CSOPT : Optimisation without Constraints
Chaitanya Krishna VIRIYALA

1. Rosenbrock function minimisation

1.1 Preliminary work (Hand written notes)

OPTIMISATION WITHOUT CONSTRAINTS

1. Rosenbrock function minimisation

$$f_0(x) = \sum_{i=1}^{n-1} b(x_{i+1} - x_i^2)^2 + (1-x_i)^2 \quad \forall\ x \in \mathbb{R}^n,\ b \in \mathbb{R}$$

1.1.1. Gradient and Hessian

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$ is a $n \times 1$ column matrix.

$$\boxed{\frac{\partial f}{\partial x_1} = 2b(-2x_1)(x_2 - x_1^2) - 2(1-x_1)}$$

$$\frac{\partial f}{\partial x_k} = \frac{\partial}{\partial x_k}\left[\sum_{\substack{i \neq k \\ i \neq k-1}} m_i + m_k + m_{k-1}\right]$$

with $m_i = b(x_{i+1} - x_i^2)^2 + (1-x_i)^2$

$$2 \leq k \leq n : \quad \frac{\partial f}{\partial x_k} = \frac{\partial}{\partial x_k} m_k + \frac{\partial}{\partial x_k} m_{k-1}$$

$$\frac{\partial}{\partial x_k} m_k = \frac{\partial}{\partial x_k}\left[b(x_{k+1} - x_k^2)^2 + (1-x_k)^2\right]$$

$$= -4bx_k(x_{k+1} - x_k^2) - 2x_k(1-x_k)$$

$$\frac{\partial m_{k-1}}{\partial x_k} = \frac{\partial}{\partial x_k}\left[b(x_k + x_{k-1}^2)^2 + (1-x_{k-1})^2\right] = 2b(x_k + x_{k-1}^2)$$

$$\boxed{\frac{\partial f}{\partial x_k} = -4b x_k(x_{k+1} - x_k^2) - 2x_k(1-x_k) + 2b(x_k + x_{k-1}^2)}$$

$$H_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \frac{\partial^2 f}{\partial x_i \partial x_j} & \vdots \\ & & & \ddots \frac{\partial^2 f}{\partial x_n^2} \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & & \end{bmatrix} \quad n \times n \text{ matrix}$$

$$\frac{\partial^2 f}{\partial x_k^2} = \frac{\partial}{\partial x_k}\left(\frac{\partial f}{\partial x_k}\right)$$

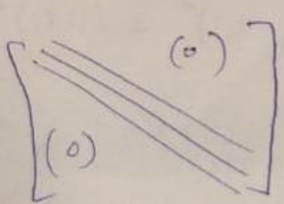
$$= \frac{\partial}{\partial x_k}\left(2b(x_k + x_{k-1}^2) - 4b x_k (x_{k+1} - x_k^2) - 2x_k(1 - x_k) \right)$$

$$= 2b - 4b + 8b x_k^2 - 2 + 4x_k$$

$$\frac{\partial^2 f}{\partial x_k^2} = 8b x_k^2 + 4x_k - 2b - 2$$

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j}\right) = \frac{\partial}{\partial x_i}\left(2b(x_j + x_{j-1}^2) - 4b x_j (x_{j+1} - x_j^2) - 2x_j(1 - x_j) \right)$$

H_f is a symmetric matrix

$H(f) =$ 

Hessian has only three diagonals: principal, superior and inferior.

1.1.2 f belongs to Class 2 C-2

$\nabla f = 0$ and $\nabla^2 f > 0$, then minima exists.

Here the global minimum point is at $\underline{\underline{(1, 1)}}$

1.1.3 Modified Rosenbrock problem

$$\phi(x) = \frac{1}{2}\left(f_1^2 + f_2^2\right)$$

$$f_1 = b\left(x_{i+1} - x_i^2\right)$$

$$f_2 = 1 - x_i$$

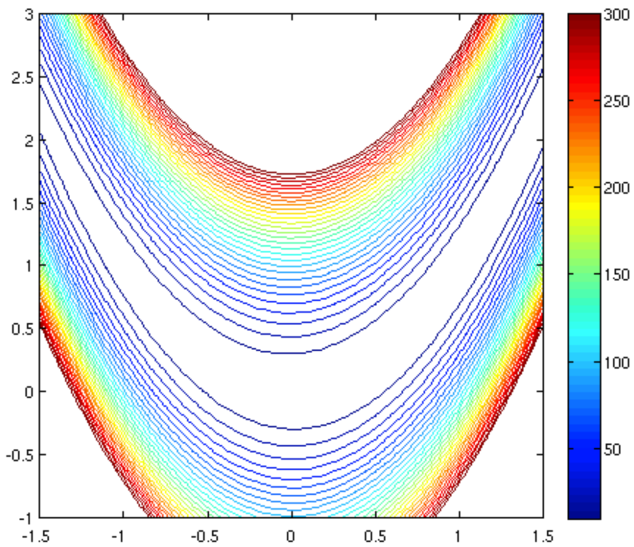
$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_i} & \frac{\partial f}{\partial x_{i+1}} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_{i+1}} \end{pmatrix} = \begin{pmatrix} -2bx_i & b \\ -1 & 0 \end{pmatrix}$$

