ECE505 Computer Project I Report

Roy Subhadeep - A20358508

Abstract

Given various unconstrained optimization problems, the convergence and performance is tested using Steepest Descent, Newton's method, BFGS Quasi-Newton, Conjugate Gradient and a relative comparison is drawn out of it.

Introduction

Unconstrained optimization problems consider the problem of minimizing an objective function that depends on real variables with no limitations on their value or conditions.

$$\text{Min } f(x)\ ;\ x \in R$$

Though unconstrained optimization problems are not very common in the real world, they often arise indirectly from reformulation of constrained optimization. An important aspect of optimization is whether the second order derivatives exists and are smooth. The general method is an iterative one where an initial point x_0 is selected and for each k_{th} iteration, x_k is computed and verified against optimality. This process is continued until and optimal solution x^* (stationary point where the function is minimized) is found.

Many algorithms have been proposed but each comes with its own challenges and the choice of algorithm for a particular unconstrained optimization problem remains highly subjective. The algorithms are often tested against speed, convergence steps, computational overhead and storage capacity.

The main idea of this report is to test a set of unconstrained optimization problems against four different algorithms:

- Steepest Descent
- Newton's method
- BFGS Quasi-Newton
- Conjugate Gradient

Development Environment

The algorithms are developed in Python 2.7 with numpy, sympy, matplotlib as helper libraries.

The iPython notebook for the above implementations can be found in the below link.
https://github.com/royxss/OptimizationAlgo

Part 1: Algorithmic Implementation

The algorithm implementation revolves around a general idea of iteration to find x^* but with different parameter settings.

1. Steepest Descent

To minimize f(x), we update x such that: $x_{k+1} = x_k + \alpha_k * p_k$ where

α_k: step size for k_{th} iteration

p_k: $-\nabla f(x_k)$ the direction of k_{th} iteration

2. Newton's Method

To minimize f(x), we update x such that: $x_{k+1} = x_k + p_k$ where

α_k: step size for k_{th} iteration

p_k: $-(\nabla^2 f(x_k))^{-1}\nabla f(x_k)$ the direction of k_{th} iteration

3. BFGS Quasi-Newton

To minimize f(x), we update x such that: $x_{k+1} = x_k + \alpha_k * p_k$ where

α_k: step size for k_{th} iteration

p_k: direction of k_{th} iteration

Update secant parameters:

$$s_k = x_{k+1} - x_k$$

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

$$\Delta \mathrm{B} = -\frac{(B_k * s_k)(B_k * s_k)^{\boldsymbol{T}}}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

$$B_{k+1} = B_k + \Delta \mathrm{B}$$

$$p_{k+1} = -{B_{k+1}}^{-1} * \nabla f(x_{k+1})$$

4. Conjugate Gradient Method

To minimize f(x), we update x such that: $x_{k+1} = x_k + \alpha_k * d_k$ where

α_k: step size for k_{th} iteration

d_k: conjugate direction of k_{th} iteration

Update parameters:

$$\alpha_k = -\frac{d_k^T * \nabla f(x_{k+1})}{d_k^T * Q * d_k}$$

$$\beta_k = \frac{d_k^T * Q * \nabla f(x_{k+1})}{d_k^T * Q * d_k}$$

$$d_{k+1} = -\nabla f(x_{k+1}) + \beta_k * d_k$$

Note: The step size (α_k) is updated using line search wherever applicable except polynomial cases in Conjugate Gradient method where instead of updating step, we update Q which is known as Fletcher-Reeves method. Since Q is a Hessian matrix, it may have storage limitations during highly complex calculations involving multivariate equations.

