UNIVERSITY OF BIRMINGHAM


**Department of Metallurgy & Materials**

# EAMPA Manual

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 What does it do?

This code takes a potential function or set of functions as an input. Given a set of atom configurations this code will:

- calculate the energy of the system, forces between atoms and stress on the system
- equation of state
- elastic constants

If the experimental (or DFT calculated) energies, forces, stresses and other properties are known, the difference between the properties predicted by the interatomic potential and the known values may be calculated.

After selecting a form of each function that constitutes the potential, and setting reasonable upper and lower bounds for the parameters of the function, the parameters of the potential functions will be adjusted to minimise the difference between the known and predicted properties.

### 1.1.1 Properties of an Interatomic Potential

It may then be used to calculate the bulk properties of a system where the interatomic potentials are used to calculate the force between atoms and the energy of the system. These properties include:

- equation of state $V0, E0, B0, B'0$
- 9 orthorhombic elastic constants
- Young's modulus ($e$), shear modulus ($g$) and Poisson's ratio ($\nu$)

# Chapter 2

# Background

## 2.1 Embedded Atom Method

### 2.1.1 EAM Potential

$$U_{EAM} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} V_{ij}(r_{ij}) + \sum_{i=1}^{N} F[\rho_i] \qquad \text{where } \rho_i = \sum_{j=i,j\neq i}^{N} \rho_{ij}(r_{ij}) \tag{2.1}$$

### 2.1.2 Two Band EAM Potential

$$U_{EAM} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} V_{ij}(r_{ij}) + \sum_{i=1}^{N} F_D[\rho_{d,i}] + \sum_{i=1}^{N} F_S[\rho_{s,i}] \qquad \text{where } \rho_{d,i} = \sum_{j=i,j\neq i}^{N} \rho_{d,ij}(r_{ij}) \text{ and } \rho_{s,i} = \sum_{j=i,j\neq i}^{N} \rho_{s,ij}(r_{ij}) \tag{2.2}$$

This allows a second embedding functional and electron density function to add/subtract energy to an atom when mixed as an alloy, but reverts to the original EAM for that element when in local concentrations of like-atoms.

### 2.1.3 Many Bands for the Embedded Atom Method

$$U_{EAM} = V_{pair} + \sum_{b}^{B} F_b(\rho_b)\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} V_{ij}(r_{ij}) + \sum_{b=1}^{B}\sum_{i=1}^{N} F_b[\rho_{b,i}] \qquad \text{where } \rho_{b,i} = \sum_{j=i,j\neq i}^{N} \rho_{b,ij}(r_{ij}) \qquad \text{and B total number of bands, 1, 2, 3 etc} \tag{2.3}$$

### 2.1.4 Calculating Energy, Force and Stress

Molecular dynamics simulations do not need to include the weak or strong force, and the gravitational force between atoms in the simulated material are so weak in comparison the the electromagnetic force, they can also be ignored. There is a force between all the atoms within a real material, but the electromagnetic force is inversely proportionally to the seperation of the atoms. Above a certain separation, the electromagnetic force can also be ignored, in order to simplify the computation.

#### Neighbour List

A cutoff radius can be introduced to limit the number of neighbours. As the lattice parameter decreases, the atoms are brought closer together, and as the cutoff radius is increased more atoms are considered to be within the sphere of influence of one another. Both increase the number of neighbours each atom has.

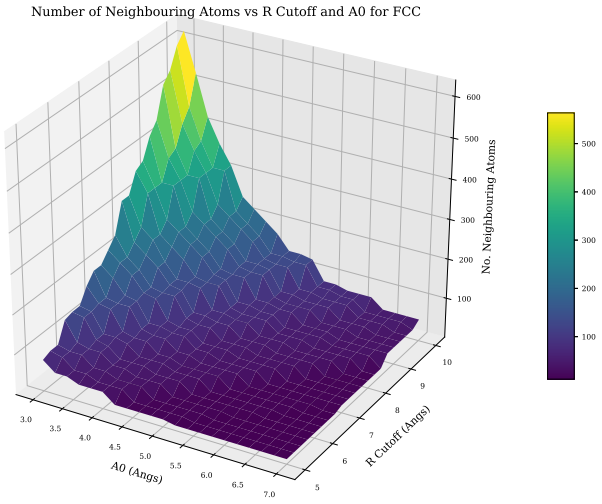
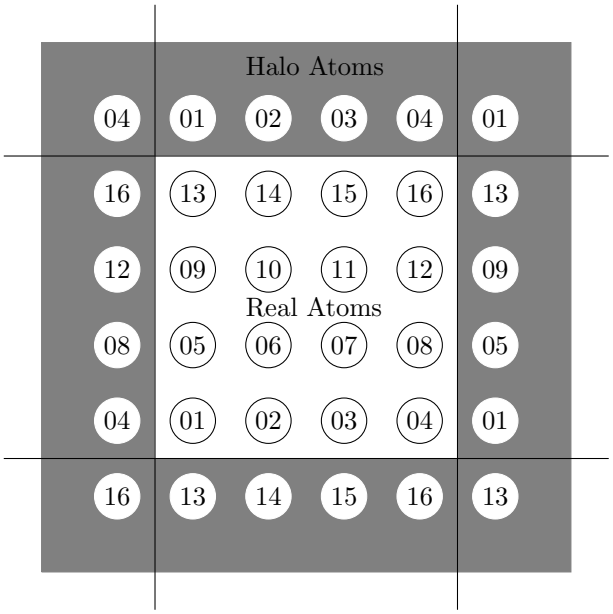



Figure 2.1: Graph caption

Building a neighbourlist may take a long while, depending on the parameters. If a simplistic approach is taken, for N atoms in the supercell, there will be $27N^2$ checks between atoms to see whether they are within the cutoff radius of one another. For larger numbers of atoms in a supercell, the whole may may be decomposed into smaller domains.

For example, a 16x16x16 FCC supercell, containing 16,384 atoms, would require looping $27 \times 16,384^2 = 7.25 \times 10^9$ times. Breaking the supercell into 64 smaller domains with 256 atoms in each, reducing the problem to $64 \times 27 \times 256^2 = 1.13 \times 10^8$ loops.

For this program, the supercells used to calculate bulk properties from the interatomic potentials, and the configurations generated by DFT, contain fewer than 1,000 atoms, removing the immediate need add subroutines to decompose the configuration into smaller subcells.




To build the neighbour list a halo configuration is created such that it lies on top of the real atoms, but extends at least as far as the cut off radius on each side of the real simulation box. Periodic boundary conditions are used to construct the halo.

Once the halo list is computed, the neighbour list may also be computed by looping through the real atoms and halo atoms. A simple pseudo coded sub routine is given below.

Listing 2.1: Simple genetic optimisation subroutine

```
1
2  // real_atoms - array holding coordinates of all real atoms
3  // halo_atoms - array holding coordinates of all halo atoms
4  // real_ids - unique id for each atom
5  // halo_ids - unique atom ids for halo atoms
6  // nlist_ids - array to store ids
7  // nlist_r - array to store separation
8
9  // Define the cutoff and counter start value
10 r_cut = 5.0
11 nl_counter = 1
12
13 // Calculate square
14 r_cut_sq = r_cut ** 2
15
16 // Loop over all real atoms
17 DO n_real = 1, real_atom_count
18   // Loop over all halo atoms
19   DO n_halo = 1, halo_atom_count
20     IF (real_ids(n_real) .LT. halo_ids(n_halo)) THEN
21       r(1:3) = halo_atoms(n_halo, :) - real_atoms(n_real, :)
22       r_sq = SUM(r(1:3) * r(1:3))
23       IF (r_sq .LE. r_cut_sq) THEN
24         r_mag = sqrt(r_sq)
25         nlist_ids(nl_counter, 1) = real_ids(n_real)
26         nlist_ids(nl_counter, 2) = halo_ids(n_halo)
27         nlist_r(nl_counter, 1) = r_mag
28         nlist_r(nl_counter, 2:4) = r(1:3)/r_mag
29         nl_counter = nl_counter + 1
30       END IF
31     END IF
32   END DO
33 END DO
```

The resulting list will contain unique a pair combinations; i.e. the pair atom 1 and atom 2 will only be recorded once, and not also as atom 2 and atom 1.

### Computing Total Energy

The total energy of the system is the sum of the individual energies for the atoms in the simulation. The type of atom (or pairs of atoms) will determine which function is used.

First, to compute the pair potentials:

- set the total energy of the system equal to zero
- set the starting energy for each atom in the system to zero
- loop through the atom pairs in the entire neighbour list
- for each atom pair, A and B, use the known separation and the potential function to compute the potential energy on atom A due to B and vice versa
- add this potential energy to both A and B
- after looping through the neighbour list, add all the energies due to the pair potential to the total energy

For an EAM or 2BEAM potential, the densities and embedding energies must also be computed:

- set the electron density at the position of (for each) atom to zero
- loop through the atom pairs in the entire neighbour list
- for each atom pair, using the density function for the atom A, the density at atom B due to atom A will be calculated and added to the density at atom B
- if the atoms are both of the same type the same density will be added to atom A due to atom B
- if the atom types are different, using the density function for the atom B, the density at atom A due to atom B will be calculated and added to the density at atom A
- following looping through the neighbour list, to calculate the densities at each atom, the list of atoms will be looped through
- for each atom, the density value at the position of that atom, will be input into the appropriate embedding function to calculate the embedding portion of the energy
- add all the embedding energies to the total energy of the system

For a 2BEAM potential, repeat the above procedure for the second group of density functions and embedding functions (do not repeat calculation of the pair potentials).

### Computing Forces on Atoms

In order to calculate the forces on the atoms with an EAM or 2 Band EAM potential, the neighbour list and atom list will need to be looped through several times. First, the pair potential and force due to the pair potential must be calculated by a complete loop through the neighbour list. At the same time, the density at each atom location is also calculated. The embedding energy may then be computed by looping through all the atoms and using the electron density at each atom to give the energy of the atom embedded in that density. The third and final loop will run through the neighbour pairs in the neighbour list once more computing the force on each atom due to it's embedding.

