Julia-Scripts

A small collection of Julia utility scripts and their associated wrappers that I use. The shell scripts all use the environmental variable JPROGRAM to find the julia scripts, so make sure these are in a location given by the directory to which you set JPROGRAM.

Convolving data with a Gaussian or Cauchy (Lorentz) distribution

For two functions $f, g: R \to R$, we can define their convolution as

```
(f*g)(x) \equiv \int\limits_{-\infty}^{\infty} f(x')g(x-x')dx'
```

The assumption here is that the data to convolve represents the output of some function that is zero for non-positive arguments. The lower integration limit can then be truncated to 0.

```
(f*g)(x) \equiv \int\limits_{0}^{\infty} f(x')g(x-x')dx'
```

I made this for functions f being scattering cross section functions $\sigma(E)$, where E is some positive energy. There are two posibilities coded for g

The Gaussian Distribution

The gaussian distribution is given by

```
e^{-(x'-x)^2 / (2\gamma^2)}
```

Cross sections $f(x)$ can be convolved with this distribution:

```
\tilde f(x)
=
\frac{
    \int\limits_{0}^{\infty} dx' \sigma(x') e^{-(x - x')^2 / (2\gamma^2)}
}{
    \int\limits_{0}^{\infty} dx' e^{-(x - x')^2 / (2\gamma^2)}
}
=
\frac{1}{\gamma\sqrt{2\pi}}
\int\limits_{0}^{\infty} dx' \sigma(x') e^{-(x - x')^2 / (2\gamma^2)}
```

In implementation, the normalization factor $\int_0^\infty dx' e^{-(x-x')^2/\left(2\gamma^2\right)}$ is computed numerically

The Cauchy (Lorentz) Distribution

The Cauchy — or Lorentz — distribution is given by

```
f \frac{\pi}{\gamma}
\left[(x - x')^2 + \gamma^2\right] ^ {-1}
```

where γ is is the distribution width.

Implementation

In practice, the upper integration limit is taken to be the last available value of x. This should not be an issue if the Gaussian width, γ, is small enough compared to this. The lower limit, 0, may not be in the domain of f or may not return a desirable value. Convolving such data must be done carefully, depending on the function $f(x)$.

The shell script `convolve` is a wrapper script for the Julia script `convolve.jl`. Script arguments

Using `convolve`

```
convolve

    Wrapper for my julia script located at : \$JPROGRAM

    usage: convolve [operation] [operand]

    operations:               operand:        function:

        -h                    none            show this message.
        -i,--input            file            specify input  file
        -o,--output           file            specify output file
        --dx,--width          float           specify gaussian width
        --nx                  integer         number of convolution x-grid points
        --logx                none            if supplied, a logarithmic grid will b
        -t, --ct, --convtype  (gauss|cauchy)  determines convolution function

    The following is a valid implementation of convolve :

        convolve -i data_to_convolve.dat -o convolved_data.dat --dx=1.3e-3 --nx=1000
```

Convolve scattering electron collision cross sections with a Maxwell-Boltzmann distribution

In the situation where we have scattering cross sections (electron-atom or electron-molecule) as a function of collision energy and want to obtain thermally averaged rate coefficients as a fucntion of the kinetic temperature, the script `thermal` can be used. Just as above, this reduces to convolving a function $f(x)$ with a function $g(x)$. Here, the function f is assumed to be the function $\sigma(E)$ — a scattering cross section as a function of collision energy (E). The function $\sigma(E)$ is zero for negative E, but may be zero or undefined at $E = 0$. In practice, the value $E = 0$ is approached but not passed exactly. Some cross sections are 0 at threshold ($E = 0$), while others behave as $1/E$. The choice of assumed behavior can be given to the script, whose usage is given below.

The Maxwell-Boltzmann Distribution

The Maxwell-Boltzmann distribution is a probability distribution, typically used for describing particle speeds in an ideal gas. In three dimensions, It is given by

```
\left( \frac{m}{2\pi kT} \right)^{\frac{3}{2}} e^{-\frac{mv^2}{kT}}
```

where v is the particle speed, m is the particle mass, k is Boltzmann's constant, and T is the kinetic temmperature. Given cross sections $\sigma(E)$, we can use the Maxwell-Boltzmann distribution to obtain thermal rate coefficients (rate coefficients in the case where we have a gas of particles at a temperature T). The cross sections are first convolved with the Maxwell-Boltzmann distribution to obtain rate coefficients at a given temperature, given by

```
\alpha(T) = \frac{
    \int\limits_{0}^{\infty} \sigma(E) \sqrt{2E/m_{e}} \sqrt{E} e^{-E/kT} dE
}{
```

```
    \int\limits_{0}^{\infty} \sqrt{E} e^{-E/kT} dE
}
```

which should then be averaged over initial states. However, this is not done by the script (this requires information like state energies and rotational quantum numbers, so this can be done separately [1]) The normalization factor is also computed numerically.