Evaluation Cases

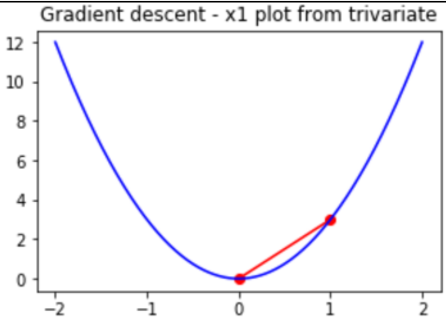
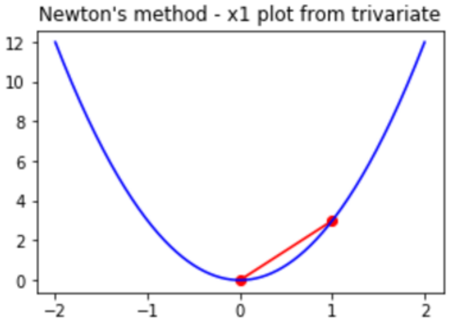
NOTE: For functional graph plots, please note that the represented function by no means is the actual function as the plot is only against X_1 and $f(X_1, X_2,\ldots X_i)$. Representing plots in greater than two dimensions has its own challenges. Below 2-D plot is one of the way to estimate comparative study between different algorithm's convergence rates.

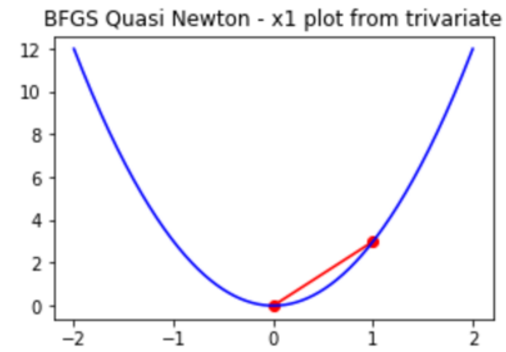
Hence, it may be possible is the initial step values lies outside the line curve which essentially means that it is mapped to a higher dimension.

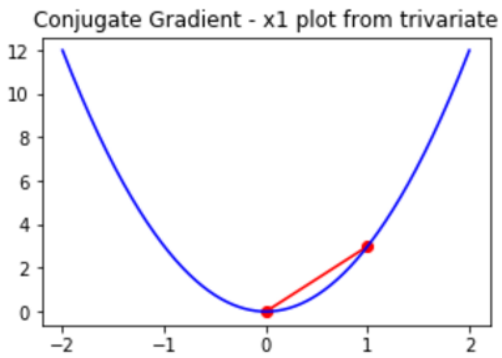
1. Example: function = $x_1^2 + x_2^2 + x_3^2$ and $x_0 = (1, 1, 1)^T$

Quadratic tri-variate Function	**Steps to Convergence**	**Initial Step Length**	**Initial Direction**	**Runtime**
Steepest Descent	1	0.5	$(-2\ -2\ -2)^T$	0.0100071
Newton's Method	1	NA	$(-1\ -1\ -1)^T$	0.0079815
BFGS Quasi-Newton	1	0.5	$(-2\ -2\ -2)^T$	0.0089840
Conjugate Gradient	1	0.5	$(-2\ -2\ -2)^T$	0.0069816

Below are the convergence plots for one variable.

Steepest Descent	***Newton's Method***
	
BFGS Quasi-Newton	***Conjugate Gradient***



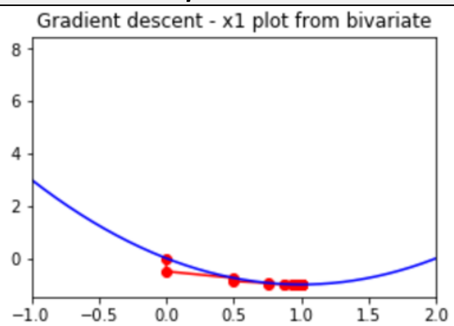


2. Example: function = $x_1^2 + 2x_2^2 - 2x_1x_2 - 2x_2$ and $x_0 = (0, 0)^T$

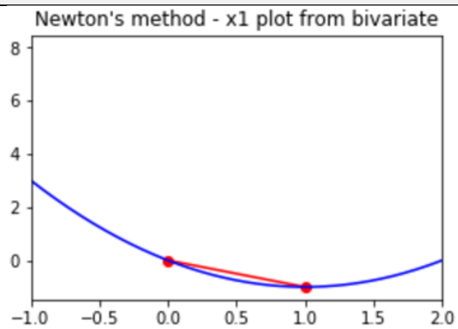
Quadratic Bivariate Function	**Steps to Convergence**	**Initial Step Length**	**Initial Direction**	**Runtime**
Steepest Descent	33	0.25	$(0\ 2)^T$	0.0740537
Newton's Method	1	NA	$(1\ 1)^T$	0.0060009
BFGS Quasi-Newton	2	0.25	$(0\ 2)^T$	0.0120358
Conjugate Gradient	2	0.25	$(0\ 2)^T$	0.0150110

Below are the convergence plots for one variable.

