

# **Taylor Series Expansion & Newton Method**

```
>> g=inline('x^2+1')
```

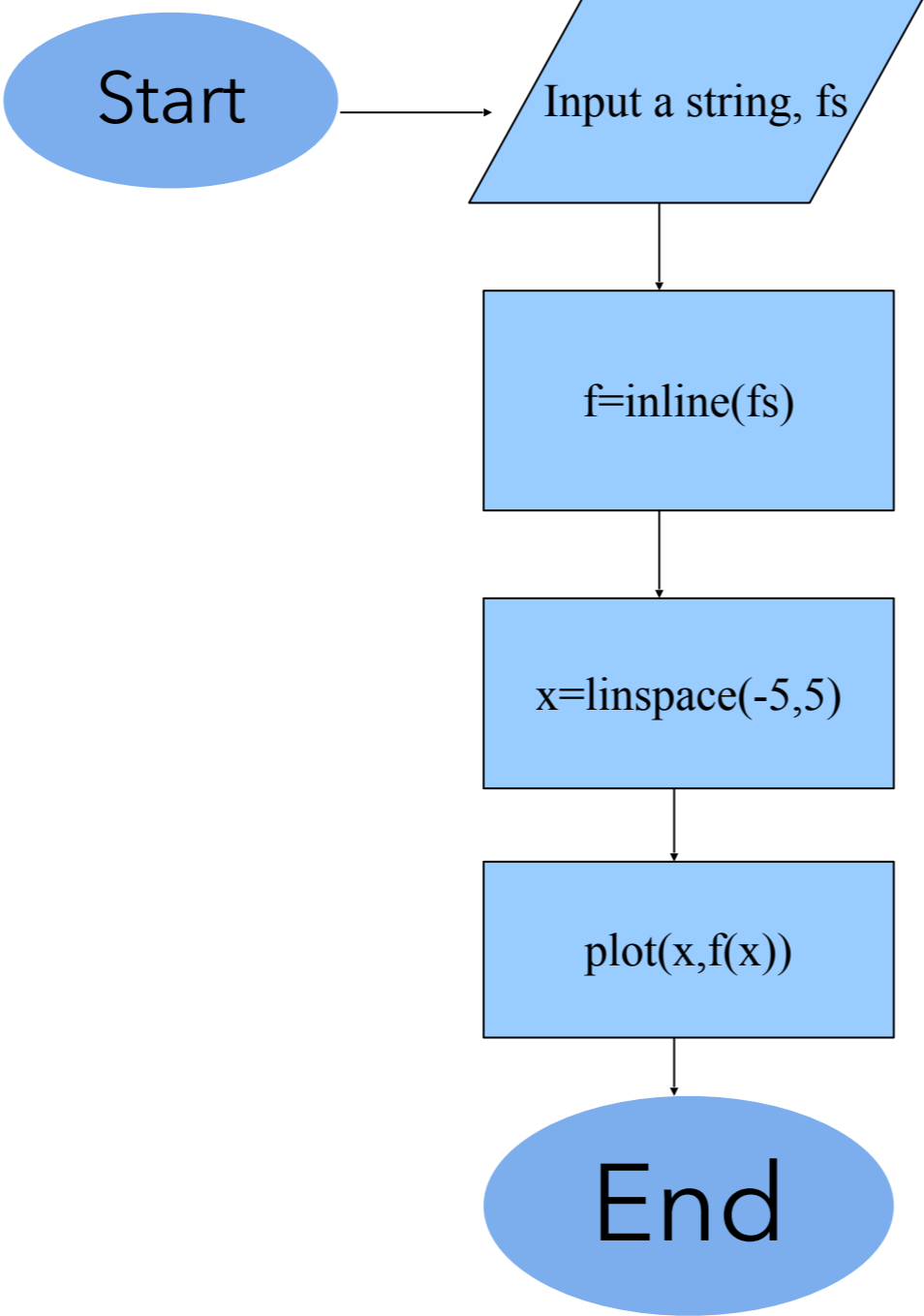
```
g =
```

Inline function:  
