Determination of Minimum Energy Atomic Configurations, via Gradient Descent and Quasi-Newton Methods

Nima Leclerc, Luther Lu
Cornell University
CS 4220
Numerical Analysis: Linear and Nonlinear Problems

April 28, 2018

Introduction

For this project, one considers solving an empirical atomic con
gurations problem. More specifically, one seeks minimal energy con
gurations of non-bonded atoms in three dimensional space using a simple model for atomic interactions. Practically, a molecular con
guration problem contains many more components then we will omit for simplicity. This model makes use of the well known Leonard-Jones potential.
One can consider the Leonard Jones Potential (1) which describes the interatomic interaction of two neighboring atoms. It assumes a form, such that the interaction energy decreases with interatomic distance. One can define this distance as $r_{ij} = ||x_i - x_j||_2$, given atomic positions in three dimensional space.

$$V_{ij} = \frac{1}{r_{ij}^{12}} - \frac{2}{r_{ij}^{6}} \tag{1}$$

Given this interatomic interaction, one can consider a collection of these interactions describing the total potential of an atomic system. Hence, one can solve the minimization problem described in (2) for a system of N atoms,

$$\min_{x_2,\ldots,x_N} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} V_{ij} \tag{2}$$

Solving (2) would yield the minimum energy configuration of atomic positions, $\{x_i\}$. It is also worth considering that these positions are with respect to an origin, $(0, 0, 0)$. In this minimization problem, one can state that $x_1 = (0, 0, 0)$. The solution to (2) would yield a local minimum or ideally a global minimum, which becomes an optimization problem in $3(N - 1)$ variables.

This problem can be achieved with three approaches, via computational methods. These include, the gradient descent method, Newton's method, and Quasi-Newton methods. In this work, the gradient descent and BFGS Quasi-Newton methods are used to solve the energy minimization described in (2).

The Gradient Descent Algorithm

The gradient descent algorithm is a method commonly used in unconstrained optimization to seek local minima that exist in complex functions. It is an iterative method, hence it generates a sequence of iterates $\{x_k\}$. In general, one can consider a stationary iterative algorithm as follows

$$x_{k+1} = G(x_k) \tag{3}$$

for fixed point $G(x_k)$. Generally, one can consider the following approach for a gradient descent-type optimization

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \tag{4}$$

The above relation of subsequent and preceding iterates must satisfy the following optimality condition, $\nabla f = 0$. One can now describe G as a nonstationary iterate, hence the following form is obtain for subsequent iterates,

$$x_{k+1} = G_k(x_k) \tag{5}$$

where there lies a dependence on the preceding iteration. The solution iterate expression in the nonstationary case can be expressed as,

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) \tag{6}$$

This now requires a dependance of α on all previous iterates. Determination of these coefficients depend on the desired outcome of optimization. Hence, the choice of α_k for a given iteration must satisfy a set of constraints, known as the Wolfe Conditions. The primary role of these constraints is to enforce sufficient descent and curvature in seeking a local minimum, via Gradient Descent methods. The determination of these coefficients is known as *line search* and greatly depends on the iteration step. In choosing these step sizes, one has to be sure that the step size is not too large, which could result skipping over a local minimum. Additionally, the risk of taking very small step sizes is having a much slower algorithm. *Backtracking* is used to decrease the step size at every iteration, hence resulting in smaller steps closer to the desired local minimum. The mentioned Wolfe Conditions are listed below,

$$(i) f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \mathbf{p}_k^T \nabla f(\mathbf{x}_k)$$

$$(ii) - \mathbf{p}_k^T \nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \leq -c_2 \mathbf{p}_k^T \nabla f \mathbf{x}_k)$$

$$(iii) |\mathbf{p}_k^T \nabla f \mathbf{x}_k + \alpha_k \mathbf{p}_k)| \leq c_2 |\mathbf{p}_k^T \nabla f \mathbf{x}_k)$$

