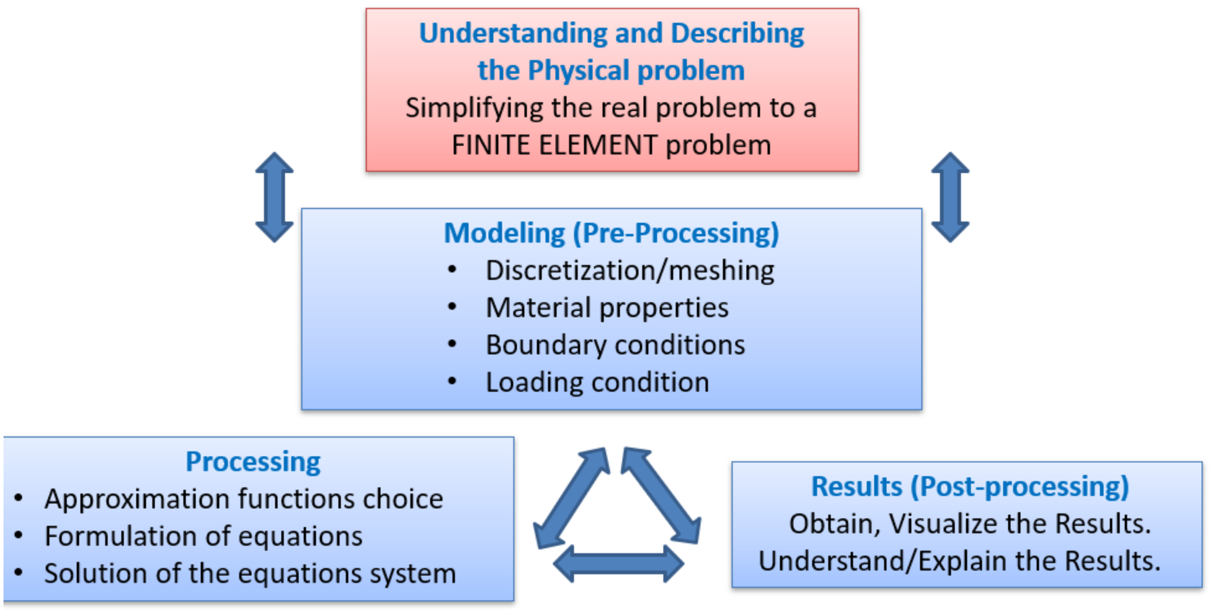

Discretization of the domain
Y
Global Coordinate System
X
Z
CAD of the solid domain
Generation of elements & nodes with known DOF's and interpolation order values
Solution of the linear equations system
Formulation of a set of linear equations with displacements at each node as unknowns
Use of a simple function to approximate the displacements in every element

Figure 5.6. The basic modeling steps of FEM.

The basic modeling steps of FEM are presented in Fig 9.6. The solid domain is geometrically represented by the help of any CAD system that is further discretized. The finite elements are generated by the help of the nodes having known Degrees Of Freedom (DOF's) and interpolation order values. A simple function is used to approximate the displacements in every element and a set of linear equations with displacements at each node as unknowns is formulated and finally the linear system of equations is solved. The flowchart depicted at Fig. 9.7 describes these steps.



Understanding and Describing the Physical problem
Simplifying the real problem to a FINITE ELEMENT problem
Modeling (Pre-Processing)
• Discretization/meshing
• Material properties
• Boundary conditions
• Loading condition
Processing
• Approximation functions choice
• Formulation of equations
• Solution of the equations system
Results (Post-processing)
Obtain, Visualize the Results.
Understand/Explain the Results.

Figure 5.7 Flowchart of modeling and processing of FEM (pre- & post- processing).

To conclude, we ask FEM to provide results for the unknown values given the minimum input data:

- Geometry: Math description, CAD input files etc.
- Material properties: Young's modulus, Density, Poisson's Ratio
- Boundary conditions: Supports, Prescribed displacements/stresses etc.
- The type of analysis: static, transient, modal, thermal, electromagnetic …

The detailed modeling process includes the selection of the Elements and Mesh for which a strategy to create a good mesh by the help of the appropriate Element types is adopted. When transient problems are solved, a time function is used to define how the prescribed boundary conditions change over time e.g. as ramp, step, sinusoidal etc. In numerical modeling time is the physical time counted by a clock, such as in dynamics and transient problems OR time simple means that one thing happens before another thing happens, such as in static problems. In summary, the Basic components of a FEM software include six basic features: i. Type of analysis, ii. Geometry (defined through mesh nodes), iii. Elements, iv. Material properties, v. Boundary conditions and vi. Time functions.

5.3 Understanding and describing the physical problem

The aim of a numerical simulation is to describe as approximately as possible the real physical problem. Developing a physical model is a process that simplifies a real-world problem into a FEM problem. The main considerations in building a physical model is the understanding of the nature of real-world problem and the a-priori knowledge of the cost of conducting the computational analysis. In many cases the cost of conducting FEM is a major barrier which imposes a great challenge and sometimes the maximum accuracy is sacrificed for the reduction of the computational cost for the achievement of a good solution.

The simulation cost demanded for the mathematical processing is strictly affected by the:

- Number of Nodes: N
- Degrees Of Freedom (DOF's / Node)
- The numbering of the Nodes: Matrix decompositions and algebraic operations costs
- Number of interpolation and integration points in every finite element: Interpolation order costs
- Nonlinear Analysis: Time integration and singularities

The type of analysis (static, dynamic, linear or nonlinear, 2D, 3D, transient, …) must be decided at the physical model level. The number of nodes and number of integration points are chosen during the building of the finite element model level. A good physical model has to reduce the number of nodes by one or two orders of magnitude [27,28].

5.4 FEM model development for the simulation of irradiated materials by nanosecond laser pulses

As depicted in Fig. 9.1, laser energy heats the target surface. The absorption of the laser pulse results to an increased localized temperature. The heating rate and the surface temperature are defined by absorption and reflection coefficients, by thermal conductivity, and by the specific heat of the solid. The target surface temperature reaches its melting point. Thermal and optical properties change during melting. In the ablation (plasma) regime, vaporization occurs. For incident laser intensities greater than the ablation threshold a large amount of electrons, ions and excited neutrals is present in the vaporized material and absorb the laser light forming plasma. For even greater laser intensities droplet or solid particle ejections occur.

At the very first moments, energy is deposited in the area of the laser beam spot resulting to the generation and propagation of Surface Acoustic Waves (SAWs) & bulge formation Bulge /Crater and Plasma formation while SAWs are generated and propagate in matter. The ultrasonic waves are generated in the solid in all directions from the laser interaction area.

The FEM model must be capable to simulate: the behavior of matter in every spatial direction, for any axisymmetric sample geometries, as well as target samples with no symmetry of material distribution. It has to be able to simulate the laser matter interaction providing a detailed view of the 3D thermo-mechanical results and giving spatiotemporal insights to solid target's dynamic reactions in any direction and in any regime of interest. The bulge formation, the generation and propagation of SAWs and the development of crater must be monitored and recorded with high resolution, especially in the cases of melting and ablation in order to be compared with the experimental results. Furthermore, the parametric model must be able to simulate cracks, enclosures or other defects, symmetric or not, that may exist in the sample.

The developed computational model presented here with a range of simulation results, covering every regime of nanosecond pulsed laser irradiation, demonstrating its capabilities. Experimental against numerical results obtained by the same model, were originally presented in the research work of Dimitriou at al. [22,29,30] and validated its effectiveness and accuracy.

5.4.1 FEM Mathematical modeling

The governing PDE of the problem is the Thermal conduction equation:

$$\rho(T)C_\rho(T)\frac{\partial T(x,y,z,t)}{\partial t}-\nabla[k(T)\nabla T(x,y,z,t)]=Q(x,y,z,t)-L_i \qquad (9.1)$$

Where, ***T*** transient temperature function, ***K*** thermal conductivity, ***ρ*** density, ***c***$\boldsymbol{c_p}$ specific heat, ***Q*** heat source, specified as the absorbed energy per volume unit per second and $\boldsymbol{L_i}$ latent heat (if $\mathbf{T<T_m}$ then $\boldsymbol{L_i = 0}$, if $\mathbf{T \geq T_m}$ then $\boldsymbol{L_i = L_m}$, if $\mathbf{T \geq T_b}$ then $\boldsymbol{L_i = L_v}$).

The wave propagation PDE is used to describe the structural part of the problem:

$$\rho\frac{\partial^2 U(x,y,z,t)}{\partial t^2}=\mu\nabla^2 U(x,y,z,t)+(\lambda+\mu)\nabla[\nabla U(x,y,z,t)]-\mathrm{a}(3\lambda+2\mu)\nabla T(x,y,z,t) \qquad (9.2)$$

Where, ***U*** displacement vector of thermal induced elastic waves, ***λ, μ*** are the *Lamé* constants, **a** thermoelastic expansion coefficient.

The model is loaded by the ns Laser heat source spatiotemporally described by:

$$Q(x,y,z,t) = I_s(t)(1-R)\exp(-4\ln 2(t/t_0))\exp(-(x^2+y^2)/r_0^2)a\exp(-az) \qquad (9.3)$$

assumed to be of Gaussian type. ***I_s*** incident laser power density (laser pulse energy per unit area per second), ***R*** optical reflectivity of the sample, ***α*** optical absorption coefficient (1 /α is the optical penetration depth), ***t_0*** is the FWHM laser pulse duration, **r_0** is the FWHM beam radius on the sample surface. Attention has to be given to the attenuation of irradiation due to the absorption in plasma (*consequence of laser irradiation*). Thus, increase in the absorption as a consequence of plasma heating is characterized by a single parameter, the density of the absorbed radiation energy E_a [31]. The temporal laser irradiance may be given by:

$$I_s(t) = I_0 e^{-\Lambda(t)}, \;\; \Lambda(t) = b\,h(t) + d\;E_a(t) \qquad (9.4)$$

Where, ***I_0*** incident laser pulse energy per unit area per second, **Λ(*t*)** optical thickness of the ablation plume, **h** the ablation depth, **b** and **d** time independent coefficients to be identified (free parameters).

The classical thermal conduction equation for finite elements with the heat capacity matrix [*C*] and the conductivity matrix [*K*] can be expressed in terms of vectors based on the finite element method:

$$[K]\{T\} + [C]\{\frac{\partial T}{\partial t}\} = \{Q\} \qquad (9.5)$$

where {*Q*} is the heat source vector, {*T*} is the temperature vector and {*∂T/∂t*} is the temperature rate vector. When temperature exceeds the melting point, the latent heat of melting L_m is subtracted from {*Q*}, which becomes

$$[K]\{T\} + [C]\{\frac{\partial T}{\partial t}\} = \{Q\} - L_m \qquad (9.6)$$

while L_m is replaced in by L_v when temperature exceeds the boiling point. For wave propagation, ignoring damping, the governing finite element equation is:

$$[M]\{\frac{\partial^2 U}{\partial t^2}\} + [S]\{U\} = \{F\} \qquad (9.7)$$

where [*M*] is the mass matrix, [*S*] the stiffness matrix, {*U*} the displacement vector, {$\partial^2 U/\partial t^2$} the acceleration vector and {*F*} the force vector. In general, the external force vector for an element is given by:

$$\{F\} = \int_V [B]^T [D]\{\varepsilon_0\}\, dV \qquad (9.8)$$

where $\{\varepsilon_0\}$ is the thermal strain vector, $[B]^T$ is the transpose of the derivative of the shape functions and $[D]$ is the material matrix.

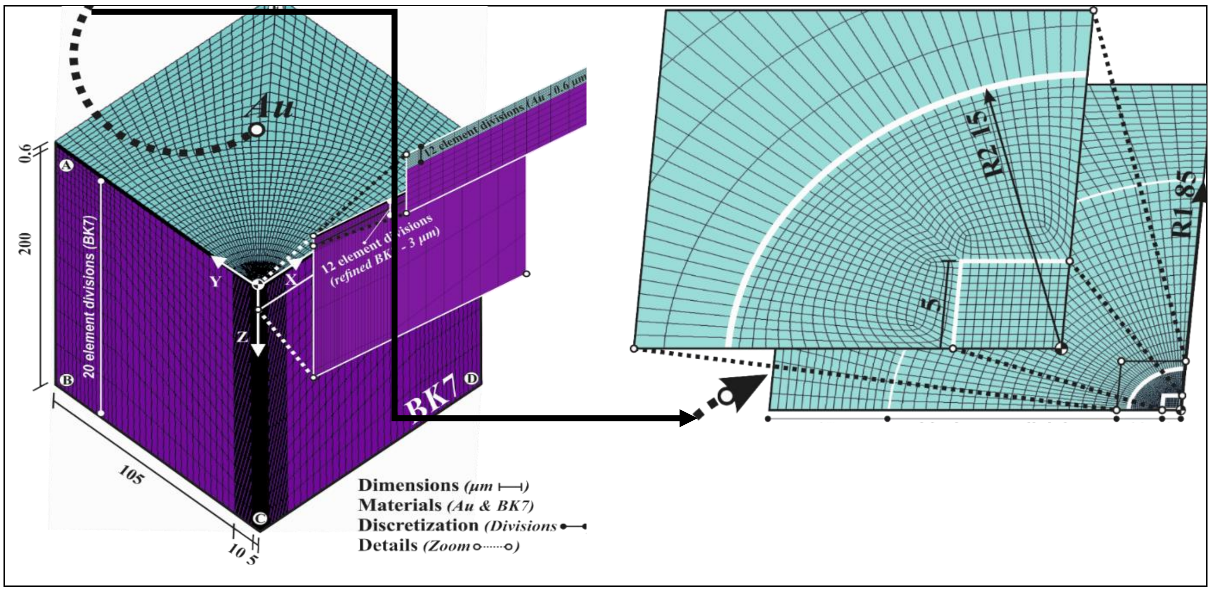
In this approximation of the 3D transient multiphysics thermal-structural mechanical problem, coupled-field analysis may be performed. To avoid individual solutions of the thermal and subsequently the structural problem in every timestep, analysis is applied by the interaction of thermal with the structural field [32]. Both of the engineering disciplines share the same 3D geometry, meshing, appropriate element type, boundary and loading conditions. All these Pre-process required input data are loaded for the entire simulation once in the beginning. Thus, the computational time is decreased and the possibility of loss of data during load transferring is eliminated.

By the help of the thermal-structural coupling direct method, a single pass solution is achieved, involving one analysis that uses the coupled-field 3D solid element type that uses eight nodes with up to six degrees of freedom per node. The weak field coupling, used in the proposed approximation, is accomplished by the calculation of the appropriate element matrices and load vectors resulting by the summation of the element matrices and load vectors. The necessary coupling terms may be included to the governing equation results to a form where the coupled effects are accounted for load terms F-coupled (*Fc*) and Q-coupled (*Qc*). This coupling requires at least two iterations in sequence to achieve a coupled response, one for each physical model applied [22]:

$$\begin{bmatrix} [M] & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} \{\ddot{u}\} \\ \{\ddot{T}\} \end{Bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & [C] \end{bmatrix} \begin{Bmatrix} \{\dot{u}\} \\ \{\dot{T}\} \end{Bmatrix} + \begin{bmatrix} [S] & 0 \\ 0 & [K] \end{bmatrix} \begin{Bmatrix} \{u\} \\ \{T\} \end{Bmatrix} = \begin{Bmatrix} \{Fc\} \\ \{Qc\} \end{Bmatrix} \tag{9.9}$$

5.4.2 FEM modeling and simulation results

To simplify the physical model and reduce the computational cost, the real problem is analyzed. A 3D quarter symmetric finite element model is chosen to simulate the laser matter interaction with the solid thin film-substrate sample, presented in Fig. 9.6. The model simulates a homogeneous, elastic, isotropic metal film-substrate system and its transient thermal-structural response when a single laser pulse interacts with the metallic film. The 3D FEM model is capable to fully describe the dynamic phenomena occurring from the laser energy concentration in the metallic film. Golden (Au) metal thin film of 0.6 μm thickness deposited on glass BK7 substrate of 200 μm thickness is assumed for the simulations performed. This model may ideally simulate the laboratory experiments performed and is validated by the resulting experimental data in every regime and under various loading conditions, as originally published in Ref. [29], where typical experimental against numerical results were compared.



Au
0.6
200
20 element divisions (BK7)
12 element divisions (Au = 0.6 μm)
12 element divisions (refined BK7 - 3 μm)
Y
X
Z
A
B
C
D
BK7
105
10.5
R2 15
R1 85
5
Dimensions (μm ⊢⊣)
Materials (Au & BK7)
Discretization (Divisions •⊣)
Details (Zoom o·····o)

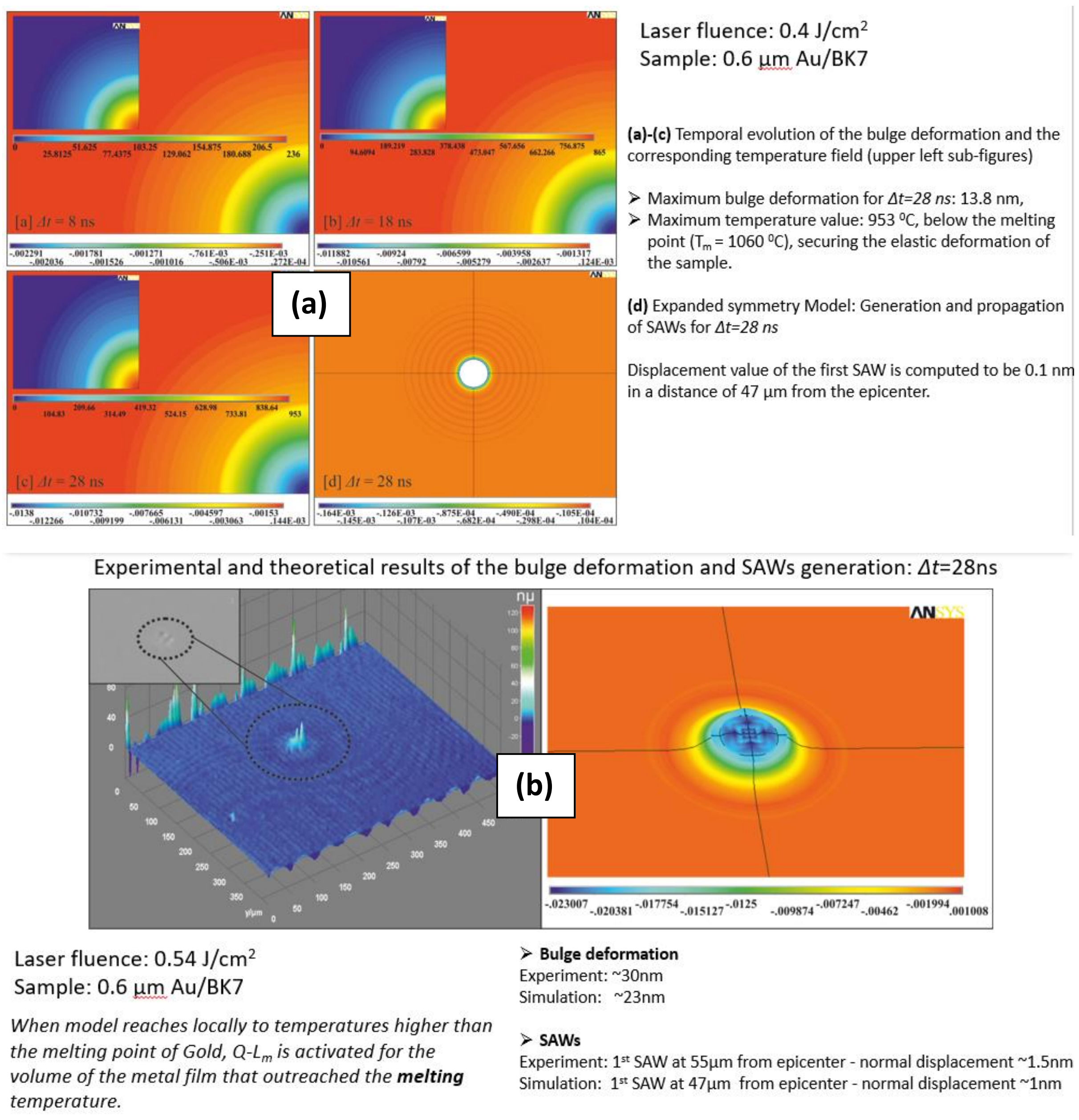
Figure 5.8. 3D quarter symmetric FEM model [22]

The thickness of the film used is of the order of a fraction of a μm and the heat-affected zone is much smaller than the domain of the material, therefore a fine mesh is necessary in order to resolve temperature distribution in the film and the irradiated region. As depicted in the zoomed detail of the irradiated surface in Fig. 9.8 (top), special treatment is given to the mesh of the cyclic area of radius R2 of the laser beam spot. At the limits of this area the temperature gradients change rapidly, thus requiring a locally smooth adaptive fine discretization [33]. The dynamic thermal effects occurring from the heat conduction are responsible for the structural response of matter around this area of high importance. That is the reason why a second cyclic area with radius R1 encloses R2 and creates the appropriate continuum smooth space, needed for the generation and propagation of SAWs. Since laser-generated SAWs have high frequencies and transient analysis is performed, a small element size is required to deal with these waves. The centre of symmetry is the laser's epicentre, where the maxima gradients of thermal effects occur and the melting and vaporization of matter are taking place. To deal with these extreme conditions a 5 μm x 5 μm rectangular square area with 12 element divisions in each side is built and mapped normal for 0.6 microns (film thickness) with 12 element divisions. This orthogonal fine meshed volume is generated to allow precision handling of the dynamic phase changes of matter in the center of the irradiated sample. These assumptions result to a sophisticated built locally adaptive fine mesh of 20160 elements (22737 nodes) in the quarter cyclic domain of radius R1=85 μm and a total of 27360 elements (30732 nodes) in the whole volume of the Au thin film. Likewise, the volume of glass is mapped for 3 μm with 12 elements in the normal direction to precisely transfer the substrate's dynamic reactions to the film. The whole sample is discretized to a total number of 88920 elements (94560 nodes).

The FEM model is set in the Global Coordinate System so that the positive Z-axis, follows the direction of the laser beam while its Origin (O) is placed to the center of the quarter symmetry, as presented in Fig. 9.8. Considering the loading conditions, a heat generation function, is applied on the film body. The initial temperature of the body is assumed to be the ambient temperature, set equal to 27 ^{0}C. Symmetric displacement loads are applied to the YZ (ABCO) and XZ (OCDE) planes, while heat flux is set to zero, regarding the quarter symmetry of the 3D model. At these planes of symmetry the discretization follows a smooth coarsening with increasing the distance from the top surface plane XY (AOEF).

A total time of 60 ns after excitation and 60 load-steps, resulting to a duration of 1 ns for each load-step for the solution, is considered. This time dependent problem is solved sequentially, with an incremental timestep of 1 ns. Regarding the laser parameters used in the performed simulations, the laser energy ranges from 0.8 to 20 μJ, the FWHM laser pulse duration, t_0, is 6 ns, while the FWHM beam radius on the sample surface, r_0, is 11.5 μm.

As indicated to the simulation flowchart in Fig. 9.7, all of the described Pre-Processing input data are stored and loaded once at the beginning of the simulation process. During the simulation the output of the preceding timestep is saved and becomes an input to the succeeding timestep. When simulation begins the first transient timestep is submitted for analysis. The criterions (flags) used are strictly related to the approximated temperature results, as shown in the decision rhombus boxes of the flow chart. At the end of a particular step, if the temperature of an element is higher than the melting temperature *(T_m)*, phase change occurs, which is taken into account by considering the latent heat of melting *(L_m)* in the model. The ablation is assumed to occur when the temperature of the corresponding elements is higher than the boiling temperature. Likewise, in this condition the model takes into account the phase change effect by considering the latent heat of vaporization. If the resulting temperature of an element overpasses the boiling point, a group of elements (GT_v) to be vaporized, is created. These elements belong to the material removal subdomain that is realized by the "killing" of the GT_v. These elements are deactivated by multiplying their stiffness by a severe reduction factor (~1x10^{-8}). When laser fluence is higher than the ablation threshold, the optical thickness of the ablation plume, $\Lambda(t)$, should be considered and Eq. 6 is activated and taken into account by the model [16, 19]. Comparison with experimental results, strictly demanded for this case, allows for the evaluation of the values of the plasma coefficients, *a* and *b*, which for our test cases were found to be 1×10^6 and 1×10^{-4}, respectively [29]. For the estimation of *a* and *b* values, for any other sample with different material properties, this comparison of simulation vs. experimental results, has to be performed. The computational time required for each successive time step of 1 ns, is approximately ~30 minutes and is increased to ~1h when killing of the elements is performed. The simulation is accomplished when the total time of the transient analysis is reached. Representative results of the simulation are depicted in Figures 9.9 and 9.10 in accordance to the correspondent experimental results and explanation details for the dynamic behavior of the model.



Laser fluence: 0.4 J/cm^2
Sample: 0.6 μm Au/BK7

(a)-(c) Temporal evolution of the bulge deformation and the corresponding temperature field (upper left sub-figures)

- Maximum bulge deformation for *Δt=28 ns*: 13.8 nm,
- Maximum temperature value: 953 ^{0}C, below the melting point (T_m = 1060 ^{0}C), securing the elastic deformation of the sample.

(d) Expanded symmetry Model: Generation and propagation of SAWs for *Δt=28 ns*

Displacement value of the first SAW is computed to be 0.1 nm in a distance of 47 μm from the epicenter.

Experimental and theoretical results of the bulge deformation and SAWs generation: *Δt*=28ns

Laser fluence: 0.54 J/cm^2
Sample: 0.6 μm Au/BK7

When model reaches locally to temperatures higher than the melting point of Gold, Q-L$_m$ is activated for the volume of the metal film that outreached the ***melting*** *temperature.*

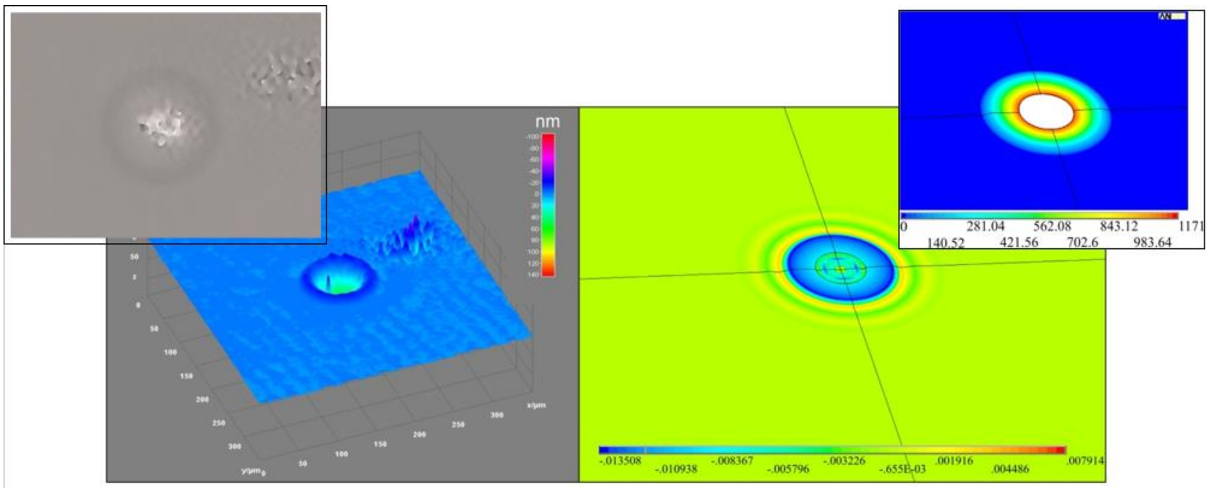
- **Bulge deformation**

Experiment: ~30nm
Simulation: ~23nm

- **SAWs**

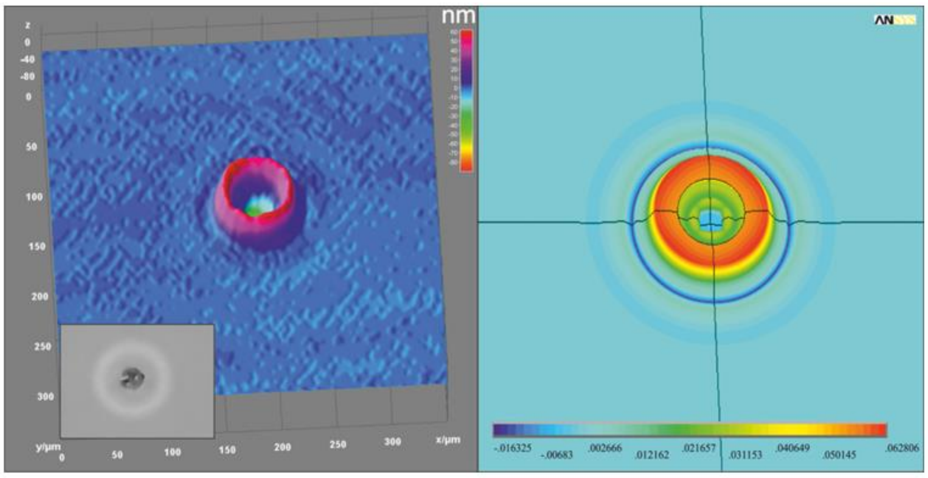
Experiment: 1st SAW at 55μm from epicenter - normal displacement ~1.5nm
Simulation: 1st SAW at 47μm from epicenter - normal displacement ~1nm

Figure 5.9 FEM results (a) and comparison to experimental results (b) [34,35].



Laser fluence: 3 J/cm^2
Sample: 0.6 μm Au/BK7

Bulge deformation: ~20 nm for both experimental and simulation data
Crater walls diameter: ~45 μm for the simulation and ~50 μm for the experiment
First SAW Experiment: ~58 μm from epicenter & normal displacement 2 nm
Simulation: ~53 μm from epicenter & normal displacement 3 nm



Laser fluence: 35 J/cm^2
Sample: 0.6 μm Au/BK7
Δt = 28ns

Bulge deformation: ~60 nm for both experimental and simulation data
Crater walls diameter: ~35 μm for the simulation and ~40 μm for the experiment
Experiment: ~50 μm from epicenter & normal displacement ~3 nm
Simulation: ~45 μm from epicenter & normal displacement ~3 nm

Figure 5.10 Comparison of experimental (left) to FEM results (right) [34,35].

5.5 FEM multiphysics simulations for MHD analysis

To extend our modeling and simulation study to the generation of plasma its features & properties, numerical simulations provide computational tools able to describe the dynamic behavior of the fourth state of matter. The study of the initiation of the plasma state of matter includes the investigation of every phase change step until the initial formation of plasma. To secure the final results of a plasma study a validation of parallel effects in every regime by the help of comparison of computational predictions versus experimental results is needed. This approach requires the development of a method, combining theory, simulations and experiments, able to investigate any regime until plasma formation. The deep investigation of the parallel effects to validate the developed theory and models may guide the experiments and vice versa. This modeling approach [36-40] of the problem by the help of experimental results secures the investigation of plasma dynamics in order to achieve:

- Clear view of solid matter's, liquid and gas dynamic response
- Validated simulation results for every regime
- Validated initial conditions for the liquid and gas phase

To understand the initial stages of explosion of dense plasmas as the phase changes from solid to plasma take place experiments using a Z-pinch pulsed powered device are performed. The experiment is implemented in a mode of producing a peak current of 35 kA with a rise time (10%-90%) of 60 ns. The results for the expansion dynamics of the exploded plasma obtained from laser probing diagnostics

- modified Fraunhofer diffraction probe method
- shadowgraphic techniques
- interferometric techniques
- time-integrated optical imaging to monitor matter dynamics.

A FEM transient multiphysics electro-magnetic-thermal-structural 3D model is developed for the numerical study of the problem due to its unique characteristics:

- versatile and flexible
- substantial insights into key physical quantities
- temperature-dependent material properties
- strength material model (Johnson-Cook) along with an equation of state (Gruneisen)

The proposed method provides quantitative parameter values: temperature, pressure, current density and expansion rate of the exploded material. Offers detailed information of temperature gradients, transformations, velocities, etc. for any time step (via FEM model) and the expansion rates of the mater and the phase change characteristics of the exploded matter. The numerical study of the initial stages of electrical exploding wire that play an important role in plasma formation in pulsed-power Z-Pinch experiments. Early time dynamics in the explosion of the wire have been proven to be important for the development of Magneto-Hydrodynamic (MHD) instabilities of the Z-pinch plasma [38,39]. It has been shown that the stages of the phase changes from thermoelastic to melting and plasma regimes are crucial and that thick metallic copper wires (300 μm) for which electrical charges flow through skin depth important role for the exploding dynamics [41,42].

5.5.1 3D multiphysics FEM analysis

A 3D coupled FEM multiphysics simulation based on the method is developed. Maxwell equations (eddy-current approximation) are solved using a finite element method (FEM) for the wire coupled with a Boundary Element Method (BEM) for the surrounding vacuum. Moreover, the skin depth effect for the wire is also taken into account. When the electromagnetic fields have been computed, the Lorentz force **F=j×B**, where j is the current density and B the magnetic field, is evaluated at the nodes and added to the mechanical solver, which computes the deformation of the wire. Furthermore, the joule heating power term j^2/σ, σ the electrical conductivity, is added to the thermal solver to update temperature [43].

In order to properly simulate the magneto-hydrodynamic response of the metal: the hydrodynamic behavior is taken into account by using analytical Gruneisen equation of state [44], the deviatoric behavior is taken into account using Johnson-Cook [45] strength material model and the electrical conductivity versus temperature and density is computed using Burgess equation of state [46]. For the mechanical response, the Johnson-Cook model is coupled with the Gruneisen equation of state. The Johnson-Cook material model considers the effect of plastic strain, strain rate and temperature rise. Temperature dependent properties of thermal expansion, thermal conductivity and specific heat, as well as the latent heat of melting are also considered. Regarding the boundary conditions, the ends of the wire are fixed at environmental temperature (27 ^{0}C). An important aspect of the developed simulation is that the Lagrangian mesh is appropriately refined to accurately simulate the dynamic phase changes of matter in the region of the skin depth. The loading source term is the alternating current from the experiments.

The mathematical modeling of the problem is provided by the Maxwell, the Mass conservation, the Momentum and the Energy equations:

Maxwell equations:

$$\nabla\times\underline{\boldsymbol{E}} = -\frac{\partial \underline{\boldsymbol{B}}}{\partial t}$$

$$\nabla\times\underline{\boldsymbol{B}} = \mu_0 \boldsymbol{j}$$

$$\nabla\cdot B = 0$$

$$\boldsymbol{j} = \sigma_{cond} E + \boldsymbol{j}_s$$

$$\nabla\cdot \boldsymbol{j} = 0$$

Mass conservation equation:

$$\rho V = \rho_0$$

$$V = \left|\frac{\partial x_i}{\partial X_j}\right|$$

Momentum equation: $\rho\ddot{x} = \nabla\sigma + \rho f + j\times B$

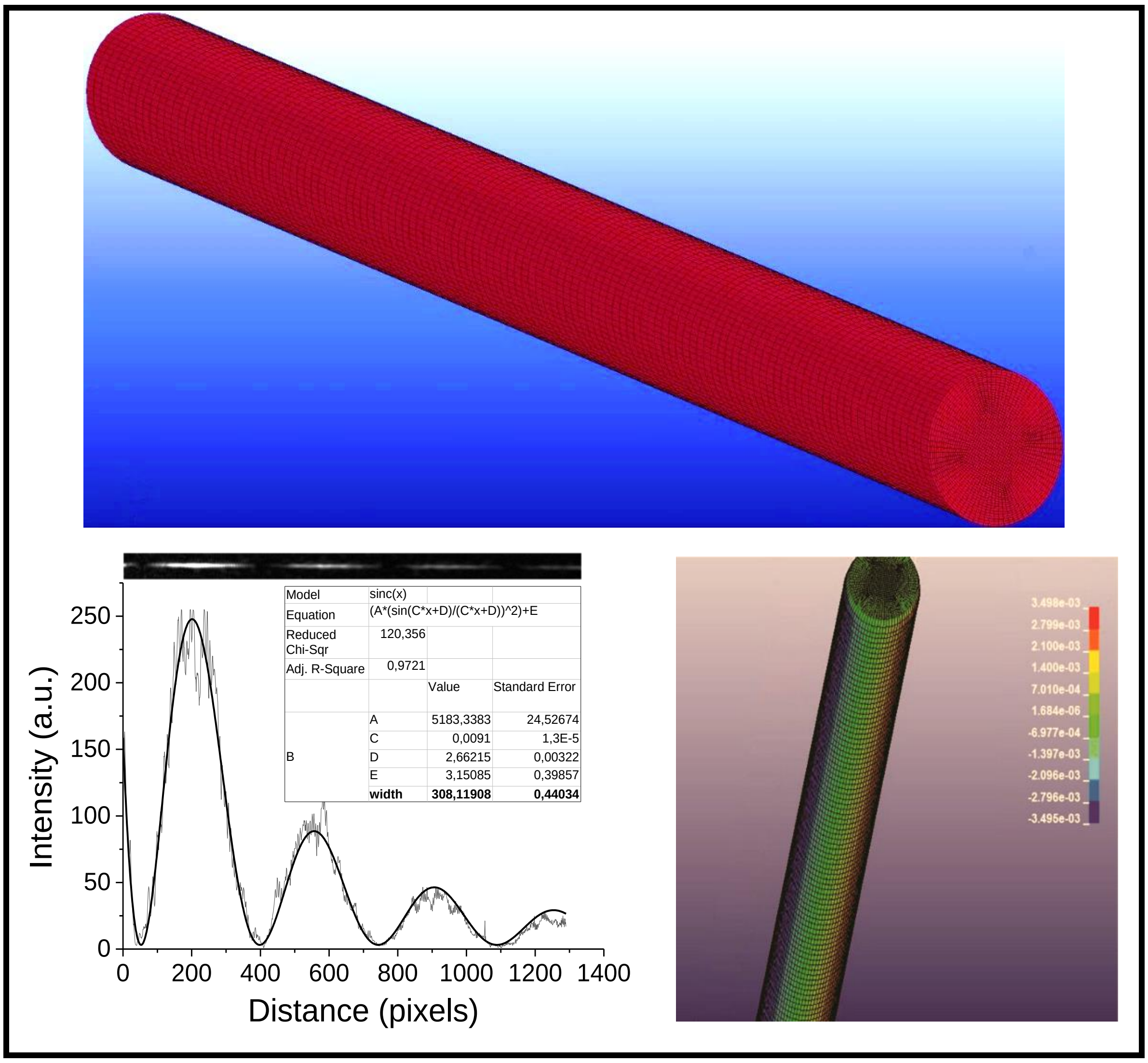
Energy equation: $\dot{E} = V s_{ij}\dot{\varepsilon}_{ij} - (p+q)\dot{V} + \dfrac{j^2}{\sigma_{cond}}$

$$s_{ij} = \sigma_{ij} + (p+q)\delta_{ij}$$

Jonhson-Cook flow stress: $\sigma_y = (A + B\varepsilon_{pl}^n)(1 + C\ln\frac{\dot{\varepsilon}_{pl}}{\dot{\varepsilon}_0})(1-\frac{T-T_r}{T_m-T_r})^m$

Equation of state for pressure: $\mathrm{p} = \mathrm{p}(\rho, E)$

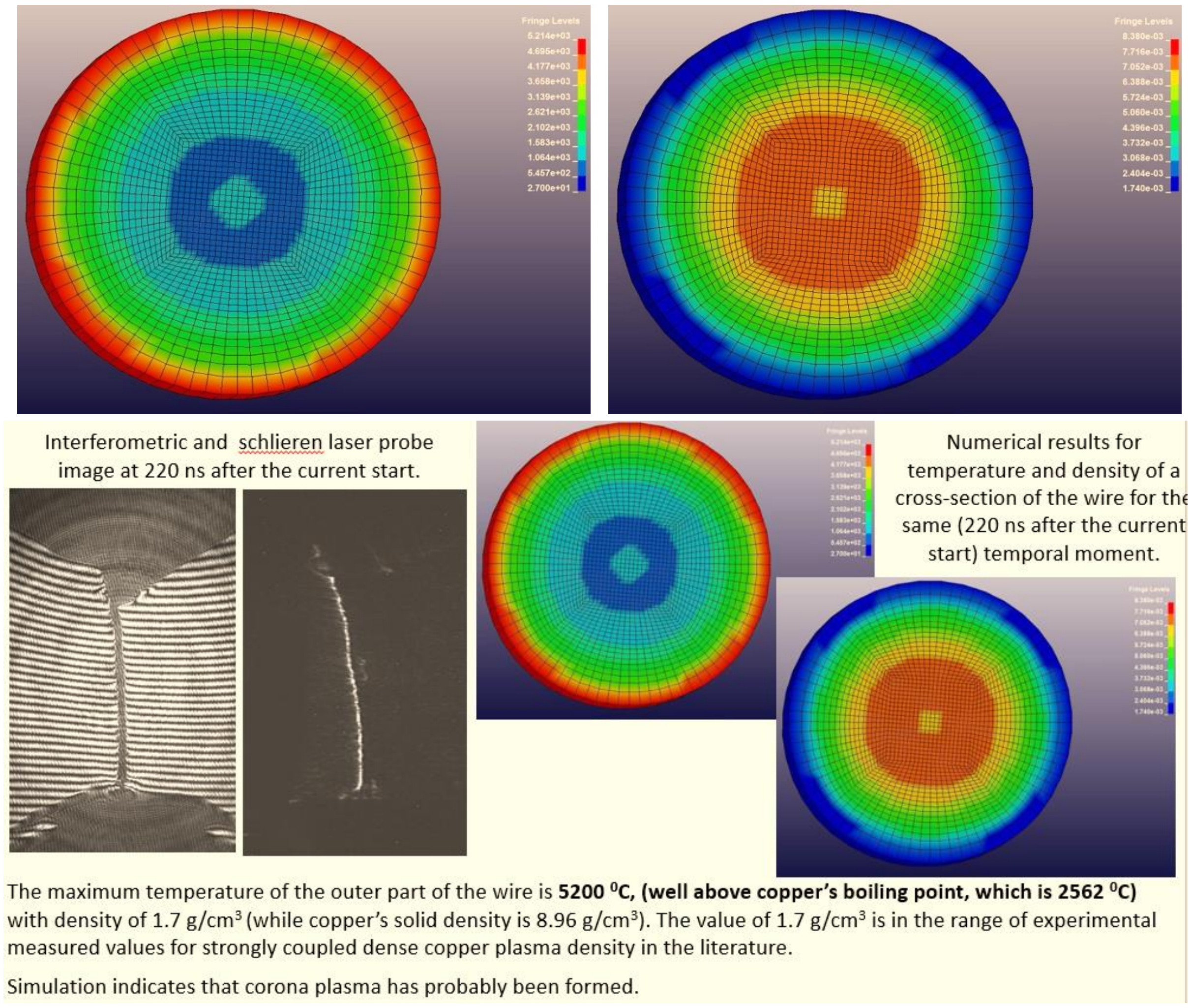
Equation of state for electrical conductivity: $\sigma_{cond} = \sigma_{cond}(\mathrm{T},\rho)$



Model	sinc(x)		
Equation	(A*(sin(C*x+D)/(C*x+D))^2)+E		
Reduced Chi-Sqr	120,356		
Adj. R-Square	0,9721		
		Value	Standard Error
B	A	5183,3383	24,52674
	C	0,0091	1,3E-5
	D	2,66215	0,00322
	E	3,15085	0,39857
	width	**308,11908**	**0,44034**

Figure 9.11 The FEM model (up), Laser probe diffraction pattern and the lineout intensity plot along the axis of the fringes at 140 ns from the current start (bottom left). Displacement of the x-axis (mm) of the wire the same temporal moment (bottom right) [47].

At earlier times, the wire, due to the Joule heating, experiences thermal expansion as well as melting and vaporization. The model and representative results are depicted in Figures 9.11 and 9.12. The measurement of the diameter takes place 140 ns from the current start and is compared with the initial measurement, presenting an expansion of 7.4 μm and the simulation gives an expansion of 7 μm, as shown in Figure 9.11 at the same time.



Interferometric and schlieren laser probe image at 220 ns after the current start.

Numerical results for temperature and density of a cross-section of the wire for the same (220 ns after the current start) temporal moment.

The maximum temperature of the outer part of the wire is **5200 ^{0}C, (well above copper's boiling point, which is 2562 ^{0}C)** with density of 1.7 g/cm^3 (while copper's solid density is 8.96 g/cm^3). The value of 1.7 g/cm^3 is in the range of experimental measured values for strongly coupled dense copper plasma density in the literature.

Simulation indicates that corona plasma has probably been formed.

Figure 5.12. Finite element numerical results for temperature (left, ^{0}C) and density (right, g/cm^3) from a cross section of the wire at 220 ns from current start (up). Interferometric and schlieren laser probe images at 220 ns after current start (down left) and numerical results of temperature and density distributions computed by FEM (down right) [42,47].

The computational results demonstrate that the combination of the multiphysics FEM model and the experimental method are capable to describe the wire expansion dynamics for temperatures below the boiling point. Satisfactory agreement of experimental versus the FEM results is also observed concerning the initial times of corona plasma formation.

Multiphysics numerical simulations of the dynamic response of the target during its heating and conversion into plasma have been developed based on the coupling of FEM and MHD methods. To study the wire's response to the heating source (Joule heating), a strength material model along with evaluated EOS data are used. The EOS data are necessary to describe the hydrodynamic response of the material, while the strength material model describes the deviatoric stress behavior or distortion of the material. The density distribution and the wire radius at the last time step of the solution provided by the FEM simulation are coupled to a resistive MHD finite difference/volume code (PLUTO). A fluid and a plasma region, surrounded by a vacuum region, are considered for the MHD transient analysis. This choice is based on experimental results suggesting that the plasma of the Z-pinch target consists of a dense fluid that surrounds the solid, which persists for a long time during the current discharge, surrounded by a low density hot coronal plasma. The proposed simulation method is identical for the study of plasma instabilities, a research topic with fundamental importance since for the majority of plasma applications they are unwanted and there is always the need for their suppression. The implementation of the FEM-MHD coupled method offered a perspective to the understanding of the seeding physical mechanisms in the generation of plasma instabilities [41]. Representative MHD results compared to Experimental are presented in Figure 9.13.

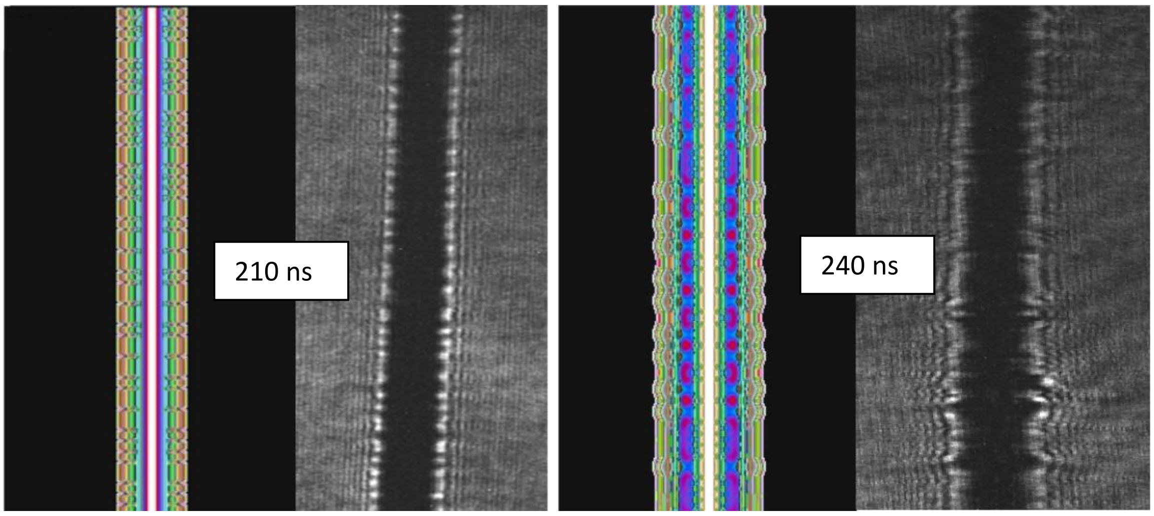


Figure 5.13 Comparison of MHD simulation with experimental picosecond laser optical probing shadowgraphy [41].

Bibliography & references

[1] H. Podlesak, T. Schnick, L. Pawlowski, S. Steinhäuser, B. Wielage, Microscopic study of Al–SiC particulate composites processed by laser shocks, Surf. Coat. Tech. 124 (2000)
[2] Al.A. Kolomenskii, H.A. Schuessler, V.G. Mikhalevich, A.A Maznev, Interaction of laser-generated surface acoustic pulses with fine particles: Surface cleaning and adhesion studies, J. Appl. Phys. 84 (1998) 2404-2410.
[3] M. Sakamoto, X. Cai, S.S. Kim, M. Fujitsuka, T. Majima, Intermolecular electron transfer from excited benzophenone ketyl radical, J. Phys. Chem. A 111 (2007) 223-229.
[4] S.C. Chen, D.G. Cahill, C.P. Grigoropoulos, Melting and surface deformation in pulsed laser surface micromodification of Ni-P disks, J Heat Transf 122 (2000) 107-112.
[5] F. Reverdy, B. Audoin, Ultrasonic measurement of elastic constant of anisotropic materials with laser source and laser receiver focused on the same interface, J. Appl. Phys. 90(9) (2001) 4829–4835.
[6] S.J. Davies, C. Edwards, G.S. Taylor, S.B. Palmer, Laser-generated ultrasound: its properties, mechanisms and multifarious applications, J. Phys. D Appl. Phys. 26 (1993).
[7] R.E. Green, Non-contact ultrasonic techniques, Ultrasonics 42 (2004) 9-16.

[8] A. Moura, A.M. Lomonosov, P. Hess, Depth evaluation of surface-breaking cracks using laser-generated transmitted Rayleigh waves, J. Appl. Phys. 103 (2008) 084911-16.
[9] S. Dixon, S.E. Burrows, B. Dutton, Y. Fan, Detection of cracks in metal sheets using pulsed laser generated ultrasound and EMAT detection, Ultrasonics 51 (2011) 7-16.
[10] S. Yashiro, J. Takatsubo, H. Miyauchi, N. Toyama, A novel technique for visualizing ultrasonic waves in general solid media by pulsed laser scan, NDT&E Int. 41 (2008) 137–144.
[11] I.M. Pelivanov, D.S. Kopylova, N.B. Podymova, A.A. Karabutov, Optoacoustic method for determination of submicron metal coating properties: Theoretical consideration, J. Appl. Phys. 106 (2009) 013507-013514.
[12] R.K. Singh, J. Narayan, Pulsed-laser evaporation technique for deposition of thin films: Physics and theoretical model, Phys. Rev. B. 41 (1990) 8843-8859.
[13] J.R. Ho, C.P. Grigoropoulos, J.A. Humphrey, Computational study of heat transfer and gas dynamics in the pulsed laser evaporation of metals, J. Appl. Phys. 78 (1995) 4696-4709.
[14] R. Coulette, E. Lafond, M.H. Nadal, C. Gondard, F. Lepoutre, O. Petillon, Laser-generated ultrasound applied to two layered materials characterization: semi-analytical model and experimental validation, Ultrasonics 36 (1998) 239-243.
[15] T.W. Murray, S. Krishnaswamy and J. D. Achenbach, Laser generation of ultrasound in films and coatings, Appl. Phys. Lett. 74 (1999) 3561-3563.
[16] N.A. Vasantgadkar, U.V. Bhandarkar, S.S. Joshi, A finite element model to predict the ablation depth in pulsed laser ablation, Thin Solid films 519 (2010) 1421-1430.
[17] B.Q. Xu, Z.H. Shen, X.W. Ni, J. Lu, Finite element model of laser-generated surface acoustic waves in coating-substrate system, J. Appl. Phys. 95 (2004) 2109–2115.
[18] B.Q. Xu, Z.H. Shen, X.W. Ni, J. Wang, J. Guan, J. Lu, Thermal and mechanical finite element modeling of laser-generated ultrasound in coating–substrate system, Opt. Laser Technol. 38 (2006) 138-145.
[19] V. Oliveira, R. Vilar, Finite element simulation of pulsed laser ablation of titanium carbide, Appl. Surf. Sci. 253 (2007) 7810-7814.
[20] H.S. Lim, J. Yoo, FEM based simulation of the pulsed laser ablation process in nanosecond fields, J. Mech. Sci. Technol. 25 (2011) 1811-1816.
[21] Y. Orphanos, V. Dimitriou, E. Kaselouris, Efthimios Bakarezos, N. Vainos, Michael Tatarakis, N.A. Papadogiannis, An integrated method for material properties characterization based on pulsed laser generated surface acoustic waves, Microelectron. Eng. 112 (2013) 249–254.
[22] V. Dimitriou, E. Kaselouris, Y. Orphanos, E Bakarezos, N. Vainos, I.K. Nikolos, M. Tatarakis, N. A. Papadogiannis, The thermo-mechanical behavior of thin metal films on dielectric substrates under nanosecond laser pulse excitation above the thermoelastic regime, Appl. Phys. A. 118 (2015) 739-748.
[23] D. Schneider, T.A. Schwarz, H.J. Scheibe, M. Panzner, Non-destructive evaluation of diamond and diamond-like carbon films by laser induced surface acoustic waves, Thin Solid films 295 (1997) 107-116.
[24] K. Nomura, H. Ohta, K. Ueda, T. Kamiya, M. Hirano, H. Hosono, Thin-film transistor fabricated in single-crystalline transparent oxide semiconductor, Science 300 (2003) 1269-1272.
[25] A.A Volinsky, N.R. Moody, W.W. Gerberich, Interfacial toughness measurements for thin films on substrates, Acta. Mater. (2002) 441-466.
[26] S. Amoruso, R. Bruzzese, N. Spinelli, R. Velotta, Characterization of laser-ablation plasmas, J. Phys. B: At. Mol. Opt. Phys. 32 (1999) R131-R172.
[27] O. Zienkevich and R. Taylor, The Finite Element Method, Vol. 1, McGraw-Hill, New York (1991)
[28] Dr. H. “Jerry” Qi, Finite Element Analysis, Lecture Notes, Colorado edu. (2006).
[29] V. Dimitriou, E. Kaselouris, Y. Orphanos, M. Bakarezos, N. Vainos, M. Tatarakis, N. A. Papadogiannis, Three dimensional transient behavior of thin films surface under pulsed laser excitation, Appl. Phys. Lett. 103 (2013) 114104.

[30] V. Dimitriou, E. Kaselouris, Y. Orphanos, E. Bakarezos, N. Vainos, I. K. Nikolos, N. A. Papadogiannis and M. Tatarakis, Matter dynamics under the interaction with laser pulses in the thermoelastic & plasma regimes, 40th EPS Conference on Plasma Physics, Helsinki, Finland, July 2013.
[31] A.V. Bulgakov, N. M. Bulgakova, Thermal model of pulsed laser ablation under the conditions of formation and heating of a radiation-absorbing plasma, Quantum Electron 29 (1999) 433-437.
[32] ANSYS Inc., Coupled-field analysis guide, Release 12.0, Southpointe Canonsburg, April 2009.
[33] V. Dimitriou, A. Kanarachos, D. Koulocheris, An Approach to Unstructured Finite Element Mesh Generation Using Coons Mapping and Smoothing Techniques, WSEAS Transactions on Circuits and Systems 2 (2003) 473 – 478.
[34] E. Kaselouris, Y. Orphanos, V. Dimitriou, E. Bakarezos, N. Vainos, M. Tatarakis, N. A. Papadogiannis, 3D finite element modeling of laser-generated surface acoustic waves in film-substrate systems validated by experiments, 10th HSTAM International Congress on Mechanics, 25-27 May 2013, Chania.
[35] E. Kaselouris, V. Dimitriou, I.K. Nikolos, Y. Orphanos, E. Bakarezos, N.A. Papadogiannis, M. Tatarakis, Numerical simulations for the study of matter behavior dynamics governed by the interaction with laser pulses or external strong currents, 10th HSTAM International Congress on Mechanics, 25-27 May 2013, Chania.
[36] M.G. Haines, *A review of the dense Z-pinch*, *Plasma Phys. Control. Fusion* 53, 093001 (2011).
[37] D.A. Hammer and D.B. Sinars, Single-wire explosion experiments relevant to the initial stages of wire array z pinches*, Laser Part. Beams* 19, 377–391 (2001).
[38] M. Tatarakis et al., Optical probing of fiber z -pinch plasmas, *Phys. Plasmas*5, 682 (1998).
[39] V.I. Oreshkin et al., Wire explosion in vacuum: Simulation of a striation appearance, *Phys. Plasmas* 11*,* 4771-76 (2004).
[40] H. Calamy et al., Use of microsecond current prepulse for dramatic improvements of wire array Zpinch implosion, *Phys. Plasmas* 15, 012701 (2008).
[41] E. Kaselouris, V. Dimitriou, I. Fitilis, A. Skoulakis, G. Koundourakis, I. K. Nikolos, E. L. Clark, M. Bakarezos, N. A. Papadogiannis and M. Tatarakis, The influence of the solid to plasma phase transition on the generation of plasma instabilities, Nature Communications 8 (1), 1713 2017
[42] E. Kaselouris, V. Dimitriou, A. Skoulakis, I. Fitilis, Y. Orphanos, I.K. Nikolos, E. Bakarezos, N.A. Papadogiannis and M. Tatarakis, Experimental & Numerical study of the initial stages of explosion of thick single wire z-pinch, 41st EPS (2014).
[43] G. Le Blanc et al., Ramp wave compression in a copper strip line: comparison between MHD numerical simulations (LS-DYNA) and experimental results (GEPI device), 10th International LSDYNA Conference, 2008.
[44] K. Nagayama, Introduction to Grüneisen Equation of State and Shock Thermodynamics, Kindle Edition.
[45] G.R Johnson and W.H. Cook, Fracture characteristics of three metals subjected to various strains, strain rates, temperatures and pressures, Eng. Fract. Mech 21, 31-48 (1985).
[46] T. Burgess, Electrical resistivity model of metals, 4th International Conference on Megagauss Magnetic-field generation and related topics, Santa Fe, NM, 1986.
[47] I. Fitilis, A. Skoulakis, E. Kaselouris, I. K. Nikolos, E. Bakarezos, N. A. Papadogiannis, V.M. Dimitriou, M. Tatarakis, Diagnosing the initial stages from solid to plasma phase for dense plasma explosions, Proceedings of Science,1st EPCD Conference on Plasma Diagnostics, 14-17 April 2015, Frascati, Italy.

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Chapter 6: Particle-in-Cell simulations of laser-plasma interactions

J. Psikal

6 Particle-in-Cell simulations of Laser-Plasma Interactions

There are three basic approaches for numerical calculations of plasma evolution [1]: particle approach, kinetic theory, and hydrodynamics.

- **Particle description** is based on the equations of motion of individual particles in electromagnetic fields. The plasma is described by electrons and ions moving under the influence of the electric and magnetic fields due to their own charge and of the laser fields. For practical reasons, computer simulation of plasma using particle codes is limited to $N \approx 10^{10}$ particles in the most demanding calculations, whereas the number of particles in typical laboratory laser-plasma system is much higher ($\sim 10^{15}$ particles). Therefore, each particle has to represent a large number of real electrons or ions in this type of simulation. However, the decreasing number of particles means increasing noise, which indicates that this approach has limitations and some specific phenomena have to be studied by kinetic approach.

- **Kinetic description** is based on a set of equations for (macroscopic) distribution function $f_s(\vec{x}, \vec{p})$ of each plasma particle species s together with Maxwell equations. The distribution function is a statistical description of a very large number of interacting particles. Each particle has its own position in the phase space $(\vec{x}, \vec{p})$, where $\vec{x}$ are the coordinates for all the degrees of freedom and $\vec{p}$ are the corresponding momentum components. In this approach, $f_s(\vec{x}, \vec{p})\mathrm{d}^3\vec{x}\mathrm{d}^3\vec{p}$ is equal to the number of particles of species s in the domain $[(\vec{x}, \vec{x}+\mathrm{d}\vec{x}), (\vec{p}, \vec{p}+\mathrm{d}\vec{p})]$.

  Vlasov equation describing the evolution of the distribution function $f_s(\vec{x}, \vec{p}, t)$ for each species of particles is given as follows:

$$\frac{\partial f_s}{\partial t} + \frac{\vec{p}}{m_s\gamma}\frac{\partial f_s}{\partial \vec{x}} + q_s\left(\vec{E} + \left(\frac{\vec{p}}{m_s\gamma}\right) \times \vec{B}\right)\frac{\partial f_s}{\partial \vec{p}} = 0, \qquad (1)$$

  where m_s is the rest mass of particle species s, q_s its charge, $\vec{E}$ and $\vec{B}$ electric and magnetic fields in the position $\vec{x}$ and time t. The statistical content of the Vlasov equation is expressed by assuming that f_s is a smooth function (i.e., differentiable) describing an average quantity over a phase space volume $\mathrm{d}^3\vec{x}\mathrm{d}^3\vec{p}$ containing a large number of particles. The electromagnetic fields, $\vec{E}$ and $\vec{B}$ are also smooth averaged quantities. The force acting on any plasma particle, describing the effect of all the other particles, is assumed to be a continuous and slowly varying function of space. This is a good approximation only if the collective effect is larger than direct collisions with nearby particles; therefore, the Vlasov equation is considered to be collisionless. If the short-range collisions are important, the collision term is added on the right hand side of Eq. (1).

- **Hydrodynamic description** uses conservation laws of mass, momentum, and energy which are coupled to Maxwell equations. In addition, for a fluid model, a local thermodynamic equilibrium (LTE) is assumed and the knowledge of the equations of state (relations between pressure, temperature, energy, entropy, etc.) is mandatory for solving the problem. The fluid theory is a good approximation for many phenomena in the interaction of plasma with relatively low laser intensities ($\approx 10^{15}$ W/cm^2) and relatively long laser pulses (ns). However, the model is not always adequate, because there is an assumption of LTE. All the variables in the fluid equations are functions of time and position, and each species in an LTE plasma has a Maxwellian distribution of the velocities everywhere. Physical quantities such as temperature and pressure can be defined only in LTE. Systems that are not in LTE (such as plasma interacting with relativistically intense femtosecond laser pulse) cannot be described by fluid equations. Hydrodynamic model can describe a target globally including nonionized solid part of the target.

For the study of ultrashort laser pulse interaction with ionized targets, particle approach is mostly used, namely particle-in-cell (PIC) simulation method. This method can study the interaction of ionized matter with very high intensity pulses in the so-called relativistic regime when the kinetic energy of electrons oscillating in the laser field exceeds its rest energy (at intensities $I > 10^{18}$ W/cm^2). In this case, the electrons with high relativistic velocities do not heat the plasma by collisional effects since the collisional frequency of those electrons with background plasma is indirectly proportional to the velocity of electrons cubed [2, 3]. Thus, the plasma is out of LTE and only kinetic or particle description of plasma can be applied in the calculations of the interaction and subsequent plasma evolution.

