

The Laser Model in FLASH

Milad Fatenejad

**Flash Center for Computational Science
University of Chicago**

**RAL Tutorial
May 2012**

FLASH contains code for modeling laser energy deposition and radiation diffusion

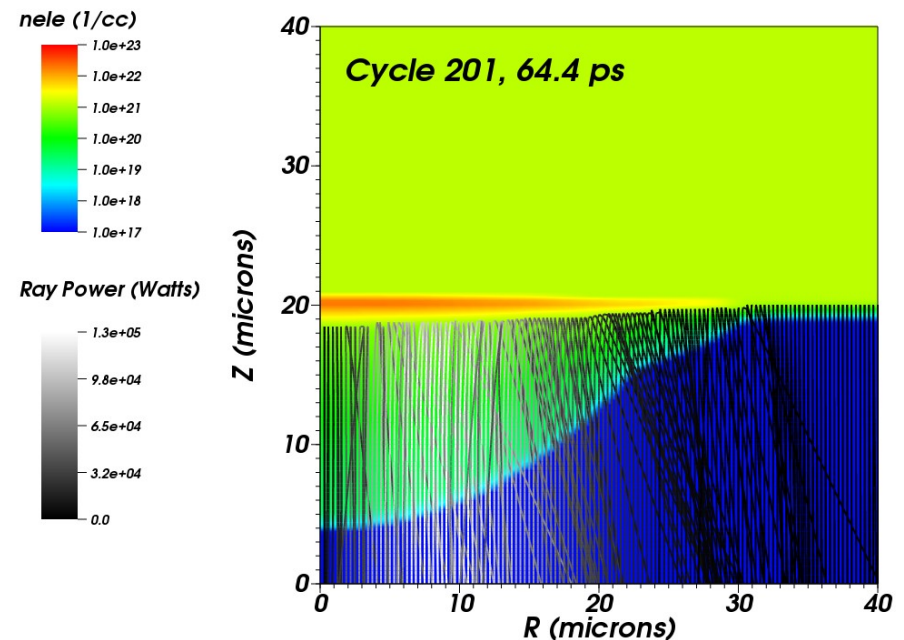


- Laser ray tracing
- Radiation diffusion
- Example problem: The `LaserSlab` simulation
 - The setup call
 - The runtime parameter file
 - Running the simulation
 - Visualizing the laser rays