Above, p_k defines the descent direction. In this case for gradient descent, $p_k = -\nabla f(x_k)$, where f is represented by the collective potential energy of the system due to Leonard Jones interactions. Condition (i) is the so-called Armijo-Rule, which enforces that there exists a sufficient decrease during the optimization with respect to the step size, α_k and hence a descent to a local minimum. Note that there exists a parameter c_1, which must be between 0 and 1. Tuning tis parameter is essential in seeking efficient convergence. Condition (i) must always be enforced. However, conditions (ii) and (iii) are strong Wolfe Conditions,which enforce the curvature constraint in seeking local minima. c_2 is another tuning parameter, which tends to be larger than c_2 used in tuning the convergence of the optimization routine. Backtracking can be described in this context by introducing a descent direction and decreasing the stepsize per iteration. Hence if α satisfies the desired Wolfe conditions, the set of α_k values can be generated by, $\frac{\alpha_k}{\alpha_{max}}$. In the case of Gradient Descent, one assumes that the descent direction is simply, $\mathbf{p}_k = -\nabla f \mathbf{x}_k)$ which is not always true for similar optimization routines. Other algorithms would scale the gradient by some matrix dependent on iteration step. Gradient descent is implemented here in determining the minimum energy configuration of atoms, for a given set of tuned parameters.

Algorithm Implementation and Results

Both Gradient Descent and BFGS methods were implemented in determining the minimum energy atomic configuration. To begin with the implemented gradient descent method, the case of two and three atoms were considered. The algorithm was implemented below in MATLAB, (along with the line search method) :

```
%%%LeonardJones_POT_GRADDESC.m is a function which computes the minimnum energy configuration
%%%of atoms, given the Leonard Jones Potential Function. This method
%%%employs all Wolfe Conditions, including the strong Wolfe Conditions
%% INPUT: {xi}Initial Guess of Atomic Configuration, c1, c2 tuning paramaters.
%% OUTPUT: Minimum Energy Atomic Configuration, x
function [xopt, Vmin] = LeonardJones_POT_GRADDESC(x_init,c1,c2)
%%Contstruct Leonard-Jones Potential for system of n atoms
Vmin=[]; %% Optimal potential construction
n=size(x_init(:,1)); %% Number of Spatial Points
X=sym('x', [n 3]); %% Construction of points
Xvar = X(:);
V=sym('v',[n n]);
R=sym('r', [n n]);
Vtot=0;
for ii=1:n
    for jj=1:n
       R(ii,jj)= norm(X(ii)-X(jj));
       V(ii,jj) = R(ii,jj).^(-12)-2*R(ii,jj).^(-6);
       Vtot = Vtot + V(ii, jj);
    end
end
LJV=Vtot;%% Leonard-Jones Potential Construction
dfk=gradient(LJV,Xvar)'; %%Unperturbed gradient iterate
Pk= -dfk;
```

```
xk=x_init;
w=1;
c=0;
del=0;
alph0=1.5;
alphk = line_search(LJV, xk,c1,c2, 1.5);
dfkn = double(subs(dfk, v, x+alphak*xk)); %%Perturbed gradient iterate
while any(abs(w)> 1e-8) && (mean(abs(del)) > 1e-8) && c < 200
    Vmin(end+1)=double(subs(LJV,Xvar,xk));
    w=alphk*Pk';
    xk=xk+w;
    dfkn=double(subs(dfk, X, xk));
    alphk=LINESEARCH(func,x,c1,c2,alph0);
    del=w+alphk*Pk';
    c=c+1;
end
 Vmin(end+1)=double(subs(V, v, xmin));
 xopt = xk;
end
```

```
 %%%LINESEARCH.m is a function which computes the linesearch coefficient, alpha
%% Employs backtracking with respect to the three Wolfe Conditions
%% INPUT: function func, paramaters c1 & c2
%% OUTPUT:value of alph for given iteration
function [alphk] = LINESEARCH(func,x,c1,c2,alph0)
Pk=-gradient(func(xk))k ;
alphk = .1/mean(abs(Pk)); %% Initialize alpha coeffient
fk = double(subs(func, v, x)); %%Unperturbed function iterate
fkn = double(subs(func, v, x + alphk*Pk)); %%Perturbed function iterate
dfk = double(subs(df, v, x)); %%Unperturbed gradient iterate
dfkn = double(subs(df, v, x+alphk*Pk)); %%Perturbed gradient iterate
%% Define Wolfe Conditions
cond1=fkn<= fk+ alphk*c1*(Pk)*dfk';
cond2=-Pk'*dfkn <= -c2*(Pk')*dfk';
cond3=abs(Pk*dfk') <= c2*abs(Pk'*Pk);
while (~cond1 || ~cond2 || ~cond3) && alphk > 1e-5/mean(abs(Pk))
    alphk=alphk/alph0;
    fkn = double(subs(f, v, x + alphk*Pk));
    cond1=fkn<= fk+ alphk*c1*(Pk)*dfk';
 end
end
```