The potential advantage of kinetic Vlasov codes is a possibility of producing smooth results as the Vlasov codes handle the distribution function which is a smoothly changing real number already giving a probability of finding the plasma particles at the corresponding point of the phase space. Therefore, this approach is useful in the studies of some specific phenomena, for example stimulated Raman (back-)scattering [4, 5]. On the other hand, Vlasov codes are very expensive from the computational point of view. The kinetic Vlasov equation (1) on the single-particle distribution function has to be generally solved in the six-dimensional phase space and even one-dimensional problem may demand the use of parallel supercomputers [6]. Even though PIC codes are also relatively demanding on computational resources, Eulerian grid in the configuration space has half of the dimensions (i.e., spatial coordinates) compared with Vlasov codes (i.e., spatial plus corresponding momentum coordinates). Therefore PIC codes are usually the best option for the calculations.

6.1 Basics of the PIC simulation technique

The core algorithm of PIC code consists of two coupled solvers: one that moves charged particles freely in space under the influence of electromagnetic fields and calculates the currents due to the particle motions (the particle pusher) and another that solves Maxwell equations on a fixed spatial grid subject to the currents calculated from the particle motions (the field solver) [7].

In fact, the PIC method replaces the distribution function f_s used in (1) with macroparticles [6]. Each macroparticle represents a large number of real particles, with the number of real particles represented by each simulation particle (i.e., by each macroparticle) called the weight. When simulations contain changes in density there is the option to either represent this by changing the weight of each macroparticle to reproduce the density structure or by keeping the weight of the macroparticles constant and changing the number density of macroparticles to match the real density.

The relationship between the distribution function and the macroparticles can be described using the following relation:

$$f_s(\vec{x},\vec{p},t) = \sum_{k=1}^{N} N_k S_x(\vec{x} - \vec{x_k}) S_p(\vec{p} - \vec{p_k})). \tag{2}$$

Each macroparticle represents N_k real particles (which is numerical weight of macroparticle) and has its own position $\vec{x_k}$ and momentum $\vec{p_k}$. The functions S_x and S_p represent how a macroparticle "looks" in the position and momentum phase space, respectively, and are usually referred to as shape functions. Choice of the shape functions has an important impact on the numerical properties of the algorithm [7].

Field solver

The finite-difference time-domain method (FDTD) is implemented in most PIC codes as a standard technique for solving Maxwell equations numerically. Electric $\vec{E}$ and magnetic $\vec{B}$ field are calculated on a Yee staggered grid. An example of this staggered grid is shown in Fig. 1. Here, the electric fields are calculated in the middle of the faces of a cubical cell while magnetic fields are calculated in the middle of the edges of the cube. This special distribution of gridpoints allows to use central differences which in turn leads to the second order accuracy. Moreover, in most PIC codes, modified or standard leapfrog method is used which means that $\vec{E}$ and $\vec{B}$ are fully or partially updated at both the full time step and the half time step.

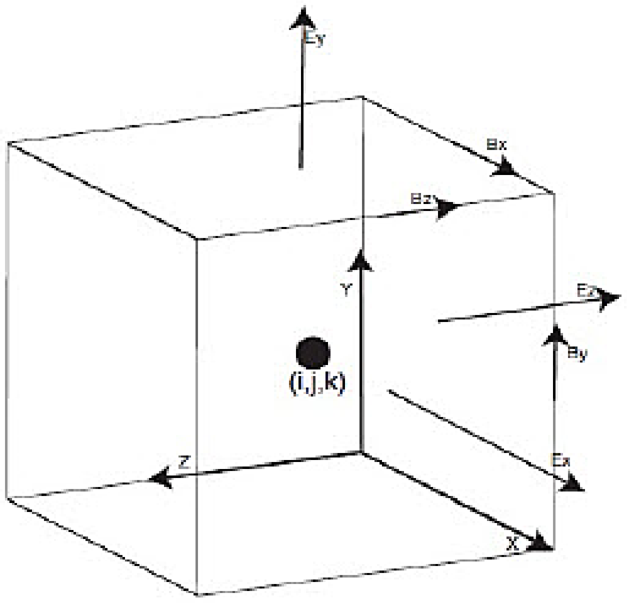


Figure 1: Yee staggered grid in 3D PIC code EPOCH [7]. Electric field components (E_x, E_y, E_z) are calculated in the middle of the faces of a cubical cell while magnetic field components (B_x, B_y, B_z) are calculated in the middle of the edges of the cube.

Particle pusher

The particle pusher solves the relativistic equation of motion under the Lorentz force for each macroparticle in the simulation. Various numerical schemes for the calculation of the motion of particles are used. The standard leapfrog method is often used where particle momentum components are calculated one half-step after the calculation of particle positions. Most PIC codes use the Boris rotation algorithm [8] which splits the equation of motion into separate parts responsible for the acceleration of the particle in the $\vec{E}$ field and the rotation of the particle in the $\vec{B}$ field.

Based on the motion of particles (updated velocities and positions of macroparticles) the currents needed for the Maxwell solver can be calculated using several methods [9, 10, 11]. These charge conserving methods ensure that divergence equation $\nabla \cdot \vec{E} = \rho/\epsilon_0$ (where ρ is the charge density) is always satisfied without solving it.

Shape functions representing macroparticles

To calculate the force acting on a macroparticle, the $\vec{E}$ and $\vec{B}$ fields must be known at the particle position rather than on the fixed spatial grid. Similarly the current has to be deposited at the grid locations to update the $\vec{E}$ field. Since each macroparticle contains many real particles it is necessary to choose a distribution of macroparticle weighting throughout the volume in phase space occupied by a macroparticle (see S_x and S_p in Eq. (2)).

In practice, the only momentum shape function S_p used is the Dirac delta function. Using any other function would result in change of the spatial shape function S_x in time, which is undesirable. Due to the fact that the momentum shape function is always the same, the term shape function will further indicate only the spatial shape function $S_x = S$.

(Spatial) shape functions have to be non-negative symmetric functions with compact support satisfying $\int S(\vec{x})d\vec{x} = 1$. The shape function should also allow for computationally inexpensive interpolation of fields and currents. These requirements are all satisfied by a class of functions called b-splines which have the support comparable to the cell size of the Eulerian grid and are piecewise low order polynomials [12].

The zero-th order b-spline (sometimes called top-hat function) is defined as

$$S_0(x) = \begin{cases} 1, & \text{if } x \leq \frac{\Delta x}{2} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The definition of b-splines of higher (n-th) order is recursive using convolutions:

$$S_n = S_{n-1} * S_0. \tag{4}$$

Since the shape functions defined above refer to the physical shape of macroparticles, interpolation from these shapes to find grid quantities, and the inverse of finding field quantities at macroparticle centers, requires weight functions which are the convolution of the shape function with top-hat function [13].

6.2 Numerical stability and accuracy, boundary conditions

Knowledge of stability conditions is crucial for all numerical methods. In PIC codes, numerical instability mostly manifests itself as a dramatic increase in the energy of the simulated system. However, numerical stability itself does not guarantee physically sound results. Even if the simulation is stable its results may be affected by various numerical artifacts. In order to reach numerical stability and satisfying accuracy of physical phenomena numerically investigated, simulation parameters have to be setup properly and appropriate boundary conditions, shape functions and possible smoothing algorithms have to be applied.

Setting of numerical parameters

PIC algorithm has a few general parameters important for accuracy, stability and computational demands of any simulation. Two most important numerical parameters are the time step Δt and the spatial step Δx also refereed to as cell width or as cell size. In multidimensional simulations, spatial steps in other directions Δy and Δz should be also taken into account. In most cases the spatial steps are all equal, but they can sometimes differ especially in highly computationally demanding PIC simulations where cell sizes are usually larger in transverse direction owing to laser beam propagation direction (e.g. [14]).

Time step is limited by Courant-Friedrichs-Lewy (CFL) condition required for stable propagation of electromagnetic waves. This condition also prohibits macroparticles from moving further than one cell per time step, otherwise the calculation of currents would be inaccurate. In the case of 3D PIC code, this condition sets the time step as follows:

$$\Delta t = \frac{K}{c} \cdot \frac{\Delta x \cdot \Delta y \cdot \Delta z}{\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}}, \text{where } 0 < K < 1. \tag{5}$$

Usually constant K is set close to 1.

The second condition for Δt (6) stems from the requirement to ensure the propagation of electron plasma waves which have to be able to undergo oscillations with the electron plasma frequency ω_{pe}.

$$\omega_{pe}\Delta t \leq 2 \tag{6}$$

Even if this condition is met, the simulated waves have slightly deformed frequency which is undesirable. It is therefore recommended to have even smaller time step than the condition requires.

The spatial step Δx (Δy, Δz) has to be adjusted in order to resolve physical phenomena investigated and/or to reduce numerical heating [8, 15]. In order to resolve physical phenomena at plasma-vacuum boundary, the spatial step in the direction of large density gradient should be less then the plasma skin depth:

$$\Delta x \leq c/\omega_{pe}. \tag{7}$$

Numerical heating can emerge in PIC simulations due to the effects of aliasing of plasma waves [8]. The aliasing occurs when the waves of higher frequencies cannot be represented on a discrete grid, thus they are merged with the waves of lower frequencies. If the condition (8) is not satisfied, modes that are affected by Landau damping are aliased with those that aren't, which leads to nonphysical instability (called numerical heating) since the energy of plasma wave loses is not the same as the energy which macroparticles receive.

$$\Delta x \leq \pi\lambda_{De} \tag{8}$$

Here, λ_{De} is the plasma Debye length. However, higher order shape functions [7] and current smoothing [12] used in modern PIC codes strongly reduce the effect of aliasing and the condition (8) is therefore not so relevant for the stability of PIC codes with higher order shape functions than top-hat function (3).

Finally, PIC simulation should initialize the number of numerical macro-particles per cells N_{ppc} occupied by plasma. This parameter can be hardly defined explicitly and depends on many factors such as the phenomenon in plasma studied, range of plasma densities included in simulation or order of the shape functions used [16]. The setting of this parameter is mostly based on the user's experience with the PIC code.

Choice of appropriate boundary conditions

Boundary conditions have strong impact on the stability and numerical accuracy of the simulations. In fact, boundary conditions are artificial conditions which enables to keep a finite size of simulation domain. There are various boundary conditions for field and particles. Their choice mostly depends on the phenomenon studied together with user's experience. *Periodic* boundaries simply mean that field and/or particles reaching one edge of the domain are wrapped round to the opposite boundary. *Open* boundaries mean that electromagnetic (EM) waves outflowing characteristics propagate through the boundary, whereas particles are simply removed from the simulation when they reach the boundary. EM waves impinging on the boundary should be transmitted with as little reflection as possible. In order to further reduce the amount of reflected EM waves from the boundary, the so-called perfectly matched layer boundary conditions can be used, alternatively called *absorbing* boundaries (e.g., [17]). In PIC simulations with plasma layer reaching the boundary of simulation domain, *reflecting* or *thermal* boundary conditions can be applied. The latter means the using of "thermal bath" of particles at the boundary. When a (macro)particle leaves the simulation domain it is replaced with an incoming particle sampled from a Maxwellian velocity distribution given by a temperature corresponding to that of the initial conditions.

6.3 Advanced algorithms in PIC codes

The core algorithm of PIC code assumes ideal collisionless plasma without any ionization, radiation losses or even quantum phenomena. In order to describes such physics, additional modules using semi-classical approach (in the case of radiation losses of electrons) or Monte Carlo algorithms (in other cases) are mostly included in modern PIC codes.

All these sophisticated approaches are well summarized in the following papers or through web pages:

- T. D. Arber et al.: *Contemporary particle-in-cell approach to laser-plasma modeling*, Plasma Physics and Controlled Fusion **57**, 113001 (2015)

- J. Derouillat et al.: *SMILEI: A collaborative, open-source, multi-purpose* *particle-in-cell code for plasma simulation*, Computer Physics Communications **222**, 351-373 (2018)

- http://www.maisondelasimulation.fr/smilei

PIC codes usually enable to model propagation of laser pulse through underdense plasma due to moving simulation frame. In this case, simulation domain operates as a moving window at constant speed which means that particles are removed from the left hand edge of the domain and new particles are introduced at the right hand edge (assuming that laser pulse propagates from the left to the right).

Even 1D geometry can enable to model obliquely incident laser pulse on target due to Lorentz boosted frame parallel to the target surface [18]. The speedup of PIC simulation can be reached in some cases by using Lorentz boosted frame in the direction of laser pulse propagation [19].

6.4 PIC simulations on large computer clusters

Large PIC simulations involve calculations with $10^9 - 10^{10}$ particles and similar number of cells. This huge amount of particles and cells has high demands on computational time and computer and storage memory. For example, each of our last 3D simulations [20] used more than 1 TB of computer memory and spent more than 10^5 CPU core hours per run. Each simulation run also produced several terabytes of data which should be archived. Since these simulations can run only on large computer clusters, modern PIC codes have to treat with simulation domain decomposition, distributed memory and parallel input/output.

Large computer clusters are usually composed of plenty of nodes, each of them has typically several tens of CPU cores. PIC code has to divide the simulation domain into pieces (subdomains) distributed to each node. Efficient algorithm of PIC code should ensure uniformly distributed workload between all nodes as much as possible. Also the amount of required computer memory per node should be balanced during the whole simulation run. Otherwise the simulation can crash in the overloaded node.

The decomposition of the whole simulation domain should be based on the equal number of macroparticles and cells per cluster node. However, this is mostly impossible, especially in the case of small very dense plasma regions surrounded by large vacuum region (e.g. ionized thin solid foils). Therefore, massively parallel PIC codes define internal rules for domain decomposition taking into account the number of macroparticles and cells with certain weights. Larger weight is usually attributed to the number of macroparticles per node since the calculation of particle motion is usually the most demanding on computational time.

When large data files are produced in PIC simulation run, they can be processed into smaller files with selected data or analyzed directly on a cluster node. Since the amount of stored data is usually limited on computer clusters, they can be transmitted to and archived in data storage centers, some of them are freely accessible for academic purposes.

References

[1] S. Eliezer, *The interaction of high-power lasers with plasmas*. Institute of Physics Publishing, 2002.

[2] P. Gibbon, *Short pulse laser interactions with matter: an introduction*. Imperial College Press, 2005.

[3] P. M. Bellan, *Fundamentals of plasma physics*. Cambridge University Press, 2006.

[4] T. W. Johnston, P. Bertrand, A. Ghizzo, M. Shoucri, E. Fijalkow, and M. R. Feix, "Stimulated Raman-scattering - Action evolution and particle trapping via Euler-Vlasov fluid simulation," *PHYSICS OF FLUIDS B-PLASMA PHYSICS*, vol. 4, no. 8, pp. 2523–2537, 1992.

[5] M. Masek and K. Rohlena, "Intensity dependence of non-linear kinetic behaviour of stimulated Raman scattering in fusion relevant plasmas," *EUROPEAN PHYSICAL JOURNAL D*, vol. 69, no. 4, p. 109, 2015.

[6] A. Pukhov, "Strong field interaction of laser radiation," *REPORTS ON PROGRESS IN PHYSICS*, vol. 66, no. 1, pp. 47–101, 2003.

[7] T. D. Arber, K. Bennett, C. S. Brady, A. Lawrence-Douglas, M. G. Ramsay, N. J. Sircombe, P. Gillies, R. G. Evans, H. Schmitz, A. R. Bell, and C. P. Ridgers, "Contemporary particle-in-cell approach to laser-plasma modelling," *PLASMA PHYSICS AND CONTROLLED FUSION*, vol. 57, no. 11, p. 113001, 2015.

[8] C. K. Birdsall and A. B. Langdon, *Plasma physics via computer simulation*. Taylor & Francis, 2005.

[9] T. Esirkepov, "Exact charge conservation scheme for Particle-in-Cell simulation with an arbitrary form-factor," *COMPUTER PHYSICS COMMUNICATIONS*, vol. 135, no. 2, pp. 144–153, 2001.

[10] J. Villasenor and O. Buneman, "Rigorous charge conservation for local electromagnetic-field solvers," *COMPUTER PHYSICS COMMUNICATIONS*, vol. 69, no. 2-3, pp. 306–316, 1992.

[11] T. Umeda, Y. Omura, T. Tominaga, and H. Matsumoto, "A new charge conservation method in electromagnetic particle-in-cell simulations," *COMPUTER PHYSICS COMMUNICATIONS*, vol. 156, no. 1, pp. 73–85, 2003.

[12] E. Cormier-Michel, B. A. Shadwick, C. G. R. Geddes, E. Esarey, C. B. Schroeder, and W. P. Leemans, “Unphysical kinetic effects in particle-in-cell modeling of laser wakefield accelerators,” *PHYSICAL REVIEW E*, vol. 78, no. 1, 2, p. 016404, 2008.

[13] S. Bennett, C. Brady, H. Schmitz, and C. Ridgers, *Developers Manual for the EPOCH PIC codes*. University of Warwick, 2013.

[14] A. Sharma, “High Energy electron and proton acceleration by circularly polarized laser pulse from near critical density hydrogen gas target,” *SCIENTIFIC REPORTS*, vol. 8, p. 2191, 2018.

[15] H. Ueda, Y. Omura, H. Matsumoto, and T. Okuzawa, “A study of the numerical heating in electrostatic particle simulations,” *COMPUTER PHYSICS COMMUNICATIONS*, vol. 79, no. 2, pp. 249–259, 1994.

[16] V. Kocur, “The effect of macroparticle number on particle-in-cell simulation results and computational demands in laser plasma physics,” Master’s thesis, Czech Technical University in Prague, Faculty of Nuclear Sciences and Physical Engineering, 2017.

[17] T. Umeda, Y. Omura, and H. Matsumoto, “An improved masking method for absorbing boundaries in electromagnetic particle simulations,” *COMPUTER PHYSICS COMMUNICATIONS*, vol. 137, pp. 286–299, JUN 15 2001.

[18] P. Gibbon, A. Andreev, E. Lefebvre, G. Bonnaud, H. Ruhl, J. Delettrez, and A. Bell, “Calibration of one-dimensional boosted kinetic codes for modeling high-intensity laser-solid interactions,” *PHYSICS OF PLASMAS*, vol. 6, pp. 947–953, MAR 1999.

[19] S. F. Martins, R. A. Fonseca, L. O. Silva, W. Lu, and W. B. Mori, “Numerical simulations of laser wakefield accelerators in optimal Lorentz frames,” *COMPUTER PHYSICS COMMUNICATIONS*, vol. 181, pp. 869–875, MAY 2010.

[20] J. Psikal and M. Matys, “Dominance of hole-boring radiation pressure acceleration regime with thin ribbon of ionized solid hydrogen,” *PLASMA PHYSICS AND CONTROLLED FUSION*, vol. 60, no. 4, p. 044003, 2018.

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Chapter 7: Monte Carlo simulations in laser plasma interaction

O. Klimo

7.1

Brief history and introduction to the Monte Carlo method

Monte Carlo (MC) method is a numerical method to search solutions of a mathematical problem using statistical sampling with random numbers. This method is often used for numerical integration, but the convergence is very slow compared to other methods $\mathcal{O}(N^{1/2})$, where N is the number of samples. As the method is so inefficient, it should be used only when other techniques are inefficient too (e.g. integration with complicated boundaries in multidimensional geometry). As the "stochastic" convergence of the method is slow, the meaningful development and application of the method was conditioned by the availability of high computational performance. The first application of this method thus became feasible only after the first powerful computer ENIAC became available [1] and the method was applied to the particle transport problems for predicting the neutron fluxes in fission devices by N. Metropolis, S. Ulam and others in Los Alamos Laboratory in the late 1940's [2]. These calculations were used to simulated neutron multiplication, scattering, propagation, absorption or eventual escape from the medium. The number of applications of the MC method started to grow very quickly and in 1949 there was already the first symposium on the Monte Carlo method [3].

As the MC method is based on the random sampling, the development of the method has also been influenced by the ability of quickly generate random numbers. The RAND Corporation created a machine to generate random numbers using a random pulse generator. It was used to generate a million of random digits with sufficiently good statistical properties. This sequence of random numbers was available as a set of 20000 punched cards in 1950 and was published in a book entitled "A Million Random Digits with 100,000 Normal Deviates" in 1955. In some sense, this was a breakthrough project as it provided a set of high-quality random numbers for the first time to everyone who needed it. Nowadays, the computers provide a fast way of generating a sequence of pseudo-random numbers, which are calculated in a deterministic way from the seed value (the same sequence is obtained for the same seed value). Most commonly, the pseudo-random number generators generate random numbers equally distributed between 0 and 1 and one has to use other methods to get the distribution with desired properties.

7.2

Applications in laser plasma interaction

Intense laser beams while interacting with plasma are capable of producing very energetic particles with energies reaching up to few GeV for electrons [4] and tens or even hundreds of MeV for ions [5]. The electrons may in turn produce energetic photons and the ions or the photons may induce nuclear reactions resulting in fast neutron production. The transport of all these energetic particle fluxes may be simulated by the MC method if the simplifying assumptions of the method given in following sections are fulfilled. We will concentrate here on the particles with energies ranging from few keV to GeV as the energies outside this range are not interesting or accessible with current laser technology.

Concerning the transport of energetic electrons, the MC method can be applied for example to study their penetration into matter, their backscattering, the ionization of the target atoms accompanied by the emission of characteristic photons [6] or Auger electrons and bremsstrahlung radiation. Another application is to study the penetration of fast electron beam and energy deposition in the compressed fuel target in the fast ignition scheme of the inertial confinement fusion like in [7]. In this case however, the transport is not treated using the pure MC method, but a hybrid model including the generation of self-consistent electric and magnetic fields.

The transport of energetic ions can be simulated using the MC approach for example in the context of warm dense matter studies, where the dense matter is rapidly isochorically heated by an intense energetic ion beam [8]. The simulations of energy deposition of proton beams in a human body can be performed in the context of the hadron cancer therapy treatment using laser accelerated protons [9]. Last but not least, energetic ions may induce nuclear reactions and this process including the resulting products can also be simulated with MC codes.

MC simulations of transport of energetic photons are also very important and widely used. For example, one may be interested in designing filters or shielding for laser-plasma experiments and thus to study attenuation of the photon beam. Other important examples are secondary electron production (e.g. for detector design [10]), nuclear activation (e.g. for photon diagnostics) and positron production in matter due to Bethe-Heitler process.

Besides the fast particle transport, MC approach can be used also to incorporate other processes in Particle-in-Cell simulations of laser-plasma interaction. Nowadays, many Particle-in-Cell codes include MC modules for simulating Coulomb collisions, ionization, radiation emission due to non-linear Compton scattering or bremsstrahlung and pair creation via Breit-Wheeler process.

7.3 The simplest MC transport problem - drunk random walk

Before starting with the fast particle transport processes, let us first give an example of a very simple problem, where we can demonstrate the basic MC algorithm. In this problem, the walker starts at the origin of the coordinate system and makes a series of consecutive steps in random directions. This process is often called drunk random walk and in spite of its simplicity, it has a wide range of applications [11] including a simplified model of physical Brownian motion. For simplicity, we will further assume that the length of each step is constant and equal to l. The question we may ask may be for example: "How far on average will the walker be from the origin after N steps?"

The end of each step is the beginning of the next one so we may repeat the same algorithm in a loop. At the beginning of each step, the walker has to choose the direction of his next step randomly. The direction is given by two angles - the polar angle $\theta \in (0, \pi)$ with respect to the direction of the last step of the walker and the azimuthal angle $\varphi \in (0, 2\pi)$ of the orthogonal projection on a reference plane that passes through the current position and is orthogonal to the direction of the last step. In the case of a drunk walk, these angles are random numbers with uniform distribution. However, as the angles are usually given with respect to the current walker's direction, they have to be transformed into the fixed laboratory frame. Assuming that the previous direction of the walker was given by the angles (θ_o, φ_o), the new direction $\mathbf{d_n}$ in the laboratory frame can be obtained as

$$\mathbf{d_n} = \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = \begin{pmatrix} \cos\theta_o \cos\varphi_o & -\sin\varphi_o & \sin\theta_o \cos\varphi_o \\ \cos\theta_o \sin\varphi_o & \cos\varphi_o & \sin\theta_o \sin\varphi_o \\ -\sin\theta_o & 0 & \cos\theta_o \end{pmatrix} \begin{pmatrix} \sin\theta \cos\varphi \\ \sin\theta \sin\varphi \\ \cos\varphi \end{pmatrix}$$

The endpoint of the next step is thus given as

$$\mathbf{x_n} = \mathbf{x_o} + l \cdot \mathbf{d_n}$$

Figure 1 shows an example of the trajectory of ten random walkers each doing 100 steps with $l = 1$.

Of course, the drunk random walk problem is oversimplified and one has to improve the model to be able to describe the transport of fast particles in a realistic way. First of all, the step length is not constant in the particle transport, but it is changing randomly with a given probability distribution depending on the energy of the particle and the properties of the medium where it propagates. Second,

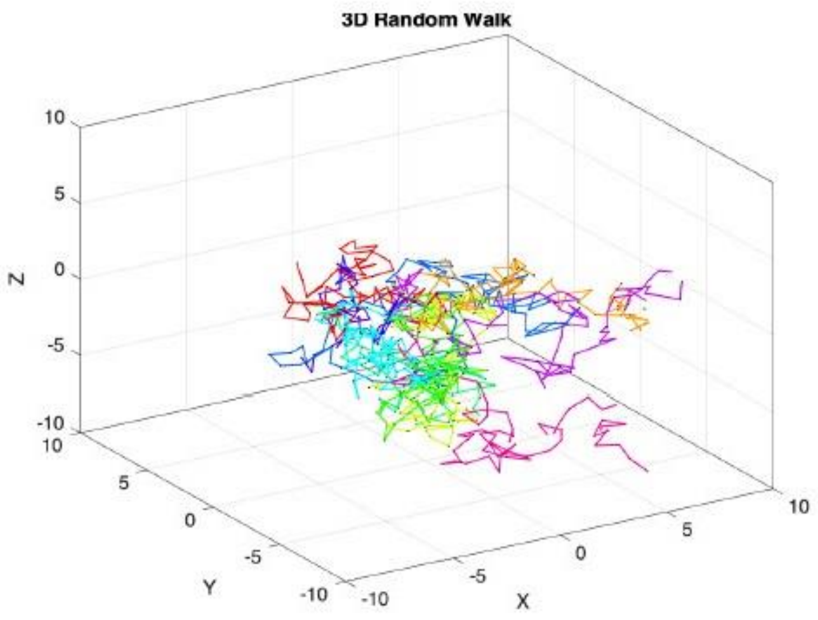


Figure 1: Each color shows a trajectory of a random walker starting at the origin of the coordinate system and doing 100 steps in arbitrary direction with $l = 1$.

the energy of the fast particle is decreasing along its trajectory. Particles may also cross the boundaries between objects with different properties. Third, the polar scattering angle does not have a uniform probability distribution, but this distribution reflects the nature of the physical process taking place at the end point of each trajectory segment. There might be a number of different processes taking place at this end point (e.g. elastic/inelastic scattering, including excitation and ionization, particle absorption, etc.). Last but not least, the simulation is performed to obtaining an estimate of some measurable quantity which is calculated together with the trajectories of the particles.

7.4 Mathematical background - transport equation and its Monte Carlo solution

The particle transport problems can be divided into two categories. The first category includes the so-called ccntinuous path processes. Diffusion is an example of such process which can be described by the Fokker-Planck equation. This description is also appropriate for transport of charged particles in plasmas when the transport is dominated by many small angle Coulomb collisions.

The second category includes the so-called discrete path or jump process. This process can be usually described as Markov chain process in which the current state depends only on the previous state and not on the past history and simulated using MC method. Such description is appropriate e.g. for neutron transport where the neutron velocity is constant, and the neutron propagates along a straight path between individual discrete collisions with the nuclei of the target material. The same applies to photon transport and fast electron and ion transport in neutral target too.

The transport equation which is solved using the MC approach is called the Fredholm integral equation of the second kind. This is a Boltzman type transport equation. Let us make some simplifying assumptions before showing this equation and the way how it is derived and solved at least for the transport of energetic neutrons. First of all, we assume that neutrons are point particles. As they are neutral, they move on a straight-line trajectory between two consecutive interaction events (collisions). Second, the neutron-neutron interactions are neglected and the collisions with other particles are instantaneous. Third, the material properties are isotropic, known and time independent. The problem to be solved is to find some expected or mean value of the neutron density/flux distribution.

The transport is described by two quantities, the neutron angular flux $n(\mathbf{r}, E, \boldsymbol{\Omega}, t)$ (expected number of neutrons at position $\mathbf{r}$ with direction $\boldsymbol{\Omega}$ and energy E at time t per unit volume per unit solid angle per unit energy) and the neutron angular flux $\phi(\mathbf{r}, E, \boldsymbol{\Omega}, t)$ which is the product of n and the neutron velocity v. The transport equation can then be written as [12]

$$\frac{\partial n}{\partial t} = \frac{1}{v}\frac{\partial \phi}{\partial t} = Q - L - R \quad ,$$

where Q is the production rate, L is the leakage rate and R is the removal rate. In the case neutrons, the production rate consists of three terms - an independent external source S, a fission source, which will not be taken into account here and a scattering source. The scattering source Q_s is obtained as

$$Q_s = \int_0^\infty \mathrm{d}E' \int \mathrm{d}\boldsymbol{\Omega}' \; \sigma_s(\mathbf{r}, E') \; C(\mathbf{r}, E' \to E, \boldsymbol{\Omega}' \to \boldsymbol{\Omega}) \cdot \phi(\mathbf{r}, E', \boldsymbol{\Omega}', t) \quad ,$$

where σ_s is the macroscopic scattering cross section and C is the probability of scattering from $(E', \boldsymbol{\Omega}')$ to $(E, \boldsymbol{\Omega})$.

The leakage rate is just due to the flow of particles and is given as a difference between the number of neutrons exiting the volume $\mathrm{d}V$ and the number of neutrons entering the volume $\mathrm{d}V$ per unit time. It can be written as

$$L = \boldsymbol{\Omega} \cdot \nabla\phi$$

The removal rate consists of two terms. The first describes absorption of neutrons in the volume element $\mathrm{d}V$ per unit time and it is proportional to the macroscopic absorption cross section σ_a. The second one is due to scattering of neutrons out from $(E, \boldsymbol{\Omega})$ and it is again proportional to the macroscopic scattering cross section σ_s. The total removal rate can be expressed as

$$R = \sigma_t \; \phi(\mathbf{r}, E, \boldsymbol{\Omega}, t) \quad ,$$

where $\sigma_t = \sigma_a + \sigma_s$.

Further simplifying assumption will help us to obtain the Fredholm integral equation of the second kind and solve it using the MC method. First of all, we are seeking for a stationary solution so we can neglect the time derivative of neutron angular flux. Furthermore, we neglect the absorption at this moment so that $\sigma_t = \sigma_s$. The term $\sigma_s\phi$ will be denoted ψ and called particle collision density (average number of collisions). After integrating the resulting transport equation along characteristics, we obtain the Fredholm integral equation of the second kind in the form

$$\psi(\mathbf{r}, E, \boldsymbol{\Omega}) = \int \mathrm{d}\mathbf{r}' \left[S + \int_0^\infty \mathrm{d}E' \int \mathrm{d}\boldsymbol{\Omega}' \; C(\mathbf{r}', E' \to E, \boldsymbol{\Omega}' \to \boldsymbol{\Omega}) \cdot \psi(\mathbf{r}', E', \boldsymbol{\Omega}') \right] T(\mathbf{r}' \to \mathbf{r}, E, \boldsymbol{\Omega})$$

Let us rewrite the equation in the form [13]

$$\psi(\mathbf{p}) = \int \mathrm{d}\mathbf{r}' \; S(\mathbf{r}') T(\mathbf{r}' \to \mathbf{r}, E, \boldsymbol{\Omega}) + \int \mathrm{d}\mathbf{p}' \; K(\mathbf{p}' \to \mathbf{p}) \cdot \psi(\mathbf{p}') \quad ,$$

where $\mathbf{p} = (\mathbf{r}, E, \boldsymbol{\Omega})$ and $K(\mathbf{p}' \to \mathbf{p}) = T(\mathbf{r}' \to \mathbf{r}, E, \boldsymbol{\Omega}) \; C(\mathbf{r}', E' \to E, \boldsymbol{\Omega}' \to \boldsymbol{\Omega})$. The term C is called the collision kernel and the term T the transport kernel. Before solving the equation with the MC approach let us summarize the main assumptions here. The medium in which the neutrons propagate is static and homogeneous, the transport is time independent and described as a Markov process. The transport of individual neutron is independent of the other neutrons and the neutrons propagate along straight-line trajectories with constant velocity between collisions. These assumptions allow us to apply the superposition principle and expand ψ into a series

$$\psi(\mathbf{p}) = \sum_{k=0}^{\infty} \psi_k(\mathbf{p}) \quad ,$$

where ψ_k is the particle collision density after k collisions. The first term in this series is

$$\psi_0(\mathbf{p}) = \int \mathrm{d}\mathbf{r}' \; S(\mathbf{r}')T(\mathbf{r}' \to \mathbf{r}, E, \mathbf{\Omega})$$

As we are talking about the Markov process, we can write

$$\psi_k(\mathbf{p}) = \int \mathrm{d}\mathbf{p}' \; K(\mathbf{p}' \to \mathbf{p}) \cdot \psi_{k-1}(\mathbf{p}')$$

Here $\psi_{k-1}(\mathbf{p}')$ is the particle collision density after $(k-1)$ collisions at $\mathbf{p}'$ and the $K(\mathbf{p}' \to \mathbf{p})$ is the conditional probability of the k-th collision at $\mathbf{p}$ provided that the $(k-1)$ collision was at $\mathbf{p}'$.

This equation can be solved with the MC approach as follows [14]:

1. Randomly sample $\mathbf{p}'$ from $\psi_{k-1}(\mathbf{p}')$
2. Randomly sample $\mathbf{p}$ from $K(\mathbf{p}' \to \mathbf{p})$
3. If $\mathbf{p} \in (\mathbf{p_i} - \mathrm{d}\mathbf{p}, \mathbf{p_i} + \mathrm{d}\mathbf{p})$ then $\psi_k(\mathbf{p_i}) = \psi_k(\mathbf{p_i}) + 1$

The previous steps $1-3$ have to be repeated N times and the final result is $\psi_k(\mathbf{p_i}) = \psi_k(\mathbf{p_i})/N$. Finally, one has to repeat the same approach for $k = 0, 1, 2, ...$

However, instead of calculation separately ψ_k for every k, we can use a better approach and write ψ_k as

$$\psi_k(\mathbf{p}) = \int ... \int \mathrm{d}\mathbf{p_0} ... \mathrm{d}\mathbf{p_{k-1}} \; K(\mathbf{p_{k-1}} \to \mathbf{p}) ... K(\mathbf{p_0} \to \mathbf{p_1}) \cdot \psi_0(\mathbf{p_0})$$

This equation can be solved using the "histories" - sequences of states going from the source up to the "absorption" state. The histories are generated as follows. The source is randomly sampled using the distribution $\psi_0(\mathbf{p_0})$ and the history is given by random sampling the k transitions with the distribution given by $K(\mathbf{p}' \to \mathbf{p})$. When the absorption state is reached the following transition probabilities are equal to 0 and we can terminate the history. Having M histories, we can measure quantities (for instance a quantity denoted A) as follows

$$A = \int \mathrm{d}\mathbf{p} \; A(\mathbf{p})\psi(\mathbf{p}) = \frac{1}{M} \sum_{m=1}^{M} \left(\sum_{k=1}^{\infty} A(\mathbf{p_{k,m}}) \right)$$

An example of such measurable quantity often used in dosimetry is the linear energy transfer (LET) which is equal to the amount of energy ionizing particle transfers to the material per unit distance and thus it represents the effect of ionizing radiation on matter where it propagates.

7.5 Simple algorithm for fast electron transport

In this section, we will describe a simplified model of fast electron transport based on [15]. The validity of this model is limited by the cross sections and the stopping power formula used. Nevertheless, it is simple and instructive and that's why it is presented here.

The model is based on the so-called single scattering approach in which all elastic scattering events are explicitly included. Elastic scattering events are those in which the initial and the final quantum state of the target atom is the same. The scattering angle in these collisions can be quite large and we can assume that there is no significant energy loss of the incident fast electron. These scattering events can be described in the keV energy range by the screened Rutherford cross section (a better cross section for a wider range of energies can be found e.g. in [16]). The formula for the total cross section taken from [15] is

$$\sigma_R = 5210 \frac{4\pi}{\alpha(E,Z)(1+\alpha(E,Z))} \frac{Z^2}{E^2} \left(\frac{E+511}{E+1024}\right)^2 \left[\frac{\text{barn}}{\text{atom}}\right] ,$$

where E is the electron kinetic energy in keV, Z is atomic number of the target atom and $\alpha(E,Z)$ is the screening parameter. This is the total elastic scattering cross section integrated over all scattering angles and it can be used to calculate the mean free path.

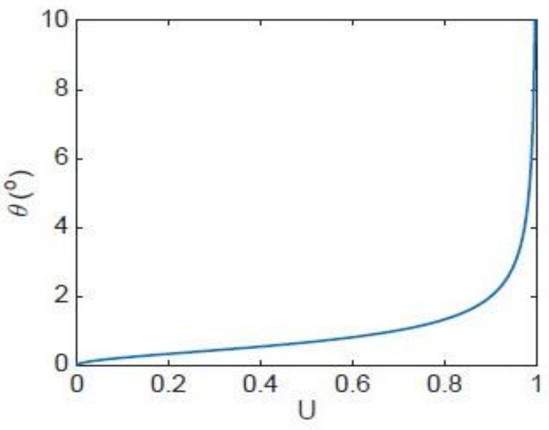


Figure 2: The dependence of the polar scattering angle sampled from the distribution based on the elastic scattering screened Rutherford cross section on the random number U uniformly distributed in the interval $(0, 1)$ for a 100 keV electron and $Z = 1$.

The inelastic scattering events in which the target atom is excited or ionized are neglected in this simple approach. The energy loss in these events would be taken into account separately using the continuous slowing down approximation and the scattering angle in these collisions is quite small, i.e. the events with large scattering angle are rare [15].

The average distance traveled by the electron between two consecutive elastic scattering events s is expressed using the mean free path, $\lambda_{mfp} = \sigma_R N_t$, where N_t is the number of target atoms per unit volume. The probability $P(x)$ of not having an elastic collision after a distance x is described by the Poisson process and thus $P(x) = \exp(-x/\lambda_{mfp})$. The distance traveled is sampled from the probability distribution $P(x)$ using the cumulative distribution function and the inversion method. The cumulative distribution function is

$$F(x) = \frac{\int_0^s \mathrm{d}x\ P(x)}{\int_0^\infty \mathrm{d}x\ P(x)} = 1 - \exp\left(-\frac{s}{\lambda_{mfp}}\right)$$

The distance traveled can thus be sampled using the random number uniformly distributed in the interval $(0, 1)$, U, which is a random sample of $F(x)$, and thus $s = -\lambda_{mfp} \ln(1-U) = -\lambda_{mfp} \ln(U)$.

The scattering angles in the elastic scattering event are calculated from the screened Rutherford cross section differential in the polar scattering angle. This is done again using the cumulative distribution function and the inversion method and the sample of the polar scattering angle is obtained as

$$\cos\theta = 1 - \frac{2\alpha(E,Z)U}{1+\alpha(E,Z)-U}$$

The azimuthal scattering angle is random with uniform distribution due to the rotational symmetry of the collision frame and thus $\varphi = 2\pi U$. An example of the polar scattering angle dependence on the random number U is shown in Figure 2 for an electron with the kinetic energy 100 keV and $Z = 1$.

The energy loss of electrons is due to inelastic collisions (dominates for lower E and lower Z targets) and bremsstrahlung emission (photon emission during the elastic collision due to electron acceleration). Simulation of all energy loss events would be very complicated and time consuming. Moreover, the energy loss in individual collisions is usually quite low (the so-called soft events). Therefore, a continuous slowing down approximation is often used where the energy loss is averaged per unit distance traveled using the so-called stopping power. Bethe formula for the stopping power is often used for electron transport in solid neutral targets. Let us give an example of a simple formula valid for the keV energy range [15]

$$\frac{\mathrm{d}E}{\mathrm{d}S} = -78500 \cdot \frac{Z}{AE} \cdot \ln\left(\frac{1.166E}{J}\right) \quad , S = s \cdot \rho \quad ,$$

where s is the distance traveled in cm, ρ is the target density in g/cm^3, A is the atomic weight and both E and the mean ionization potential J are in the units of keV. This formula is applicable only for $E \gg J$. The tables of stopping power of fast electrons can be found in the NIST database ESTAR [17] for a wide range of energies and target materials.

The MC simulation algorithm for a single electron can be summarized as follows:

1. Calculate λ_{mfp} based on σ_R
2. Randomly sample s using the exponential distribution with λ_{mfp}
3. Update the coordinates of the electron
4. Decrease E based on the $\mathrm{d}E/\mathrm{d}S$ and the distance traveled s
5. Randomly sample θ and φ
6. Transform these angles into the laboratory frame

The steps $1-6$ have to be repeated until the electron leaves the target or its energy decreases below some threshold value.

7.6 Advanced techniques - condensed history, variance reduction

The basic single scattering algorithm described in the previous section has a limited applicability and it is not very efficient. With the continuous slowing down approximation used, we know from the beginning what would be the distance traveled by the electron before it is "absorbed". Instead of simulating all the single scattering events with many small angle deflections, one may divide the electron trajectory into a fixed set of segments and take into account the average effect of electron scattering in these individual trajectory segments (i.e. group many collisions into one where the deflection angle corresponds to the cumulative effect of scattering in all those collisions). This approach is called condensed history technique (sometimes also referred to as multiple or plural scattering). The step size is no longer related to the mean free path, but it is often related to the constant energy loss per step, e.g. $\Delta E/E = 4\%$ is often used. Depending on the application, the condensed history simulation may outperform the single scattering approach by a factor $10^3 - 10^5$.

The condensed history class II scheme is implemented in many MC codes nowadays. This scheme simulates all sub-threshold soft events using multiple scattering with the continuous slowing down approximation while the hard events where the energy of the bremsstrahlung photon is higher than the threshold value E_γ or the kinetic energy of the knock-on electron coming from ionization is higher than E_δ are simulated explicitly by creation and transport of these newborn high-energy particles.

The classical so-called analog MC method works well if the probability of the event in which we are interested is not very low. Otherwise the method is too time consuming and provides bad statistics, i.e. high variance of the results. However, there exist variance reduction techniques, which help to reduce this variance significantly and thus they increase the efficiency of the MC method too. These techniques can be divided into two main groups: 1) probability distribution function (PDF) biasing or 2) particle splitting. In the PDF biasing technique, the probability of unlikely events of interest is artificially increased. However, each particle is assigned with a different statistical weight w to obtain unbiased results and the weights are related as

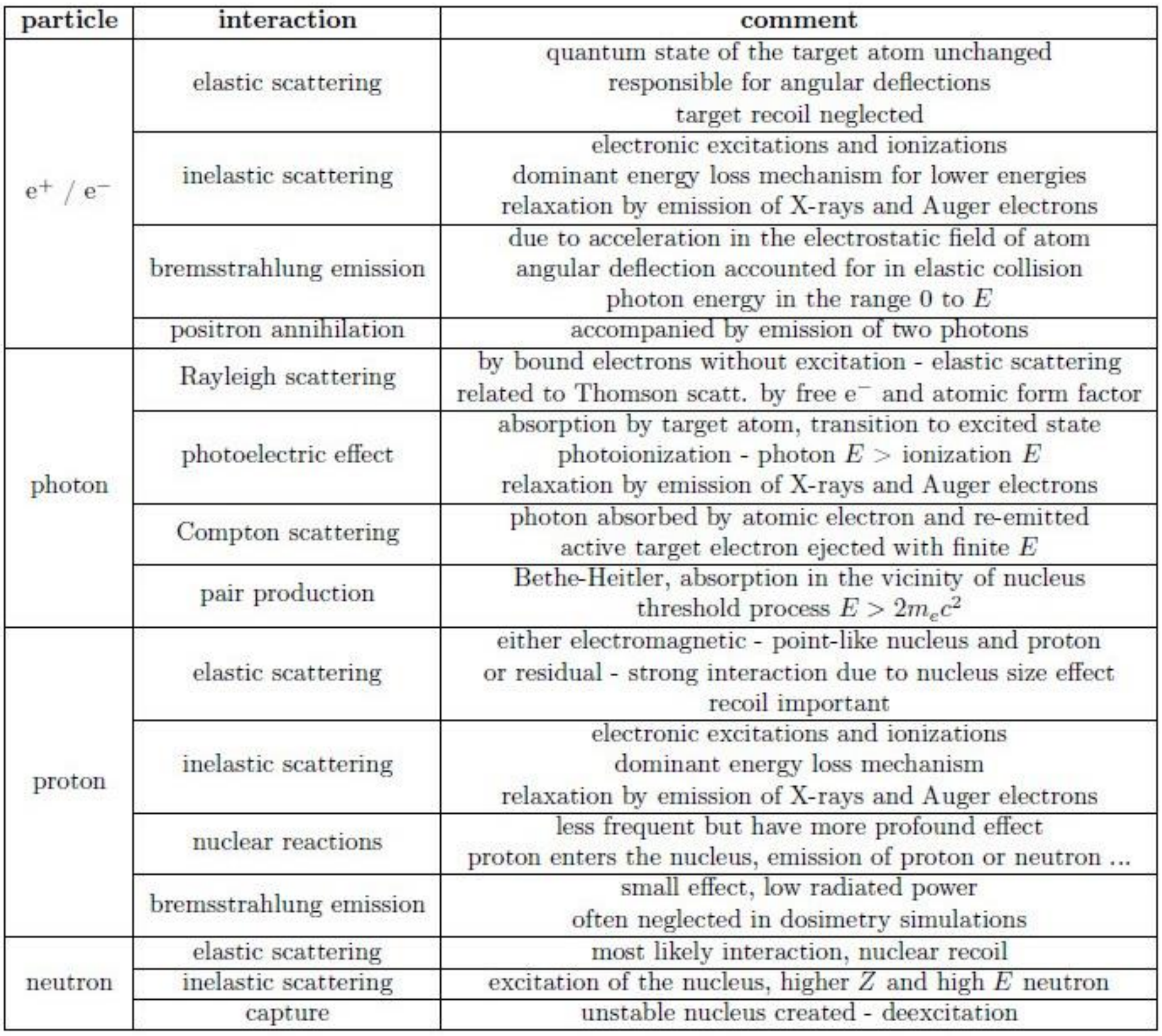
$$w_{biased} = w_{unbiased}\frac{PDF_{unbiased}}{PDF_{biased}}$$

If the value of PDF_{biased} is artificially increased, w_{biased} is decreased proportionally so the averaged outcome of the unbiased probability distribution function is preserved. Let us give an example based on a simple technique called implicit capture (survival). In this technique, the particle may be forced to continue to propagate instead of being "absorbed" but with a reduced weight w_{biased}. If the value of w_{biased} is already too low, the particle may play a Russian roulette with two possible outcomes and their corresponding probabilities. The first outcome is the absorption of the particle, the second one is increase of its weight.

The particle splitting technique can on the other hand be used to increase the survivability of important particles and eliminate less important ones. For example, we may be interested in the response of a detector, which is surrounded by shielding walls. The particles which reach the detector are the most important for us, while the ones backscattered from the shielding walls are much less important. In this case, we may apply the technique of geometric splitting with Russian roulette, where each volume is assigned a different importance. The particles crossing the boundary from the less important to the more important region will be split and their weight will be correspondingly decreased. The particle propagating into a less important region will play a Russian roulette and it will be eliminated with a certain probability, while the weight of the surviving particles will be increased. There are many other variance reduction techniques. Their brief overview is given e.g. in [18].

7.7 Main particle interactions and the most important MC codes

In the table below, we provide a short summary of the main interaction processes of the particles in the energy range of interest in the context of laser-plasma interaction.



particle	interaction	comment
e^+ / e^-	elastic scattering	quantum state of the target atom unchanged responsible for angular deflections target recoil neglected
	inelastic scattering	electronic excitations and ionizations dominant energy loss mechanism for lower energies relaxation by emission of X-rays and Auger electrons
	bremsstrahlung emission	due to acceleration in the electrostatic field of atom angular deflection accounted for in elastic collision photon energy in the range 0 to E
	positron annihilation	accompanied by emission of two photons
photon	Rayleigh scattering	by bound electrons without excitation - elastic scattering related to Thomson scatt. by free e^- and atomic form factor
	photoelectric effect	absorption by target atom, transition to excited state photoionization - photon $E >$ ionization E relaxation by emission of X-rays and Auger electrons
	Compton scattering	photon absorbed by atomic electron and re-emitted active target electron ejected with finite E
	pair production	Bethe-Heitler, absorption in the vicinity of nucleus threshold process $E > 2m_ec^2$
proton	elastic scattering	either electromagnetic - point-like nucleus and proton or residual - strong interaction due to nucleus size effect recoil important
	inelastic scattering	electronic excitations and ionizations dominant energy loss mechanism relaxation by emission of X-rays and Auger electrons
	nuclear reactions	less frequent but have more profound effect proton enters the nucleus, emission of proton or neutron ...
	bremsstrahlung emission	small effect, low radiated power often neglected in dosimetry simulations
neutron	elastic scattering	most likely interaction, nuclear recoil
	inelastic scattering	excitation of the nucleus, higher Z and high E neutron
	capture	unstable nucleus created - deexcitation

Table 1: Review of the most important interactions of electrons, positrons, photons, protons and neutrons in the energy range of interest keV-GeV.

The most widely used MC particle transport codes which are available:

- EGSnrc
  - https://nrc-cnrc.github.io/EGSnrc/
  - photons, electrons and positrons with kinetic energies between 1 keV and 10 GeV
- GEANT4
  - http://geant4.web.cern.ch
  - electron, ion, muon, gamma ray, electromagnetic (EM), hadronic, and optical photons (many kinds of particles) and very wide energy range
- MCNP
  - https://mcnp.lanl.gov
  - neutrons up to 20 MeV for all isotopes and up to 150 MeV for some, photons from 1 keV to 100 GeV and electrons from 1 keV to 1 GeV
- PENELOPE
  - https://www.oecd-nea.org/tools/abstract/detail/nea-1525
  - electron/positron-photon transport in energy range between 50 eV and 1 GeV
- FLUKA
  - http://www.fluka.org
  - in particular charged hadrons, neutrons, electrons, photons, heavy ions but also other particles, very wide energy range

References

[1] N. Metropolis, "THE BEGINNING of the MONTE CARLO METHOD," *Los Alamos Science* **12**, 125 (1987), https://library.lanl.gov/cgi-bin/getfile?00326866.pdf.

[2] N. Metropolis and S. Ulam, "The Monte Carlo method," *J. Amer. Statistical Assoc.* **44**, 335 (1949).

[3] M. Mayer, "Report on a Monte Carlo calculation performed with the ENIAC," in *Monte Carlo Method*, ed. by A.S. Householder, G.E. Forsythe and H.H. Germond, (U.S. Nat. Bur. Standards Applied Math. Series, No. 12) pp. 307-20 (1951).

[4] A. J. Gonsalves *et al.*, "Petawatt Laser Guiding and Electron Beam Acceleration to 8 GeV in a Laser-Heated Capillary Discharge Waveguide," *Phys. Rev. Lett.* **122**, 084801 (2019).

[5] A. Higginson *et al.*, "Near-100 MeV protons via a laser-driven transparency-enhanced hybrid acceleration scheme," *Nature Communications* **9**, 724 (2018).

[6] J. Limpouch *et al.*, "Numerical Studies on the Ultra-short Pulse K-α Emission Sources Based on Femtosecond Laser-target Interactions", *Laser and Particle Beams* **22**, 147 (2004).

[7] J. J. Honrubia *et al.*, "Hybrid simulations of fast-electron transport in conducting media," *Laser Particle Beams* **22**, 129 (2004).

[8] P. K. Patel *et al.*, "Isochoric Heating of Solid-Density Matter with an Ultrafast Proton Beam", *Phys. Rev. Lett.* **91**, 125004 (2003).

[9] S. V. Bulanov and V. S. Khoroshkov, "Feasibility of using laser ion accelerators in proton therapy," *Plasma Physics Reports* **28**, 453 (2002).

[10] S. Singh *et al.*, "Compact high energy x-ray spectrometer based on forward Compton scattering for high intensity laser plasma experiments," *Rev. Sci. Instrum.* **89**, 085118 (2018).

[11] Random walk, Applications `https://en.wikipedia.org/wiki/Random_walk#Applications`

[12] Y. Wu, "Fusion Neutronics," Springer (2017).

[13] W. L. Dunn and J. K. Shultis, "Exploring Monte Carlo Methods," Academic Press (2012).

[14] P. Vaz, "Neutron transport simulation," *Radiation Physics and Chemistry* **78**, 829 (2009).

[15] D.C. Joy, "Monte Carlo Modeling for Electron Microscopy and Microanalysis," Oxford (1995).

[16] N. F. Mott and H. S. F. Massey, "The Theory of Atomic Collisions," Pergamon, Oxford (1965).

[17] ESTAR: Stopping Powers and Ranges for Electrons, `https://physics.nist.gov/PhysRefData/Star/Text/method.html`

[18] A. Haghighat, "Monte Carlo Methods for Particle Transport," CRC Press, Taylor & Francis Group (2014).

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Chapter 8: Simulations of HD behavior in high-intensity short-pulse LPI

J. Pasley

8.1 Introduction

When high-intensity laser pulses interact with matter, they can generate strong shock waves. This article is particularly concerned with the case in which a high-intensity laser pulse interacts with plasma that is over-dense to the laser. That is to say that the electron gas density is sufficiently high, in the media, that the bulk of the material lies beyond the relativistic critical surface. The relativistic critical surface is the surface at which the relativistically corrected electron plasma frequency is equal to the frequency of the laser radiation. The electron number density at which this occurs is given by the equation

$$n_{c\gamma} = \frac{\langle\gamma\rangle m_e \varepsilon_0 \omega_L^2}{e^2} = \langle\gamma\rangle n_c$$ **Equation 1**

Where $\langle\gamma\rangle = (1 + a_0^2/2)^{1/2}$ is the relativistic gamma factor introduced by the oscillation of the electrons in the electromagnetic field of the laser beam and $a_0 = eA_0/m_e c^2$ and ω_L are the relativistically normalized laser amplitude and the laser frequency respectively.

At such high densities, the laser beam cannot propagate. Hydrodynamics is driven directly by the light pressure, but also at depth within the plasma by the relativistic electron beam that is accelerated by the action of the laser near the critical surface. This electron beam can accelerate the plasma by non-uniform Ohmic heating and consequent introduction of pressure gradients, as well as by the **j x B** force induced.

In this paper we will firstly put our work on short-pulse laser generated shock waves into context by summarizing the Fast Ignition approach to Inertial Confinement Fusion and the role of fast heating effects. Secondly we will describe our theoretical investigation of the hydrodynamic processes caused by the rapid heating of a plasma by a fast electron beam. Finally, we will report on a recent experiment in which detailed measurements of a shock wave generated by a short-pulse laser were made.

8.2 Fast ignition

In the fast ignition approach [1] to inertial confinement fusion (ICF) [2] a laser with a focused intensity on the order of 10^{25}W/m^2, and pulse length on the order of 10ps, interacts with a dense plasma target containing fusion fuel. The dense target is formed by the implosion of a spherical shell containing cryogenically frozen deuterium-tritium (DT). The implosion is driven by a pulse of radiation lasting approximately 10ns. This radiation may be in the form of either soft x-rays or focused laser beams. The incident intensity at the surface of the capsule is roughly 10^{18}W/m^2. At the end of the implosion process, when the fuel “stagnates” up against itself at the centre, the DT is compressed to thousands of times its normal solid density of 220kg/m^3.

A range of variants upon the fast ignition principle have been thought of [3,4,5], however the basic principle is similar in all cases. The secondary laser acts, by some means, to heat a small portion of the imploded DT fuel to the conditions required for thermonuclear ignition.

Ignition in the context of ICF takes place in a hotspot. The hotspot is a region of fuel in which, as the name suggests, the temperature is significantly higher than in the bulk of the DT fuel. This hotspot may be formed by compression, as in the case of conventional ICF, or by means of a secondary driver, as in the case of fast ignition. Ignition occurs when the hotspot is able to self-heat from the conditions in which it is left by the driver ($T_{ion} \approx$ 10-12keV) to a much higher temperature of around 70keV. Once the hotspot is burning vigorously, the power radiated into the surrounding “cold” fuel, in the form of thermonuclear alpha particles, is sufficient that the burn readily spreads. It is important that the bulk of the fuel is heated to ignition temperatures by the spreading thermonuclear burn wave rather than by the driver. If this were not the case then the driver would have to be excessively large and the available energy gain insufficient for the purposes of electrical power production.

In the case of fast ignition, the hotspot is formed in a region near the surface of the compressed fuel mass. The hotspot is surrounded by lower density material of similar temperature on one side, and by material of similar density and much lower temperature on all others. This means that, during its formation, the hotspot is far from being in pressure equilibrium with the surrounding plasma and will tend to expand rapidly. Where the hotspot faces the cold dense fuel, this expansion is led by strong shock waves.

In order that we can properly formulate the ignition problem, it is critical to quantify both the hotspot expansion and shock wave generation and propagation processes. The driver must deposit its energy in a hotspot that is continuously evolving in size and density, and the thermonuclear burn rates at any point in time are a strong function of the density profile of the DT fuel. The situation is complicated by the fact that in fast ignition the intense laser pulse acts to drive enormous currents as well as ultra-strong magnetic and electric fields. In order to tackle the evolving hydrodynamics therefore we must take into account the possibility of magnetohydrodynamic effects such as the ***JxB*** force on the fluid.

In addition to the evolution of the hotspot, there is also the issue of hydrodynamic motion in other regions of the target. The fast electrons will heat any material present between the point of laser absorption and electron deposition in the hotspot. The heating duration (10-20ps) is sufficient for some regions to experience significant hydrodynamic motion during this time, which in turn can affect the transport of fast electrons to the hotspot. This effect has not been thoroughly studied.

To summarize, the pursuit of fusion energy via Fast Ignition ICF requires one to consider a situation where shock wave generation by rapid heating of a plasma with a high energy relativistic electron beam is important and impinges on many facets of the whole problem. This largely motivates our current efforts to study this form of shock wave generation.

8.3 Modelling of shock waves generated by intense laser-plasma interaction

When a laser of focused intensity 10^{22}-10^{25}W/m^2 interacts with dense fuel, it accelerates electrons from the plasma background to approximately MeV energies. These electrons propagate forward into the dense fluid beyond the critical surface. These high energy electrons have relatively long mean free paths and, to a fair approximation, propagate ballistically. However, in order to conserve charge, a so-called "return current" is drawn from the plasma background. This return current is collisional, and is therefore subject to the resistivity of the medium. We can therefore write,

$$j_{fast} + j_{return} \approx 0 \qquad \textbf{Equation 2}$$

and thence,

$$E = \eta j_{return} = -\eta j_{fast} \qquad \textbf{Equation 3}$$

where η is the resistivity of the background plasma. For plasma at temperatures in excess of 100eV (11.8 Million Kelvin), at solid densities, the resistivity is given approximately by the Spitzer formulation as:

$$\eta = 10^{-4} \frac{Z \ln \Lambda}{T_{eV}^{3/2}} \ \Omega\text{m}$$ **Equation 4**

Where Z is the atomic number and ln Λ is the dimensionless plasma parameter. The background plasma is heated such that,

$$\frac{\partial T}{\partial t} = \frac{\eta j^2}{k_B n_e}$$ **Equation 5**

Since the current is not uniformly distributed throughout the plasma, this Ohmic heating leads to pressure gradients that drive expansion and shock wave formation. Furthermore, by combining Faraday's Law and Ohm's Law, it can be seen that the growth of the magnetic field in the plasma (assuming current flowing along the y-axis) is given by:

$$\frac{\partial B_Z}{\partial t} + v_x \frac{\partial B_Z}{\partial x} = -\frac{\partial(\eta j_y)}{dx}$$ **Equation 6**

This results in a ***JxB*** force on the background plasma in addition to the influence of the kinetic pressure.

8.4 Magnetohydrodynamics simulations of laser-generated electron beam driven strong shock waves

A magnetohydrodynamics simulation code has been written [6], based upon the methods described by Ziegler in reference 7. As can be seen from figure 1, the generation of strong shock waves is anticipated on timescales of only a few picoseconds.

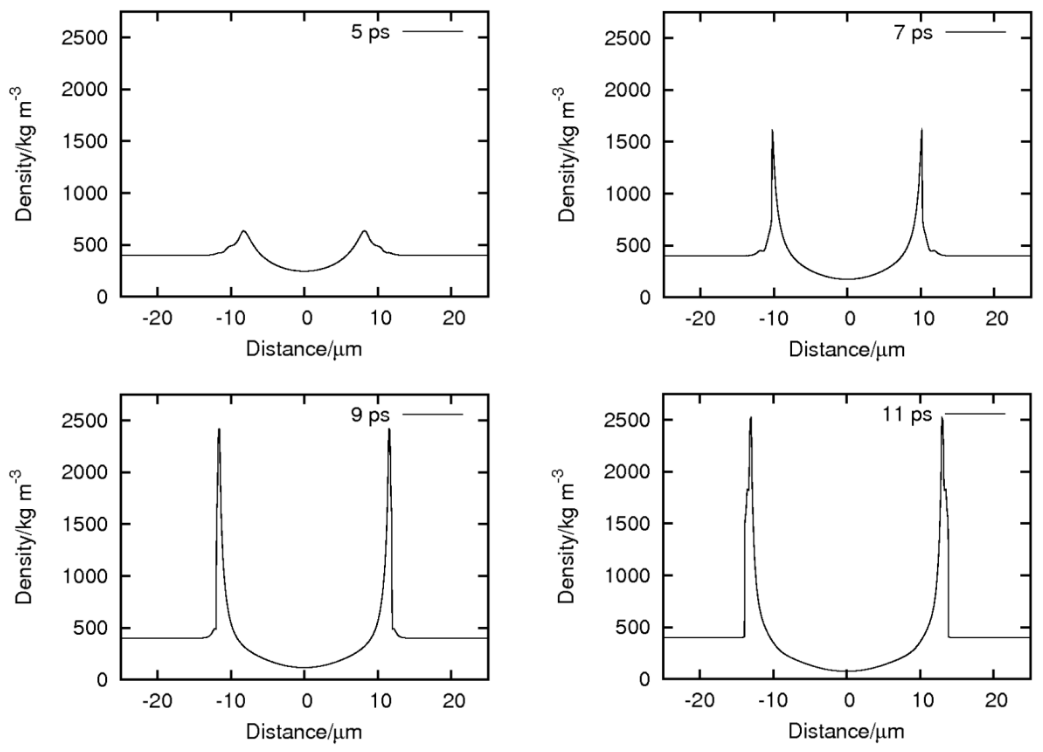


Figure 1. Showing the formation of strong shock waves driven by a cylindrically symmetric continuous current profile with amplitude of 6x10^{17}Am^{-2}, Gaussian distributed about the cylinder-axis with a FWHM of 7µm. Background is hydrogen at 400kg/m^{3}. The heated region is centred on r=0. The outward propagating density features clearly show the rapid formation and propagation of strong shock waves from the periphery of the heated region.