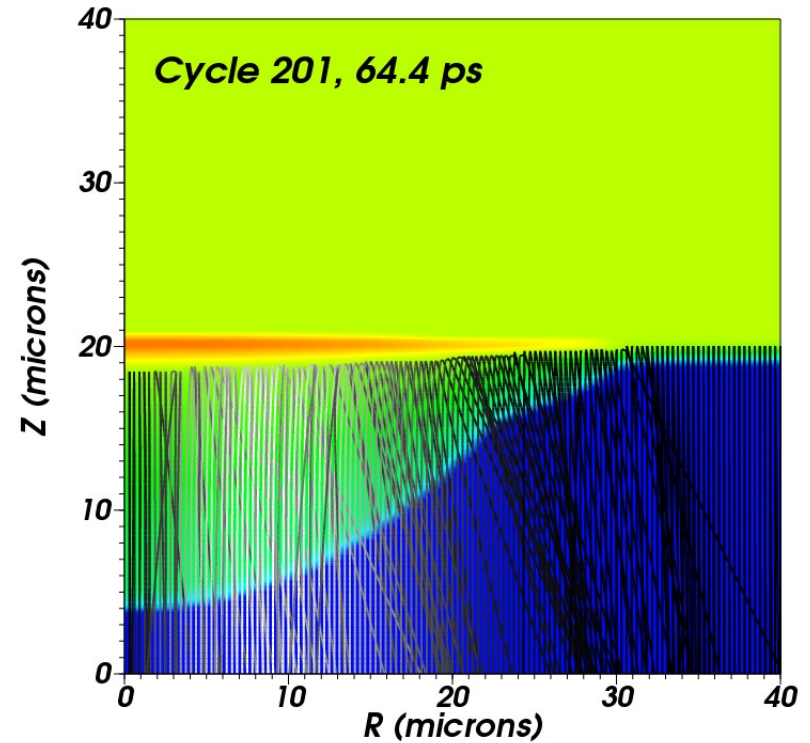
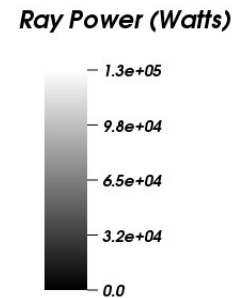
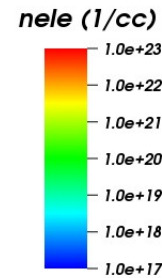
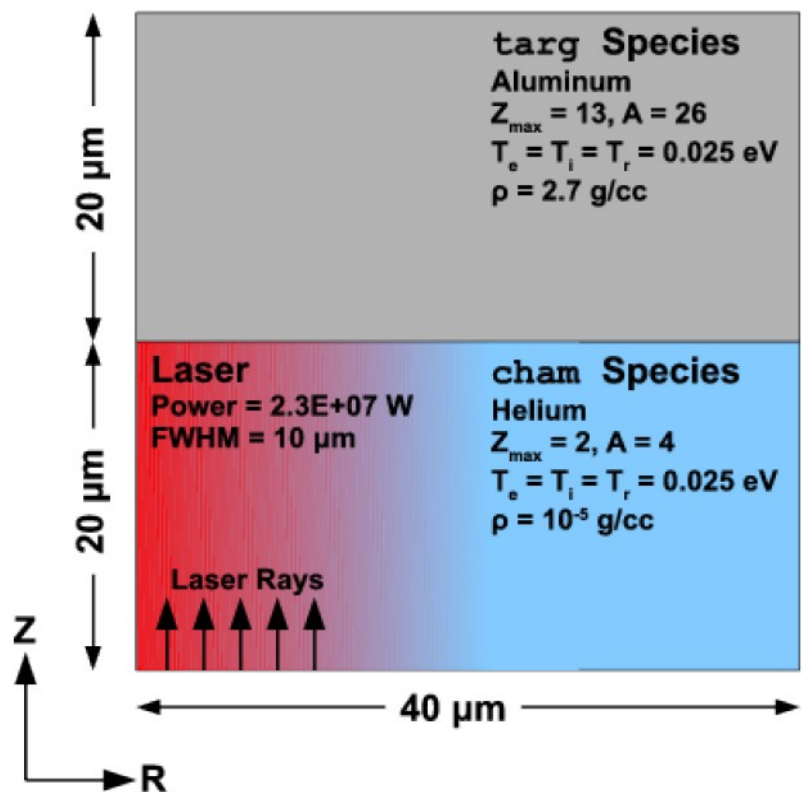
Many rays are launched on each time step and deposit energy within the domain as they travel



- The user's guide (section 25.7.5) describes the `LaserSlab` simulation which models a laser illuminating a slab
- The description is very detailed and focuses on how to define the laser geometry and the radiation options
- Please take a look at this section



Many rays are launched on each time step and deposit energy within the domain as they travel



FLASH uses ray-tracing in the geometric optics approximation to model laser energy deposition



- In this approximation, the equation of motion of a ray is given by:

$$\frac{d^2 \mathbf{x}}{dt^2} = \nabla \left(\frac{c}{2} \eta^2 \right)$$

- The index of refraction is:

$$\eta^2 = 1 - \frac{\omega_p^2}{\omega^2} = 1 - \frac{n_e}{n_c}, \quad n_c = \left(\frac{m_e}{4\pi} \right) \left(\frac{\omega}{e} \right)^2$$

- And...

$$\frac{d^2 \mathbf{x}}{dt^2} = \nabla \left(-\frac{c^2}{2} \frac{n_e}{n_c} \right)$$

FLASH uses the Kaiser¹ algorithm which assumes that the electron number density is linear within a computation cell



- The electron number density is given by:

$$n_e(\vec{x}) = \langle n_e \rangle + \langle \vec{\nabla} n_e \rangle \cdot (\vec{x} - \langle \vec{x} \rangle) + O(\epsilon^2)$$