1.2 Visualisation of Objective Function

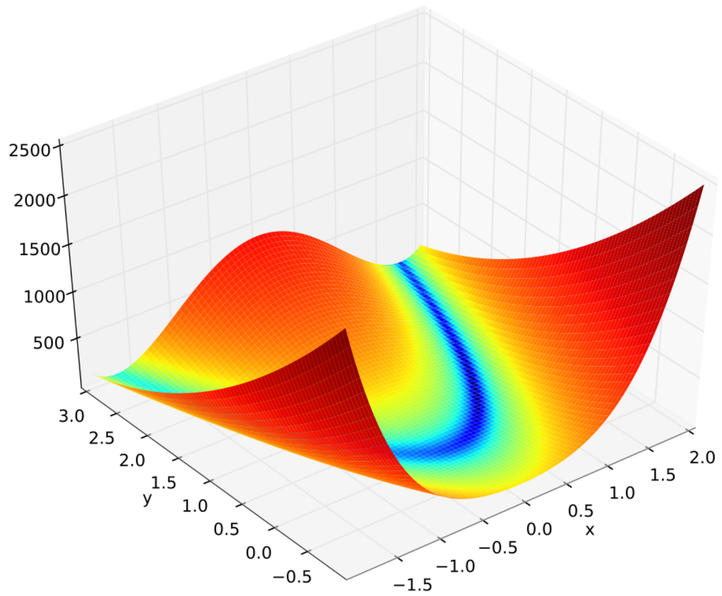
1. Complete the Rosenbrock function for n = b = 2

```
function [f,g,H] = rosenbrock(x)
% x : valeur de la variable d'optimisation
% params : structure contenant les parametres necessaires pour evaluer la fonction objectif
% f,g,h : valeur de la fonction objectif, son gradient et son hessien
    f = 2*(x(2)-x(1)^2)^2 + (1-x(1))^2;
    g(1) = - 8*(x(2)-x(1)^2)*x(1) - 2*(1-x(1)) ;
    g(2) = 4*(x(2)-x(1)^2);
    H(1,1) = 24*x(1)^2 + 8*x(2) + 2;
    H(1,2) = -8*x(1);
    H(2,1) = H(1,2);
    H(2,2) = 4;
end
```

2. Visualisation

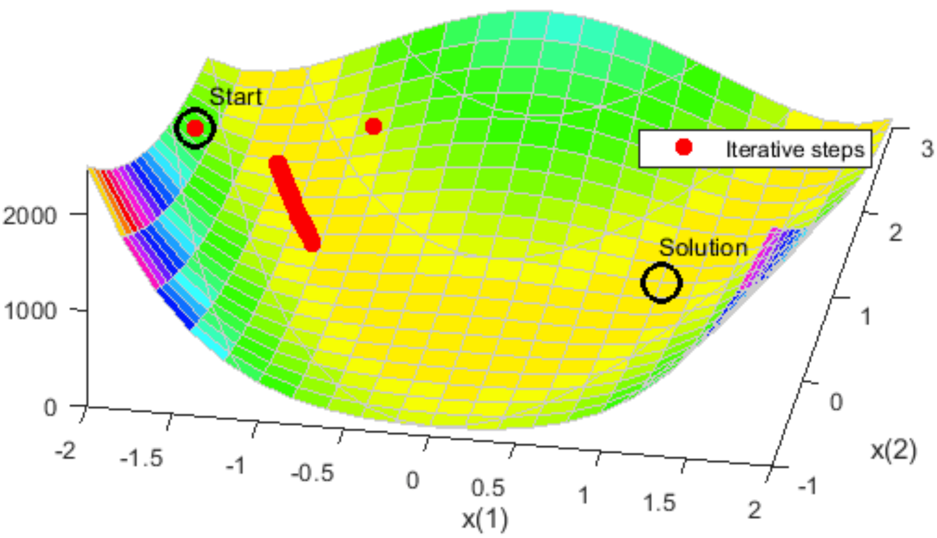



```
f = @(x,y) (1-x).^2 + 2*(y-x.^2).^2;

x = linspace(-4,4,0.05);
y = linspace(-5,20,0.05);
[xx,yy] = meshgrid(x,y);
ff = f(xx,yy);
levels = 10:10:300;
LW = 'linewidth'; FS = 'fontsize'; MS = 'markersize';
figure, contour(x,y,ff,levels,LW,1.2), colorbar
axis([-1.5 1.5 -1 3]), axis square, hold on
```