A wide ranging parameter scan was performed in order to determine the timescales for shock wave formation with a range of background plasma densities and drive currents. It is important to consider a wide ranging parameter space, since, even in the case of fast ignition, electrons must traverse a wide range of plasma densities between the point where they are generated by the laser and their being absorbed in the dense fuel. The results of this study are shown in figure 2. Here it is assumed that shocks are formed whenever material is accelerated to above the sound speed in the background plasma.

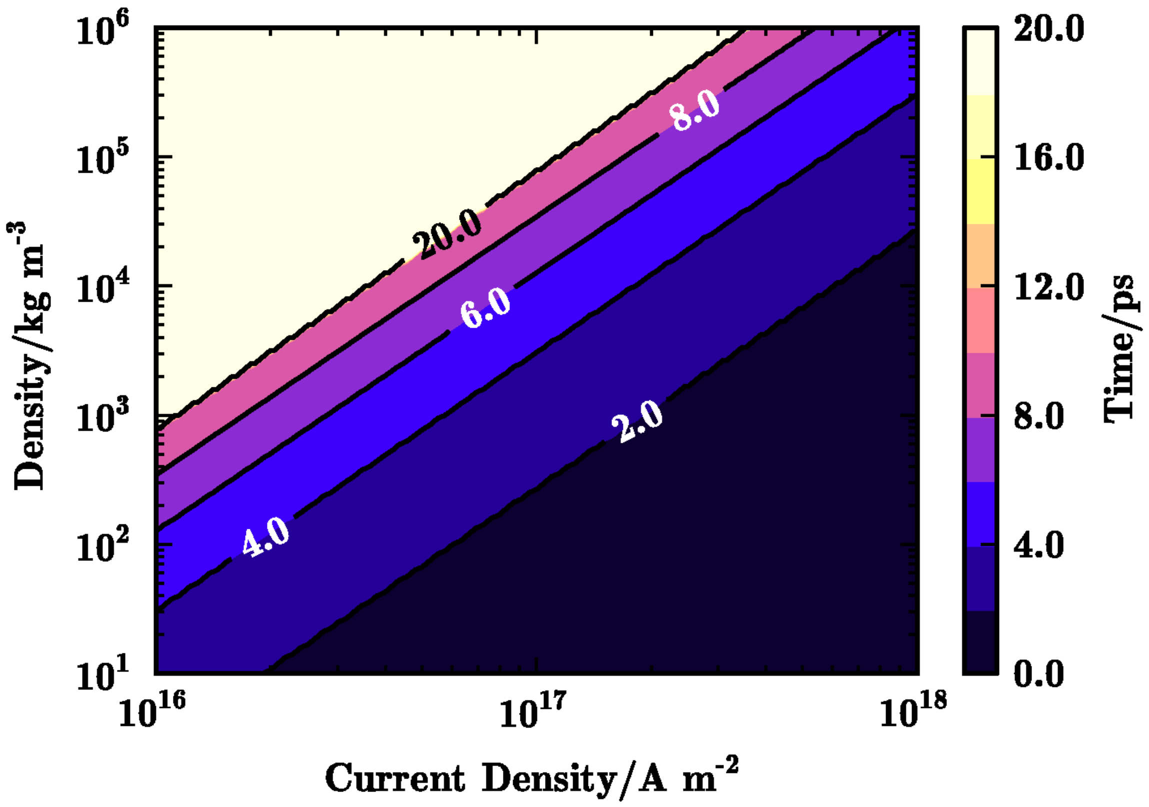


Figure 2. Timescales for shock wave formation in a variety of different current and plasma density regimes.

In fast ignition, it is necessary to raise a region of fuel with a ρr of approximately 5kg/m^2 to a temperature of around 12keV. Assuming a cylindrical hotspot with initial density of approximately 3x10^5 kg/m^3 and given that the heat capacity of DT is around 100GJ/kg/keV, then this correlates to raising 8.73μg of DT to 12keV. This requires approximately 10.5kJ of energy to be deposited. Presuming a time scale for depositing the energy of around 20ps, this necessitates a heating power of around 0.5PW must be supplied to the cylinder. Assuming the energy enters the cylinder from one end, then the power density must be around 6x10^{23}W/m^3. In order that the heating be localized to the hotspot, electrons with energy of around 1MeV must be employed. This gives a total minimum beam current of around 6x10^{17}A/m^2. The data shown in figure 2 suggests that shock waves would form around 6ps into such a 20ps ignition pulse. Therefore, at the moment of ignition the hotspot would be rarefied and bounded by strong shock waves where it interfaced with the cold dense fuel mass. It is clear, therefore, that in modelling fast ignition it is important to properly take into account the effects of such shock waves upon the ignition process, and also in the deposition of energy by the driver.

These calculations also suggest the utility of short pulse lasers for generating extremely strong shock waves for laboratory investigation. For instance, the shock wave in figure 1 is propagating at approximately 900km/s. Furthermore, the results of the simulations, taken together with other analyses presented in reference 6 clearly demonstrate that MHD effects play no significant role in the parameter range explored here, and that the driving of shock waves is due entirely to the steep kinetic pressure gradients accelerating the fluid.

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Chapter 9: Parallel algorithms for LLP simulations

J. Vyskocil

The size of many problems in contemporary physics which have to be solved numerically, by far surpasses the computational power of a typical desktop computer. Moreover, as nowadays even your everyday phone is likely running a multi-core processor, in most cases of smaller calculations, efficient parallelization is the way to harness the full power of your machine. In the course of rewriting the serial algorithms into their efficient parallel counterparts, some of your design choices would nicely follow from the theory of parallel computation, while other would, sometimes counter-intuitively, come from the considerations given by the hardware design. The theory and the machines evolved together, and they continue to influence one another after more than 70 years after the commissioning of the grandfather of all modern computers, the "Electronic Numerical Integrator and Calculator", known as the ENIAC [1]

1 Parallel computer architectures

Within the framework of physics, we are concerned with the topic of massively parallel scientific computing, i.e. processing data on a large number of processors all (or most) of which perform computations at the same time. First of the two main massively parallel computing paradigms is grid computing, where individual computers called "nodes" are distributed, possibly all around the globe. These can be heterogeneous, i.e. each node can run a different processor architecture, have a different amount of memory, and run a different operating system, and are often connected with general-purpose networks such as the Internet. The other is cluster computing, where the nodes usually reside in the same data centre, forming a single super-computer. They are mostly homogeneous, i.e. each node has the same processor, the same amount of memory, and run the same version of the operating system, and connected with specialized fast networks (10 Gbit/s Ethernet, Infiniband, etc.). Nowadays clusters are composed of roughly 10 - 10,000s of nodes (the K-Computer has 80,000 nodes) which provide a total of 1,000s – 1,000,000s of cores (the Tianhe-2 computer has 3,120,000 cores) [2]. To use this combined computing power, we need efficient algorithms which scale well with the rising number of cores.

In the most basic sense, an algorithm is a sequence of operations. In the case of computer simulations in physics, these are manipulating numbers in computer's memory. A sequential, also called "serial", algorithm is one where operations are performed one after another, while a parallel is one where some operations are being performed at the same time. These numbers represent data from the physical world – scalars, vectors, matrices, scalar fields, vector fields, tensor fields, etc. stored as numbers in the computer's memory – a single place, a place together with the number of succeeding elements, or several disjoint areas, and translated into data types defined by the programming language – integers, floating point numbers, pointers (these don't have a true analogue in the physical world), arrays, vectors, lists, etc.

The way we write the algorithms nowadays is surprisingly strongly influenced by the earliest computer designs – the Von-Neumann architecture realized in the "Electronic Discrete Variable Automatic Computer" (EDVAC) in 1951, which is still the base of all contemporary computers [1]. It consists of an input and output device, the central processing unit (CPU), and the memory unit.

The memory contains the data and the instructions. These are passed through a "bus" between the memory and the CPU which executes the instructions to operate on the data. The instructions are executed in discrete cycles, and the timing of these cycles is driven by an internal clock. As you increase the speed of the clock, the calculations get faster.

In 1971, Intel created the first microprocessor – an integrated circuit made of transistors on a single chip that contained the CPU, and controllers for the memory access, the inputs, and the outputs [1]. Since then, the computing power of microprocessors has been exponentially rising together with the number of constituent transistors [3] which, according to the empirical Moore's law, doubles every 18 months [4]. The frequency of the internal clock followed a similar trend until the mid-2000s when it hit a limit known as the "power wall". The problem is that the power consumed by the processor scales with the frequency cubed [5]. At some point, the power requirement for running the chip and cooling it to keep it within operational limits becomes utterly impractical. The processors cannot get faster any more, and in order to get more work done in a given time, they have to do it in parallel. Thus, all contemporary high-performance systems deal with many concurrent calculations.

With respect to the number of instructions performed in parallel and the amount of data manipulated at the same time, computer architectures can be classified by Flynn's taxonomy [6]:

- SISD – Single Instruction Single Data, sequential processing, no parallelism
- SIMD – Single Instruction Multiple Data, vector instructions, GPUs
- MISD – Multiple Instruction Single Data, redundant systems
- MIMD – Multiple Instruction Multiple Data, distributed systems

For large-scale numerical simulations, the two relevant are SIMD, where a data stream is manipulated by the same instruction, and MIMD where different processing units concurrently manipulate different data streams.

While the performance of an algorithm depends largely on higher-level considerations of its computational complexity and memory consumption, its design must be informed by specifics of the hardware architecture in order for it to run efficiently. On modern machines, efficiency is influenced by the 7 dimensions of performance[1]:

- Computing nodes – interconnect speed, communication by message passing
- CPU sockets – memory layout, channels, memory pinning
- Cores – number of cores, package topology
- Hardware Threads – topology, OS numbering, shared resources
- Ports – superscalar parallelism, in-order/out-of-order, done in HW
- Pipelining – instruction-level parallelism, done in HW
- Vectors – vector instructions usage and efficiency

Efficient parallel programs need to tackle into there resources in such a way that the possible speed improvements promised by the hardware manufacturers would not go to waste. Apart from choosing the correct algorithm, we can gain performance by taking the data flow into account, i.e. optimizing the way data travels between individual processing units, and between the processing unit and the memory. In the first case, the slowest path is the one between individual computing nodes which are connected via a network (Ethernet, Infiniband, etc.). Within one node, there can be multiple CPUs, each in its own socket. The communication between the sockets is realized through a shared memory accessed via a bus. Each CPU might include multiple cores, which can access shared "caches" – areas of fast memory embedded in the chip. In the second case, the time it takes the processing unit to access the memory, called latency, depends on the type of memory unit about to be accessed. There are several levels of storage, apart from the familiar RAM and hard drives. In general, faster memory units have a higher cost per byte. Thus the hardware manufacturer has to balance the cost (and speed) and the available storage size. Since the bus access to the RAM can take several hundreds of cycles, data which is accessed repeatedly would benefit from being stored in a cache right on the processor die. These caches are very fast, but also quite expensive. Therefore, they are much smaller than RAM. A modern super-computer might consist of following types of memory units (the sizes and latencies are approximate and depend on the particular hardware)

memory type	size	latency [cycles]
Registers in the CPU core	16 x 64 B	1
L1 cache per core	64 kB	4
L2 cache per core	256 kB	12
L3 cache per CPU	8 MB	40
RAM accessed via bus	GBs – TBs	200 – 500
Persistent memory (HDD, SSD)	TBs	$10^4 – 10^6$
GPU memory (offloading via PCIe bus)	GBs	10^5
Other nodes (via network)	TBs	$10^4 – 10^7$

Since the latencies vary within several orders of magnitude, it is crucial to try to minimize the costlier communication paths by keeping the data as close to the processor as possible. The more the same data is reused, the more it will benefit from staying in the faster memory units. The algorithm should therefore take into account the memory access patterns since choosing a wrong one could outweigh the benefits of a fast algorithm by adding too much communication overhead.

2 Parallel programming

2.1 Scaling

When converting a serial algorithm into a parallel one, we are mostly interested in how much speed gain are we about to get from such conversion [6]. Formally, we want to know the "parallel speed-up" *S* of solving a problem of size *n* on *p* processors

$$S(n,\ p) = \frac{SU(n)}{T(n,\ p)}$$

where *SU* is the upper bound on time complexity, i.e. the worst-case time complexity of the fastest know sequential algorithm, and *T* is the parallel time, i.e. the time elapsed from the beginning of a p-processor parallel algorithm until the last processor finishes execution. The parallel time comprises of computational steps and communication steps (overhead). The best speed-up we can hope for is *p*. In practice, linear speed-up *kp* with $0 < k < 1$ is quite satisfactory.

For any non-trivial algorithm, there will be some parts which cannot be parallelized and have to remain sequential; thus, the actual speed-up will vary with the number of processors. The measure of how well the algorithm performs when adding more processors is called scaling. There are two basic measures of scaling, called "strong" and "weak".

Strong scaling, also called Amdahl's law assumes a problem of fixed size *n* being solved by an algorithm which has a sequential part of computation *f*, and parallel part of computation 1-*f*. Then the maximum speed-up on *p* processors is

$$S(n,\ p) = \frac{1}{f - \frac{1-f}{p}}$$

This can be intuitively understood from the fact that the sequential part always takes the same time to execute no matter the number of processors, therefore the algorithm can never run faster than the whole sequential part. It is a considerably strict limit – suppose your algorithm is 95% parallel and 5% sequential. Then the maximum theoretical speed-up on an infinite number of processors is only 20.

Often, problem size can increase while sequential parts remain constant. In this case, we talk about weak scaling, also called Gustafson's law. It assumes a fixed number of processors *p*, and increasing the problem size *n*. The sequential part *f* has constant complexity, and the parallel part can scale linearly with *p*. Then the maximum speed-up is

$$S(n,\ p) = f + p(1-f)$$

In reality, at large scale, the sequential part might would not have a strictly constant complexity, there would be some additional communication overhead coming from using a lot of processors, but well-designed algorithms which closely fit the stated assumptions often come close to this theoretical ideal.

2.2 Processing

Concerning the flow of instructions, the algorithms can be roughly distinguished by the concepts of data or task parallelism. In data-parallel algorithms, the same task is performed simultaneously by many processing units each operating on a subset of the dataset. There is a single control flow, and the concept fits the SIMD model. In task-parallel algorithms, different tasks are performed simultaneously by independent processing units on the same or different datasets. There are multiple control flows, and the concept fits the MIMD model.

The flow of data is mostly realized by message passing, threading, or vector processing. In message passing, every parallel process is working in its own memory space in isolation from the others. There is an explicit point-to-point communication, where outgoing messages are stored on the sender's queue until their recipient retrieves them. It is by far the most common model of inter-node communication. Threading, on the other hand, relies on the presence of shared memory through which every parallel thread has access to all of the data. It is useful in intra-node (CPU-CPU, core-core) communication, and is generally faster than message passing on the same machine. Though it is arguably easier to implement, it may create problems such as synchronization and memory protection. Computationally, threading is equivalent to message passing – emulating message passing by threading can be achieved by a designated memory area that serves as message storage, and emulating threading by message passing could be achieved by sending messages that ask for specific areas of the memory in the other node, though such emulations are rarely useful. Finally, data flow in strictly data-parallel sub-problems can be achieved via vector processing. Here, vector is an instruction operand containing a set of data elements packed into a one-dimensional array. The operand is then passed to SIMD instructions in the CPU (on x86 architecture: SSE, AVX) which performs a single operation on the whole vector at once, e.g. in case two vectors containing 4 floating point numbers each can be added in a single instruction pass instead of calling 4 addition instructions. In GPUs, vectorization is the main mode of operation with "stream processors" (CUDA cores) which allow writing whole pipelines of operations.

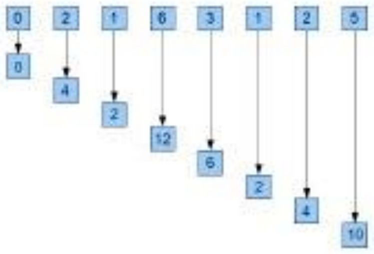
2.3 Fundamental parallel algorithms

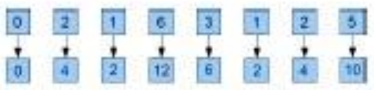
Numerical algorithms can be though to be composed of a control flow, i.e. passing through the numbers to be computed or selecting whether what to do based on some condition, and computation, i.e. performing the algebraic operations. The algebra itself works the same way in both sequential and parallel cases. We are thus concerned with the parallelization of the control flow. First of the fundamental sequential control patterns is the "sequence", an ordered list of tasks/ commands to be carried out in that order. Since the order is specified, necessarily only one task is executed at a time (think procedural execution in code), thus this pattern cannot be parallelized. The second is "selection" between two commands to be possibly executed, depending on a condition (think "if statement"). In some cases, selection can be replaced by parallel masking or task dispatching. The last one is "iteration", where a certain function is repeatedly executed as long as a condition is true (think "while" or "for" statements). This is a prime candidate for parallelization.

Out of the many existing parallel patterns, we will focus on two types of the most fundamental ones. The iteration patterns which are the analogues to serial iteration: Partition, Map, Reduce, and Scan. And the decomposition patterns which are not applicable for strictly serial code, and include: Pipeline, Superscalar Sequences, and Domain Decomposition. [2]

Sometimes, it is useful to treat multiple items together, e.g. split a large array into vectors that fit in the SIMD registers. *Partition* allows for a custom split of the collection into non-overlapping sub-collections (in case of overlapping sub-collections, see domain decomposition later)

Map applies the same function on multiple elements, e.g. to multiply an array of values by a constant. It is a direct parallel analogy to a for cycle. Speed-up scales linearly with the number of elements which can be processed in parallel.





Care must be taken not to introduce "loop dependencies". Data dependencies between reads and writes might prevent direct parallelization of a cycle to a map. In such case, a parallel unit operating on some position would read/write a value which would then get overwritten by the second one which had no idea that the value has changed since the time it fetched it. They come in three types:

read-after-write

```
for(i=1; i<N; ++i) {
    a[i] = a[i-1] + b[i];
}
```

write-after-write

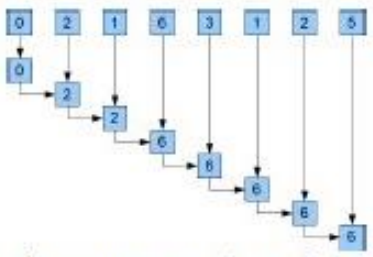
```
for(i=0; i<N-2; ++i) {
    a[i]     = b[i] + c[i];
    ...
    a[i+2] = 2 * i;
}
```

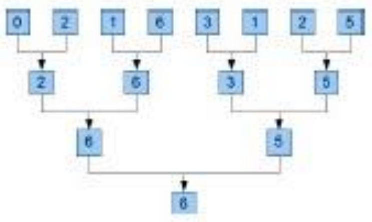
write-after-read

```
for(i=1; i<N; ++i) {
    a[i-1] = a[i] + c[i];
}
```

writing to a variable, then reading its value	writing to the same variable in more than one iteration	possible data race if i-th iteration is performed before (i-1)-th
RAW	**WAW**	**WAR**

Reduce combines the elements into a single result using a combining function, e.g. summing all elements, or finding the maximum element in a collection. It can run in *log n* steps on *n* processors.





Map-Reduce is a common pattern where a function is applied to every single element (Map), and the result is passed to a combiner function (Reduce).

Example: How many times does a word appear in a text?

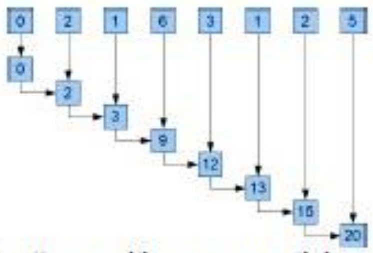
- Partition: Split the text into equal chunks and distribute them among PUs
- Map: Count the word appearance in each chunk
- Reduce: Add the counts

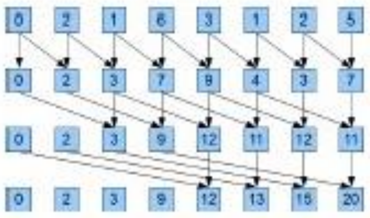
Another Example: Find the norm on an N-dimensional vector

- Partition: Distribute the numbers to P workers, each gets N/P numbers
- Map: Each worker squares and adds assigned numbers
- Reduce: Add the sub-results
- Serial operation: Parent worker finds the square root

Finally, *scan* solves the problem of parallelizing a loop in which the *i*-th result depends on the (*i*-1)-th. It can run in *log n* steps on *n* processors. A canonical example is the "prefix sum", with the input sequence **b**, and the output sequence **a**, $a_i = a_{i-1} + b_i$. In sequential code:

```
a[0] = b[0];
for (int i=1; i<N; ++i) {
    a[i] = a[i-1] + b[i];
}
```





Out of parallel patterns with no sequential counterpart, *Superscalar Sequences* refer to the concept of splitting work into many tasks, and defining data dependencies between the tasks. A task scheduler then simplifies the resulting data dependency graph and decides which of the tasks to run in parallel

The *Domain Decomposition* pattern divides the dataset into subsets which are computed independently. If dependencies exist, think of the algorithm as several independent steps separated by communication steps which resolve the dependencies. It can be performed at node-level with communication by message-passing, or thread-level with communication by reading and writing to a shared memory. In the latter case, the decomposition can be static – each thread is pre-assigned its data subset, or dynamic – subsets boundaries might change between iterations or even during one iteration (often difficult or practically impossible at node-level).

In laser-plasma physics simulations, we often encounter large multi-dimensional arrays of data. These are the number one candidates for domain decomposition, which maps naturally to the problem. In some cases, the mapping leads to "embarrassingly parallel" algorithms, i.e. those with little (or no) need for communication. Examples include normalizing an array of vectors – no communication needed, or finite-difference integration using a stencil (see later) – communication on border cells only.

Domain decomposition and the Map-Reduce patterns are examples of the more general "divide and conquer" concept which comprises of three steps

- Divide: divide the given problem into smaller independent sub-problems of the same type as the given problem and solve them concurrently.
- Conquer: If the problem is small enough, then solve it.
- Merge: Combine the solutions to the sub-problems to get the solution to the given problem.

A common task in laser-plasma physics simulations is to solve differential equations over a large possibly multi-dimensional array. There is a vast class of algorithms implementing the " finite-difference integration" method, where the integrated value in a specific cell of one variable depends on the values of neighbouring cells of some second variable. This data dependence can be described in the form of a stencil, where memory addresses for inputs are expressed as offsets relative to the location of outputs. The stencil then accesses a regular neighbourhood of the input for each output. Since it is possible to read from outside of valid index range, you need to treat boundary conditions.

Some finite-difference (and other) methods can achieve faster convergence by using multiple grids with different resolutions. In a general "multigrid methods" [9], first, a coarse grid of points used. With these points, the iteration process will start to converge quickly. At some stage, the number of points is increased to include points of the coarse grid and extra points between the points of the coarse grid, where the initial values of the extra points are found by interpolation. Computation continues with this finer grid. The grid can be made finer and finer as the computation proceeds, or computation can alternate between fine and coarse grids. Coarser grids take into account distant effects more quickly and provide a good starting point for the next finer grid.

Last of the commonly used parallel patterns is *load balancing* [10], a set of methods used to distribute the workload among the individual processing units in order to optimize resource use by avoiding overloaded and idling workers. Two simplest examples would be either to partition the problem into many more parts than the number of available threads, and let the thread scheduler do the rest, or to send parts of a decomposed domain from busy nodes to the idling ones.

3 Implementation of parallel algorithms

As we have already seen, the parallelization touches many levels. At the high level of parallel algorithm implementation stands the realization of parallel patterns. This is done either by dispatching threads (e.g. OpenMP), or by sending messages (e.g. MPI). At the low level, we have vectorization and memory access patterns with concerns of data types and cache utilization.

3.1 Vectorization

Vectorization is performing one operation on multiple operands simultaneously with SIMD instructions – CPU instructions which perform the operation, and SIMD registers – stores in the CPU which hold the operands. The most prevalent Intel x86 architecture has first introduced SIMD instruction in 1997 with the "MMX" instruction set with 80 bit integer-only registers. SSE (1999) introduced 128 bit floating point registers, which can hold 4 single precision floats. SSE2 (2001) added support for double precision FP, while still being 128 bit (4 single precision float, or 2 double precision). Up to SSE4.x (2006) new instructions were added working with 128 bit registers. Then AVX/AVX2 (2008) increased the length to 256 bit, which can hold 4 double precision FP numbers, and the state-of-the-art AVX-512 (2016) upcoming processors, introduced 512 bit registers – that's 8 double precision, or 16 single precision FP numbers being manipulated by a single instruction, thus offering up to 8x or 16x algorithm speed-up, if implemented properly.

In the C-language family, SIMD vectors are represented using "packed types" , such as *__m128*, a 128-bit pack which holds 4 single-precision floats. SIMD instructions are represented using "intrinsic functions" (also called "intrinsics") which are translated directly to vector instructions by the compiler:

__m128 _mm_add_ps (__m128 a, __m128 b)	SSE2 – adds 4 floats
__m256 _mm256_add_pd (__m256 a, __m256 b)	AVX – adds 4 doubles

Modern compilers understand constructs like *a+b* on vector types and will emit the correct vector instruction. They will also auto-vectorize inner loops if not prevented by loop dependencies, or by an unsuitable collection data type.

When mapping multiple "mathematical vectors" to the "hardware vectors", one might be tempted to go by the road of "vertical vectorization", as in the following example performing a vector addition:

```
__m128 a, b, c;
...
c = _mm_add_ps ( a, b );
```

While this approach is simple to implement, and would certainly work adding the whole vectors in just one run, it is not scalable. Suppose we are working with 3D vectors; *__mm128* can hold 4 floats, but only 3 are utilized if mapped to a 3D vector, and even more wasting on 'longer' architectures.

The solution to this problem is called "horizontal vectorization", and it works by operating on a single component of multiple vectors at once. In the 3D case:

instead of

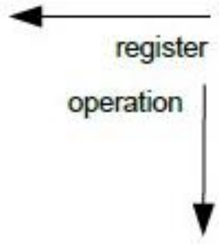
$$\begin{pmatrix} c_x^i & c_y^i & c_z^i \end{pmatrix} = \begin{pmatrix} a_x^i & a_y^i & a_z^i \end{pmatrix} + \begin{pmatrix} b_x^i & b_y^i & b_z^i \end{pmatrix}$$

for each vector

we do blocks of

$$\begin{pmatrix} c_x^i & c_x^{i+1} & c_x^{i+2} & c_x^{i+3} \end{pmatrix} = \begin{pmatrix} a_x^i & a_x^{i+1} & a_x^{i+2} & a_x^{i+3} \end{pmatrix} + \begin{pmatrix} b_x^i & b_x^{i+1} & b_x^{i+2} & b_x^{i+3} \end{pmatrix}$$

for each component
(and all vectors jumping by i += register size)



This way, we avoid wasting of registers, and the code will better scale to "longer" architectures, but we might have to reorder our loops significantly.

3.2 Cache optimization

We have already discussed the hierarchy of memory. From fastest to slowest, it goes as [11]:

register -> L1 cache -> L2/L3 cache -> main memory.

Data from main memory is always loaded to a cache in a "line", where one "cache line" is 64 bytes on all x86 architectures. Subsequent access to the same cache line is fast, but if the requested data is not found in the cache, the whole line gets evicted, and a whole new line has to be loaded. Thus, we want to "stream" the data to the processor as fast as possible – ideally without "cache misses". In order to achieve this, we have to think about how is our data laid out in memory.

Considering the example of many 3D vectors, two most common data layouts are the "Array of Structures" (AoS) data model, where all members of one vector are next to each other, and the "Structure of Arrays" (SoA) model, where same members of different vectors are next to each other. There are algorithms which work better in an AoS layout, e.g. those which perform a lot of sorting operations on the whole vectors, but the SoA model is better suited for horizontal vectorization because it can continuously stream needed data. The choice of the data layout is at the base of the architecture of your program, and changing to a different one can be a daunting task in large codebases. The appropriate course of action should always be based on rigorous performance measurement and profiling.

If you already have existing code that would be too difficult to change, or you have seen that your algorithm performs faster in AoS, you can achieve horizontal vectorization by a runtime Aos-to-SoA transformation by shuffling the data from several vectors into the appropriate registers, preforming the calculation with a full register width, and shuffling the data back. Apart from the need for the two shuffling operations, this approach also suffers from bad cache utilization. Yet, it might be better than wasting registers on "long" architectures. Unless necessary, it is best to be avoided. Midway between AoS and SoA lies a seldom used AoSoA layout – an array containing structures containing smaller arrays where each component is the same length as the register. It is well suited for vectorization, especially if you often access multiple members of the same vector, but arguably most difficult to implement.

Another consideration is that of alignment, i.e. the position of the beginning of the data structure. The main memory is addressed in blocks of the size of a cache line, and the reading always starts at the beginning of a cache line boundary. There are different instructions for loading aligned vs. unaligned data into the registers, with the unaligned version being slower, and unaligned access fraught with the possibility of crossing a cache line boundary, in which case the processor has to do two reads to fill the register. In auto-vectorized loops over compact data structures, modern compilers will emit code which does aligned access if it's possible (e.g. iterating over an array with step = 1). In an AoS setting, alignment might also require padding, e.g. a struct of three 32 bit floats with 128 bit alignment needs to be padded by additional 32 "useless" bits. This will increase memory consumption, and for long arrays also traversal time.

Vectorization is impossible for read-after-write and write-after-write loop dependencies. The write-after-read dependency is vectorizable. Since the vectorization traverses the loop by blocks in strictly increasing order, we know that the i-th element will not be written to before being read. [12]

3.3 Threading

Parallelization by dispatching threads relies on the presence of a shared memory. It is used in multi-core CPUs and multi-socket nodes. There are many libraries which implement threading, e.g. the Intel Thread Building Blocks, Posix threads, and language-specific libraries. An ubiquitous cross-platform threading approach is the OpenMP standard included in most modern compilers. OpenMP [13] is an API for multi-threaded programming. It uses a declarative approach to threading – you specify regions to be parallelized by using #pragma directives. It is well suited for use in data-parallel scenarios. The number of spawned threads can be controlled either at runtime, or by the environment variable OMP_NUM_THREADS. It usually defaults to the number of cores in your machine, so most of the time you don't have to set anything.

An example structure of a basic OpenMP-enabled program might look like:

```
// declaring a parallel section of code
#pragma omp parallel
{
  // this code will be executed by all threads
  #pragma omp for
  for(int i=0; i<N; ++i) {
     // work will be divided between the threads
  }
}
// a shorthand for loop parallelization
#pragma omp parallel for
for(int i=0; i<N; ++i){ ... }
// has syntax to perform reductions
#pragma omp parallel for private(val) reduction(+:sum)
for(int i=0; i<N; ++i) {
    val = calculate_local_sub_result();
    sum += val;
}
```

This is already enough to parallelize simple loops without data dependencies. A basic rule of thumb to easily implement threading is to write vectorization-friendly code. A code which is well vectorized is usually easily parallelized. Cache optimization matters in a similar way as in vectorization. Unlike auto-vectorization though, where a potentially problematic loop wouldn't be vectorized, when writing thread-parallel programs you have to watch for these problems yourself. Loop dependencies between reads and writes might lead to data races and silent failures (wrong results), making threading more difficult than with vectorization, because we don't know which thread will execute first. RAR cannot be done in thread-parallel scenarios, while the WAW and WAR possibly could, but not automatically – care must be taken to avoid possible data races.

3.4 Message passing

From the architectural point of view, message passing is the most difficult of the three parallelization techniques. It is also the only general method of communication between different nodes. Instead of a declarative approach, the programmer has to decide what will be the contents of the messages, explicitly set the point of sending and receiving, and tak care of proper synchronization. By far the most prevalent implementation of message passing in the context of numerical simulations are the MPI libraries [14]. The canonical "Ping pong" example from the MPI tutorial website goes as:

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
  const int PING_PONG_LIMIT = 10;