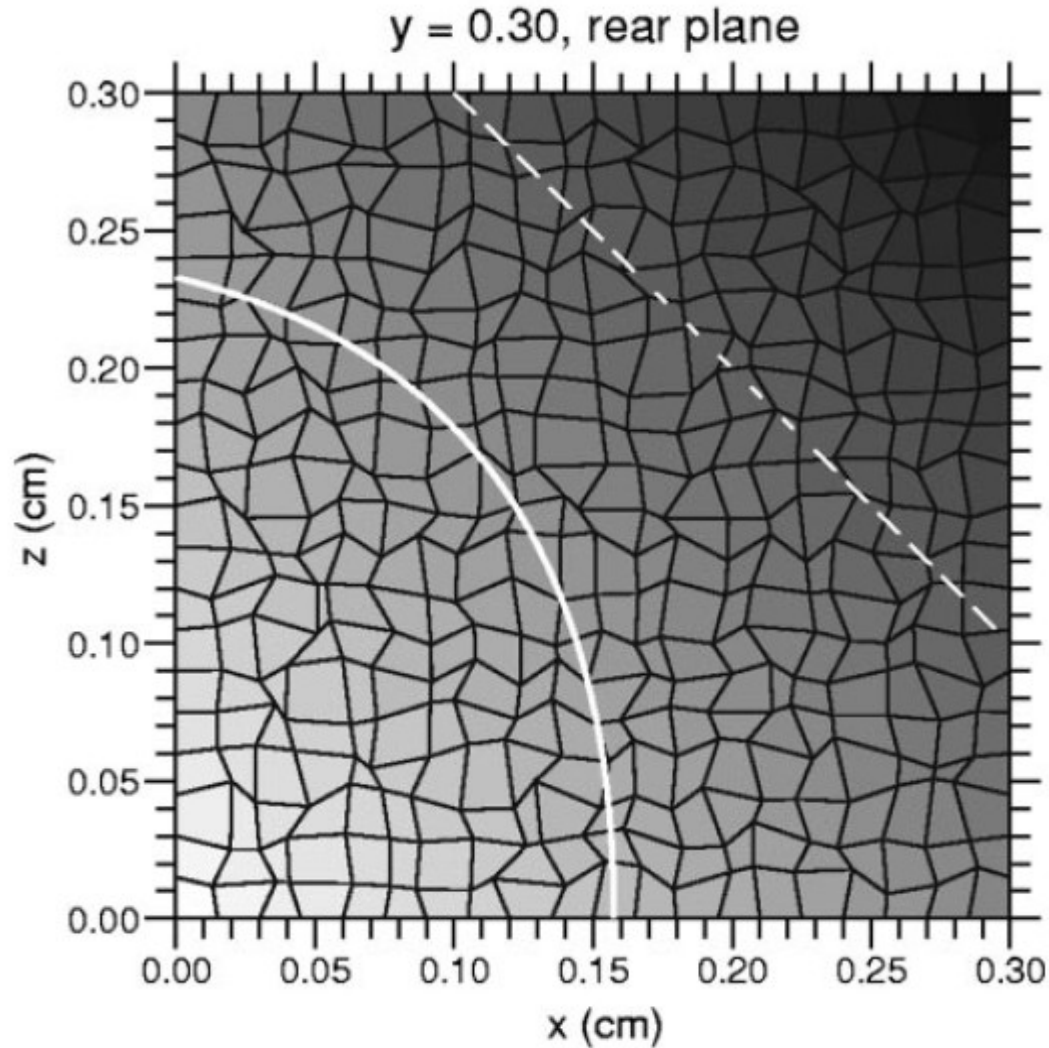
- The ray equation of motion becomes:

$$\frac{d^2 \mathbf{x}}{dt^2} = - \left(\frac{c^2}{2} \right) \frac{\langle n_e \rangle}{n_c}$$

- This shows that when n_e is linear within a cell, the rays follow a parabolic trajectory through the cell
- The electron number density will not be continuous in general. The Kaiser algorithm fixes this by applying Snell's law at the cell interfaces
 - Thus, rays can reflect or refract off of cell interfaces

¹Kaiser, *Phys Rev E*, 61, 895 (2000)

Rays follow a parabolic trajectory and turn around as they approach the critical surface



Over a time-step, the laser energy is exponentially reduced with the inverse-Bremsstrahlung rate



- As a ray travels through a cell its energy (or power) is reduced:

$$P(\Delta t) = P(0) \exp \left\{ - \int_0^{\Delta t} dt \nu_{ib}[\vec{x}(t)] \right\}$$

- The energy loss rate is:

$$\nu_{ib} = \frac{4}{3} \left(\frac{2\pi}{m_e} \right)^{1/2} \frac{n_e^2 Z e^4 \ln \Lambda}{n_c T_e^{3/2}}$$

You can modify this by customizing the file `ed_inverseBremsstrahlungRate.F90`

Currently, the ray trace works in several geometries and additional capabilities are being continuously added



- The supported geometries are:
 - “2D-in-2D” for X-Y and R-z ← this is the most stress-tested
 - 3D X-Y-Z ← implemented, but not well tested
- 2D-in-2D means that the simulation is 2D *and* the ray trace is also done in 2D
- Using the 2D-in-2D ray trace with beams entering the domain obliquely can lead to non-physical heating near the z-axis in R-z simulations
- It can also be difficult to correctly define the beam geometry
- We are working on implementing “3D-in-2D” geometry where the ray-trace is done in 3D but the mesh is 2D
 - This should be working in the next month
- We are also working to improve the accuracy of the ray trace algorithm itself

