slf: Users Manual and Theory Book

Nicholas N. Gibbons

July, 2025

Abstract

slf is a program for simulating steady flamelets using the GDTk gas models and kinetics files. This document will explain how it works and present a number of usage examples, but is neccessarily brief. Details of the implementation and API can be found in `gdtk/src/slf`.

Quick Start

Install the program using the following terminal commands:

```
cd gdtk/src/slf
make FLAVOUR=fast install
```

You should also install the optional gas library from GDTK, which can be done with the instructions available here. Once installed, test the code using the following example:

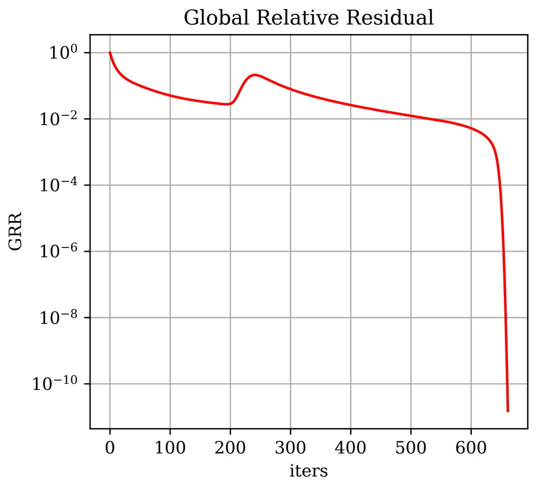
```
cd gdtk/examples/slf

cp $DGD_REPO/examples/kinetics/hydrogen-ignition-delay/Jachimowski-1992-species.inp ./
cp $DGD_REPO/examples/kinetics/hydrogen-ignition-delay/Jachimowski-1992.lua ./

prep-gas gm-jach92-air13.inp gm.lua
prep-chem gm.lua rr-jach92-air13.inp rr.lua

slf hydrogen.yaml
```

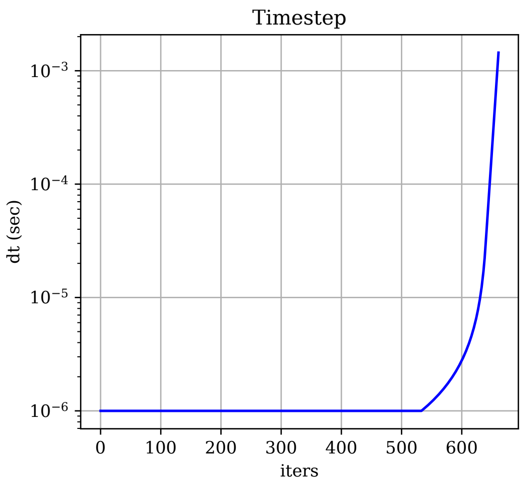
You should get the following output:

```
Called slf.run(): Target Global Relative Residual=1.000000e-10
iter  0000 dt 1.000000e-06 GRmax 3.098193e+08 GR 3.098193e+08 GRR 1.000000e+00
iter  0050 dt 1.000000e-06 GRmax 3.098193e+08 GR 3.099558e+07 GRR 1.000441e-01
iter  0100 dt 1.000000e-06 GRmax 3.098193e+08 GR 1.569775e+07 GRR 5.066743e-02
iter  0150 dt 1.000000e-06 GRmax 3.098193e+08 GR 1.049509e+07 GRR 3.387489e-02
iter  0200 dt 1.000000e-06 GRmax 3.098193e+08 GR 8.873138e+06 GRR 2.863973e-02
iter  0250 dt 1.000000e-06 GRmax 3.098193e+08 GR 6.059617e+07 GRR 1.955856e-01
iter  0300 dt 1.000000e-06 GRmax 3.098193e+08 GR 2.443113e+07 GRR 7.885606e-02
iter  0350 dt 1.000000e-06 GRmax 3.098193e+08 GR 1.298824e+07 GRR 4.192199e-02
iter  0400 dt 1.000000e-06 GRmax 3.098193e+08 GR 8.080834e+06 GRR 2.608241e-02
iter  0450 dt 1.000000e-06 GRmax 3.098193e+08 GR 5.446319e+06 GRR 1.757902e-02
iter  0500 dt 1.000000e-06 GRmax 3.098193e+08 GR 3.839074e+06 GRR 1.239133e-02
iter  0550 dt 1.203723e-06 GRmax 3.098193e+08 GR 2.753937e+06 GRR 8.888852e-03
iter  0600 dt 2.781427e-06 GRmax 3.098193e+08 GR 1.618075e+06 GRR 5.222642e-03
iter  0650 dt 1.943555e-04 GRmax 3.098193e+08 GR 9.177269e+03 GRR 2.962137e-05
Reached target residual 1.000000e-10 (1.529302e-11), stopping...
Done simulation in 16 seconds.
```