 $g(x) = x^2+1$

```
>> g(2)
```

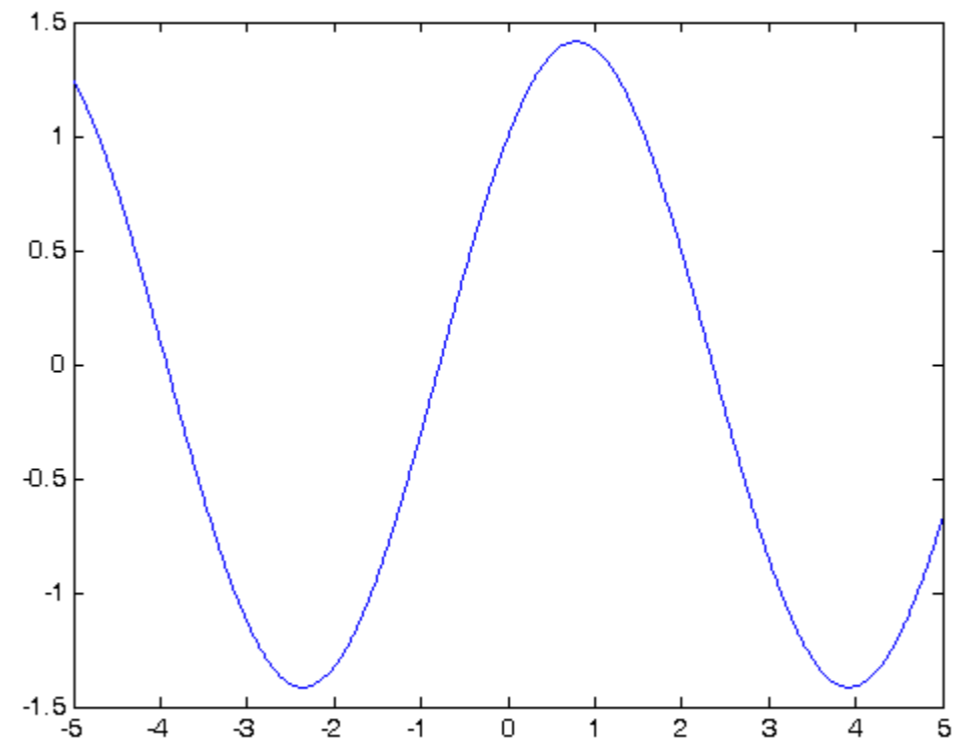
```
ans =
```

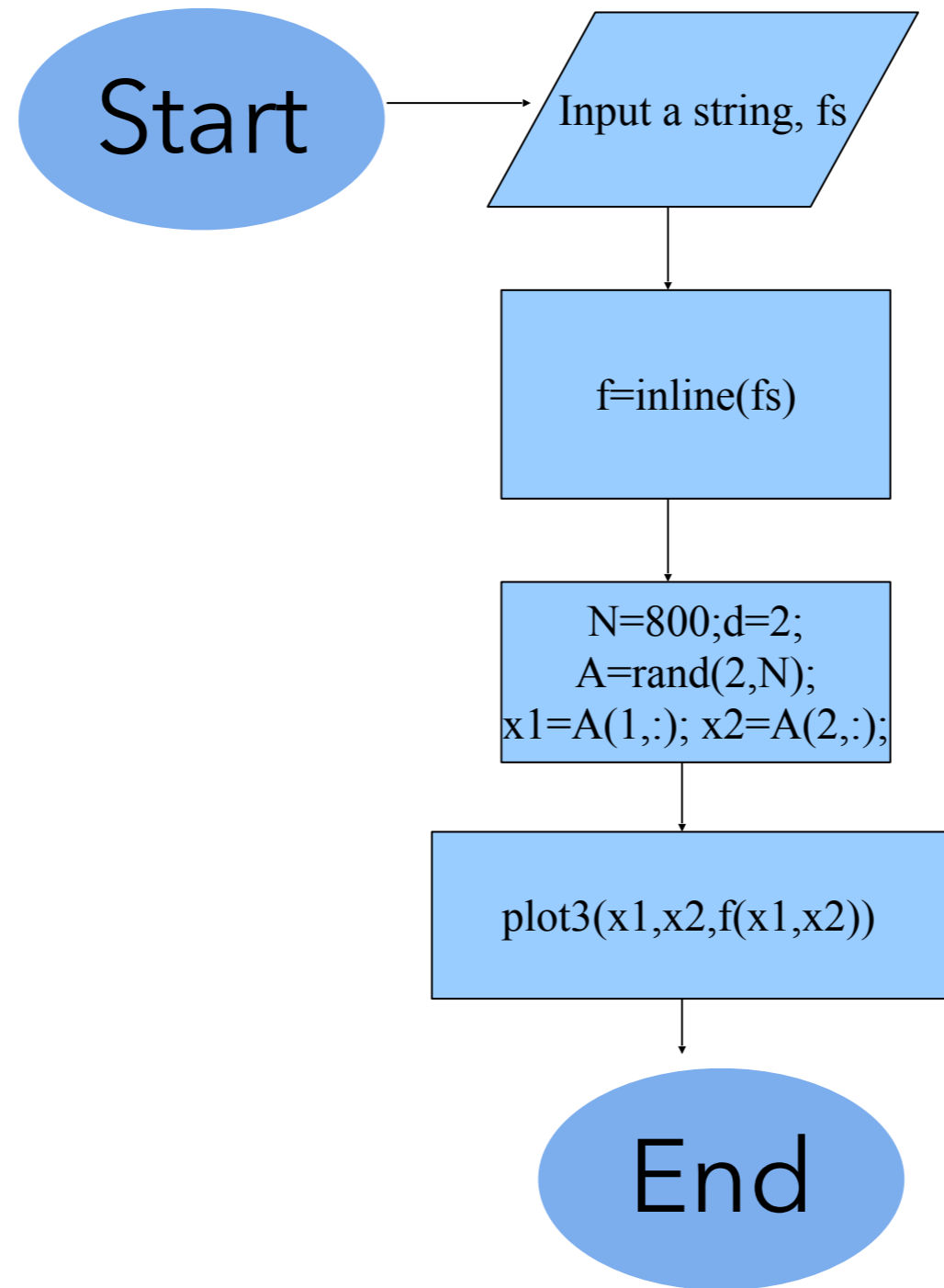
```
5
```



