CRANE: (**C**hemical **R**e**A**ction **NE**twork)

An Open-Source Software for Plasma Chemical Reactions

Shane Keniley, Davide Curreli

kenileyl@illinois.edu, dcurreli@illinois.edu

University of Illinois at Urbana-Champaign - Nuclear, Plasma, and Radiological Engineering

The 71st Annual Gaseous Electronics Conference

CRANE Tutorial Contents I

CRANE: **C**hemical **R**e**A**ction **NE**twork

- CRANE is an open source software dedicated to modeling the time evolution of coupled rate equations.
- Utilizing the Action system of the MOOSE framework, the software is designed to accept an arbitrary number of human-readable reaction equations.
- CRANE was developed with two use cases in mind:
  1. Global (0D) model, as a standalone application
  2. Spatial (1D-3D) model, used as a subapp of other MOOSE applications

Solving Systems of Reactions

- The mathematical kernel of the software is a system of ODE equations:

$$\frac{d[n_i]}{dt} = \sum_{j=1}^{j_{max}} S_{ij} \tag{1}$$

- Consider a reaction involving species A, B, and C with rate coefficient K:

$$a_1 A + bB \xrightarrow{K} a_2 A + cC \tag{2}$$

- The rate of production for each species S_{ij} is described as

$$S_A = (a_2 - a_1) K_j (n_A)^a (n_B)^b \tag{3}$$

$$S_B = -b K_j (n_A)^a (n_B)^b \tag{4}$$

$$S_C = c K_j (n_A)^a (n_B)^b \tag{5}$$

Action Description

- The focus of CRANE is the *ChemicalReactions* Action, which accepts explicitly written reaction equations.
- The Action automatically adds all of the necessary Kernels, AuxKernels, UserObjects, etc. needed to solve equation (1).
- It includes two subblocks: [./Network] for spatial problems, and [./ScalarNetwork] for 0D problems.

$$\frac{dn_e}{dt} = k_1[e^-][Ar] - k_2[e][Ar^+][Ar]$$

$$\frac{dn_{Ar^+}}{dt} = k_1[e^-][Ar] - k_2[e][Ar^+][Ar]$$

$$\frac{dn_{Ar}}{dt} = -k_1[e^-][Ar] + k_2[e][Ar^+][Ar]$$

$$\downarrow$$

```
[ChemicalReactions]
  [./ScalarNetwork]
    species = ‘e Ar+ Ar’
    file_location = ’RateCoefficients’

    reactions = ‘e + Ar -> e + e + Ar+       : EEDF
                 e + Ar+ + Ar -> Ar + Ar     : 1e-25’
  [../]
[]
```

Action Description

```
[ChemicalReactions]
  [./ScalarNetwork]
    species = ‘e Ar+ Ar’
    file_location = ‘RateCoefficients’
    reactions = ‘e + Ar -> e + e + Ar+      : EEDF
                 e + Ar+ + Ar -> Ar + Ar    : 1e-25’
  [../]
[]
```

- The Action automatically counts the stoichiometric coefficients and applies Kernels.
- This problem adds six total Kernels:
  1. **e**: Product2BodyScalar (reaction 1), Reactant3BodyScalar (reaction 2)
  2. **Ar**: Reactant2BodyScalar (reaction 1), Product3BodyScalar (reaction 2)
  3. **Ar+**: Product2BodyScalar (reaction 1), Reactant3BodyScalar (reaction 2)
- Rate coefficients are included as Auxiliary variables, which are calculated at the beginning of each timestep

Action Description

```
[ChemicalReactions]
  [./ScalarNetwork]
    species = ‘e Ar+ Ar’
    file_location = ‘RateCoefficients’
    reactions = ‘e + Ar -> e + e + Ar+       : EEDF
                  e + Ar+ + Ar -> Ar + Ar     : 1e-25’
  [../]
[]
```

- **species** - Denotes the active species in the reaction equations.
  1. If a species appears in an equation but not in the “species” parameter, it is ignored by the problem.
  2. All reactants must be either nonlinear or Auxiliary variables.
- **reactions** - Lists all reactions to be solved, with rate coefficients separated by a ‘:‘ character.
- **file_location** - CRANE includes the option of having rate coefficients tabulated as a function of Auxiliary variables.
  1. *ChemicalReactions* looks up a corresponding file for each reaction with a ‘EEDF‘ rate coefficient. In this case, it will look for a file named ‘reaction_e + Ar − > e + e + Ar+.txt‘ in the directory ‘RateCoefficients‘.