### Computing Forces on Atoms

### Pseudo Code for the Energy, Force and Stress Subroutine

Listing 2.2: Pseudo Code for Energy and Stress Force Calculation

```
1
2  electron_density[1:n_atoms, 1:bands] = 0.0 // electron density at each atom
3  density_grad_ab[:,:] = 0.0              // grad density function
4  density_grad_ba[:,:] = 0.0              // grad density function
5  energy = 0.0                            // total energy
6  forces[:,:] = 0.0                       // forces each atom, 3D
7  force_between_pairs[:,:] = 0.0          // used to store force between pairs, for stress computation
8  stress[:,:] = 0.0D0
9
10 // LOOP 1 - Pair energy, force and calculate densities
11 DO n = 1, neighbour_count
12   E = get_PairEnergy(atom_a, atom_b, r[n,:])
13   F[:] = get_PairForce(atom_a, atom_b, r[n,:])
14
15   // Save energy
16   energy = energy + E
17
18   // Save Force
19   f[atom_a, :] = f[atom_a, :] - F[:]
20   f[atom_b, :] = f[atom_b, :] + F[:]
21   force_between_pairs[:,:] = force_between_pairs[:,:] - F[:] // Used to calculate stress
22
23   // Loop through density bands
24   DO band = 1, bands
25     // Electron density at A due to atom B
26     electron_density[atom_a, b] = get_Density(atom_b, r)
27     density_grad_ab[n, band] = get_DensityGradient(atom_b, r)
28
29     // Electron density at B due to atom A
30     electron_density[atom_a, band] = get_Density(atom_a, r)
31     density_grad_ba[n, band] = get_DensityGradient(atom_a, r)
32   END DO
33 END DO
34
35 // LOOP 2 - Embedding energy
36 DO n = 1, atom_count
37   // Loop through density bands
38   DO band = 1, bands
39     energy = energy + get_EmbeddingEnergy(n, band)
40   END DO
41 END DO
42
43 // LOOP 3 - Embedding force
44 DO n = 1, neighbour_count
45   // Loop through density bands
46   DO band = 1, bands
47     epA = get_EmbeddingGradient(atom_a, electron_density(atom_a, b))
48     epB = get_EmbeddingGradient(atom_b, electron_density(atom_b, b))
49
50     F[:] = (epA * density_grad_ba(n, band) + epB * density_grad_ab(n, band)) * r[n, :]
51
52     f[atom_a, :) = f(atom_a, :) - F[:]
53     f(atom_a, :) = f(atom_a, :) + F[:]
54
55     force_between_pairs[:,:] = force_between_pairs[:,:] - F[:] // Used to calculate stress
```

```
  END DO
END DO

// LOOP 4 STRESS
DO n = 1, neighbour_count
  // Only compute if the second atom is in the halo
  IF(nlisthalo(n))THEN
    DO i = 1,3
      DO j = 1,3
        stress[i,j] = stress[i,j] + (r[n, i] * force_between_pairs[n,j])
      END DO
    END DO
  END IF
END DO
stress[1:3, 1:3] = stress[1:3, 1:3] / (2.0 * volume)
```

## 2.2 Equation of State for Cubic Crystals

### 2.2.1 Murnaghan Equation of State

Hooke's law implies a linear relationship between stress and strain. In practice, where a pressure is applied to a material, the application of Hooke's law is limited [**murnaghaneq**]. Muraghan derived a new equation to improve upon formulae developed in the 1930's, using compression data from high pressure experiments.

$$P(V) = \frac{B_0}{B'_0}\left(\left(\frac{V_0}{V}\right)^{B'_0} - 1\right) \tag{2.4}$$

As pressure is the negative derivative of the internal energy of the system with respect to change in volume, $p = -(\partial E/dV)$, and the equation can be integrated and written in terms of the energy, volume, bulk modulus and its derivative[**crystaleos**].

$$E(V) = E_0 + \frac{B_0 V}{B'_0}\left[\left(\frac{V_0}{V}\right)^{B'_0}\frac{1}{B'_0 - 1} + 1\right] - \frac{B_0 V_0}{B'_0 - 1} \tag{2.5}$$

### 2.2.2 Birch-Murnaghan Equation of State

Several years after Murnaghan's equation, Birch developed further upon the experimental data provided by Bridgman. For cubic symmetry, the description of free energy now includes third order terms in the strain components[**birchmurnaghaneq**].

$$P(V) = \frac{3B_0}{2}\left[\left(\frac{V_0}{V}\right)^{\frac{7}{3}} - \left(\frac{V_0}{V}\right)^{\frac{5}{3}}\right]\left[1 + \frac{3}{4}(B'_0 - 4)\left(\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right)\right] \tag{2.6}$$

The energy-volume relationship may again be constructed[**crystaleos**].

$$E(V) = E_0 + \frac{9V_0B_0}{16}\left[\left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right]^3 B'_0 + \left[\left(\frac{V_0}{V}\right)^{\frac{2}{3}} - 1\right]^2\left[6 - 4\left(\frac{V_0}{V}\right)^{\frac{2}{3}}\right]\right] \tag{2.7}$$

### 2.2.3 Fitting Method

The first step is to fit a second order polynomial to the energy-volume data. This may be achieved using least-squares fitting with a vandermode matrix. The coefficients from this polynomial may then be used to calculate reasonable value for $E_0$, $V_0$ and $B_0$; sane starting values of $B'_0$ are between 1 and 10, and the code takes a starting value of 2[**gilgamesheos**].

$$
\begin{aligned}
E(V) &= c_0 + c_1 V + c_2 V^2 \\
V_0 &= -\frac{c_1}{2c_2} \\
E_0 &= c_2 * {V_0}^2 + c_1 V_0 + c_0 \\
B_0 &= 2c_2 V_0 \\
{B'}_0 &= 2
\end{aligned}
\tag{2.8}
$$

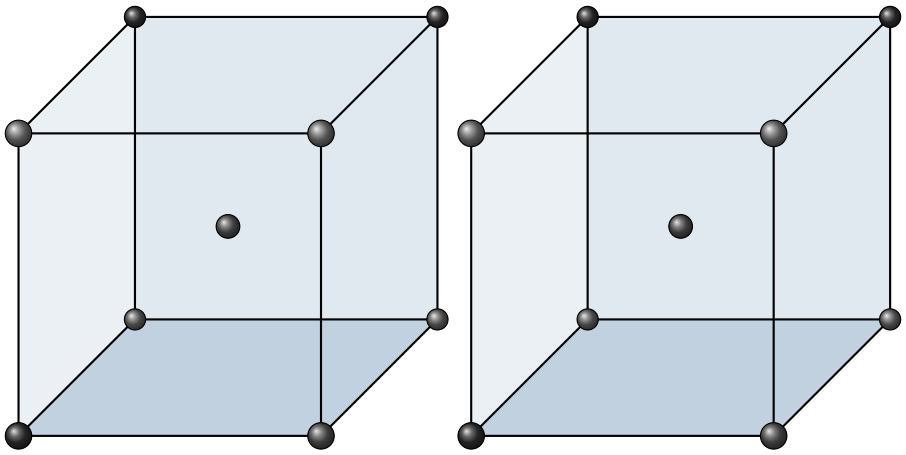
Newton Gauss is then used to minimise $E_0$, $V_0$ and $B_0$ while ${B'}_0 \in 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0$

$$
\left[J^T J\right] P = J^T R \tag{2.9}
$$

The parameters with the lowest residual square sum are returned.

### 2.2.4 Calculating Elastic Constants for Orthorhombic Crystal

The DFT work here includes Palladium and Iron. The natural arrangement of Pd atoms in a pure sample are FCC within a cubic crystal. Pure iron at room temperature is BCC, but this work is interested in austenetic stainless steel where the structure of atoms in the alloy are FCC. When modelling FCC iron using DFT with a non-polarized calculation, the crystal favours a cubic crystal with the atoms fixed in the FCC positions. When a spin-polarized calculation is computed, with magnetization along the x-axis, the crystal becomes tetragonal (once again, the atoms are fixed in FCC positions).



$$C_{ij} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{13} & C_{23} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix} \quad (2.10)$$

(9 independent values)

Once the optimised parameters have been determined for the orthorhombic crystal, nine strains are applied to the crystal [**DftTiSiRavindran**] in order to calculate the nine independent elastic constants.

$$E(V,\sigma) = E(V_0,0) + V_0\left(\sum_i \tau_i \epsilon_i \sigma_i + \frac{1}{2}\sum_{ij} c_{ij}\sigma_i \epsilon i \sigma_j \epsilon j\right) + O(\sigma^3) \quad (2.11)$$

The first three strains applied to the orthorhombic crystal

$$D_1 = \begin{bmatrix} 1+\delta & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.12)$$

$$E(V,\sigma) = E(V_0,0) + V_0\left(\tau_1\sigma + \frac{c_{11}}{2}\sigma^2\right) \quad (2.13)$$

$$D_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1+\delta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

$$E(V,\sigma) = E(V_0,0) + V_0\left(\tau_2\sigma + \frac{c_{22}}{2}\sigma^2\right) \tag{2.15}$$

$$D_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1+\delta \end{bmatrix} \tag{2.16}$$

$$E(V,\sigma) = E(V_0,0) + V_0\left(\tau_3\sigma + \frac{c_{33}}{2}\sigma^2\right) \tag{2.17}$$

Volume conserving monoclinic distortions are then applied to the crystal to calculate the $C_11$, $C_22$ and $C_33$ elastic constants.