  // Initialize the MPI environment
  MPI_Init(NULL, NULL);
  // Find out rank, size
  int world_rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
  int world_size;
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);
...
  int ping_pong_count = 0;
  int partner_rank = (world_rank + 1) % 2;
  while (ping_pong_count < PING_PONG_LIMIT) {
    if (world_rank == ping_pong_count % 2) {
      // Increment the ping pong count before you send it
      ping_pong_count++;
      MPI_Send(&ping_pong_count, 1, MPI_INT, partner_rank, 0, MPI_COMM_WORLD);
      printf("%d sent and incremented ping_pong_count %d to %d\n",
             world_rank, ping_pong_count, partner_rank);
    } else {
      MPI_Recv(&ping_pong_count, 1, MPI_INT, partner_rank, 0, MPI_COMM_WORLD,
               MPI_STATUS_IGNORE);
      printf("%d received ping_pong_count %d from %d\n",
             world_rank, ping_pong_count, partner_rank);
    }
  }
  MPI_Finalize();
}
```

The same program is run on two computers. After initializing the MPI routines, the processors get their "rank" - an identifier unique among the running processes. Inside a *while* loop, the the processes alternate between sending and receiving. First, process 0 sends the message, and process 1 waits to receive it. After this interaction, the roles are reversed, and the exchange goes on until the number of exchanges reaches *PING_PONG_LIMIT*.

There are more advanced modes of communication with specialized MPI routines for broadcasts, reductions, scans, and many others, as well as various options of setting virtual topologies suited for the particular problem at hand.

4 Conclusions

Modern computers achieve high performance by running many tasks in parallel. This is likely to be even more true for many years to come. Serial algorithms are not going to run much faster than now; therefore they must be parallelized by means of replacing the traditional serial patterns with their parallel counterparts and implementing new patterns seen only in parallel computing environments while keeping an eye on the scaling properties of the parallelized versions. At the same time, the low-level architecture of even the biggest super-computers still carries the legacy of the early pioneers. Therefore, care must be taken to implement correct communication and memory access patterns in order to utilize the maximum possible performance of the theoretical promise of given hardware.

The main implementation tools are vectorization, multi-threading (e.g. OpenMP), and message-passing (e.g. MPI). The hardware, standards and tools are continually evolving; thus the best source of up-to-date information is the Internet. Wikipedia articles on the topics discussed here are usually of high quality and can be used as a good starting point.

Many of the tools have integrated features which will help you to realize the parallel patterns which represent your particular algorithm. Nevertheless, in order to use these tools efficiently, it always helps to understand the underlying ideas and the problems they were proposed to solve.

Last but not least, whenever you are going to optimize an algorithm, reason on the basis of performance testing, measurement, and profiling. Otherwise, it is too easy to fall into the trap of endless micro-optimizations in those parts of the code where partial performance gains make up a minuscule portion of the real performance bottlenecks.

Most of all, I wish you a pleasant journey on the exciting path to the future of computing.

5 References

[1] G. O'Regan, *Introduction to the History of Computing: A Computing History Primer*. Springer, 2016.
[2] "TOP500 Supercomputer Sites." [Online]. https://www.top500.org/
[3] K. Rupp, "42 Years of Microprocessor Trend Data | Karl Rupp." .[Online] https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/
[4] "Moore's Law." [Online] http://www.mooreslaw.org/
[5] "Why has CPU frequency ceased to grow?," [Online] https://software.intel.com/en-us/blogs/2014/02/19/why-has-cpu-frequency-ceased-to-grow
[6] B. Schmidt, J. Gonzalez-Dominguez, C. Hundt, and M. Schlarb, *Parallel Programming: Concepts and Practice*. Morgan Kaufmann, 2017.
[7] S. G. Akl, *The Design and Analysis of Parallel Algorithms*, First Printing edition. Englewood Cliffs, N.J: Prentice Hall, 1989.
[8] "Dr. Dobb's | Good stuff for serious developers" [Online] https://www.drdobbs.com/.
[9] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*, 2nd ed. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2000.
[10] M. Tamura, "Load Balancing 101: Nuts and Bolts," [Online] https://f5.com/resources/white-papers/load-balancing-101-nuts-and-bolts-31392
[11] U. Drepper, *What Every Programmer Should Know About Memory*. 2007.
[12] "Using Automatic Vectorization," [Online] https://software.intel.com/en-us/cpp-compiler-developer-guide-and-reference-using-automatic-vectorization. [Accessed: 13-Sep-2019]
[13] "OpenMP tutorial" [Online] https://computing.llnl.gov/tutorials/openMP/
[14] "MPI Tutorial." [Online] https://mpitutorial.com/

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Chapter 10: Astrophysics with high-power lasers

A. Ciardi

10.1 From laboratory plasmas to astrophysical plasmas

10.1.1 Macrophysics-Microphysics

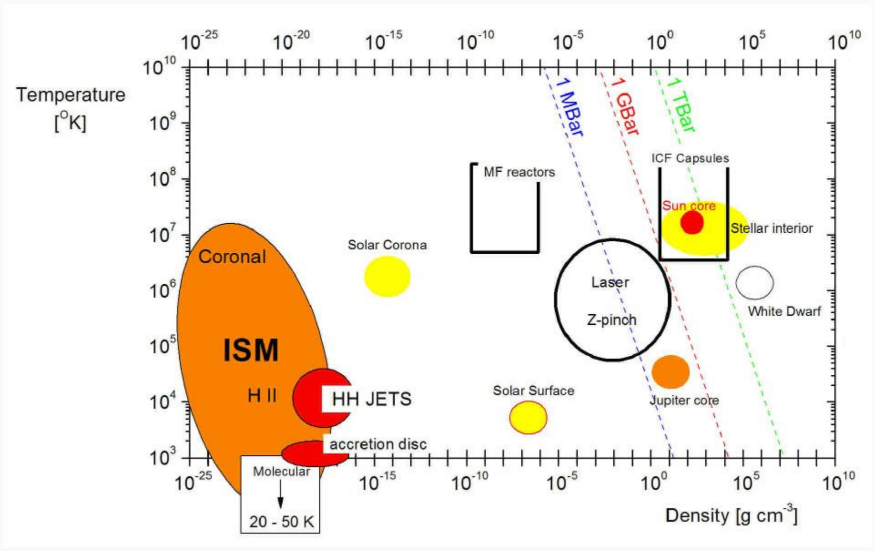
Broadly speaking high-energy density laboratory astrophysics studies can be divided into those addressing the

- microphysics: equation of state, opacities, . . . Where plasma conditions in the laboratory are the same as those found in space
- macrophysics: shocks, jets, particle acceleration, magnetic reconnection, . . . Where the plasma conditions in the laboratory are a "scaled" version of those found in space.

10.1.2 High-energy density laboratory astrophysics

Typical plasmas on z-pinch and laser facilities have pressures of ~Mbar, corresponding to energy densities ~10^{12} erg cm^{-3}, at a fraction of solid density. An overview of the plasma conditions attainable on experimental installations, together with some of those found in space are shown in the figure. To obtain such conditions, z-pinch facilities rely on stored electrical energy (hundreds of kJ) to deliver large currents (~ of a few Mega Amperes) over a short time (~ 100-1000 ns) to a "load" usually consisting of a gas or thin metallic wires and produce typical volumes of plasma ~1 cm^{-3}. Laser facilities instead rely on focusing onto a solid or gaseous target a high-power laser beam, or beams. These concentrate several kJ of energy, over timescales ~pico- to nano-second, into plasma volumes of ~1 mm^{-3}. The Laser Megajoule (LMJ) facility in France and the National Ignition Facility (NIF) in the USA, can produce fusion plasmas under conditions similar to stellar interiors. Producing plasma conditions similar to those found in space is not always necessary for the laboratory study of astrophysical phenomena. For the present discussion it is more interesting to look at the dynamical conditions that can be obtained in the laboratory. For example, the typical energies available on z-pinches and lasers, when partially converted into kinetic energy, can generate hypersonic (Mach numbers $M > 5$), radiatively cooled flows with characteristic velocities of the order of 100 - 1000 km s^{-1}. These can include dynamically important magnetic fields, ~several 10^6 Gauss, and a large range of plasma-β (the ratio of thermal to magnetic pressure), $1 >> \beta >> 1$. Then, our inability to obtain the adequate plasma conditions may be overcome by producing scaled "conditions" of the phenomena of interest.

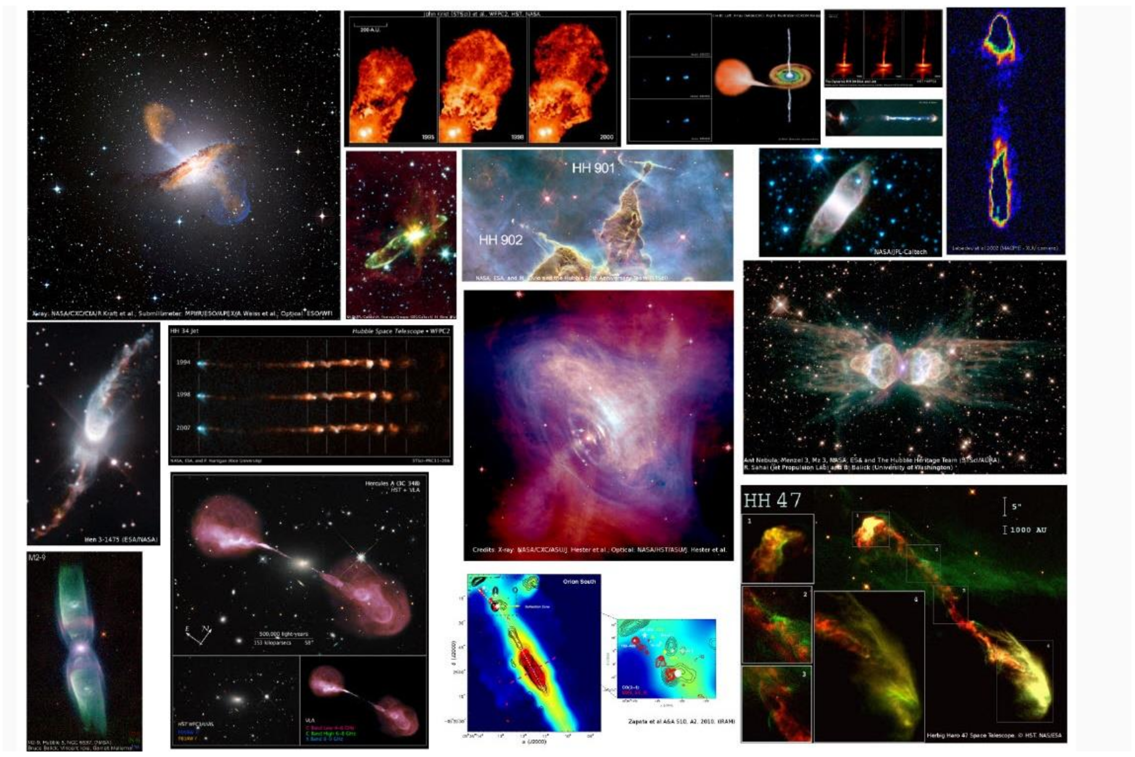
Working definition. Energy density $\gtrsim 10^{12}$ erg cm^{-3}; pressure $p \gtrsim 1$ Mbar



Ciardi, Lecture Notes in Physics, 2010

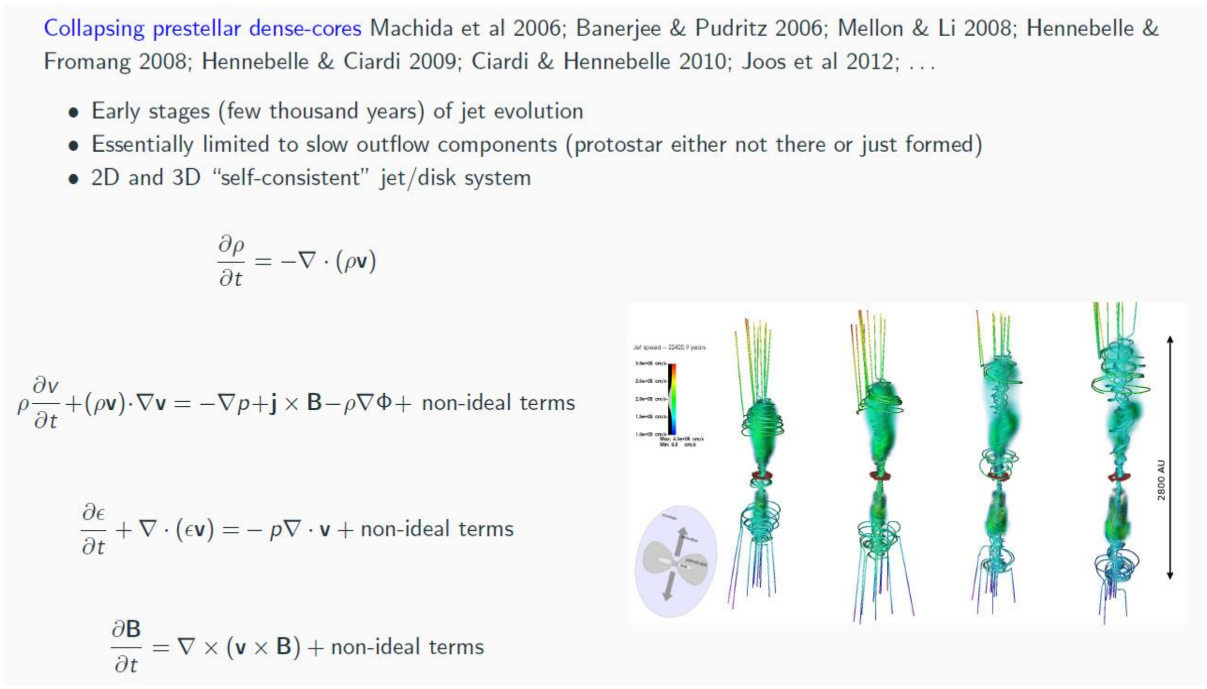
10.2 Mass ejection in young stars

Jets are ubiquitous in space. In general, jets are thought to be powered by the combination of rotation and magnetic fields, which extract the rotational energy from an accreting system and create magnetic stresses which accelerate and collimate the flow. Depending on the details of the models, the winding of an initially poloidal magnetic field results in a flow pattern dominated by a toroidal field. A similar situation is also attained when the foot-points of a field line, connecting the disc to a central compact object or connecting different parts of a disc, rotate with different angular velocities.



10.2.1 Basics of jet numerical modelling

Collapsing prestellar dense-cores
3D modelling is limited to the early stages (tens of thousand of years) of jet evolution
Essentially limited to the slow outflow components (protostar either not there or just formed)
The jet and disk are treated “self-consistent”
Disk included (and star)
Start with an initial star-disk/ambient structure and large-scale poloidal field
Essentially limited to 2D and relatively short time-scales
Jets can have a feedback on the disk and star
Disk (or Poynting flux injection) as a boundary condition
The magnetic field distribution, rotation and mass injection at the base of the jet are imposed as boundary conditions or initial conditions.
There is no jet/wind feedback on the disk
Simulation in 2D and 3D over long time and spatial scales are possible
Gravity is often neglected



Collapsing prestellar dense-cores Machida et al 2006; Banerjee & Pudritz 2006; Mellon & Li 2008; Hennebelle & Fromang 2008; Hennebelle & Ciardi 2009; Ciardi & Hennebelle 2010; Joos et al 2012; . . .

- Early stages (few thousand years) of jet evolution
- Essentially limited to slow outflow components (protostar either not there or just formed)
- 2D and 3D “self-consistent” jet/disk system

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{v})$$

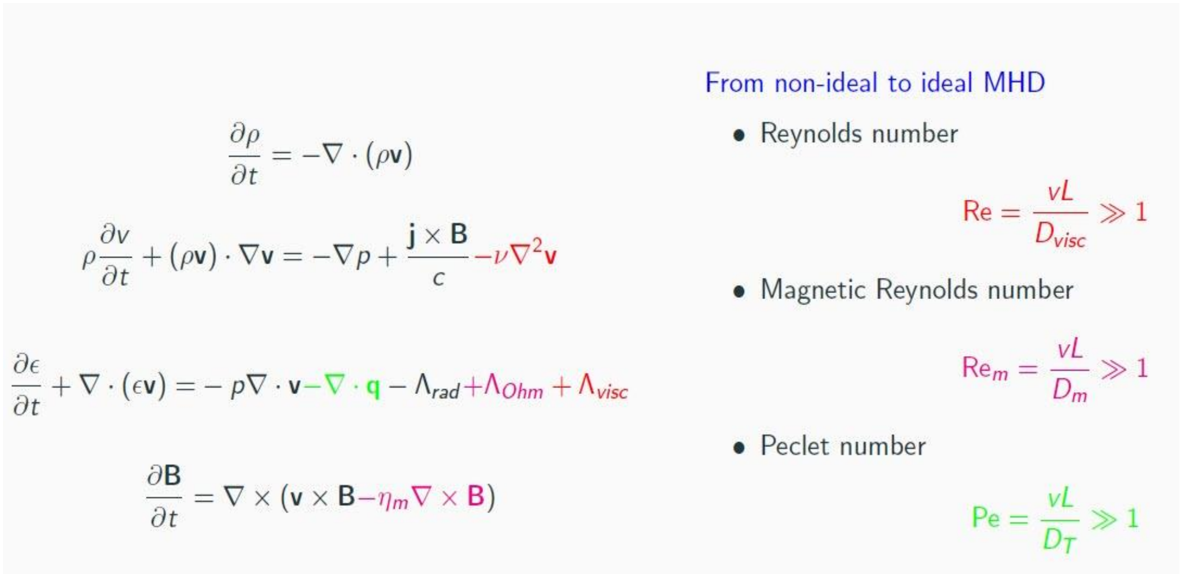
$$\rho \frac{\partial \mathbf{v}}{\partial t} + (\rho \mathbf{v}) \cdot \nabla \mathbf{v} = -\nabla p + \mathbf{j} \times \mathbf{B} - \rho \nabla \Phi + \text{ non-ideal terms}$$

$$\frac{\partial \epsilon}{\partial t} + \nabla \cdot (\epsilon \mathbf{v}) = -\, p \nabla \cdot \mathbf{v} + \text{non-ideal terms}$$

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B}) + \text{non-ideal terms}$$

10.2.2 Modelling jets as ideal-magnetofluids

Scaling the laboratory to astrophysical dynamics relies on MHD being applicable. That requires the advection of momentum, magnetic field and thermal energy to dominate over their diffusion. This can be quantified by three dimensionless number.



$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{v})$$

$$\rho \frac{\partial v}{\partial t} + (\rho \mathbf{v}) \cdot \nabla \mathbf{v} = -\nabla p + \frac{\mathbf{j} \times \mathbf{B}}{c} - \nu \nabla^2 \mathbf{v}$$

$$\frac{\partial \epsilon}{\partial t} + \nabla \cdot (\epsilon \mathbf{v}) = -p \nabla \cdot \mathbf{v} - \nabla \cdot \mathbf{q} - \Lambda_{rad} + \Lambda_{Ohm} + \Lambda_{visc}$$

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B} - \eta_m \nabla \times \mathbf{B})$$

From non-ideal to ideal MHD

- Reynolds number

$$\mathrm{Re} = \frac{vL}{D_{visc}} \gg 1$$

- Magnetic Reynolds number

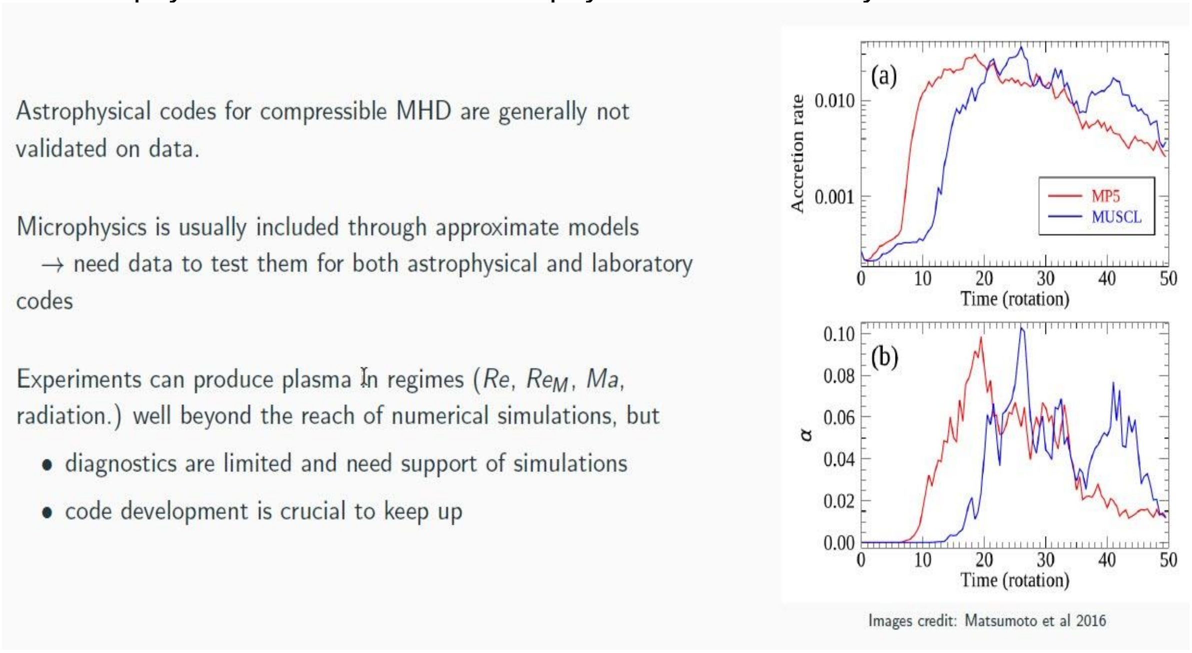
$$\mathrm{Re}_m = \frac{vL}{D_m} \gg 1$$

- Peclet number

$$\mathrm{Pe} = \frac{vL}{D_T} \gg 1$$

10.2.3 Compressible, radiative magneto-hydrodynamics: Why is it important?

It is only recently that compressible, radiative MHD flows can be produced in the laboratory, and in general astrophysics codes are not validate on real data. For example even different numerical schemes, within the same code, can give largely different answers. Laboratory data can help better understand and constraint the numerical models. Other examples of the importance of laboratory data is the need to test microphysics models for both astrophysical and laboratory codes.

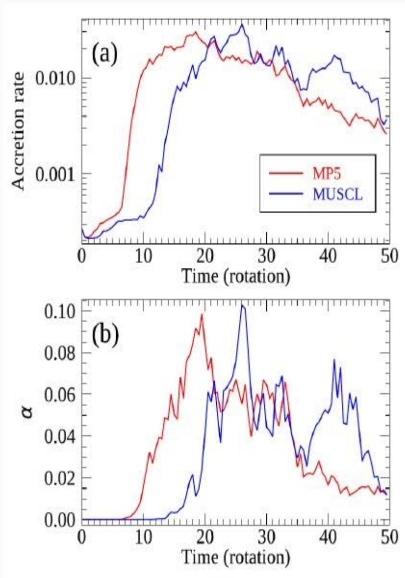


Astrophysical codes for compressible MHD are generally not validated on data.

Microphysics is usually included through approximate models
→ need data to test them for both astrophysical and laboratory codes

Experiments can produce plasma in regimes (Re, Re_M, Ma, radiation.) well beyond the reach of numerical simulations, but

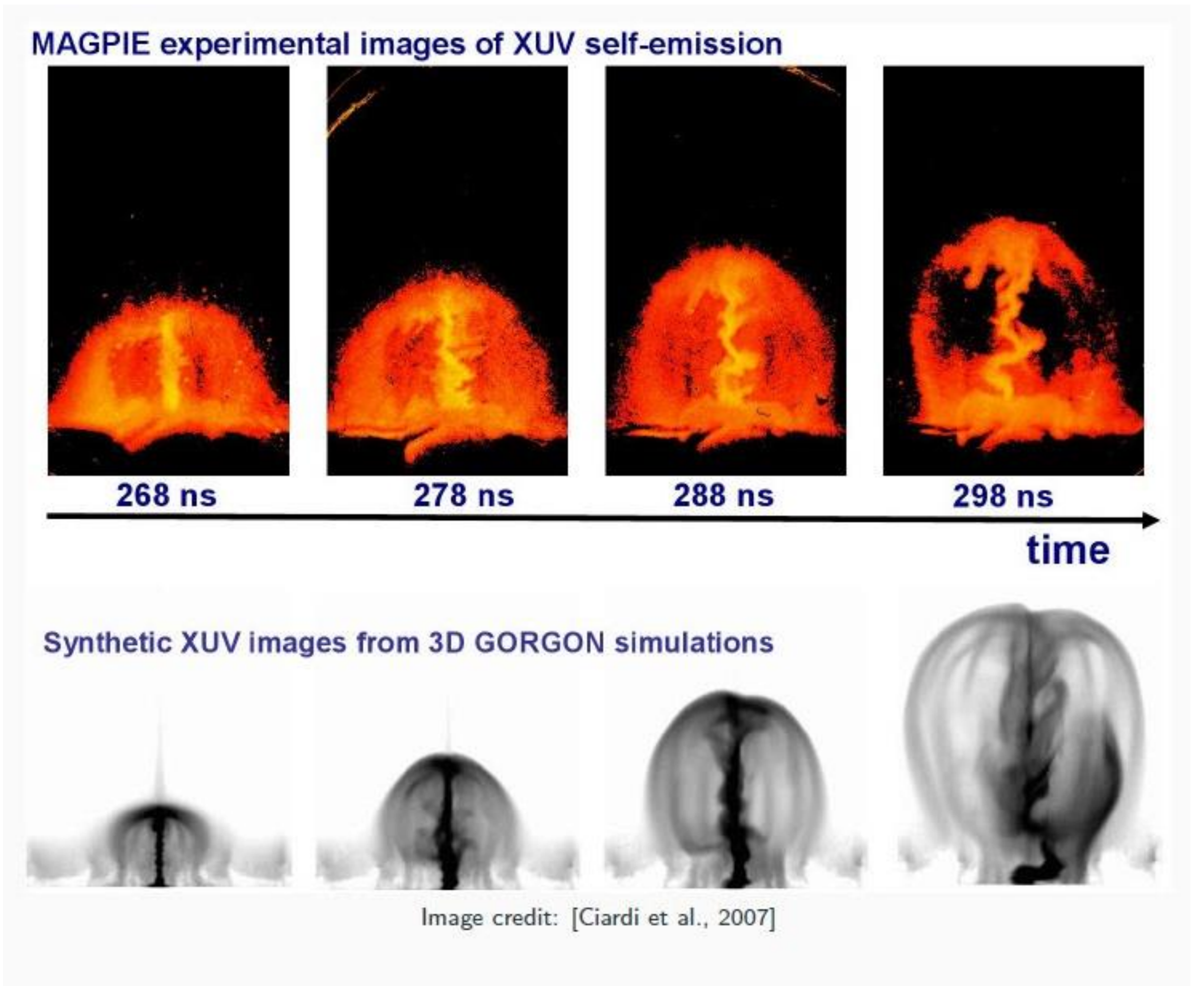
- diagnostics are limited and need support of simulations
- code development is crucial to keep up



Images credit: Matsumoto et al 2016

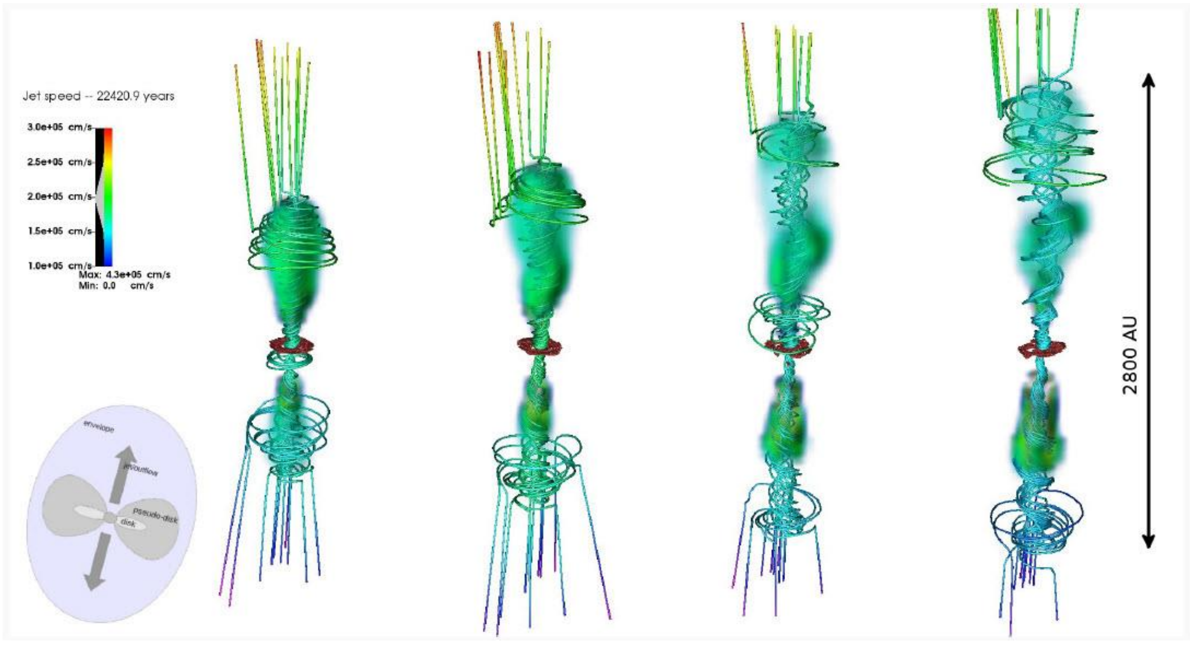
10.2.4 Experiments and 3D simulations show kink-unstable jets

The basic astrophysical mechanism studied in the z-pinch experiments is the interaction of a toroidal magnetic field with a plasma ambient medium, leading to the formation of jets and magnetic "bubbles". The plasma ablated from the wires is accelerated vertically filling the space (few cm) above the array. Below the wires there is only a toroidal magnetic field. The initial formation of the magnetic cavity, and jet occurs at the time when the magnetic pressure is large enough to break through the wires. This occurs only over a small region close to the central electrode, where the vacuum toroidal field is strongest. The results show the system evolving into a structure consisting of an approximately cylindrical magnetic cavity with an embedded jet on its axis confined by the magnetic “pinching" force. A shell of swept-up plasma surrounds and partially confines the magnetic bubble. The subsequent evolution is dominated by current-driven instabilities and the development of the asymmetric “kink” mode (m = 1) which leads to a distortion of the jet and a re-arrangement of the magnetic field. The end result of the instabilities however is not to destroy the jet, but to produce an inhomogeneous or “knotty" jets. Simulations show that the resulting jet has typical super-fast-magnetosonic Mach numbers in excess of 5, it is kinetically dominated and its opening angle < 20^o.


MAGPIE experimental images of XUV self-emission
268 ns
278 ns
288 ns
298 ns
time
Synthetic XUV images from 3D GORGON simulations
Image credit: [Ciardi et al., 2007]

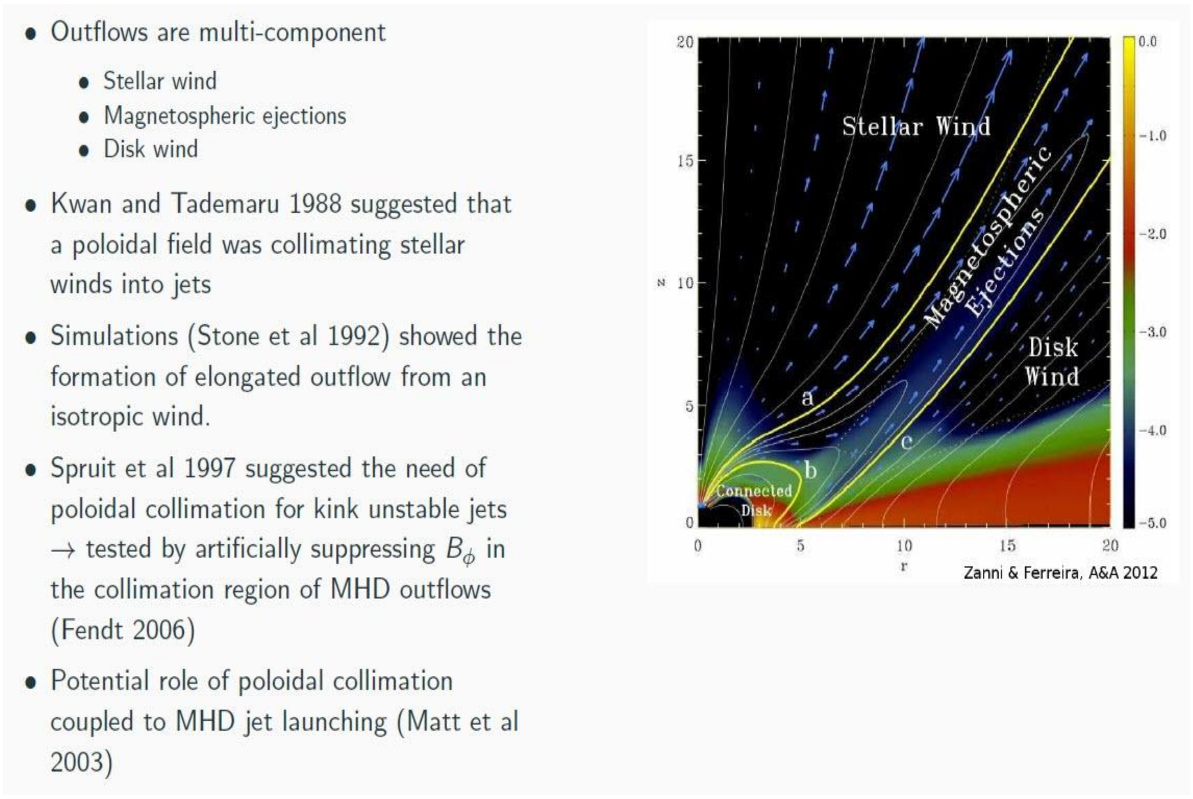
10.2.5 Poloidal collimation in the astrophysical context

It is worth pointing out that while two-dimensional, axisymmetric MHD simulations, reproduce very well the experimental results up to the development of the non asymmetric current-driven instabilities, there are fundamental differences in the long term evolution of the system, which can only be captured by fully three-dimensional simulations.

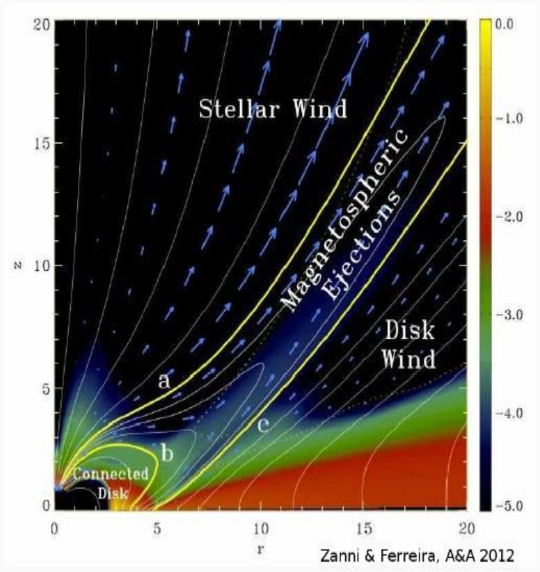


10.3 Mass accretion in stars

Mass ejection in young stars is thought to be multi-component: stellar wind, magnetospheric ejections and disk wind. These components, and their magnetic field, interact with each other in complex ways and a complete picture of their dynamics does not yet exist.

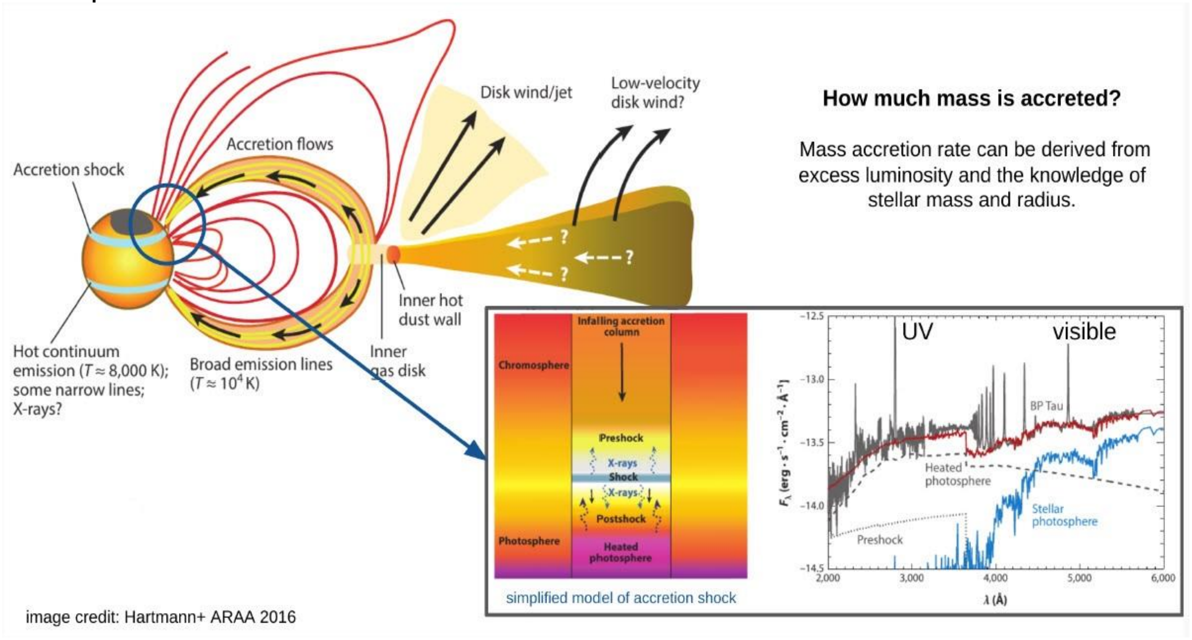


- Outflows are multi-component
  - Stellar wind
  - Magnetospheric ejections
  - Disk wind
- Kwan and Tademaru 1988 suggested that a poloidal field was collimating stellar winds into jets
- Simulations (Stone et al 1992) showed the formation of elongated outflow from an isotropic wind.
- Spruit et al 1997 suggested the need of poloidal collimation for kink unstable jets → tested by artificially suppressing B_ϕ in the collimation region of MHD outflows (Fendt 2006)
- Potential role of poloidal collimation coupled to MHD jet launching (Matt et al 2003)



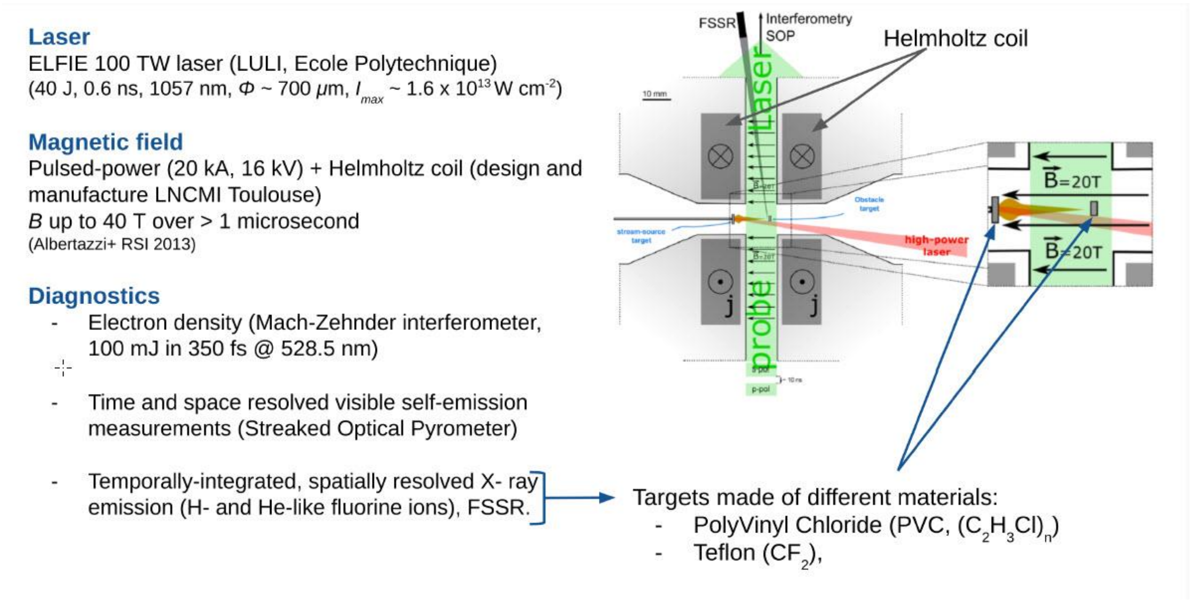
10.3.1 Magnetospheric accretion: mass transfer from disk to stars

The accretion of mass from a disk is mediated by the magnetic field of the star. Magnetized accretion funnels channel the mass onto the stellar surface where an “accretion” shock is formed. Emission in the x-ray band from the shock is reprocessed by the environment and observed as an “excess luminosity” with respect to the stellar photosphere emission.


Accretion shock
Accretion flows
Disk wind/jet
Low-velocity disk wind?
How much mass is accreted?
Mass accretion rate can be derived from excess luminosity and the knowledge of stellar mass and radius.
Inner hot dust wall
Hot continuum emission ($T \approx 8{,}000$ K); some narrow lines; X-rays?
Broad emission lines ($T \approx 10^4$ K)
Inner gas disk
Infalling accretion column
Chromosphere
Preshock
X-rays
Shock
Postshock
Photosphere
Heated photosphere
simplified model of accretion shock
UV
visible
BP Tau
Heated photosphere
Stellar photosphere
Preshock
F_λ (erg · s^{-1} · cm^{-2} · Å^{-1})
λ (Å)
image credit: Hartmann+ ARAA 2016

10.3.2 Magnetized accretion columns in the laboratory

The jet and target material were made of different materials. Using x-ray spectrometry the “mixing” of the two materials was quantified.



Laser
ELFIE 100 TW laser (LULI, Ecole Polytechnique)
(40 J, 0.6 ns, 1057 nm, $\Phi \sim 700\ \mu$m, $I_{max} \sim 1.6 \times 10^{13}$ W cm^{-2})

Magnetic field
Pulsed-power (20 kA, 16 kV) + Helmholtz coil (design and manufacture LNCMI Toulouse)
B up to 40 T over > 1 microsecond
(Albertazzi+ RSI 2013)

Diagnostics
- Electron density (Mach-Zehnder interferometer, 100 mJ in 350 fs @ 528.5 nm)
- Time and space resolved visible self-emission measurements (Streaked Optical Pyrometer)
- Temporally-integrated, spatially resolved X- ray emission (H- and He-like fluorine ions), FSSR.

Targets made of different materials:
- PolyVinyl Chloride (PVC, $(C_2H_3Cl)_n$)
- Teflon (CF_2),

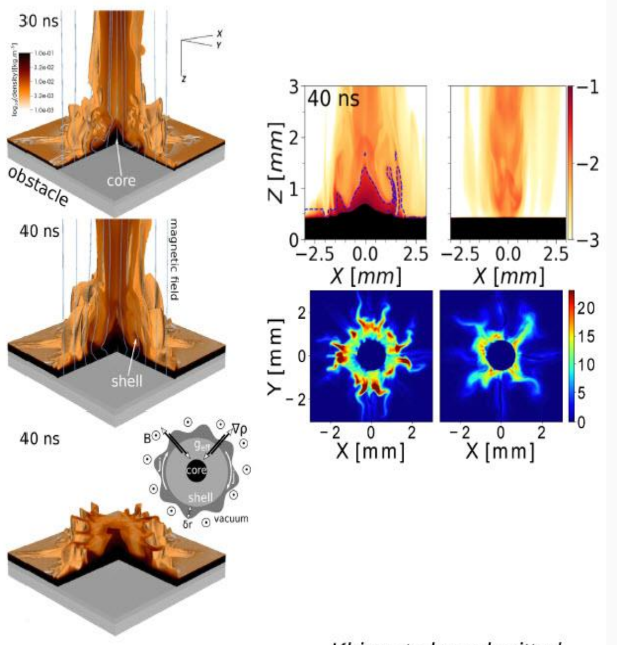
11.3.3 Magnetized laboratory accretion columns

Impact of the accretion ow ablates the target material and it is ejected to distances of ~2 mm from the initial target surface. The shell of material surrounding the accretion

shock is mostly made of target material. Astrophysical simulations show a similar effect, with the chromosphere making up most of the dense shell.
Laboratory simulations show that the shell interface with the external medium (the corona) is unstable to the magnetized Rayleigh-Taylor instability. The instability can generate more material mixing. The take away message is that 2D simulations are unable to capture instabilities and chromospheric ablation can be important.

- "Chromosphere/photosphere" is ablated by the "accretion flow"
- Experiments indicate that obstacle material is ejected "large" distances ~ 2 mm
- Obstacle is ejected alongside with the post-shock accretion plasma leading to mixing
- Shell is mostly made of obstacle material
- Astrophysical simulations show the same effect → absorption by the shell of emission from accretion shock
- Unstable shell (which is mostly chromosphere/photosphere) interface is MRTI

  → 2D simulations are insufficient and that "correct" of chromosphere/photosphere is important



Khiar+ to be submitted

10.4 Magnetic reconnection and particle acceleration

10.4.1 Introduction

Magnetic reconnection changes the topology of the magnetic field and releases magnetic energy in the form of bulk plasma kinetic energy, thermal energy and energetic charged particle. Astrophysical systems usually have very large Lundquist number ($S>>10^{10}$) (i.e. large ratio of Ohmic diffusion time to the crossing time of Alfven waves).

- the reconnection rate in collisionless plasma is faster than in the standard Sweet-Parker reconnection rate

In magnetic reconnection, the energy of the magnetic field is converted in heating, bulk fluid motion and accelerated particles

Outside the current sheet:

$$\vec{E} + \vec{v} \times \vec{B} = 0$$

in the current sheet:

$$\vec{E} + \vec{v} \times \vec{B} = \eta \vec{j} \neq 0$$

from force balance, steady-state and conservation of mass, the (dimensionless) reconnection rate is

$$M = \frac{v_{in}}{vAlfv} = \frac{\delta}{L} = \frac{1}{\sqrt{S}}$$

where the Lunquist number is

$$S = \frac{\tau_{Ohmic}}{\tau_{Alfv}} = \frac{\mu_0 v_A L}{\eta}$$

However multi-fluid effects, kinetic effects, instabilities, etc. are generally very important in determining the reconnection rate

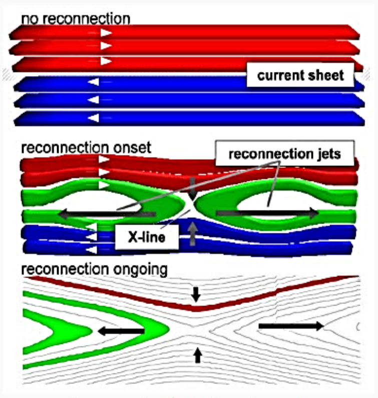


image credit: [Vaivads et al., 2006]

10.4.2 Particle acceleration in magnetic reconnection

Several particle acceleration mechanism are possible in the reconnection layer. For typical parameters for the Omega Laser Facility, electrons are expected to gain energies ~ 25-75 keV, which should be compared to their thermal energy ~1 keV.

Acceleration mechanisms in the reconnection layer

→ reconnection electric field ~ vB

→ Fermi-type due to the motion of plasmoids

→ Fermi-type

→ betatron

→ ...

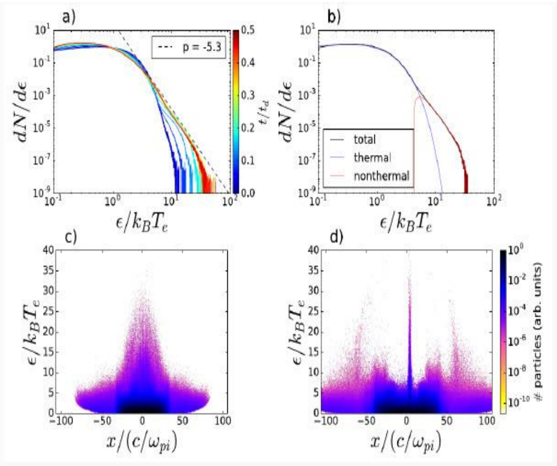


image credit: [Totorica et al., 2017]

10.4.3 Magnetic reconnection in z-pinch experiments: anomalous ion Heating

Experiments on magnetic reconnection in high-energy density plasmas are also carried out on other facilities besides lasers. On z-pinch machines, wire ablation is used to generate converging, magnetized plasma flows that meet to generate a reconnection layer. Because of the longer time- and spatial scales, these experiments provide in general easier diagnostic access than laser experiments. The magnetic field measurements by Faraday rotation show a Harris-type current sheet, indicative of magnetic reconnection. Temperature measurements with Thomson scattering show anomalous ion heating Ti > Te.

Reconnection experiments [Hare et al., 2017] on the MAGPIE generator (1.4 MA, 500 ns current pulse). Carbon wires ($\phi = 400\ \mu$m).

$\rightarrow$ size of reconnection layer: 1×20 mm

Magnetic field measurements by Faraday rotation clearly show a Harris-type current sheet, indicative of magnetic reconnection

Temperature measurements with Thomson scattering show anomalous ion heating $T_i > T_e$

$\rightarrow$ micro-instabilities?

$\rightarrow$ plasmoids?

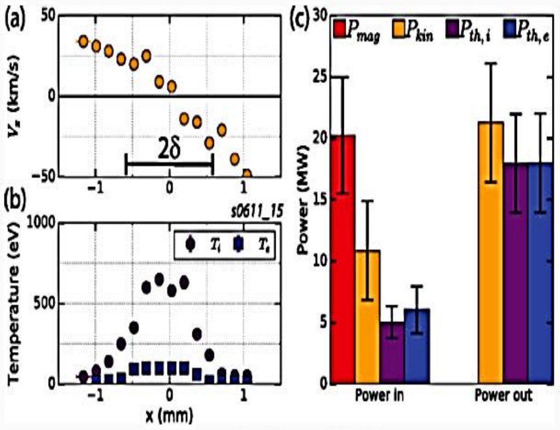


image credit: [Hare et al., 2017]

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

Chapter 11: Atomic Physics Simulations for Plasmas

J. Limpouch

11. Atomic Physics Simulations for Plasmas

Detailed knowledge of atomic physics of laser-produced plasmas (LPP) is important both for plasma diagnostics and for applications of LPP as a short-wavelength radiation source. Emission and absorption spectra are very important tools of plasma diagnostics containing information about electron and ion temperatures, plasma density and ion charge distribution. Laser-produced plasmas from targets of medium and heavy elements are efficient pulsed point-like sources of extreme ultraviolet (EUV) and X-ray radiation that are suitable for applications such as EUV lithography and imaging of biological objects in the water-window spectral range. For these applications, suitable laser and target parameters have to be found and optimized.

Detailed simulations of atomic physics include two basic tasks. First, atomic structure modelling is performed to calculate energy levels, wave functions and transition probabilities. Second,populations of ionization and excitation states are calculated for given density, temperature and plasma size or for known history of the above parameters. External source of radiation can be also taken into account. Then, plasma emission or absorption spectra may be synthetized. The second task can also be solved as a post-processor to a plasma dynamics code.

Various simplified approaches that are used for calculations of mean ion charge and for the radiative transport of energy in the fluid approach to simulations of plasma dynamics are not discussed in this section.

11.1 Introduction to simulations

Atomic structure codes can be divided into 2 groups - codes solving non-relativistic Schrodinger equation with relativistic corrections and fully relativistic codes solving Dirac equation. Nonrelativistic codes include the configuration interaction codes CIV3 [1] and SUPERSTRUCTURE [2], the multi-configuration Hartree-Fock code MCHF [3] and the very popular Cowan code [4]. Fully relativistic codes include the HULLAC package [5], the SZ code [6] and the popular modern FAC (Flexible Atomic Code) code [7].

The fully relativistic approach based on the fully consistent physics description is more computationally demanding but it can be performed on modern personal workstations. However, students at advanced bachelor and master levels are well familiar with the Schr¨odinger equation that is the base for the classical approach and the interpretation of code results is significantly simpler. Thus, this subsection is based on the classical approach with relativistic corrections. Here, we shall briefly explain the atomic structure and the process of its calculation essentially according to the Cowan's book [4].

11.1.1 Basic notations

Ionization degrees are often marked by Roman numerals where I denotes neutral atom, II is singly ionized ion, III doubly ionized ion and so on. Generally, Roman numeral is equal to the ionization degree+1. For example, the triply ionized boron is denoted B IV. On the other hand, there is certain similarity of spectral features of ions with the same number of bound electrons. The emission spectra of ions with only one bound electron have a certain similarity with the spectrum of neutral hydrogen, so these ions are classified as H-like (hydrogen-like). Similarly ions with two bound electrons are called He-like (helium-like). Thus, ion B IV is called helium-like boron, as it has two remaining bound electrons. Atomic states of a particular ion include ground state (the state with minimum energy for an ion with given degree of ionization), excited states (also called resonance states) with one electron from the outmost shell excited to an upper shell or subshell and autoionization states with energy higher than the ionization potential. Autoionization states have either one electron excited from an inner shell or more excited electrons. Spontaneous ionization is possible from an autoionization state.

We shall use here cgs units. Radii will be normalized to the Bohr radius $a_0 = \mathrm{n}^2/m_e\, e^2 = 5.29177\ 10^{-9}$ cm and energies will be expressed in Rydbergs 1 Ry = 1 $R_\infty = e^2/2\, a_0 = 13.6058$ eV = 109737.3 cm^{-1}.

11.1.2 One-electron atom (ion)

The electron of mass m_e is moving in one-electron ion in the central electrostatic potential $V(r)$ of the nucleus. The electron energy E is an eigenvalue of the Hamiltonian operator $\mathbf{H}$. The electron wave function φ is found by solving the time-independent Schrödinger equation

$$\mathbf{H}\varphi = E\varphi \tag{1.1}$$

$$\mathbf{H} = \frac{p^2}{2m_e}+V(r) = \frac{p_r^2}{2m_e}+\frac{\mathbf{L}^2}{2m_e}+V(r) = -\frac{\hbar^2}{2m_e}\left[\frac{1}{r}\frac{\partial^2}{\partial r^2}r + \frac{1}{r^2\sin\theta}\left(\frac{\partial}{\partial\theta}\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial\phi^2}\right]+V(r)$$

where the electrostatic potential of a nucleus with the charge number Z is $V(r) = -Ze^2/r$. When the distances and energies are measured in units of the Bohr radius a_0 and units of Rydberg Ry, the time-independent Schrödinger equation is written, as follows

$$\left[-\frac{1}{r}\frac{\partial^2}{\partial r^2}r + \frac{\mathbf{L}^2}{r^2} - \frac{2Z}{r}\right]\varphi = E\varphi\ . \tag{1.2}$$

In any central field problem angular momentum $\mathbf{L}$ is a constant of motion. Quantum mechanically, we may expect that φ should be an eigenfunction of $\mathbf{L}^2$ and $\mathbf{L}_z$. Indeed, this is true as $\mathbf{L}^2$ is the only term in the Schrödinger equation dependent on θ and ϕ.

The solution may be written in the form

$$\varphi(\vec{r}) = \varphi_{nlm_lm_s}(\vec{r}) = \frac{1}{r}P_{nl}(r)\cdot Y_{lm_l}(\theta,\phi)\cdot\sigma_{m_s}(s_z)\ , \tag{1.3}$$

where $n = 1,2,\ldots$ is the principal quantum number, $l = 0,1,\ldots,n-1$ characterizes the orbital angular momentum, $m_l = -l,-l+1,,l-1,l$ is the orientation of the orbital angular momentum and $m_s = -1/2, 1/2$ is the orientation of the spin angular momentum.

All operators of angular momentum have following properties (here written for arbitrary angular momentum J). Operator $\mathbf{J}^2 = \vec{\mathbf{J}}\cdot\vec{\mathbf{J}} = \mathbf{J}_x^2+\mathbf{J}_y^2+\mathbf{J}_z^2$ has eigenvalues $j\,(j+1)\,\hbar^2$, where $j = 0, 1/2, 1, 3/2, 2, \ldots$ and J_z has eigenvalues $m\hbar$ $(m = -j, -j+1, -j+2, \ldots, j-1, j)$. Eigenfunctions of orbital angular momentum operators $\mathbf{L}^2$, $\mathbf{L}_z$ are

$$Y_{lm}(\theta,\phi) = \Theta_{lm}(\theta)\Phi_m(\phi) = (-1)^{(m+|m|)/2}\left[\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}\right]^{1/2} P_l^{|m|}(\cos\theta)\exp(im\phi)\ . \tag{1.4}$$

Spherical harmonics are orthonormal and sum over m

$$\sum_{m=-l}^{l}|Y_{lm}(\theta,\phi)|^2 = \frac{2l+1}{4\pi}$$

is spherically symmetric. Thus, the electron density in any fully occupied subshell is spherically symmetric. States with angular momentum $l = 0$ are denoted by s, p denotes $l = 1$, d $(l = 2)$, f $(l = 3)$, g $(l = 4)$, h $(l = 5)$, i $(l = 6)$, k $(l = 7)$, l $(l = 8)$, m $(l = 9)$.

Electron has an intrinsic angular momentum - spin $j = 1/2$ (z component $m_s = -1/2, 1/2$), eigenfunction is $\sigma_{m_s}(s_z) = \delta_{m_s s_z}$. States with differences in quantum numbers are orthonormal $\left\langle Y_{lm_l}\sigma_{m_s} | Y_{l'm_l'}\sigma_{m_s'}\right\rangle = \delta_{ll'}\delta_{m_l m_l'}\delta_{m_s m_s'}$.

Radial part of wavefunction

The wavefunction vanishes at the position of nucleus $P_{nl}(0) = 0$. For free states ($E_{\varepsilon l} > 0$), open boundary condition at $r \to \infty$ leads to continuous energy spectrum. For the bound states ($E_{nl} < 0$), only discrete energy levels are allowed as the wavefunction must vanish for $r \to \infty$. The equation for the radial part of wavefunction is expressed, as follows

$$\left[-\frac{\mathrm{d}^2}{\mathrm{d}r^2} + \frac{l(l+1)}{r^2} - \frac{2Z}{r}\right] P_{nl}(r) = E P_{nl}(r)\ . \tag{1.5}$$

Here, the effective potential is $V_{eff}(r) = V(r) + l(l+1)/r^2 = -2Z/r + l(l+1)/r^2$. After substitution $\rho = 2Zr/n$ and $E = -Z^2/n^2$, the wave equation is transformed to the following form

$$\left[\frac{\mathrm{d}^2}{\mathrm{d}\rho^2} - \frac{1}{4} + \frac{n}{\rho} - \frac{l(l+1)}{\rho^2}\right] P_{nl}(r) = 0\ . \tag{1.6}$$

The analytic solution is known and can be expressed, as follows

$$P_{nl}(r) = -\left[\frac{Z(n-l-1)!}{n^2(n+l)!^3}\right] \rho^{l+1} \mathrm{e}^{-\rho/2} \mathrm{L}_{n+l}^{2l+1}(\rho)\ , \tag{1.7}$$

where the associated Laguerre polynomial is

$$\mathrm{L}_{n+l}^{2l+1}(\rho) = -(n+l)!^2 \sum_{k=0}^{n-l-1} \frac{(-\rho)^k}{k!(n-l-1-k)!(2l+1+k)!}\ .$$

The phase of general solution is arbitrary, here we use the convention $P_{nl}(r) > 0$ for $r \to 0$. Number of nodes is $n - l - 1$, number of antinodes is $n - l$.

The Bohr-Sommerfeld orbits have elliptical form. The orbit with $l = n - 1$ is spherical while with increasing n for given l the orbit is increasingly elongated in the direction of main axis with increasing maximum distance from nucleus but also with decreasing distance of the orbit point nearest to the nucleus. The close approach of electron in state s with high principal quantum number n to the nucleus causes an increase in the binding energy and leads to the anomaly for neutral atoms of transition elements, lanthanides and actinides when s state of higher principal number is occupied prior to d and f states of lower quantum number. This anomaly exists also in some singly and doubly ionized high-Z ions, but it never occurs for higher ionization degrees.

For atoms (ions) with more than 1 bound electron, the effective potential has no simply expressible form and analytic solution for the wavefunction cannot be found. During numerical solving, the value of energy is iterated and the number of nodes should correspond to the particular state.

Relativistic corrections

Relativistic corrections to the Hamiltonian must be added to reach qualitative and quantitative agreement with the observed line splitting and transition energies. The Hamiltonian then reads

$$\mathbf{H} = -\nabla^2 + V - \frac{\alpha^2}{4}(E - V)^2 - \frac{\alpha^2}{4}\left(\frac{\mathrm{d}V}{\mathrm{d}r}\right)\frac{\partial}{\partial r} + \frac{\alpha^2}{2r}\left(\frac{\mathrm{d}V}{\mathrm{d}r}\right)\left(\vec{\mathrm{l}} \cdot \vec{\mathrm{s}}\right)\ , \tag{1.8}$$

where the fine structure constant $\alpha = e^2/(\hbar c) = \hbar/(mca_0) = 1/137.036$ and $V = -2Z/r$ for hydrogen-like ions. The 3^{rd} term of the Hamiltonian is the mass-velocity term caused by the relativistic dependence of the electron mass on its velocity. The 4^{th} term is the Darwin term expressing the relativistic correction caused by the uncertainty in the electron position. The 5^{th} term is the spin-orbital term caused by the magnetic interaction of the spin and orbital magnetic moments. While the mass-velocity and Darwin terms lead just to shifts in energies of the levels, spin-orbit coupling leads to splitting of the energy levels with $l \neq 0$. Operator $\vec{\mathbf{l}}{\cdot}\vec{\mathbf{s}} = \left(\mathbf{j}^2 - \mathbf{l}^2 - \mathbf{s}^2\right)/2$ has eigenvalues $X = [j(j+1) - l(l+1) - s(s+1)]$ that are simplified to $X = l$ for $j = l + 1/2$, and $X = -l - 1$ for $j = l - 1/2$. The radial part of the wave functions depends on quantum numbers n, l and newly additionally on j and it can be found in principle directly from radial part of Schrödinger equation. However, we shall find energy corrections via perturbative approach using classical $P_{nl}(r)$ and calculating the energy perturbations due to the relativistic correction terms $\mathbf{H}_t$ of Hamiltonian

$$\delta E_t = \int_0^\infty P_{nl}^* \mathbf{H}_t P_{nl} \mathrm{d}r \ .$$

The energy correction δE_m due to the mass-velocity term and δE_D due to the Darwin term are expressed, as follows

$$\delta E_m = -\frac{\alpha^2 Z^4}{4n^4}\left[\frac{4n}{l+1/2} - 3\right] \qquad \delta E_D = \delta_{l0}\frac{\alpha^2 Z^4}{n^3} \ . \tag{1.9}$$

The shift of energies due to the spin-orbit coupling is expressed, as follows

$$\delta E_{so} = \alpha^2 Z X \left\langle r^{-3}\right\rangle = (1 - \delta_{l0})\frac{\alpha^2 Z^4}{n^3 l(l+1)(2l+1)}\left[j(j+1) - l(l+1) - s(s+1)\right] \ . \tag{1.10}$$

The energy of the state nlj is thus $E_{nlj} = -Z^2/n^2 + \delta E_m + \delta E_D + \delta E_{so}$. All relativistic corrections are proportional to Z^4 and they grow faster with Z than the basic term proportional to Z^2.

11.1.3 Multielectron atom (ion)

As the mass-velocity and the Darwin terms lead to shifts in energies of atomic states only, we shall omit them first when solving Schrödinger equation and the energy shifts will be then calculated for the previously obtained wavefunctions via perturbative approach. Thus, we shall use the following Hamiltonian

$$\mathbf{H} = \mathbf{H}_{kin} + \mathbf{H}_{e-nuc} + \mathbf{H}_{e-e} + \mathbf{H}_{so} = -\sum_i \nabla_i^2 - \sum_i \frac{2Z}{r_i} + \sum_i \sum_{j<i} \frac{2}{r_{ij}} + \sum_i \xi_l(r_i)\left(\vec{\mathbf{l}}_i \cdot \vec{\mathbf{s}}_i\right) \ . \tag{1.11}$$

The spin-orbit interaction has a profound effect on the energy level structure, so it has to be retained in the Hamiltonian. The proportionality factor ξ_l has been left in unspecified form. If a suitable energy-potential $V(r)$ is used, the factor $\xi_l = \alpha^2/2r \times (\mathrm{d}V/\mathrm{d}r)$.

Wavefunctions $\Psi^k(\vec{r})$ of complex atoms are solutions of the Schrödinger equation and they are expressed as linear combination of basis functions ψ_b

$$\mathbf{H}\Psi^k(\vec{r}) = E^k \Psi^k(\vec{r}) \qquad \Psi^k(\vec{r}) = \sum_b y_b^k \psi_b \ . \tag{1.12}$$

Basis functions are orthonormal and they form a complete set of functions. In general, the set consists of infinite number of functions so the equation represents a sum of an infinite series. In practice, it is necessary to truncate the series, and thus it is essential to choose a suitable type of basis functions. When scalar products of the above equation with each basis function are carried out, a homogeneous system of linear equations for the coefficients y_b^k is obtained and energies E^k are the eigenvalues of the matrix.

The basis functions are constructed from one-electron wavefunctions (also called "spin-orbitals") $\varphi_i(\vec{r}_i)$. Pauli exclusion principle requires basis functions antisymmetric upon exchange of any 2 electrons. Such functions can be formed by using determinantal functions of the form

$$\psi_b = (N!)^{-1/2} \sum_P (-1)^p \varphi_1(\vec{r}_{j_1})\varphi_2(\vec{r}_{j_2}) \dots \varphi_N(\vec{r}_{j_N}) \ , \tag{1.13}$$

where $p = 0$ for even permutations and $p = 1$ for odd permutations. In the above formula $\vec{r}$ includes also spin $\vec{s}$. The antisymmetrized functions eliminates the possibility of any two orbitals being identical as this would lead to $\psi_b = 0$. Moreover, electrons with the same spin cannot lie in the same position, thus ψ_b must be very small when electrons with same spin are close together. Thus, the antisymmetric basis wavefunctions include correlations of electrons with the same spin via Pauli exclusion principle. However, electrons with opposite spins are not correlated as the basis functions do not contain any correlation due to Coulomb repulsion. The basis functions ψ_b are mutually orthonormal. Each of the basis functions is an eigenfunction of the operator $\mathbf{J}_z$ since each term involves the same set of one-electron quantum numbers. As the resulting wavefunctions Ψ^k have to be eigenfunctions of the operator $\mathbf{J}^2$, it is favourable to use basis functions ψ_b that are already eigenfunctions of the above operator. However, functions constructed by multiplication of one electron functions must be coupled to become eigenfunctions of $\mathbf{J}^2$, the coupling procedure will be described later.

The electrons with the same n, l form a subshell and they are called equivalent electrons. The list of N pairs $n_i l_i$ defines configuration. If the occupation number of electrons in a subshell k is denoted by w_k then the configuration is specified by means of the following notation

$$(n_1 l_1)^{w_1} (n_2 l_2)^{w_2} \dots (n_h l_h)^{w_h} \ , \quad \text{where} \ \sum_{k=1}^{h} w_k = N \ .$$

A fully occupied subshell k (s^2, p^6, d^{10}, $f^{14}, \dots$) is called closed and its angular momenta $L_k = S_k = J_k = 0$. Closed subshells are usually skipped in brief notations, thus configuration Ne I $1s^2 2s^2 2p^5 3s$ is written as Ne I $2p^5 3s$.

Configuration-average energies and radial wavefunctions

The configuration-average energy E_{av} is the mean value of energy of a set of all basis functions belonging to the particular configuration, i.e. all allowed combinations of orbital and spin angular momenta of all electrons. It is expressed, as follows

$$E_{av} = \langle b|\mathbf{H}|b\rangle_{av} = \sum_b \langle b|\mathbf{H}|b\rangle /M = \sum_b E^b/M \ ,$$

where M is the number of basis functions. E_{av} is also the energy of the spherically averaged atom. Spin-orbital contributions to E_{av} cancel due to the presence of the functions with spins $\pm 1/2$ in the set.

The contribution of interaction between electrons i, j consists of direct and exchange terms, thus the configuration-average energy is expressed, as follows

$$E_{av} = \sum_i \langle i| - \nabla^2 |i\rangle_{av} - \sum_i \left\langle i|\frac{2Z}{r_1}|i\right\rangle_{av} + \sum_i \sum_{j<i} \left[\left\langle ij|\frac{2Z}{r_{12}}|ij\right\rangle_{av} - \left\langle ij|\frac{2Z}{r_{12}}|ji\right\rangle_{av} \right] \ , \tag{1.14}$$

basis functions. When scalar products of the above equation with each basis function are carried out, a homogeneous system of linear equations for the coefficients y_b^k is obtained and energies E^k are the eigenvalues of the matrix.

The basis functions are constructed from one-electron wavefunctions (also called "spin-orbitals") $\varphi_i(\vec{r}_i)$. Pauli exclusion principle requires basis functions antisymmetric upon exchange of any 2 electrons. Such functions can be formed by using determinantal functions of the form

$$\psi_b = (N!)^{-1/2} \sum_P (-1)^p \varphi_1(\vec{r}_{j_1}) \varphi_2(\vec{r}_{j_2}) \dots \varphi_N(\vec{r}_{j_N}) \quad , \tag{1.13}$$

where $p = 0$ for even permutations and $p = 1$ for odd permutations. In the above formula $\vec{r}$ includes also spin $\vec{s}$. The antisymmetrized functions eliminates the possibility of any two orbitals being identical as this would lead to $\psi_b = 0$. Moreover, electrons with the same spin cannot lie in the same position, thus ψ_b must be very small when electrons with same spin are close together. Thus, the antisymmetric basis wavefunctions include correlations of electrons with the same spin via Pauli exclusion principle. However, electrons with opposite spins are not correlated as the basis functions do not contain any correlation due to Coulomb repulsion. The basis functions ψ_b are mutually orthonormal. Each of the basis functions is an eigenfunction of the operator $\mathbf{J}_z$ since each term involves the same set of one-electron quantum numbers. As the resulting wavefunctions Ψ^k have to be eigenfunctions of the operator $\mathbf{J}^2$, it is favourable to use basis functions ψ_b that are already eigenfunctions of the above operator. However, functions constructed by multiplication of one electron functions must be coupled to become eigenfunctions of $\mathbf{J}^2$, the coupling procedure will be described later.

The electrons with the same n, l form a subshell and they are called equivalent electrons. The list of N pairs $n_i l_i$ defines configuration. If the occupation number of electrons in a subshell k is denoted by w_k then the configuration is specified by means of the following notation

$$(n_1 l_1)^{w_1} (n_2 l_2)^{w_2} \dots (n_h l_h)^{w_h} \ , \quad \text{where} \ \sum_{k=1}^{h} w_k = N \ .$$

A fully occupied subshell k (s^2, p^6, d^{10}, $f^{14}, \dots$) is called closed and its angular momenta $L_k = S_k = J_k = 0$. Closed subshells are usually skipped in brief notations, thus configuration Ne I $1s^2 2s^2 2p^5 3s$ is written as Ne I $2p^5 3s$.

Configuration-average energies and radial wavefunctions

The configuration-average energy E_{av} is the mean value of energy of a set of all basis functions belonging to the particular configuration, i.e. all allowed combinations of orbital and spin angular momenta of all electrons. It is expressed, as follows

$$E_{av} = \langle b|\mathbf{H}|b\rangle_{av} = \sum_b \langle b|\mathbf{H}|b\rangle / M = \sum_b E^b / M \ ,$$

where M is the number of basis functions. E_{av} is also the energy of the spherically averaged atom. Spin-orbital contributions to E_{av} cancel due to the presence of the functions with spins $\pm 1/2$ in the set.

The contribution of interaction between electrons i, j consists of direct and exchange terms, thus the configuration-average energy is expressed, as follows

$$E_{av} = \sum_i \langle i| - \nabla^2 |i\rangle_{av} - \sum_i \left\langle i| \frac{2Z}{r_1} |i \right\rangle_{av} + \sum_i \sum_{j<i} \left[\left\langle ij| \frac{2Z}{r_{12}} |ij \right\rangle_{av} - \left\langle ij| \frac{2Z}{r_{12}} |ji \right\rangle_{av} \right] \ , \tag{1.14}$$

one assumes some a_0 setting $P \simeq a_0 r^{l+1}$ for $r \to 0$ and solves inhomogeneous and homogeneous (setting all $\varepsilon_{ij} = 0$, $B_{ij} = 0$) HF equations. Then $P_i(r) = P^I(r) + \beta P^H(r)$, where P^I, P^H are the solutions of the inhomogeneous and homogeneous HF equations, respectively, and β is chosen so that $P_i(r) \to 0$ for $r \to \infty$. Then the trial function $P^{(m+1)}$ for the next iteration is set by the relaxation relation

$$P_i^{(m+1)}(\text{input}) = cP_i^{(m)}(\text{output}) + (1-c)P_i^{(m)}(\text{input}) \;, \tag{1.18}$$

where $c = \langle 0.05, 1.1 \rangle$, and usually $c \simeq 0.5$. The parameter ε_i was originally introduced as the Lagrangian multiplier associated with the normalization condition on $P_i(r)$. It must be chosen by means of a secondary iterative procedure so that $||P_i|| = 1$. The additional requirements that a_0 must be positive and that $P_i(r)$ has $n - l - 1$ nodes lead to a unique solution for $P_i(r)$.

There are several complications when solving SCF equations. For high orbits of neutral and lowly ionized atoms SCF need not converge, but the convergence has to be achieved via relaxation of the inhomogeneous part of the HF equations. Other ways are preferred for calculating integrals with P_i in the denominator, as the nodes of P_i lead to singularities in the integrand.

There are several approximate local-potential methods for calculating radial wavefunctions and binding energies. Homogeneous equation is solved with an assumed potential $V(r)$ in all these methods. They are free of complexities encountered in the inhomogeneous equation and their solutions are suitable for calculation of the relativistic corrections, spin orbit parameters etc. Thomas-Fermi (TF) and Thomas-Fermi-Dirac (TFD), which is TF with added exchange potential, methods use semi-free-electron atom. They are simple, but the applied potential is poor. Hartree method uses $V(r) = -2Z/r + V_H(r)$, but misses the exchange effects. In Hartree-Fock-Slater (HFS) method, the exchange effects are added in the same way as in TFD method. The most accurate replacement of HF term is the Hartree-plus-Statistical-Exchange (HX) method. The potential is set to $V_i(r) = -2Z/r + V_H(r) + V_x(r)$, where $V_x(r)$ is a statistically based approximation for non-self-exchange terms. The HX approximation is often used as it is the most accurate known approximation.

Detailed energy level structure

Energy level splitting inside one configuration is caused by the coupling of angular momenta. The coupled function of two angular momenta is expressed via the Clebsch-Gordon coefficients C, as follows

$$|j_1 j_2 j m\rangle = \sum_{m_1=-j_1}^{j_1} C(j_1 j_2 m_1, m - m_1; jm) \, |j_1 j_2 m_1, m - m_1\rangle = (-1)^{j_1+j_2-j} \, |j_2 j_1 j m\rangle \;. \tag{1.19}$$

It is eigenfunction of 4 operators $\mathbf{J}_1^2$, $\mathbf{J}_2^2$, $\mathbf{J}^2 = (\mathbf{J}_1 + \mathbf{J}_2)^2$ and $\mathbf{J}_z = \mathbf{J}_{1z} + \mathbf{J}_{2z}$. The coupling of two angular momenta is not commutative. The coupling of 3 momenta is even more complicated and it is not associative.

The basic splitting of the energy levels is influenced by the relative importance of various terms in the Hamiltonian. We describe here the basic schemes - **LS** coupling and **jj** coupling. Some configuration behave according to other schemes (**LK** coupling, **jK** coupling) and some configurations have an intermediate coupling, meaning that they cannot be assigned to any general coupling scheme.

The **LS** coupling is characteristic for low Z atoms where Coulomb repulsions dominate spin-orbit interaction. The basic splitting is according to the total orbital angular momentum $\mathbf{L} = \sum_i \vec{\mathbf{l}}_i$ and the total spin $\mathbf{S} = \sum_i \vec{\mathbf{s}}_i$. Then $\mathbf{L}$ and $\mathbf{S}$ are coupled together to give eigenfunctions of $\mathbf{J}^2$, $\mathbf{J}_z$.

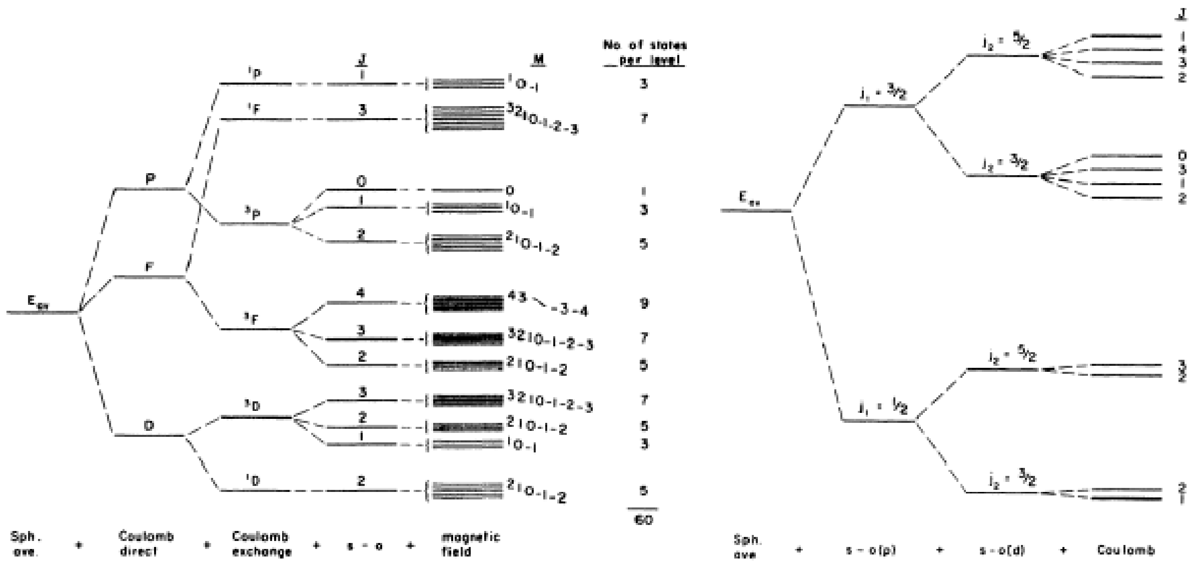

No of states per level
J
M
E_av
P
F
D
¹P
¹F
³P
³F
³D
¹D
j₁ = 3/2
j₁ = 1/2
j₂ = 5/2
j₂ = 3/2
Sph. ave. + Coulomb direct + Coulomb exchange + s-o + magnetic field
60
Sph. ave + s-o(p) + s-o(d) + Coulomb

Figure 1: Energy level splitting of a ***pd*** configuration under **LS** coupling (left panel) and **jj** coupling (right panel) (reprinted from [4]).

The usual notation of the levels is $^{2S+1}L_J$, where 2S+1 is the multiplicity due to the total spin, total angular momentum L is represented by letters S, P, D,... for L = 0, 1, 2,... and odd parity is denoted by upper index o, e.g. $^3S_1$, $^2P^o_{1/2}$.

On the other hand high-Z elements where spin-orbit interactions dominate Coulomb repulsions behave according to **jj** coupling. Here, the orbital and spin angular momenta couple for each electron $\vec{l}_i + \vec{s}_i = \vec{j}_i$ first, and then the electron total angular momenta are coupled together. The level notation for 2 electrons is $[(l_1, s_1)\,j_1, (l_2, s_2)\,j_2]$ **JM**.

Schematics of energy level splitting of a ***pd*** configuration under **LS** and **jj** couplings are demonstrated in Fig. 1. Both couplings start from configuration average energy E_{av}. **LS** coupling is adding successively large Coulomb interaction (direct and exchange), spin-orbit interaction and finally external magnetic field. Under **jj** coupling, 2 strong spin-orbit interactions result in 4 energies; small Coulomb interaction then leads to a small splitting according to the total angular momentum **J**.

Coupled antisymmetric basis functions have to be created. For non-equivalent electrons one can start from antisymmetrized wave functions and then couple the angular momenta by means of the Clebsch-Gordon (CG) formula. For equivalent electrons such procedure will fail due to involvement of the combinations forbidden by the Pauli exclusion principle. Mechanism was developed by Racah to form the set of coupled antisymmetric basis functions using coefficients of fractional parentage and seniority number. It is feasible both in **LS** and **jj** coupling (the result can be transferred to other coupling schemes). Recently, a new method of performing angular integration was proposed [8] that is based on the second quantization form of the operators and extends the use of Racah algebra to the quasi-spin space. In this method, instead of recoupling basis states, one recouples the creation and annihilation operators with the help of Racah algebra. The new method simplifies the procedures for complex configurations considerably.

Using the coupled antisymmetric basis functions, the matrix elements $\langle i|H|i\rangle = E_{av} + \Delta H_{ii}$ and $\langle i|H|j\rangle = H_{ij}$ for $i \neq j$ are then calculated. The kinetic energy and electron-nuclear terms make

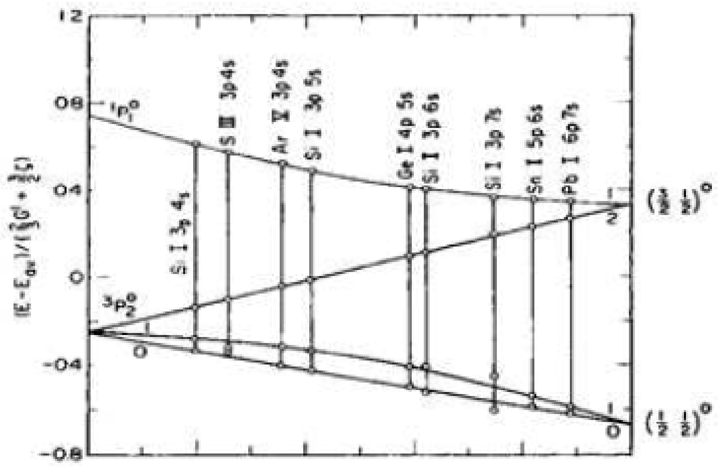


Figure 2: Energy levels for *ps* configuration during transitions from **LS** to **jj** coupling (reprinted from [4]).

no contribution to H_{ij} and ΔH_{ii}. Also closed subshells do not contribute to H_{ij} and ΔH_{ii}. When evaluating matrix elements of symmetric operators, complications due to antisymmetrization can be removed.

In simple configurations, evaluations are much simplified. If only 1 electron is outside closed subshells, no Coulomb interactions are involved in calculation of level structure, there is only $l-s$ interaction. The calculations differ from the calculations for the H atom only by a more complicated potential $V^i(r)$. For 2 non-equivalent electrons outside closed subshells, spin-orbit interactions are easily calculated in **jj** coupling. However, it is still better to use **LS** coupling if the Coulomb interactions are strong. If there are 2 equivalent electrons (l^2 configuration) outside closed subshells, **LS**-coupled functions are antisymmetric without permutation, exchange energy is zero and only even L+S basis functions are allowed. In the intermediate coupling, the Hamiltonian matrix is far from diagonal in any coupling scheme and numerical diagonalization or secular equation have to be used. An example of the transition from **LS** to **jj** coupling with increasing strength of the spin-orbit interaction is demonstrated in Fig. 2.

Up to now we have limited ourselves to a single configuration but the accuracy of the model can be improved by including more configurations. Only configurations with the same parity can influence given configuration. As the Hamiltonian includes only 1- and 2-electron operators, one needs to include only configurations differing at most in 2 orbitals. As the basis set must be kept manageable, a suitable choice of interacting configurations has to be made.

Radiative transitions

Atom (ion) in the excited state j can make a spontaneous radiative transition to the lower energy state i and photon of frequency $\nu_{ji} = (E_j - E_i)/h$ is emitted. There are $2J_i + 1$ degenerate states i and Einstein coefficient $A_{ji} = \sum_{M_i} a_{ji}$. Under isotropic excitation, the emitted intensity in the spectral line is $I(t) = hc\sigma_{ji} g_j A_{ji} N_j(t)$, where $\sigma_{ji} = 1/\lambda_{ji}$ is the inverse wavelength and the quantity $g_j A_{ji}$ is called the weighted spontaneous-emission transition probability.

The most intense transitions are electric dipole (E1) transitions. Spontaneous emission probability for transition from state Ψ_j to state Ψ_i is

$$a_{ji} = \frac{64\pi^4 e^2 \nu^3}{3c^3 h} \left|\langle \Psi_{i0} | \vec{r} | \Psi_{j0} \rangle\right|^2 \ , \tag{1.20}$$

where Ψ_{i0} is the time-independent part of the wavefunction $\Psi_i = \Psi_{i0} \exp(-iE_i t/\hbar)$. The weighted transition probability $g_j A_{ji}$ is then expressed

$$g_j A_{ji} = \frac{64\pi^4 e^2 a_0^2 \nu^3}{3c^3 h} S_{ji} \ ,$$

where S_{ji} is the line strength. The line strength is calculated via integration of the transition matrix term over r and angular coordinates. The line strength S_{ij} is related to the oscillator strength f_{ij} for absorption

$$f_{ij} = \frac{8\pi^2 m_e a_0^2 \nu}{3h\,(2J_i + 1)} S_{ji} = \frac{E_j - E_i}{3\,(2J_i + 1)} S_{ji} \ ,$$

that refers to the total probability of absorption from a specific state of the lower level i to all $(2J_j + 1)$ states of the upper level j. Oscillator strength f_{ji} for emission is usually taken negative and it is given by the relation $f_{ji} = -f_{ij}\,(2J_i + 1)\,/\,(2J_j + 1)$.

Integration over the angular coordinates leads to selection rules excluding E1 transition between certain states. As the electric dipole operator has odd parity, E1 transition can occur only between states of opposite parities. From the properties of angular momentum operator the transitions may occur only if

$$\Delta J = J - J' = 0, \ \pm 1 \quad \text{with not allowed} \quad J = J' = 0 \ .$$

Moreover, the rules allow only transitions with $\Delta M = M - M' = 0$ leading photon linear polarization along z-axis (when observed from xy plane) and $\Delta M = \pm 1$ leading to clock- and counter-clockwise circular polarization in xy plane. In **LS** (spin-orbit) coupling, $\Delta S = S - S' = 0$ as the dipole operator does not include spin. Then the selection rule has the form $\Delta L = L - L' = 0, \pm 1$ and $L = L' = 0$ is excluded. However, the selection rules for **LS** coupling are often violated. Intercombination lines with $\Delta S \neq 0$ are observed for $Z > 5$ due to non-ideal **LS** coupling where the spin-orbit interaction leads to the mixing of states with different spins and the same L and J. Violations of ΔL rule and 2 electron jumps occur due to configuration interactions. Mixing of basis states due to Stark effect can lead to parity change violations.

Magnetic dipole (M1) and electric quadrupole (E2) transitions are usually weaker than E1 transitions. The transition probabilities may be calculated in a similar way as for E1 transitions. Higher multipole transitions are observed rarely.

Photoionization and radiative recombination

Continuum states have to be included in calculation of probabilities of photoionization and radiative recombination. Rydberg series of the bound configurations $\ldots (n_i l_i)^{w_i} nl$ with n = 6,7,8 ,..., ∞ of the center-of-gravity energies $E^{6l}_{av}, E^{7l}_{av}, E^{8l}_{av}, \ldots, E^{\infty l}_{av}$ are naturally extended to the state $\ldots (n_i l_i)^{w_i} \varepsilon l$, where ε is the kinetic energy of the free electron with the angular momentum $\hbar l$ and the average energy of the configuration is $E^{\infty l}_{av} + \varepsilon$. Similarly, for each possible Rydberg series of states, ε is added to the limiting energy of the series.

While the angular part of wavefunction is identical with the bound states, the radial part of the continuum states has to be determined from the following equation

$$\left[-\frac{\mathrm{d}^2}{\mathrm{d}r^2} + \frac{l(l+1)}{r^2} + V^{\varepsilon l}(r) \right] P_{\varepsilon l}(r) = \varepsilon P_{\varepsilon l}(r) \ . \tag{1.21}$$

The limit of the approximate local-potential calculated by HX method is used for $V^{\varepsilon l}(r)$. The asymptote for $r \to \infty$ is $P_{\varepsilon l} \sim \cos(qr + \delta)$, where $q = \sqrt{\varepsilon}$. The radial part of wavefunction is normalized according to relation $\int_0^\infty P_{\varepsilon l}(r) P_{\varepsilon' l}(r) = \delta(\varepsilon - \varepsilon')$. Large mesh has to be used for numerical calculation of $P_{\varepsilon l}$.

Instead of the line strength S_{ij} for transition between bound states, the derivative $\mathrm{d}S_i/\mathrm{d}\varepsilon$ is calculated for the free-bound transitions. The photoionization cross-section is then

$$Q_{i\varepsilon} = 4\pi^2 \alpha \mathrm{d}S_i/\mathrm{d}\varepsilon \ . \tag{1.22}$$

11.2 Population of states and spectra synthesis

Knowledge of the populations of the ionization and excitation states is needed for the synthesis of emission and/or absorption spectra. Populations and spectra are usually calculated as a post-processor to fluid simulations for the estimation of the plasma parameters in experiment via the comparison with the measured spectra. .

Various selections of included states are used. **K-shell spectroscopy** (emission or backlighter) is used for relatively hard spectra of highly ionized atoms. It includes a detailed description of the states of H-, He- and Li-like ions while the other ionization stages are treated in a simplified way. **EUV spectroscopy** uses detailed model for all ionization stages but the atomic database will be enormous for high-Z atoms and the calculations may be computationally very demanding due to high number of populations solved. The number of the states in the equations for the populations can be decreased by grouping of states with near energies into one summary population. Populations of fine-split lower levels are often calculated separately, while higher states are usually grouped together as deviations from their mutual equilibrium are small due to fast collisional transitions between these states with very near energies. The number of bound states decreases with plasma density, as it is limited due to the **continuum lowering** caused by electron interactions with neighbouring ions. In a simplified image, if the electron orbit of a bound state of an isolated ion reaches far behind the neighboring ion, such bound state cannot exist. The populations of the states with energy above the lowered continuum may be set to zero or the statistical probability of their existence may be strongly reduced. The codes may also include selected autoionization states that lead to emission of satellite lines that are often used for the spectroscopic diagnostics.

When spectra are calculated, one way how to decrease the number of transitions is the statistical approach [9] using unresolved transition arrays (UTAs) where large number of near transitions is coupled together. UTAs form quasi-continuum features in the emission spectra and their parameters are the mean wavelength and the spectral bandwidth. Intense broad UTAs are often an efficient source of EUV radiation that may be suitable for applications.

11.2.1 Models of population kinetics

In dense plasmas, the collisional processes may be much faster than the radiative ones, and then the populations may be near to the **local thermodynamic equilibrium (LTE)**. In LTE, the thermodynamic equilibrium is assumed for massive particles, but not for radiation. Electrons have the Maxwellian distribution with the temperature T_e. Boltzmann relation $n_j/n_k = (g_j/g_k)\exp[(\varepsilon_k - \varepsilon_j)/k_B T_e]$ holds for the populations n of any two energy states j, k of the same ion charge, g denotes the degeneracy of the particular energy state. Ionization equilibrium is governed by the Saha equation which determines the ratio of the population of any level k of p-times ionized atom to the ground state 1 of $p+1$-times ionized atom depending on the electron density n_e and temperature

T_e, as follows

$$\frac{n_e n_{(p+1)1}}{n_{pk}} = 2\frac{g_{(p+1)1}}{g_{pk}} \left(\frac{2\pi m_e k_B T_e}{h^2}\right)^{3/2} \exp\left(-\frac{\varepsilon_{k\lambda}}{k_B T_e}\right) . \tag{1.23}$$

Spectrum of radiation in laboratory plasmas usually differs strongly from the blackbody radiation due to their small optical thicknesses. While the blackbody spectrum is a smooth continuous function, emission in laboratory plasmas is usually dominated by narrow spectral features - spectral lines and recombination edges.

On the other hand, in very dilute plasmas, the populations may be calculated according so called **coronal equilibrium**. Here the collisional de-excitation is negligible compared to the spontaneous emission, the three-body and dielectronic recombination is negligible compared to the radiative recombination and the omission of the above negligible processes simplifies calculation of the populations significantly.

When neither of the above approximations is valid, one has to use **collisional-radiative model** and the populations are usually called non-LTE (NLTE) populations. The system of the rate equations is written, as follows

$$\frac{\mathrm{d}n_i}{\mathrm{d}t} = \sum_{j=1(j\neq i)}^{m} R_{ji} n_j - \left(\sum_{j=1(j\neq i)}^{m} R_{ij}\right) n_i \quad \left(\text{in matrix form } \frac{\mathrm{d}\vec{n}}{\mathrm{d}t} = \mathbf{R}\vec{n}\right) \tag{1.24}$$

where $\vec{n}$ is the vector of populations and $\mathbf{R}$ is the rate matrix which is block diagonal as only transitions between neighbouring charge states have non-negligible probability. The rates usually include photoexcitation and photodexcitation, photoionization and photorecombination, collisional excitation and de-excitation, collisional ionization and three-body recombination and dielectronic recombination. The rates of the processes can be calculated by the atomic physics codes described above or via simplified analytic formulas. For certain transitions, experimental data are available and may be used either directly or for the correction of the data obtained from simulations. The rates of inverse processes are calculated using the principle of detailed balance. In equilibrium, the rates of direct and inverse processes are equal and the calculated rate coefficient of the inverse process is valid even outside the equilibrium. Such way of the rate calculation eliminates possible errors in the calculation of equilibrium populations.

When plasma is assumed optically thin, photoionization, photoexcitation and stimulated emission are omitted. In optically thin plasmas, the rate equations are independent of the radiation intensity. The rate equations in optically thin plasmas can be solved locally, the populations in each Lagrangian cell depend only on the history of density and electron temperature in the particular cell. The rates of collisional processes depend on electron density that in its turn depends on the ion mean charge calculated from the vector of populations. When the electron density n_e is fixed, then the rate equations are linear for optically thin plasmas. For given ion density, the system can be solved easily via iteration.

If optical thickness is taken into account, populations of charge and energy states at different places are interdependent through the transport of radiation. Detailed modelling of radiation transport is a very difficult and computationally demanding task as the narrow spectral features require a very fine spectral grid and additionally, ray-tracing in various directions must be performed. Simplified treatment via escape factor formalism [10] is often used. Escape factors generally depend on the shapes of spectral lines.

11.2.2 Synthesis of spectrum

When populations of charge and energy states are known, emission and/or backlighter spectra can be calculated. Backlighter spectrum is absorption spectrum produced by known source backlighting plasma. Line and recombination emission uses known populations and photoexcitation and photoionization rates. Continuous bremsstrahlung spectrum is then added. Line broadening including natural, pressure and Doppler broadening of spectral lines has to be taken into account. Pressure broadening in high-temperature plasmas is dominated by Stark broadening caused by electric microfields that include both the impact broadening caused by free electrons moving in the ion neighborhood and the quasi-static broadening caused by the neighbouring ions. Lines shapes may be calculated by specialized codes, such as STARCODE [11] and SimU [12]. This is a tedious task and thus, more often, simplified models of spectral broadening are directly included into the codes calculating populations and spectra.

11.2.3 Codes for population calculations and spectra synthesis

This subsection presents only a few examples of codes used without any effort for a complete overview of existing codes.

FLYCHK [13] is a rapid tool providing populations and spectra of plasmas in zero dimension assuming planar homogenous emitting (absorbing) plasma layer. Its advantage is an excellent availability as it can be used freely by anybody (after registration) via web page http://nlte.nist.gov/FLY/. It can be applied from low- to high-Z plasmas both for LTE and either steady-state or time-dependent NLTE situations. It can be used in all areas of EUV spectroscopy but it is especially suited for K-shell spectroscopy as it evolved from the older RATION and FLY codes developed R.W. Lee for this purpose. Plasmas with single or multiple electron temperatures as well as radiation driven plasmas can be modelled. Finite width of plasmas is taken into account via escape factors. Schematic atomic structures and scaled hydrogenic cross-sections are employed to achieve fast response and excellent versatility of the code. Continuum lowering (also called ionization potential depression) is included via the Stewart-Pyatt model [14]. The depression increases with the plasma density and is also temperature dependent. The states above the lowered ionization potential are considered to become continuum states and they are excluded from the calculation of populations. For the spectrum synthesis, the jj configuration averaged atomic states and oscillator strengths calculated via the Dirac-Hartree-Slater model are used. Voigt line profiles including natural and Doppler broadening are used for the calculation of the emissivity and opacity of line radiation and also for calculation of the escape factors. Instrumental broadening can be defined by the user. Stark broadening is included only for K-shell spectrum (H-, He- and Li-like ions) of elements $Z \leq 26$ which has to be taken into account in the spectra interpretation. FLYCHK is easy-to-use, simple, versatile and fast code providing sufficiently reasonable spectroscopy to most users to have design and analysis tool.

PrismSpect [15] and Spect3D [16] are zero- and three-dimensional commercial codes developed by the same group and based on the same models of atomic physics. Zero-dimensional code PrismSpect can treat planar, cylindrical and spherical geometry. SPECT3D is a multi-dimensional collisional-radiative code primarily meant for post-processing the output from radiation hydrodynamic and particle-in-cell codes to simulate diagnostic results (e.g. images, spectra). User can supply geometry and properties of the detecting system and thus the diagnostic results may be simulated for the design of experiment and/or for the interpretation of experimental data. In both codes, the atomic database is available for elements with $Z \leq 36$. It is calculated via ATBASE suite of codes [17]. An extensive configuration list for each isoelectronic sequence has been carefully setup and tested to support modelling for a wide range of plasma spectral properties. Complete collisional coupling between both all non-autoionizing and autoionizing states is included. The ionization potential depression is treated via model decreasing probability of existence of excited bound states with energies near to or above the lowered ionization potential. Line profiles are modelled using a Voigt profile, and include natural (including autoionization), Doppler, and Stark broadening. Photoabsorption is treated via escape factors in PrismSpect. Additionally, multi-angle radiative transfer models are available in SPECT3D. Two-temperature electron distribution is assumed, single ion temperature may be different. User can choose from 4 pre-configured atomic models: (1) Emission K-Shell Spectroscopy; (2) Emission Visible/UV/EUV Spectroscopy; (3) Back-lighter K-Shell Spectroscopy and (4) Low Temperature Spectroscopy. The energy level scheme is very detailed in Emission Visible/UV/EUV Spectroscopy model. The disadvantage of these codes is their difficult availability due to high price that has to be paid yearly.

Cretin [18] is a multi-dimensional NLTE radiation transfer code. Cretin is freely available upon request to interested researchers. Cretin treats radiation in three separate phases. Continuum and lines are handled separately, and are both coupled to the atomic kinetics and other physical processes. The third phase, spectral radiation, is actually a diagnostic for efficiently constructing detailed synthetic spectra. The spectral radiation accurately reflects the plasma conditions but does not couple back to the rest of the simulation. All three phases are available in the usual set of geometries (planar, cylindrical and spherical in 1D, Cartesian and cylindrical in 2D, Cartesian in 3D). Cretin includes an advanced treatment of radiation transport but it contains no atomic data. The level structure and transition processes for each element included in a simulation are specified in an external datafile. The database added to the code distribution is very simple aimed for testing the code and also serves as an example for database construction by users. While this allows a great flexibility in choosing a suitable atomic model, the construction of atomic model for spectroscopic diagnostics requires rather high qualification and extensive effort.

Modern plasma atomic physics, atomic kinetics and lineshape package ALICE is described in the recent paper [19].

11.3 Summary of the section

The first part of this section is devoted to a brief introduction into the structure of the electron cloud of an atom (ion) and the ways how electron orbitals, wave functions and energy levels are calculated. We have described atomic structure calculations via non-relativistic Schrödinger equation with relativistic corrections. While analytic solution can be found for atoms/ions with single bound electron (H atom and H-like ions), the effective potential has no simply expressible form and analytic solution for the wavefunction cannot be found for atoms (ions) with more than 1 bound electron. Electrons with the same principal and orbital numbers are called equivalent electrons and the list all pairs of those numbers defines configuration. Configuration-average energies and radial wavefunction can be determined via self-consistent field (SCF) iteration or via approximate local-potential methods. Energy level splitting inside one configuration is caused by the relativistic coupling of orbital and spin angular momentum. We have described two basic coupling schemes - **LS** and **jj** coupling. The former scheme is characteristic for low-Z elements where Coulomb repulsion dominates over spin-orbit interaction, while the latter is usual for high-Z elements where spin-orbit interaction is dominant. However, some configurations behave according to other coupling schemes and some (intermediate coupling) cannot be assigned to any general scheme. The accuracy of the atomic structure model may be improved by including more configurations with the same parity differing at most in 2 orbital from the given configuration.

When the wavefunctions are known, the probabilities of transitions may be calculated. The most intense radiative transitions are the electric dipole (E1) transitions. As the electric dipole operator has odd parity, E1 transitions can occur only between states with opposite parities. In **LS** coupling, selection rules allow only transitions with no change of the total spin and with the alteration of the total orbital quantum number by ± 1 or 0 and at least either the initial or the final total orbital quantum number is non-zero. However, these selection rules are often violated as **LS** coupling is usually not ideal in practice. Magnetic dipole (M1) and electric quadrupole (E2) transitions are typically weaker than E1 transitions and higher multipole transitions are observed rarely. Continuum states have to be included in calculations of photoionization and radiative recombination rates.

The second part of this section briefly describes the way how the populations of the ionization and excitation states are calculated and how emission and absorption spectra are synthetized. The spectrum of radiation is usually far from equilibrium Planckian distribution due to insufficient optical thicknesses of laboratory plasmas. However, collisional processes may be much faster than radiative ones in dense plasmas, and then the plasma may be near to LTE (local thermodynamic equilibrium), where thermodynamic equilibrium is assumed for massive particles, but not for radiation. Populations may be also calculated easily in the opposite case of dilute plasmas where in so called coronal equilibrium collisional deexcitation, three-body and dielectronic recombination can be omitted. When neither of above models is valid, non-LTE populations have to be calculated via collisional-radiative model solving rate equations. Various selections of included states are used. For K-shell spectroscopy, a detailed description of the states of H-, He- and Li-like ions is used while the other ionization stages are treated in a simplified way. EUV spectroscopy uses a detailed model for all ionization stages but the atomic database is enormous for high-Z atoms.

The rates of all elementary processes can be taken from calculations by the atomic structure codes or simplified analytic formulas may be used. The principle of detailed balance is used for calculation of the rates of the reverse processes. In optically thin plasmas, the rate equations can be solved locally, as the populations in each Lagrangian cell depend only on the history of density and electron temperature in the particular cell. If optical thickness is taken into account, populations of charge and energy states at different places are interdependent through the transport of radiation. Simplified treatment of the radiation transport via escape factors is often used. When populations of charge and energy states are known, emission and/or backlighter spectra can be calculated. Line broadening including natural, pressure and Doppler broadening of spectral lines has to be taken into account. Pressure broadening in high-temperature plasmas is typically dominated by Stark broadening. We have also presented a concise description of several computer codes intended for calculations of populations and for the spectrum synthesis.

11.4 References

[1] A. Hibbert. CIV3 - general program to calculate configuration interaction wave-functions and electric-dipole oscillator-strengths. *Comput. Phys. Commun.*, 9:141, 1975.

[2] W. Eissner, M. Jones, and H. Nussbaumer. Techniques for calculation of atomic structures and radiative data including relativistic corrections. *Comput. Phys. Commun.*, 8:270, 1974.

[3] C. Froese Fischer, T. Brage, and P. Jonsson. *Computational atomic structure: An MCHF approach*. Institute of Physics Publishing, 2000.

[4] R. D. Cowan. *The theory of atomic structure and spectra*. University of California Press, 1981.

[5] A. Bar-Shalom, M. Klapisch, and J. Oreg. H-ULLAC, an integrated computer package for atomic processes in plasmas. *J. Quant. Spectrosc. Rad. Transfer*, 71:169, 2001.

[6] D.H. Sampson, H.L. Zhang, A. K. Mohanty, and R.E.H. Clark. A Dirac-Fock-Slater approach to atomic-structure for highly charged ions. *Phys. Rev. A*, 40:604, 1989.

[7] M.F. Gu. The flexible atomic code. *Can. J. Phys.*, 86:675, 2008.

[8] G. Gaigalas, Z. Rudzikas, and C.F. Fischer. An efficient approach for spin-angular integrations in atomic structure calculations. *J. Phys. B*, 30:3747, 1997.

[9] J. Bauche, C. Bauche-Arnoult, and M. Klapisch. Transition arrays in the spectra of ionized atoms. *Adv. At. Mol. Phys.*, 27:131, 1987.

[10] F.E. Irons. Escape factor in plasma spectroscopy .1. escape factor defined and evaluated. *J. Quant. Spectrosc. Radiat. Transf.*, 22:1, 1979.

[11] S. Alexiou and R.W. Lee. Semiclassical calculations of line broadening in plasmas: Comparison with quantal results. *J. Quant. Spectrosc. Radiat. Transf.*, 99:10, 2006.

[12] E. Stambulchik and Y.A. Maron. A study of ion-dynamics and correlation effects for spectral line broadening in plasma: K-shell lines. *J. Quant. Spectrosc. Radiat. Transf.*, 99:730, 2006.

[13] H.-K. Chung, M.H. Chen, W.L. Morgan, Y. Ralchenko, and R.W. Lee. FLYCHK: an extension to the K-shell spectroscopy kinetics model FLY. *High Energy Density Phys.*, 1:3, 2005.

[14] J.C. Stewart and K.D. Pyatt. Lowering of ionization potentials in plasmas. *Astrophys. J.*, 144:1203, 1966.

[15] J.J. MacFarlane, I.E. Golovkin, P.R. Woodruff, D.R. Welch, B.V. Oliver, T.A. Melhorn, and R.B. Campbell. Simulation of the ionization dynamics of aluminum irradiated by intense short-pulse lasers. In B.A. Hammel, editor, *Inertial Fusion Sciences and Applications 2003*, page 453. American Nuclear Society, 2004.

[16] J.J. MacFarlane, I.E. Golovkin, P. Wang, P.R. Woodruff, and N.A. Pereyra. SPECT3D - A multi-dimensional collisional-radiative code for generating diagnostic signatures based on hydrodynamics and PIC simulation output. *High Energy Density Phys.*, 3:181, 2007.

[17] P. Wang. *Computation and application of atomic data for inertial confinement fusion plasmas.* PhD thesis, University of Wisconsin, Madison, WI, USA, 1991.

[18] H.A. Scott. Cretin - a radiative transfer capability for laboratory plasmas. *J. Quant. Spectrosc. Radiat. Transf.*, 71:689, 2001.

[19] E.G. Hill, G. Prez-Callejo, and S.J. Rose. ALICE: A non-LTE plasma atomic physics, kinetics and lineshape package. *High Energy Density Phys.*, 26:56, 2018.

O3 – Simulation laboratories

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

EXP 1: Laser matter interaction FEM simulations- Introduction and getting started using LS-DYNA/LS-PREPOST

E. Kaselouris, A. Baroutsos, V. Dimitriou

HELLENIC
MEDITERRANEAN
UNIVERSITY

AIM

The aim of the Finite Element Method - FEM, Lab session is to provide to the attendants the ability of understanding and implementing simulations of laser mater interaction problems.

1. LS-DYNA

LS-DYNA from Livermore Software Technology Corporation (LSTC), is a highly advanced general purpose nonlinear finite element program that is capable of simulating complex real-world problems. The distributed and shared memory solver provides very short turnaround times on desktop computers and clusters operated using Linux, Windows and UNIX.

LS-DYNA is suitable to investigate phenomena involving large deformations, sophisticated material models and complex contact conditions for structural dynamic problems. LS-DYNA allows switching between explicit and different implicit time stepping schemes. Disparate disciplines, such as coupled thermal analyses, Computational Fluid Dynamics (CFD), fluid-structure interaction, Smooth Particle Hydrodynamics (SPH), Element Free Galerkin (EFG), Corpuscular Method (CPM), Discrete Element Method (DEM) and the Boundary Element Method (BEM) can be combined with structural dynamics. For pre- and post-processing, LS-DYNA comes with the LS-PrePost tool. LS-PrePost can be utilized to generate inputs and visualize numerical results.

1.1 Keyword format input files

An LS-DYNA input file is a text-file in so called Keyword format usually with a *.k, *.key or *.dyn suffix, e.g. laser.k. A finite element model in LS-DYNA is built up by different keywords, which is defined for all ingoing definitions and parameters in a model (e.g. *PART, *NODE). A short overview of the basic structure of such an input file for a basic 1 element finite element model is provided.

Consider a cube consisting of one element with eight node points as shown in Figure 1.1.

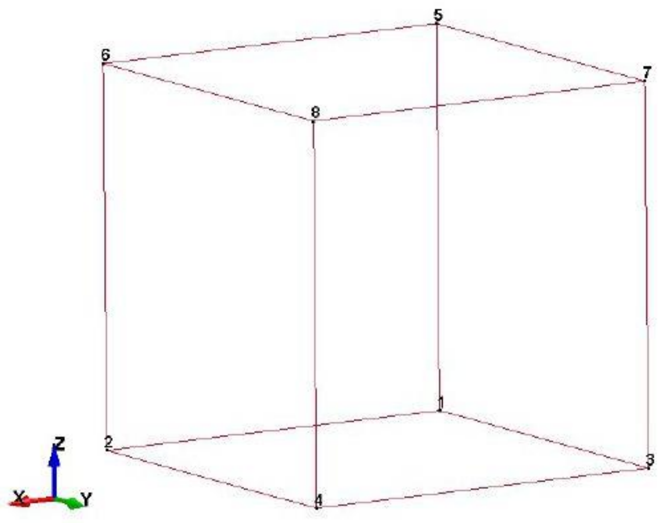
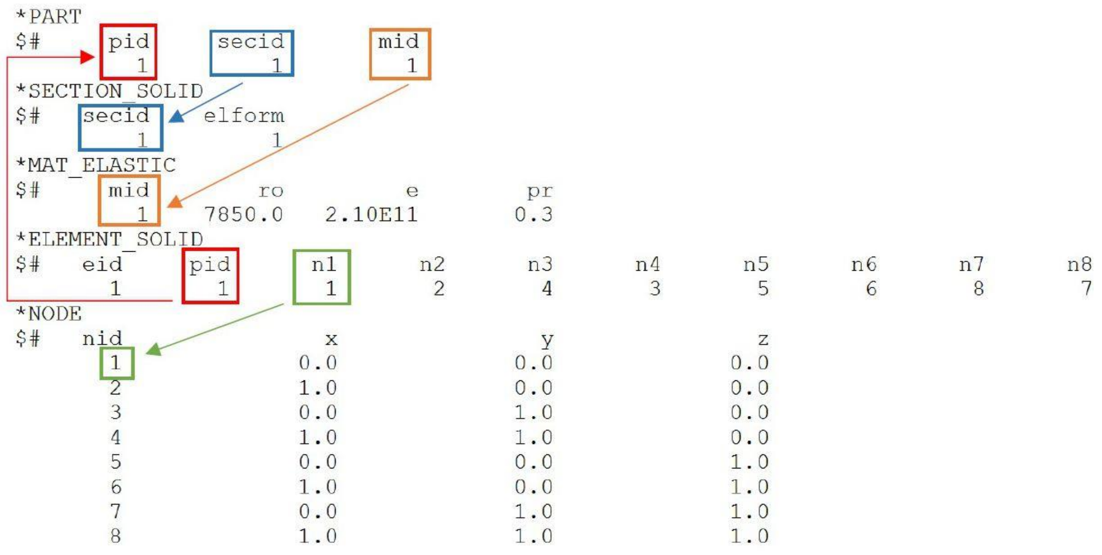


Figure 1.1 Cube consisting of one element with eight node points

The *PART keyword is used to begin the definition of the finite element model. The keyword *PART contains data that points to other attributes of this part, e.g. material properties. Keywords for these other attributes, in turn, point elsewhere to additional attribute definitions. The organization of the keyword input for the cube looks like this:



```
*PART
$#      pid       secid         mid
          1           1           1
*SECTION_SOLID
$#    secid      elform
          1           1
*MAT_ELASTIC
$#      mid          ro           e          pr
          1      7850.0     2.10E11         0.3
*ELEMENT_SOLID
$#    eid       pid      n1      n2      n3      n4      n5      n6      n7      n8
        1         1       1       2       4       3       5       6       8       7
*NODE
$#    nid               x                   y                   z
        1             0.0                 0.0                 0.0
        2             1.0                 0.0                 0.0
        3             0.0                 1.0                 0.0
        4             1.0                 1.0                 0.0
        5             0.0                 0.0                 1.0
        6             1.0                 0.0                 1.0
        7             0.0                 1.0                 1.0
        8             1.0                 1.0                 1.0
```

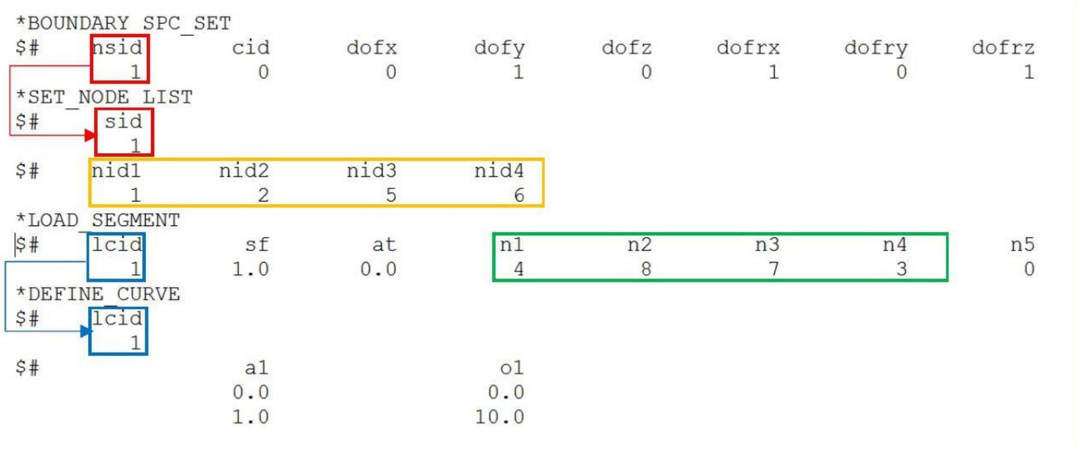
A brief description follows:

***PART**: We have one part with identification pid=1. This part has attributes identified by section identification secid=1 and material identification mid=1.

***SECTION_SOLID**: Parts definitions that reference secid=1 are defined as constant stress 8 node brick elements (elform=1).
***MAT_ELASTIC**: Parts definitions that reference mid=1 are defined as an elastic material with density, Young’s modulus and Poisson’s ratio.
***ELEMENT SOLID**: The element with identification eid=1 are defined by nid=1 to nid=8 and belongs to pid=1.
***NODE**: The node identified by nid has coordinates x,y,z.

Boundary conditions and time dependent loads are also set by keywords and are usually applied on nodes, elements, segments or parts. Set definitions are often used to define groups of these entities. Since all loads are time dependent, curves need to be defined that states time vs load unit (force, pressure etc.).

One example of defining boundary conditions and loads on the cube looks like this:



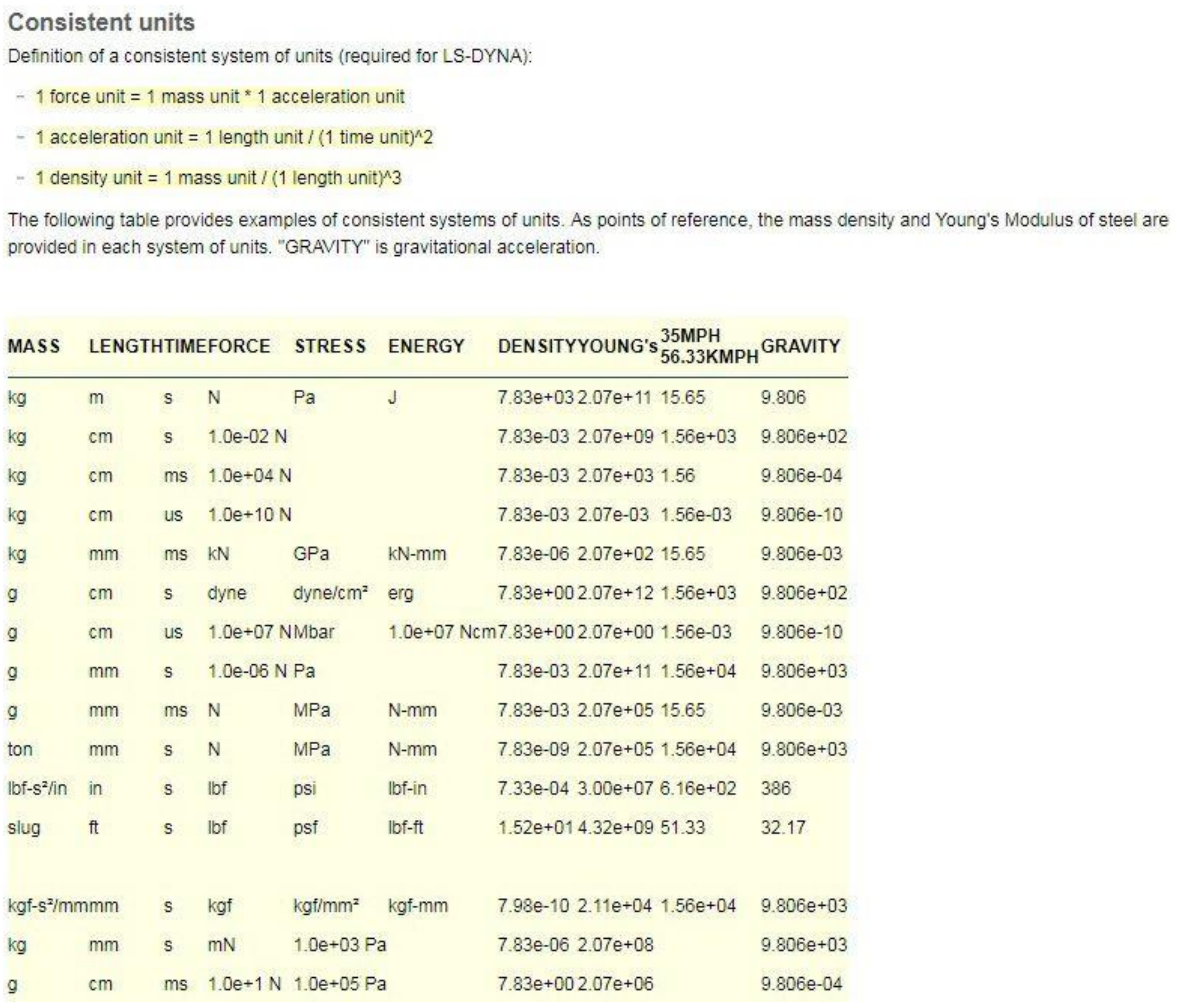
```
*BOUNDARY_SPC_SET
$#    nsid       cid      dofx      dofy      dofz     dofrx     dofry     dofrz
         1         0         0         1         0         1         0         1
*SET_NODE_LIST
$#     sid
         1
$#    nid1      nid2      nid3      nid4
         1         2         5         6
*LOAD_SEGMENT
$#    lcid        sf        at        n1        n2        n3        n4        n5
         1       1.0       0.0         4         8         7         3         0
*DEFINE_CURVE
$#    lcid
         1
$#                a1                  o1
                 0.0                 0.0
                 1.0                10.0
```

About the keywords above:
***BOUNDARY_SPC_SET**: The node set with identification nsid=1 are constrained in y-translation and x- and z-rotations.
***SET_NODE_LIST**: This keyword defines that node 1, 2, 5 and 6 belongs to node set sid=1.

***LOAD_SEGMENT**: A pressure load is applied on a segment that are defined by node 4, 8, 7 and 3.
***DEFINE_CURVE**: The curve consists of two points that defines the time vs pressure. This curve with identification lcid=1 is used for the load.

1.2 Consistent units

On the table below are presented consistent units for using LS-DYNA. The inappropriate selection of consistent units is one of the most common error done by the user.



Consistent units

Definition of a consistent system of units (required for LS-DYNA):

- 1 force unit = 1 mass unit * 1 acceleration unit
- 1 acceleration unit = 1 length unit / (1 time unit)^2
- 1 density unit = 1 mass unit / (1 length unit)^3

The following table provides examples of consistent systems of units. As points of reference, the mass density and Young's Modulus of steel are provided in each system of units. "GRAVITY" is gravitational acceleration.

MASS	LENGTH	TIME	FORCE	STRESS	ENERGY	DENSITY	YOUNG's	35MPH 56.33KMPH	GRAVITY
kg	m	s	N	Pa	J	7.83e+03	2.07e+11	15.65	9.806
kg	cm	s	1.0e-02 N			7.83e-03	2.07e+09	1.56e+03	9.806e+02
kg	cm	ms	1.0e+04 N			7.83e-03	2.07e+03	1.56	9.806e-04
kg	cm	us	1.0e+10 N			7.83e-03	2.07e-03	1.56e-03	9.806e-10
kg	mm	ms	kN	GPa	kN-mm	7.83e-06	2.07e+02	15.65	9.806e-03
g	cm	s	dyne	dyne/cm²	erg	7.83e+00	2.07e+12	1.56e+03	9.806e+02
g	cm	us	1.0e+07 N	Mbar	1.0e+07 Ncm	7.83e+00	2.07e+00	1.56e-03	9.806e-10
g	mm	s	1.0e-06 N	Pa		7.83e-03	2.07e+11	1.56e+04	9.806e+03
g	mm	ms	N	MPa	N-mm	7.83e-03	2.07e+05	15.65	9.806e-03
ton	mm	s	N	MPa	N-mm	7.83e-09	2.07e+05	1.56e+04	9.806e+03
lbf-s²/in	in	s	lbf	psi	lbf-in	7.33e-04	3.00e+07	6.16e+02	386
slug	ft	s	lbf	psf	lbf-ft	1.52e+01	4.32e+09	51.33	32.17
kgf-s²/mm	mm	s	kgf	kgf/mm²	kgf-mm	7.98e-10	2.11e+04	1.56e+04	9.806e+03
kg	mm	s	mN	1.0e+03 Pa		7.83e-06	2.07e+08		9.806e+03
g	cm	ms	1.0e+1 N	1.0e+05 Pa		7.83e+00	2.07e+06		9.806e-04

Table 1.1 Consistent units using LS-DYNA

2. LS-PrePost

LS-PrePost is an advanced pre- and post-processor designed specifically for LS-DYNA. It is developed for Windows, Linux and Apple and it is free to download from the web link http://ftp.lstc.com/anonymous/outgoing/lsprepost/4.5/.

LS-Prepost main functions contain:

- Full support of LS-DYNA keyword files
- Full support of LS-DYNA result files
- Robust handling of geometry data (new CAD engine)
- Pre-processing (meshing, model clean-up, entity creation)

- Post-processing (animation, fringe plotting, curve plotting)

2.1 Input and output files

Input

FEM: LS-DYNA Keyword, Nastran, I-DEAS Universal, PAM-CRASH, RADIOSS, ABAQUS

CAD: IGES, STEP

ASCII: glstat, matsum, etc.

Binary: d3plot, binout, etc.

Output

FEM: LS-DYNA Keyword, Nastran

Image: PNG, TIFF, BMP, GIF, JPG, PostScript

Movie: AVI, MPEG, Animated GIF, JPEG

XY Data: CRV, CSV, XML o CAD: IGES, STEP, STL

Other: Post.db, Project File

2.2 Mouse and Keyboard

Dynamic Model Operation

Rotate: Shift + Left-click

Translate: Shift + Middle-click

Zoom: Shift + Right-click/Scroll-wheel

Graphics Selection

Pick (single): Left Click o Area (rectangle): Left-click + Drag

Poly (polygon): Left-click at corners / Right-click to finish

List Selection

Multi-Select: Left-click + Drag / Ctrl + Left-click

2.3 Graphical user interface

The graphical interface of LS-PrePost is shown in Figure 2.1. On the right hand side, you can see the main toolbar. When clicking on one of these, a sub-toolbar just to the left will be shown. That is the location where you'll find most of the tools needed to create/modify/delete entities in your model.

In the bottom toolbar, are found the tools which are the most commonly used to determine how LS-PrePost should render mesh/surfaces, orient the model, etc. There are a couple of the drop-down menus on the top left corner that you will use more or less frequently: File/View/Application/Settings. The Floating Toolbar is used to toggle between different views.

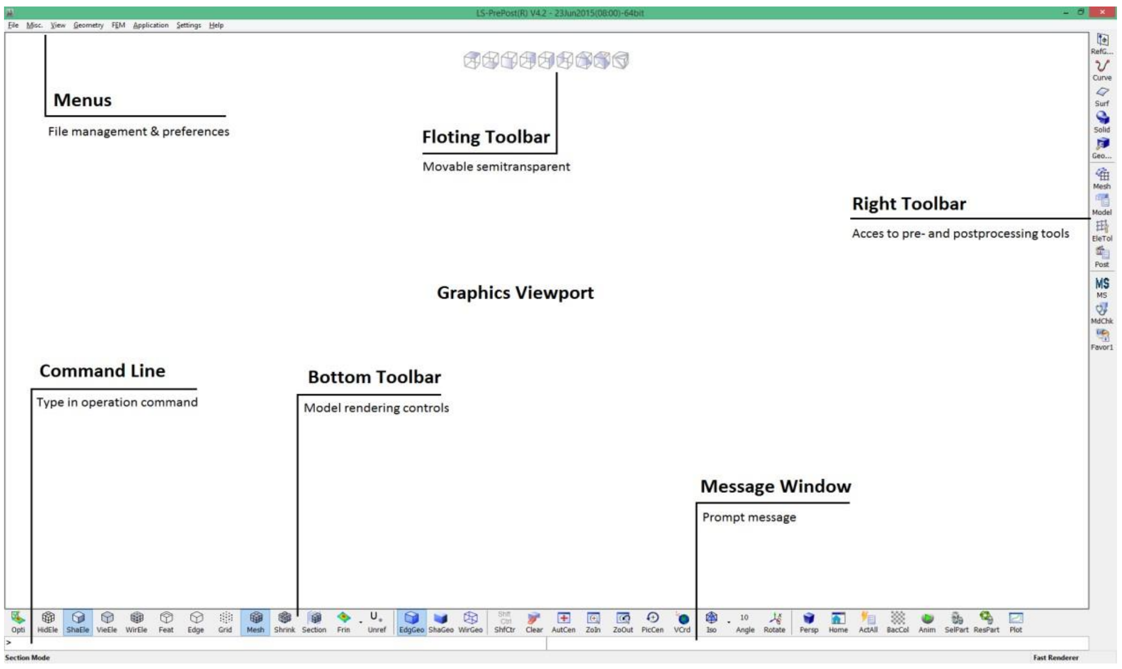
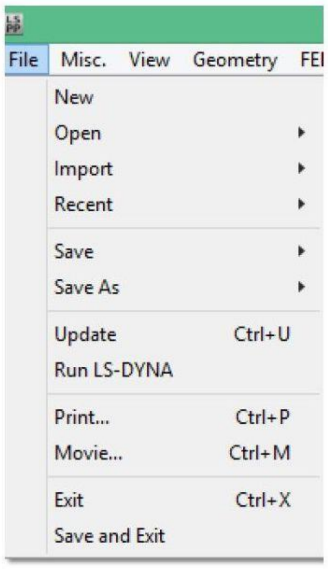

LS-PrePost(R) V4.2 - 23Jun2015(08:00)-64bit
File Misc. View Geometry FEM Application Settings Help
Menus
File management & preferences
Floting Toolbar
Movable semitransparent
Right Toolbar
Acces to pre- and postprocessing tools
Graphics Viewport
Command Line
Type in operation command
Bottom Toolbar
Model rendering controls
Message Window
Prompt message
Section Mode
Fast Renderer

Figure 2.1 Graphical interface of the LS-PREPOST

2.4 Menus

In this section are presented the useful menus for the user, the File Menu, the Geometry Menu and the FEM menu.

File Menu



New – Launch a new session of LS-PrePost, all model/data will be closed

Open – Open file (new model created for each file opened)

Import – Import file (adds keyword data to current model)

Recent – Open recent files (stored in /user/.lspp_recent)

Save – Over-write current *Keyword* or *Project* file

Save As – Save any of the following file formats using advanced options: *Keyword, Active Keyword* (visible data), *Project*, *Post.db* (condensed d3plot data), *Geometry, Keyword and Project* (using the same file name).

Update – Load new d3plots for run in progress

Run LS-DYNA – Pop up LS-DYNA job submission dialog, currently only limited to the same local machine LS-PrePost is running

Print... – Launch printing interface (send to printer or image file)

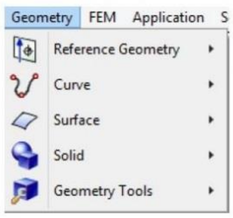
Movie... – Launch movie generation interface

Exit – Exit LS-PrePost

Save and Exit – Save data to current file and exit LS-PrePost

Figure 2.2 File menu

Geometry Menu



Reference Geometry – Access tools for creating and editing reference geometry (Axis, Plane, Coordinate System, Point, Reference Geometry Edit)

Curve – Access tools for creating and editing curves (Point, Line, Circle, Circular Arc, Ellipse, Elliptical Arc, BSpline Curve, Helix, Composite Curve, Break Curve, Merge Curve, Bridge Edge, Smooth Curve, Middle Curve, Morphing Curve, Fillet Curve, Parabola, Hyperbola, Function, Polygon, Convert, Sketch)

Surface – Access tools for creating and editing surfaces (Plane, Cylinder, Cone, Sphere, Torus, Ellipsoid, Fill Plane, Extrude, Revolve, Sweep, Loft, N-Side Surface, Patch Surface, Bridge Two Faces, Combine Faces, Fit From Points/Mesh, Middle Surface, Surface Morphing, Fit Primary Surface, Break Surface)

Solid – Access tools for creating and editing solids (Box, Cylinder, Cone, Sphere, Torus, Extrude, Revolve, Sweep, Loft, Fillet, Chamfer, Draft, Thicken, Wedge, Boolean, Prism)

Geometry Tools – Access other geometry tools (Delete Face, Blank Entity, Extend Curve, Extend Face, Intersection, Offset, Project, Replace Face, Stitch Faces, Trim, Transform, Reverse Direction, Copy Entity, Management, Heal, Topology Simplify, Measure, Text Object, Array flow)

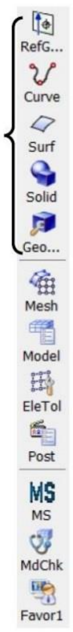
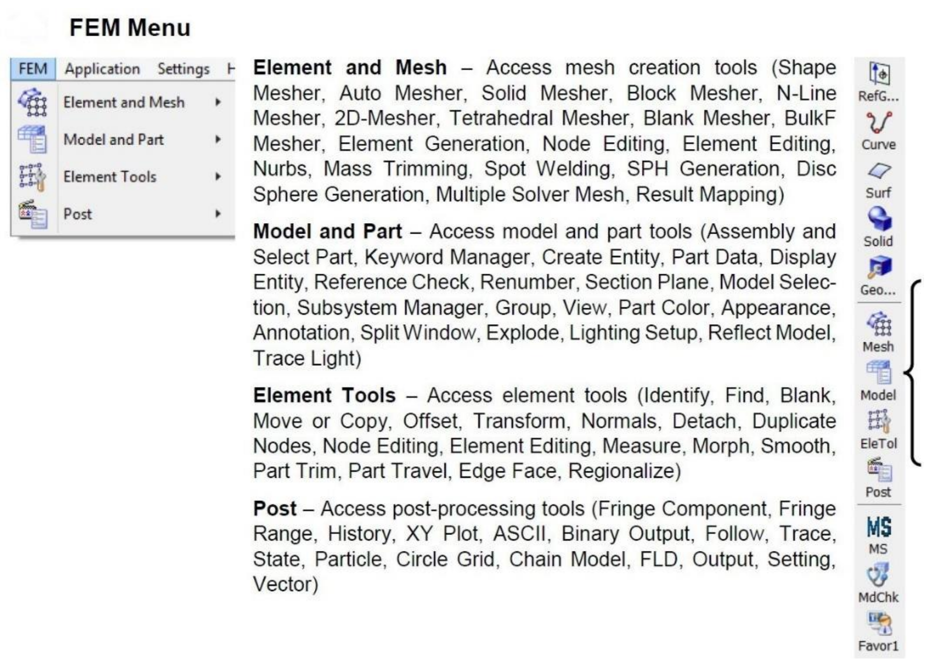


Figure 2.3 Geometry menu



Element and Mesh – Access mesh creation tools (Shape Mesher, Auto Mesher, Solid Mesher, Block Mesher, N-Line Mesher, 2D-Mesher, Tetrahedral Mesher, Blank Mesher, BulkF Mesher, Element Generation, Node Editing, Element Editing, Nurbs, Mass Trimming, Spot Welding, SPH Generation, Disc Sphere Generation, Multiple Solver Mesh, Result Mapping)

Model and Part – Access model and part tools (Assembly and Select Part, Keyword Manager, Create Entity, Part Data, Display Entity, Reference Check, Renumber, Section Plane, Model Selection, Subsystem Manager, Group, View, Part Color, Appearance, Annotation, Split Window, Explode, Lighting Setup, Reflect Model, Trace Light)

Element Tools – Access element tools (Identify, Find, Blank, Move or Copy, Offset, Transform, Normals, Detach, Duplicate Nodes, Node Editing, Element Editing, Measure, Morph, Smooth, Part Trim, Part Travel, Edge Face, Regionalize)

Post – Access post-processing tools (Fringe Component, Fringe Range, History, XY Plot, ASCII, Binary Output, Follow, Trace, State, Particle, Circle Grid, Chain Model, FLD, Output, Setting, Vector)

Figure 2.4 FEM Menu

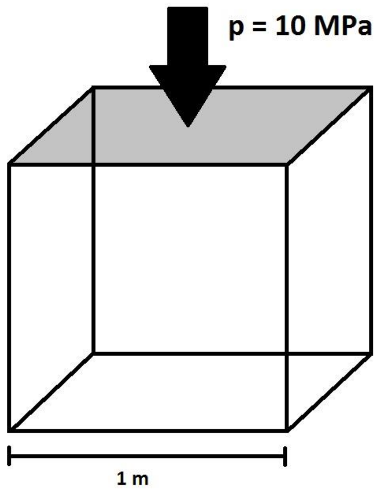
3. Getting started

3.1 Purpose

The purpose of this tutorial is to get familiar with the pre- and post-processing tools in LS Pre-Post and also the basics of the LS-DYNA solver.

3.2 Problem Description

Consider the deformation of a cube on the ground with an applied pressure on the top surface. The task is to compute the vertical displacement of the cube due to this pressure.

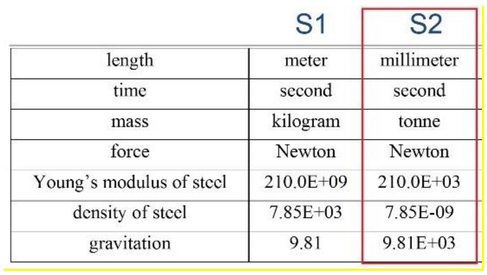


Material properties	
Density, ρ	7850 kg/m^3
Young’s modulus, E	210 GPa
Poisson’s Ratio, ν	0.3 0.3

3.3 Data files and unit system

The code can be found in **cube_results.k**

The S2 unit system is used in this tutorial



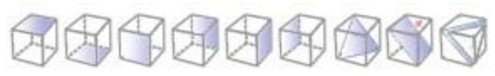
	S1	S2
length	meter	millimeter
time	second	second
mass	kilogram	tonne
force	Newton	Newton
Young's modulus of steel	210.0E+09	210.0E+03
density of steel	7.85E+03	7.85E-09
gravitation	9.81	9.81E+03

3.4 Preparation LS-PrePost

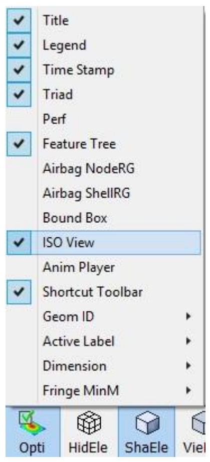
The most common way to work with/open LS-PrePost is to have a short-cut on the desktop directly. This gives you control over which version of LS-PrePost you would like to use and you can easily update LS-PrePost separately.

3.4.1 View settings

In LS-PrePost, go to **View > Toolbar** and activate **Text and Icon (Right)** and **Text and Icon (Bottom).** This is done to easier navigate through the different toolbars.



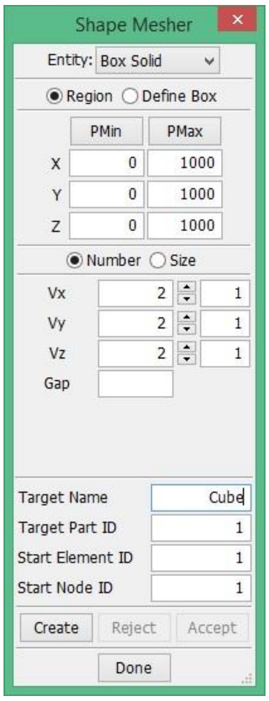
Check so you can see your Floating Toolbar in the LS-PrePost window.



If not, activate it by clicking on **Opti > ISO View** in the bottom toolbar.

3.5 Create model

3.5.1 Geometry and mesh


Shape Mesher

Entity: Box Solid

Region [✓] Define Box

	PMin	PMax
X	0	1000
Y	0	1000
Z	0	1000

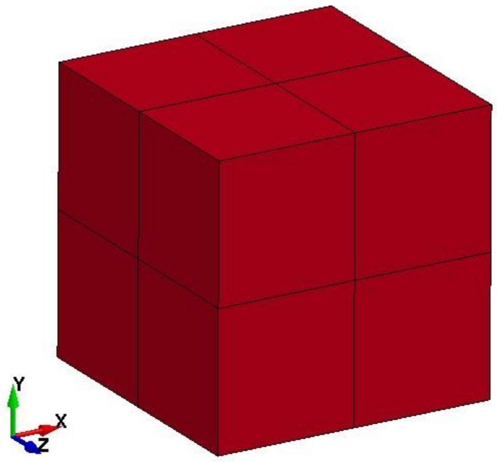
Number [✓] Size

Vx	2	1
Vy	2	1
Vz	2	1
Gap		

Target Name	Cube
Target Part ID	1
Start Element ID	1
Start Node ID	1

- Click **Mesh > ShapeM** (always use the menu on the right side, if nothing else is mentioned).
- Enter the values as in the picture to create a 1000x1000x1000 mm solid cube with two elements in the x-, y- and z-directions.
- Set **Target Name** to **Cube**.
- Click **Create, Accept** and then **Done.**

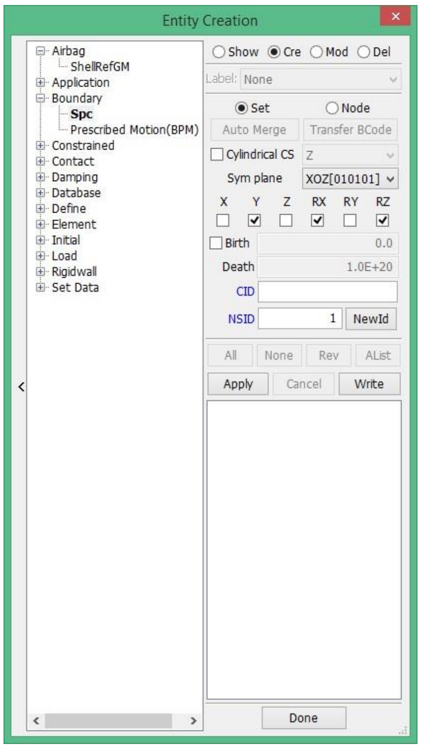
If you can't see your mesh, activate **Mesh** in the bottom toolbar.



3.5.2 Boundary conditions

Apply boundary conditions to fix one side of the cube:

- Click **Model > CreEnt**
- In the Entity Creation box, double-click on **Boundary** and click **Spc** in the dropdown that appears.
- Select **Cre**
- **Set** shall be activated.
- Select **XOZ** as **Sym Plane**, Y, RX and RZ will then be activated. The boundary conditions will then have a translational constraint in global y –direction and rotational constraints about x- and z-axis.
- From the ISO-views on the top of the screen, click at the one called **Top**.



Entity Creation

Airbag, ShellRefGM, Application, Boundary, Spc, Prescribed Motion(BPM), Constrained, Contact, Damping, Database, Define, Element, Initial, Load, Rigidwall, Set Data

Show [✓ Cre] Mod Del

Label: None

[✓ Set] Node

Auto Merge | Transfer BCode

Cylindrical CS: Z

Sym plane: XOZ[010101]

X, [✓ Y], Z, [✓ RX], RY, [✓ RZ]

Birth: 0.0

Death: 1.0E+20

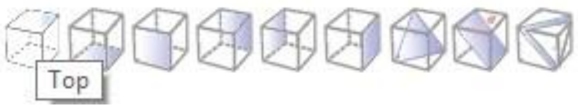
CID

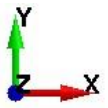
NSID: 1 | NewId

All | None | Rev | AList

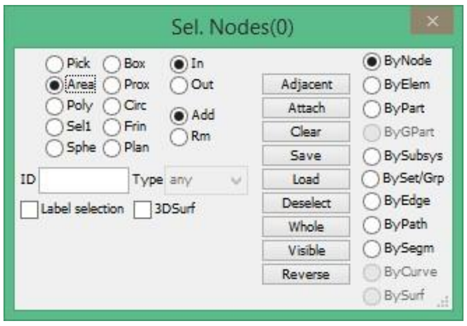
Apply | Cancel | Write

Done





Check that your coordinate system looks like the one above. To show the coordinate system, click **Opti** and activate **Triad**.



Sel. Nodes(0)

Pick, Box, [✓ Area], Prox, Poly, Circ, Sel1, Frin, Sphe, Plan

[✓ In] Out

[✓ Add] Rm

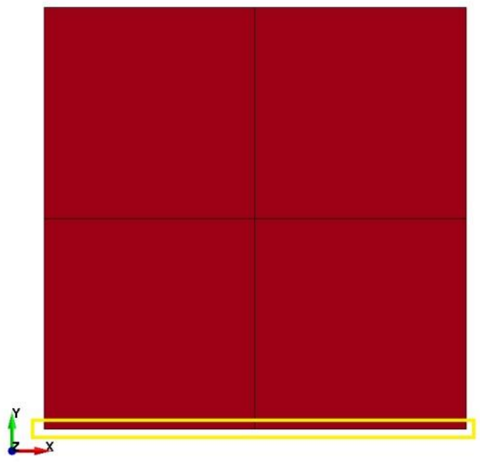
Adjacent, Attach, Clear, Save, Load, Deselect, Whole, Visible, Reverse

[✓ ByNode] ByElem, ByPart, ByGPart, BySubsys, BySet/Grp, ByEdge, ByPath, BySegm, ByCurve, BySurf

ID | Type: any

Label selection | 3DSurf

This box shows alternatives to select the nodes. Select **Area**.



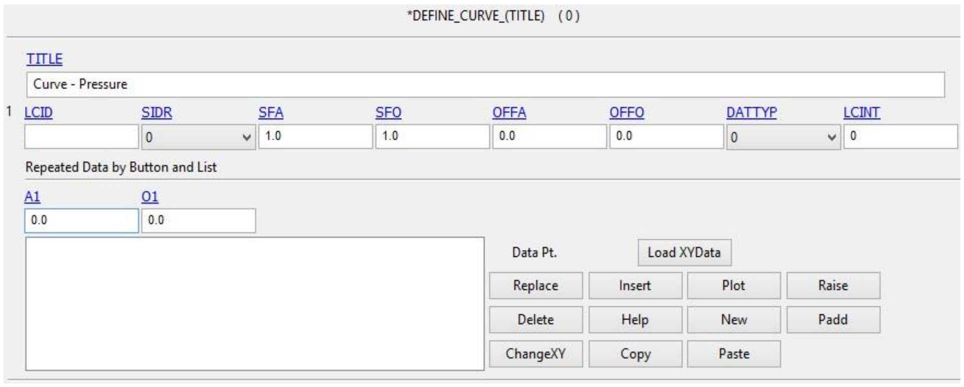
- Select the nodes in the yellow square by making a box with the mouse.
- A node set will be created from the nodes that were chosen, **NSID = 1** in **Entity Creation** indicates that it will get an Id = 1.
- Click **Apply,** then **Done** in the Entity Creation box.

The nodes are now constrained.

3.5.3 Apply the load

For loads, a curve must be defined that states the variation of the load over time. Click **Model > Keywrd**. In the dialog window that opens, there one can change between **Model** and **All. Model** shows the keywords that already have been created. **All** shows all possible keywords that are available in LS-DYNA.

Select **All** at the top of Keyword Manager window. Double-click **DEFINE > CURVE.** Name the curve **Curve – Pressure** for example. All titles are optional, but it is good practice to make use of them to make the model clear and structured. The points for the curve will be written in **A1** and **O1:**


*DEFINE_CURVE_(TITLE) (0)

TITLE: Curve - Pressure

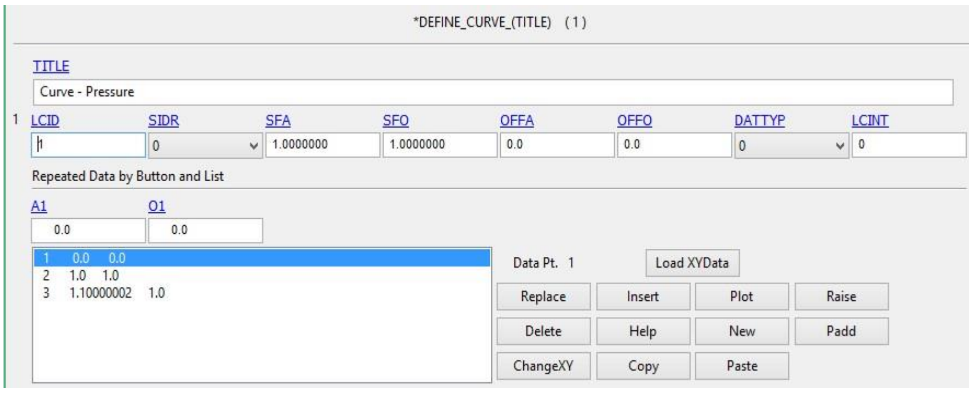
	LCID	SIDR	SFA	SFO	OFFA	OFFO	DATTYP	LCINT
1		0	1.0	1.0	0.0	0.0	0	0

Repeated Data by Button and List

A1	O1
0.0	0.0

- Write **0** and **0,** Click **Insert.**
- Then **1** and **1, Insert.**
- Finally, **1.1** and **1, Insert.**
- Click **Accept**

It is important that the curve extends beyond the end time of the simulation. The simulation will have the termination time 1 s (will be set later). Therefore, the last point of 1.1 was added.


*DEFINE_CURVE_(TITLE) (1)

TITLE: Curve - Pressure

	LCID	SIDR	SFA	SFO	OFFA	OFFO	DATTYP	LCINT
1	1	0	1.0000000	1.0000000	0.0	0.0	0	0

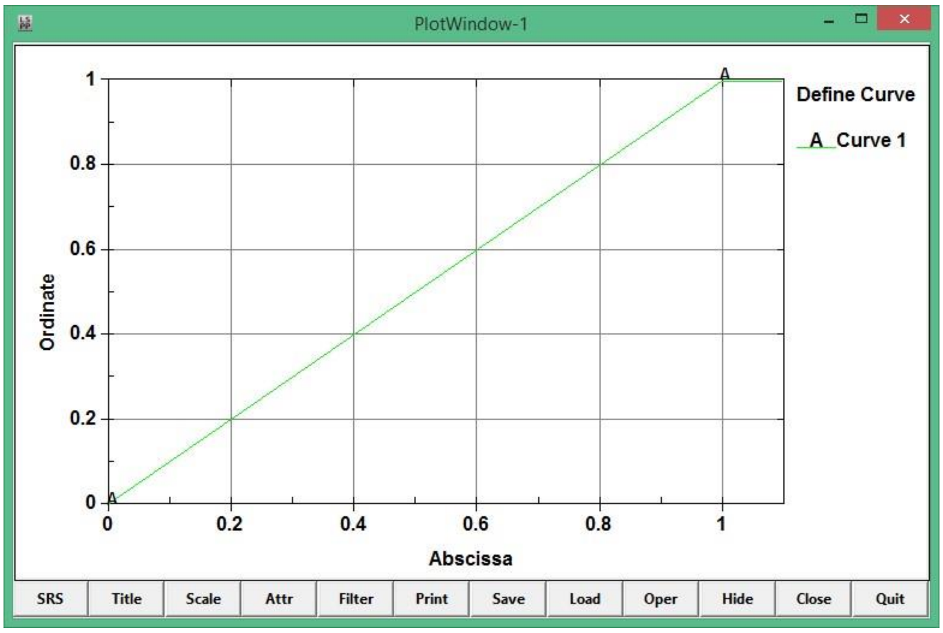
Repeated Data by Button and List

A1	O1
0.0	0.0

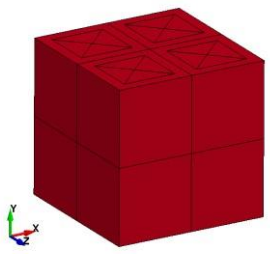
```
1   0.0   0.0
2   1.0   1.0
3   1.10000002   1.0
```

Data Pt. 1

To view the curve, Click **Plot.** Close the **PlotWindow (X or Quit)** and the ***DEFINE_CURVE (Done)** window.



Now click **Model > CreEnt**:



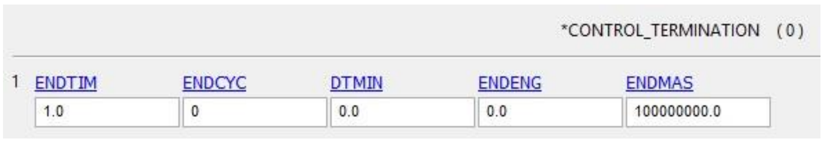
- In the Entity Creation box, double-click on **Load** and click on **Segment** in the dropdown that appears.
- Click **Cre**
- Change **Type:** to **LOAD_SEGMENT_SET**.
- Give the load the title **Pressure**.
- Click on **LCID** and select **1 Curve – Pressure**, press **Done**.
- To obtain a pressure of 10 MPa, the scale factor **SF** will be used.
- Set **SF** to **10** (the pressure unit is MPa for the selected unit system).
- From the selection box, **Pick** can be activated.
- Click on the **four** segments on the top of the cube, as in the figure. If necessary, deactivate entities with right mouse button.
- Click **Apply**, then **Done**

3.5.4. Termination

The end time for the simulation needs to be set. This keyword is almost always mandatory for any simulation using LS-DYNA:

- Click **Model > Keywrd**.

- Double-click **CONTROL > TERMINATION**
- Set **ENDTIM** to **1**. The simulation will then last for 1 time unit, which is second in this case.
- **Accept**, then **Done**.



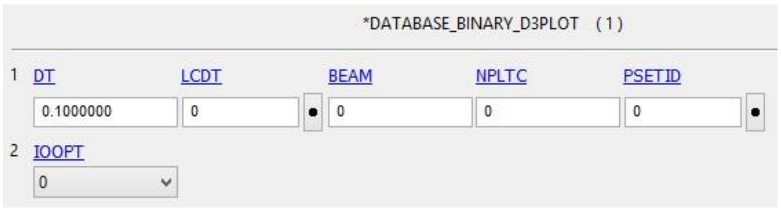
*CONTROL_TERMINATION (0)

ENDTIM	ENDCYC	DTMIN	ENDENG	ENDMAS
1.0	0	0.0	0.0	100000000.0

3.5.5 Output

The user must request all the data needed to post-process an analysis using LS-DYNA, before starting the simulation. We will create something called d3plot, which gives complete output states of the simulation:

- Click **Model > Keywrd**.
- Double-click **DATABASE > BINARY_D3PLOT**
- Set **DT** to **0.1**. This implies that results will be printed every 0.1 time unit.
- **Accept**, then **Done**



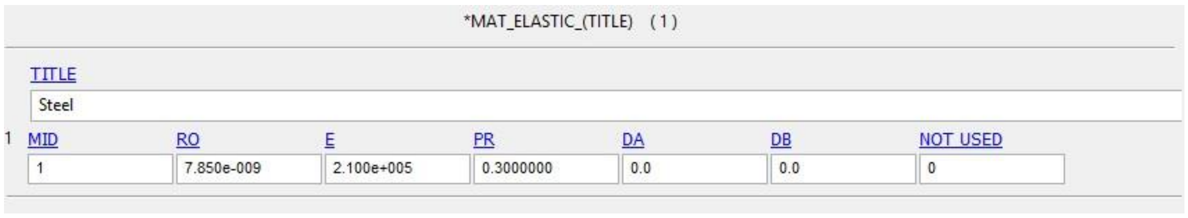
*DATABASE_BINARY_D3PLOT (1)

DT	LCDT	BEAM	NPLTC	PSETID
0.1000000	0	0	0	0

IOOPT
0

3.5.6 Material properties

- To create a material card to define the material properties:
- Click **Model > Keywrd**
- Double-click **MAT > 001-ELASTIC**. This is an isotropic elastic materia
- Name the material to **Steel.**
- Set the material properties **RO**, **E** and **PR** as in the figure below (also stated in section 1.3).
- Click **Accept**, then **Done**.



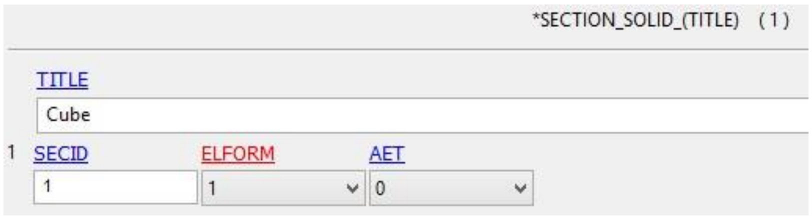
*MAT_ELASTIC_(TITLE) (1)

TITLE						
Steel						
MID	RO	E	PR	DA	DB	NOT USED
1	7.850e-009	2.100e+005	0.3000000	0.0	0.0	0

3.5.7 Element properties

The element type to be used:

- From **Keyword Manager**, double-click **SECTION > SOLID**.
- Name the section to **Cube**
- Use **ELFORM = 1**, which is the default element formulation.
- Click **Accept**, then **Done**.



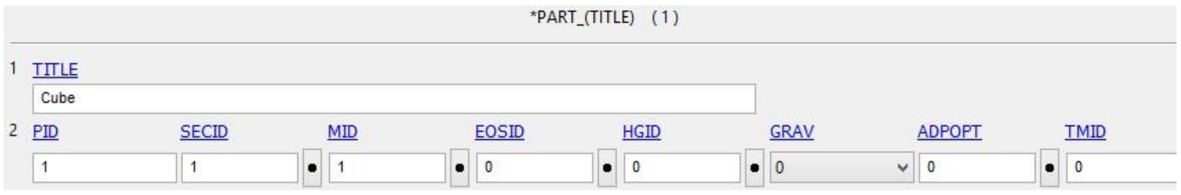
*SECTION_SOLID_(TITLE) (1)

TITLE		
Cube		
SECID	ELFORM	AET
1	1	0

Now apply the material and element properties to the part. Since the part already is created, one can activate **Model**, instead of **All**, in the **Keyword Manager**. This makes it easier to navigate through the list of keywords.

- Double-click **PART > PART**
- Click on the black dot next to SECID, defined entities will then be shown Select your newly created section (**1 Cube**) and **Accept** and then **Done**
- Do the same thing for **MID**. Click **Accept**, then **Done**.

The result should be as shown below.



*PART_(TITLE) (1)

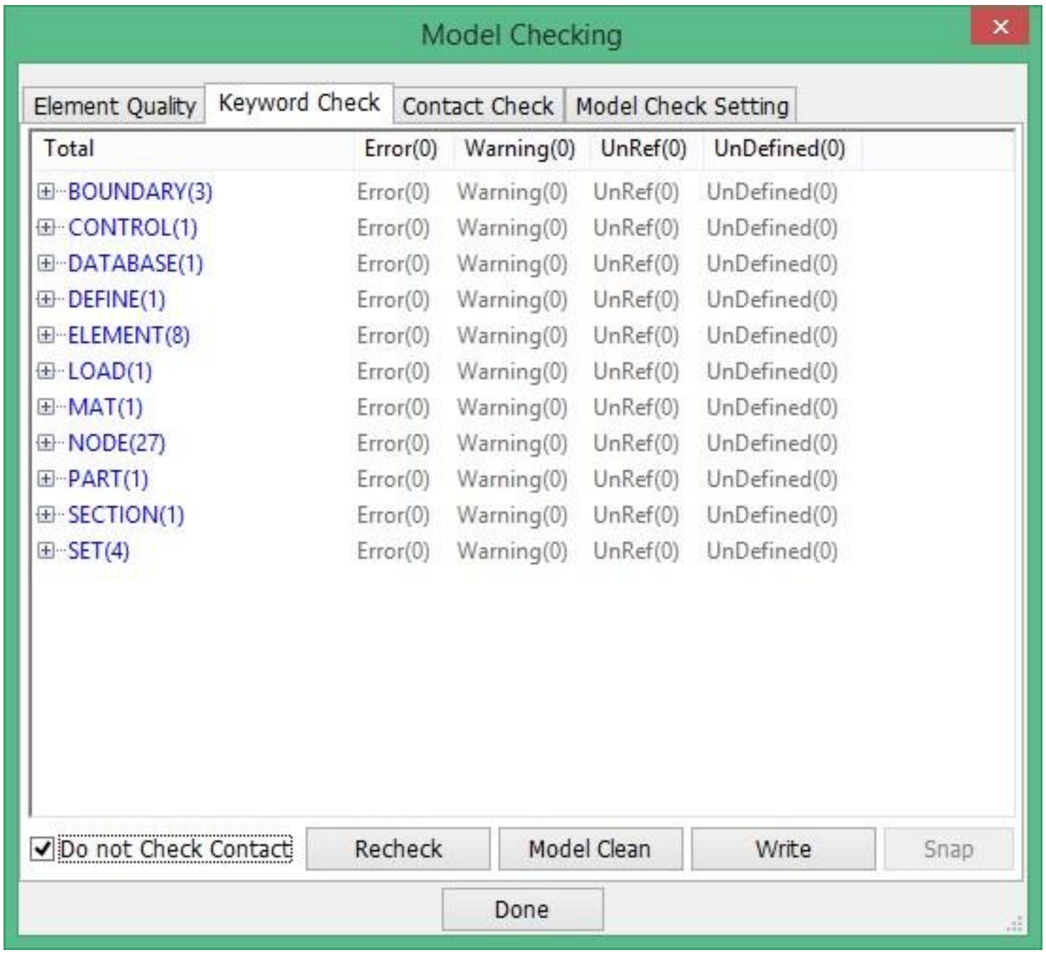
TITLE							
Cube							
PID	SECID	MID	EOSID	HGID	GRAV	ADPOPT	TMID
1	1	1	0	0	0	0	0

3.5.8 Check the model before running

Now check for errors using the **Model Check**:

- From the top menu, click **Application > Model Checking > General Checking**.
- Switch to **Keyword Check**. The warnings and error should not exist.
- Click **Done**.

Note that even if no errors or warnings occurs, the model can still be incomplete or wrong. There is no way for any pre-processor to know your intended use of the model. Hence, the loadings and boundary conditions can only be checked if they make any sense, not if they are correct with respect to your load case.


Model Checking

Element Quality | Keyword Check | Contact Check | Model Check Setting

Total	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)
⊞ BOUNDARY(3)	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)
⊞ CONTROL(1)	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)
⊞ DATABASE(1)	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)
⊞ DEFINE(1)	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)
⊞ ELEMENT(8)	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)
⊞ LOAD(1)	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)
⊞ MAT(1)	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)
⊞ NODE(27)	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)
⊞ PART(1)	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)
⊞ SECTION(1)	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)
⊞ SET(4)	Error(0)	Warning(0)	UnRef(0)	UnDefined(0)

☑ Do not Check Contact

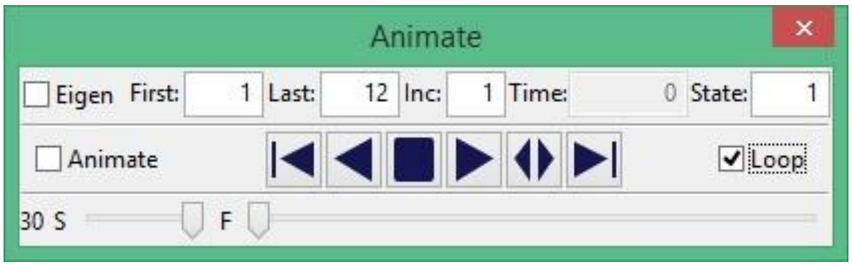
3.5.9 Save and run the simulation

It is preferable to run each simulation in a separate folder, thus create one before saving if you have not done so, e.g. CUBE. First save the finished keyword model from LS-PrePost in the new “CUBE” folder that you have created on your computer using **File > Save As > Save Keyword As.** Use the file name **cube.k,** note the **.k** suffix.

To how to run the simulation, you will be given orders from the instructor.

3.5.10 Post processing

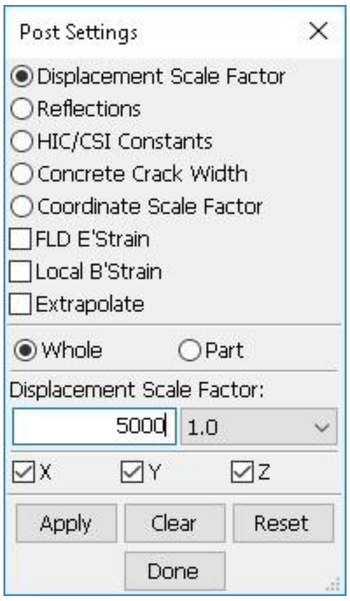
To visualize the results, you need to open the **d3plot** result file. This is done by selecting File>Open>LS DYNA Binary Plot. Once Binart Plot is opened, the animate toolbar is the tool that enables you to step through the different states of the simulation. Hold the mouse over the different buttons, a text box will pop up and show information about the different possibilities using the animator toolbar. Play around with the buttons to see what happens. Note that the deformations are very small, therefore you will probably not notice that anything happens with the cube.


Animate

Eigen ☐ First: 1 Last: 12 Inc: 1 Time: 0 State: 1

☐ Animate ☑ Loop

30 S F


Post Settings

- (●) Displacement Scale Factor
- ( ) Reflections
- ( ) HIC/CSI Constants
- ( ) Concrete Crack Width
- ( ) Coordinate Scale Factor
- ☐ FLD E'Strain
- ☐ Local B'Strain
- ☐ Extrapolate

(●) Whole ( ) Part

Displacement Scale Factor: 5000 1.0

☑ X ☑ Y ☑ Z

Apply Clear Reset Done

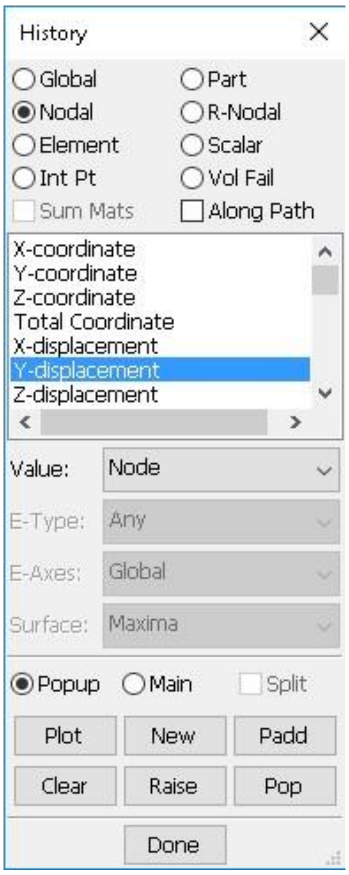
To easier see what happens, you can scale up the deformations:

• Click **Settings > Post Settings** in the top menu.

• Select **Displacement Scale Factor,** write **5000** as the factor. X, Y and Z shall be activated, which implies that the displacements will be scaled in all directions.

• Click **Apply**, then **Done**.

Play around with the animate toolbar again and see how the cube deforms.



History

Global [✓ Nodal] Element Int Pt Sum Mats
Part R-Nodal Scalar Vol Fail Along Path

X-coordinate
Y-coordinate
Z-coordinate
Total Coordinate
X-displacement
[✓ Y-displacement]
Z-displacement

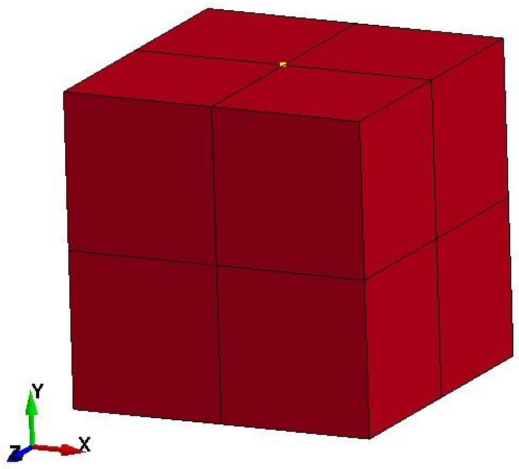
Value: Node
E-Type: Any
E-Axes: Global
Surface: Maxima

[✓ Popup] Main Split

Then plot the deformation history as a curve:

• Click **Post > History**.

• In the **History Box**, select **Nodal > Y-displacement.**

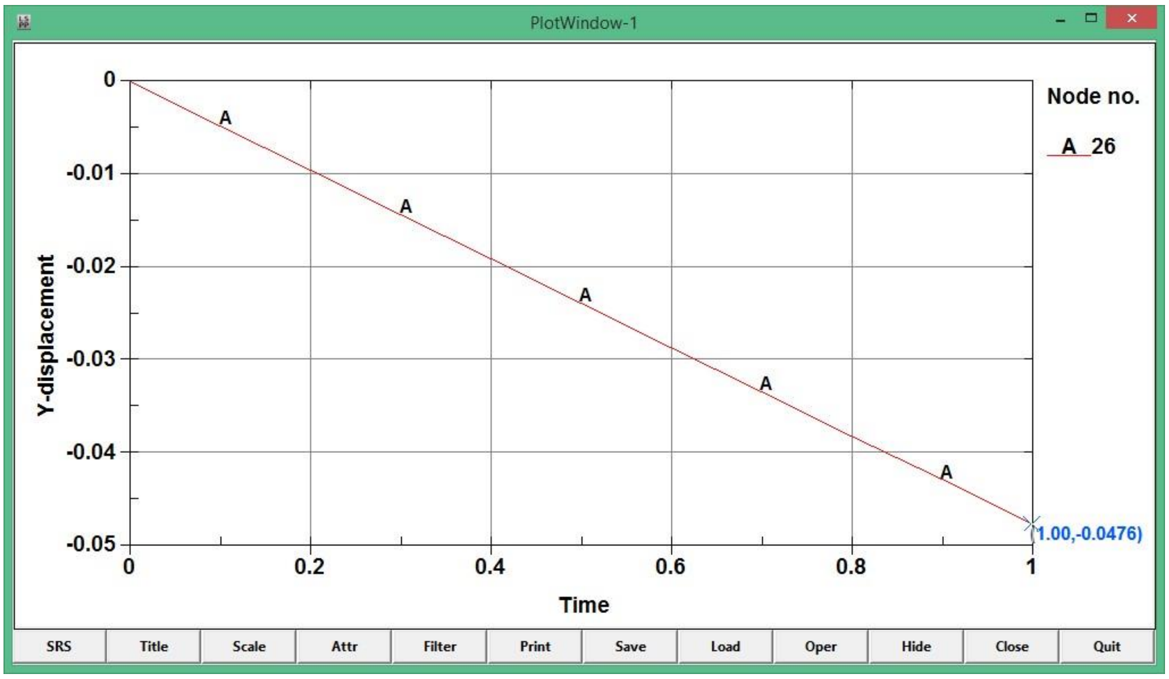


• Select a node on the top of the box.

• Click **Plot**.

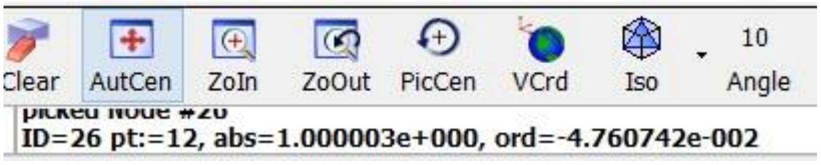
In the Plot Window, Y-displacement vs Time are stated. Zoom by pressing Ctrl and make a box with the mouse. A right-click will reset the window to original. Zoom in on the curve around Time=1, click on the final state.

Note that the Y-displacement is **-0.0476 mm**.



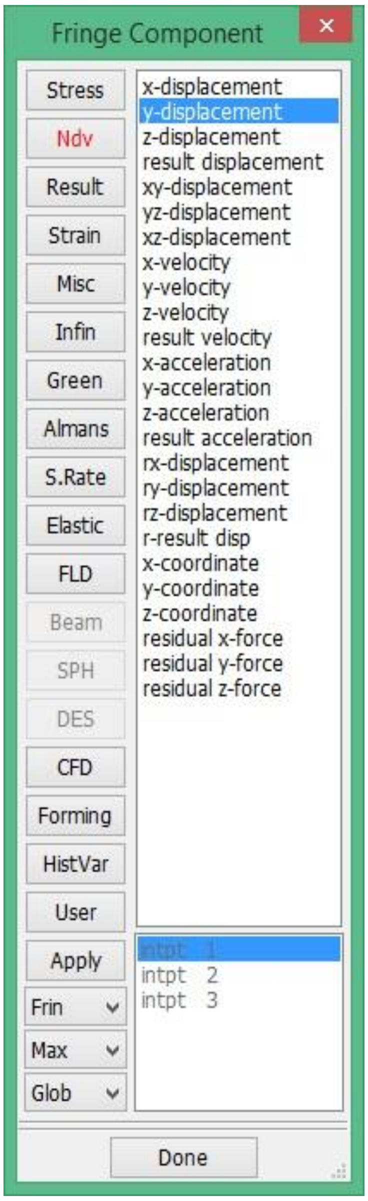
When you click on the curve, information about the picked point will be stated in the **Message box**, which is located at the bottom of the LS-PrePost window. If you double-click in this box, a bigger box will pop up.

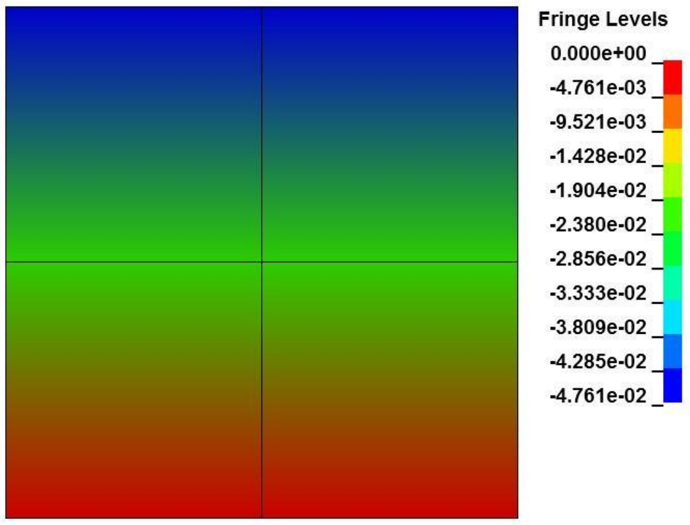
Close the **Message box** and **Plot Window.**