FLASH uses flux-limited multigroup diffusion to model radiation



- The total radiation internal energy is given by:

$$\frac{\partial}{\partial t}(\rho e_{\text{rad}}) + \nabla \cdot (\rho e_{\text{rad}} \mathbf{v}) + P_{\text{rad}} \nabla \cdot \mathbf{v} = \nabla \cdot \mathbf{q}_{\text{rad}} - Q_{\text{abs}} + Q_{\text{emis}}$$

- The `Hydro` unit solves the advection/work parts of this equation. So we are left with:

$$\frac{\partial}{\partial t}(\rho e_{\text{rad}}) = \nabla \cdot \mathbf{q}_{\text{rad}} - Q_{\text{abs}} + Q_{\text{emis}}$$

- To compute the flux, absorption, and emission, we use flux limited multigroup diffusion
- Let the frequency space be divided into N_g groups and u_g be the energy density in each group

The total absorption, emission, flux, energy density is just a sum over each group



- The energy density within a group obeys:

$$\frac{\partial u_g}{\partial t} + \nabla \cdot (u_g \mathbf{v}) + \left(\frac{u_g}{e_{\text{rad}}/\rho} \right) P_{\text{rad}} \nabla \cdot \mathbf{v} = -\nabla \cdot \mathbf{q}_g + Q_{\text{emis},g} - Q_{\text{abs},g}$$

where:

$$Q_{\text{abs}} = \sum_{g=1}^{N_g} Q_{\text{ele},g}, \quad Q_{\text{emis}} = \sum_{g=1}^{N_g} Q_{\text{emis},g}, \quad \mathbf{q}_{\text{rad}} = \sum_{g=1}^{N_g} \mathbf{q}_g, \quad \rho e_{\text{rad}} = \sum_{g=1}^{N_g} u_g$$

- The group radiation energy flux is approximated as:

$$\mathbf{q}_g = \frac{c}{3\sigma_{t,g}} \nabla u_g$$

- And the absorption rate is determined by the absorption opacity within a particular group:

$$Q_{\text{abs},g} = -c\sigma_{a,g}u_g$$

The total absorption, emission, flux, energy density is just a sum over each group



- Usually, σ_t is the Rosseland averaged absorption opacity and σ_a is the Planck averaged absorption opacity
- The plasma is assumed to emit radiation in a blackbody defined by a local radiation temperature with an emission opacity varying from group to group:

$$Q_{\text{abs,emis}} = \sigma_{e,g} a c (T_e^n)^4 \frac{15}{\pi^4} [P(x_{g+1}) - P(x_g)] \quad P(x) = \int_0^x \frac{(x')^3}{\exp(x') - 1} dx'$$

- Using these definitions, we have:

$$\frac{\Delta u_g}{\Delta t} - \nabla \cdot \left(\frac{c}{3\sigma_{t,g}} \nabla u_g^{n+1} \right) + c\sigma_{a,g} u_g^{n+1} = c\sigma_{e,g} a (T_e^n)^4 \frac{15}{\pi^4} [P(x_{g+1}) - P(x_g)] \quad , \quad x_g = h\nu_g / k_B T_e$$

This diffusion equation is solved (semi) implicitly on each time step using the HYPRE library within the RadTrans/MGD unit

The electron internal energy must be updated to account for the emission/absorption



- Emission/absorption represents an exchange of energy between the electrons and ions
- The change in this energy is given by:

$$\Delta(\rho e_{\text{ele}}) = \sum_g \left\{ \sigma_{a,g}^n u_g^{n+1} - \sigma_{e,g}^n a(T_e^n)^4 \frac{15}{\pi^4} [P(x_{g+1}) - P(x_g)] \right\}$$

- As you can see, the emission term is evaluated at time level n
- While the solution of the *diffusion* equation stable for large time steps, the coupling between the electrons and radiation field can become unstable
- This is especially true when the radiation energy is large and the opacity and opacity are large



General Purpose Implicit Diffusion Solver

- Electron thermal conduction and radiation diffusion require implicit solutions of diffusion equations
- FLASH has a general purpose implicit diffusion solver in the `Diffuse` unit that can be used to “diffuse” any cell centered variable over a time step. The subroutine is:

`Diffuse_solveScalar`

- This subroutine can be used to solve an equation of the form:

$$A \frac{\partial f}{\partial t} + C f = \nabla \cdot B \nabla f + D$$

- You can easily call this subroutine to solve other diffusion equations. For example for ion conduction, charged particle diffusion, resistivity, etc...