Rate Coefficients

- Three rate coefficient formats are accepted:
  1. **Constant** - denoted as a single value, e.g. $1e-25$
  2. **EEDF** - calculated externally by an EEDF software and tabulated. Denoted as 'EEDF'.
  3. **Equation** - Explicit equations may be denoted by brackets. e.g. $\{(6.06e-6/Tgas)*exp(-15130.0/Tgas)\}$
- *ChemicalReactions* automatically recognizes each format and adds the appropriate Auxiliary kernels.
- BOLSIG+ is the recommended Boltzmann solver for EEDF rate coefficients. An open source python equivalent, BOLOS, may also be used.

Tabulated Rate Coefficients

- If an ‘EEDF’ rate coefficient is included, CRANE will search for files of tabulated rate coefficients in the location specified by **file_location**
- Rate coefficient files are typically stored in /problems/: `/projects/crane/problems/[RateCoefficientsFile]`
- If no file is specified, CRANE will search the `/projects/crane/problems` directory for the necessary files
- Files must be named after the reaction they are applied to, e.g.:
  - Reaction: `e + Ar -> e + e + Ar+ :  EEDF`
  - File: `reaction_e + Ar -> e + e + Ar+.txt`

Tabulated Rate Coefficients (cont.)

- Rate coefficient units are up to the user; however, make sure that the units are consistent with the units being used for the nonlinear variables in the system (if species densities are measured in $\#cm^{-3}$, rate coefficients must be s^{-1}, $cm^3 s^{-1}$, $cm^6 s^{-1}$.
- The sampling variable can be either electron temperature (eV) or reduced electric field (Vm^2), both computed by BOLSIG+.
  - In either case, the sampling variable must be a nonlinear or auxiliary variable in the CRANE input file.

Kernels

- The suite of Kernels available in CRANE includes treatment of one-, two-, and three-body reactions.
- ν is the stoichiometric coefficient, k is the rate coefficient, and n_i is the species concentration.

ScalarKernel Name	**Governing Equation**	**Reaction Equation**
[Product/Reactant]1BodyScalar	$\nu k n_A$	$A \xrightarrow{k} B$
[Product/Reactant]2BodyScalar	$\nu k n_A n_B$	$A + B \xrightarrow{k} C + D$
[Product/Reactant]3BodyScalar	$\nu k n_A n_B n_C$	$A + B + C \xrightarrow{k} D + E + F$

Installing CRANE

1. Ensure MOOSE is installed correctly and the MOOSE environment is being used
2. Clone from source repository (Laboratory of Computational Physics):

   ```
   cd ~/projects
   git clone https://github.com/lcpp-org/crane
   ```

3. Compile CRANE:

   ```
   cd ~/projects/crane
   make
   ```

4. Run tests

   ```
   ./run_tests
   ```

- MOOSE input files are typically stored in your application's /problems/ directory
- To run a CRANE simulation, simply navigate to the /problems/ directory and run:

```
cd ~/projects/crane/problems
../crane-opt -i [input_file_name].i
```

Simulation Examples

- CRANE may be used for any general system of ODEs
- The following examples focus on solving reaction networks both within plasma discharges and in more general cases:
  1. **Predator-Prey** - basics of running CRANE
  2. **Collisional-Radiative Model** - stiff and nonstiff ODEs
  3. **Argon chemistry** - equation-based rate coefficients; auxiliary variables; reading tabulated data; adaptive timestepping
  4. **Nitrogen chemistry** - reading time-dependent data; including auxiliary species in reaction equations
  5. **Two-reaction case** - hands-on example

Example 1: Predator-Prey Equations

- As a simple example, consider the Lotka-Volterra (predator-prey) equations:

$$\begin{aligned}\frac{dx}{dt} &= ax - bxy \\ \frac{dy}{dt} &= -cy + dxy\end{aligned}$$

- In this problem, x and y are the nonlinear variables that will be solved by CRANE.
- a, b, c, and d are the “rate coefficients”.