```
picked Node #26
ID=26 pt:=12, abs=1.000003e+000, ord=-4.760742e-002
```

Now, click **Post > FriComp> Ndv > y-displacement.** Use the **Animate toolbar** and go to the last state. The values in the **Fringe Level** shows that the maximum y displacement is -4**.761e-02 mm.**





3.5.11 Analytical solution

The analytical solution of the vertical displacement due to a 10 MPa pressure load is derived from Hooke's law.

$$Displacement = \frac{pL}{E} = \frac{10e^{6} \cdot 1}{210e^{9}} = 4.76e^{-5}\ m = \mathbf{0.0476}\ \boldsymbol{mm}$$

The simulation result should be nearly identical to the analytical solution.

4. Exercises

There are several possibilities to try out different features in LS-DYNA with this simple model. What happens if you change:

- material?

- boundary conditions?
- element formulations?
- load level?

References

Tutorials taken from: https://www.dynasupport.com/tutorial/introduction-ls-dyna-ls-prepost-for-explicit-and-implicit-analysis
https://www.dynasupport.com/manuals

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

EXP 2: Laser matter interaction FEM simulations - Tensile stress & thermal stress problem

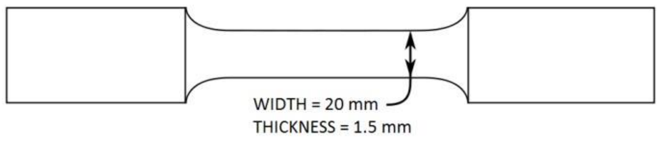
E. Kaselouris, A. Baroutsos, V. Dimitriou

HELLENIC
MEDITERRANEAN
UNIVERSITY

Tensile test

1.1 Problem Description

The task is to perform a tensile test on a flat specimen. The end of the specimen is constrained while a prescribed motion is applied on the other end. The task is to compare the stress vs. strain curve in an element when the uniaxial tension test is simulated using the explicit solver in LS-DYNA.



Material properties

Density, ρ	7850kg/m^3
Young's modulus, E	210 GPa
Poisson's Ratio, ν	0.3
Yield limit	250 MPa
Tangent modulus	1000MPa

1.2 Data files

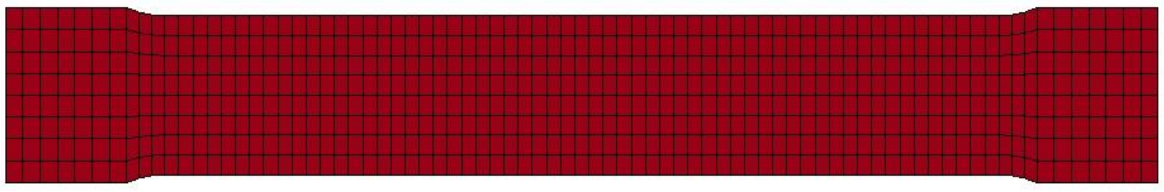
The geometry that will be used - **tensile_test.k**. and the whole code **tensile_test_results.k**

•

2. Explicit structural analysis

2.1 Read geometry

Open **tensile_test.k** in LS-PrePost, which contains the geometry of the test specimen.



2.2 Material properties

LS-DYNA accepts, for most materials, input in terms of true stress vs. true strain. Normally an experimental uniaxial tension test is performed and engineering stress and strain data are obtained. Before these data are used as an input in a material model in LS-DYNA, they are converted to true stress and strain. The curves behave as in the figure.

Engineering stress is the applied load divided by the original cross-sectional area of a material, while true stress is the applied load divided by the actual cross-sectional area (changing area with respect to time) of the specimen at that load.

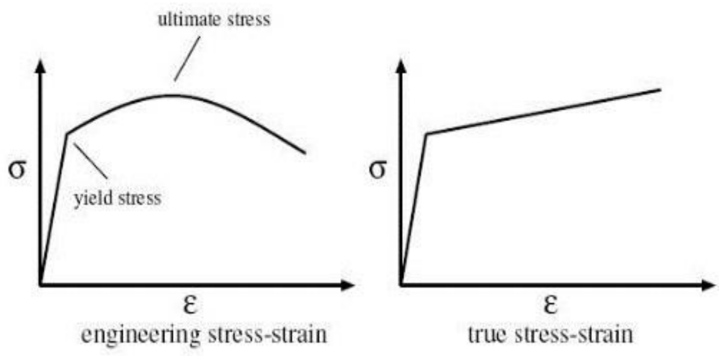
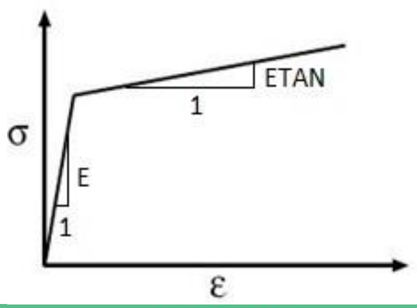


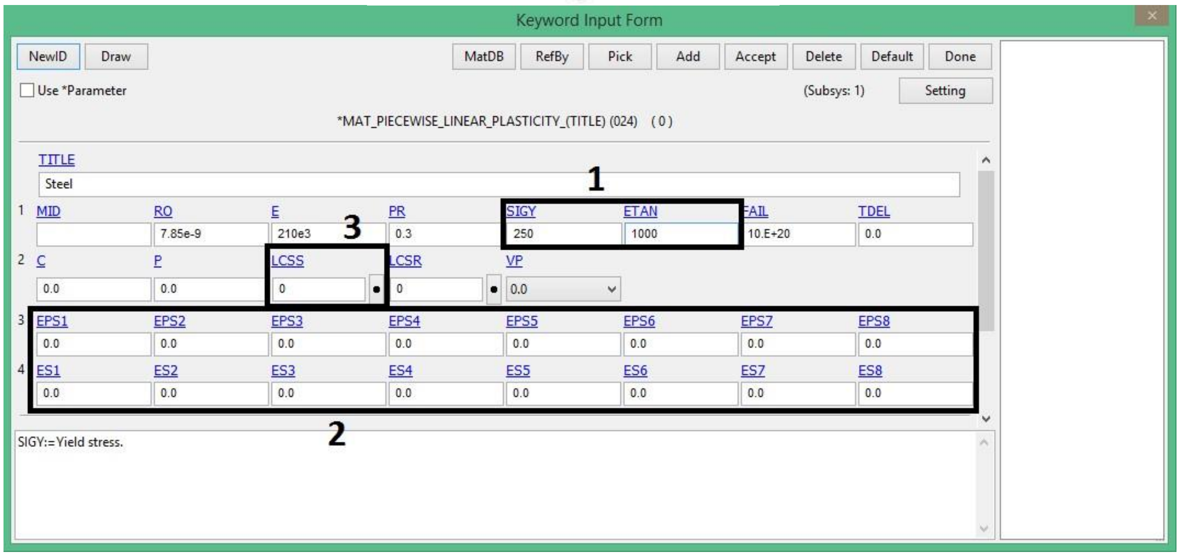
Figure 2.1 Engineering and true stress-strain curves

To create the material card:

Click **Model > Keywrd**. Activate **All** in the Keyword Manager. Double-click **MAT > 024-PIECEWISE_LINEAR_PLASTICITY**, which is an elasto-plastic material.

Define Yield stress **SIGY** and Tangent Modulus **ETAN**, which gives linear hardening.





Use *Parameter

(Subsys: 1)

*MAT_PIECEWISE_LINEAR_PLASTICITY_(TITLE) (024) (0)

TITLE

Steel

1	MID	RO	E	PR	SIGY	ETAN	FAIL	TDEL
		7.85e-9	210e3	0.3	250	1000	10.E+20	0.0
2	C	P	LCSS	LCSR	VP			
	0.0	0.0	0	0	0.0			
3	EPS1	EPS2	EPS3	EPS4	EPS5	EPS6	EPS7	EPS8
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	ES1	ES2	ES3	ES4	ES5	ES6	ES7	ES8
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

SIGY:=Yield stress.

Enter the title and the values as in the figure above and click **Accept**, then **Done**.

2.3 Element properties

To set the element formulation and properties do as follows:

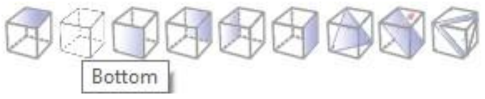
- Double click **SECTION > SHELL** in the **Keyword Manager**.
- Enter **Specimen_shell** as **TITLE**.
- Set **T1** to **1.5** and press **Enter**, **T2-T4** will then be changed to **1.5** as well and this defines the thickness of the nodes in every element.
- Click **Accept**, then **Done**.

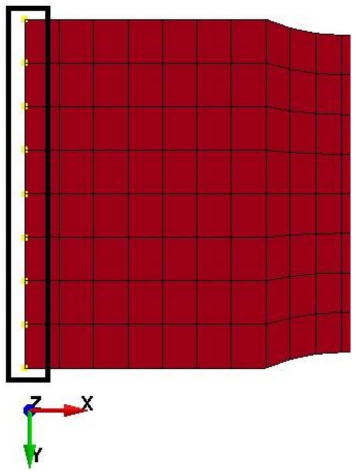
Now go to **Part** in the Keyword Manager and assign the newly created material and section. There one can also change the name of the part, to **Tensile specimen** for example**.** Click **Accept,** then **Done**.

2.4 Boundary conditions

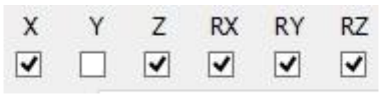
Apply the fixed boundary conditions as follows:

- Click on the **Bottom** view in the Floating Toolbar.





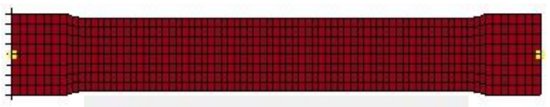
- Click **Model > CreEnt**.
- Double-click **Boundary > Spc**, click **Cre**. Make sure that **Set** is selected in the **Entity Creation** window.
- Select **Area** in the node selection window (Sel.Node).
- Make a box around the nine nodes as in the figure.
- Fix the nodes in X- and Z-translation and all rotations. This is done by activating **X**, **Z**, **RX**, **RY** and **RZ**.
- Click **Apply**.

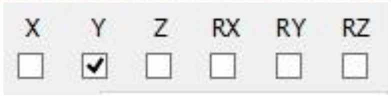


X	Y	Z	RX	RY	RZ
✓	☐	✓	✓	✓	✓

Now create another boundary condition:

- In the node selection window, write **80** in the ID box and press **Enter**. Write **87** and press **Enter**. Two nodes will now be selected at the outer edges of the test specimen. Fix the nodes in Y translation by only activating **Y**.
- Click **Apply**.
- Close **Entity Creation** window.





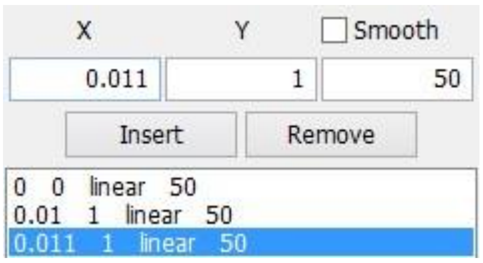
X	Y	Z	RX	RY	RZ
☐	✓	☐	☐	☐	☐

2.5 Prescribed motion/displacement

For motions and loads, a curve must be defined that states the variation of the load/displacement/velocity etc. over time.

First create the curve defining the motion:

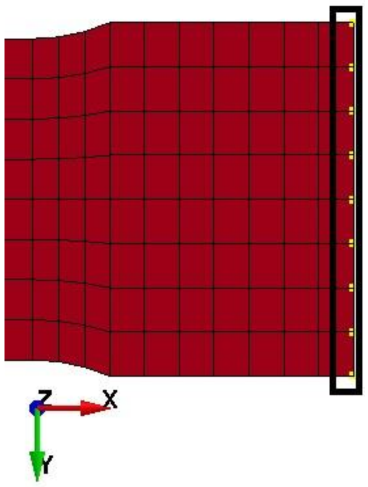
- In the top menu, click **Application > Tools > CurveGen**
- Change **Method** to **X-Y**.
- Deactivate **Smooth**.
- Write **X=0** and **Y=0** (ignore the value of smooth)
- Click **Insert.** Then **0.01** and **1, Insert.**
- Finally, **0.011** and **1, Insert** (the termination time will be 0.01 s for this tutorial).
- Change **Curve Name** to e.g. **Motion**.
- Click **Create**, then **Done**.



```
0.011    1    50
0  0  linear  50
0.01  1  linear  50
0.011  1  linear  50
```

Note that the curve could alternatively have been created using **DEFINE > CURVE** in the Keyword Manager. A curve created in CurveGen will contain a larger amount of points, compared to a curve created by **DEFINE > CURVE**.

Now apply the motion to the end of the tensile specimen:



- Click **Model > CreEnt**.
- In the **Entity Creation** window, double-click **Boundary > Prescribed Motion**.
- Select **Cre**. Change **Type** to **SET** and activate **Pick**.
- Make a box around the nine nodes as in the figure.
- Set **DOF = 1** (X-translational as degree of freedom) and **VAD = 2** (displacement as the prescribed nodal quantity, there are possibilities to prescribe the nodal velocity or acceleration as well).
- Click on **LCID** and select the previously created curve.
- Set **SF = 20**, which implies that the node set will be moved 20 mm.
- Click **Apply**, then **Done**.

2.6 Set the termination time

To set the termination time:

- Click **Model > Keywrd**.
- Double-click **CONTROL > TERMINATION**.
- Set **ENDTIM** to **0.01**. The simulation will then last for 0.01 seconds.

- Click **Accept** and then **Done**.

2.7 Output

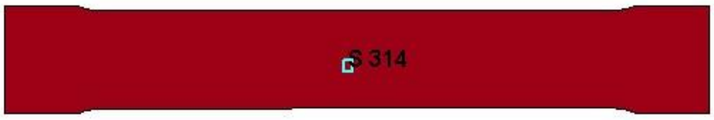
Here the output to be saved by LS-DYNA during the simulation is defined, first specify d3plot output:

- Create d3plots.
- Click **Model > Keywrd**.
- Double-click **DATABASE > BINARY_D3PLOT**.
- Set **DT = 5e-4**.
- Click **Accept** and then **Done**.

Since the task was to compare the stress vs. strain curve for an element with the material curve, we want to obtain this data frequently. In the first tutorial (1–Getting Started) we used the **History** command to plot information. The time-history graphs created by **History,** use information found in the d3plot files. To obtain data more frequently we can specify which type of data are of interest and how often it will be printed for selected elements. Therefore:

- Click **Model > Keywrd**.
- Double-click **DATABASE > ASCII_option**. We are interested in the stress and strains in the elements, therefore activate **ELOUT**. On the same row, set **DT = 5e-6**.
- Click **Accept**, then **Done**.

The element/elements that we want to gather data from must be defined:



- Double-click **DATABASE > HISTORY_SHELL** in the **Keyword Manager**.
- Write, for example, shell element number **314** under **ID1**, which is an element close to the center of the specimen.
- Click **Insert, Accept,** then **Done.**
- Check the model to see that everything looks okay.

2.8 Save

The model is now ready to be saved. **File > Save As > Save Keyword As**. Choose a folder path and name your file **tensile_test_model.k** for example. **Note** that the folder path cannot contain any spaces. Close **LS-PrePost**.

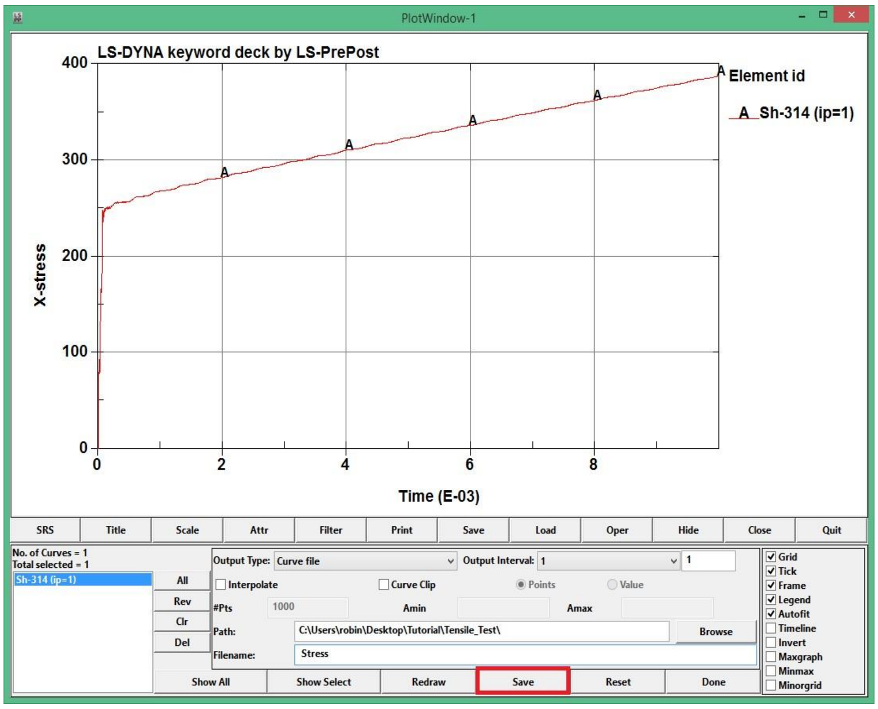
2.9 Run the simulation

Run the simulation you will get information by the instructor.

2.10 Post processing

Open the **d3plot** in LS-PrePost from **LS-Run** or from the File menu of LS-PrePost. Start to press the **Forward** button in **Animate** toolbar to see what happens with the specimen. Now plot the stress-strain-curve from the elout file, first plot and save stress vs. time:

- Click **Post > ASCII**.
- Select **elout*** and click **Load**.
- Select **Sh-314**, **Ip-1** or **Ip-2** (doesn't matter which integration point in this case) and **9Effective Stress (v-m)**.
- Click **Plot.**
- In the **PlotWindow,** click **Save.**
- Let the **Output Type** be **Curve file**.
- Enter **Stress** as the **Filename** (also set correct folder path).
- Click **Save** (located at the bottom toolbar, see red square).
- Click **Quit**



No. of Curves = 1
Total selected = 1
Sh-314 (ip=1)

Output Type: Curve file
Output Interval: 1 | 1
☐ Interpolate
☐ Curve Clip
● Points
○ Value
#Pts 1000
Amin
Amax
Path: C:\Users\robin\Desktop\Tutorial\Tensile_Test\
Filename: Stress

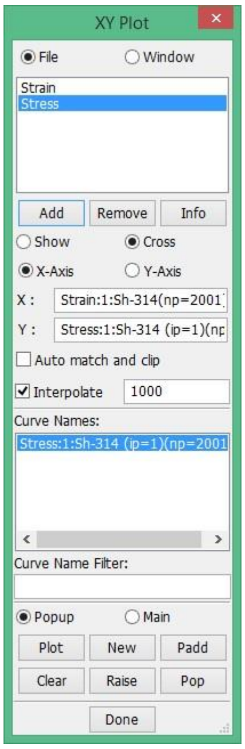
✓ Grid
✓ Tick
✓ Frame
✓ Legend
✓ Autofit
☐ Timeline
☐ Invert
☐ Maxgraph
☐ Minmax
☐ Minorgrid

Now plot and save the effective plastic strain vs. time:

- Select **7-Yield Value (eps)** instead of **Effective Stress (v-m)**.
- Click **Plot**.
- Save the curve in the same way as for the stress, but name the curve **Strain**.

- Close **PlotWindow** and **ASCII**

Now combine the above two curves (Stress and Strain) to plot the stress vs. strain:



[✓ File] Window

[✓ Cross] Show

[✓ X-Axis] Y-Axis

X : Strain:1:Sh-314(np=2001

Y : Stress:1:Sh-314 (ip=1)(np

[✓ Interpolate] 1000

Curve Names: Stress:1:Sh-314 (ip=1)(np=2001

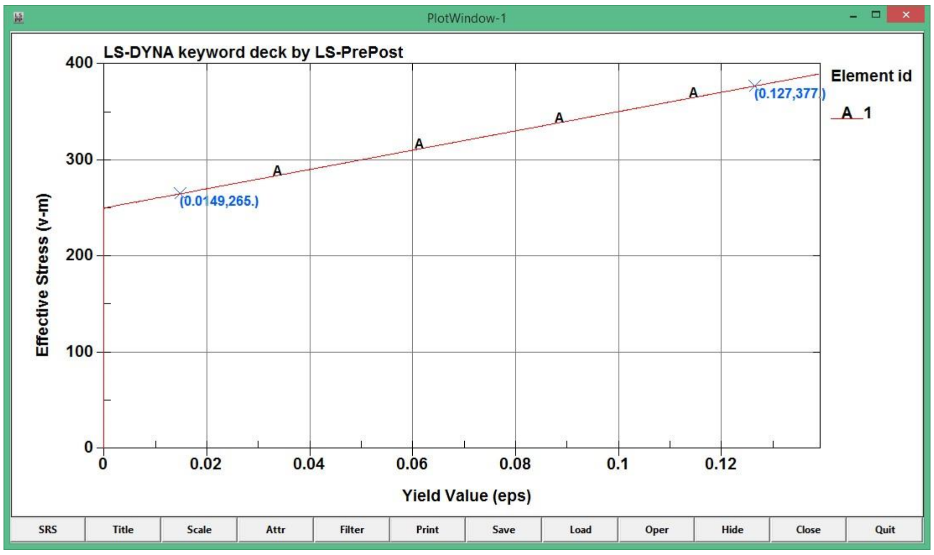
[✓ Popup] Main

- Click **Post > XYPlot**.
- Activate **Cross.**
- Note that **X-Axis** is activated, select **Strain** in the top box and it will popup under **Curve Names**.
- Click on **Strain:1:Sh-314…..** and it will be inserted in **X:**.

When you do this, **Y-Axis** will be activated.

- Select **Stress** in the top box and click on **Stress:1:Sh-314…** under **Curve Names** and it will be inserted in **Y:**.
- Click **Plot**.

The curve shows effective plastic strain vs. effective stress (von Mises). The yield limit is about 250 MPa as was stated in the material model.



Let's check that the tangent modulus **ETAN** (1000 MPa) are correct as well. Click on two points on the curve and calculate the tangent. Using 2 points in the figure gives:

$$\frac{377\ MPa - 265\ MPa}{} = 999\ MPa$$

$$0.127 - 0.0149$$

2.11 Work to be done

1) For practice change the boundary conditions so that it is fixed on the right side and loaded on the left side.

2) **3D Thermal stress problem**. This problem addresses the unconstrained expansion of a block due to heating. The model consists of one 8 node brick element at an initial temperature of 10°C. The brick material is given a volumetric thermal generation rate. It holds:

$$qVt = mc\Delta T$$

Where:

volumetric heat generation rate	$q = 10\ \mathrm{W/m^3}$
volume	$V = 1\ \mathrm{m^3}$
mass	$m = \rho V = (1.)(1.) = 1.\ \mathrm{kg}$
heat capacity	$c = 1\ \mathrm{J/kg\ C}$
time for heat generation	$t = 3\ \mathrm{sec}$

The block increases in temperature by 30 C. The final block temperature is 10C + 30C = 40C.

A tangent coefficient of thermal expansion is defined by

$$\lambda = 2.0e-07 * T$$

The thermal expansion can be calculated by

$$\Delta l = \int_{10}^{40} 2.0e-07 * TdT = \left[1.0e-07 * T^2\right]_{10}^{40} = 1.5e-04$$

The problem should be specified as a coupled thermal structural analysis in the (*CONTROL_SOLUTION) section.
LS-DYNA uses a tangent coefficient of thermal expansion, which is defined as the slope of the thermal strain versus temperature curve for the material.
The mechanical mass scaled time step is set to 0.01 seconds (*CONTROL_TIMESTEP) and the thermal time step is set to 0.1 seconds (*CONTROL_THERMAL_TIMESTEP). Explicit time integration is used for the structural calculations and implicit time integration is used for the thermal calculations. Implicit time integration is unconditionally stable and, therefore, a larger thermal time step can be taken.

Task: You will be given the LS-DYNA input file and the results. Your task is to understand the building of the model and the usage of the keywords.
The following **keywords** are needed for the simulation:

***CONTROL_SOLUTION**

```
$#   soln      nlq     isnan     lcint
       2        0        0      100
```

The Keyword *CONTROL_SOLUTION is used to specify the analysis solution procedure when there is thermal, coupled thermal or structural analysis is performed.

Variables:
Soln: Analysis solution procedure:
0: Structural analysis only,
1: Thermal analysis only,
2: Coupled structural thermal analysis.

***CONTROL_TERMINATION**

```
$#  endtim    endcyc     dtmin    endeng    endmas     nosol
      1.2         0       0.0       0.01.000000E8        0
```

The Keyword *CONTROL_TERMINATION is used to set the termination conditions.

Variables:
endtim: It defines the termination time of the analysis. It is a **mandatory** variable for the LS-DYNA software.
endcyc: It defines the maximum number of steps that will be analyzed, according to the step time (dtstart) that we will set for out problem's analysis. Always variable endtim is more important and analysis will be terminated if we reach endtim value.
dtmin: It defines the minimum time between steps that is allowed. When the time step drops to dtmin, LS-DYNA terminates. **By default, is inactive**.
endeng: This variable sets the maximum total energy change ratio that is allowable. If succeeded analysis is terminated. **By default, is inactive**.
endmas: This variable sets the maximum total mass change ratio that is allowable. If succeeded analysis is terminated. **By default, is inactive**.

***CONTROL_THERMAL_SOLVER**

```
$#   atype     ptype    solver     cgtol      gpt    eqheat     fwork       sbc
       1        2       31.00000E-6        1       1.0       1.0       0.0
$#  msglvl    maxitr    abstol    reltol     omega    unused    unused       tsf
       0       5001.0000E-101.00000E-4       1.0                            1.0
```

The keyword `*CONTROL_THERMAL_SOLVER` sets options for the thermal solution in a thermal only or coupled structural-thermal analysis. To use it is required the usage of the proper solver (variable `soln`) in `*CONTROL_SOLUTION` keyword.

***CONTROL_THERMAL_TIMESTEP**

```
$#     ts      tip      its      tmin      tmax    dtemp      tscp     lcts
        0      0.5    0.0011.00000E-7   0.0139    100.0      0.5        0
```

The keyword `*CONTROL_THERMAL_TIMESTEP` sets timestep for a thermal only or coupled structural-thermal analysis. To use it is required the usage of the proper solver (variable `soln`) in `*CONTROL_SOLUTION` keyword and the `*CONTROL_THERMAL_SOLVER` keyword.

***CONTROL_TIMESTEP**

```
$#  dtinit   tssfac     isdo    tslimt     dt2ms      lctm     erode     ms1st
      0.0      0.6        0       0.0       0.0         0        0         0
$#  dt2msf  dt2mslc    imscl   unused   unused     rmscl
      0.0        0        0                  0.0
```

The keyword *CONTROL_TIMESTEP sets the timestep for structural problem analysis.

dtini: This variable defines the initial timestep size. By default, is blank or 0 and LS-DYNA determines initial step size.

***DATABASE_BINARY_D3PLOT**

```
$#     dt     lcdt     beam     npltc    psetid
      0.0        0        0      100        0
$#   ioopt
        0
```

The keyword *DATABASE_BINARY_D3PLOT defines parameters for the output from entire of the model. From these outputs by post processing them we will conclude to an final presentable output of our solution analysis.

dt: In this variable we set the time interval between output states.
nr: In this variable we set Number of RUNning ReStart Files, RUNRSF, written in a cyclical fashion. The default is 1, i.e., only one runrsf file is created and the data there in is overwritten each time data is output.
npltc: In this variable we calculate DT = ENDTIME/NPLTC applies to D3PLOT and D3PART only. This overrides the DT specified in the first field.

***DATABASE_GLSTAT**

```
$#        dt     binary       lcur      ioopt
       0.003          0          0          1
```

The keyword `*DATABASE` has many options to define to obtain output files containing results information. One of them is the `*DATABASE_GLSTAT,` which provide us with the global data of the solution analysis of our model.

`Dt` : In this variable we se the time step between outputs. If is set to 0 then no output is printed.
`binary` : In this variable we set the type for binary output.

1 : Data written to an ASCII file default for SMP LS-DYNA.
2 : Data written to binary database “binout”, default for MPP LS-DYNA.
3 : Both ASCII and binary outputs.

***PART**

```
$#                                                    title

$#    pid    secid     mid    eosid     hgid     grav   adpopt     tmid
        2        1       1        1        0        0        0        1
```

The keyword *PART is essential for the model identification and characterization in the solution analysis. Within it variables we combine various properties and messes to form the parts we want to analyze. To form a part in LS-DYNA we combine the identity of a formed mess with the identity of the material, with the identity of the used Equation Of State for that material, with the thermal properties identity, etc.

pid: In this variable we put the part id number
secid: In this variable we put the section id number, defined in a *SECTION keyword.
mid: In this variable we put the material id number, defined in a *MAT keyword.
eosid: In this variable we put the Equation Of State id number, defined in a *EOS keyword.
tmid: In this variable we put the thermal properties id number, defined in a *MAT_THERMAL keyword.

***SECTION_SOLID**

```
$#   secid    elform      aet
         1         1        0
```

In this keyword, we define section properties for solid continuum and fluid elements.

secid: In this variable we put the section id number, defined in a *SECTION keyword.
elform: Element formulation options:

-2: Fully integrated S/R solid intended for elements with poor aspect ratio, accurate formulation

-1: Fully integrated S/R solid intended for elements with poor aspect ratio, efficient formulation

0: 1 point corotational for *MAT_MODIFIED_HONEYCOMB
1: Constant stress solid element: default element type.
2: Fully integrated S/R solid
3: Fully integrated quadratic 8 node element with nodal rotations
4: S/R quadratic tetrahedron element with nodal rotations
5: 1 point ALE
6: 1 point Eulerian
7: 1 point Eulerian ambient
8: Acoustic
9: 1 point corotational for *MAT_MODIFIED_HONEYCOMB
10: 1 point tetrahedron
11: 1 point ALE multi-material element

and various others.

***MAT_ELASTIC_PLASTIC_THERMAL**

Task: Determine the usage of this material keyword.

***MAT_THERMAL_ISOTROPIC**

```
$#    tmid      tro    tgrlc   tgmult     tlat      hlat
         1       0.0      0.0      0.0      0.0       0.0
$#      hc        tc
     560.03.70000E-5
```

This keyword *MAT_THERMAL_ISOTROPIC defines thermal isotropic properties to the material that selects it.

tmid: In this variable we set the unique ID number of the thermal material identification.
tro: In this variable we set the thermal density of the material. If set to 0.0 then it is equal to the structural density.
hc: In this variable we set the specific heat of the material
tc: In this variable we se the thermal conductivity of the material.

***ELEMENT_SOLID**

```
$#   eid     pid     n1      n2      n3      n4      n5      n6      n7      n8
       1       1   4313    4309    5283    5313    4308    4308    5138    5138
       2       1   5313    5283    5284    5314    5138    5138    5139    5139
       3       1   5314    5284    5285    5315    5139    5139    5140    5140
```

In this keyword *ELEMENT_SOLID we define each element of the created mess of our model that we are going to perform solution analysis. Elements are defined in three-dimensional solid elements including 4 noded tetrahedrons and 8-noded hexahedrons. Most common are the 8-noded hexahedrons.

eid: In this variable we set the unique ID number of the element.
pid: In this variable we set part ID number from with this element comes from.
n1 – n8: In these variables we set the nodal points ID that consists the element. If we use tetrahedron shape for our elements then we set only n1 – n4 nodal point variables.

***NODE**

```
$#   nid               x                y                z      tc      rc
       1       -0.027596         0.019934           0.2715       0       0
       2       0.0576701              1.0           0.2715       0       0
       3       -0.027596         0.019934           0.1285       0       0
```

In the keyword *NODE we define the nodes of our model. We also can define boundary conditions for that nodes. Nodes are defined by setting their coordinates at X-Y-Z axis.

nid: In this variable we set the unique node ID number
x: In this variable we set the X coordinate of the node
y: In this variable we set the Y coordinate of the node
z: In this variable we set the Z coordinate of the node
tc: In this variable we set the translational constraint:

    0: no constraints,
    1: constrained x displacement,

2: constrained y displacement,
3: constrained z displacement,
4: constrained x and y displacements,
5: constrained y and z displacements,
6: constrained z and x displacements,
7: constrained x, y, and z displacements.

rc: In this variable we set the rotational constraint:

0: no constraints,
1: constrained x rotation,
2: constrained y rotation,
3: constrained z rotation,
4: constrained x and y rotations,
5: constrained y and z rotations,
6: constrained z and x rotations,
7: constrained x, y, and z rotations.

***INITIAL_TEMPERATURE_NODE**

It defines the initial temperature for the selected nodes.

References

Tutorials taken from: https://www.dynasupport.com/tutorial/introduction-ls-dyna-ls-prepost-for-explicit-and-implicit-analysis
https://www.dynasupport.com/manuals

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

EXP 3: Laser matter interaction FEM simulations- Laser heating of a metal

E. Kaselouris, A. Baroutsos, V. Dimitriou

HELLENIC
MEDITERRANEAN
UNIVERSITY

1. Laser heating of a metal

1.1 Problem Description

The heating of a AISI H13 steel workpiece is examined during irradiation by a CW continuous laser. 3D transient coupled thermal-structural numerical simulations take place using LS-DYNA.

The workpiece is defined as a deformable body. The laser beam is modeled as a Gaussian moving heat source. For the dynamic elastoplastic behavior of the workpiece: a Johnson-Cook constitutive strength material model will be used.

2 Governing equations

2.1 3D transient heat conduction equation

$$\rho c\left(\frac{\partial T}{\partial t}+V_x\frac{\partial T}{\partial x}\right)=\frac{\partial}{\partial i}\left(\kappa(T)\frac{\partial T}{\partial i}\right)+\dot{Q}-a_T T_0(3\lambda+2\mu)\dot{\varepsilon}_{ii} \tag{1}$$

i=x,y,z coordinates, *t* time

ρ the density, *c* the specific heat, V_x the laser scanning speed along the x-direction

k thermal conductivity and $\dot{Q}$ the power generation per unit volume

T temperature, T_0 ambient temperature, α_T thermal expansion coefficient,

λ, μ lame coefficients, $\dot{\varepsilon}_{ii}$ strain rate

2.2 3D transient mechanical equation

$$\rho\frac{\partial^2 U_i}{\partial^2 t}=\mu\frac{\partial^2 U_i}{\partial^2 k}+(\lambda+\mu)\frac{\partial}{\partial i}\left(\frac{\partial U_k}{\partial k}\right)-(3\lambda+2\mu)\alpha_T\frac{\partial T}{\partial i} \tag{2}$$

U the displacement, *ρ* the density, *λ and μ* lame coefficients, α_T thermal expansion coefficient

The strain tensor

$$\varepsilon_{ij}=\frac{1}{2}\left(\frac{\partial U_i}{\partial U_j}+\frac{\partial U_j}{\partial U_i}\right) \tag{3}$$

The stress tensor

$$\sigma_{ij}=2\mu\varepsilon_{ij}+\lambda\varepsilon_{kk}\delta_{ij}-(3\lambda+2\mu)a_T\,(T-T_0)\delta_{ij} \tag{4}$$

2.3 Modeling of laser heat source

The Gaussian distribution of the *q(r)* absorbed laser heat flux or laser power intensity is given by:

$$q(x,z) = \frac{2P_{tot}}{\pi r_b^2} e^{-(\frac{2((x-tV_x)^2+z^2)}{r_b^2})} \qquad (5)$$

where P_{tot} is the total absorbed power and r_b is the laser beam radius. It also holds that:

$$P_{tot} = \eta P_{inc} \qquad (6)$$

where P_{inc} is the incident laser power, η is the average absorptivity of the workpiece material and *t* is the time.

The laser heat flux is applied to the top surface of the workpiece. The boundary condition on the top laser irradiated surface takes into account the heat flux, convection and radiation and it holds:

$$-k\frac{\partial T}{\partial y} = q(x,z) - h(T - T_0) - \sigma\varepsilon(T^4 - T_0^4) \qquad (7)$$

where *h* is the convective heat transfer coefficient, T_0 is the ambient temperature, σ is the Stefan–Boltzmann constant (5.67 10^8W/m^{2}K^4) and ε is emmisivity. Heat flux is considered to be normal to the laser irradiated surface, while the motion of the laser beam is considered along the X-direction.

Moreover, the laser beam diameter is considered to be **250 μm**, the laser power is **98.2 W** and the heat flux is **2 kW/mm^2**.

3 Finite Element Modeling

The workpiece dimensions are 3.0 x 0.4 x 0.5 mm. The workpiece is modeled with approximately 115,000 solid elements. The mesh of the workpiece is shown in Figure 1.

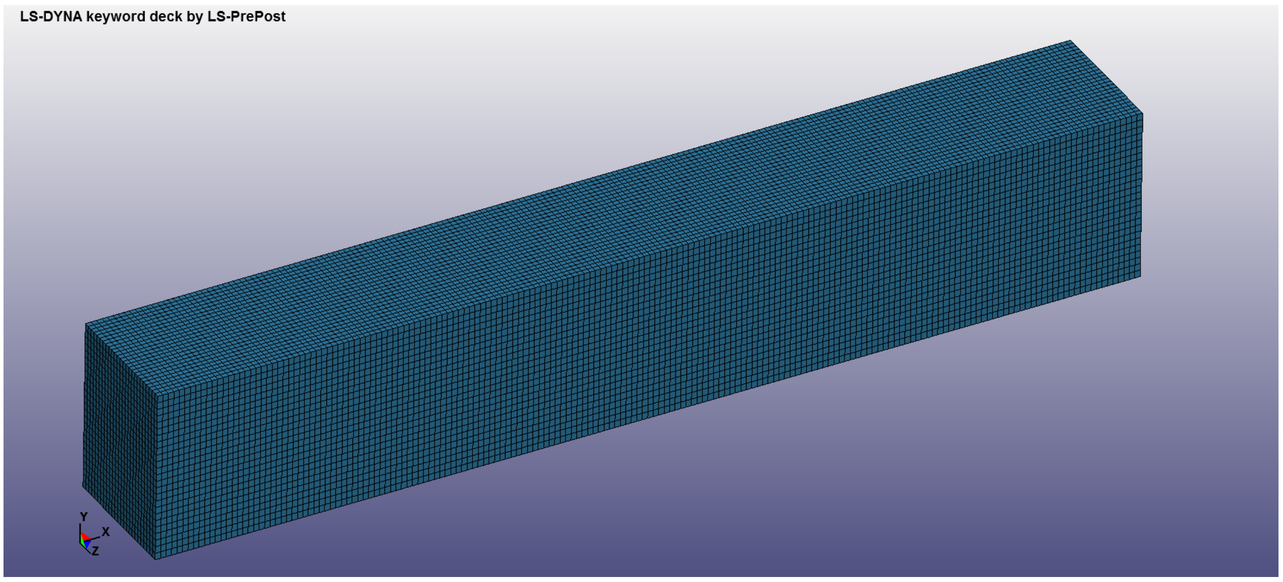


Figure 1.1 Mesh and geometry of the workpiece

Regarding the initial and boundary conditions, all translations and rotations of the bottom and the left side of the workpiece are fully constrained and Z translations at its front and back side are also constrained. The ambient temperature is assumed to be 20 ^{0}C.

3.1 Material model

The adoption of a suitable material-constitutive model for the workpiece is critical. The selected material model is the Johnson-Cook (J-C), a purely empirical one that takes into account the effect of plastic strain, strain rate and temperature. The flow stress is expressed as:

$$\sigma_y = (A + B\varepsilon^n)(1 + Cln\frac{\dot{\varepsilon}}{\dot{\varepsilon}_0})(1 - \frac{T - T_r}{T_m - T_r})^m \tag{8}$$

where ε is the equivalent plastic strain, $\dot{\varepsilon}/\dot{\varepsilon}_0$ is the dimensionless plastic strain rate, $\dot{\varepsilon}_0$=1s^{-1} is a reference strain rate used to normalize the strain rate, *A* is the yield stress, *B* is the hardening constant, *C* is the strain rate sensitivity, *n* is the hardening exponent, *m* is the thermal softening exponent, T_m is the melting temperature of the workpiece and T_r the room temperature. The material constants *A*, *B*, *C*, *n*, *m* are determined from experimental results.

3.2 Material properties

The physical properties of the AISI H-13 steel workpiece are listed in Table 1 for room temperature. The J-C material model and failure parameters of the workpiece are listed in Table 2.

Property	Workpiece
Density [kg/m^3]	7800
Elastic modulus [GPa]	211
Poisson's ratio [-]	0.28
Specific heat [J/kg K]	560
Thermal conductivity [W/m K]	37
Thermal expansion [10^{-6}C]	10.4
Melting Point [K]	1700

Table 1.1 Mechanical and physical properties of AISI H-13 in room temperature

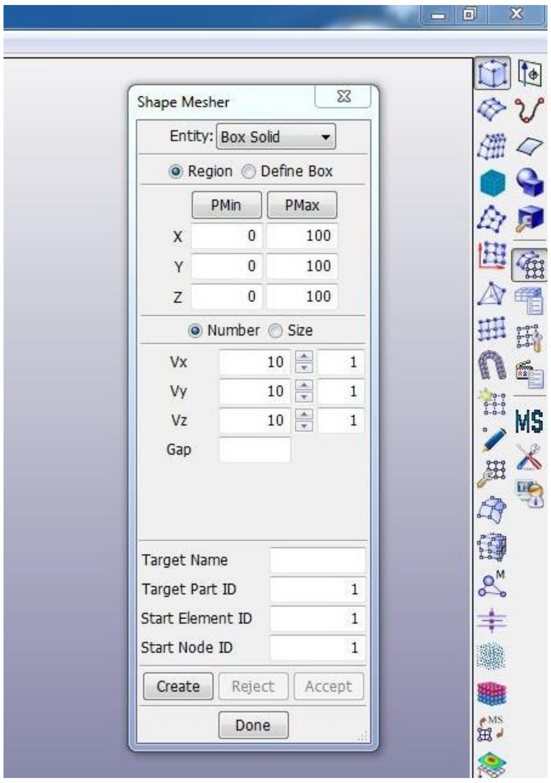
Material model Parameters	A [MPa]	B [MPa]	n [-]	C [-]	m [-]
Values	674.8	239.2	0.28	0.027	1.3

Table 1.2 Johnson-Cook material model and failure parameters of AISI H-13

Furthermore, a mean value of absorptivity η=**0.5** and a mean value of emmisivity ε=**0.4** are considered based on the work of Singh et al. For the convectional heat transfer to the surrounding air, a heat transfer coefficient of h=5 W/m^{2}K is also considered. The laser scanning speed is considered to be **150 m/min**.

4 Work to be done

1) Read the keywords of the code so you can understand what each of them does.
2) From the code that you will be given identify first which consistent units you use.
3) Create a mesh that will have dimensions 3 x 0.4 x 0.5 mm along the x,y,z directions and the discretization will be 225 x 30 x 30 respectively resulting in a total number of 202500 elements using the Shapemesher command.


Shape Mesher

Entity: Box Solid

(● Region) ○ Define Box

	PMin	PMax
X	0	100
Y	0	100
Z	0	100

(● Number) ○ Size

Vx	10	1
Vy	10	1
Vz	10	1
Gap		

Target Name	
Target Part ID	1
Start Element ID	1
Start Node ID	1

Figure 1.2 Shapemesher

4) Constrain all the translations and rotations at the bottom and the left side of the workpiece. Moreover constrain the Z translations at its front and back side. For this purpose you will need to use the keywords *BOUNDARY_SPC_SET, *SET_NODE_LIST.
5) Define the surface where the laser heat flux is applied. This will be on the top of the workpiece. Use the *SET_SEGMENT command.
6) Then your work will be examined by the instructor and he will further help you to complete the simulation.

The following **keywords** needed for the simulation are explained:

***CONTROL_SOLUTION**

```
$#     soln       nlq     isnan     lcint
          2         0         0       100
```

The Keyword *CONTROL_SOLUTION is used to specify the analysis solution procedure when there is thermal, coupled thermal or structural analysis is performed.

Variables:
Soln: Analysis solution procedure:
0: Structural analysis only,
1: Thermal analysis only,
2: Coupled structural thermal analysis.

***CONTROL_TERMINATION**

```
$#   endtim    endcyc     dtmin    endeng    endmas     nosol
        1.2         0       0.0       0.01.000000E8         0
```

The Keyword *CONTROL_TERMINATION is used to set the termination conditions.

Variables:
endtim: It defines the termination time of the analysis. It is a **mandatory** variable for the LS-DYNA software.
endcyc: It defines the maximum number of steps that will be analyzed, according to the step time (dtstart) that we will set for out problem's analysis. Always variable endtim is more important and analysis will be terminated if we reach endtim value.
dtmin: It defines the minimum time between steps that is allowed. When the time step drops to dtmin, LS-DYNA terminates. **By default, is inactive**.
endeng: This variable sets the maximum total energy change ratio that is allowable. If succeeded analysis is terminated. **By default, is inactive**.
endmas: This variable sets the maximum total mass change ratio that is allowable. If succeeded analysis is terminated. **By default, is inactive**.

***CONTROL_THERMAL_SOLVER**

```
$#    atype     ptype    solver     cgtol       gpt    eqheat
fwork       sbc
          1         2          31.00000E-6         1       1.0
1.0       0.0
$#   msglvl    maxitr    abstol    reltol     omega    unused
unused      tsf
          0       5001.0000E-101.00000E-4       1.0
```

The keyword `*CONTROL_THERMAL_SOLVER` sets options for the thermal solution in a thermal only or coupled structural-thermal analysis. To use it is required the usage of the proper solver (variable `soln`) in `*CONTROL_SOLUTION` keyword.

***CONTROL_THERMAL_TIMESTEP**

```
$#        ts       tip        its      tmin       tmax      dtemp
tscp      lcts
           0       0.5     0.0011.00000E-7    0.0139      100.0
0.5          0
```

The keyword `*CONTROL_THERMAL_TIMESTEP` sets timestep for a thermal only or coupled structural-thermal analysis. To use it is required the usage of the proper solver (variable `soln`) in `*CONTROL_SOLUTION` keyword and the `*CONTROL_THERMAL_SOLVER` keyword.

***CONTROL_TIMESTEP**

```
$#  dtinit    tssfac      isdo    tslimt     dt2ms      lctm
erode     ms1st
       0.0       0.6         0       0.0       0.0         0
0         0
$#  dt2msf   dt2mslc     imscl    unused    unused     rmscl
       0.0         0         0                           0.0
```

The keyword *CONTROL_TIMESTEP sets the timestep for structural problem analysis.

dtini: This variable defines the initial timestep size. By default, is blank or 0 and LS-DYNA determines initial step size.

***CONTROL_THERMAL_NONLINEAR**

```
$#  refmax       tol       dcp    lumpbc    thlstl    nlthpr
phchpn
       100       0.0       1.0         0       0.0         0
0.0
```

The keyword *CONTROL_THERMAL_NONLINEAR set parameters for a nonlinear thermal or coupled structural-thermal analysis. The control card, *CONTROL_SOLUTION, (variable soln) is also required.

***DATABASE_BINARY_D3PLOT**

```
$#    dt    lcdt    beam    npltc   psetid
     0.0       0       0      100        0
$#  ioopt
       0
```

The keyword *DATABASE_BINARY_D3PLOT defines parameters for the output from entire of the model. From these outputs by post processing them we will conclude to an final presentable output of our solution analysis.

dt: In this variable we set the time interval between output states.
nr: In this variable we set Number of RUNning ReStart Files, RUNRSF, written in a cyclical fashion. The default is 1, i.e., only one runrsf file is created and the data there in is overwritten each time data is output.

npltc: In this variable we calculate DT = ENDTIME/NPLTC applies to D3PLOT and D3PART only. This overrides the DT specified in the first field.

***DATABASE_GLSTAT**

```
$#        dt     binary       lcur      ioopt
      0.003          0          0          1
```

The keyword `*DATABASE` has many options to define to obtain output files containing results information. One of them is the `*DATABASE_GLSTAT,` which provide us with the global data of the solution analysis of our model.

`Dt`: In this variable we se the time step between outputs. If is set to 0 then no output is printed.
`binary`: In this variable we set the type for binary output.

1: Data written to an ASCII file default for SMP LS-DYNA.
2: Data written to binary database “binout”, default for MPP LS-DYNA.
3: Both ASCII and binary outputs.

***MAT_ADD_THERMAL_EXPANSION**

```
$#   pid   lcid   mult   lcid   multy   lcid   multz
       2     69    1.0      0     1.0      0     1.0
```

With this keyword *MAT_ADD_THERMAL_EXPANSION we add thermal expansion properties to all nonlinear solid, shell, thick shell and beam elements and all material models.

pid: In this variable we put the PART ID of the part that we add thermal expantion
lcid: For isotropic material models, LCIDY, MULTY, LCIDZ, and MULTZ are ignored, and LCID is the load curve ID defining the thermal expansion coefficient as a function of emperature. If zero, the thermal expansion coefficient is constant and equal to MULT.
`mult`: Scale factor scaling load curve given by LCID.

***PART**

```
$#                                                title

$#   pid   secid    mid   eosid   hgid   grav   adpopt   tmid
       2       1      1       1      0      0        0      1
```

The keyword *PART is essential for the model identification and characterization in the solution analysis. Within it variables we combine various properties and messes to form the parts we want to analyze. To form a part in LS-DYNA we combine the identity of a formed mess with the identity of the material, with the identity of the used Equation Of State for that material, with the thermal properties identity, etc.

pid: In this variable we put the part id number

secid: In this variable we put the section id number, defined in a *SECTION keyword.
mid: In this variable we put the material id number, defined in a *MAT keyword.
eosid: In this variable we put the Equation Of State id number, defined in a *EOS keyword.
tmid: In this variable we put the thermal properties id number, defined in a *MAT_THERMAL keyword.

***SECTION_SOLID**

```
$#   secid    elform      aet
         1          1          0
```

In this keyword, we define section properties for solid continuum and fluid elements.

secid: In this variable we put the section id number, defined in a *SECTION keyword.
elform: Element formulation options:
-2: Fully integrated S/R solid intended for elements with poor aspect ratio, accurate formulation
-1: Fully integrated S/R solid intended for elements with poor aspect ratio, efficient formulation
0: 1 point corotational for *MAT_MODIFIED_HONEYCOMB
1: Constant stress solid element: default element type.
2: Fully integrated S/R solid
3: Fully integrated quadratic 8 node element with nodal rotations
4: S/R quadratic tetrahedron element with nodal rotations
5: 1 point ALE
6: 1 point Eulerian
7: 1 point Eulerian ambient
8: Acoustic
9: 1 point corotational for *MAT_MODIFIED_HONEYCOMB
10: 1 point tetrahedron
11: 1 point ALE multi-material element
and various others

***MAT_JOHNSON_COOK_TITLE**

```
*MAT_JOHNSON_COOK_TITLE
blank
$#      mid         ro          g          e         pr        dtf
vp    rateop
          17.80000E-6       81.0      211.0       0.281.80000E-7
0.0        0.0
$#        a          b          n          c          m         tm
tr       epso
     0.675      0.239       0.28      0.027        1.3     1427.0
20.0      0.001
$#       cp         pc      spall         it         d1         d2
d3         d4
     560.0        0.0        2.0        0.0       -0.8        2.1       -
0.52.00000E-4
$#       d5       c2/p       erod      efmin     numint
       2.7        0.0        01.00000E-6        0.0
```

Johnson-Cook is an empirical model for the flow stress and strain rate. It is the most common used Viscoplasticity and yield stress model.

So in this keyword (*MAT_JOHNSON_COOK_TITLE) we set the variables of the Johnson-Cook model for a material.

mid: In this variable we put the material identification number.
ro: In this variable we set the mass density of the material
g: In this variable we set the shear modulus of the material.
e: In this variable we set the Young's modulus of the material
pr: In this variable we set the Poisson's ratio of the materia
dtf: In this variable we set the Minimum time step size for automatic element deletion (shell elements).

vp: In this variable we set the formulation for rate effects:
  0.0 for scale yield stress
  1.0 for Viscoplastic formulation

a,b,n,c,m: These variables are the input constants of the Johnson-Cook formula
tm: In this variable we set the melt temperature of the material
tr: In this variable we set the room temperature
cp: In this variable we set the specific heat
pc: In this variable we set the tensile failure stress or tensile pressure cutoff (PC < 0.0)

***EOS_LINEAR_POLYNOMIAL**

```
*EOS_LINEAR_POLYNOMIAL
$#   eosid        c0        c1        c2        c3        c4
c5        c6
         1       0.0     160.0       0.0       0.0       0.0
0.0       0.0
$#      e0        v0
       0.0       1.0
```

This is the keyword *EOS_LINEAR_POLYNOMIAL where we define coefficients for a linear polynomial Equation Of State and initialize the thermodynamic state of the material.

$$P = C_0 + C_1\mu + C_2\mu^2 + C_3\mu^3 + (C_4 + C_5\mu + C_6\mu^2)E.$$

eosid: In this variable we set the unique number ID of the EOS
c0: In this variable we set the 0th polynomial equation coefficient.
c1: In this variable we set the 1st polynomial equation coefficient (**when used by itself, this is the elastic bulk modulus**)

***MAT_THERMAL_ISOTROPIC**

```
$#    tmid       tro     tgrlc    tgmult      tlat      hlat
         1       0.0       0.0       0.0       0.0       0.0
$#      hc        tc
     560.03.70000E-5
```

This keyword *MAT_THERMAL_ISOTROPIC defines thermal isotropic properties to the material that selects it.

tmid: In this variable we set the unique ID number of the thermal material identification.
tro: In this variable we set the thermal density of the material. If set to 0.0 then it is equal to the structural density.
hc: In this variable we set the specific heat of the material
tc: In this variable we se the thermal conductivity of the material.