$$D_4 = \begin{bmatrix} \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & 0 \\ 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} \\ 0 & \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \tag{2.18}$$

$$E(V,\sigma) = E(V_0,0) + V_0\left(2\tau_4\sigma + 2\frac{c_{44}}{2}\sigma^2\right) \tag{2.19}$$

$$D_5 = \begin{bmatrix} \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} \\ 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \tag{2.20}$$

$$E(V,\sigma) = E(V_0,0) + V_0\left(2\tau_5\sigma + 2\frac{c_{55}}{2}\sigma^2\right) \tag{2.21}$$

$$D_6 = \begin{bmatrix} \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ \frac{\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ 0 & 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \tag{2.22}$$

$$E(V,\sigma) = E(V_0,0) + V_0\left(2\tau_6\sigma + 2\frac{c_{66}}{2}\sigma^2\right) \tag{2.23}$$

$$D_7 = \begin{bmatrix} \frac{1+\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & 0 \\ 0 & \frac{1-\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ 0 & 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \tag{2.24}$$

$$E(V,\sigma) = E(V_0,0) + V_0\left((\tau_1 - \tau 2\sigma + \frac{1}{2}(c_{11}+c_{22}-2c12)\sigma^2\right) \tag{2.25}$$

$$D_8 = \begin{bmatrix} \frac{1+\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & 0 \\ 0 & \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ 0 & 0 & \frac{1-\sigma}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \tag{2.26}$$

$$E(V,\sigma) = E(V_0,0) + V_0\left((\tau_1 - \tau 3\sigma + \frac{1}{2}(c_{11}+c_{33}-2c13)\sigma^2\right) \tag{2.27}$$

$$D_9 = \begin{bmatrix} \frac{1}{(1-\sigma^2)^{\frac{1}{3}}} & 0 & 0 \\ 0 & \frac{1+\sigma}{(1-\sigma^2)^{\frac{1}{3}}} & 0 \\ 0 & 0 & \frac{1-\sigma}{(1-\sigma^2)^{\frac{1}{3}}} \end{bmatrix} \tag{2.28}$$

$$E(V,\sigma) = E(V_0,0) + V_0\left((\tau_2 - \tau 3\sigma + \frac{1}{2}(c_{22}+c_{33}-2c23)\sigma^2\right) \tag{2.29}$$

### 2.2.5 Elastic Constants: Bulk Modulus, Young's Modulus, Shear Modulus and Poisson Ratio

The bulk modulus, as noted earlier, is a measure of the effect of strain on stress. While it may be calculated by taking the second derivative of energu with respect to volume, or by fitting an equation of state, it may also be calculated using the elastic constants of a material or the compliance constants.

The Young's modulus is a measure of the stiffness of a material, and the shear modulus (also know as the modulus of rigidity)

$$B_V = \frac{1}{9}\left(C_{11}+C_{22}+C_{33}+2(C_{12}+C_{13}+C_{23})\right) \tag{2.30}$$

$$B_R = \left(S_{11}+S_{22}+S_{33}+2(S_{12}+S_{13}+S_{23})\right)^{-1} \tag{2.31}$$

## 2.3 Force Matching

To derive a potential, one may approach the problem from first principles in an attempt to replicate reality. It has been more useful, however, to lose any physical elegance [**twobandackland**] to give potentials that work for specific elements under certain conditions. Force data, gathered experimentally or by first-principles calculations, has been used to develop potentials since the 1990s. The force matching method was developed in 1994 by Ercolessi and Adams [**forcematchingmethod**] to link the more accurate, more processor and memory intensive, world of first-principles calculations to Molecular Dynamics.

The force-matching method uses the difference between the actual force (either measured experimentally or calculated by first-principles calculations)

Given a set of M different atomic configurations, and a potential with a set of L parameters ($\vec{p}$), the function $Z_F$ is a measure of the difference between the the forces calculated using the potential for all configurations and the actual (or DFT generated) forces.

$$Z_F(\vec{\alpha}) = \sum_{k=1}^{M} \sum_{i=1}^{k} \sum_{j=1}^{3} |F_{i,j}^{k}(\vec{\alpha}) - \vec{F_{i,j}^{0}}|^2 \tag{2.32}$$

This may be extended to include the calculation of other properties, including the cohesive energy of atoms, the lattice parameter, bulk modulus, elastic constants and so on. Each may be weighted depending on how important the property is to the simulation the potential is required for.

$$Z(\vec{p}) = w_F Z_F(\vec{p}) + w_{b0} Z_{b0}(\vec{p}) + w_{e0} Z_{e0}(\vec{p}) + w_{a0} Z_{a0}(\vec{p}) + w_{ec} Z_{ec}(\vec{p}) + w_{ecoh} Z_{ecoh}(\vec{p}) \tag{2.33}$$

## 2.4 D-Band and S-Band

### 2.4.1 Finnis-Sinclair Potential with S and D Band Contributions

The transition element, Caesium, has a the electron configuration [Xe] $6s^1$. The single valence electron in the 6s shell can be promoted to the higher energy and more compact 6d shell under compression, reducing the atom's size. The bond energy may be calculated from the energies of each band plus the energy required to promote from one band to the other[**twobandacklandreed**].

$$\begin{aligned} U_{bond} &= \sum_i \frac{w_{i,1}}{nN_1} n_{i1}(n_{i1} - N_1) + \sum_i \frac{w_{i2}}{2N_2} n_{i1}(n_{i2} - N_2) + E_{prom} \\ N_1, N_2 &\text{capacities of each band} \\ n_{i1}, n_{i2} &\text{electrons in each band of} i^{th} \text{atom} \\ E_{prom} &= \text{Energy required to promote electron from band 1 to 2} \end{aligned} \tag{2.34}$$

A Finnis-Sinclair potential was derived for Caesium by Ackland and Reed.

The pair functions are defined as:

$$\begin{aligned} V_s(r_{ij}) &= \sum_j \frac{A_s}{{r_{ij}}^{12}} \\ V_d(r_{ij}) &= \sum_j \frac{A_d}{{r_{ij}}^{12}} \end{aligned} \tag{2.35}$$

For the s-band electron density:

$$\phi_s(r_{ij}) = \begin{cases} C_s \left(d_s - r_{ij}\right)^3 & r_{ij} <= d_s \\ 0 & r_{ij} > d_s \end{cases} \tag{2.36}$$

For the d-band electron density:

$$\phi_d(r_{ij}) = \begin{cases} C_d \left(d_d - r_{ij}\right)^3 & r_{ij} <= d_d \\ 0 & r_{ij} > d_d \end{cases} \tag{2.37}$$

The embedding function is the standard Finnis-Sinclair embedding function:

$$F(\rho) = \sqrt{\rho} \tag{2.38}$$

As pressure is applied to Caesium, it changes several times.

I Cs BCC at ambient pressure (115.9 angstrom cubed per atom)

↓

II Cs FCC with slight pressure (67.5 angstrom cubed per atom)

↓

III Cs FCC isostructural transformation (48.7 angstrom cubed per atom)

The parameters derived by Ackland and Reed were "in good agreement with experiment" with the 4.3GPa transition pressure between Cs II and Cs III.

### 2.4.2 Density and Embedding Groups

This program may be use potentials with just one density and embedding function per element. It may also be configured to use two or more groups of densities and embedding energies per element; for example s-band and d-band electrons.

$$U_{EAM} = V_{pair} + \sum_{b}^{B} F_b(\rho_b) \frac{1}{2} \sum_{i=1}^{N} \sum_{j \neq i}^{N} V_{ij}(r_{ij}) + \sum_{b=1}^{B} \sum_{i=1}^{N} F_b[\rho_{b,i}] \text{where } \rho_{b,i} = \sum_{j=i, j \neq i}^{N} \rho_{b,ij}(r_{ij}) \text{and B total number of bands, 1, 2} \tag{2.39}$$

# Chapter 3

# Installation

## 3.1 Requirements

The program was written to work with the following:

- Linux Debian
- Fortran with OpenMP
- Python 3

The program takes advantage of multicore processors. The amount of RAM needed by a calculation is specified in the configuration file. Several gigabyte should be enough.

## 3.2 How To Install

```
sudo apt install gfortran libomp-dev
sudo apt install python3 python3-matplotlib python3-numpy python3-tk

cd ~/
mkdir eampa && cd ~/eampa
git clone https://github.com/BenPalmer1983/eampa_v3
cd eampa_v3
cd f2py_src
./build.sh
```

# Chapter 4

# How To Use

# Appendices

# Appendix A

# Functions

## A.1 Introduction

This section of the appendix covers the types of potential function and common choices of function used and details on how to use these in the potential fitting code.

## A.2 Pair Functions

### A.2.1 Lennard-Jones

$$V(r) = e\left(\left(\frac{r_m}{r}\right)^1 2 - 2\left(\frac{r_m}{r}\right)^6\right) \tag{A.1}$$

Listing A.1: Lennard-Jones

```
#A
#TYPE lennard_jones
#P 2.3 3.5
#L 0.0
#U 7.0
```

### A.2.2 Morse

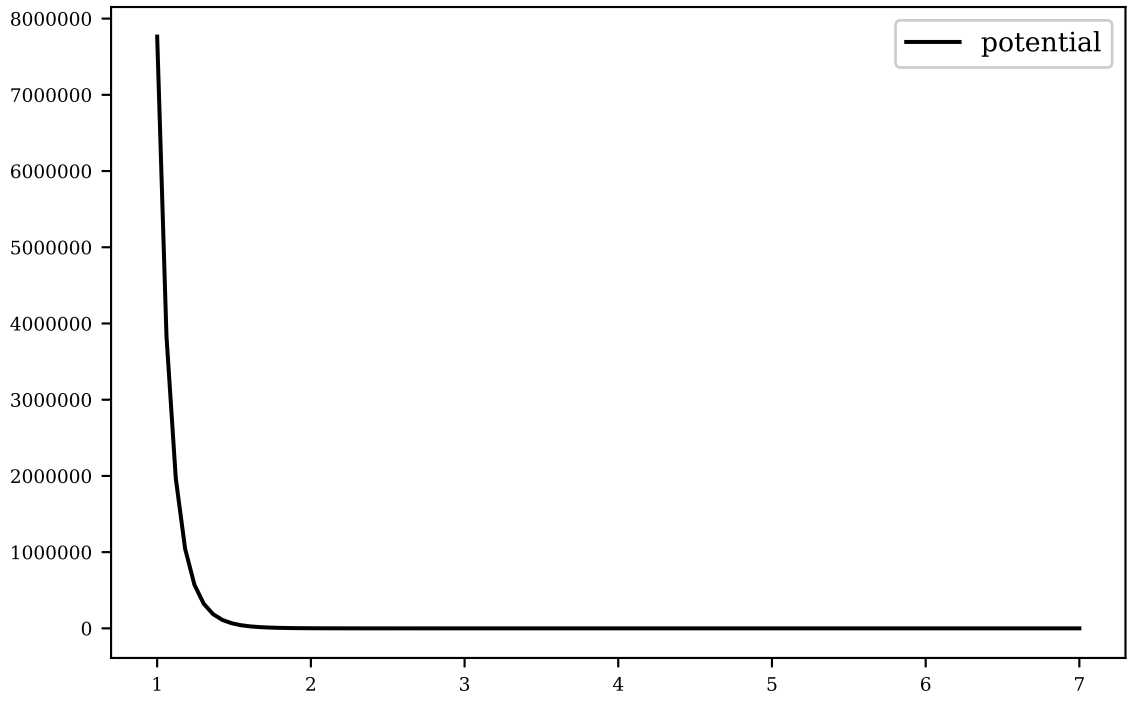
$$V(r) = exp(-2a(r - re)) - 2exp(-a * (r - re)) \tag{A.2}$$