Example 1: Predator-Prey Equations

- In order to be solved by CRANE, the problem must be cast into a system of “reactions”:

$$\begin{aligned} \frac{dx}{dt} &= ax - bxy \\ \frac{dy}{dt} &= -cy + dxy \end{aligned}$$

$$\begin{aligned} x &\xrightarrow{a} 2x \\ x + y &\xrightarrow{b} y \\ y &\xrightarrow{c} z \\ x + y &\xrightarrow{d} x + 2y \end{aligned}$$

- Problem statement:

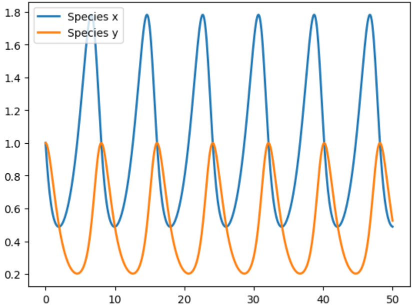
$$\begin{aligned} &t = [0, 50] \\ &x(0), \quad y(0) = 1 \\ &[a, b, c, d] = [\tfrac{2}{3}, \tfrac{4}{3}, 1, 1] \end{aligned}$$

Example 1: Predator-Prey Equations - Running

- The problem input file is located in the `problems` directory, named `example1.i`
- To run, navigate to the problems directory and provide the input file to the executable:
  1. Navigate to the problems directory:
     ```
     cd ~/projects/crane/problems
     ```
  2. Locate the input file:
     ```
     problems/example1.i
     ```
  3. Run the problem with CRANE:
     ```
     ../crane-opt -i example1.i
     ```
  4. Check output files:
     ```
     problems/example1_out.csv
     ```
  5. Plot results:
     ```
     python example1_plot.py
     ```

Example 1: Predator-Prey Equations Results

- Outputs may be stored in either CSV or Exodus format; for scalar problems, CSV is convenient for immediate viewing and plotting
- The results show a typical predator-prey system, whereby population ‘x’ is consumed by ‘y’, ‘y’ becomes saturated, and ‘x’ begins to rise again as ‘y’ decreases

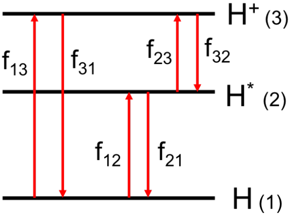


Example 2: Collisional-Radiative Model

- CRANE was primarily designed to solve chemical reaction networks in plasmas, but it is equally capable of solving general ODE systems as well.
- This example shows a simplified three-level hydrogen system for assumed sets of rate coefficients [1]
- The model is cast into an ODE system of three mass fractions with six excitation, deexcitation, and ionization frequencies

[1] Rehman, T. (2018). "Studies on plasma-chemical reduction." Eindhoven: Technische Universiteit Eindhoven.

Example 2: Collisional-Radiative Model



- The system corresponds to three electron-impact reactions:

$$H + e \leftrightarrow H^* + e$$
$$H * + e \leftrightarrow H^+ + e + e$$
$$H + e \leftrightarrow H^+ + e + e$$

Example 2: Collisional-Radiative Model

	s^{-1}	
Frequency	Stiff	Nonstiff
f_{12}	2.7×10^{10}	9×10^{1}
f_{13}	9.0×10^{8}	1×10^{2}
f_{23}	1.0×10^{6}	5×10^{1}
f_{32}	7.5×10^{4}	3×10^{1}
f_{21}	3.8×10^{1}	1×10^{1}
f_{31}	1.7×10^{2}	2×10^{1}

- The system may be cast from a system of three reactions (excitation, deexcitation, and ionization) into an ODE system:

$$\frac{dy_1}{dt} = -(f_{12} + f_{13})y_1 + f_{21}y_2 + f_{31}y_3$$
$$\frac{dy_2}{dt} = f_{12}y_1 - (f_{23} + f_{21})y_2 + f_{32}y_3$$
$$\frac{dy_3}{dt} = f_{13}y_1 + f_{23}y_2 - (f_{31} + f_{32})y_3$$

$$y_1 \xrightarrow{f_{12}} y_2$$
$$y_1 \xrightarrow{f_{13}} y_3$$
$$y_2 \xrightarrow{f_{21}} y_1$$
$$y_3 \xrightarrow{f_{31}} y_1$$
$$y_2 \xrightarrow{f_{23}} y_3$$
$$y_3 \xrightarrow{f_{32}} y_2$$

