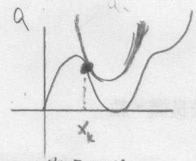
1. (20%) Show that Newton's method for optimization (for single variable) is using a quadratic model, which means in each step it

   (a) builds a tangent quadratic model at x_k;
   (b) finds the minimum (or the maximum) of the quadratic model x_{k+1}; and
   (c) walks from x_k to x_{k+1}.

a



依泰勒展開式

$f(x) \approx f(x_k) + f'(x_k)(x-x_k) + \frac{1}{2}f''(x_k)(x-x_k)^2$

b、

$\min f(x) \approx \min f(x_k) + f'(x_k)(x-x_k) + \frac{1}{2}f''(x_k)(x-x_k)^2$

$\Rightarrow$ 令 $f'(x) = 0 \Rightarrow f'(x_k) + f''(x_k)(x-x_k) = 0$

c、 $x = x_k - \frac{f'(x_k)}{f''(x_k)}$

2. (30%) Besides Taylor series, we can use interpolation to build polynomial models. For example, if we know $f(x_1), f(x_2)$ and $f(x_3)$, we can build $m(x) = ax^2 + bx + c$ where (a, b, c) are the solution of

$$\begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \end{pmatrix}. \tag{1}$$

We can design an algorithm to find $\min_x f(x)$.

- For x_k, build $m(x)$ with $f(x_k)$, $f(x_k + \alpha)$ and $f(x_k - \alpha)$ for some small $0 < \alpha$.
- Find $\min_x m(x)$.

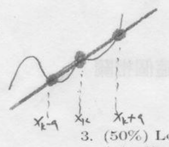
(a) Give an example to show this method can be failed.
(b) Compare the pros/cons of Newton's method and this method.
(c) We can also combine derivative method and interpolation method to build models. Suppose we know $f(x_1), f'(x_1), f(x_2)$ and $f'(x_2)$. Given an equation, like (1), to build a cubic model $m(x) = ax^3 + bx^2 + cx + d$.

a.

令f(x)為

a、Idea：先找不出 U 形的 model 都算失敗，∵無 min 值

example：



b Newton method

pros：因有Hessian的幫助，收斂很快

cons：1. Hessian（計算成本高，尤其size很大時）

2. Hessian 可能不存在反矩陣

interpolation

pros：計算成本相較 newton 便宜

cons：容易失敗，eg：

c、

$$\begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ x_2^3 & x_2^2 & x_2 & 1 \\ 3x_1^2 & 2x_1 & 1 & 0 \\ 3x_2^2 & 2x_2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ f'(x_1) \\ f'(x_2) \end{bmatrix}$$

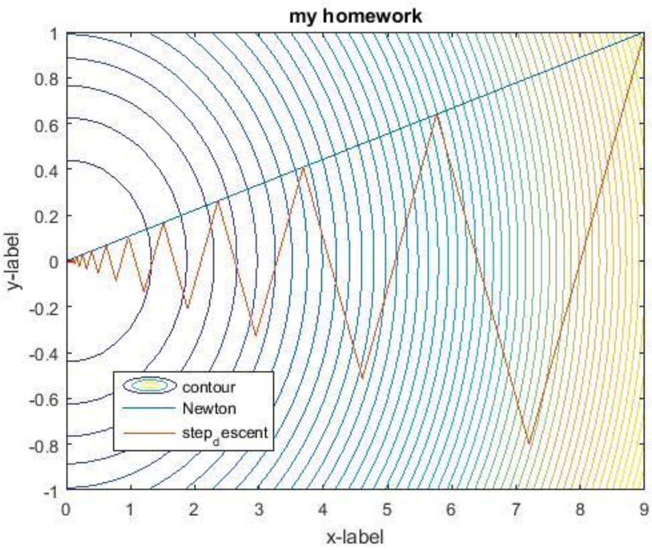
3. (50%) Let $f(x,y) = \frac{1}{2}x^2 + \frac{9}{2}y^2$. This is a positive definite quadratic with minimizer at $(x^*, y^*) = (0, 0)$.

   (a) Derive the gradient g and the Hessian H of f.
   (b) Write Matlab codes to implement the steepest descent method and Newton's method with $\vec{x}_0 = (9, 1)$, and compare their convergent results. The formula of the steepest descent method is

$$\vec{x}_{k+1} = \vec{x}_k - \frac{\vec{g}_k^T \vec{g}_k}{\vec{g}_k^T H_k \vec{g}_k} \vec{g}_k.$$

a、 gradient $g = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} x \\ 9y \end{bmatrix}$

Hessian $H = \begin{bmatrix} \frac{\partial f}{\partial x \partial x} & \frac{\partial f}{\partial x \partial y} \\ \frac{\partial f}{\partial x \partial y} & \frac{\partial f}{\partial y \partial y} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix}$



(此圖由於 y 軸間距 與 x 軸間距 不同 所以無法呈現, step descent 方向為該點之-f'(x))

由此圖可發現 step descent 在大方向來說,與 newton 方向是一致的.
都能找到最佳解.

若是 f(x) 呈現之形狀 為 凹型 (H 全部之 λ > 0) 與 newton 建立 motel 相符合 Or 相似
則 newton 能很快找到 solution (如圖,一步到達)

否則 step descent 是比較快速的方法.

```matlab
function draw_trace()
step = 0.1;
X = 0:step:9;
Y = -1:step:1;
n = size(X,2);
m = size(Y,2);
Z = zeros(m,n);
for j = 1:m
for i = 1:n
 Z(j,i) = f(X(i),Y(j));
end
end
contour(X,Y,Z,50);

hold on; % this is important!! This will overlap your plots.
Xs=[9;1];
g=[1;9];
H=[1,0;0,9];
pathS=step_descent(Xs);
pathN=Newton(Xs);
plot(pathN(1,:),pathN(2,:));
plot(pathS(1,:),pathS(2,:));
%axis  equal ;

title('my homework');xlabel('x-label');ylabel('y-label');
h_leg =legend('contour','Newton','step_descent');
set(h_leg,'position',[0.2 0.2 0.2 0.1]);
hold off;
 % function definition
 function z = f(x,y)
 z = (x*x+9*y*y)/2;
 end
```

```
function N = Newton(Xs)
   N = Xs;
   G=[1;1];
   while( ~isequal(round(G,5),[0;0]) )
     G=[g(1)*Xs(1);g(2)*Xs(2)];
     Xs = Xs-(H\G);
     N = [N Xs];
   end
 end
 function S = step_descent(Xs)
   S = Xs;
   G=[1;1];
    while( ~isequal(round(G,5),[0;0]) )
     G=[g(1)*Xs(1);g(2)*Xs(2)];
     Xs = Xs-(((G)'*H*(G))^(-1)*((G)'*(G)))*(G);
     S = [S Xs];
   end
 end
end
```