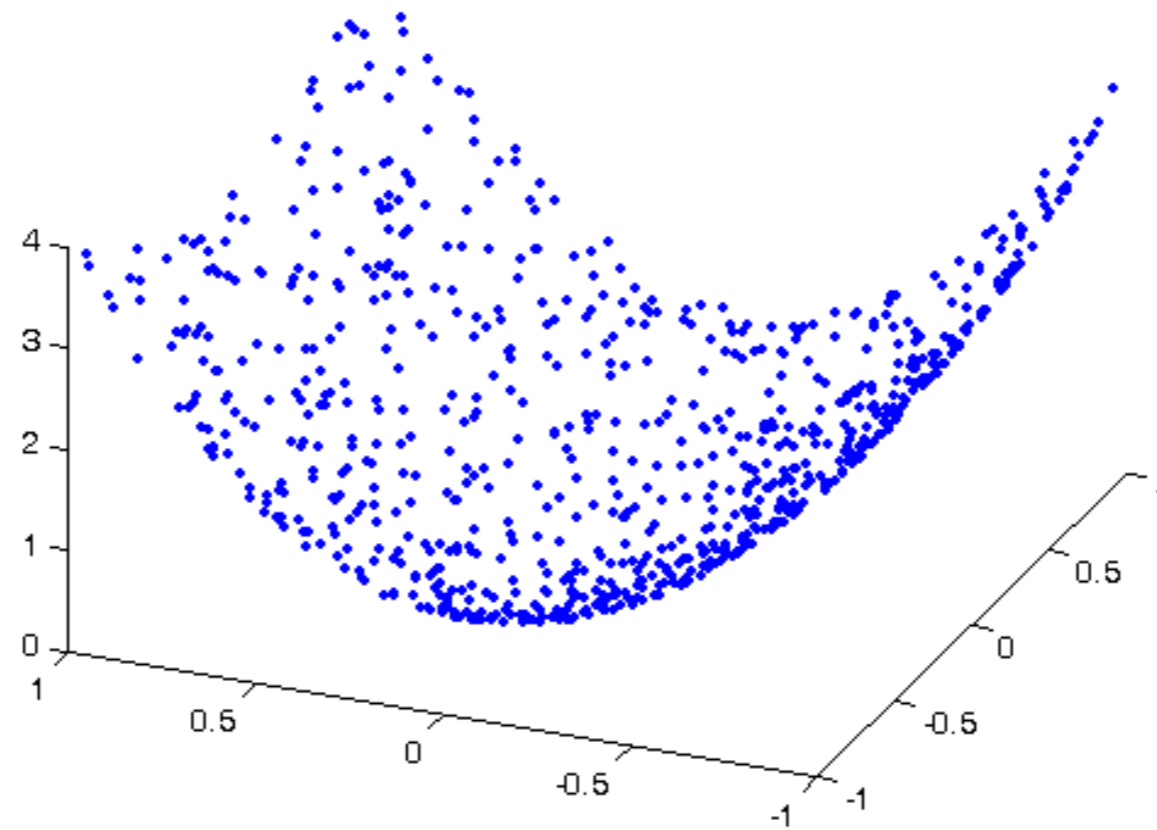
```
% Create an inline function  
fs= 'tanh(x)';  
f=inline(fs);  
range=5;  
x=-range:0.01:range;  
y=f(x);  
plot(x,y);
```