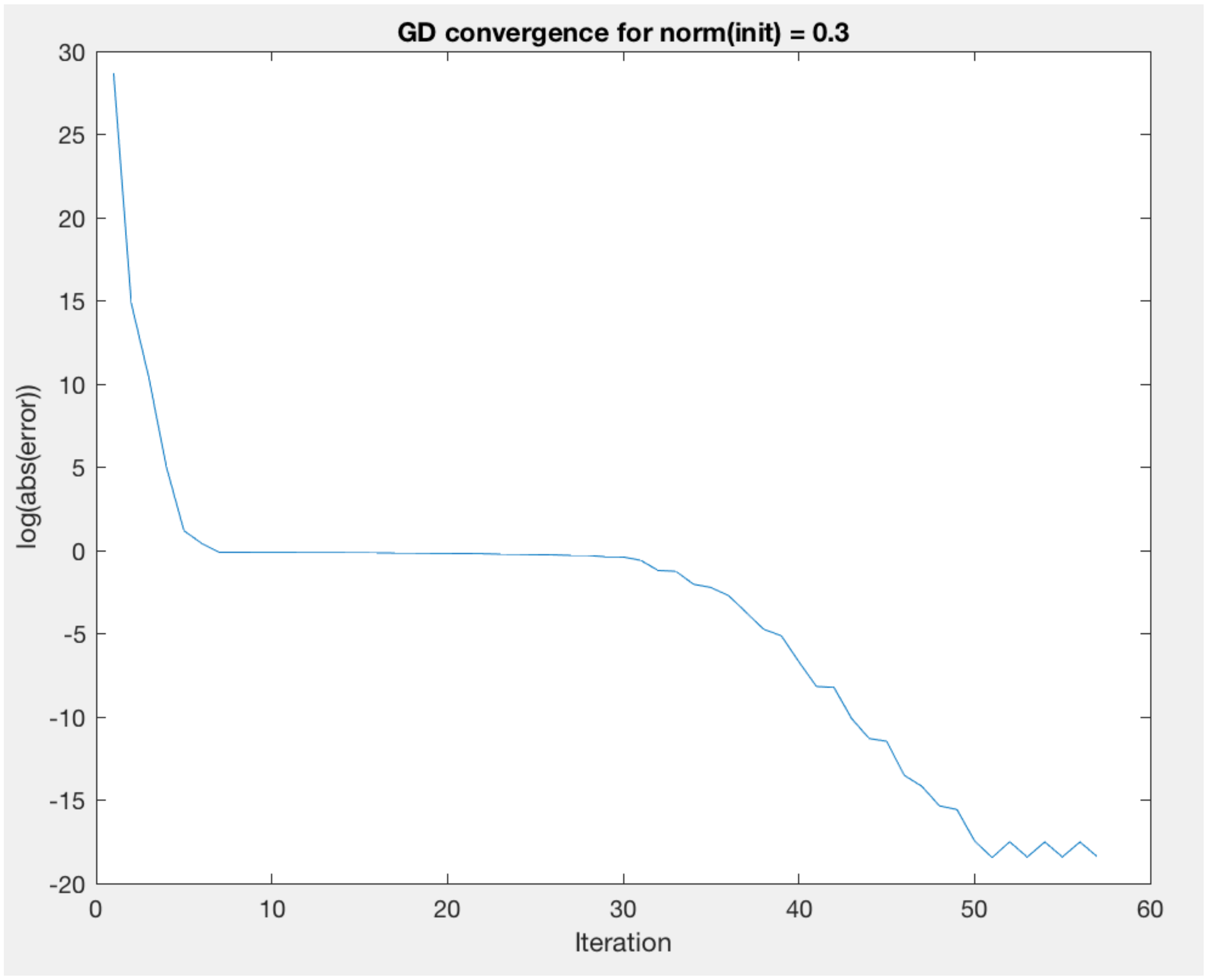
```
%%%BFGS.m is a function which computes the minimnum energy configuration
%%%of atoms, given the Leonard Jones Potential Function. This method
%%%employs all Wolfe Conditions, including the strong Wolfe Conditions
%% INPUT: {xi}Initial Guess of Atomic Configuration, c1, c2 tuning paramaters.
%% OUTPUT: Minimum Energy Atomic Configuration, x
function [xopt, Vmin] = BFGS(x_init,c1,c2,Bk)
%%Contstruct Leonard-Jones Potential for system of n atoms
Vmin=[]; %% Optimal potential construction

n=size(x_init(:,1)); %% Number of Spatial Points
X=sym('x', [n 3]); %% Construction of points
Xvar = X(:);
V=sym('v',[n n]);
R=sym('r', [n n]);
Vtot=0;
```

```
for ii=1:n
    for jj=1:n
       R(ii,jj)= norm(X(ii)-X(jj));
       V(ii,jj) = R(ii,jj).^(-12)-2*R(ii,jj).^(-6);
       Vtot = Vtot + V(ii, jj);
    end
end
LJV=Vtot;%% Leonard-Jones Potential Construction
dfk=gradient(LJV,Xvar)'; %%Unperturbed gradient iterate
Pk= -dfk/Bk;
xk=x_init;
w=1;
c=0;
del=0;
alph0=1.5;
alphk = line_search(LJV, xk,c1,c2, 1.5);
dfkn = double(subs(dfk, v, x+alphak*xk)); %%Perturbed gradient iterate
while any(abs(w)> 1e-8) && (mean(abs(del)) > 1e-8) && c < 200
    Vmin(end+1)=double(subs(LJV,Xvar,xk));
    w=alphk*Pk';
    Bkn = Bk+( x_{k}*x_{k}^{T})/( x_{k}^{T}*w)) - ( Bk*sk*sk'*Bk/( w_{k}^{T}*Bk*w))
    xk=xk+w;
    dfkn=double(subs(dfk, X, xk));
    alphk=LINESEARCH(func,x,c1,c2,alph0);
    del=w+alphk*Pk';
    c=c+1;
end
 Vmin(end+1)=double(subs(V, v, xmin));
 xopt = xk;
end
```

