Project 2: Unconstrained Optimization

Faezeh Habibi and Ye Zheng.

We present the convergence results for the given three functions by Gradient Descent, Newton Method, and Quasi-Newton, and by our implemented Adam optimizer.

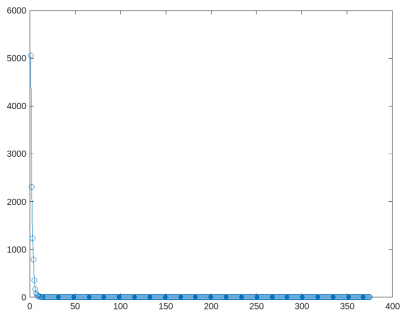
Function 1

Backtracking Line Search: $\rho = 0.9, c = 0.1$, initial step $\alpha = 1$.

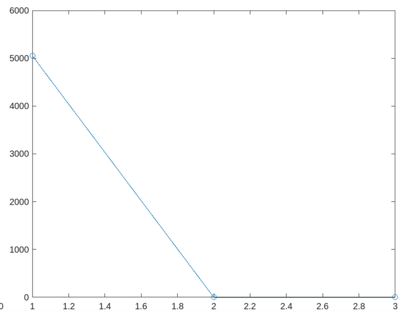
Start point: $x = ones(100, 1)$.

Method	Gradient Descent	Newton Method	Quasi-Newton
Founded Minimum	0	0	0
Iteration Number	373	1	117

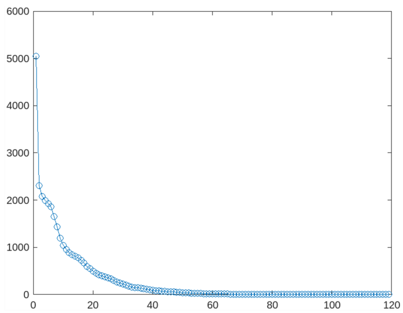
with convergence figures:



Gradient Descent



Newton Method



Quasi-Newton

Analysis

For function 1, its gradient and Hessian are:

$$\nabla f_i = 2ix_i, \quad H_{ij} = \begin{cases} 2i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

the gradient function has single zero-point $[0; 0; 0; \ldots; 0]$ and the Hessian function is strict positive, so the global minimum is 0. It is also a quadratic function that satisfying the assumption of Newton Method, so Newton Method can find its minima in one step.

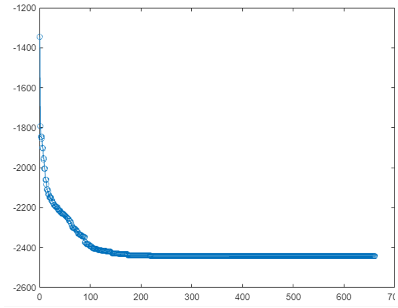
Function 2

Backtracking Line Search: $\rho = 0.9, c = 0.1$, initial step $\alpha = 1$.

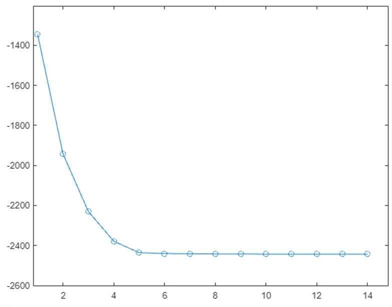
Start point: $x = zeros(100, 1)$.

Method	Gradient Descent	Newton Method	Quasi-Newton
Founded Minimum	-2.4432×10^3	-2.4432×10^3	-2.4432×10^3
Iteration Number	659	12	266

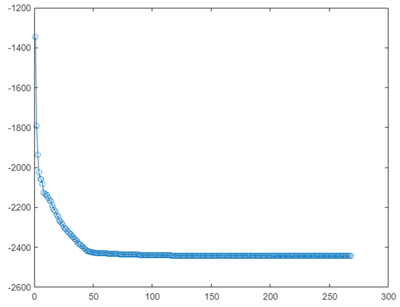
with convergence figures:



Gradient Descent



Newton Method



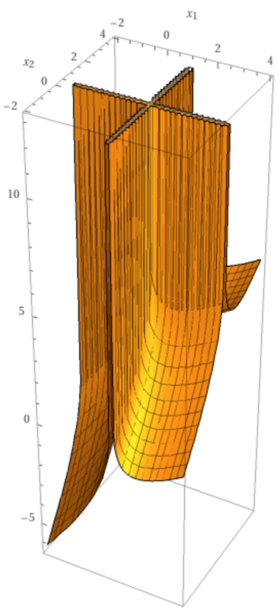
Quasi-Newton

Analysis

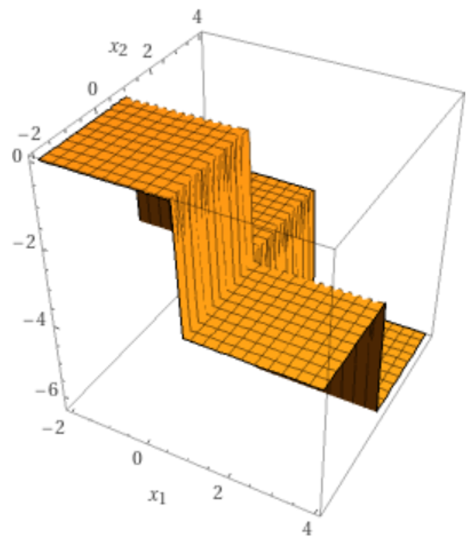
If we fix $n = 2, c = [1; 1], A = [1, 1; 1, 1], b = [1; 1]$ for function 2, i.e.

$$f = x_1 + x_2 - (\ln(1 - x_1) + \ln(1 - x_2))$$

its value plot w.r.t x_1, x_2 is:



Real part



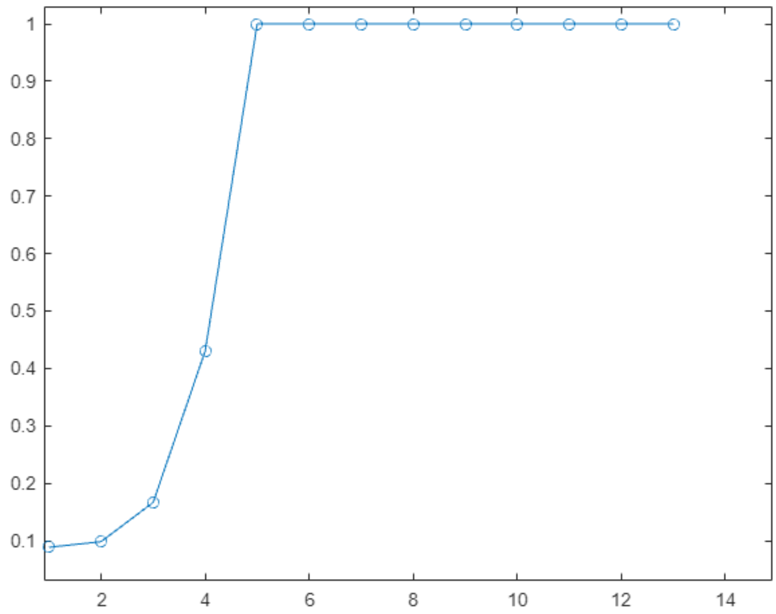
Imaginary part

Under the strict definition of $\log$ function, the above function is only defined on the $x_1 < 1 \wedge x_2 < 1$.

For the original 100-dimensional function, it is not defined on the whole R^{100} domain. Under some inputs, $\log$ will output complex number (log a negative x). To address this, we:

- find $zeros(100, 1)$ is an acceptable start point.
- lower the step α if $b - Ax < 0$, until $\log b - Ax$ is defined.
- lower the tolerance to 10^{-5} and the maximum iteration number 10^3 for gradient descent.

The following figure is the 12 chosen α in Newton method:



It can be found that even for Newton Method, the first four steps are very small.

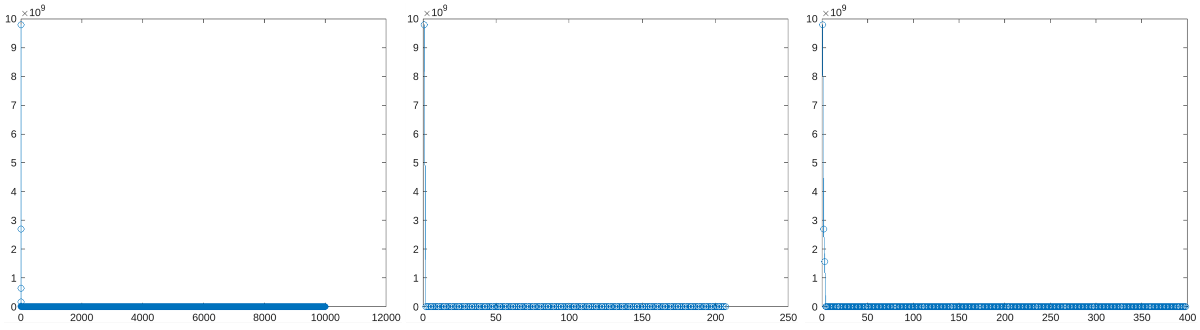
Function 3

Backtracking Line Search: $\rho = 0.9, c = 0.1$, initial step $\alpha = 1$.