$\cos(x)$



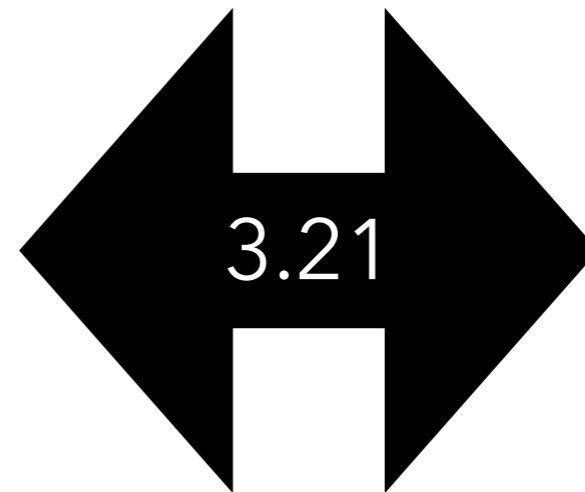


# A sample from function surface



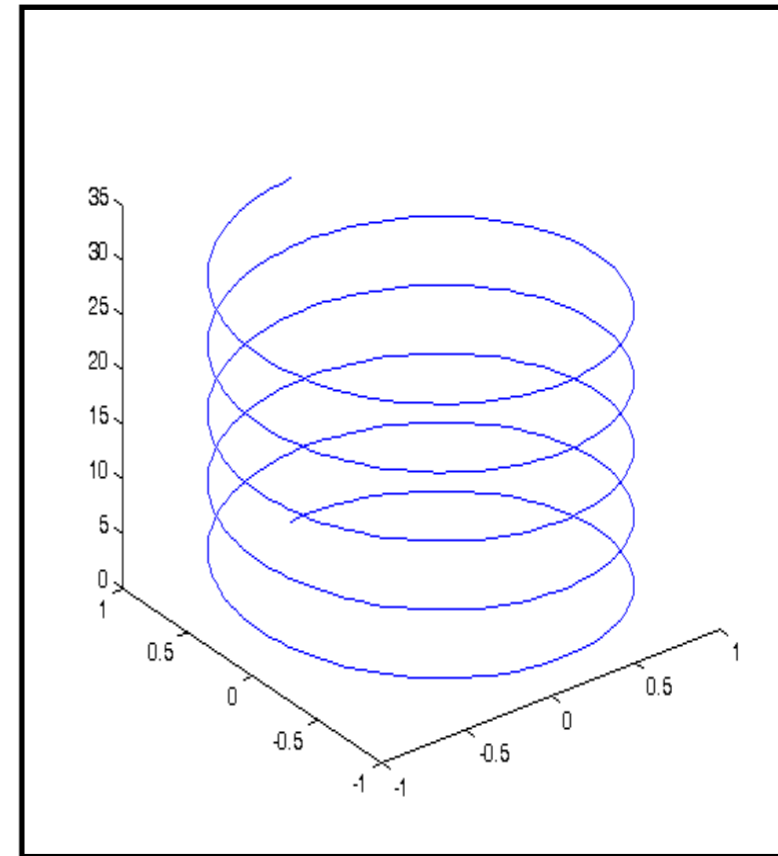
## Demo 2

```
% Inline function creation
% The created inline function has two input arguments
% Plot a sample from the function surface
fs='x.^2+3*y.^2';
fxy=inline(fs);
a=rand(2,800)*2-1;
x=a(1,:);y=a(2,:);
plot3(x,y,fxy(x,y),'.');
```