In this implementation, the same line search method (LINESEARCH.m) was used for both gradient descent and BFGS algorithms. The potential function due to the Leonard-Jones interaction is expanded as a function of input positions symbollically. In the implementation, *xinit* is the initial guess of atomic configuration. Here, 2 initial guesses were made for each the 2 atom and three atom configurations. In the case of 2 atoms, this initial guess was $X_1 = [(0.1, 0.1, 0.1), (0.3, 0.2, 0.1)]$ and $X_2 = [(0, 0, 0), (.4, .8, 0.6)]$. The parameters c_1 and c_2 were tuned in accordance to convergence to an optimal value. Initially, c_1 was set to 1e-9 and c_2 was set to 0.9. These parameters were set for both initial guesses of 2 atoms. Convergence was achieved in each case when either $|\nabla f(x_k)| \leq 10^{-15}$ or $\alpha_{k+1}p_{k+1} - \alpha_k p_k < 10^{-6}$. These conditions were implemented for, if the the gradient of the function is near 0 , one would expect convergence and the second condition enforces that the solution does not oscillate. In addition, the step size α was initiated to $\alpha_k = \frac{0.1}{mean(|P_k|)}$ for, if the gradient approaches small values, large step sizes would be allowed. In each case, the maximum value of the step size) was set to 1.5 (as shown in code, each step size was normalized by this factor). Note, that this method employed all 3 of the Wolfe conditions. From this, it was found that both initial guesses of configurations began to converge an energy of -1 and then suddenly diverged in the positive direction. The values of near -1, converging from the positive direction were the following $-1.000123, -1.000367, -1.00001, -1.00004, -1.000112$. However, at iteration 70 the value of the minimum value began to climb up. It then became clear that convergence has not been achieved. This behavior was observed for both initial atomic configurations. The tuning knob that seemed most relevant to change were those in the Wolfe conditions. This is for the initial assumption for atomic configuration was arbitrary and was assumed to converged given the large number of iterations. Hence, the the strong Wolfe conditions were removed. Consequently, removing the strong Wolfe conditions made the choice of c_2 irrelevant and the backtracking method only employed Armijo's principle. This approach was surprisingly effective. As a result, the energy converged to -1 as expected with a minimum energy atomic configuration of $X_{min} = [(0, 0, 0), (0.4, 0.8, 0.2)]$ .The norm of the distance between the atoms is 1.077, whcih is relativley close to one. Solving this equation numerically on Mathematica yields solution of $r = 1$ at the value $V = 1$ given the