Listing A.2: Morse

```
#A
#TYPE morse
#P 4.669 1.256 2.8
#L 0.0
#U 7.0
```

### A.2.3 Buckingham

$$V(r) = A * exp(-B * r) - \frac{C}{r * *6} \tag{A.3}$$




[scale=0

Figure A.1: Lennard-Jones

Listing A.3: Buckingham

```
#A
#TYPE buckingham
#P 6.0 0.5 12.0
#L 0.0
#U 7.0
```

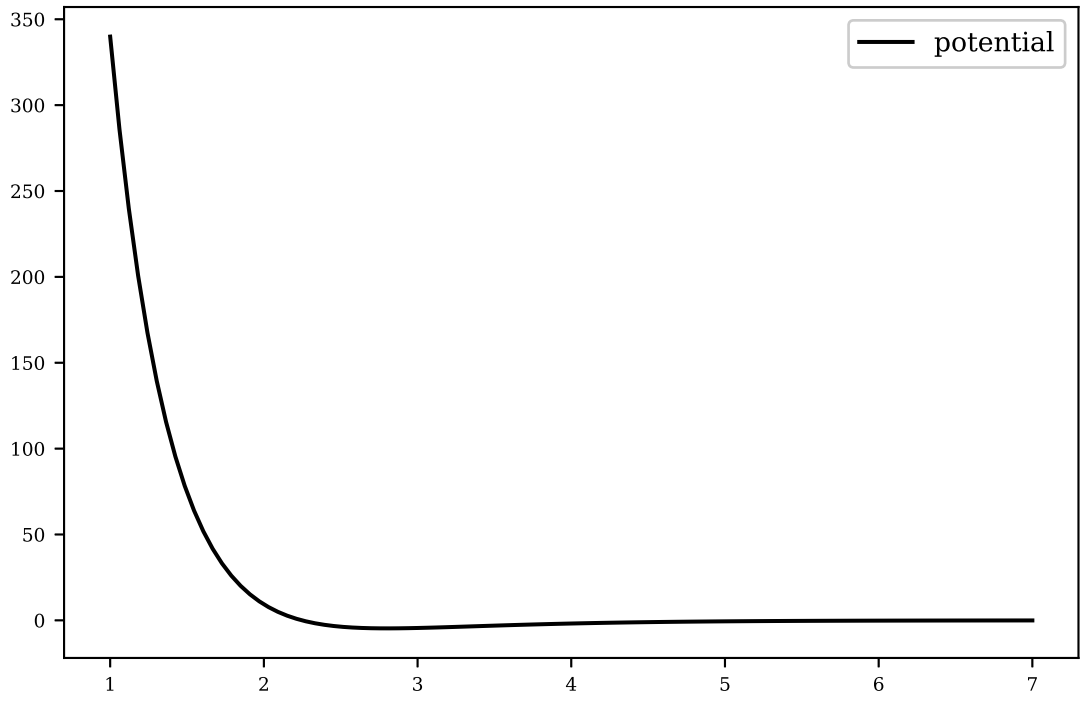
### A.2.4 ZBL

$$\phi(x) = 0.181e^{-3.2x} + 0.5099e^{-0.9423x} + 0.2802e^{-0.4029x} + 0.02817e^{-0.2016x}$$
$$\text{where } a_{ij} = \frac{0.8854a_0}{Z_i^{0.23} + Z_j^{0.23}} \tag{A.4}$$
$$\text{and } a_0 = 0.529\text{angstrom}$$

### A.2.5 Quartic Polynomial Plus Exponential Repulsion

$$H(x) = \begin{cases} (r - r_c)^2(c_1 + c_1 r + c_2 r^2) + Dexp(-\frac{r}{q}) & r <= r_c \\ 0 & r > r_c \end{cases} \tag{A.5}$$

Listing A.4: Quartic Polynomial with Repulsive Term




[scale=0

Figure A.2: Morse

```
#A
#TYPE quartic_poly_rep
#P -4.5377 4.0659 -0.8548 9.5272e7 0.1193
#L 0.0
#U 7.0
```

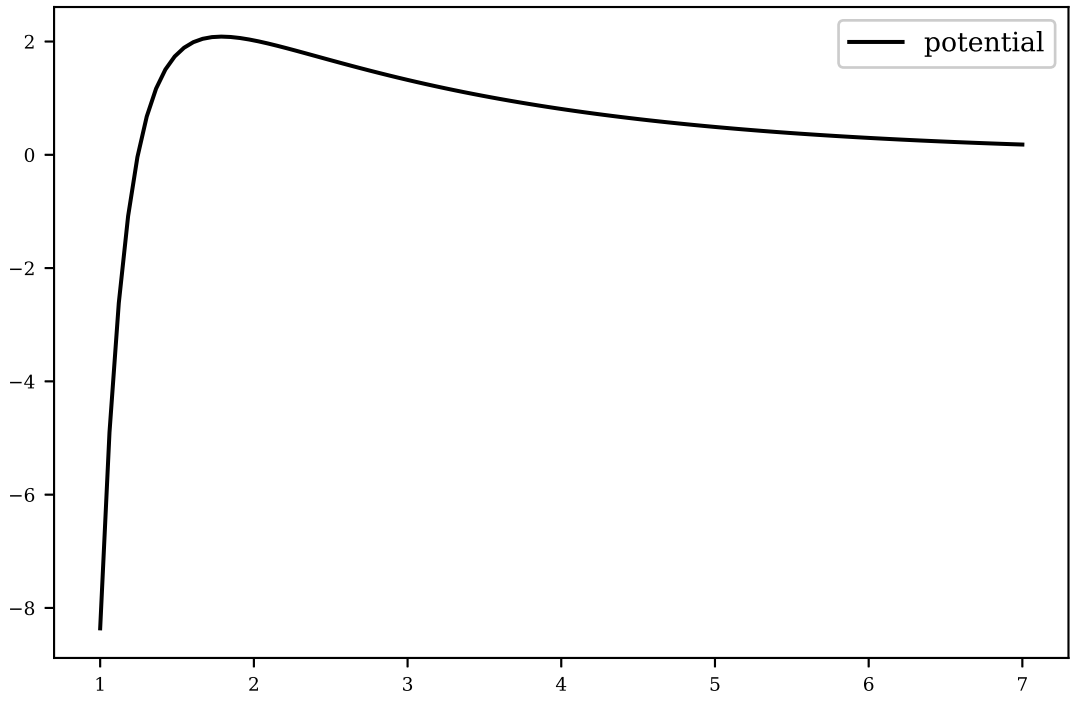
## A.3 Density Functions

### A.3.1 Quadratic Density

$$\rho(r) = (r - r_c)^2 \tag{A.6}$$

Listing A.5: Quadratic Density

```
#A
#TYPE quadratic_density
#P 3.816
#L 0.0
#U 7.0
```




[scale=0

Figure A.3: Buckingham

### A.3.2 Slater 4S (Squared)

$$\rho(r) = (N_s r^3 exp(-\eta r))^2 \tag{A.7}$$

Listing A.6: Slater 4S

```
#A
#TYPE slater_4s
#P 5.0 1.323
#L 0.0
#U 7.0
```

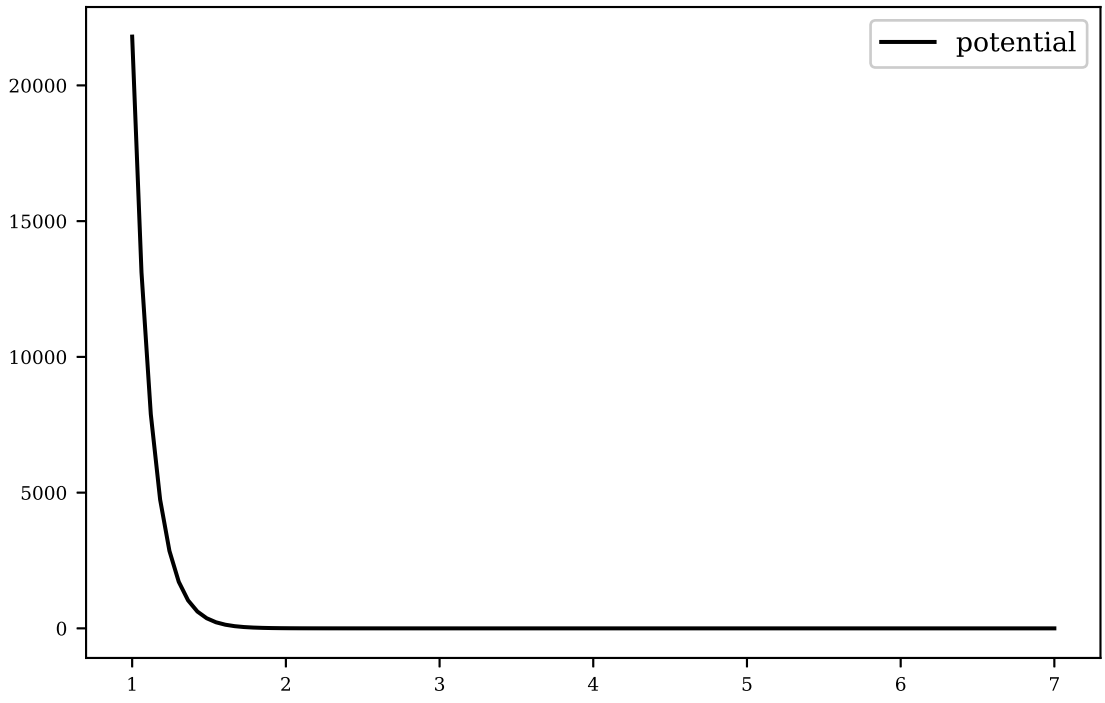
## A.4 Embedding Functions

### A.4.1 Finnis-Sinclair Embedding

$$F[\rho] = -A\sqrt{(\rho)} \tag{A.8}$$

Listing A.7: Finnis-Sinclair Embedding

```
#A
#TYPE fs_embedding
#P 10.0
```




[scale=0

Figure A.4: Quartic Polynomial Plus Exponential Repulsion

```
4 #L 0.0
5 #U 7.0
```