Let's go through the LaserSlab simulation and see how to put together a laser driven simulation which uses all of the HEDP physics in FLASH



- The setup line:

```
-auto LaserSlab -2d +cylindrical +pm4dev
```

```
-nxb=16 -nyb=16 +hdf5typeio
```

```
species=cham,targ +mtmmmt +laser +uhd3t
```

```
+mgd mgd_meshgroups=6 -parfile=example.par
```

- The `species` setup variable tells FLASH that there are going to be two separate materials in this simulation called `targ` (target material) and `cham` (chamber material)
- When this is specified many runtime parameters are automatically created which let you specify the properties of these materials at runtime in the runtime parameters file

Let's go through the LaserSlab simulation and see how to put together a laser driven simulation which uses all of the HEDP physics in FLASH



- The setup line:

```
-auto LaserSlab -2d +cylindrical +pm4dev  
-nxb=16 -nyb=16 +hdf5typeio  
species=cham,targ +mtmmmt +laser +uhd3t  
+mgd mgd_meshgroups=6 -parfile=example.par
```

- This option turns on a special IO (developed by Chris Daley) unit that is really good:
 - Plotting is easier (trust me)
 - Allows you to have processes with no blocks
- I now use +hdf5typeio for all simulations

Let's go through the LaserSlab simulation and see how to put together a laser driven simulation which uses all of the HEDP physics in FLASH



- The setup line:

```
-auto LaserSlab -2d +cylindrical +pm4dev
```

```
-nxb=16 -nyb=16 +hdf5typeio
```

```
species=cham,targ +mtmmmt +laser +uhd3t
```

```
+mgd mgd_meshgroups=6 -parfile=example.par
```

- The `+mgd` setup shortcut tells FLASH to include the code for multigroup radiation diffusion
- The `mgd_meshgroups` tells FLASH the *maximum* number of energy groups that will be represented on a single process

Let's go through the LaserSlab simulation and see how to put together a laser driven simulation which uses all of the HEDP physics in FLASH



- The setup line:

```
-auto LaserSlab -2d +cylindrical +pm4dev  
-nxb=16 -nyb=16 +hdf5typeio  
species=cham,targ +mtmmmt +laser +uhd3t  
+mgd mgd_meshgroups=6 -parfile=example.par
```

- The `+mtmmmt` option turns on the multitemperature, multimaterial, multitype EOS
- This 3T EOS lets you specify a different EOS model for each species (material) in the simulation very conveniently (through the `flash.par` file)

To use radiation diffusion, you need to specify the energy group structure and boundary conditions



```
rt_useMGD          = .true.           ! Turn on radiation diffusion
rt_mgdNumGroups    = 6                 ! Specify that there are six groups
rt_mgdBounds_1     = 1.0e-01          ! Specify group boundaries in eV
rt_mgdBounds_2     = 1.0e+00
rt_mgdBounds_3     = 1.0e+01
rt_mgdBounds_4     = 1.0e+02
rt_mgdBounds_5     = 1.0e+03
rt_mgdBounds_6     = 1.0e+04
rt_mgdBounds_7     = 1.0e+05
rt_mgdFlMode       = "fl_harmonic"    ! Specify the type of flux limiter
rt_mgdFlCoef       = 1.0              ! Specify the coefficient of the
                                       ! flux-limiter

rt_mgdXlBoundaryType = "reflecting"
rt_mgdXrBoundaryType = "vacuum"
rt_mgdYlBoundaryType = "vacuum"
rt_mgdYrBoundaryType = "reflecting"
rt_mgdZlBoundaryType = "reflecting"
rt_mgdZrBoundaryType = "reflecting"
```

The dirichlet and vacuum boundary conditions are also useful



- **Dirichlet boundary:**

- Set `rt_mgdXlBoundaryType = "dirichlet"`
- Specify a fixed value for the radiation *temperature* on the boundary:
`rt_mgdXlBoundaryTemp = 11604.55 # in Kelvin`
- If you need more control (for example, a monoenergetic radiation source) you can customize the boundary condition on a group-by-group basis in your simulation by modifying `Simulation_init.F90`

- **Vacuum:**

- Set `rt_mgdXlBoundaryType = "vacuum"`
- The “vacuum” boundary condition for diffusion is:

$$\frac{u_g}{2} + \frac{1}{4} \nabla \cdot u_g \cdot \mathbf{n} = 0$$

- **Beware: Vacuum boundary conditions are not very accurate in diffusion theory – too much radiation energy can get trapped in the domain!**

To use radiation diffusion, you must also tell FLASH how to compute the opacity for each material