Example 2: Collisional-Radiative Model - Running

- The problem input file is located in the `problems` directory, named `example2.i`
- To run, navigate to the problems directory and provide the input file to the executable:
  1. Navigate to the problems directory:

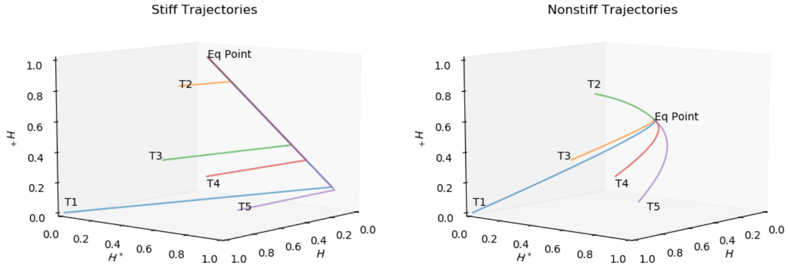
     ```
     cd ~/projects/crane/problems
     ```
  2. Locate the input file:

     ```
     problems/example2.i
     ```
  3. Run the problem with CRANE:

     ```
     ../crane-opt -i example2.i
     ```
  4. Check output files:

     ```
     problems/example2_out.csv
     ```
- Both the stiff and nonstiff cases were run with five different initial concentrations (y_0, y_1, y_2):
  1. T1: (1.0, 0, 0)
  2. T2: (0.2, 0.05, 0.75)
  3. T3: (0.5, 0.2, 0.3)
  4. T4: (0.4, 0.4, 0.2)
  5. T5: (0.4, 0.6, 0)

Example 2: Collisional-Radiative Model Results



- The figures show the phase space in both the stiff and nonstiff cases.
- Starting at several different initial conditions, the trajectories end up at the theoretical equilibrium points
  1. Stiff: $[6.62e-10, 0.00698, 0.930]$
  2. Nonstiff: $[0.0769, 0.385, 0.538]$

Example 3: Microcathode Argon Discharge

- **ZDPlasKin example**[2]: 13 reactions between 5 species ($Ar, Ar^{+}, Ar^{2+}, Ar^{*}, e^{-}$)
- This example shows the capability of CRANE to use expressions for both rate coefficients and other variables in the system (in this case, reduced electric field)
- All species are nonlinear variables
- Reduced electric field is calculated based on electron mobility, gas density, and physical domain
- Auxiliary variables: reduced electric field, mobility, current (used in reduced electric field calculation)
- Electron-impact rate coefficients and electron mobility calculated by *Bolsig+* (results pre-computed and available in /projects/crane/problems/Example3)

[2] https://www.zdplaskin.laplace.univ-tlse.fr/micro-cathode-sustained-discharged-in-ar/

Example 3: Input Parameters

- Initial Conditions:
  - $e, Ar^+, Ar^* = 1e6cm^{-3}$
  - $Ar = 3.21883e18cm^{-3}$ ($p = 100$ Torr)
  - $Ar^{2+} = 1.0cm^{-3}$
  - $d = 0.004cm$, $r = 0.004cm$, $R = 1e5\Omega$
- $T_{gas} = 300K$, $t = [0, 1ms]$
- $\left(\frac{E}{N}\right)^n = V/\left(d + R * J/\left(\left(\frac{E}{N}\right)^{n-1} * n_{Ar}\right)\right)/n_{Ar}$
- $J = \left(\left(\frac{E}{N}\right)^n * \mu^n * n_{Ar}\right) * q_e * \pi r^2 * n_e$
- Input file: 'example3.i'