## 3d plot

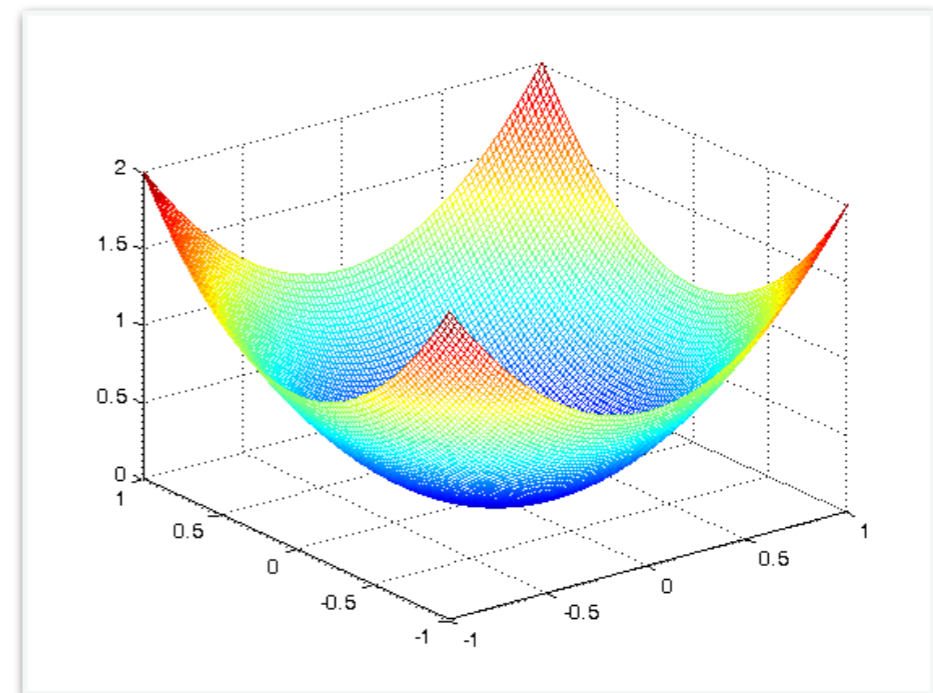
```
t = 0:pi/50:10*pi;  
plot3(sin(t),cos(t),t);
```



# mesh

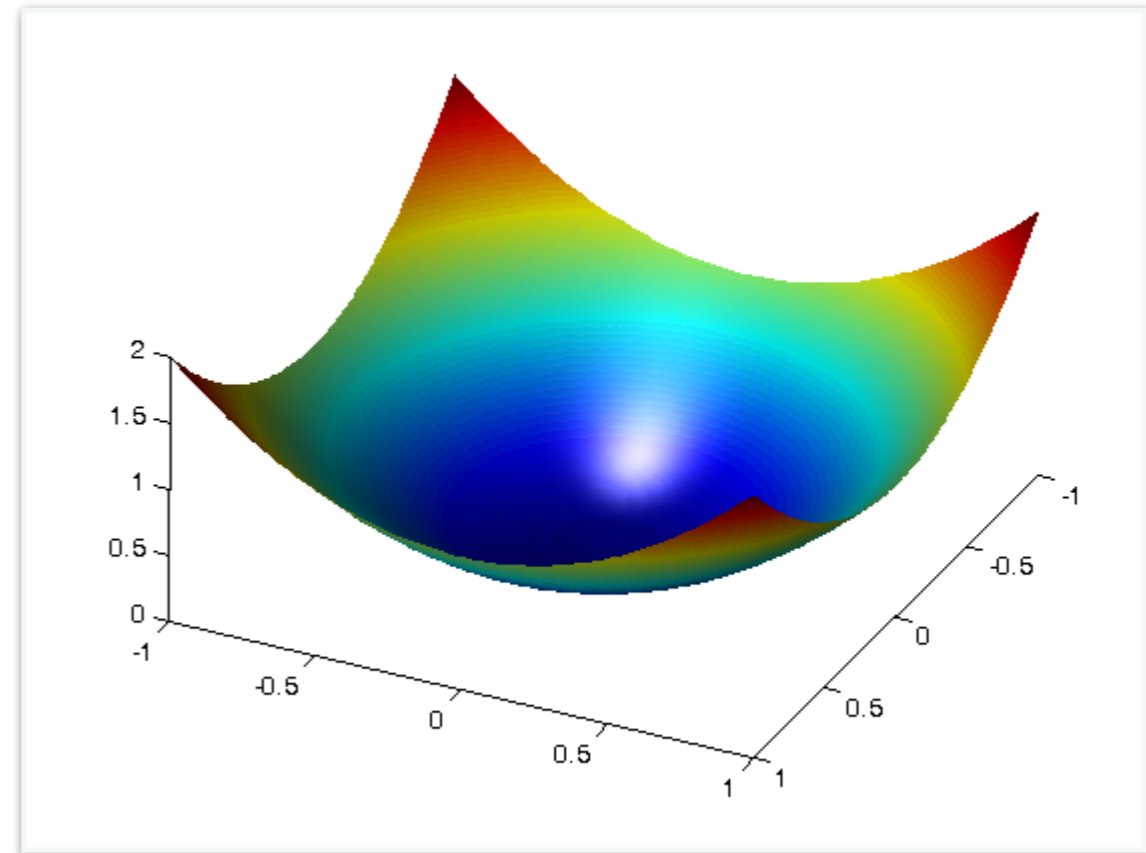
```
x=linspace(-1,1);  
y=linspace(-1,1)';  
X=repmat(x,100,1);  
Y=repmat(y,1,100);  
mesh(x,y,X.^2+Y.^2)
```

```
x=linspace(0,1);
```