You can view the convergence of the calculation using the residuals script:

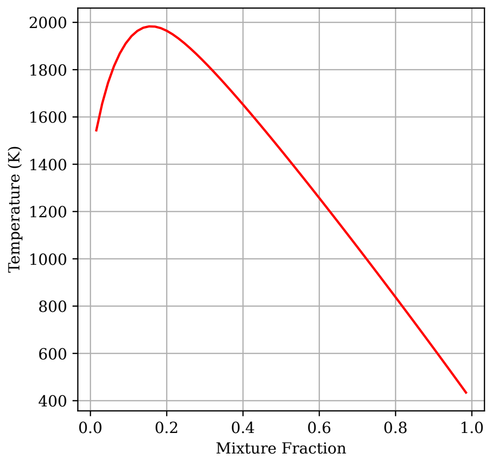
```
python3 residuals.py hydrogen.log
```

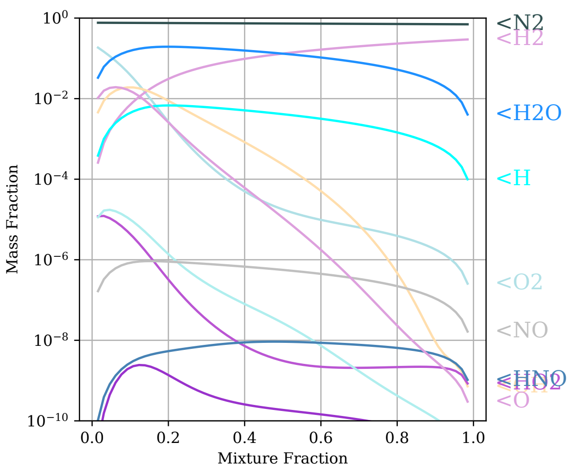




or the final steady solution with the solution script:

```
python3 solution.py hydrogen.sol
```





Theory

slf solves the steady-state 1D reaction diffusion equations in mixture fraction space, using a finite-difference method and an fixed analytic expression for the scalar dissipation. This method was originally proposed by Ref. [1], which should be consulted for mathematical details. Following the implementation of Ref. [2], the equations are formulated in terms of temperature and species mass fractions as follows.

$$\frac{dY_s}{dt} = \frac{\chi}{2}\frac{\partial^2 Y_s}{\partial Z^2} + \frac{\dot{\omega}_s M_s}{\rho} \tag{1}$$

$$\frac{dT}{dt} = \mathrm{Le}\,\frac{\chi}{2}\frac{\partial^2 T}{\partial Z^2} + \frac{1}{c_p}\sum_s \frac{h_s \dot{\omega}_s M_s}{\rho} \tag{2}$$

Note that Equation 1 is a vector equation, there is one instance of it for each species s in the reaction mechanism. In contrast, Equation 2 is a scalar equation. The index s here is used in the summation sign to add up the species production rates to get the effect of heat release on the temperature.

The scalar dissipation χ is computed using the following expression.

$$\chi = D \exp(-2 \,\mathrm{erfc}^{-1}(2Z)^2) \tag{3}$$

Where erfc^{-1} is the inverse complementary error function, which is approximated using an expression from Ref. [3]. The parameter D is a constant that can be adjusted by the user to increase or decrease the strain rate that the flame is experiencing.

Numerical Method

Equation 1 and Equation 2 are the time dependent unsteady reaction diffusion equations. We wish to seek the values of temperature and mass fractions that minimise these time derivatives over the entire domain $Z = 0 \to 1$. To do this, first we construct the semi-discrete equations using central differences at N finite difference points, which are labeled using the index i.

$$\frac{dY_s}{dt}^i = \frac{\chi^i}{2} \frac{Y_s^{i+1} - 2Y_s^i + Y_s^{i-1}}{(dZ)^2} + \frac{\dot{\omega}_s^i M_s}{\rho^i} \tag{4}$$

$$\frac{dT}{dt}^i = \mathrm{Le}\, \frac{\chi^i}{2} \frac{T_s^{i+1} - 2T_s^i + T_s^{i-1}}{(dZ)^2} + \frac{1}{c_p^i} \sum_s \frac{h_s^i \dot{\omega}_s^i M_s}{\rho^i} \tag{5}$$

The number of indices here is becoming somewhat unreasonable, for which I apologise. Keep in mind that the index i tends to run over the spatial dimension of the problem, and that s runs over the collection of species. Note that the central differences require values from either side of the given node i. At the boundaries of the calculation, we assume that the unknown T or Y outside the domain is specified as a boundary condition.