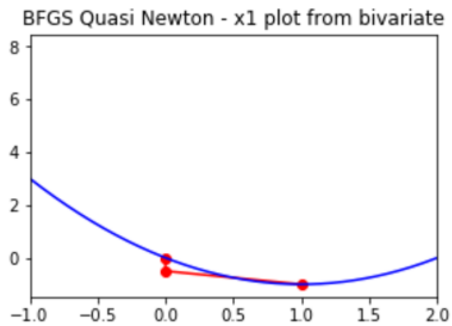
Steepest Descent



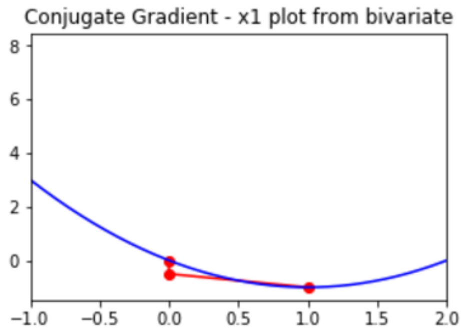
Newton's Method



BFGS Quasi-Newton



Conjugate Gradient

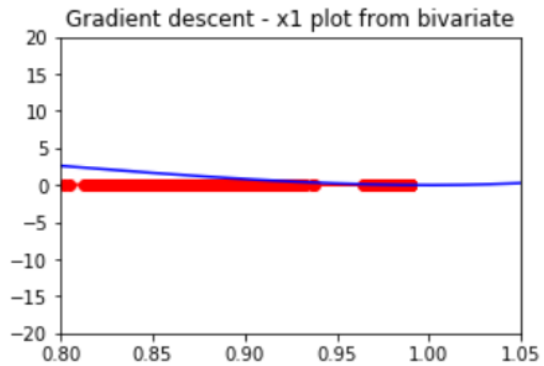


3. Example: function = $100(x_2 - x_1^2)^2 + (1-x_1)^2$ and $x_0 = (-1.2, 1)^T$

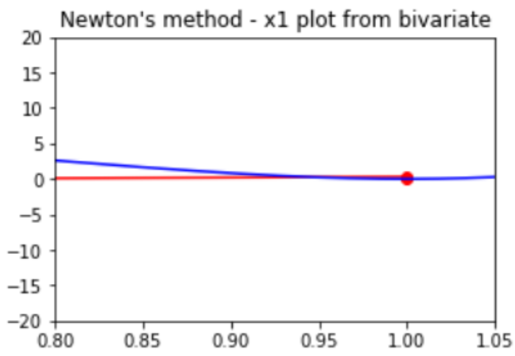
Polynomial Bivariate Function	**Steps to Convergence**	**Initial Step Length**	**Initial Direction**	**Runtime**
Steepest Descent	> 1000	0.00093	$(215.6\ 88)^T$	2.7049007 +
Newton's Method	5	NA	$(0.0247\ 0.3806)^T$	0.0270168
BFGS Quasi-Newton	35	0.00093	$(215.6\ 88)^T$	0.0900614
Conjugate Gradient	29	0.00066	$(215.6\ 88)^T$	0.0630214

Below are the convergence plots for one variable.