Rosenbrock Function, also known as Banana function

1.3.1 Steepest Descent Minimisation



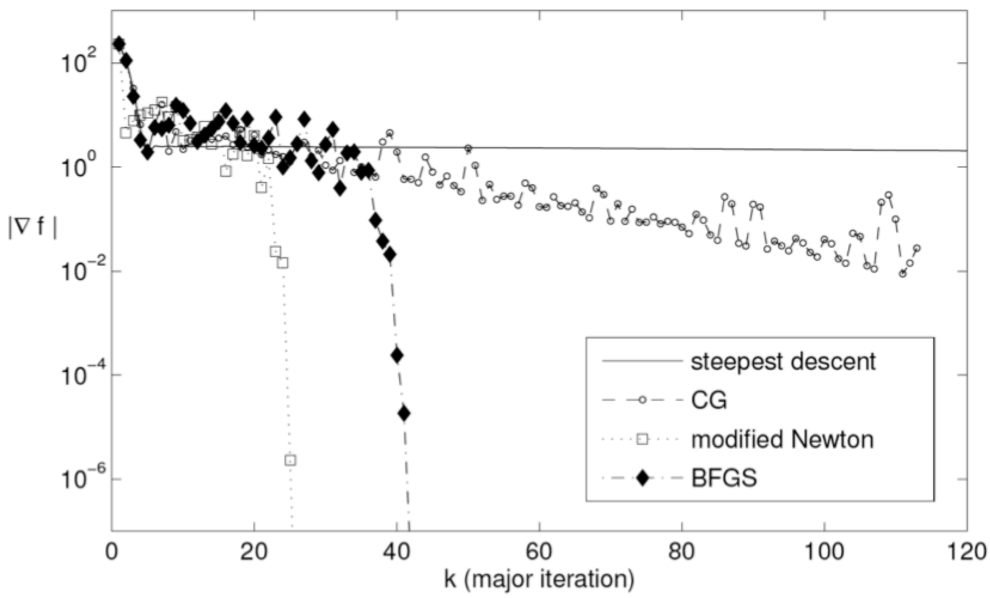
1.3.2

```
function alpha = mb_backtrackingLineSearch(objFunc,objFuncValue,x,dx,dir)

% backtracking line search using armijo criterion
% objFunc      - handle for objective function
% objFuncValue - current objective function value @ x
% x            - x
% dx           - dx
% dir          - search direction
%
% example : mb_backtrackingLineSearch(objFunc,objFuncValue,x,dx,dir)

alphaMax     = 1; % this is the maximum step length
alpha        = alphaMax;
fac          = 1/2; % < 1 reduction factor of alpha
c_1          = 1e-1;

while objFunc(x+alpha*dir) > objFuncValue + c_1*alpha*dir'*dx;

    alpha = fac*alpha;

    if alpha < 10*eps
        error('Error in Line search - alpha close to working precision');
    end

end

end
```

- The backtracking line search strategy starts with a relatively large step size, and repeatedly shrinks it by a factor

- The search will terminate after a finite number of steps for any positive values of c and tau that are less than 1

1.3.3 (code) 1.3.4 and 1.3.5, 6



This comparison is done using python for its ease of comparisons for different methods and also my proficiency with the same, attached in the folder

When we compare the convergence rates for the Rosenbrock function, we can see that the modified Newton converges the fastest, followed by BFGS and then the CG and then steepest descent method.

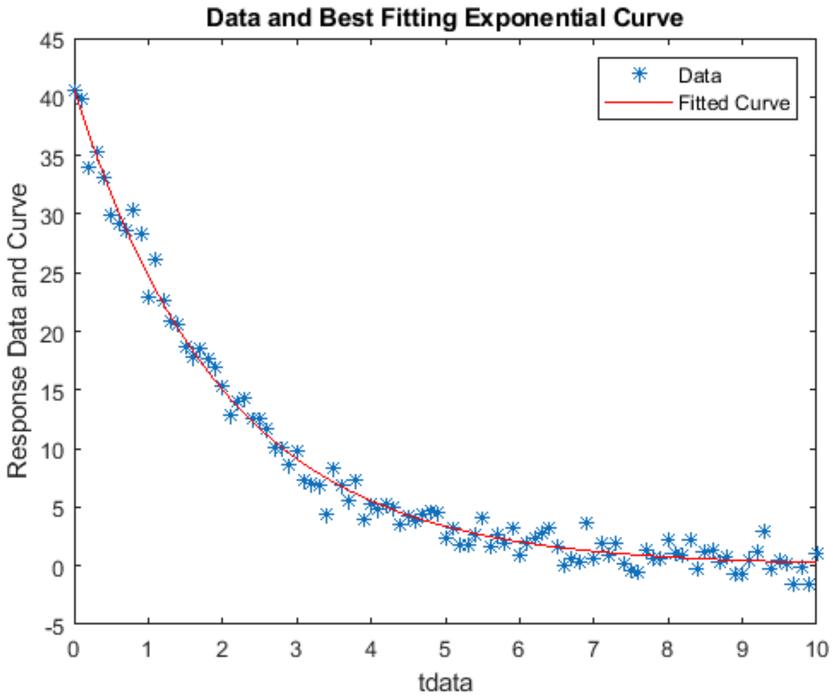
Steepest descent : The rate of convergence is linear.