### A.4.2 Mendelev Embedding

$$F[\rho] = -\sqrt{(}\rho) + A\rho^2 \tag{A.9}$$

Listing A.8: Mendelev Embedding

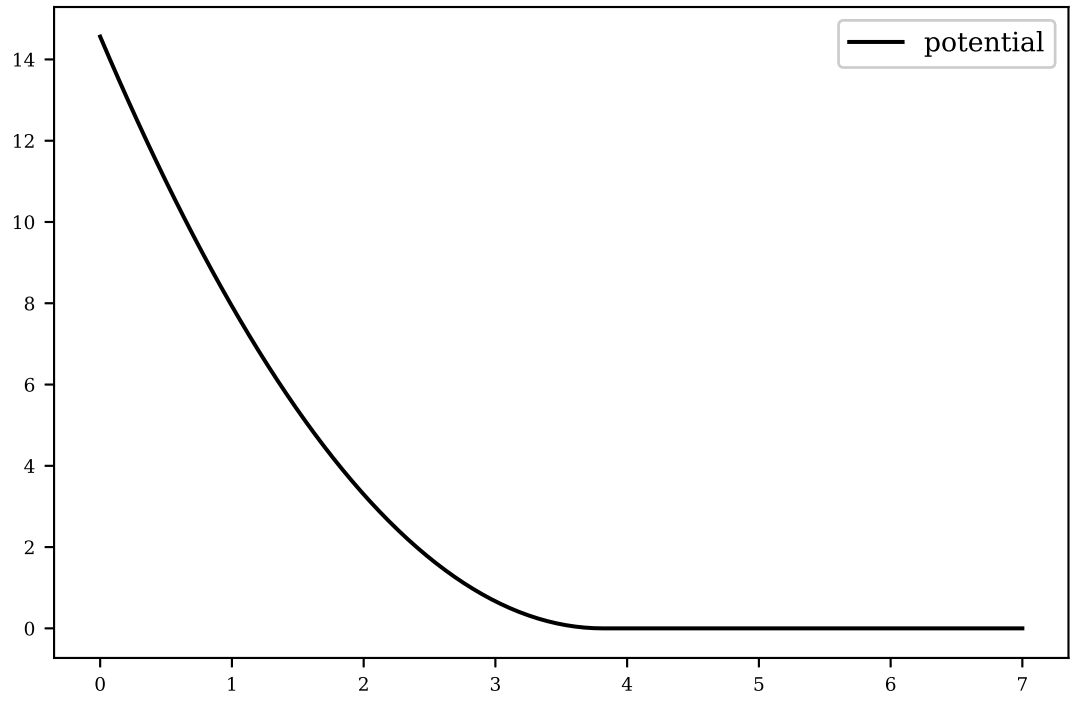
```
1 #A
2 #TYPE mendelev_embedding
3 #P 10.0
4 #L 0.0
5 #U 7.0
```

### A.4.3 Triple Embedding

$$F[\rho] = A\sqrt{(}\rho) + B\rho + C\rho^2 \tag{A.10}$$

Listing A.9: Triple Embedding

```
1 #A
2 #TYPE triple_embedding
```




[scale=0

Figure A.5: Quadratic Density

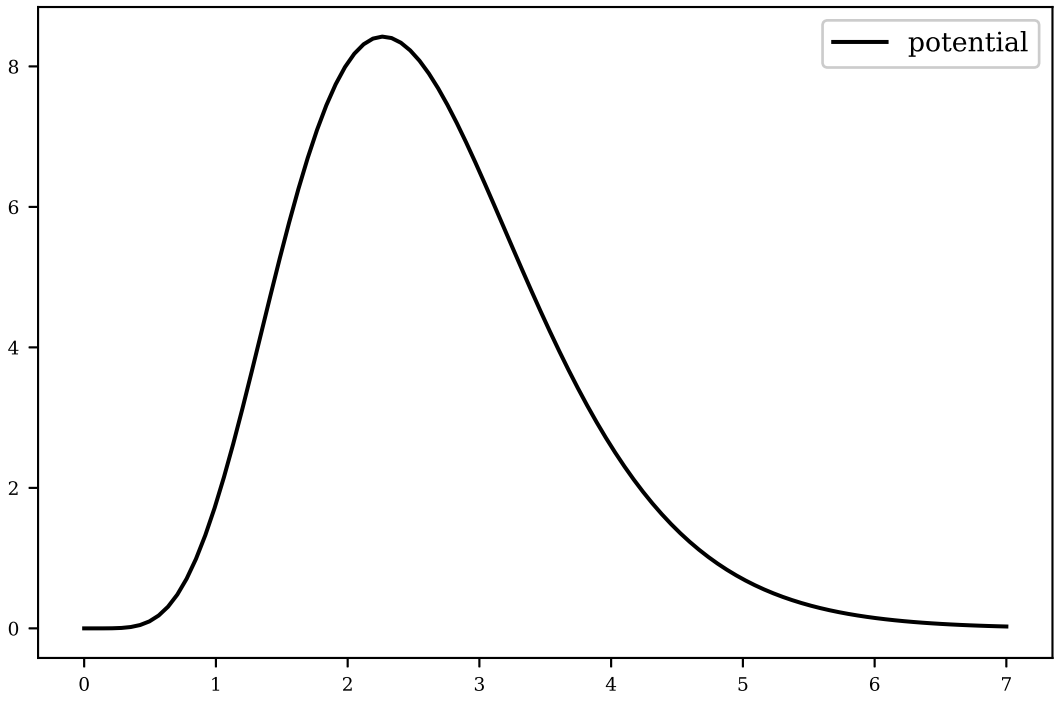
```
3  #P 1.0 1.0 1.0
4  #L 0.0
5  #U 7.0
```

### A.4.4 Ackland Embedding

$$F[\rho] = A\sqrt{(}\rho) + B\rho^2 + C\rho^4 \tag{A.11}$$

Listing A.10: Ackland Embedding

```
1  #A
2  #TYPE ackland_embedding
3  #P 1.0 1.0 1.0
4  #L 0.0
5  #U 7.0
```




[scale=0

Figure A.6: Quadratic Density

## A.5 Generic Functions

### A.5.1 Cubic Splines

$$V(r) = \sum_{i}^{N} a_i (r - r_i)^3 H(r_i - r)$$

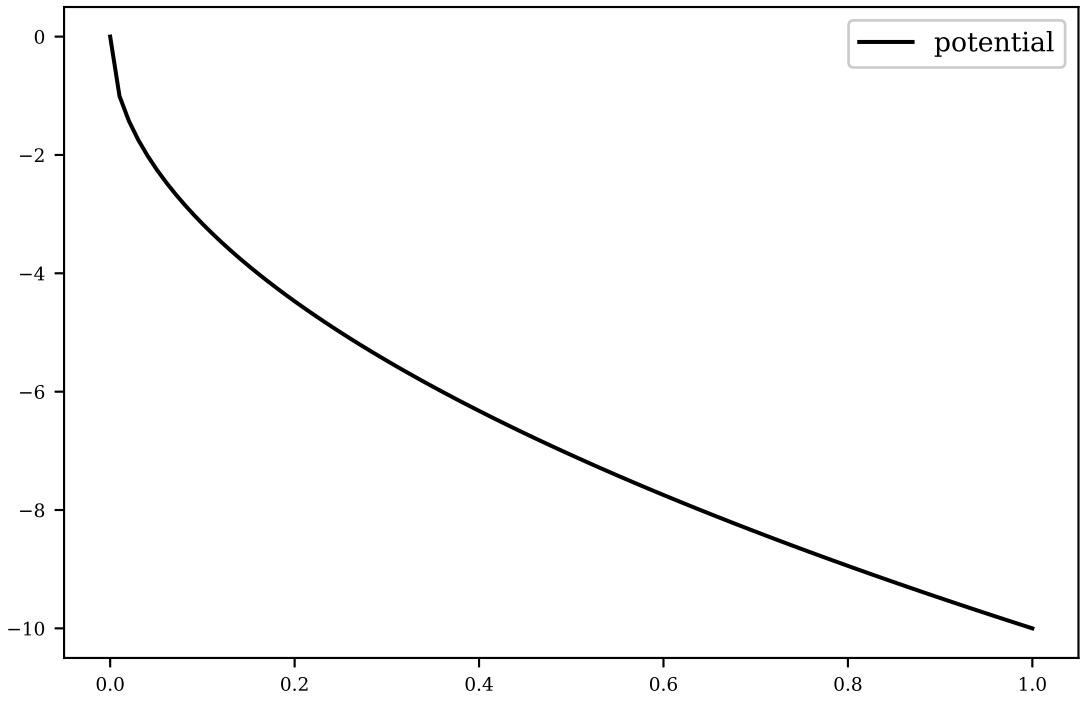
where (A.12)

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x >= 0 \end{cases}$$

This function requires two sets of parameters. P is a list of N coefficients and PF is a list of N cutoffs. The cubic polynomials are summed and individually scaled by the coefficients, and by virtue of its form and the heaviside step function, they cut off at the desired radius.