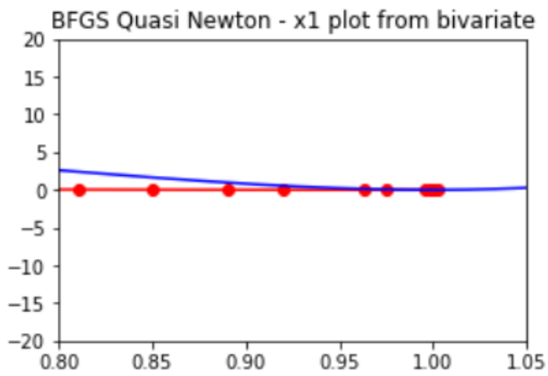
Steepest Descent



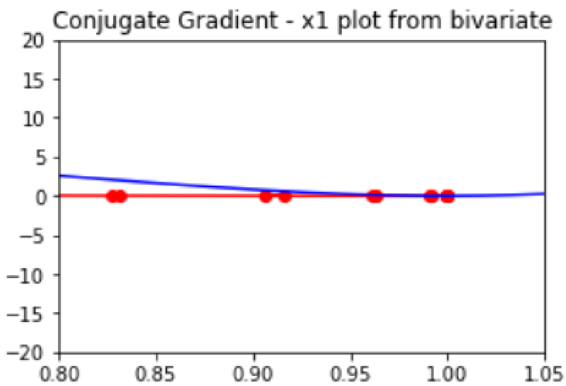
Newton's Method



BFGS Quasi-Newton



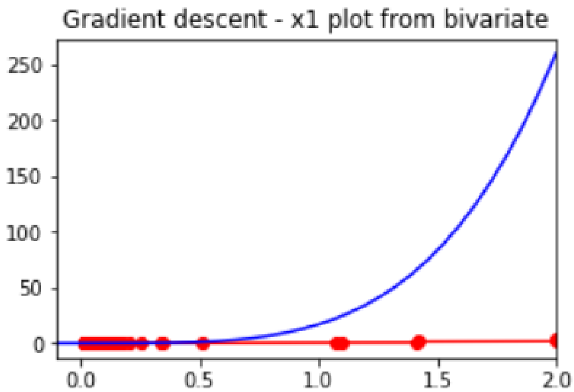
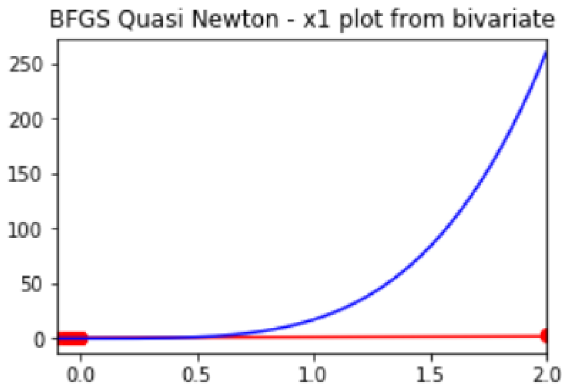
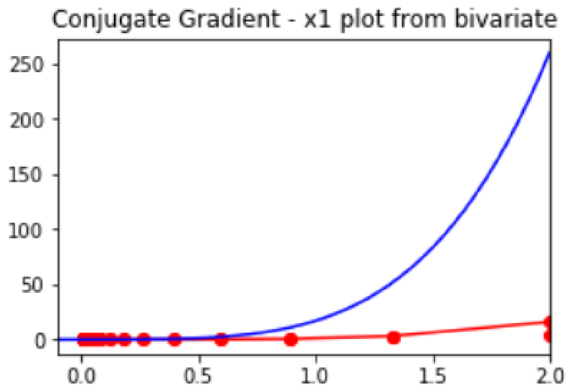
Conjugate Gradient



4. Example: function = $(x_1+x_2)^4 + x_2^2$ and $x_0 = (2, -2)^T$

Polynomial Bivariate Function	**Steps to Convergence**	**Initial Step Length**	**Initial Direction**	**Runtime**
Steepest Descent	711	0.25	$(0\ 4)^T$	0.6254215
Newton's Method	Singularity	NA	NA	NA
BFGS Quasi-Newton	26	0.25	$(0\ 4)^T$	0.0726697
Conjugate Gradient	38	0.5	$(0\ 4)^T$	0.0710306

Below are the convergence plots for one variable.

<table>
<tr><th>Steepest Descent</th><th>Newton's Method</th></tr>
<tr><td></td><td>Singularity. Newton's method fails to converge and errors out at the first step.
LinAlgError: Singular matrix</td></tr>
<tr><th>BFGS Quasi-Newton</th><th>Conjugate Gradient</th></tr>
<tr><td></td><td></td></tr>
</table>

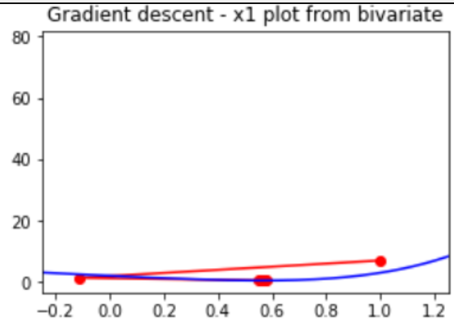
5. Example: function = $(x_1-1)^2 + (x_2-1)^2 + c(x_1^2 + x_2^2 - 0.25)^2$ and $x_0 = (1, -1)^T$