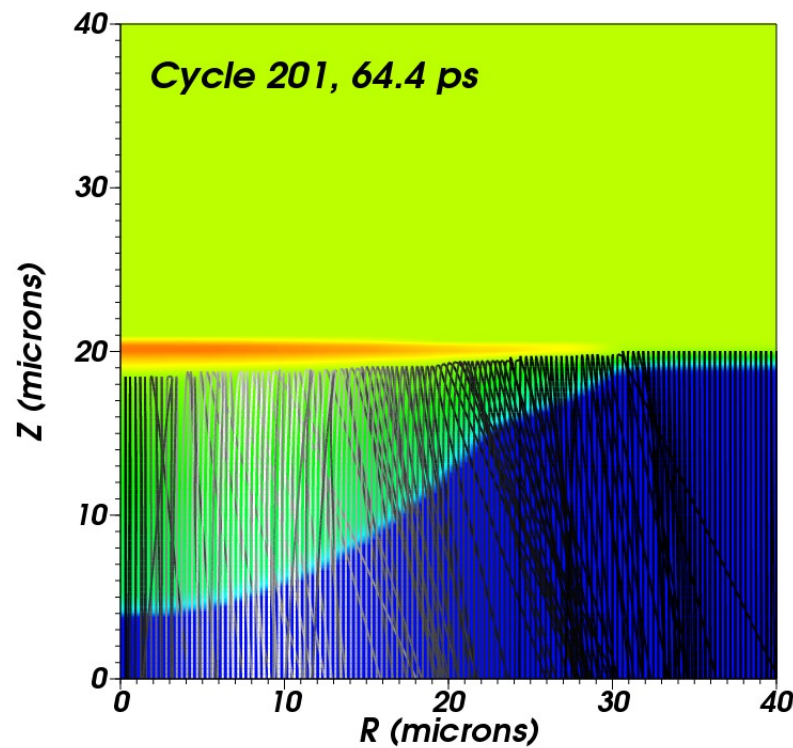
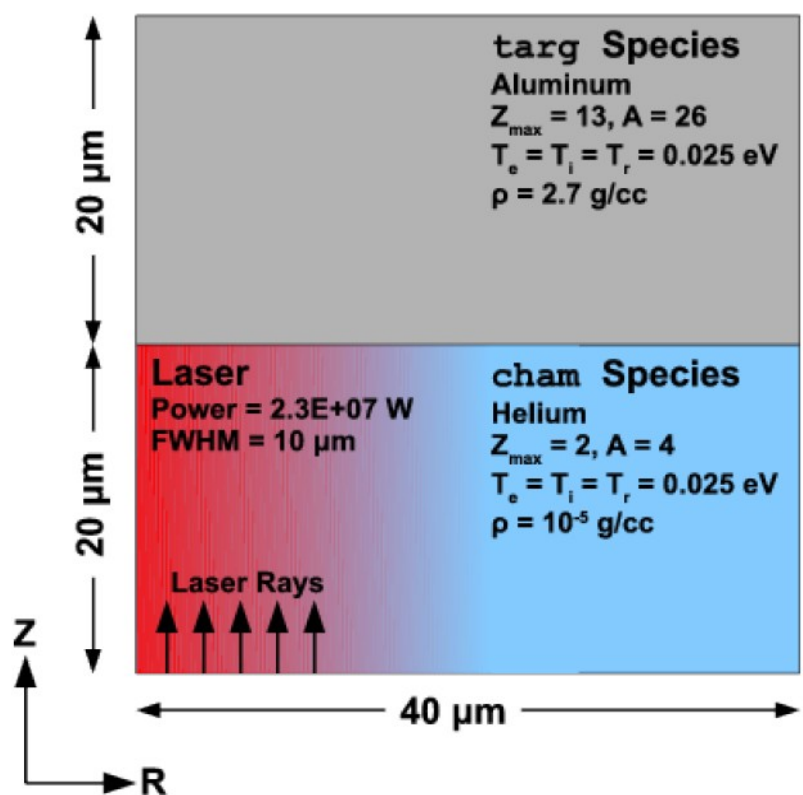
```
useOpacity      = .true.      ! Turn on opacity calculations
                                   ! At all times useOpacity must
                                   ! equal rt_useMGD
```

```
### SET CHAMBER (HELIUM) OPACITY OPTIONS ###
op_chamAbsorb   = "op_tabpa"
op_chamEmiss    = "op_tabpe"
op_chamTrans    = "op_tabro"
op_chamFileType = "ionmix4"
op_chamFileName = "he-imx-005.cn4"
```

```
### SET TARGET (ALUMINUM) OPACITY OPTIONS ###
op_targAbsorb   = "op_tabpa"
op_targEmiss    = "op_tabpe"
op_targTrans    = "op_tabro"
op_targFileType = "ionmix4"
op_targFileName = "al-imx-003.cn4"
```

Please see the opacity section in the user's guide 21.4 for a description of the IONMIX4 format. If you can write these files, then you can use your own tabulated opacities with FLASH!