Implicit Time Advancement

In what follows we are going to construct a numerical method for seeking the steady-state concentrations and temperature. To do this, it will be helpful to consider all the unknown variables in a node as members of the vector $\boldsymbol{U}_j = [Y_0, Y_1, Y_2, ...T]$. We then consider stacking up all of the equations from all of the nodes, (over both i and j), to create a large system of coupled nonlinear equations. The right-hand side, or time-derivatives, of these equations is called the residual $\boldsymbol{R}$.

$$\boldsymbol{R}_j^i = \frac{dU_j}{dt}^i = \begin{pmatrix} \frac{dY_0}{dt}^0 \\ \frac{dY_1}{dt}^0 \\ \vdots \\ \frac{dT}{dt}^0 \\ \frac{dY_0}{dt}^1 \\ \frac{dY_1}{dt}^1 \\ \vdots \end{pmatrix} \tag{6}$$

Inside the slf program, both $\boldsymbol{U}$ and the residual vector $\boldsymbol{R}$ are stored as a single contiguous block of numbers with the species and temperature for each node packed together. We will want to think about $\boldsymbol{R}$ as a block vector, or a vector of vectors, where each node's packed data is like a mini vector making up the elements of the larger one. The numerical method is developed using a kind of implicit timestepping approach, where the values of $\boldsymbol{U}$ at the next timestep $\boldsymbol{U}^{n+1}$ are evaluated using the residual at the next timestep $\boldsymbol{R}^{n+1}$.

$$\boldsymbol{R} = \frac{d\boldsymbol{U}}{dt} \tag{7}$$

$$\boldsymbol{U}^{n+1} = \boldsymbol{R}^{n+1}\ \mathrm{dt} + \boldsymbol{U}^{n} \tag{8}$$

Since $\boldsymbol{R}^{n+1}$ is not known, it is evaluated using a first-order linearisation that introduces the Jacobian matrix $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}$.

$$\boldsymbol{U}^{n+1} = \left[\boldsymbol{R}^{n} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}(\boldsymbol{U}^{n+1} - \boldsymbol{U}^{n})\right]\ \mathrm{dt} + \boldsymbol{U}^{n} \tag{9}$$

This matrix equation can be rearranged as follows, to create the following linear system problem for the unknown change $\Delta\boldsymbol{U} = \boldsymbol{U}^{n+1} - \boldsymbol{U}^{n}$.

$$\left[\frac{1}{\mathrm{dt}}\boldsymbol{I} - \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]\Delta\boldsymbol{U} = \boldsymbol{R}^{n} \tag{10}$$

Construction of the Jacobian

At this stage an important property of the 1D reaction diffusion equations comes into play: That is, the Jacobian matrix is block tridiagonal. What this means is that most of the matrix is occupied by zeros. Only the central diagonal block, and the blocks on either side have any entries that are non zero. This means that the entire matrix can be constructed and stored relatively cheaply, and then inverted exactly using the block version of Thomas's algorithm.

Constructing the Jacobian proceeds using the complex-step finite differencing method. The algorithm is similar to a colour-mapping scheme, where a small imaginary component ε is added to the first mass fraction $\boldsymbol{U}_0$ in every fourth node. Every fourth node because the stencils we use are three nodes wide, and we want to compute the influence of each $\boldsymbol{U}$ on the surrounding nodes' $\boldsymbol{R}$ without contaminating influences. We then compute the residual and get the derivative as follows.

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}} = \frac{\Im(\boldsymbol{R}(\boldsymbol{U} + \varepsilon i))}{\varepsilon} \tag{11}$$

These derivatives are inserted into the correct column of the Jacobian matrix, then the imaginary components are cleared, then a new set of imaginary perturbations are applied to every fourth node but shifted one node over, and the process is repeated twice more to get all of the matrix filled out for variable Y_0. Then the other mass fractions, and finally the temperature are treated with the same process.

Solution and Convergence

Solution of the flamelet equations begins with an backward-Euler timemarching scheme that constructs and then solves the following linear system using a default timestep of $\mathrm{dt} = 1e - 6$.

$$\left[\frac{1}{\mathrm{dt}}\boldsymbol{I} - \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]\Delta\boldsymbol{U} = \boldsymbol{R}^{n} \tag{12}$$

After solving for $\Delta\boldsymbol{U}$, the field variables are corrected with no under-relaxation, and the loop repeats until convergence. To assess this, at each step we compute a scalar quantity called the Global Residual GR, which is defined as the L2 norm of the residual vector.

$$\mathrm{GR} = \sqrt{\sum_j \sum_i \boldsymbol{R}^i_j \boldsymbol{R}^i_j} \tag{13}$$

The global residual at the beginning of the calculation is recorded, and at subsequent steps, we keep track of the reduction in residual relative to its starting value. This quantity, the GR divided by the initial GR is sometimes called the Global Relative Residual GRR. The user may set a target GRR to stop the calculation at, or a default of 1e-10 is used.