5.1 Example: c = 1

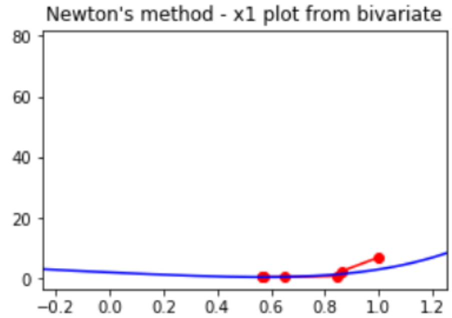
Polynomial Bivariate Function	**Steps to Convergence**	**Initial Step Length**	**Initial Direction**	**Runtime**
Steepest Descent	6	0.15	$(0.0003\ 0.0003)^T$	0.0330238
Newton's Method	6	NA	$(-0.1377\ 0.5822)^T$	0.0230166
BFGS Quasi-Newton	9	0.15	$(-7\ 11)^T$	0.0560398
Conjugate Gradient	11	0.041	$(-7\ 11)^T$	0.0481445

Below are the convergence plots for one variable.

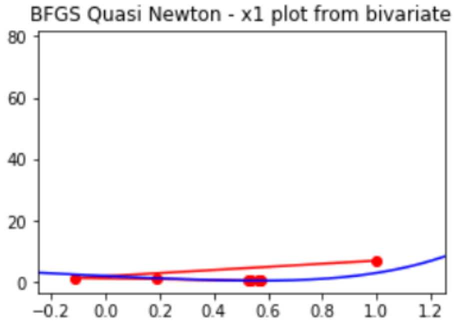
Steepest Descent



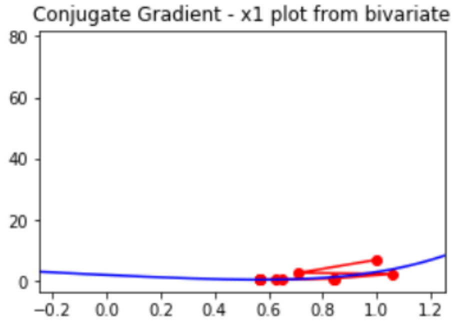
Newton's Method



BFGS Quasi-Newton



Conjugate Gradient

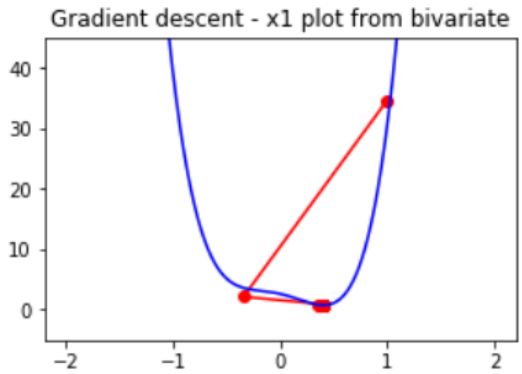


5.2 Example: c = 10

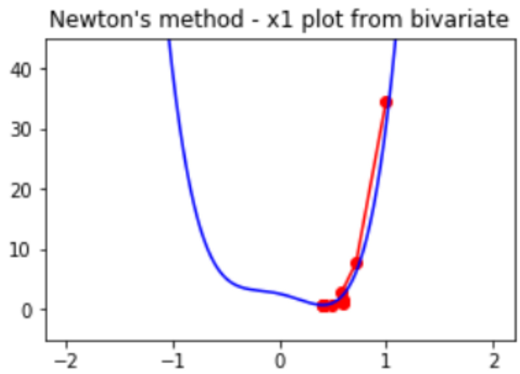
Polynomial Bivariate Function	**Steps to Convergence**	**Initial Step Length**	**Initial Direction**	**Runtime**
Steepest Descent	26	0.019	$(-70\ 74)^T$	0.0830721
Newton's Method	9	NA	$(-0.2825\ 0.3381)^T$	0.0410299
BFGS Quasi-Newton	12	0.019	$(-70\ 74)^T$	0.0580441
Conjugate Gradient	16	0.0043	$(-70\ 74)^T$	0.0520136