Many rays are launched on each time step and deposit energy within the domain as they travel



To use the laser, you must define at least one beam and one pulse



- **First, turn on the laser:**

```
useEnergyDeposition = .true. ! Turn on the laser
ed_maxRayCount      = 2000  ! Max. number of rays per process
ed_gradOrder        = 2     ! Linear density gradient in a cell
```

- **Each “beam” definition provides the following information:**
 - Spatial orientation
 - Intensity profile
 - Number of rays
 - Wavelength (microns)
 - Which pulse to use
- **Each “pulse” defines a power profile as a function of time. Multiple beams can use the same pulse**

The example simulation has a single beam with rays traveling in the +z direction



- Rays are traced from the *lens* to the *target* (focal spot)

```
### SETUP LASER BEAM ###  
ed_numBeams = 1 ! Total number of beams  
                ! in this simulation
```

$$I(r) = I_0 \exp \left[- \left(\frac{r}{\lambda} \right)^2 \right]^\gamma$$

```
# Setup Gaussian Beam:
```

```
ed_lensX_1          = 0.00e-04  
ed_lensY_1          = -0.1  
ed_targetX_1        = 20.0e-04  
ed_targetY_1        = 20.0e-04  
ed_semiaxis1_1      = 40.0e-04  
ed_pulseNumber_1    = 3           ! Use pulse number 3  
ed_wavelength_1     = 1.053       ! In microns  
ed_crossSectionID_1 = 3           ! Three means  
                               ! supergaussian  
ed_decayExponent_1  = 1.0         ! Gamma, 1 = Gaussian  
ed_decayRadius1_1   = 1.201122e-03 ! Lambda (cm): note, this  
                               ! is NOT the FWHM  
  
ed_lensEqualsTarget_1 = .true.  
ed_numRays_1         = 512        ! Number of rays to  
                               ! launch per cycle for  
                               ! this beam
```

$$P = 2\pi \int_0^R dr r I(r),$$

A single pulse is associated with each beam. A pulse is a piecewise linear power vs. time function



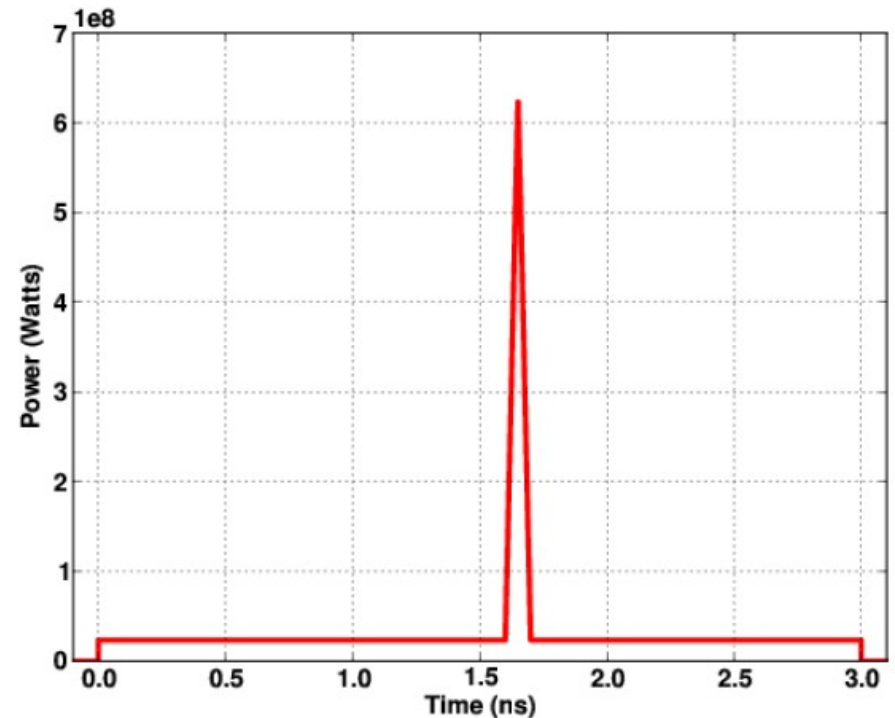
```
ed_numSections_3 = 5 # This pulse is described using 5
                    # points
```

```
# Specify times at which power changes (s):
```

```
ed_time_3_1 = 0.0
ed_time_3_2 = 1.6e-09
ed_time_3_3 = 1.65e-09
ed_time_3_4 = 1.7e-09
ed_time_3_5 = 3.0e-09
```

```
# Specify powers (Watts):
```

```
ed_power_3_1 = 2.333333e+07
ed_power_3_2 = 2.333333e+07
ed_power_3_3 = 6.233333e+08
ed_power_3_4 = 2.333333e+07
ed_power_3_5 = 2.333333e+07
```