Timestep Growth

slf borrows the timestep growth algorithm as the steady state solver in Eilmer. That is, once the Global Relative Residual drops below a prespecified threshold (by default, 1e-2), the timestep begins to grow as the residual drops further using the following relationship.

```
    double dtnew = dt*pow(GRRold/GRR, 1.6);
    dtnew = fmin(dtnew, 1.2*dt);
    dtnew = fmax(dtnew, 0.5*dt);
    dtnew = fmin(dtnew, 1.0);
    return dtnew;
```

As the residual drops further, this increases the timestep up to a maximum value of 1.0, and the identity matrix term in Equation 12 becomes smaller relative to the Jacobian term. Eventually the equation approaches a Newton's method with quadratic convergence, and the steady-state is approached extremely rapidly.

Anatomy of an Input File

An example input file for slf is shown below. These are in yaml format.

```
gas_file_name: "gm.lua"             # GDTk gas model file
reaction_file_name: "rr.lua"        # GDTk reactions model file

p: 49767.1                          # Constant pressure over the domain
N: 64                               # Number of finite-difference nodes
D: 2000.0                           # Strain rate parameter

T0: 1400.0                          # Left side temperature boundary condition
T1: 400.0                           # Right side temperature boundary condition

Y0: {'N2': 0.767, 'O2': 0.233}      # Left side species boundary condition
Y1: {'N2': 0.70, 'H2': 0.30}        # Right side species boundary condition
```

Glossary of Variables

T	Temperature (K)
p	Pressure (Pa)
N	Number of finite difference nodes
Z	Mixture Fraction

GR	Global Residual: $\mathrm{GR} = \sqrt{\sum_j \sum_i R^i_j R^i_j}$
GRR	Global Relative Residual: $\frac{\mathrm{GR}}{\mathrm{GR}_0}$
dt	Timestep for Implicit Time Advancement (sec)
Y_s	Mass fraction of species s
χ	Scalar Dissipation
D	Strain Rate Parameter
$\dot{\omega}_s$	Molar Production Rate of species s (mol/m^3/s)
M_s	Molar mass of species s (mol/kg)
ρ	Mixture density (kg/m^3
Le	Lewis Number
c_p	Mixture specific heat at constant pressure (J/kg)
h_s	Specific enthalpy of species s (J/kg)
$\boldsymbol{U}$	Vector of unknown field quantities to be solved for: $[Y_0, Y_1, Y_2, ..., T]$
$\boldsymbol{R}$	Rate of change in time of unknown field quantities: $\boldsymbol{R}_j = \frac{d\boldsymbol{U}_j}{dt}$
$\Delta\boldsymbol{U}$	Increment in U over an implicit time step
$\Im$	Function that returns the imaginary component of a complex number
ε	Perturbation sized used in estimating derivatives
$\boldsymbol{I}$	Identity Matrix
T0	Left side boundary condition temperature (K)
T1	Right side boundary condition temperature (K)
Y0	Left side boundary condition composition
Y1	Right side boundary condition composition

Explanation of Commonly used Index Quantities

i	i refers to a specific node in the domain. It runs from zero to $N-1$.
j	j refers to an entry in the individial node unknowns $\boldsymbol{U}$ or $\boldsymbol{R}$. It runs over the number of species + 1 to include the temperature
s	s refers to a species, typically in the mass fractions $\boldsymbol{Y}$. It runs over the number of species in the chemistry mechanism
n	n is used to refer to the timestep in the section on implicit time advancement

References

[1] H. Pitch and N. Peters, "A Consistent Flamelet Formulation for Non-Premixed Combustion Considering Differential Diffusion Effects," *Combustion and Flame*, vol. 114, pp. 26–40, 1998.

[2] R. A. W. M. J. M. S. Lapointe Y. Xuan, "A computationally-efficient method for steady-state flamelet calculations," *LLNL-JRNL-807545*, Mar. 2020.

[3] S. Winitzki, "A Handy Approximation for the Error Function and Its Inverse." [Online]. Available: https://scholar.google.com/citations?user=Q9U40gUAAAAJ&hl=en