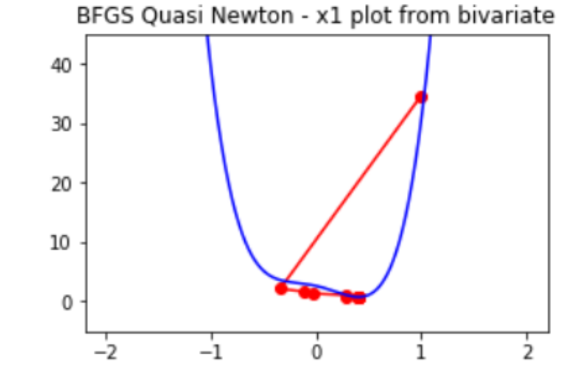
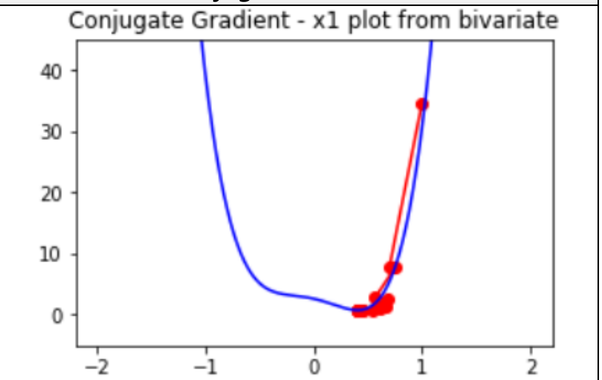
Below are the convergence plots for one variable.

Steepest Descent



Newton's Method

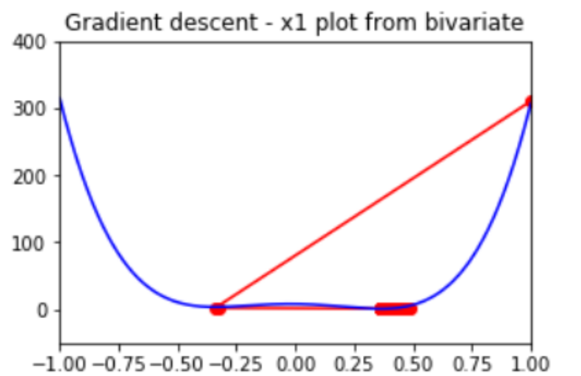
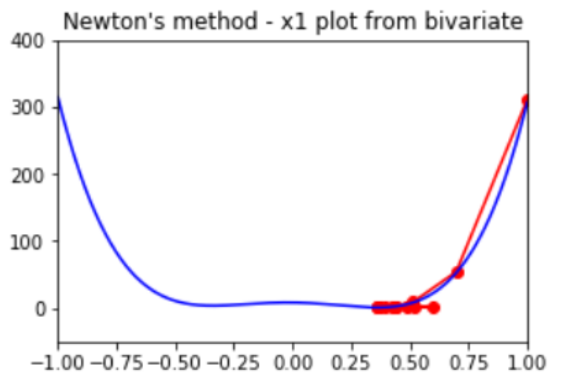
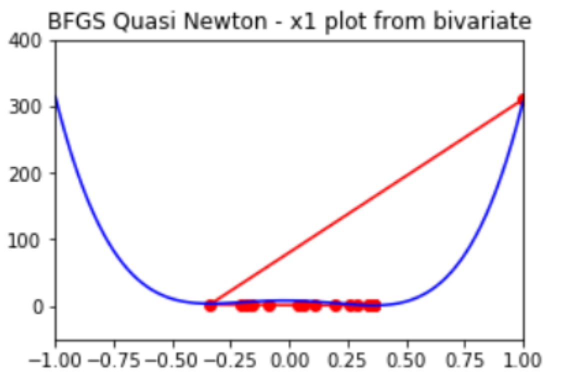
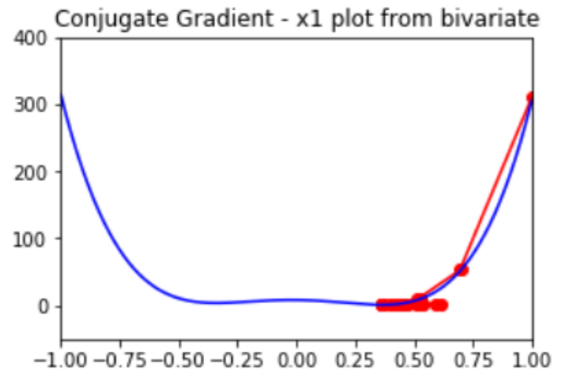