# surface

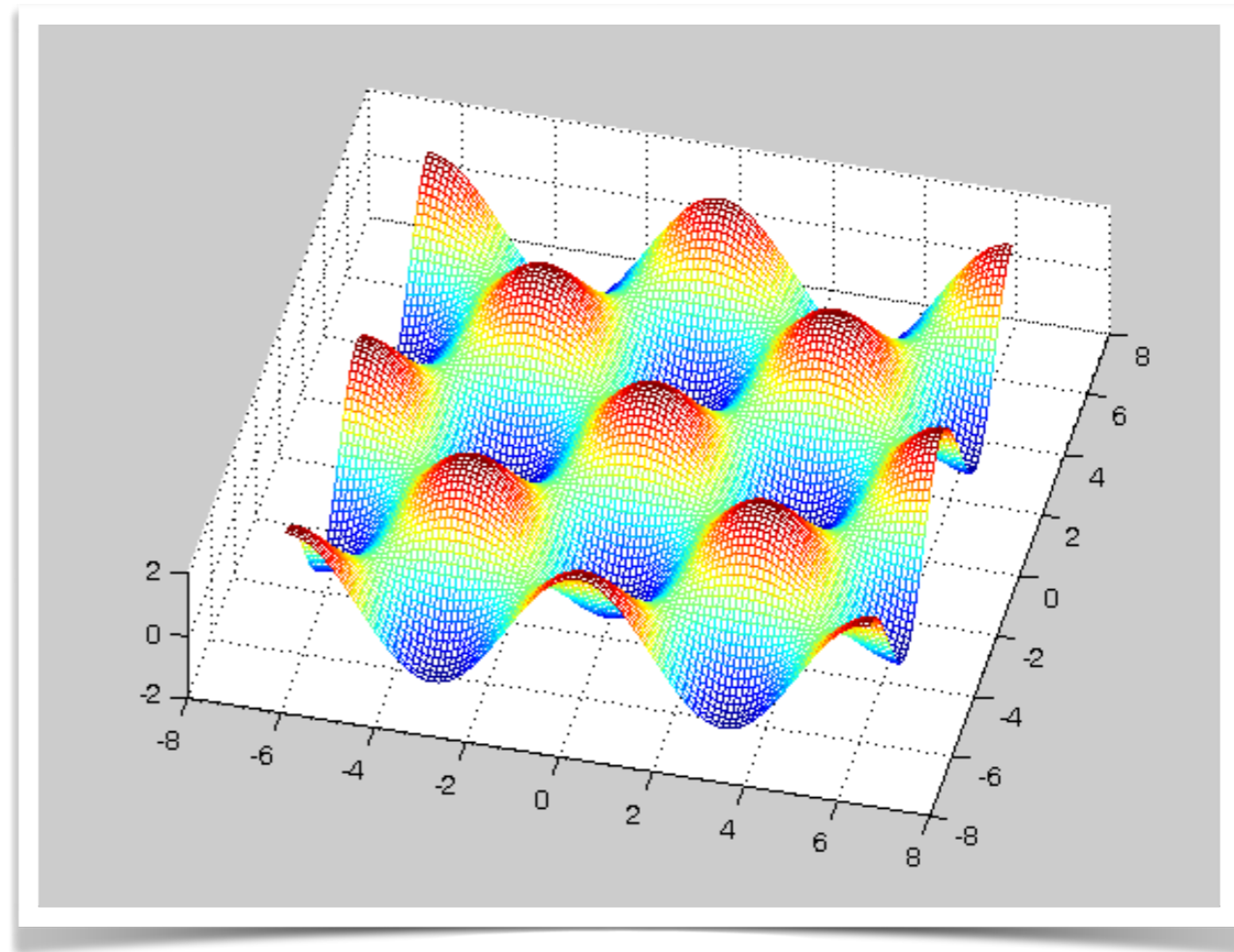
```
x=linspace(-1,1);  
y=linspace(-1,1)';  
X=repmat(x,100,1);  
Y=repmat(y,1,100);  
surface(x,y,X.^2+Y.^2)  
shading interp  
light  
lighting phong
```