Example 3: Calculating Parameters

```
[./reduced_field_calculate]
  type = ParsedAuxScalar
  variable = reduced_field
  constant_names = ‘V d qe R’
  constant_expressions = ‘1000 0.004 1.602e-19 1e5’
  args = ‘reduced_field Ar current’
  function = ‘V/(d+R*current/(reduced_field*Ar*1e6))/(Ar*1e6)’
  execute_on = ‘TIMESTEP_END’
[../]
```

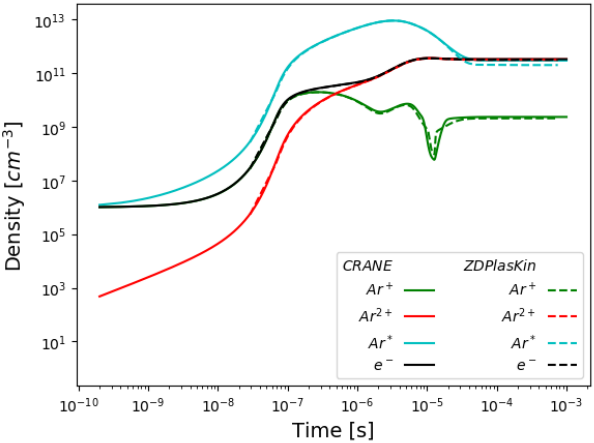
- Reduced electric field and current may be calculated by MOOSE as Auxiliary variables, which may be coupled into problems like any other variable
- ‘**constant_names**’ and ‘**constant_expressions**’ supplies the function with names and values for each constant appearing in the functions
- **‘args’** - nonlinear or auxiliary variable values that must be accessed by function
- **‘execute_on’** - Tells MOOSE when to run the AuxKernel

Example 3: Reading Parameters From File

```
[./mobility_calculation]
  type = DataReadScalar
  variable = mobility
  sampler = reduced_field
  property_file = ‘Example3/electron_mobility.txt’
  execute_on = ‘INITIAL TIMESTEP_BEGIN’
[../]
```

- Electron mobility and temperature are both read from data files tabulated as a function of reduced electric field (calculated by *Bolsig+*)
- The ‘DataReadScalar’ AuxKernel may be used to pull this data from the files
- ‘property_file’ tells MOOSE where to find the tabulated data file

Example 3: Results



Example 4: Nitrogen Reaction Network

- **ZDPlasKin example**[3]: 34 reactions between 11 species
- In this example, electric field and electron density are read from files containing time-varying data
- Electron temperature is calculated from *Bolsig+*, as are electron-impact rate coefficients
- In this case, electrons *are not considered a nonlinear species* and do not contribute to the jacobian
- Reduced electric field is calculated based on electron mobility, gas density, and physical domain

[3] https://www.zdplaskin.laplace.univ-tlse.fr/external-profiles-of-electron-density-and-electric-field/

Example 4: Input Parameters

- Initial Conditions: $N_2(t = 0) = 2.447464e19$; all other species start with $n_i(0) = 0cm^{-3}$
- $T_{gas} = 300K$, $t = [0, 2.5ms]$
- $T_{eff} = T_{gas} + \left(0.12 * \left(\frac{E}{N} * 1e21\right)^2\right)$
- Reduced field, electron temperature, and electron density pulled from tabulated data
- Input file: ‘example4.i’

Example 4: Reading Time Data From File

```
[AuxKernels]
  [./field_calculation]
    type = DataReadScalar
    variable = reduced_field
    use_time = true
    property_file = ‘Example3/reduced_field.txt’
    execute_on = ‘TIMESTEP_BEGIN’
  [../]
[]
```

- ‘DataReadScalar’ AuxKernel is used as in Example 3
- Setting ‘use_time’ to ‘true’ tells MOOSE to sample from the tabulated data set using the time variable

Example 4: Scaling Data Read From File

```
[AuxKernels]
  [./field_calculation]
    type = DataReadScalar
    variable = Te
    scale_factor = 1.5e-1
    sampler = reduced_field
    property_file = ‘Example4/electron_temperature.txt’
    execute_on = ‘TIMESTEP_BEGIN’
  [../]
[]
```

- Electron temperature for this problem is also read from a data file, but sampled based on the ‘reduced_field’ AuxVariable rather than time
- The **scale_factor** parameter is multiplied into the result after being read from the data file. This allows the user to convert the data to their desired units.