BFGS Quasi-Newton	***Conjugate Gradient***
	

5.3 Example: c = 100

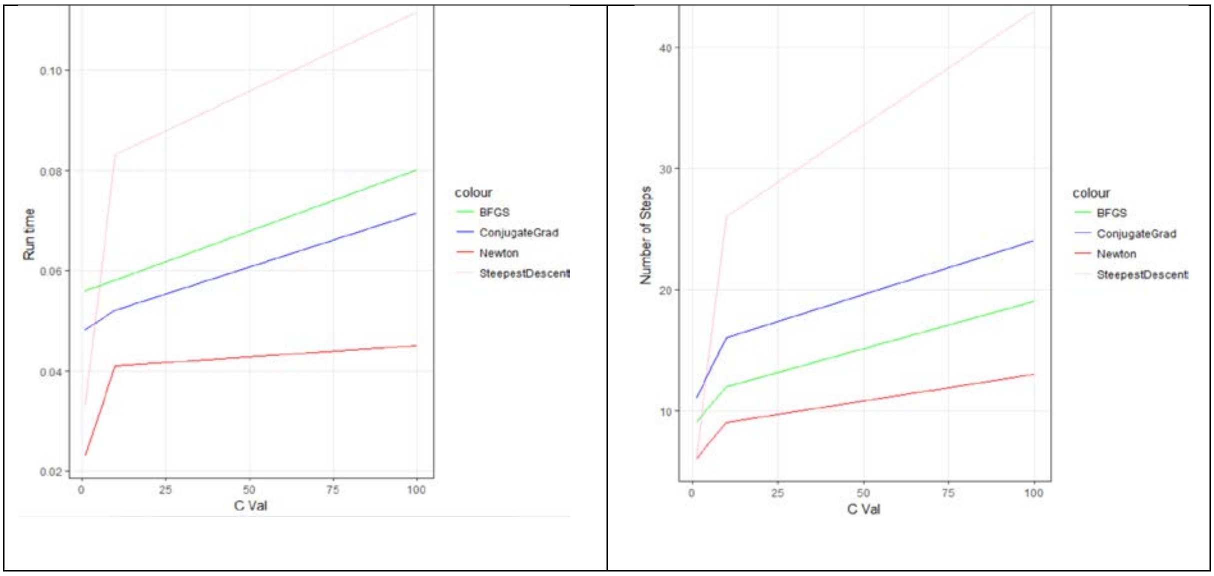
Polynomial Bivariate Function	**Steps to Convergence**	**Initial Step Length**	**Initial Direction**	**Runtime**
Steepest Descent	43	0.00191	$(-700\ 704)^T$	0.1116192
Newton's Method	13	NA	$(-0.3021\ 0.3078)^T$	0.0450334
BFGS Quasi-Newton	19	0.00191	$(-700\ 704)^T$	0.0800561
Conjugate Gradient	24	0.00043	$(-700\ 704)^T$	0.0714280

Below are the convergence plots for one variable.

Steepest Descent	***Newton's Method***
	
BFGS Quasi-Newton	***Conjugate Gradient***
	

Conclusion

We notice that Newton's method performance better in terms of runtime and steps to converge. But it must be noted that the above examples are mostly bivariate. Since Newton's method requires computing the Hessian and its inverse, there will be computational overhead and storage problems once the number of variables vastly increase. Also, at one instance the inverse could not be computed due to singularity which introduces another problem for Newton's method. This issue may be handled by processors with higher bit computing architecture.



The above plots represent how condition number affects convergence and run time. It is observed that as C increases, the run time as well as the convergence steps increases. Steepest descent is highly impacted with the increase in C value which Newton's method remains comparatively less affected.

The choice of the algorithm remains highly subjective to the use case as each algorithm comes with its own challenges.