Listing A.11: Cubic Splines

```
#A
#TYPE cubic_spline
#P -165.0 -78.5 -78.15 1.868
#PF 0.976 1.15 1.216 1.650
#L 0.0
#U 7.0
```




[scale=0

Figure A.7: Finnis-Sinclair Embedding

### A.5.2 Quintic Splines

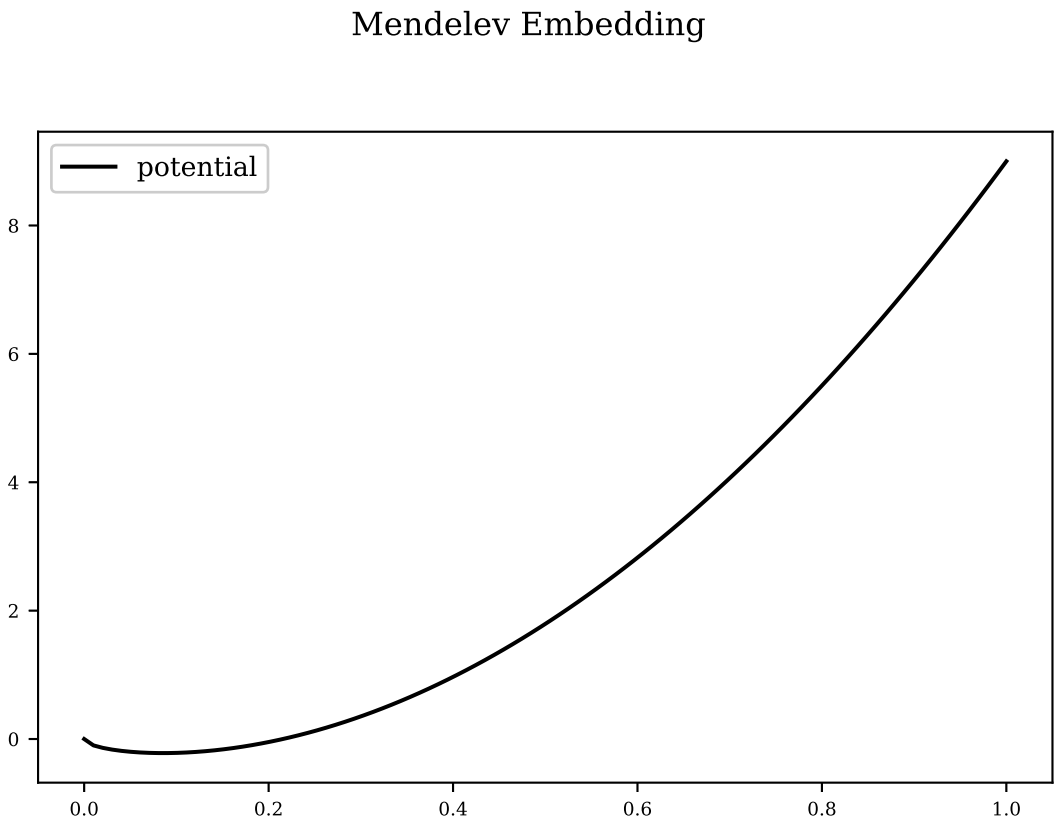
$$V(r) = \sum_{i}^{N} a_i (r - r_i)^5 H(r_i - r)$$

where

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x >= 0 \end{cases} \tag{A.13}$$

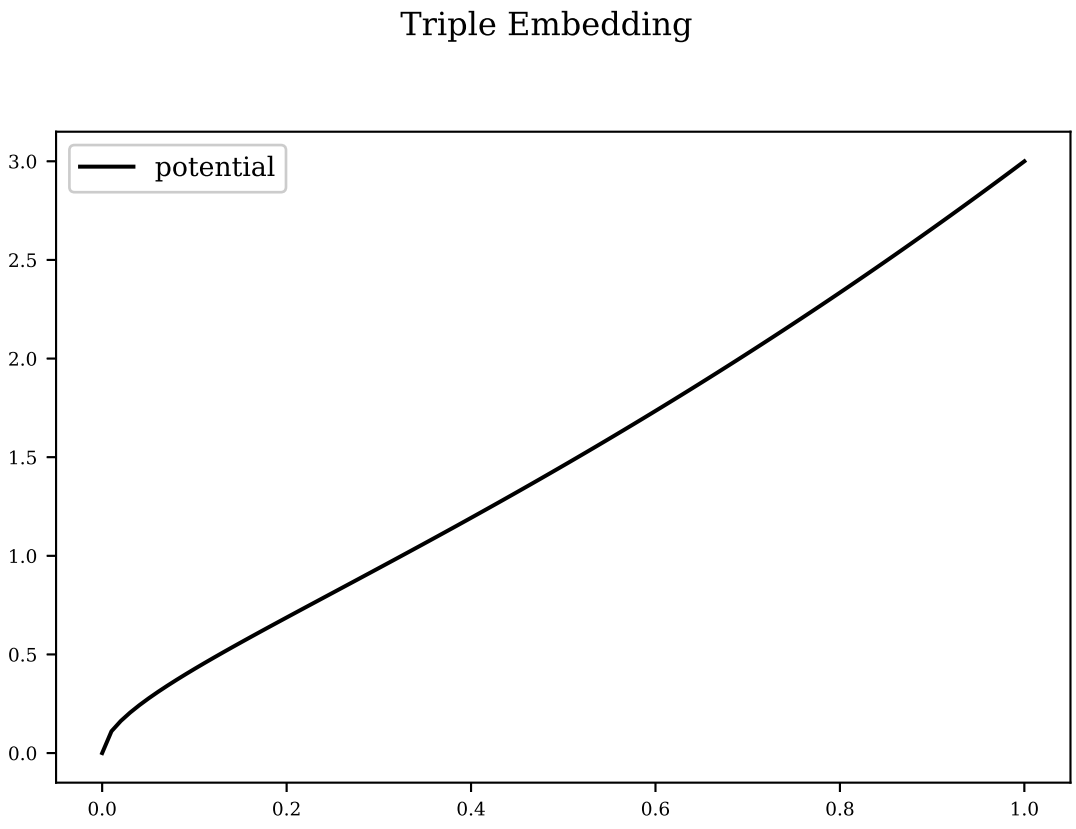
Listing A.12: Quintic Splines

```
#A
#TYPE quintic_spline
#P -165.0 -78.5 -78.15 1.868
#PF 0.976 1.15 1.216 1.650
#L 0.0
#U 7.0
```