Start point: $[100; 100]$.

Method	Gradient Descent	Newton Method	Quasi-Newton
Founded Minimum	85	0	0
Iteration Number	10^4 (max)	205	396
Minima $[x_1, x_2] =$	$[10.2, 104.8]$	$[1, 1]$	$[1, 1]$

with convergence figures:

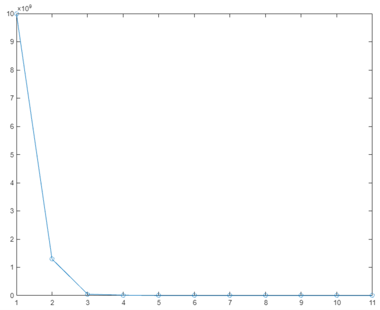


Gradient Descent Newton Method Quasi-Newton

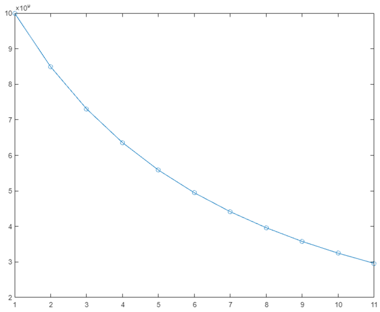
We also tried other start points for Gradient Descent:

Start Point	Founded Minimum	Iteration Number	Minima $[x_1, x_2] =$
$[10, 10]$	4.9	1000 (max)	$[3.2, 10.3]$
$[1, 10]$	4.2	1000 (max)	$[3.0, 9.2]$

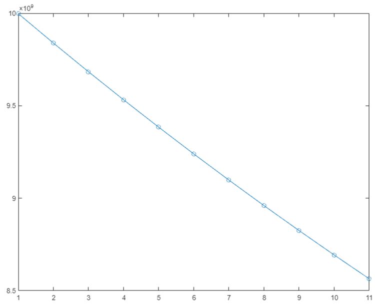
Generally, smaller step means slower convergence. We test smaller steps in backtracking line search (*c* = 0.001 and different *ρ*), the following is the first 10 steps:



ρ=0.1



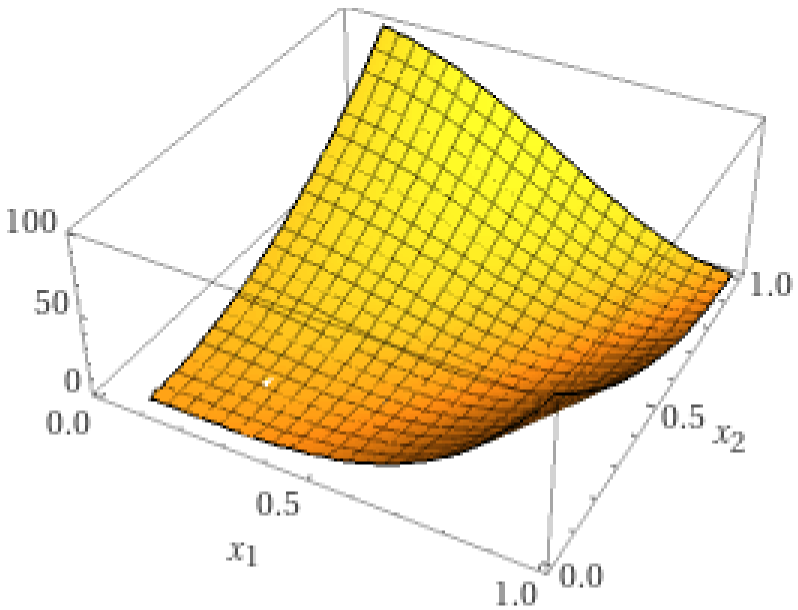
ρ=0.01



ρ=0.001

Analysis

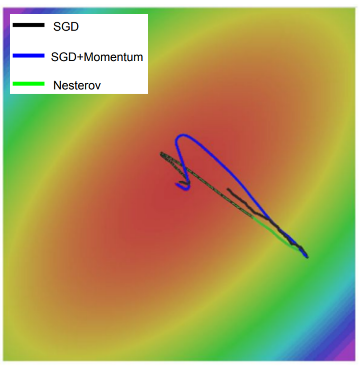
The following is the plot figure of function 3:



It is a non-convex function. In fact, this function has global minima at $[1, 1]$, we find this under the help of Wolfram Alpha. Here we can see the drawback of Gradient Descent: once it stuck into a local minima, it can not escape.