```
*ELEMENT_SOLID
$#   eid     pid       n1       n2       n3       n4       n5       n6       n7       n8
       1       1     4313     4309     5283     5313     4308     4308     5138     5138
       2       1     5313     5283     5284     5314     5138     5138     5139     5139
       3       1     5314     5284     5285     5315     5139     5139     5140     5140
```

In this keyword *ELEMENT_SOLID we define each element of the created mess of our model that we are going to perform solution analysis. Elements are defined in three-dimensional solid elements including 4 noded tetrahedrons and 8-noded hexahedrons. Most common are the 8-noded hexahedrons.

eid: In this variable we set the unique ID number of the element.
pid: In this variable we set part ID number from with this element comes from.
n1 – n8: In these variables we set the nodal points ID that consists the element. If we use tetrahedron shape for our elements then we set only n1 – n4 nodal point variables.

```
*NODE
$#   nid               x               y               z      tc      rc
       1       -0.027596        0.019934          0.2715       0       0
       2       0.0576701             1.0          0.2715       0       0
       3       -0.027596        0.019934          0.1285       0       0
```

In the keyword *NODE we define the nodes of our model. We also can define boundary conditions for that nodes. Nodes are defined by setting their coordinates at X-Y-Z axis.

nid: In this variable we set the unique node ID number
x: In this variable we set the X coordinate of the node
y: In this variable we set the Y coordinate of the node
z: In this variable we set the Z coordinate of the node
tc: In this variable we set the translational constraint:

    0: no constraints,
1: constrained x displacement,
2: constrained y displacement,
3: constrained z displacement,
4: constrained x and y displacements,
5: constrained y and z displacements,
6: constrained z and x displacements,
7: constrained x, y, and z displacements.

rc: In this variable we set the rotational constraint:

0: no constraints,
1: constrained x rotation,
2: constrained y rotation,
3: constrained z rotation,
4: constrained x and y rotations,
5: constrained y and z rotations,
6: constrained z and x rotations,
7: constrained x, y, and z rotations.

***DEFINE_FUNCTION**

```
$#     fid                                                              heading
         3
$#                                                                     function
f(x,z,time,temp)=-(2)/(exp(128*((x-2.5*time)*(x-2.5*time)+(z-0.2)*(z-0.2))))+(5*
10**(-8))*(temp-20)+(5.7*10**(-17))*0.5*((temp**4)-160000)
```

In this keyword *DEFINE_FUNCTION we define a function that we can used by other keywords.

fid: In this variable we set the function ID number
heading: In this variable we set an optional descriptive heading.
function: In this variable we set the arithmetic expression that represents our function.

***BOUNDARY_FLUX_SET**

```
$#    ssid
         1
$#    lcid      mlc1      mlc2      mlc3      mlc4       loc     nhisv       fid
         3       1.0       1.0       1.0       1.0         0         0         3
```

The keyword `*BOUNDARY_FLUX` is to apply a flux boundary condition.

ssid: In this variable we set the set ID number
lcid : In this variable we set the ID of a reference load curve
mlc1-mlc4: In this variables we set the curve multipliers.
fid: in this variable we set the ID number of the function we use

***SET_SEGMENT**

```
$#     sid       da1       da2       da3       da4    solver
         1       0.0       0.0       0.0       0.0MECH
$#      n1        n2        n3        n4        a1        a2        a3        a4
    129285    129645    129647    129287       0.0       0.0       0.0       0.0
    123165    123525    123527    123167       0.0       0.0       0.0       0.0
    117045    117405    117407    117047       0.0       0.0       0.0       0.0
```

In this keyword `*SET_SEGMENT` we define set of segments with optional identical or unique attributes.

sid: In this variable we set the unique set ID number.

solver: In this variable we set the name of the solver of the set.
n1–n4: In this variable we set the nodal points

REFERENCES

[1] V. Dimitriou, E. Kaselouris, Y. Orphanos, E Bakarezos, Nikolaos Vainos, M. Tatarakis, N. A. Papadogiannis, Three dimensional transient behavior of thin films surface under pulsed laser excitation, *Appl. Phys. Letters* **103** 114104 (2013).
[2] E. Kaselouris, I.K. Nikolos, Y. Orphanos, E Bakarezos, N. A. Papadogiannis, M. Tatarakis, V. Dimitriou, Elastoplastic study of nanosecond-pulsed laser interaction with metallic films using 3D multiphysics fem modeling, *Int. J. Damage Mech.* **25**(1) 42-55 (2016).
[3] R. Singh, M.J. Alberts and S.N. Melkote, "Characterization and prediction of the heat-affected zone in a laser-assisted mechanical micromachining process", *International Journal of Machine Tools & Manufacture*, **48** 994–1004 (2008).

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

High Power Laser Matter Interactions/High Energy Density Physics - Theory and Experiments

EXP 4: Laser plasma interaction-PIC simulations

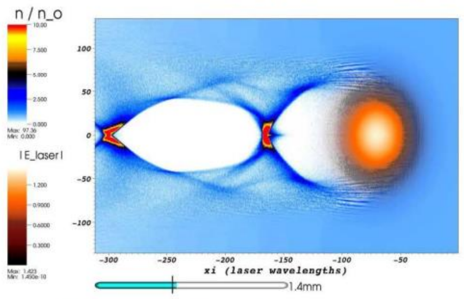
E. Clark, A. Grigoriadis, G. Andrianaki, G. Tazes

HELLENIC
MEDITERRANEAN
UNIVERSITY

Outline

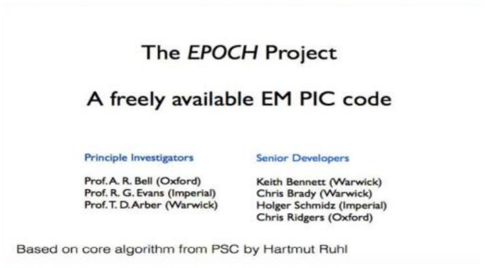
- We will simulate a laser wakefield accelerator in 2D using the EPOCH PIC code
- You will get hands on experience on how to run a PIC code and analyse the results
- We will see how the wakefield accelerator works and experiments with some parameters to see how the it affects the acceleration of electrons

Electron Acceleration



•Pondermotive force of the laser can expel electrons away from the focal spot in a plasma creating a cavity or bubble
•Electrons accumulate at the rear of the bubble creating a region of large electric field propagating close to speed of light
•Electrons accelerated in this relativistic plasma wave

The EPOCH Particle In Cell code



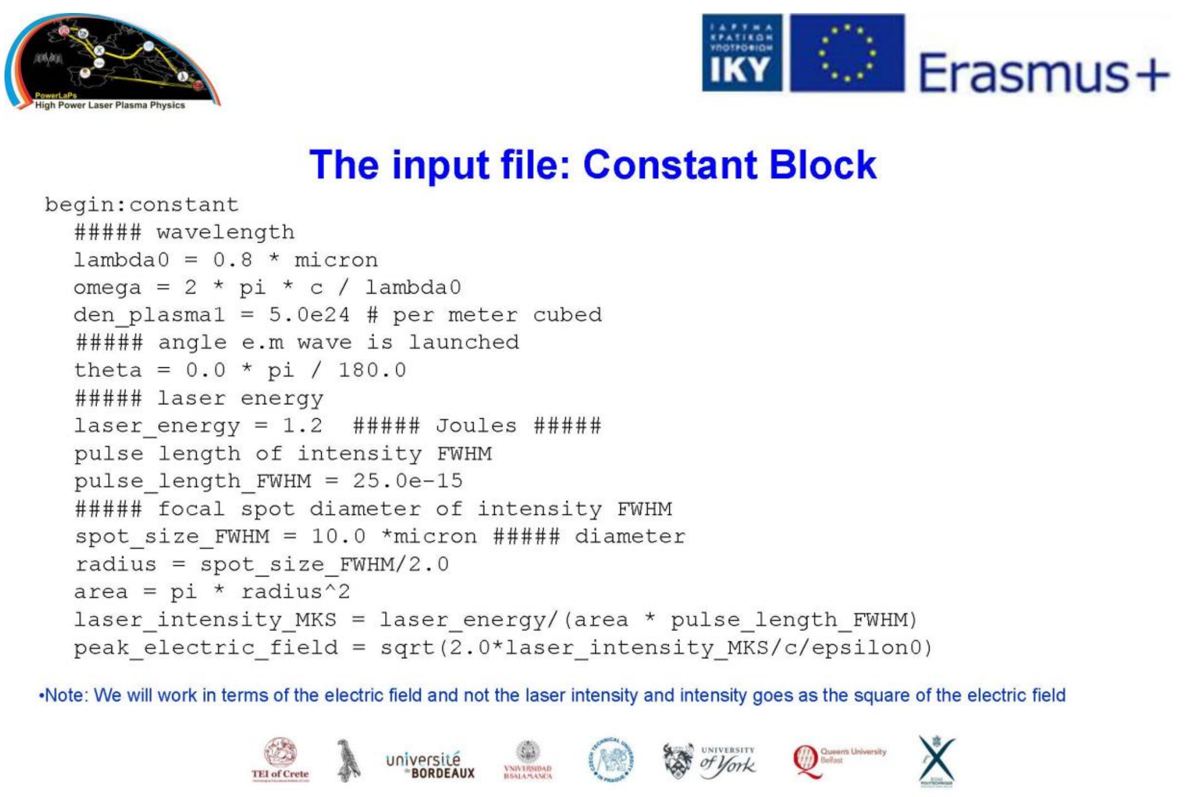
The *EPOCH* Project

A freely available EM PIC code

Principle Investigators	Senior Developers
Prof. A. R. Bell (Oxford)	Keith Bennett (Warwick)
Prof. R. G. Evans (Imperial)	Chris Brady (Warwick)
Prof. T. D. Arber (Warwick)	Holger Schmidz (Imperial)
	Chris Ridgers (Oxford)

Based on core algorithm from PSC by Hartmut Ruhl

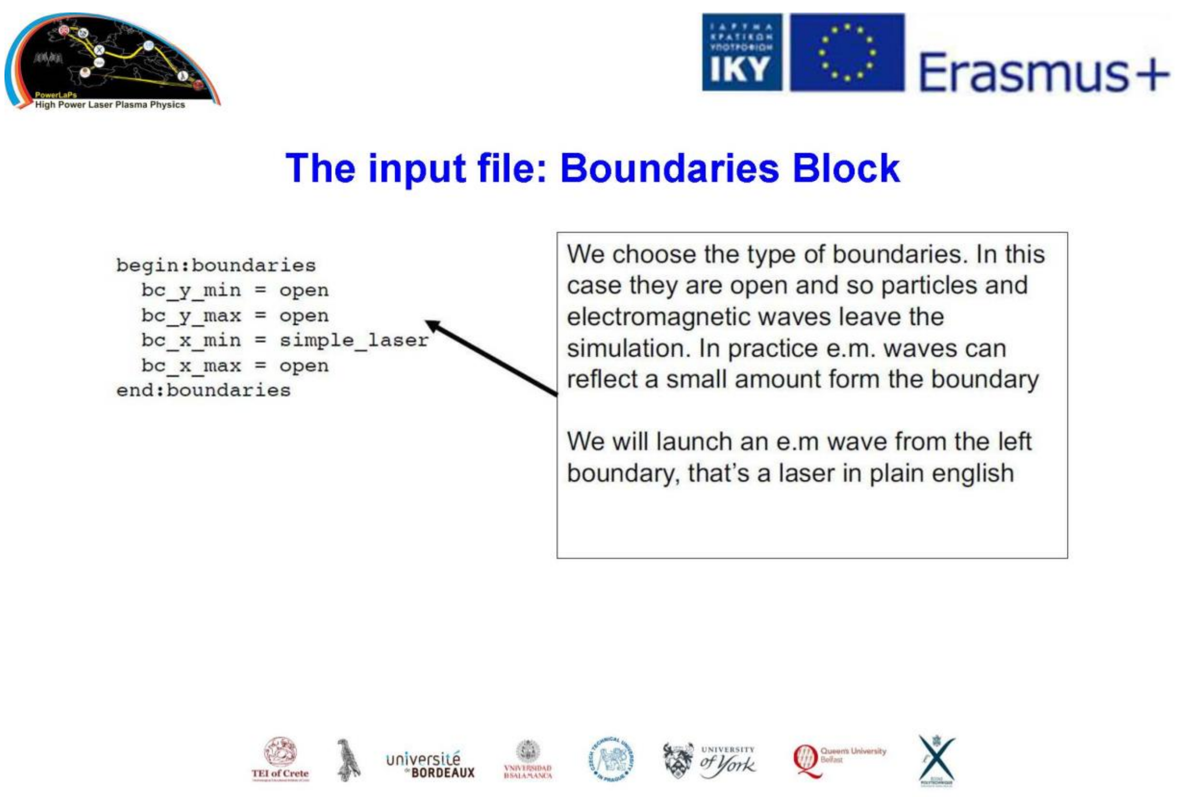
- We will use the EPOCH code in this session
- If you want to continue to use it afterwards then go to the EPOCH website and register as a user and download it
- We will just use the code….no explanation of how it works!
- There are many things to understand about PIC codes but we won’t discuss them here



The input file: Constant Block

```
begin:constant
  ##### wavelength
  lambda0 = 0.8 * micron
  omega = 2 * pi * c / lambda0
  den_plasma1 = 5.0e24 # per meter cubed
  ##### angle e.m wave is launched
  theta = 0.0 * pi / 180.0
  ##### laser energy
  laser_energy = 1.2  ##### Joules #####
  pulse length of intensity FWHM
  pulse_length_FWHM = 25.0e-15
  ##### focal spot diameter of intensity FWHM
  spot_size_FWHM = 10.0 *micron ##### diameter
  radius = spot_size_FWHM/2.0
  area = pi * radius^2
  laser_intensity_MKS = laser_energy/(area * pulse_length_FWHM)
  peak_electric_field = sqrt(2.0*laser_intensity_MKS/c/epsilon0)
```

•Note: We will work in terms of the electric field and not the laser intensity and intensity goes as the square of the electric field

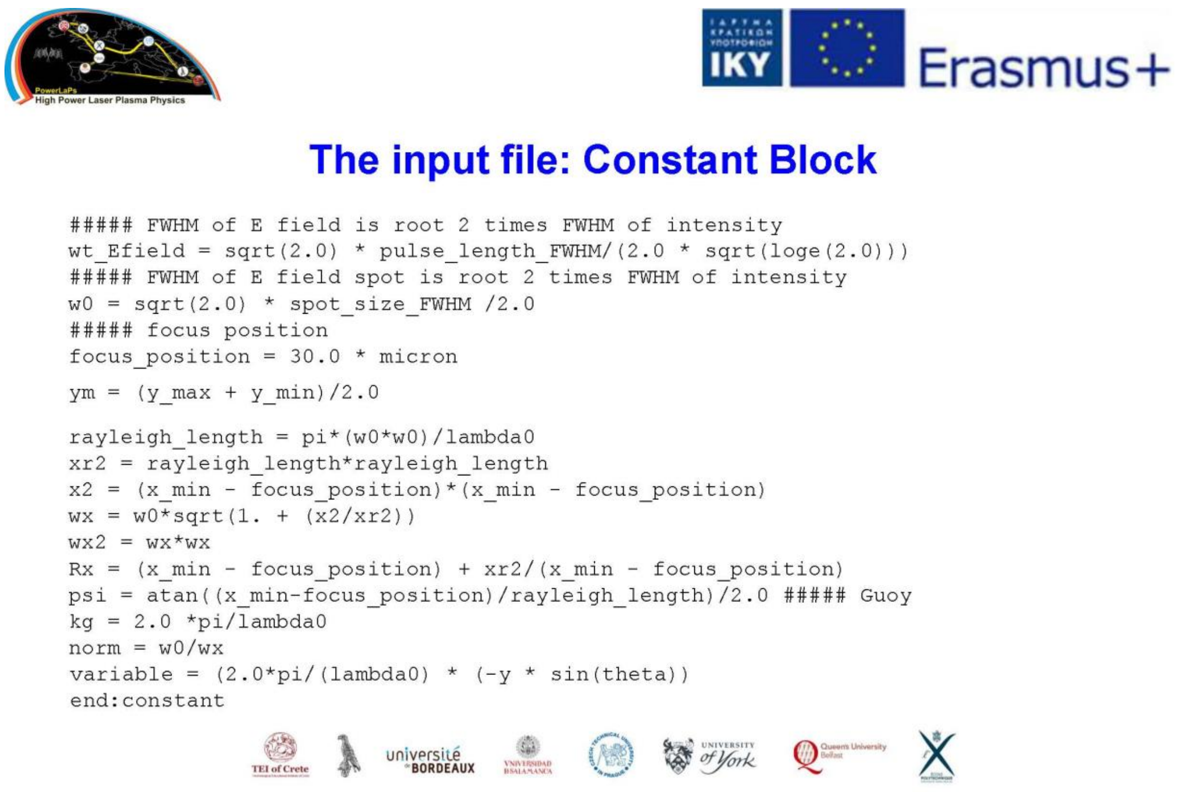


The input file: Boundaries Block

```
begin:boundaries
  bc_y_min = open
  bc_y_max = open
  bc_x_min = simple_laser
  bc_x_max = open
end:boundaries
```

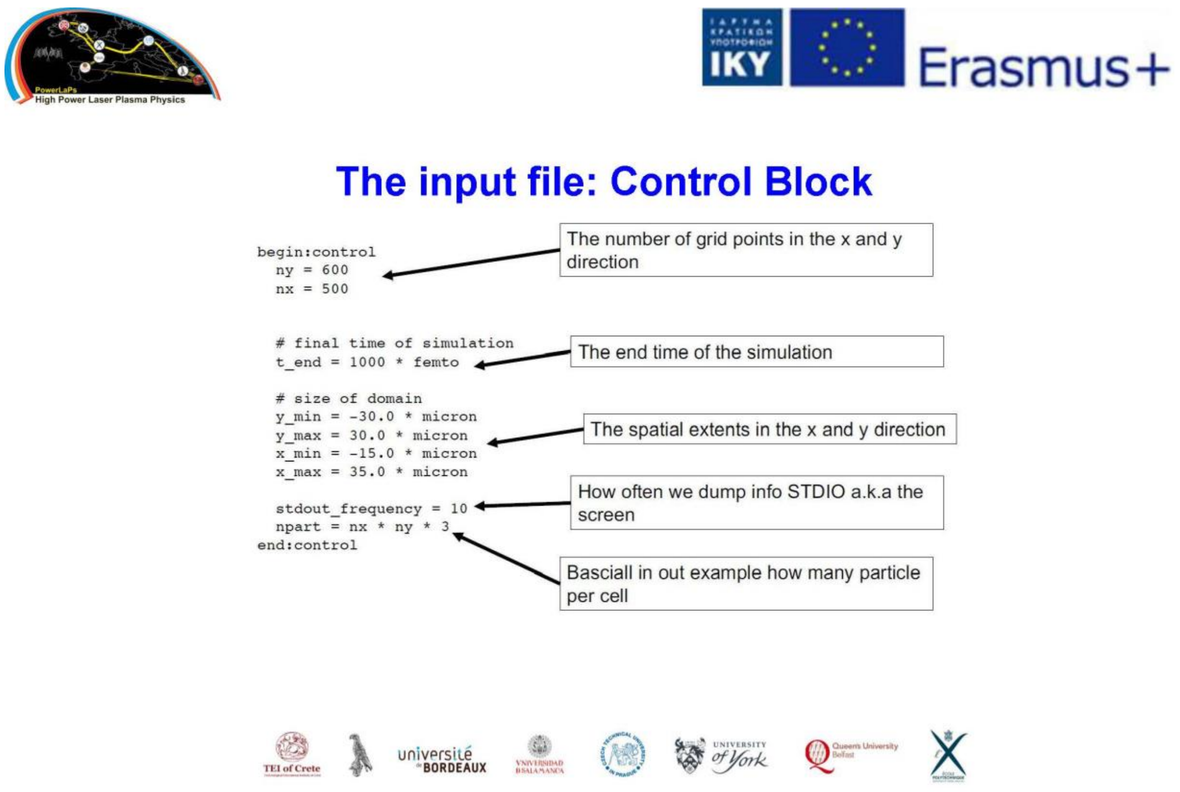
We choose the type of boundaries. In this case they are open and so particles and electromagnetic waves leave the simulation. In practice e.m. waves can reflect a small amount form the boundary

We will launch an e.m wave from the left boundary, that's a laser in plain english



The input file: Constant Block

```
##### FWHM of E field is root 2 times FWHM of intensity
wt_Efield = sqrt(2.0) * pulse_length_FWHM/(2.0 * sqrt(loge(2.0)))
##### FWHM of E field spot is root 2 times FWHM of intensity
w0 = sqrt(2.0) * spot_size_FWHM /2.0
##### focus position
focus_position = 30.0 * micron
ym = (y_max + y_min)/2.0

rayleigh_length = pi*(w0*w0)/lambda0
xr2 = rayleigh_length*rayleigh_length
x2 = (x_min - focus_position)*(x_min - focus_position)
wx = w0*sqrt(1. + (x2/xr2))
wx2 = wx*wx
Rx = (x_min - focus_position) + xr2/(x_min - focus_position)
psi = atan((x_min-focus_position)/rayleigh_length)/2.0 ##### Guoy
kg = 2.0 *pi/lambda0
norm = w0/wx
variable = (2.0*pi/(lambda0) * (-y * sin(theta))
end:constant
```



The input file: Control Block

```
begin:control
  ny = 600
  nx = 500

  # final time of simulation
  t_end = 1000 * femto

  # size of domain
  y_min = -30.0 * micron
  y_max = 30.0 * micron
  x_min = -15.0 * micron
  x_max = 35.0 * micron

  stdout_frequency = 10
  npart = nx * ny * 3
end:control
```

The number of grid points in the x and y direction

The end time of the simulation

The spatial extents in the x and y direction

How often we dump info STDIO a.k.a the screen

Basciall in out example how many particle per cell

Gaussian Beam

$$\mathbf{E}(r,z) = E_0\,\hat{x}\,\frac{w_0}{w(z)}\exp\left(\frac{-r^2}{w(z)^2}\right)\exp\left(-i\left(kz + k\frac{r^2}{2R(z)} - \psi(z)\right)\right)\ ,$$

where[1]

r is the radial distance from the center axis of the beam,

z is the axial distance from the beam's focus (or "waist"),

i is the imaginary unit,

$k = 2\pi/\lambda$ is the wave number (in radians per meter) for a wavelength λ,

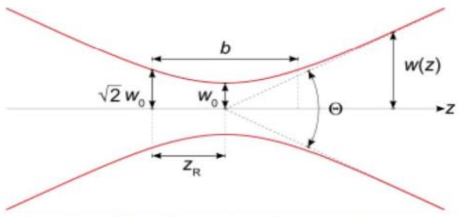
$E_0 = E(0,0)$, the electric field amplitude (and phase) at the origin at time 0,

$w(z)$ is the radius at which the field amplitudes fall to 1/*e* of their axial values, at the plane *z* along the beam,

$w_0 = w(0)$ is the waist size,

$R(z)$ is the radius of curvature of the beam's wavefronts at *z*, and

$\psi(z)$ is the Gouy phase at *z*, an extra phase term beyond that attributable to the phase velocity of light.



At a position *z* along the beam (measured from the focus), the spot size parameter *w* is given by[1]

$$w(z) = w_0\sqrt{1 + \left(\frac{z}{z_\mathrm{R}}\right)^2}\ .$$

where[1]

$$z_\mathrm{R} = \frac{\pi w_0^2}{\lambda}$$

is called the Rayleigh range as further discussed below.

The spot size *w*(*z*), at any position *z* along the beam, is related to the full width at half maximum (FWHM) at that position according to:[5]

$$w(z) = \frac{\mathrm{FWHM}}{\sqrt{2\ln 2}}.$$

The input file: Laser Block

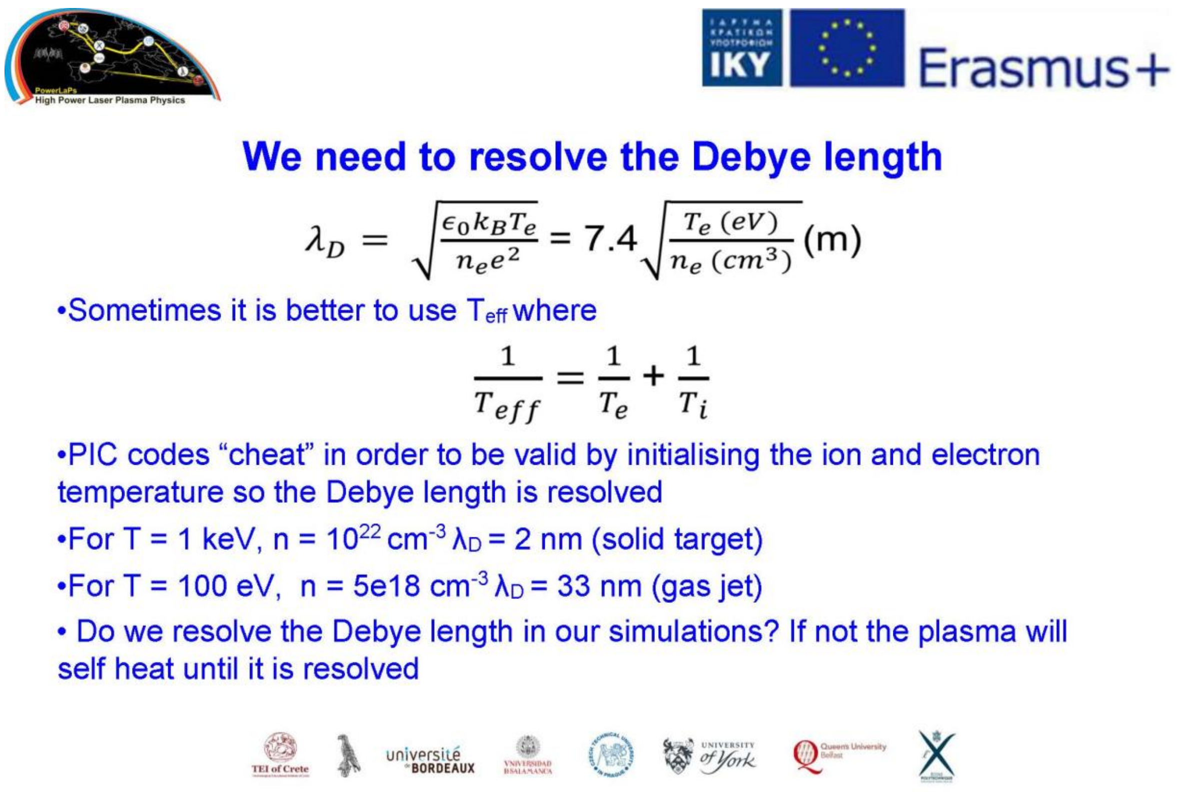
```
begin:laser
  boundary = x_min
  amp = peak_electric_field *norm
  lambda = lambda0 * cos(theta)
  ##### temporal gauss profile
  t_profile = gauss(time,2*wt_Efield,wt_Efield)
  ##### spatial gauss profile
  profile = gauss(y,0,(2*wx/1.6651))
  ##### phase
  phase = (-1)*((kg * 0.5 * (y-ym)^2 / Rx) + (kg * (x_min -
focus_position)) - psi) + variable

  pol_angle = (-1) * pi / 2.0

end:laser
```

- Polarisation angle in this case is in the z direction, out of the plane of the simulation. For `pol_angle =0` polarisation is in the y direction
- We have to set the phase, profile, polarization, time and amplitude of the e.m. wave

The input file: Species Block

```
begin:species
  name = helium2plus
  charge = 2.0
  mass = 1836.2 * 4.0
  frac = 0.333
  density = den_plasma1 /2.0
  temp_ev = 100.0
end:species

begin:species
  name = electron
  charge = -1.0
  mass = 1.0
  frac = 0.666
  density = den_plasma1
  temp_ev = 100.0
end:species
```

- We assume a fully ionised helium plasma with an electron density of 5e18 cm^{-3} uniformly over the grid
- We arbitrarily initialise the plasma temperature



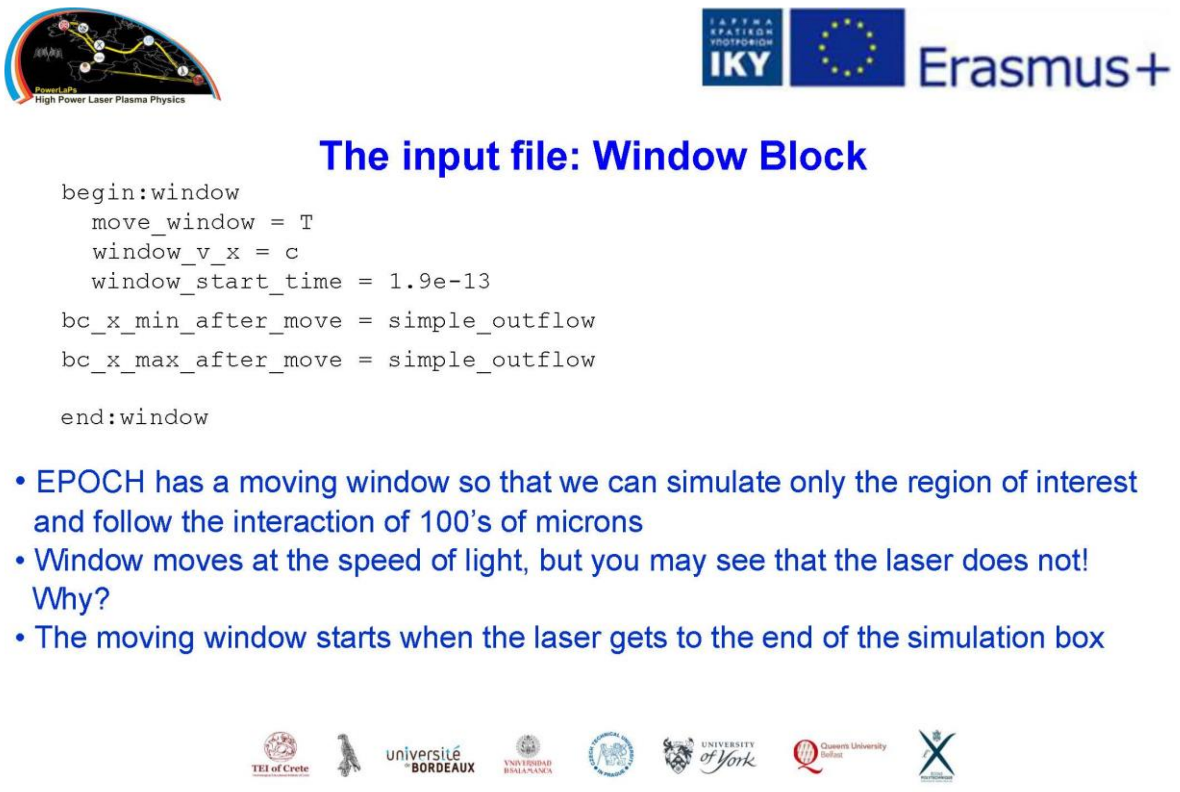
We need to resolve the Debye length

$$\lambda_D = \sqrt{\frac{\epsilon_0 k_B T_e}{n_e e^2}} = 7.4\sqrt{\frac{T_e\,(eV)}{n_e\,(cm^3)}}\,(\mathrm{m})$$

- Sometimes it is better to use T_{eff} where

$$\frac{1}{T_{eff}} = \frac{1}{T_e} + \frac{1}{T_i}$$

- PIC codes “cheat” in order to be valid by initialising the ion and electron temperature so the Debye length is resolved
- For T = 1 keV, n = 10^{22} cm^{-3} λ_D = 2 nm (solid target)
- For T = 100 eV, n = 5e18 cm^{-3} λ_D = 33 nm (gas jet)
- Do we resolve the Debye length in our simulations? If not the plasma will self heat until it is resolved



The input file: Window Block

```
begin:window
  move_window = T
  window_v_x = c
  window_start_time = 1.9e-13
bc_x_min_after_move = simple_outflow
bc_x_max_after_move = simple_outflow

end:window
```

- EPOCH has a moving window so that we can simulate only the region of interest and follow the interaction of 100’s of microns
- Window moves at the speed of light, but you may see that the laser does not! Why?
- The moving window starts when the laser gets to the end of the simulation box

We determine the minimum time step

- In two dimensions we define a length d based on the grid spacings Δx and Δy

$$d = \frac{1}{\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}}}$$

- The time step is given Δt

$$\Delta t = \frac{C \times d}{c}$$

- An em wave or particle should not move more than 1 grid spacing per time step: Courant–Friedrichs–Lewy (CFL) condition
- In EPOCH C =0.95 by default
- For em simulations we must resolve gyration of electrons too
- $\Delta t < 0.1\ \omega_{pe}$
- EPOCH calculates time step for us

The input file: Output Block

```
begin:output
  # number of timesteps between output dumps
  dt_snapshot =  25.0 * lambda0 /(2.0*c)
  # Number of dt_snapshot between full dumps
  #full_dump_every = 5
  #restart_dump_every = 5
  force_final_to_be_restartable = T
  # Properties at particle positions
  #particles = full
  #particle_weight = full #
  Properties on grid
  ex = always
  ez = always
  number_density = always + species
  distribution_functions = always
end:output
```

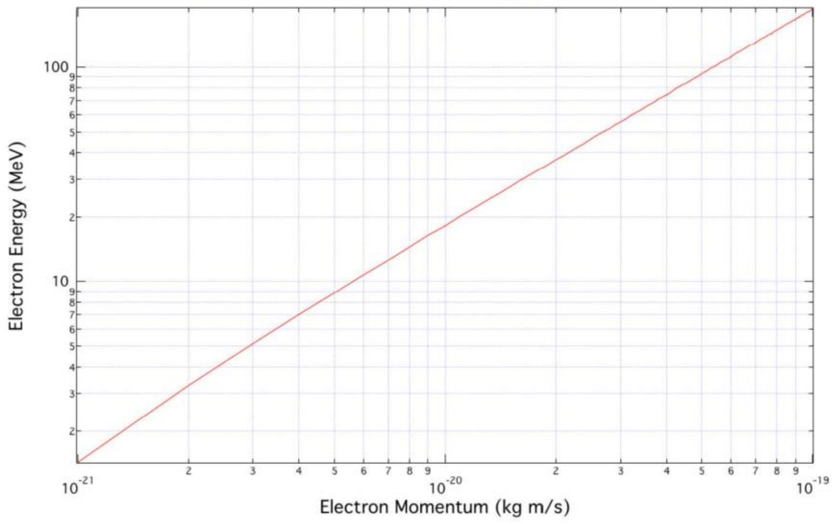
- EPOCH has many diagnostic outputs using its own SDF format
- A file is dumped at specified intervals including the electron density, Ex, Ez and user specified distribution functions

The input file: Output Block

```
begin:dist_fn
  name = x_px
  ndims = 2
  direction1 = dir_x
  direction2 = dir_px
  # range is ignored for spatial
coordinates
  range1 = (1, 1)
  range2 = (-2.0e-21, 5.0e-20)
  # resolution is ignored for spatial
coordinates
  resolution1 = 1
  resolution2 = 100
  include_species:electron
end:dist_fn
```

- Here we tell EPOCH to output the phase space of momentum in the x direction as function of x

Relationship between energy and momentum



How to run EPOCH

- We will use 2D executable of EPOCH located in `pathtoepoch/epoch2d/bin` and called `epoch2d`
- First, we change directory to the epoch2d directory where the folders containing our input files are located

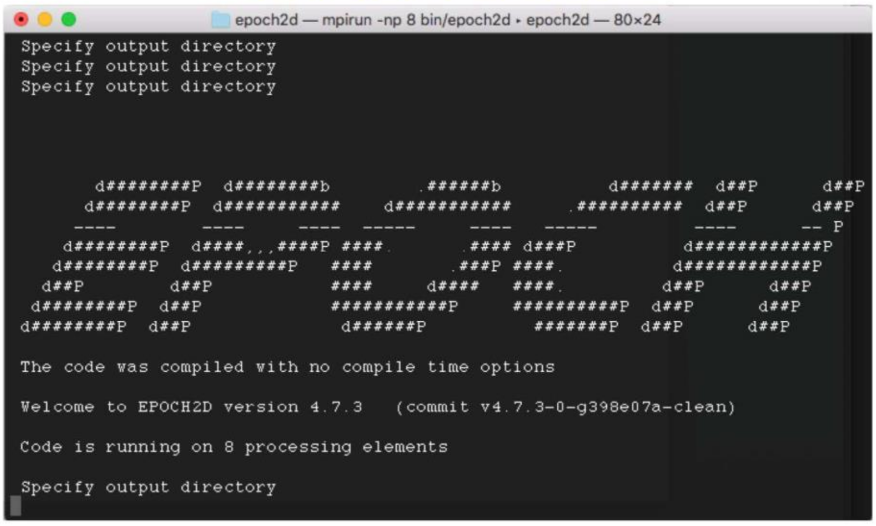
```
cd pathtoepoch/epoch2d
```

- EPOCH is compiled to be used with OpenMPI so to run EPOCH in parallel we type

```
mpirun -np 8 bin/epoch2d
```

- You will be prompted to type the name of the directory that contains the input file
- Choose number of processors with `-np 8` argument

How to run EPOCH

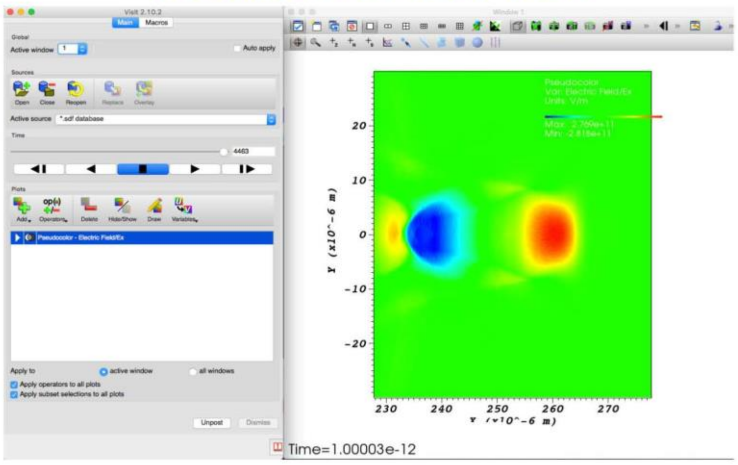


```
Specify output directory
Specify output directory
Specify output directory



       d########P  d########b        .######b          d#######  d##P      d##P
      d########P  d###########    d###########     .##########  d##P      d##P
     ----        ----     ----  -----     ----   -----         ----      -- P
    d########P  d####,,,####P ####.      .#### d###P          d############P
   d########P  d#########P   ####       .###P ####.          d############P
  d##P        d##P          ####       d####   ####.         d##P      d##P
 d########P  d##P           ###########P     ##########P   d##P      d##P
d########P  d##P            d#######P        #######P    d##P      d##P

The code was compiled with no compile time options

Welcome to EPOCH2D version 4.7.3   (commit v4.7.3-0-g398e07a-clean)

Code is running on 8 processing elements

Specify output directory
```

What to do

- First let's run the input file `laser`
  - Change the spot size
  - Change the angle
- This runs quick as there are no particles in the simulation
- Now run the input file `gasjet`
- This will take a while depending on fast your computer is
- Laser wakefield accelerator is sensitive to the gas density, spot size and pulse length
- A laser wakefield is optimised when laser pulse length ($c\tau$) is half the plasma period (λ_p) where $\lambda_p = 2\pi c/\omega_p$ and wp = $\sqrt{(n_e e^2/m_e \varepsilon_0)}$
- So for our 25 fs pulse, we require n_e = 5e18 cm^{-3}
  - Change the electron plasma density
  - Change the focal spot size
  - See how the maximum electron energy is changed after 1 ps

Visualisation with VisIt



Type `visit` at command line to start Visit

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

High Power Laser Matter Interactions/High Energy Density Physics - Theory and Experiments

EXP 5: Target Normal Sheath Acceleration (TNSA) - PIC simulations

G. Tazes, A. Grigoriadis, G. Andrianaki.

HELLENIC
MEDITERRANEAN
UNIVERSITY

5.1 Simulating Particles In Cells

The particle-in-cell (PIC) method is used to solve a certain class of partial deferential equations (PDEs) in order to model physical systems behavior. Often used is a plasma physics simulation.

In this method, collections of physical particles are represented using a smaller number of macroparticles. The E, B fields generated by the motion of these macroparticles are calculated using a Finite Differences technique on an underlying grid of fixed spatial resolution. The forces on the macroparticles due to the calculated fields are then used to update the macroparticle velocities, and these velocities are then used to update their positions.

In PIC method, the individual particles (or fluid elements) are tracked in a continuous phase space, whereas moments of the distribution such as densities and currents are computed concurrently on stationary mesh points

Scope of this presentation is to introduce and familiarize the participants with the opensource advance relativistic EM MPI parallelized code EPOCH, developed under the Extendable PIC Open Collaboration project in UK. Likewise, perform of numerical simulations of an intense, fs laser pulse interacting with solid density plasma.

EPOCH initially calculates the electric and magnetic field values on the grid, by solving the Maxwell equations. At the next step the velocity of the particles is calculated and their new position on the grid is updated. Finally, the current density from the particle flux, through the grid is calculated, which in turn affects the electric and magnetic fields on the grid. After a full iteration these steps are repeated, as it is schematically depicted in the flowchart of *figure 1*.

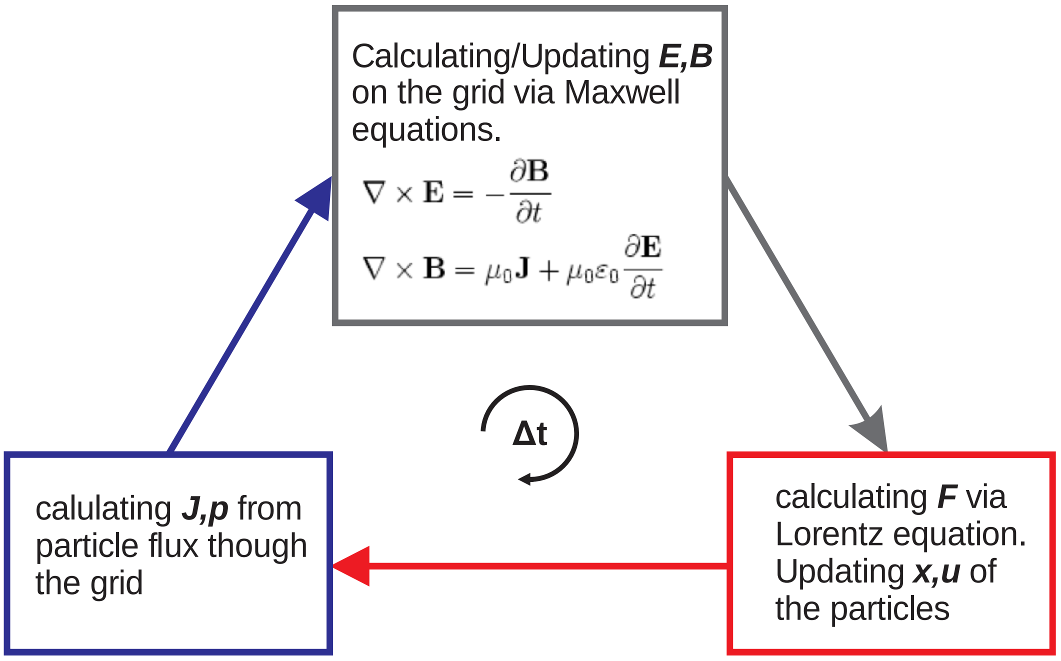

Calculating/Updating ***E,B*** on the grid via Maxwell equations.
$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$
$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}$
Δt
calulating ***J,p*** from particle flux though the grid
calculating ***F*** via Lorentz equation. Updating ***x,u*** of the particles

Figure 0-1: *The particle-in-cell scheme. The steps of one full iteration.*

EPOCH follows the Kinetic description of plasma that is based on a set of equations for (macroscopic) distribution function $f_s(x,p)$ of each plasma particle species together with Maxwell equations. The distribution function is a statistical description of a very large number of interacting particles. Each particle has its own position in the phase space (x,p), where x, y, z are the coordinates for all the degrees of freedom and p_x, p_y, p_z are the corresponding momentum components.

EPOCH solves the following set of equations (**Normalized Vlasov-Maxwell equations):**

- $\frac{\partial}{\partial t}E(x,t) = \nabla \times B(x,t) - J_T(x,t)$ (Ampere)
- $\frac{\partial}{\partial t}B(x,t) = -\nabla \times E(x,t)$ (Faraday)
- $\nabla \cdot E(x,t) = \rho_T(x,t)$ (Poisson)
- $\nabla \cdot B(x,t) = 0$ (Gauss)
- $\frac{\partial}{\partial t}f_n(x,v,t) + v\nabla_x f_n(x,v,t) - \frac{q_n}{m_n}(E + v \times B)\nabla_v f_n(x,v,t) = 0$ (Vlasov)
- $\rho_T(x,t) = \int [\sum_n q_n f_n(x,v,t)]\, dv$
- $J_T(x,t) = \int [\sum_n q_n f_n(x,v,t)]\, v dv$

During the pre-processing of a simulation, numerous parameters and limitations must be taken into account. The dimensions of the simulation are chosen. EPOCH allows for 1,2 and

3D simulations, while the more the dimensions used, the higher the computational demands. Then, the simulation solution domain (box size) and the total simulation duration are set, based on the interaction time of interest. The spatial resolution of the simulation comes from the chosen grid points that divide the simulation box into cells. The cell size should meet certain criteria, e.g.: if the cell size exceeds the Debye length of the simulated plasma, nonphysical electric fields will arise on the grid, leading to rapid increase of the particle energy. This issue is known as numerical heating. EPOCH uses macroparticles which represent many real particles, acting collectively. This division of actual particles into less macroparticles relaxes the computational demands of a simulation, although a very low number of macroparticles will lead to unnatural heating of the plasma. The electron temperature should be also taken into consideration. For higher electron temperatures the Debye length increases in size and this relaxes the spatial discretization requirements in order to resolve it although one should be careful not to excide the actual temperatures the electrons would acquire from a real laser prepulse. Additionally, unnaturally high electron temperature will lead to unreal plasma expansion rate. If λ_D is resolved, the time step must be

$$\Delta t \leq \frac{1}{c\sqrt{\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)}}$$

to preserve the numerical stability.

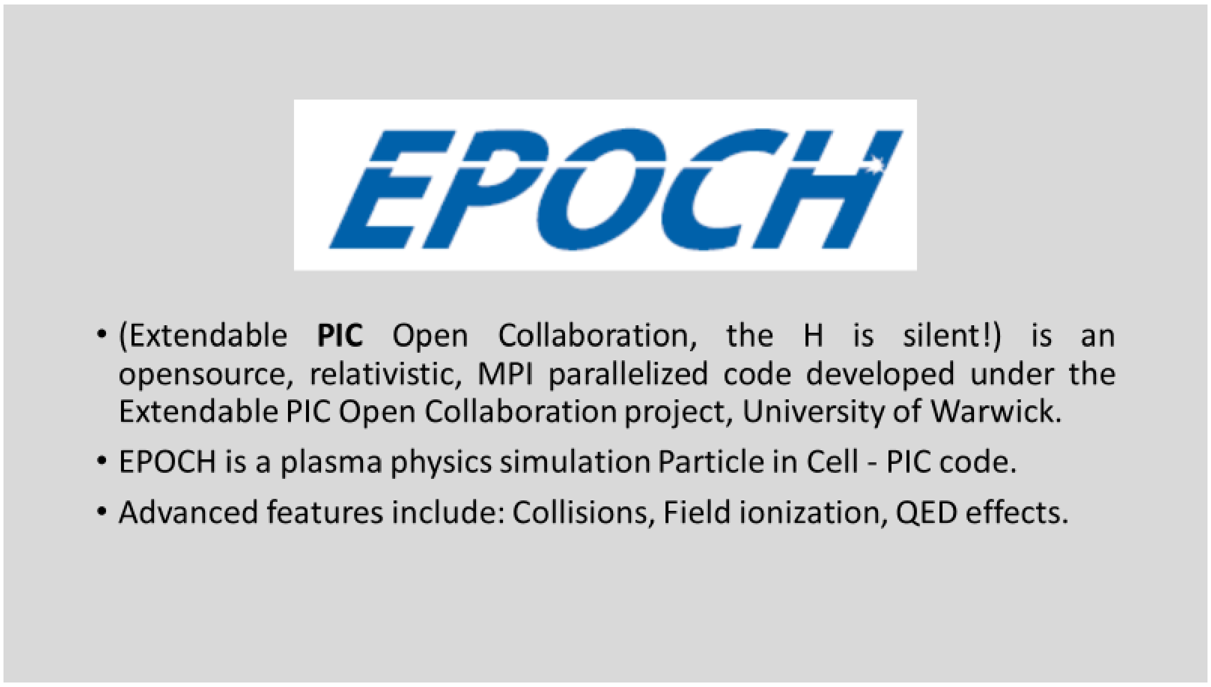
5.2 EPOCH for TNSA

During the hands-on course, the participants initially installed a Virtual box machine running on Ubuntu 18.04 with pre-installed all the required software packages. During the lecture, step by step instruction were given in order to run the provided TNSA testcase on EPOCH 2D. Followed by visualization of the output e.g. particle density, laser electric field and particle momentum using the **Open Source**, interactive, scalable, visualization, animation and analysis tool VisIt. The first simulation was only the laser pulse without the target, this was done by having pre-set the number of sudoparticles per cell equal zero. Afterwards, it was required to modify the input file of the TNSA testcase in order to familiarize the participants with the setting of a 2D testcase. The training simulations were perform using low number of particles per cell and a low-resolution grid in order not to be extremely time consuming and able to be performed in any laptop using at least 2 parallel processors. The required modifications of the input file were to change the number of grid point, set the number of particles per species per cell from 0 to 1 and set the simulation duration to 100fs. This run was estimated to require approximately 30 minutes using 2 CPU cores. After the visualization of the outputs, it was required one more modification of the testcase, this time it was asked to change the plasma density, pulse duration, energy, focal spot diameter and aluminum ionization degree, based the participants preferences. This was given as a homework to the participants along with an extra task to create on new block of partially ionized carbon.

5.3 References

[1] Kumar, A., & Mathew, V. (2018, May). Computational study of proton acceleration from the laser irradiated metal substrate. In AIP Conference Proceedings (Vol. 1953, No. 1, p. 140011). AIP Publishing.
[2] Ferri, J., Siminos, E., & Fülöp, T. (2019). Enhanced target normal sheath acceleration using colliding laser pulses. Communications Physics, 2(1), 40.
[3] Wang, D., Shou, Y., Wang, P., Liu, J., Li, C., Gong, Z., ... & Yan, X. (2018). Enhanced proton acceleration from an ultrathin target irradiated by laser pulses with plateau ASE. Scientific reports, 8(1), 2536
[4] Silva, L. O., Marti, M., Davies, J. R., Fonseca, R. A., Ren, C., Tsung, F. S., & Mori, W. B. (2004). Proton shock acceleration in laser-plasma interactions. Physical Review Letters, 92(1), 015002.
[5] Pukhov, A. (2001). Three-dimensional simulations of ion acceleration from a foil irradiated by a short-pulse laser. Physical review letters, 86(16), 3562.
[6] Silva, L. O., Marti, M., Davies, J. R., Fonseca, R. A., Ren, C., Tsung, F. S., & Mori, W. B. (2004). Proton shock acceleration in laser-plasma interactions. Physical Review Letters, 92(1), 015002.
[7] Wang, P., Cao, Z., Gao, Y., Geng, Y., Kong, D., Lin, C., ... & Pan, Z. (2019, June). A Simple Way to Introduce an Ajustable Femtosecond Pre-Pulse to Enhance Laser-Driven Proton Acceleration. In 10th Int. Partile Accelerator Conf.(IPAC'19), Melbourne, Australia, 19-24 May 2019 (pp. 3686-3688). JACOW Publishing, Geneva, Switzerland.
[8] Arber, T. D., Bennett, K., Brady, C. S., Lawrence-Douglas, A., Ramsay, M. G., Sircombe, N. J., Ridgers, C. P. (2015). Contemporary particle-in-cell approach to laser-plasma modelling. Plasma Physics and Controlled Fusion, 57(11), 113001.
[9] Turrell, A. E., Sherlock, M., & Rose, S. J. (2015). Ultrafast collisional ion heating by electrostatic shocks. Nature communications, 6, 8905.

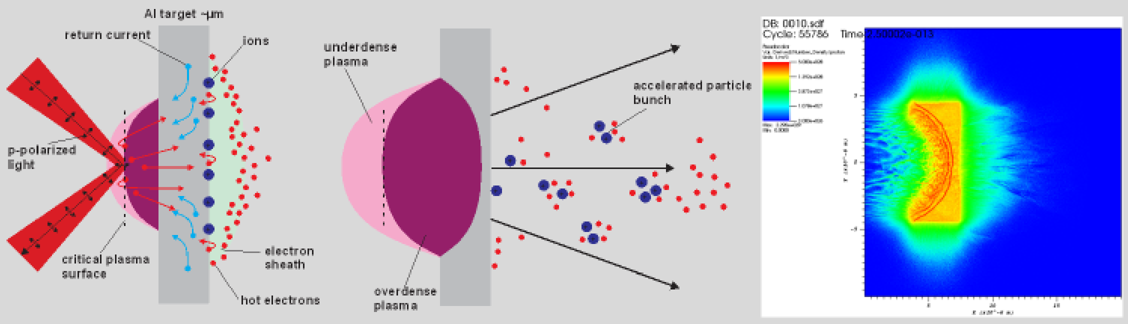
5.4 Hands on TNSA with PIC



EPOCH

- (Extendable **PIC** Open Collaboration, the H is silent!) is an opensource, relativistic, MPI parallelized code developed under the Extendable PIC Open Collaboration project, University of Warwick.
- EPOCH is a plasma physics simulation Particle in Cell - PIC code.
- Advanced features include: Collisions, Field ionization, QED effects.

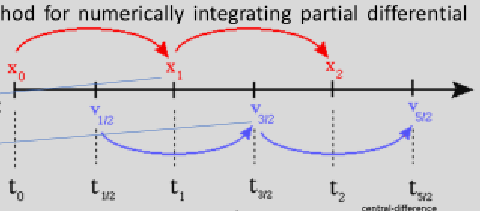
Target Normal Sheath Acceleration



TNSA schematic representation: The peak intensity pulse interacts with the formed preplasma. Hot electrons from the critical plasma surface accelerate through the overdense plasma and the foil target while cold electrons rush to shield the abundant positive charge. An electrons sheath is formed to the rear of the target. Ions of the contamination layer are accelerated as the electrons drag them to the vacuum behind the target.

Finite-Difference Time-Domain – FDTD - Leapfrog

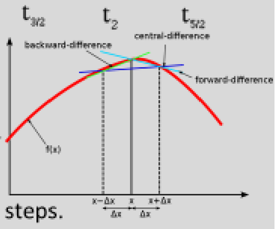
Finite-Difference Time-Domain (FDTD) is a grid-based numerical technique used for modeling computational electrodynamics. The time-dependent finite-difference equations are discretized using central-difference approximations and solved using the **leapfrog scheme**. Leapfrog integration is a method for numerically integrating partial differential equations.



In leapfrog, the equations for updating position and velocity are:

$$a_i = F(x) \qquad x_{i+1} = x_i + u_{i+\frac{1}{2}}\Delta t \qquad u_{i+\frac{1}{2}} = u_{i-\frac{1}{2}} + a_i\Delta t$$

Where x at steps *i*, *u* at steps *i+1/2*, *Δt is the time step.*



These equations can be expressed in a form that gives velocity at integer steps as well.

$$x_{1+1} = x_i + u_i\Delta t + \frac{1}{2}a_i\Delta t^2 \qquad u_{i+1} = u_i + \frac{1}{2}(a_i + a_{i+1})\Delta t$$

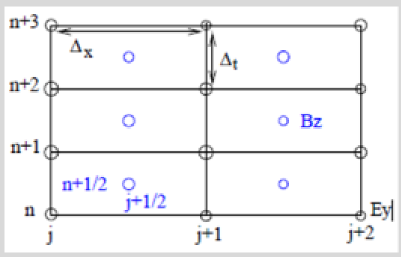
Field solver scheme

These equations can be expressed in a form that gives velocity at integer and half-time steps.

$$u_{i+\frac{1}{2}} = u_i + a_i\frac{1}{2}\Delta t \qquad x_{1+1} = x_i + u_{i+\frac{1}{2}}\Delta t \qquad u_i = u_{i+\frac{1}{2}} + a_{i+1}\frac{1}{2}\Delta t$$

Particle pusher 'Boris'

Ampere and Faraday coupling



The **black grid** has nodes at integer values of (x, t) = j , n , it is defined at full space and time steps.

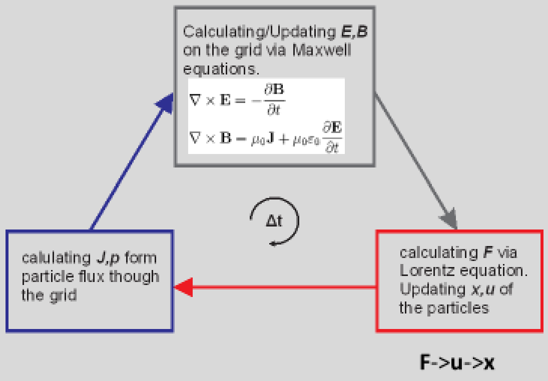
The **blue grid** is defined at (x, t) = (j + 1/2, n + 1/2); it is shifted by x/2 and t/2.

Replace continuous fields **B**(**x**, **t**), **E**(**x**, t) and Maxwell's equations by their discretized counterparts.

$$\left.\begin{aligned}\frac{\partial}{\partial t}E_y(x,t) &= -\frac{\partial}{\partial x}B_z(x,t)\\ \frac{\partial}{\partial t}B_z(x,t) &= -\frac{\partial}{\partial x}E_y(x,t)\end{aligned}\right\} \quad \begin{aligned}\frac{\boldsymbol{E}_y(j,n+1)-\boldsymbol{E}_y(j,n)}{\Delta t} &= -\frac{\boldsymbol{B}_z\left(j+\frac{1}{2},n+\frac{1}{2}\right)-\boldsymbol{B}_z\left(j-\frac{1}{2},n+\frac{1}{2}\right)}{\Delta x}\\ \frac{\boldsymbol{B}_z(j+1/2,n+1/2)-\boldsymbol{B}_z(j+1/2,n-1/2)}{\Delta t} &= -\frac{\boldsymbol{E}_y(j,n)-\boldsymbol{E}_y(j+1,n)}{\Delta x}\end{aligned}$$

Particle In Cell – PIC scheme

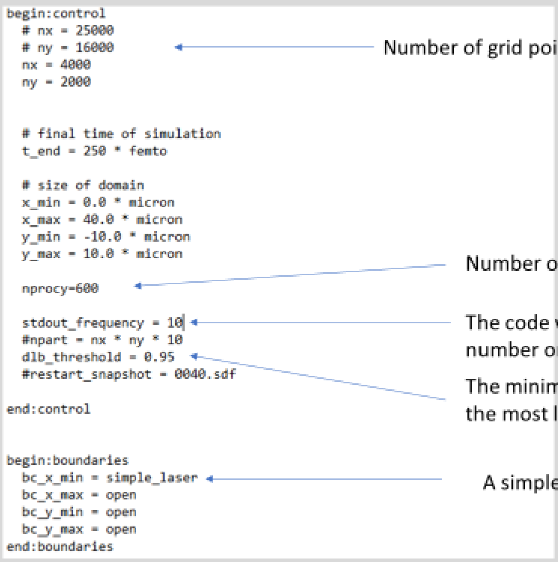
Kinetic description is based on a set of equations for (macroscopic) distribution function $f_s(x,p)$ of each plasma particle species s together with Maxwell equations. The distribution function is a statistical description of a very large number of interacting particles. Each particle has its own position in the phase space (x,p), where x are the coordinates for all the degrees of freedom and p are the corresponding momentum components.



Calculating/Updating *E,B* on the grid via Maxwell equations.
$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$
$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}$
Δt
calulating *J,p* form particle flux though the grid
calculating *F* via Lorentz equation. Updating *x,u* of the particles
F->u->x

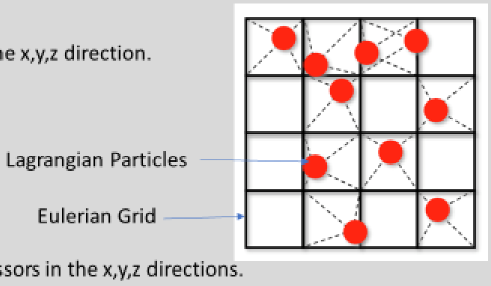
1. Initialize plasma phase space distribution:
– Place particles in space according to density
– Initialize velocities with random numbers
2. Initialize E and B fields
3. From E,B fields calculate acceleration
4. Multiply acceleration with time step
–> Velocity increment
5. Multiply velocity with time step
–> Position increment
6. From new positions and velocities:
Calculate new E,B

EPOCH Input.deck File Blocks/Control Block