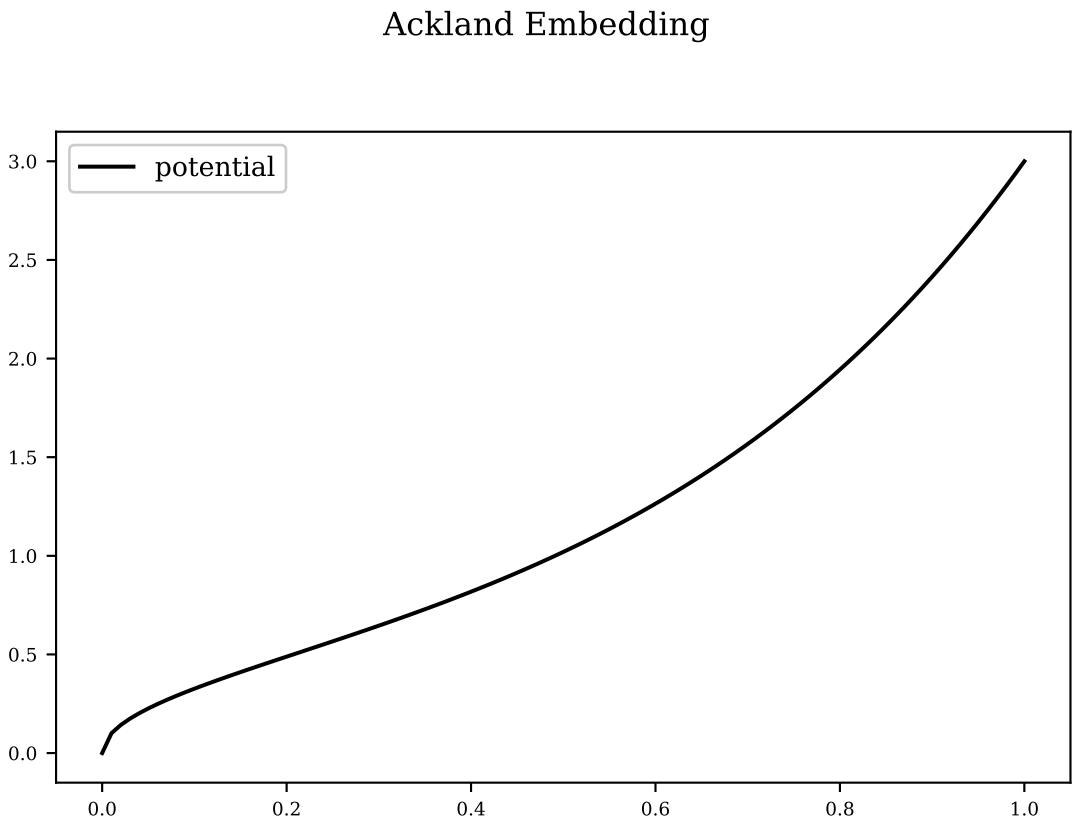

[scale=0

Figure A.8: Mendelev Embedding



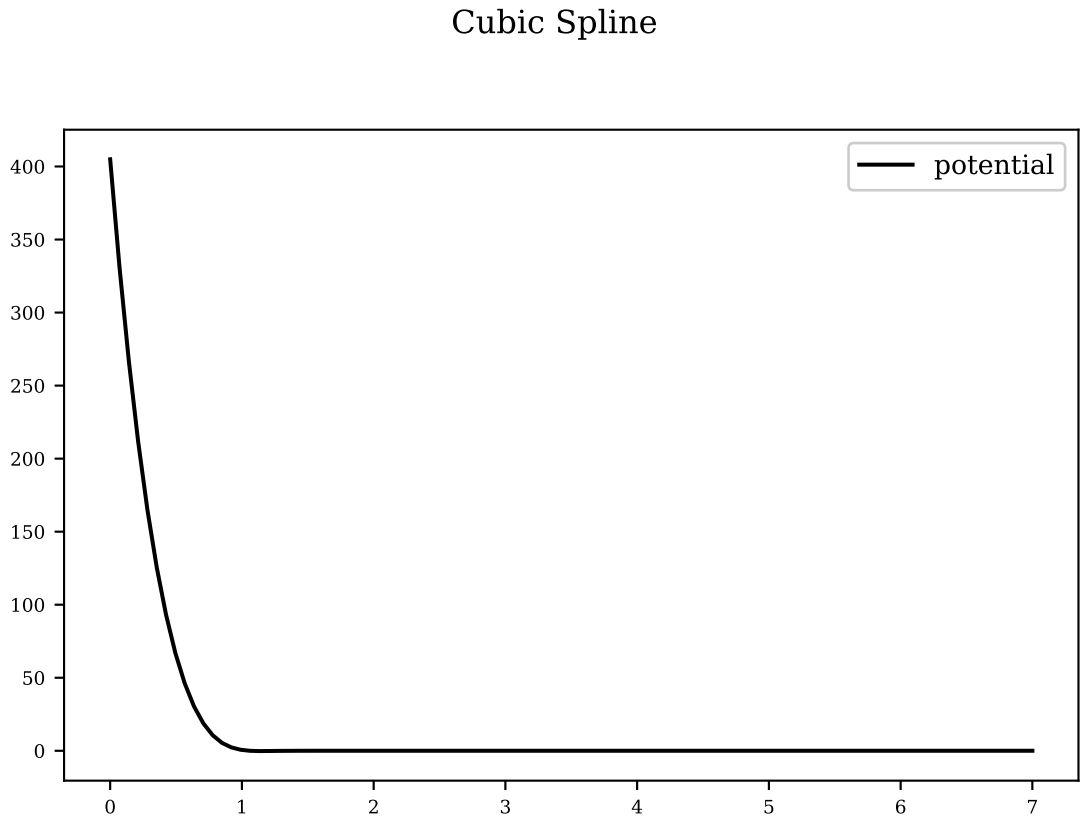

[scale=0

Figure A.9: Triple Embedding



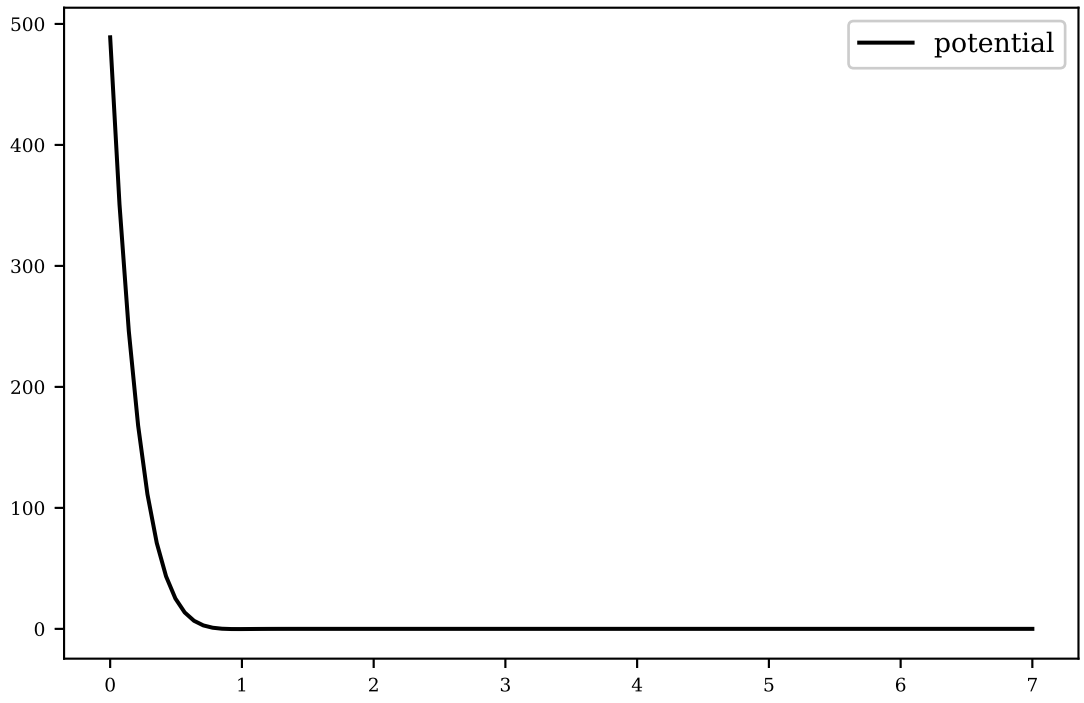

[scale=0

Figure A.10: Ackland Embedding




[scale=0

Figure A.11: Ackland Embedding




[scale=0

Figure A.12: Ackland Embedding

# Appendix B

# Simulated Annealing vs Genetic Algorithm

## B.1 Introduction

Simulated annealing and genetic algorithms are used to find the global minimum of a given problem.

## B.2 Algorithm Comparison

Brief comparisons fits were run on the same fitting problem for both algorithms. For each test, the optimisation time was limited, and the rss for each fit attempt at the end of that time period was recorded. The fitting problem was a double exponential $f(x) = aexp(bx) + cexp(dx)$ with four parameters, and the initial parameters were the same for both algorithms.