Adam Optimizer

Adam (Adaptive Moment Estimation) optimizer is used for highly non-convex problem. Such optimizers does not very believe the local gradients. They use "momentum" to add noise to the gradient to escape local minimum. For example, the trace of SGD+Momentum may be like the following: (picture from Google)

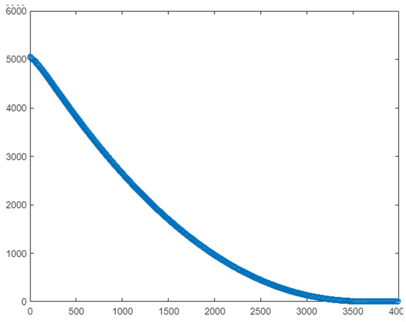


An interesting thing is there exist a so-called best learning rate (step length) of Adam optimizer:

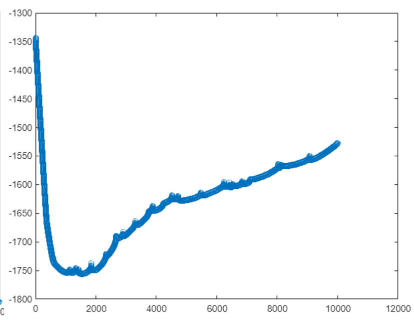
So we implement Adam for solving the above three closed-from functions, with the famous parameter $\alpha = 3e - 4$. Following is the results:

	Function 1	Function 2	Function 3
Founded Minimum	0	-1.5×10^3	0
Iteration Number	4.0×10^3	1×10^4 (max)	3.3×10^5

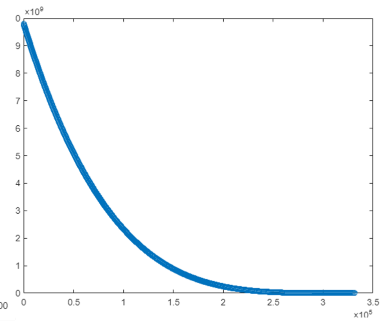
with the convergence figures:



Function 1



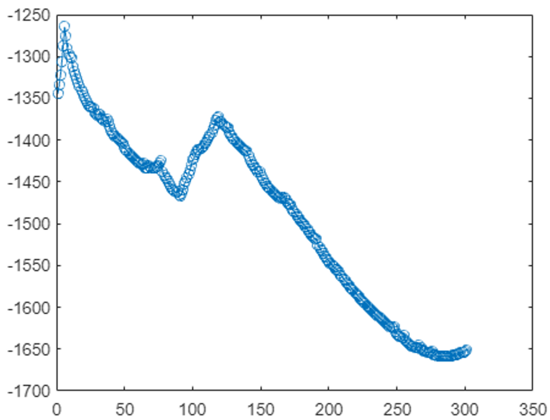
Function 2



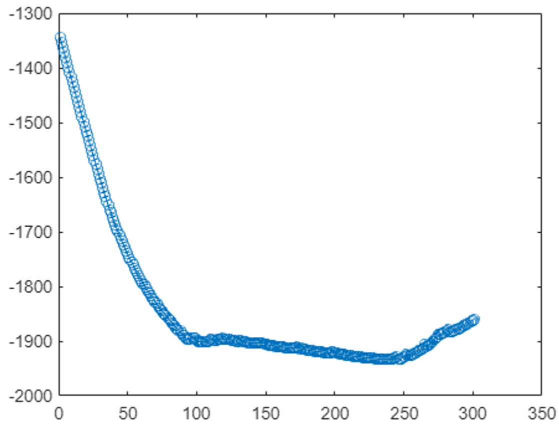
Function 3

Due to the very small step length (and even decreasing), the convergence rate of Adam is much slower than the above fully gradient-guided methods. And it does not guarantee decreasing the value of objective function at each step (as backtracking line search).

We then tried much bigger learning rate:



Learning rate 0.1



Learning rate 0.01

We can see the decreasing learning rate of Adam (increasing point density). Due to the complex definition domain, it seems hard for Adam to get the "correct" direction.