```
begin:control
  # nx = 25000
  # ny = 16000
  nx = 4000
  ny = 2000


  # final time of simulation
  t_end = 250 * femto

  # size of domain
  x_min = 0.0 * micron
  x_max = 40.0 * micron
  y_min = -10.0 * micron
  y_max = 10.0 * micron

  nprocy=600

  stdout_frequency = 10
  #npart = nx * ny * 10
  dlb_threshold = 0.95
  #restart_snapshot = 0040.sdf

end:control


begin:boundaries
  bc_x_min = simple_laser
  bc_x_max = open
  bc_y_min = open
  bc_y_max = open
end:boundaries
```

Number of grid points in the x,y,z direction.



Number of processors in the x,y,z directions.

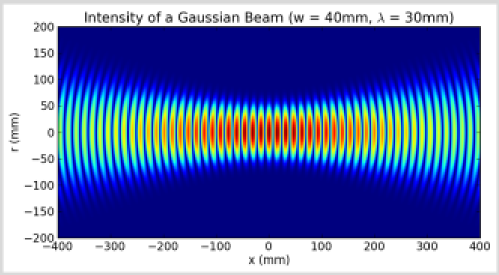
The code will print a one-line status message to stdout after every given number or timesteps.

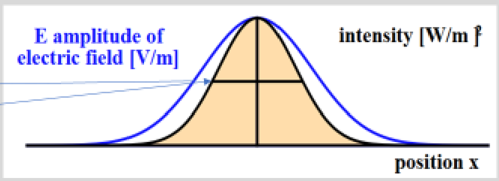
The minimum ratio of the load on the least loaded processor to that on the most loaded processor allowed before the code load balances.

A simple case of simulation domain boundaries block

EPOCH Constant block

```
begin:constant
  ##### wavelength
  lambda0 = 0.8 * micron
  omega = 2 * pi * c / lambda0
  den_plasma = 60.0 * critical(omega)
  ##### angle
  theta = 0.0 * pi / 180.0
  ##### laser energy
  laser_energy = 1.0 #0.144 ##### Joules
  ##### pulse length of intensity FWHM
  pulse_length_FWHM = 25.0e-15
  ##### focal spot diameter of intensity FWHM
  spot_size_FWHM = 3.0 *micron ##### diameter
  radius = spot_size_FWHM/2.0
  area = pi * radius^2
  laser_intensity_MKS = laser_energy/(area * pulse_length_FWHM)
  peak_electric_field = sqrt(2.0*laser_intensity_MKS/c/epsilon0)
  ##### FWHM of E field is root 2 times FWHM of intensity
  wt_Efield = sqrt(2.0) * pulse_length_FWHM/(2.0 * sqrt(loge(2.0)))
  ##### FWHM of E field spot is root 2 times FWHM of intensity
  w0 = sqrt(2.0) * spot_size_FWHM /2.0
  ##### focus position
  focus_position = 10.0 * micron
  ym = (y_max + y_min)/2.0
```





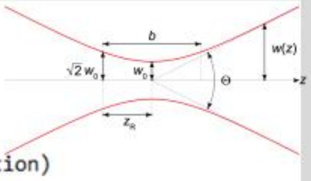
The radius of the beam $w(z)$, at any position z along the beam, is related to the full width at half maximum (FWHM) at that position according to:[5]

$$w(z) = \frac{\mathrm{FWHM}(z)}{\sqrt{2\ln 2}}$$

EPOCH Constant Block

$$z_R = \frac{\pi w_0^2}{\lambda}$$

```
 rayleigh_length = pi*(w0*w0)/lambda0
 xr2 = rayleigh_length*rayleigh_length
 x2 = (x_min - focus_position)*(x_min - focus_position)
 wx = w0*sqrt(1. + (x2/xr2))
 wx2 = wx*wx
 #Rx = (x_min - focus_position) + xr2/(x_min - focus_position)
 Rx = (x_min - focus_position) * (1+(xr2/x2))
 psi = atan((x_min-focus_position)/rayleigh_length)
 kg = 2.0 *pi/lambda0
 norm = w0/wx
 variable = (2.0*pi/(lambda0 * cos(theta)) * (-y * sin(theta))
end:constant
```



The radius of the beam at a distance z from the waist is

$$w(z) = w_0\sqrt{1+\left(\frac{z}{z_R}\right)^2}.$$

x2

xr2

wx

the *radius of curvature* as a function of position along the beam, given by

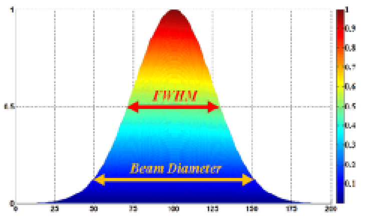
$$R(z) = z\left[1+\left(\frac{z_R}{z}\right)^2\right]$$

Rx

The *Gouy phase* of the beam at z is given by

psi $$\psi(z) = \arctan\left(\frac{z}{z_R}\right).$$

EPOCH Laser Block



```
begin:laser
  boundary = x_min
  amp = peak_electric_field
  lambda = lambda0 * cos(theta)
  ##### temporal gauss profile
  t_profile = gauss(time,2*wt_Efield,wt_Efield)
  ##### spatial gauss profile
  profile = gauss(y,0,(2*wx/1.6651))
  ##### phase
  phase = (-1)*((kg * 0.5 * (y-ym)^2 / Rx) + (kg * (x_min - focus_position)) - psi) + variable
  pol_angle = 0.0
end:laser
```

The Gaussian function has a 1/e^2 diameter 2w =1.7*FWHM.

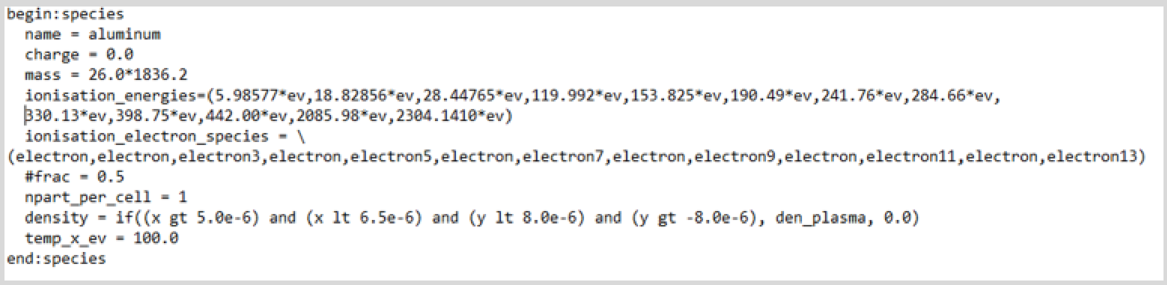
$$\left(kz + k\frac{r^2}{2R(z)} - \psi(z)\right)$$

For a Gaussian beam, the relationship between the 1/e^2 width and the full width at half maximum is $2w = \frac{\sqrt{2}\,\mathrm{FWHM}}{\sqrt{\ln 2}} = 1.699 \times \mathrm{FWHM}$, where $2w$ is the full width of the beam at 1/e^2.

EPOCH Particles Block

```
begin:species
  name = aluminum
  charge = 5.0
  mass = 26.0*1836.2
  #frac = 0.5
  npart_per_cell = 4
  density = if((x gt 10.0e-6) and (x lt 15.0e-6) and (y lt 10.0e-6) and (y gt -10.0e-6), den_plasma, 0.0)
  temp_x_ev = 100.0
end:species

begin:species
  name = proton
  charge = 1.0
  mass = 1836.2
  #frac = 0.5
  npart_per_cell = 4
  density = if((x gt 10.0e-6) and (x lt 10.1e-6) and (y lt 10.0e-6) and (y gt -10.0e-6), den_plasma/1000, 0.0)
  density = if((x gt 14.9e-6) and (x lt 15.0e-6) and (y lt 10.0e-6) and (y gt -10.0e-6), den_plasma/1000, density(proton))
  temp_x_ev = 100.0
end:species

begin:species
  name = electron
  charge = -1.0
  mass = 1.0
  npart_per_cell = 4
  #frac = 0.5
  density = if((x gt 10.0e-6) and (x lt 15.0e-6) and (y lt 10.0e-6) and (y gt -10.0e-6), den_plasma*5, 0.0)
  temp_x_ev = 1000.0
end:species
```

EPOCH Particles Ionization Block

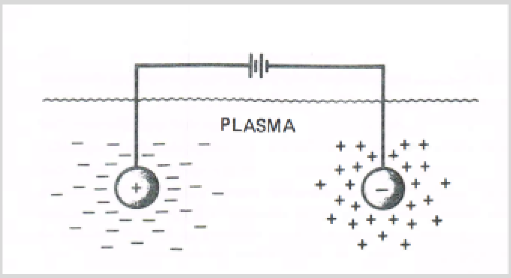


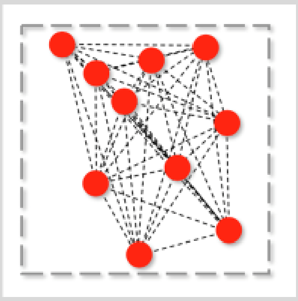
```
begin:species
  name = aluminum
  charge = 0.0
  mass = 26.0*1836.2
  ionisation_energies=(5.98577*ev,18.82856*ev,28.44765*ev,119.992*ev,153.825*ev,190.49*ev,241.76*ev,284.66*ev,
  330.13*ev,398.75*ev,442.00*ev,2085.98*ev,2304.1410*ev)
  ionisation_electron_species = \
(electron,electron,electron3,electron,electron5,electron,electron7,electron,electron9,electron,electron11,electron,electron13)
  #frac = 0.5
  npart_per_cell = 1
  density = if((x gt 5.0e-6) and (x lt 6.5e-6) and (y lt 8.0e-6) and (y gt -8.0e-6), den_plasma, 0.0)
  temp_x_ev = 100.0
end:species
```

Set field_ionization = T

- Multiphoton ionization F/T
- Tunneling ionization F/T
- Barrier Suppression Ionization F/T

We need to resolve the Debye Length





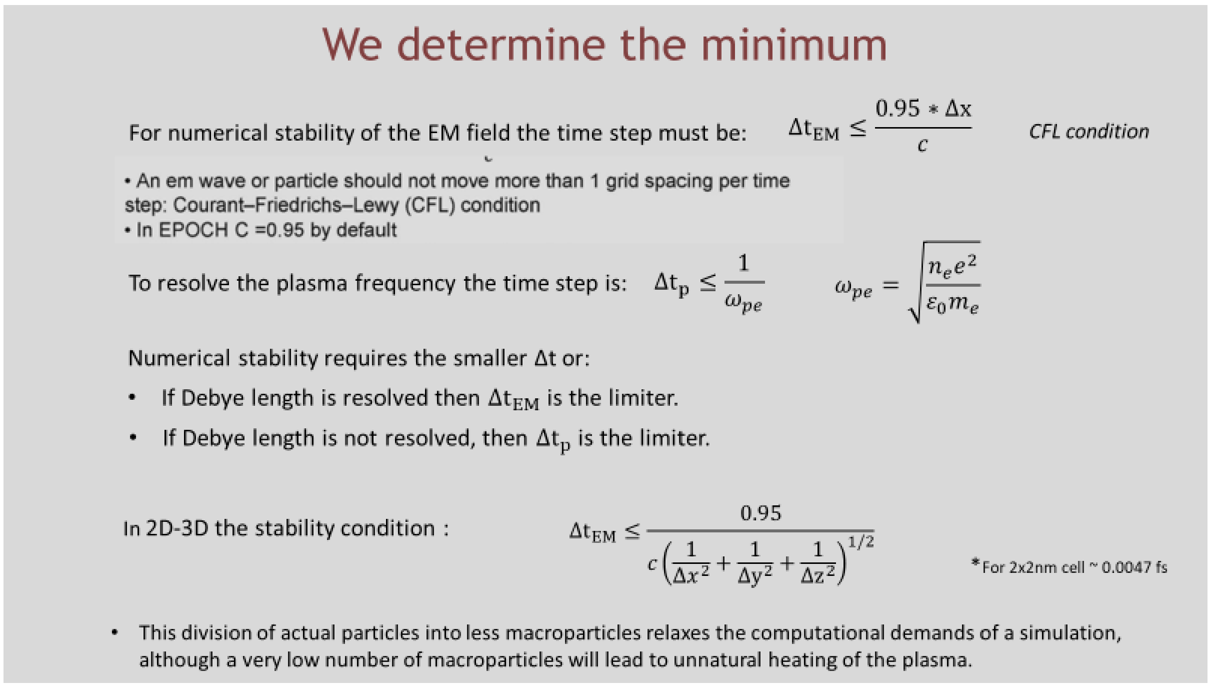
To calculating the Coulomb force between all particles, the calculation would take a prohibitively long time

•Over distances longer the Debye length, charged particles are unaware of the presence of other
•So we don't need to calculate the forces between all particles
•We calculate locally on a grid and we resolve the Debye length

PIC codes will exponentially heat the plasma until the Debye length is resolved

$$\lambda_D = \sqrt{\frac{\varepsilon_0 k_B T_e}{ne^2}}$$

*For $T_e \sim 1000\ eV$
$n \sim 1.7 * 10^{21}/cm^3$
$\lambda_D \sim 2\text{nm}$



We determine the minimum

For numerical stability of the EM field the time step must be: $\Delta t_{EM} \leq \dfrac{0.95 * \Delta x}{c}$ *CFL condition*

- An em wave or particle should not move more than 1 grid spacing per time step: Courant–Friedrichs–Lewy (CFL) condition
- In EPOCH C =0.95 by default

To resolve the plasma frequency the time step is: $\Delta t_p \leq \dfrac{1}{\omega_{pe}}$ $\omega_{pe} = \sqrt{\dfrac{n_e e^2}{\varepsilon_0 m_e}}$

Numerical stability requires the smaller Δt or:

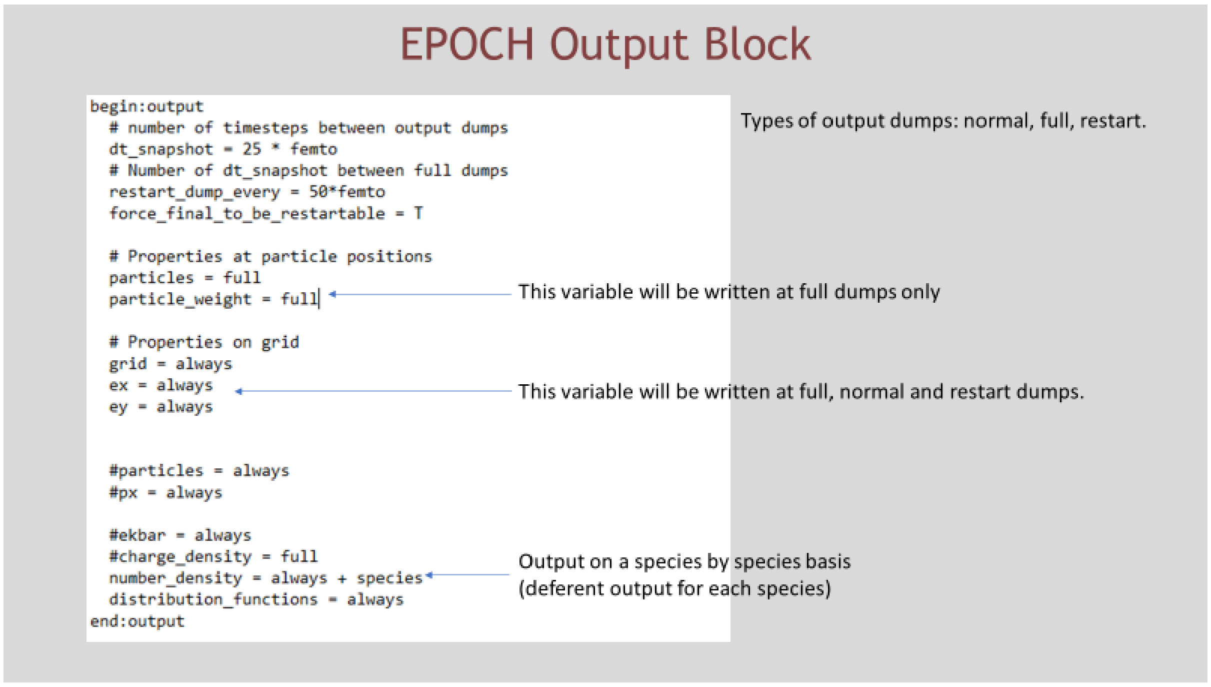
- If Debye length is resolved then Δt_{EM} is the limiter.
- If Debye length is not resolved, then Δt_p is the limiter.

In 2D-3D the stability condition :

$$\Delta t_{EM} \leq \frac{0.95}{c\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\right)^{1/2}}$$

*For 2x2nm cell ~ 0.0047 fs

- This division of actual particles into less macroparticles relaxes the computational demands of a simulation, although a very low number of macroparticles will lead to unnatural heating of the plasma.



EPOCH Output Block

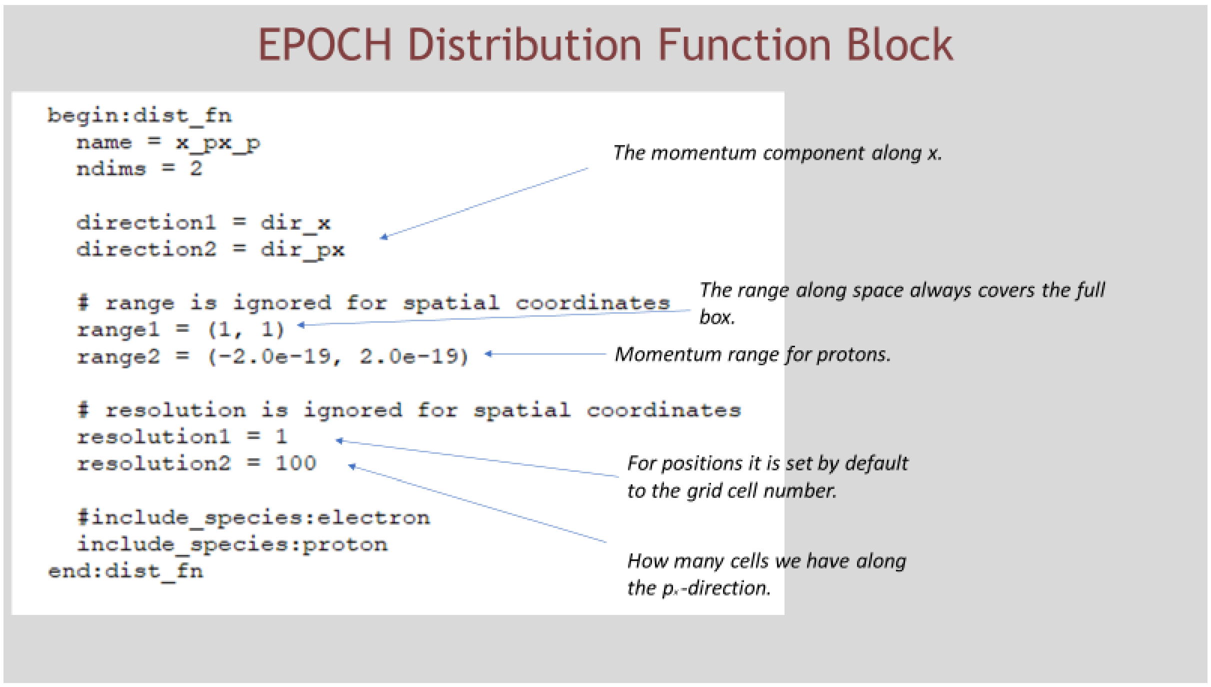
```
begin:output
  # number of timesteps between output dumps
  dt_snapshot = 25 * femto
  # Number of dt_snapshot between full dumps
  restart_dump_every = 50*femto
  force_final_to_be_restartable = T

  # Properties at particle positions
  particles = full
  particle_weight = full

  # Properties on grid
  grid = always
  ex = always
  ey = always


  #particles = always
  #px = always

  #ekbar = always
  #charge_density = full
  number_density = always + species
  distribution_functions = always
end:output
```

Types of output dumps: normal, full, restart.

This variable will be written at full dumps only

This variable will be written at full, normal and restart dumps.

Output on a species by species basis
(deferent output for each species)



EPOCH Distribution Function Block

```
begin:dist_fn
  name = x_px_p
  ndims = 2

  direction1 = dir_x
  direction2 = dir_px

  # range is ignored for spatial coordinates
  range1 = (1, 1)
  range2 = (-2.0e-19, 2.0e-19)

  # resolution is ignored for spatial coordinates
  resolution1 = 1
  resolution2 = 100

  #include_species:electron
  include_species:proton
end:dist_fn
```

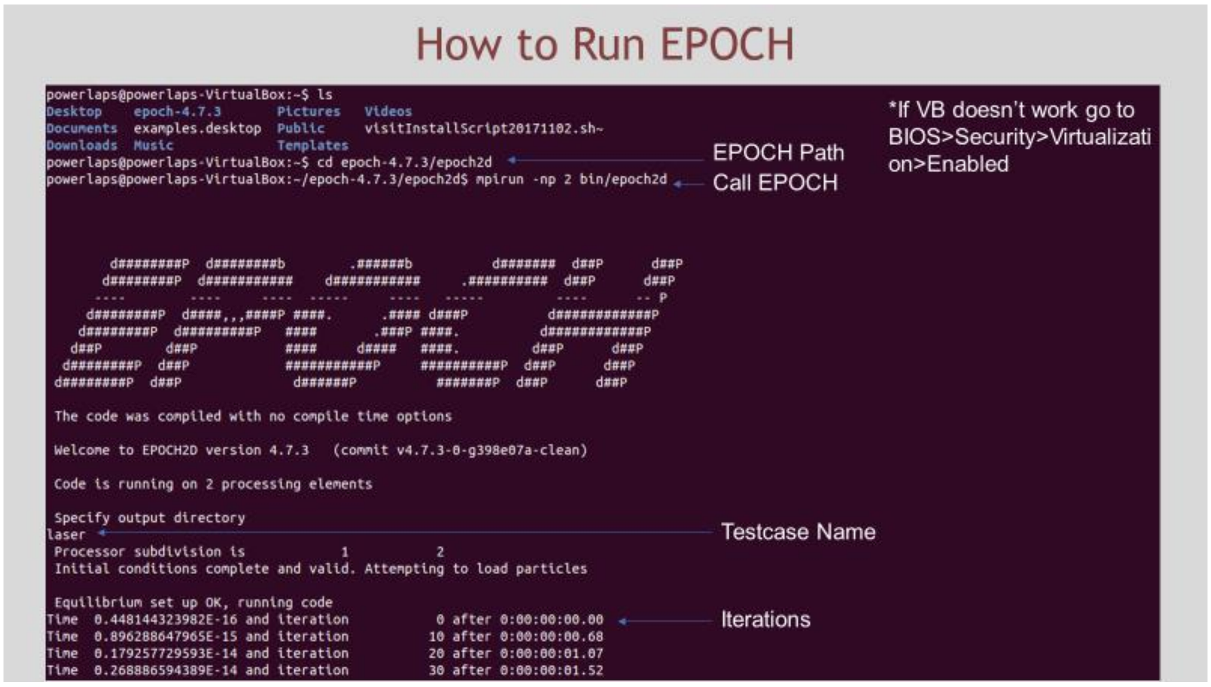
The momentum component along x.

The range along space always covers the full box.

Momentum range for protons.

For positions it is set by default to the grid cell number.

How many cells we have along the p_x-direction.



How to Run EPOCH

```
powerlaps@powerlaps-VirtualBox:~$ ls
Desktop    epoch-4.7.3       Pictures   Videos
Documents  examples.desktop  Public     visitInstallScript20171102.sh~
Downloads  Music             Templates
powerlaps@powerlaps-VirtualBox:~$ cd epoch-4.7.3/epoch2d
powerlaps@powerlaps-VirtualBox:~/epoch-4.7.3/epoch2d$ mpirun -np 2 bin/epoch2d



        d########P  d########b        .######b          d#######  d##P      d##P
       d########P  d###########    d###########     .##########  d##P      d##P
      ----        ----     ----  -----       ----   -----         ----      -- P
     d########P  d####,,,####P ####.      .#### d###P          d############P
    d########P  d#########P   ####       .###P ####.          d############P
   d##P        d##P           ####     d####   ####.         d##P      d##P
  d########P  d##P           ###########P    ##########P  d##P      d##P
 d########P  d##P             d######P         #######P  d##P      d##P

 The code was compiled with no compile time options

 Welcome to EPOCH2D version 4.7.3   (commit v4.7.3-0-g398e07a-clean)

 Code is running on 2 processing elements

 Specify output directory
laser
 Processor subdivision is            1           2
 Initial conditions complete and valid. Attempting to load particles

 Equilibrium set up OK, running code
Time  0.448144323982E-16 and iteration          0 after 0:00:00:00.00
Time  0.896288647965E-15 and iteration         10 after 0:00:00:00.68
Time  0.179257729593E-14 and iteration         20 after 0:00:00:01.07
Time  0.268886594389E-14 and iteration         30 after 0:00:00:01.52
```

EPOCH Path

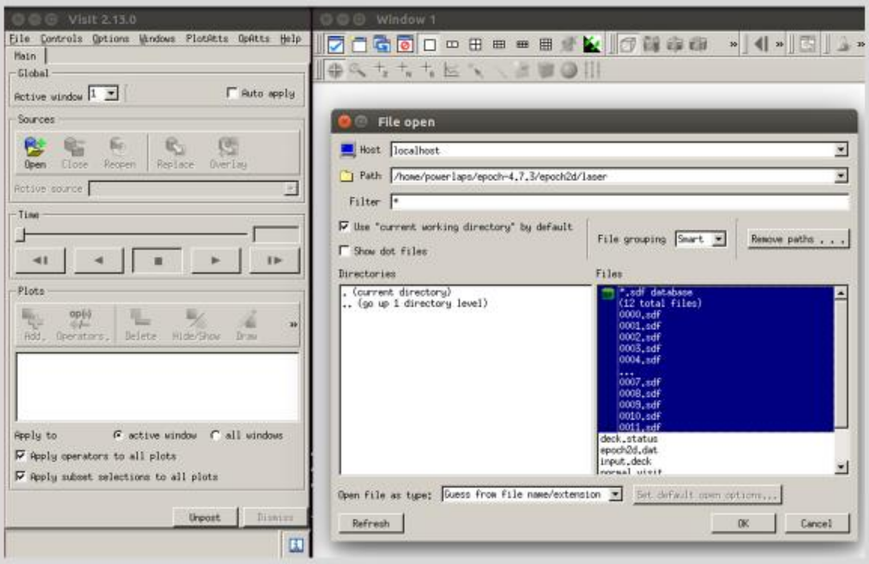
Call EPOCH

Testcase Name

Iterations

*If VB doesn't work go to BIOS>Security>Virtualization>Enabled

Post Prossesing - SDF File Visualization



VisIt 2.13.0

File Controls Options Windows PlotAtts OpAtts Help

Main

Global

Active window 1

☐ Auto apply

Sources

Open Close Reopen Replace Overlay

Active source

Time

Plots

Add.. Operators.. Delete Hide/Show Draw

Apply to [✓ active window] all windows

☑ Apply operators to all plots

☑ Apply subset selections to all plots

Unpost Dismiss

Window 1

File open

Host localhost

Path /home/powerlaps/epoch-4.7.3/epoch2d/laser

Filter *

☑ Use "current working directory" by default

☐ Show dot files

File grouping Smart

Remove paths . . .

Directories

```
. (current directory)
.. (go up 1 directory level)
```

Files

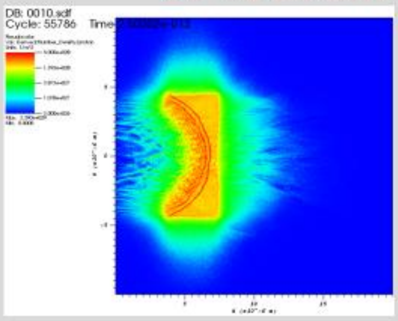
```
*.sdf database
(12 total files)
0000.sdf
0001.sdf
0002.sdf
0003.sdf
0004.sdf
...
0007.sdf
0008.sdf
0009.sdf
0010.sdf
0011.sdf
deck.status
epoch2d.dat
input.deck
normal.visit
```

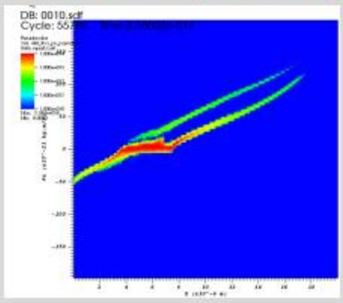
Open file as type: Guess from file name/extension

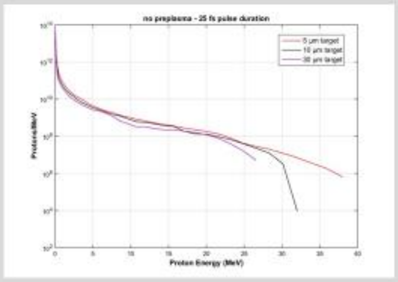
Set default open options...

Refresh OK Cancel

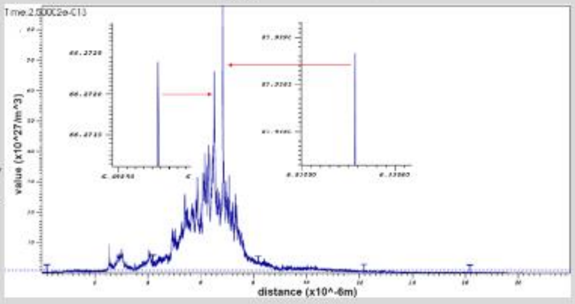
25 fs beam interaction with 5 μm target



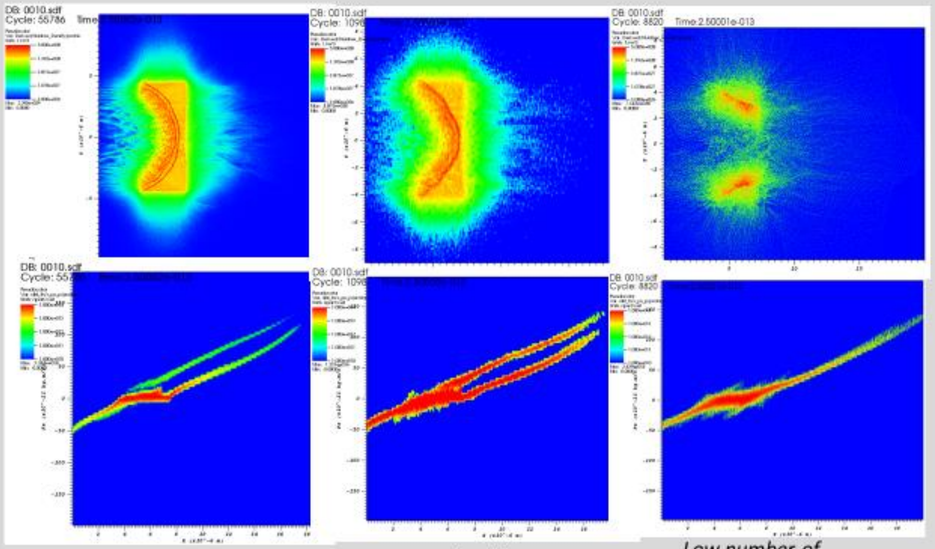




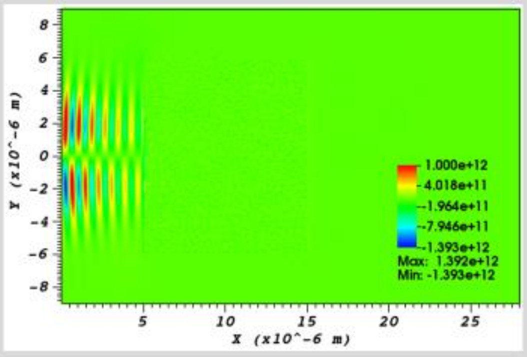
Density lineout

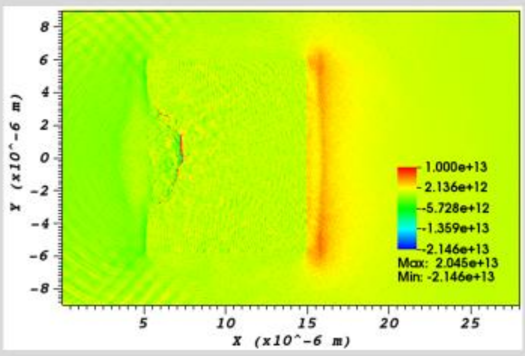


25 fs beam interaction with 5 μm target

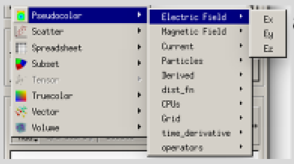
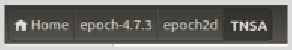


Laser Electric Field (V/m)





Homework

- Run VirtualBox-5.2.14-123301-Win.exe. After the installation is complete, Run virtual box.
- Got to *file>Import Appliance* an choose the powerlaps.ova file. Select the *powerlaps* session and press start (if it fails to start you should restart/bios and virtualization [enabled], ask for help)
- Open *Terminal*. Type *cd epoch-4.7.3/epoch2d*
- Run Epoch by typing *mpirun -np 2 bin/epoch2d (-np 2 for 2 processors)*. Type TNSA.
- Type visit to visualize the result. Press Open find the path of TNSA file and then click the SDF files.
- to Plot the E-field of the laser  and the hit Draw.
- Open *File>Epoch-4.7.3>epoch2d>TNSA*  *input.deck file* to alter the parameters of the simulation

Homework

- Open input.deck file.
- Set nx=1000, ny=500, number of particles per cell =1 for each species. set Simulation time 100fs.
- Run the testcase. Plot Pseudocolor> Derived>Particle density. take a lineout of electron density.
- Plot, Pseudocolor, dist_fn, Protons. (save figures)

- Open input.deck file.
- Change plasma density, pulse duration, energy, focal spot, Aluminum ionization degree, target thickness Etc.
- Run the testcase. Plot as before and compare.

- Extra! Create a partially ionized Carbon species block.

PowerLaPs

Innovative Education & Training in High Power Laser Plasmas

Computational Modeling & Simulations in Laser Matter Interactions

EXP 6: Smilei PIC simulation studies

J. Psikal

1 Computer labs: Particle-in-cell simulations of laser-plasma interaction

In the following simulation study cases, we will use particle-in-cell code *Smilei*. *Smilei* is open-source code protected by a licence CeCILL, the french equivalent to the open-source Gnu GPL license.

Extract: *This software is governed by the CeCILL-B license under French law and abiding by the rules of distribution of free software. You can use, modify and/ or redistribute the software under the terms of the CeCILL-B license as circulated by CEA, CNRS and INRIA at the following URL* `http://www.cecill.info`.

Figure 1: Smilei is a Particle-In-Cell code for plasma simulation. Open-source, collaborative, user-friendly and designed for high performances on super-computers, it is applied to a wide range of physics studies: from relativistic laser-plasma interaction to astrophysics.

- *Smilei* source code, user's manual, and tutorials are available at `http://www.maisondelasimulation.fr/smilei`
- Smilei's development depends on its visibility from publications or presentations featuring its results. When publishing simulation results involving *Smilei*, please cite the following article:

  J. Derouillat, A. Beck, F. Pérez, T. Vinci, M. Chiaramello, A. Grassi, M. Flé, G. Bouchard, I. Plotnikov, N. Aunai, J. Dargent, C. Riconda, M. Grech, *SMILEI: a collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation*, Comput. Phys. Commun. 222, 351-373 (2018)

1.1 Case study: Laser-driven ion acceleration

Aim of this study

- to study the interaction of ultrashort intense laser pulse with overdense plasma
- to demonstrate how to run a simulation in both the shared memory and distributed memory parallel mode (i.e., OpenMP and MPI)
- to become familiar with particle-in-cell code Smilei

Simulation parameters

- 1D simulation
- laser parameters: intensity 2×10^{21} W/cm^2, wavelength 800 nm, duration 30 fs at FWHM (in intensity), circularly polarized
- plasma parameters: hydrogen plasma layer of thickness 1 μm and electron density 50 n_{ec}, where n_{ec} is plasma critical density
- numerical parameters: cell size 10 nm, length of the simulation box 40 μm, duration of the simulation 200 fs

```
#-----------------------------------------------------------
#   SIMULATION PARAMETERS FOR THE PIC-CODE SMILEI
#-----------------------------------------------------------

# DEFINING MY OWN VARIABLES
# Simple Python script
# -----------------------------

import math

l0 = 2.0*math.pi          # laser wavelength [in code units]
t0 = l0                   # optical cycle
Lsim = 50.*l0             # length of the simulation box (= 40 mum)
Tsim = 75.*t0             # duration of the simulation (= 200 fs)
resx = 80.                # nb of cells in on laser wavelength
rest = resx/0.8           # nb of timesteps in one optical cycle
dt = t0/rest
ne = 50.    # electron density (code units => 1 = plasma
    critical density)
Te = 0.01   # Te normalised in mec^2 (code units) (= 5110 eV)

nppc = 200  # number of particle-per-cell

diagEvery   = int(2.5*t0/dt)  # frequency of some outputs

# DEFINING SMILEI's VARIABLES
# All in "blocks"
# ------------------------------

Main(
     geometry = "1Dcartesian",
     interpolation_order = 2,
     timestep = dt,
     simulation_time = Tsim,
     cell_length = [l0/resx],
     grid_length  = [Lsim],
     number_of_patches = [ 16 ],
     EM_boundary_conditions = [ ['silver-muller'] ] ,
    print_every = 50,
    random_seed = smilei_mpi_rank
)

Species(
    name = 'proton',
    position_initialization = 'random',
    momentum_initialization = 'maxwell-juettner',
    particles_per_cell = nppc,
    mass = 1836.,
    charge = 1.0,
    number_density = trapezoidal(ne, xvacuum=25.*l0, xplateau
    =1.25*l0),
    temperature = [Te],
    boundary_conditions = [ ['remove'] ],
    time_frozen = 24.*t0
)

Species(
    name = 'electron',
    position_initialization = 'random',
    momentum_initialization = 'maxwell-juettner',
    particles_per_cell = nppc,
    mass = 1.0,
    charge = -1.0,
    number_density = trapezoidal(ne, xvacuum=25.*l0, xplateau
    =1.25*l0),
    temperature = [Te],
    boundary_conditions = [ ['remove'] ],
    time_frozen = 24.*t0
)

LoadBalancing(
    initial_balance = True,
    every = 100
)

LaserPlanar1D(
    box_side         = 'xmin',
    a0               = 30.0,  # laser intensity 2.0e+21 W/cm2,
    wavelength 800 nm
    omega            = 1.,
    polarization_phi = 0.,
    ellipticity      = 1.,    # circular polarization
    time_envelope    = tgaussian(start=0., duration=24.*t0, fwhm
    =11.25*t0, center=12.*t0)
)

### DIAGNOSTICS

DiagScalar(
    every = 5,
)

```

```
DiagFields(
    every = diagEvery,
    fields = ['Ex','Ey','Bz','Rho_proton','Rho_electron']
)

DiagParticleBinning(
    deposited_quantity = 'weight',
    every = diagEvery,
    time_average = 1,
    species = ['proton'],
    axes = [
        ['ekin', 0., 50., 500]
    ]
)

DiagParticleBinning(
    deposited_quantity = 'weight',
    every = diagEvery,
    time_average = 1,
    species = ['electron'],
    axes = [
        ['ekin', 0., 50., 500]
    ]
)
```

Tasks

1. Explain the meaning of parameters defined in a namelist (an input file) for *Smilei* simulation. You can look at `http://www.maisondelasimulation.fr/smilei/namelist.html` for help.

2. Compare the resolution of cell size with skin depth and plasma Debye length.

3. Run the simulation on four CPU cores using MPI, type *top* command in Linux terminal in order to observe the load of computer during simulation run.

4. Run the simulation on multiple threads with shared memory using openMP (set the environment variable OMP_NUM_THREADS before the run), observe the load of computer during simulation run.

5. Plot energy balance during simulation: how particle kinetic energies, electromagnetic field energy and total energy evolves in time. Explain that. If you need help, you can look at `http://www.maisondelasimulation.fr/smilei/post-processing.html` or on tutorial at `https://smileipic.github.io/tutorials/tutorial1.html`

6. Plot the evolution of electromagnetic fields and electron and proton densities. Explain what you see.

7. Plot energy spectra of electrons and protons. Explain.

8. Try to run the same simulation except for another laser polarization (use linear polarization). Observe differences in energy balance, particle densities and energy spectra of particles.

Tips: You can use *ipython* to plot data or you can write a script in Python. Examples of two Python scripts plotting densities and energy spectra of protons and electrons are shown below.

```
import happi
S=happi.Open("./")
ne=S.Field(0,field="-Rho_electron", label="ne")
np=S.Field(0,field="Rho_proton", label="np")
happi.multiPlot(ne,np,saveAs="./dens.png")
raw_input()
```

```
import happi
S=happi.Open("./")
tstop=S.namelist.Main.simulation_time
tstep=S.namelist.Main.timestep
tfrozen=S.namelist.Species["electron"].time_frozen
ntfrozen=int(tfrozen/tstep+0.000001)
ntstop=int(tstop/tstep+0.000001)
eenspectr=S.ParticleBinning(1, timesteps=[ntfrozen, ntstop],
    data_log=True, label="electrons")
penspectr=S.ParticleBinning(0, timesteps=[ntfrozen, ntstop],
    data_log=True, label="protons")
happi.multiPlot(eenspectr,penspectr)
raw_input()
```

1.2 Case study: Laser-driven electron acceleration

Aim of this study

- to study the interaction of ultrashort intense laser pulse with underdense plasma
- to demonstrate the function of moving window and the model of frozen ions, which both simplifies the calculation
- to run multidimensional particle-in-cell simulation
- to use advanced diagnostics implemented in *Smilei*

Simulation parameters

- 2D simulation

- laser parameters: intensity 5.8×10^{19} W/cm^2 (dimensionless amplitude $a_0 = eE_0/(m_e \omega c) = 5.2$), wavelength 800 nm, duration 21.3 fs (8 laser cycles) at FWHM (in intensity), gaussian beam width $10 \mu m$ at FWHM, linearly p-polarized

- plasma parameters: underdense plasma layer of total thickness 10 mm, electron density profile $1/(1 + \exp\left(\frac{|x - x0| - wx}{Lx}\right) * n_{max}$, where $n_{max} = 10^{19}$ cm^{-3}, $x0 = 5000$ μm, $wx = 4700$ μm, $Lx = 50$ μm, ions are frozen in the whole simulation

- numerical parameters: cell size 50 nm along laser propagation axis, 400 nm in transverse direction, length of the simulation box 120 μm $\times$ 64 μm, duration of the simulation 4 ps

```
#----------------------------------------------------------------
#   SIMULATION PARAMETERS FOR THE PIC-CODE SMILEI
#----------------------------------------------------------------

# DEFINING MY OWN VARIABLES
# Simple Python script
# -------------------------------

import math

l0 = 2.0*math.pi       # laser wavelength [in code units]
t0 = l0                # optical cycle
Lsimx = 150.*l0    # length of the simulation box in x (= 120 mum)
Lsimy = 80.*l0     # length of the simulation box in y (= 64 mum)
Tsim = 1500.*t0        # duration of the simulation (= 4 ps)
resx = 16.
resy = 2.              # nb of cells in on laser wavelength
rest = resx/0.8        # nb of timesteps in one optical cycle
dt = t0/rest
nx = 150*16            # nb of cells along x-axis
npatchx = 32           # nb of patches along x-axis

xtot = 12500.*l0       #(= 10000 mum)
x0 = xtot/2.0
widthx = 5875.*l0      #(= 4700 mum)
Lx = 62.5*l0           #(= 50 mum)

n0max = 5.734e-3  # electron density (code units => 1 = plasma
    critical density)

# initial density profile of electrons
def n0_electron(x,y):
    return 1./(1.+math.exp((abs(x-x0)-widthx)/Lx))*n0max

FWHMinI  = 12.5*l0   #(= 10 mum)
waistinI = FWHMinI/(2.0*math.sqrt(math.log(2.0)))
```

```
waistinE = FWHMinI/math.sqrt(2.0*math.log(2.0))
FWHMtinI = 8.0*t0                            #(= 21.3 fs)
FWHMtinE = FWHMtinI*math.sqrt(2.0)      #(= 30 fs)

diagEvery    = int(37.5*t0/dt)  # frequency of some outputs (=
    100 fs)

# DEFINING SMILEI's VARIABLES
# All in "blocks"
# ——————————————————————

Main(
    geometry = "2Dcartesian",
    interpolation_order = 2,
    timestep = dt,
    simulation_time = Tsim,
    cell_length  = [ 10/resx, 10/resy ],
    grid_length = [ Lsimx, Lsimy ],
    number_of_patches = [32, 16],
    clrw = nx/npatchx,
    EM_boundary_conditions = [
        ["silver-muller","silver-muller"],
        ["silver-muller","silver-muller"],
    ],
    solve_poisson = False,
    print_every = 100,
    random_seed = smilei_mpi_rank
)

MovingWindow(
    time_start = 0.9*Main.grid_length[0],
    velocity_x = 0.9997
)

LoadBalancing(
    initial_balance = False,
    every = 20,
    cell_load = 1.,
)

Species(
    name = "electron",
    position_initialization = "regular",
    momentum_initialization = "cold",
    particles_per_cell = 4,
    mass = 1.0,
    charge = -1.0,
    number_density = n0_electron,
    mean_velocity = [0.0, 0.0, 0.0],
    pusher = "boris",
    time_frozen = 0.0,
    boundary_conditions = [
        ["remove", "remove"],
        ["remove", "remove"],
    ],
)

LaserGaussian2D(
    box_side         = "xmin",
    a0               = 5.2,  # intensity 5.8e+19
    omega            = 1.,
    focus            = [0., Main.grid_length[1]/2.],
    waist            = waistinE,
    time_envelope    = tgaussian(start=0., duration=2.*FWHMtinE,
                                 fwhm=FWHMtinE, center=FWHMtinE)
)

list_fields = ['Ex','Ey','Bz','Rho','Jx']

DiagFields(
    every = diagEvery,
    fields = list_fields
)

DiagProbe(
    every = diagEvery,
    origin = [0., Main.grid_length[1]/2.],
    corners = [
        [Main.grid_length[0], Main.grid_length[1]/2.],
    ],
    number = [nx],
    fields = list_fields
)

DiagScalar(
    every = int(diagEvery/10),
    vars=[
        'Uelm','Ukin_electron',
        'ExMax','ExMaxCell','EyMax','EyMaxCell','RhoMin','
RhoMinCell',
        'Ukin_bnd','Uelm_bnd','Ukin_out_mvw','Ukin_inj_mvw',
        'Uelm_out_mvw','Uelm_inj_mvw'
    ]
)

DiagParticleBinning(
    deposited_quantity = "weight",
    every = diagEvery,
    species = ["electron"],
    axes = [
        ["moving_x", 0., Lsimx, 300],
        ["ekin", 1., 500., 200]
    ]
)

DiagPerformances(
    every = diagEvery
)
```

Tasks

1. Run the simulation on four CPU cores using MPI, type *top* command in Linux terminal in order to observe the load of computer during simulation run.
2. Explain the meaning of parameters defined in a namelist (an input file) for *Smilei* simulation. You can look at `http://www.maisondelasimulation.fr/smilei/namelist.html` for help.
3. Compare the resolution of cell size with plasma Debye length.
4. Plot the evolution of electron densities and laser fields throughout the whole simulation (2D images). If you need help, you can look at `http://www.maisondelasimulation.fr/smilei/post-processing.html`
5. Plot the evolution of kinetic energy of electrons depending on their position along laser propagation axis. Explain what you see.
6. Select electrons trapped in the first bubble (based on the results from previous diagnostics) and plot their energy spectrum.
7. Plot the evolution of electric field component along the laser propagation axis in the middle of the simulation box (use Smilei's Probe diagnostics).

Tips: You can use *ipython* to plot data or you can write a script in Python. Example of Python script plotting energy spectra of electrons in the selected region is shown below.

```
import happi
S=happi.Open("./")
eenspectr=S.ParticleBinning(0, sum={"moving_x":[500, 800]},
    data_log=True)
eenspectr.animate()
raw_input()
```

1.3 Case study: Generation of electron-positron plasma

Aim of this study

- to study the interaction of two counter-propagating extremely intense laser pulses with thin dense plasma layer leading to the production of electron-positron pairs (the so-called multiphoton Breit-Wheeler process)
- to demonstrate QED (quantum electrodynamics) effects which are expected to be investigated in near future with next generation laser technology

- to analyze a simulation run performed on larger number of CPU cores than in previous cases (single node of computer cluster with 16 CPU cores was used to obtain our data)

Simulation parameters

- 1D simulation
- laser parameters: two counter-propagating laser pulses of maximum intensity 5×10^{23} W/cm^2, wavelength 1 μm, duration 30 fs at FWHM (in intensity), linearly p-polarized
- plasma parameters: hydrogen plasma layer of thickness 1 μm, and electron density 50 n_{ec}, where n_{ec} is plasma critical density
- numerical parameters: cell size 12.5 nm, length of the simulation box 51.2 μm, duration of the simulation 233 fs; the total number of cells in the simulation box is set to the value which enables to create larger number of patches

```
#------------------------------------------------------------
#   SIMULATION PARAMETERS FOR THE PIC-CODE SMILEI
#------------------------------------------------------------

# DEFINING MY OWN VARIABLES
# Simple Python script
# ----------------------------

import math

l0 = 2.0*math.pi   # laser wavelength [in code units]
t0 = l0            # optical cycle
Lsim = 51.2*l0     # length of the simulation box (= 51.2 mum)
Tsim = 70.*t0      # duration of the simulation (= 233 fs)
resx = 80.         # nb of cells on laser wavelength
rest = resx/0.8    # nb of timesteps in one optical cycle
dt = t0/rest
ne = 50.      # electron density (code units => 1 = plasma
    critical density)
Te = 0.01     # Te normalised in mec^2 (code units) (= 5110 eV)

nppc = 100    # number of particle-per-cell

c = 299792458                # Speed of light
lambdar = 1e-6               # Wavelength for normalization
wr = 2*math.pi*c/lambdar     # Normalization angular frequency

diagEvery    = int(3.33*t0/dt) # frequency of some outputs

# DEFINING SMILEI's VARIABLES
# All in "blocks"
# ———————————————————————

Main(
    geometry = "1Dcartesian",
    interpolation_order = 2,
    timestep = dt,
    simulation_time = Tsim,
    cell_length = [10/resx],
    grid_length  = [Lsim],
    number_of_patches = [ 512 ],
    EM_boundary_conditions = [ ['silver-muller'] ] ,
    print_every = 50,
    reference_angular_frequency_SI = wr,
    random_seed = smilei_mpi_rank
)

Species(
    name = 'proton',
    position_initialization = 'regular',
    momentum_initialization = 'maxwell-juettner',
    particles_per_cell = nppc,
    mass = 1836.,
    charge = 1.0,
    number_density = trapezoidal(ne, xvacuum=24.5*10, xplateau
    =1.0*10),
    temperature = [Te],
    boundary_conditions = [ ['remove'] ],
    time_frozen = 2.*Tsim
)

Species(
    name = 'electron',
    position_initialization = 'random',
    momentum_initialization = 'maxwell-juettner',
    particles_per_cell = nppc,
    mass = 1.0,
    charge = -1.0,
    number_density = trapezoidal(ne, xvacuum=25.1*10, xplateau
    =1.0*10),
    temperature = [Te],
    radiation_model='Monte-Carlo',
    radiation_photon_species = "photon",
    radiation_photon_sampling = 1,
    radiation_photon_gamma_threshold = 2,
    boundary_conditions = [['remove']],
    time_frozen = 20.*t0
)

Species(
    name = 'positron',
    position_initialization = "random",
    momentum_initialization = "cold",
    particles_per_cell = 0,
    c_part_max = 1.0,
    mass = 1.0,
    charge = 1.0,
    charge_density = 0.,
    mean_velocity = [0.0, 0.0, 0.0],
    temperature = [0.],
    radiation_model = 'Monte-Carlo',
    radiation_photon_species = "photon",
    radiation_photon_sampling = 1,
    radiation_photon_gamma_threshold = 2,
    boundary_conditions = [['remove']],
    time_frozen = 20.*t0,
)

Species(
    name = 'photon',
    position_initialization = "random",
    momentum_initialization = "cold",
    particles_per_cell = 0,
    c_part_max = 20.0,
    mass = 0,
    charge = 0.,
    number_density = 0.,
    mean_velocity = [0.0 ,0.0, 0.0],
    temperature = [0.],
    pusher = "norm",
    multiphoton_Breit_Wheeler = ["electron","positron"],
    multiphoton_Breit_Wheeler_sampling = [1,1],
    boundary_conditions = [['remove']],
)

LoadBalancing(
    initial_balance = True,
    every = 100
)

# first laser pulse
LaserPlanar1D(
    box_side        = 'xmin',
    a0              = 604.,  # laser intensity 5.0e+23 W/cm2, wavelength 1 um
    omega           = 1.,
    polarization_phi = 0.,
    ellipticity     = 0.,
    time_envelope   = tgaussian(start=0., duration=19.2*t0, fwhm=9.0*t0, center=9.6*t0)
)

# second laser pulse
LaserPlanar1D(
    box_side        = 'xmax',
    a0              = 604.,  # laser intensity 5.0e+23 W/cm2, wavelength 1 um
    omega           = 1.,
    polarization_phi = 0.,
    ellipticity      = 0.,
    time_envelope    = tgaussian(start=0., duration=19.2*t0, fwhm
   =9.0*t0, center=9.6*t0)
)

#----------------------------------------------------------------
# QED parameters

RadiationReaction(
    chipa_disc_min_threshold = 1e-2,
    output_format = 'hdf5',
    table_path = "./databases"
)

MultiphotonBreitWheeler(
    output_format = 'hdf5',
    table_path = "./databases"
)
#----------------------------------------------------------------
### DIAGNOSTICS

DiagScalar(
    every = 5,
    vars=['Uelm','Ukin','Utot',
          'Uexp',
          'Ubal',
          'Urad',
          'Ukin_electron',
          'Ukin_positron',
          'Ukin_photon',
          'Ntot_electron',
          'Ntot_positron',
          'Ntot_photon']
)

DiagFields(
    every = diagEvery,
    fields = ['Ex','Ey','Bz',
              'Rho_proton',
              'Rho_electron',
              'Rho_positron',
              'Rho_photon']
)

DiagParticleBinning(
    deposited_quantity = 'weight',
    every = diagEvery,
    time_average = 1,
    species = ['electron'],
    axes = [
        ['ekin', 0., 5000., 500]
    ]
)
```

```

DiagParticleBinning(
    deposited_quantity = 'weight',
    every = diagEvery,
    time_average = 1,
    species = ['positron'],
    axes = [
        ['ekin', 0., 5000., 500]
    ]
)

DiagParticleBinning(
    deposited_quantity = 'weight',
    every = diagEvery,
    time_average = 1,
    species = ['photon'],
    axes = [
        ['ekin', 0., 5000., 500]
    ]
)

DiagPerformances(
    every = diagEvery
)
```

Tasks

1. Download data from the simulation already performed on computer cluster. You can download this data at `http://kfe.fjfi.cvut.cz/~psikal/PowerLaPs/case3.tar.gz`.
2. Explain the meaning of parameters defined in input file *QEDpairs_1d.py* for this simulation.
3. Plot laser electric field. Observe the amplitudes of the fields before and after the interaction with plasma. Explain that.
4. Plot temporal evolution of the number of electrons, positrons, and photons in the simulation. Determine the ratio of the number of generated positrons to the initial number of electrons.
5. Show the number of cells and particles per each CPU core during the whole simulation run (use Performances diagnostic in *Smilei*).
6. Image densities of particles during the whole interaction. Interpret the results.
7. Plot energy spectra of positrons and photons.

Tips: You can use *ipython* to plot data or you can write a script in Python. Examples of Python script plotting the number of positrons and electrons during the whole simulation and determining their ratio as well as the script illustrating the number of cells and particles per each CPU core are shown below.

```
import happi
import matplotlib.pyplot as plt
S=happi.Open("./")
Nel=S.Scalar("Ntot_electron")
Npos=S.Scalar("Ntot_positron")
Nphot=S.Scalar("Ntot_photon")
#happi.multiPlot(Nel,Npos,Nphot,figure=1)
happi.multiPlot(Nel,Npos,figure=1)
fig=plt.figure(1)
plt.savefig('./Npartnbs.png')
Nposmax=max(Npos.getData())
Nel0=min(Nel.getData())
print("Ratio of the number of generated positrons to the initial
    number of electrons: " + str(Nposmax/Nel0))
raw_input()
```

```
import happi
S=happi.Open("./")
diagc=S.Performances(raw="number_of_cells")
diagc.animate(saveAs="./ncells.png")
raw_input()
diagp=S.Performances(raw="number_of_particles")
diagp.animate(saveAs="./nparts.png")
raw_input()
```

O3 – Annex