Conjugate Gradient : The conjugate gradient method does not produce well-scaled search directions, so we can use same strategy to choose the initial step size as for steepest descent.

BFGS : This is generally considered to be the most effective quasi-Newton updates.

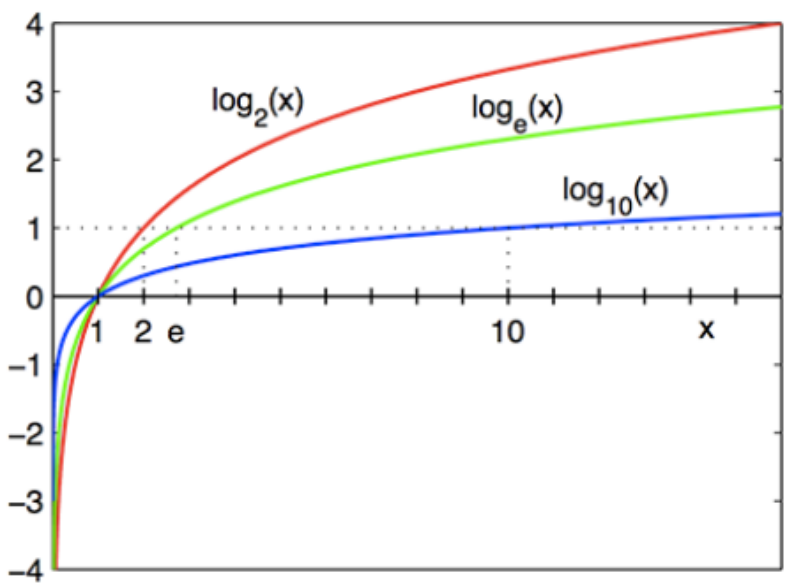
Modified Newton : A small modification to Newton's method is to perform a line search along the Newton direction, rather than accepting the step size that would minimise the quadratic model.

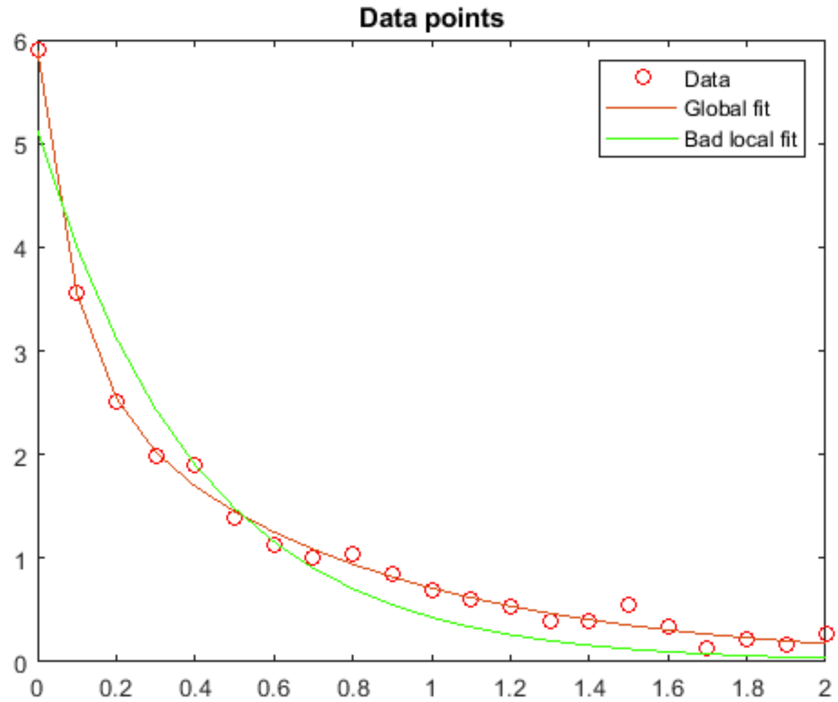
2. Adjustment of a non linear curve



This is a standard exponential decay curve, we are going to apply a few techniques of the BFGS, Quasi Newton

This is the log scale curve





When we compare the convergence rates for the Rosenbrock function, we can see that the modified Newton converges the fastest, followed by BFGS and then the CG and then steepest descent method.