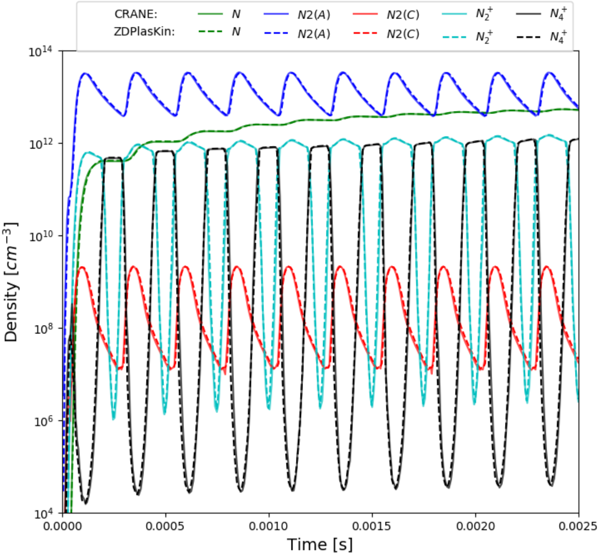
Example 4: Action Parameters

```
[ChemicalReactions]
  [./ScalarNetwork]
    species = ‘e N N2 N2A N2B N2a1 N2C N+ N2+ N3+ N4+’
    aux_species = ‘e’
    file_location = ‘Example4’

    # These are parameters required equation-based rate coefficients
    equation_variables = ‘Te Teff’
    rate_provider_var = ‘reduced_field’
    reactions = ‘...’
  [../]
[]
```

- Since electrons are being read from a data file, they are not a nonlinear species. The **aux_species** parameter allows auxiliary variables to be coupled into the reaction network.
- Variables that are used in any equation-based rate coefficients must be listed in the **equation_variables** parameter

Example 4: Results



Example 5 (Hands On): Two Argon Reactions

- So far you have seen 4 pre-made examples of using CRANE. All of the input files are already in place, so you can ensure that the results are reproduced on your own machine(s).
- ...but the point of this code is to let you use it for your own applications!
- This next example will be **hands-on** - try to write your own input file!
- A "skeleton" input file is included in `example5.i`. **You will need to add scalar variables, scalar kernels (for time derivatives), and the two reactions.**

Example 5 (Hands On): Two Argon Reactions

- 2 reactions between 3 species (Ar, Ar+, e-)
- **ZDPlasKin example**[4]: Two Reaction Test Case

$$e + Ar \xrightarrow{k_1} 2e + Ar^+$$

$$e + Ar^+ + Ar \xrightarrow{k_2} 2Ar$$

- $\frac{E}{N} = 50\,Td \rightarrow 50 \times 10^{-21}\,Vm^2$
- $T_{gas} = 300K$, $t = [0, 0.25\mu s]$
- With these parameters, this becomes a system of three coupled ODEs with constant rate coefficients
- The reduced electric field is added as an Auxiliary variable

[5] https://www.zdplaskin.laplace.univ-tlse.fr/two-reaction-test-case-start-here/

Example 5 (Hands On): Two Argon Reactions

- Input Parameters:

$$e + Ar \xrightarrow{k_1} 2e + Ar^+$$

$$e + Ar^+ + Ar \xrightarrow{k_2} 2Ar$$

  - $*$ $n_{Ar}(t=0) = 2.5e19[cm^{-3}]$
  - $*$ $n_e(t=0), n_{Ar^+}(t=0) = 1.0[cm^{-3}]$
  - $*$ $k_1 =$ {EEDF convolution}
  - $*$ $k_2 = 1e-25 \quad [cm^6 s^{-1}]$
  - $*$ $t = [0, 0.25\mu s]$

- **Instructions**:
  1. Add scalar variables (e^- is already filled in for you)
  2. Add scalar kernels (`type = ODETimeDerivative`)
  3. Fill in ChemicalReactions action (2 reactions)
     - Rate constant k_1 tabulated as a function of reduced electric field included in 'projects/crane/problems/Example5/'