### B.2.1 Genetic Algorithm, Pop 32 vs Simulated Annealing

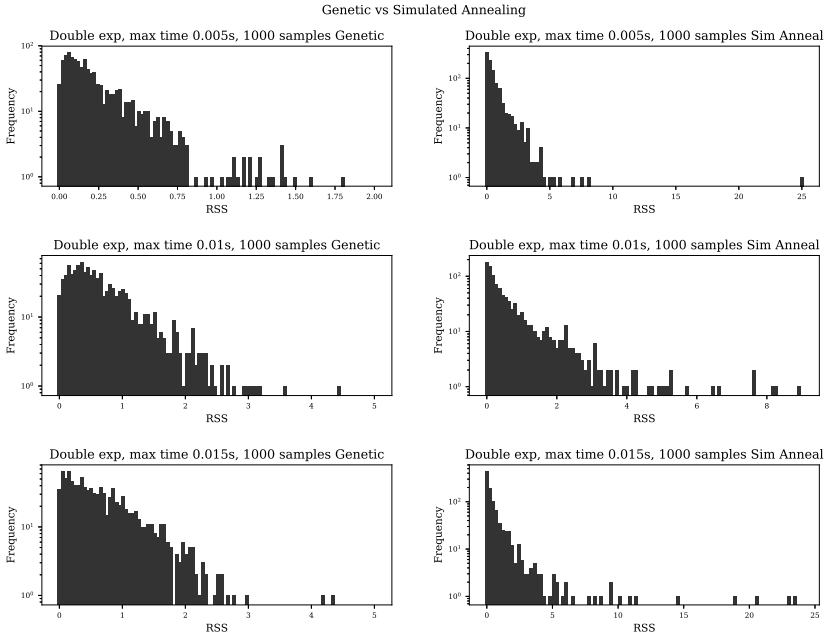



*Figure B.1: Genetic Algorithm Population Size 32 vs Simulated Annealing*

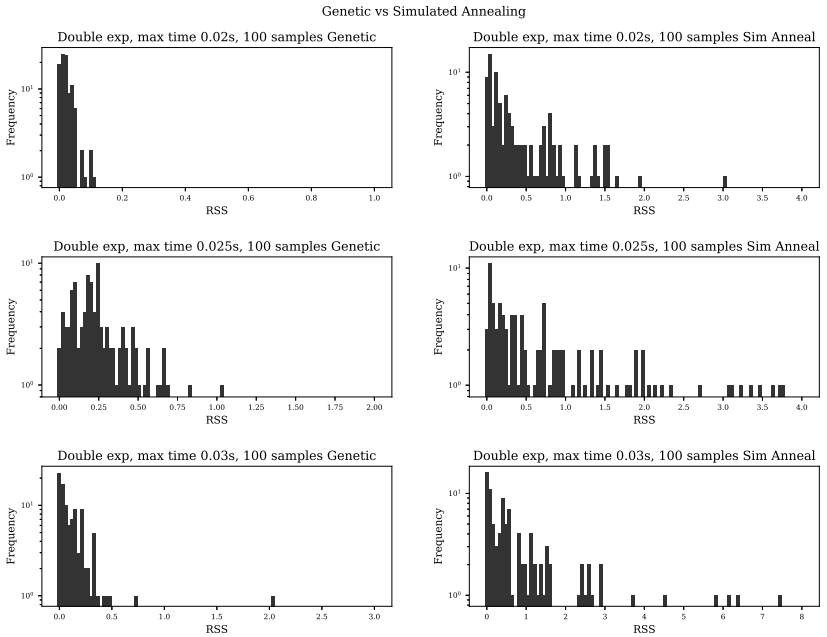



*Figure B.2: Genetic Algorithm Population Size 32 vs Simulated Annealing*

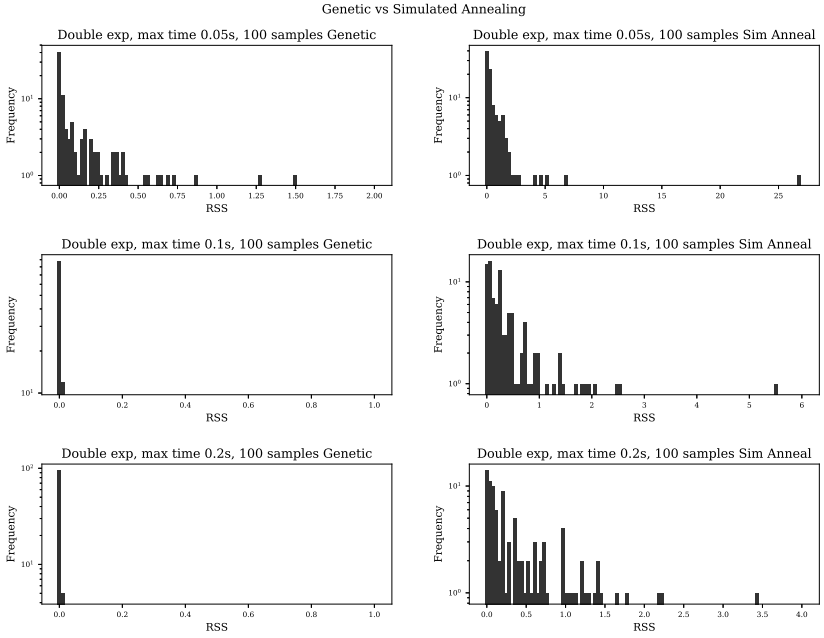



*Figure B.3: Genetic Algorithm Population Size 32 vs Simulated Annealing*

### B.2.2 Genetic Algorithm, Pop 64 vs Simulated Annealing

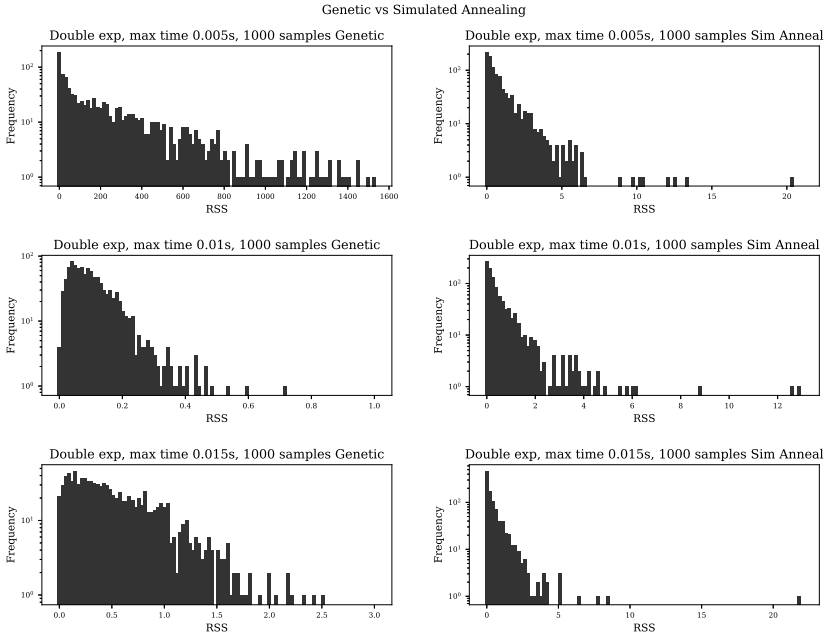



*Figure B.4: Genetic Algorithm Population Size 64 vs Simulated Annealing*

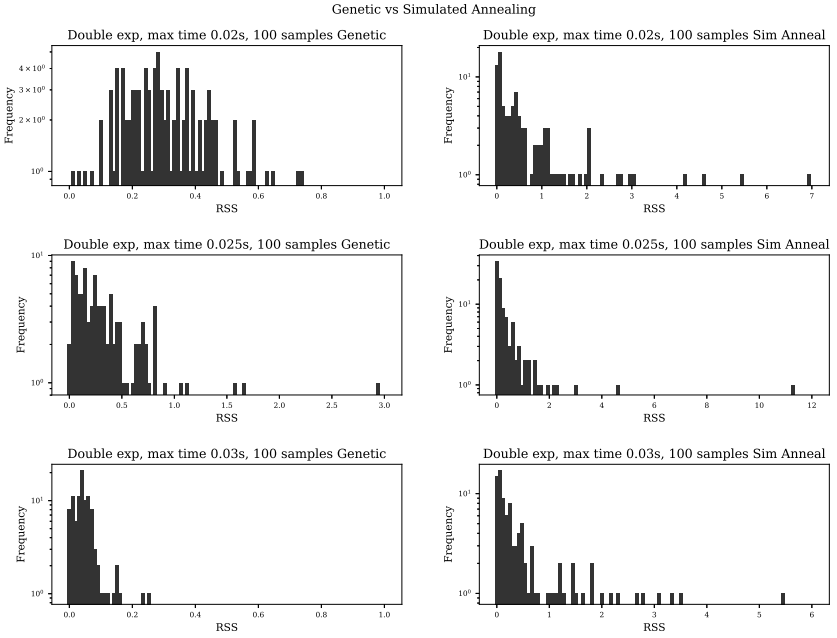



*Figure B.5: Genetic Algorithm Population Size 64 vs Simulated Annealing*

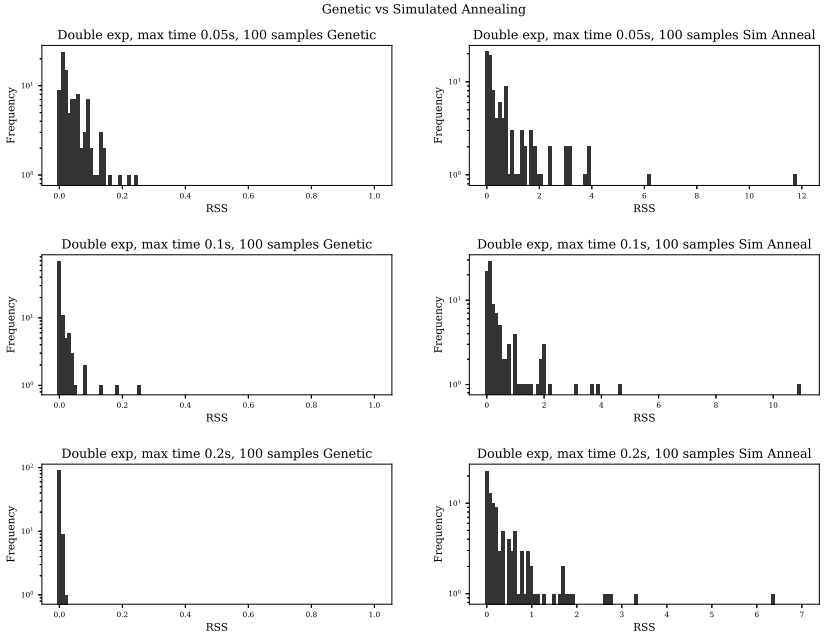



*Figure B.6: Genetic Algorithm Population Size 64 vs Simulated Annealing*

### B.2.3 Genetic Algorithm, Pop 512 vs Simulated Annealing

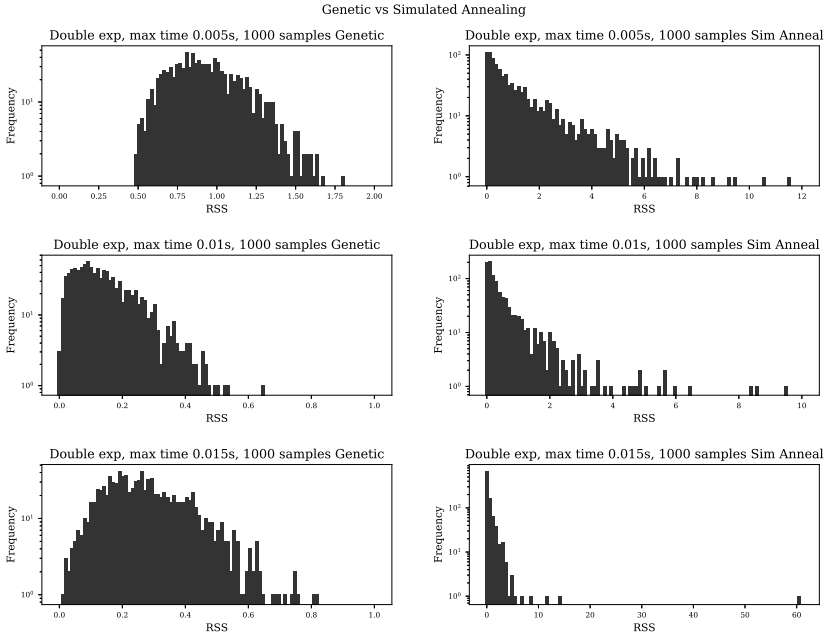



*Figure B.7: Genetic Algorithm Population Size 512 vs Simulated Annealing*

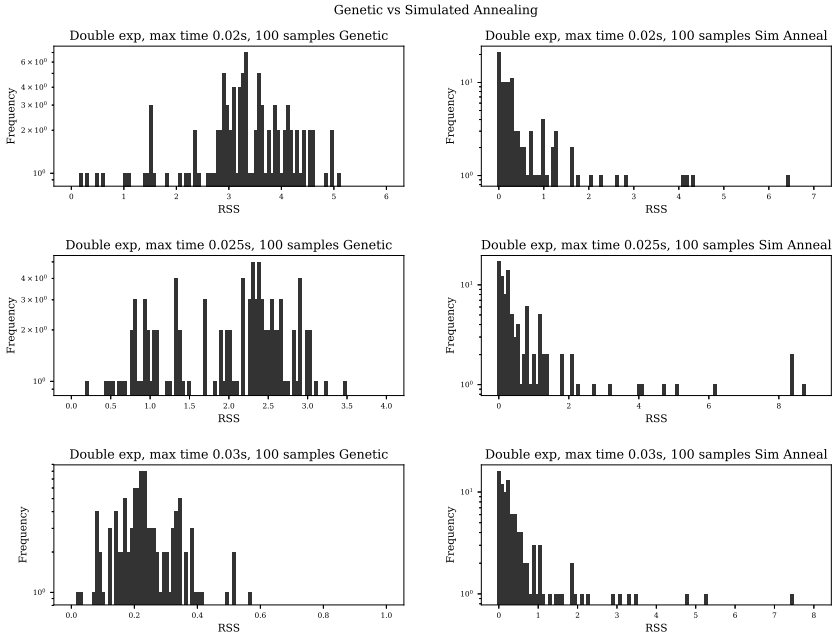



*Figure B.8: Genetic Algorithm Population Size 512 vs Simulated Annealing*

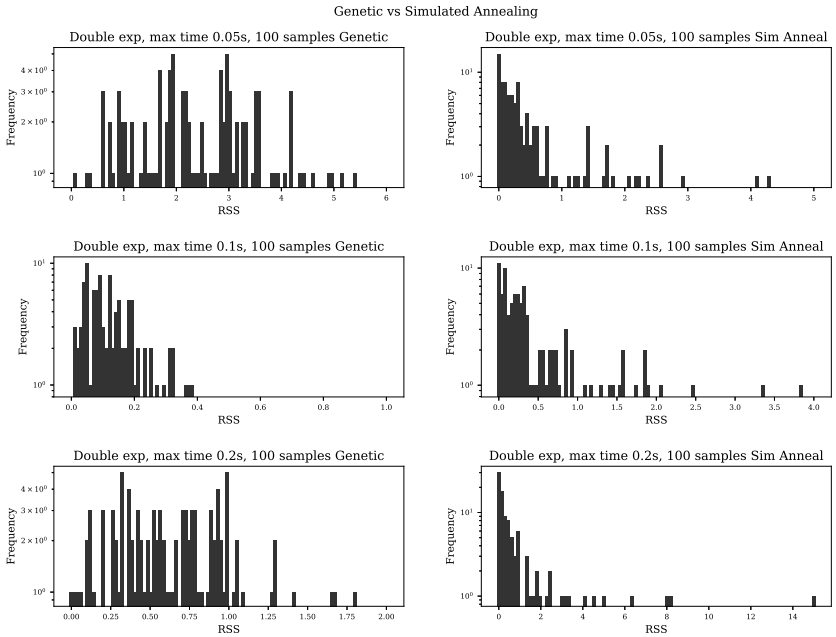



*Figure B.9: Genetic Algorithm Population Size 512 vs Simulated Annealing*