Leonard Jones potential. This agreement is promising in determining the global minimum of the potential. From here, the c_1 parameter was also tuned from 0.9 to 0.8 and the same value of energy as well as atomic configuration was obtained. That result assured that the local minimum for 2 atoms occurred at $X_{min} = [(0,0,0),(0.4,0.8,0.2)]$ with a value of $V = -1$. I argue that this is the global minimum, for the following reasons: the step size as small enough such that it did not skip over any local minima and various intial guesses of atomic configurations were used and yielded the same minimum value. Since there exists no other local minima, the final check would be for minima at the boundaries of the potential function. However, this is an unbounded function and hence there would be none to screen. This was also attempted for 3 atoms, with an initial guess of $X_1 = [(0,0,0),(.4,.8,0.6),(0.5,0.5,0.5)]$. Here, c_1 was set to 0.9 as well. Using the same paramaters, the mininimum energy converged to -3. In fact, this value was achieved with both gradient descent and BFGS. It appeared that the distance between atoms 1 and two for this optimal configruation were the same as the distance between atoms 2 and 3. Here, again the coordinates obtained for atoms 2 and three were $X_{min} = [(0,0,0),(0,-0.15,0.987),(0.205,0.763,0.612)]$. Here again, the distance between the two atoms are near 1. This would make sense physically, for one would expect interactions from either sides to be equal and opposite (opposite direction) in energy, to balance the total energy of the system. One can compare the BFGS and gradient descent algorithms and notice that the norm of the initial guess of the position should not be less than 0.001 if a global minimum energy of -3 needs to be achieved. One expects BFGS to converge faster than gradient descent. The attached plot shows the convergence of gradient descent for an initial guess of 0.3. The slope of the plot suggests that the gradient descent algoirithm does have a relativley slow convergence.


GD convergence for norm(init) = 0.3
log(abs(error))
Iteration
30
25
20
15
10
5
0
-5
-10
-15
-20
0
10
20
30
40
50
60