For cross sections behaving as $1/E$, the low-energy (near-0) part of the integral is very important, especially at lower temperatures. In the case where the cross section data is not available at a low enough energy, the cross sections can be extrapolated assuming a $1/E$ behavior based on the datum corresponding to the lowest available energy. This is sensitive whether that point is a resonance and obviously does not take into account resonances that might be present at lower energies. Ideally, this is done with data that was obtained at low enough scattering energies such that there are no resonances. See the following for more detail on the usage.

Using `thermal`

This script requires my script `units` (https://github.com/banana-bred/units).

`thermal`

```
    Wrapper for my julia script (\$JPROGRAM), which produces state-selected
    kinetic rate coefficients from cross sections behaving as 1/E (E being electron
    at the E=0 threshold.

    usage: thermal [operation] [operand]

    operations:                operand:   function:

        -h                     none       show this message.
        -i,--input             file       specify input  file
        -o,--output            file       specify output file
        --logx                 none       if supplied, a logarithmic grid will be use
        --Ti                   float      lowest  kinetic temperature (K)
        --Tf                   float      highest kinetic temperature (K)
        --nT                   integer    number of kinetic temperatures
        --input-xs-units       string     the unit type of the input data   (e.g., "c
                                          Output rates will be in these units ^3 / s
        --input-energy-units   string     the unit type of the input energy (e.g., "e
        --extrap               none       extrapolate cross sections to E = 0 thresho
                                          assume it is zero.
        --electron-energy-min  float      lowest electron energy (should be closer to
        --num-extrap-energies  integer    number of extrapolation energies (extrapola

    The following would be a valid implementation of thermal :

        thermal -i data_to_convolve.dat -o convolved_data.dat \
                  --logx \
                  --Ti=1e-6 \
                  --Tf=1e3 \
                  --extrap \
                  --electron-energy-min=1e-8 \
                  --num-extrap-energies=1000 \
```

```
                --nT=500 \
                --input-xs-units=cm \
                --input-energy-units=eV

    Input file: a file of electron energies (Eel) and cross sections (σ)
    in any units of energy and length^2, formatted as

      Eel  σ
       .   .
       .   .
       .   .

    Output file: a file of temperatures (T) in K and rate coefficients (α) in cm^3/s

      T  α
      .  .
      .  .
      .  .
```

Fitting interatomic / intermolecular potentials

Given data that (hopefully) resembles an interatomic or intermolecular potential, the script `morseFit` can be used to fit the data to a Morse potential. One case in which this might be useful is when such a potential is calculated and needs to be extrapolated to larger distances.

The Morse Potential

The Morse potential is often used as an improved approximation to the harmonic oscillator model for molecular vibration. It is given by

```
V(r) = D_e \left( e^{-2a(r - r_e) - 2e^{-a(r - r_e)}} \right) + E_{\text{diss}}
```

where r is a separation distance, D_e is the well depth with respect to the dissociation limit, r_e is the equilibrium distance, and $E_{textdiss}$ is the dissocation limit of the potential.

Implementation

The scripts take as input a file containing space separated x (r), y ($V(r)$) data. A fit can be generate of all of the data or only a part of the data for $r > r_{min}$. This is useful for fitting only the tail of the data. The optimized parameters (using the method of least squares) can then be used to generate data over a user-defined range of r values. Should the `--tailonly` option be supplied, the script will append the fitted data to the supplied data. A decent guess for E_{diss} should be supplied, otherwise the least squares method might not work.

Using `morseFit`.

```
morseFit

    Wrapper for my julia script located at (\$JPROGRAM)

    usage: morseFit [operation] [operand]

    operations:                 operand:        function:
```

```
-h                          none            show this message.
-i,--input                  file            specify input  file
-o,--output                 file            specify output file
--tailonly                  none            if specified, only fit the tail of t
--rcut                      number          cutoff   distance for fitting only t
--rmin                      number          smallest distance for the fit. Ignor
--rmax                      number          largest  distance for the fit
--rstep                     number          linear step size  for the fit
-a                          number          initial guess for the 'a'  parameter
-r0                         number          initial guess for the 'r0' parameter
-D                          number          initial guess for the 'D'  parameter
--dissociation-limit        number          initial guess for the absolute value
--positive-dissoc-limit     none            if specified, the dissociation limit
--print-parameters          none            if specified, print the fit paramete
```

The following is a valid implementation of morseFit :

```
morseFit -i input.dat -o output.dat --tailonly --rcut=2.6 --rstep=0.01 --rma
```

For a full fit: `morseFit -i input.dat -o output.dat --rmin=0.8 --rstep=0.01 --rmax=10 --dissociation-limit 75 --positive-dissoc-limit`

For a tail fit: `morseFit -i input.dat -o output.dat --rcut=2.6 --rstep=0.01 --rmax=10 --dissociation-limit 75 --positive-dissoc-limit`

References

[1] Forer, J. et. al. (2023) *Kinetic rate coefficients for electron-driven collisions with CH: dissociative recombination and rovibronic excitation.* Monthly Notices of the Royal Astronomical Society 527, 5238-5234