Example 5 (Hands On): Mesh and Variables

- First a “dummy” mesh is needed to run a problem in MOOSE (dim = 1 with nx = 1 is sufficient)
- Three variables are considered in this problem: electrons, argon ions, and argon neutrals (background gas)
- An example of the Mesh and Variables are shown on the right. Note that you will need to add two more Variables
- Initial conditions:
  - e^{-}: 1
  - Ar^{+}: 1
  - Ar: 2.5×10^{19}

```
[Mesh]
  type = GeneratedMesh
  dim = 1
  xmin = 0
  xmax = 1
  nx = 1
[]

[Variables]
  [./e]
    family = SCALAR
    order = FIRST
    initial_condition = 1
  [../]
[]
```

Example 5 (Hands On): Kernels and AuxVariables

- Time derivative kernels must be added for each variable in the system. (Note that you will need to add two more to the example shown here)
- One of the reaction rate coefficients is read from a tabulated file. In general these need to be computed manually by the user through BOLSIG+ or an equivalent software.
- In this case, the reaction file already exists in the `/problems/Example5` directory

```
[ScalarKernels]
  [./de_dt]
    type = ODETimeDerivative
    variable = e
  [../]
[]

[AuxVariables]
  [./reduced_field]
    order = FIRST
    family = SCALAR
    initial_condition = 50e-21
  [../]
[]
```

Example 5 (Hands On): Kernels and AuxVariables cont.

- The data is tabulated with respect to the reduced electric field, Vm^{-2}. **The sampling variable must be a variable in the system, either Nonlinear or Auxiliary.**
- With this in mind, we will create a (constant) `AuxVariable` called `reduced_field`, with an initial condition of $50 \times 10^{-21} Vm^{-2}$

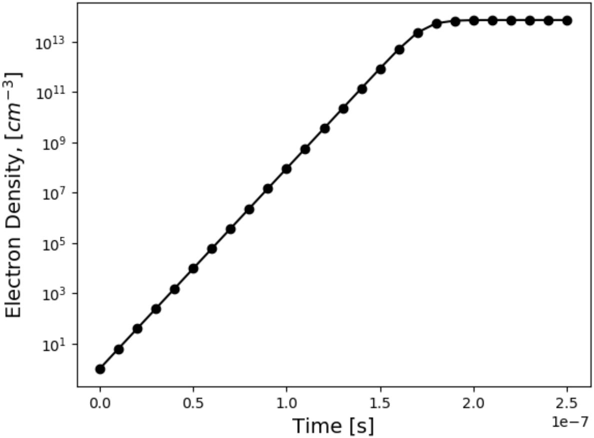
```
[ScalarKernels]
  [./de_dt]
    type = ODETimeDerivative
    variable = e
  [../]
[]

[AuxVariables]
  [./reduced_field]
    order = FIRST
    family = SCALAR
    initial_condition = 50e-21
  [../]
[]
```

Example 5 (Hands On): Two Argon Reactions - Running

- To run, navigate to the problems directory and provide the input file to the executable:
  1. Navigate to the problems directory:
     ```
     cd ~/projects/crane/problems
     ```
  2. Locate the input file:
     ```
     problems/example5.i
     ```
  3. Run the problem with CRANE:
     ```
     ../crane-opt -i example5.i
     ```
  4. Check output files:
     ```
     problems/example5_out.csv
     ```
  5. Plot results:
     ```
     python example5.py
     ```

Example 5 Results



Contribute!

- CRANE is available at the Laboratory for Computational Physics github:

  `https://github.com/lcpp-org/crane`
- A repository wiki is available for more information:

  `https://github.com/lcpp-org/crane/wiki`
- Fully open source!
  1. Fork from https://github.com/lcpp-org/crane
  2. Clone to your computer:

     ```
     git clone https://github.com/[your_user_name]/crane
     ```
  3. Create a new branch
  4. Submit a pull request when a new feature is ready!

Feedback and Credits

- For questions, concerns, or comments, contact Shane Keniley:

  `keniley1@illinois.edu`

  `https://www.github.com/keniley1`

- The MOOSE google group is useful for questions about the MOOSE framework in general (feedback directly from the MOOSE developers is generally on the order of hours):

  `https://groups.google.com/forum/#!forum/moose-users`