Specify the EOS/properties of each material

```
# Target material defaults set for Aluminum at room temperature:
```

```
sim_rhoTarg = 2.7  
sim_teleTarg = 290.11375  
sim_tionTarg = 290.11375  
sim_tradTarg = 290.11375  
ms_targA = 26.9815386  
ms_targZ = 13.0  
ms_targZMin = 0.02  
eos_targEosType = "eos_tab"  
eos_targSubType = "ionmix4"  
eos_targTableFile = "al-imx-003.cn4"
```

```
# Chamber material defaults set for Helium at pressure 1.6 mbar:
```

```
sim_rhoCham = 1.0e-05  
sim_teleCham = 290.11375  
sim_tionCham = 290.11375  
sim_tradCham = 290.11375  
ms_chamA = 4.002602  
ms_chamZ = 2.0  
eos_chamEosType = "eos_tab"  
eos_chamSubType = "ionmix4"  
eos_chamTableFile = "he-imx-005.cn4"
```



Lets look at the output!

- We will examine the output in visit, I won't go into a lot of detail on how to use visit, but will give some basic information
- Useful variables:
 - `depo` → the amount of laser energy deposited in a cell per unit mass
 - `tele`, `tion`, `trad` → the electron, ion, radiation temperature (K)
 - `r001`, ..., `r006` → the specific radiation energy in each group (ergs/g)

You can visualize the trajectories in VisIt, but it requires a few steps



- These runtime parameters setup the laser ray visualization:

```
### LASER IO OPTIONS ###
ed_useLaserIO           = .true. # Turn on ray visualization
ed_laserIOMaxNumPositions = 10000
ed_laserIONumRays       = 128    # Number of rays to write
```

- The rays are automatically written to the *plot* files every time a plot file is generated
- You must use the `extract_rays.py` script (in the `tools/scripts` directory in the FLASH source tree)

```
../tools/scripts/extract_rays.py lasslab_hdf5_plt_cnt_*
```

- The `extract_rays.py` script has several dependencies, including NumPy and PyTables, so you need to install these to use it!

FLASH now contains a laser ray trace model and radiation diffusion



- **Development is ongoing – especially in the laser package:**
 - 3D-in-2D R-z ray-trace will be available in the next release
 - This will be followed by continued improvements to the accuracy of the ray trace algorithm
 - Improve radiation/matter coupling to eliminate (or at least reduce) stability
- **The radiation diffusion documentation in the user's guide is currently out-of-date**
 - This will be rewritten for the next release (my fault)



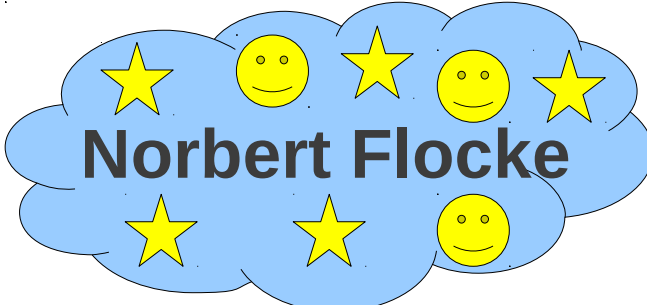
FLASH now contains a laser ray trace model and radiation diffusion

- Development is ongoing – especially in the laser package:
 - 3D-in-2D R-z ray-trace will be available in the next release
 - This will be followed by continued improvements to the accuracy of the ray trace algorithm
 - Improve radiation/matter coupling to eliminate (or at least reduce) stability
- The radiation diffusion documentation in the user's guide is currently out-of-date
 - This will be rewritten for the next release (my fault)
- Thanks to **Norbert Flocke** for writing the laser ray trace package!!



FLASH now contains a laser ray trace model and radiation diffusion

- Development is ongoing – especially in the laser package:
 - 3D-in-2D R-z ray-trace will be available in the next release
 - This will be followed by continued improvements to the accuracy of the ray trace algorithm
 - Improve radiation/matter coupling to eliminate (or at least reduce) stability
- The radiation diffusion documentation in the user's guide is currently out-of-date
 - This will be rewritten for the next release (my fault)

- Thanks to  **Norbert Flocke** for writing the laser ray trace package!!