# demo mesh

- demo\_mesh.m

Key in a 2D function:  $\cos(x+y)$



# Taylor series

$$f(x) \approx f(c) + f'(c)(x - c) + \frac{f''(c)}{2!}(x - c)^2 + \dots$$

$$= \sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!} (x - c)^k$$

if  $f^{(k)}(c)$  exists for  $k = 0, 1, \dots$

# Taylor's Theorem

$$f(x) \approx f(c) + f'(c)(x-c) + \frac{f''(c)}{2!}(x-c)^2 + \dots$$

$$= \sum_{k=0}^n \frac{f^{(k)}(c)}{k!}(x-c)^k + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-c)^{n+1}$$

if  $f^{(n+1)}$  exists at interval  $[a, b]$

$x, c \in [a, b]$

$\xi(x) \in (x, c)$

# Taylor's Theorem

- Approximate  $f(x)$  by first  $n+1$  terms of Taylor series

$$\sum_{k=0}^n \frac{f^{(k)}(c)}{k!} (x - c)^k$$

- Truncating error:  $O(h^{n+1})$ ,  $h = x - c$

$$\frac{f^{(n+1)}(\xi(x))}{(n+1)!} (x - c)^{n+1}$$

$h$  is a small value far less than one. The error decreases as  $n$  increases

# Taylor series expansion

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2) \quad \text{manual}$$

$$f'(x) = \frac{df(x)}{dx} = -2xf(x)$$

$$\begin{aligned} f''(x) &= -2f(x) - 2xf'(x) = -2f(x) + 4x^2f(x) \\ &= f(x)(4x^2 - 2) \quad f'(x)(4x^2 - 2) + 8xf(x) \end{aligned}$$

$$\begin{aligned} f'''(x) &= 4xf(x) + 8xf(x) - 8x^3f(x) \\ &= f(x)(12x - 8x^3) \quad -2xf(x)(4x^2 - 2) + 8xf(x) = f(x)(-8x^3 + 12x) \end{aligned}$$

# Taylor series expansion

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2) \quad \text{manual}$$

$$f'(x) = \frac{df(x)}{dx} = -2xf(x)$$

$$\begin{aligned} f''(x) &= -2f(x) - 2xf'(x) = -2f(x) + 4x^2f(x) \\ &= f(x)(4x^2 - 2) \quad f'(x)(4x^2 - 2) + 8xf(x) \end{aligned}$$

$$\begin{aligned} f'''(x) &= 4xf(x) + 8xf(x) - 8x^3f(x) \\ &= f(x)(12x - 8x^3) \quad -2xf(x)(4x^2 - 2) + 8xf(x) = f(x)(-8x^3 + 12x) \end{aligned}$$

$$c=1, k=0\dots 3$$

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2)$$

$$\begin{aligned} f(x) &\approx \sum_{k=0}^3 \frac{f^{(k)}(c)}{k!} (x-c)^k \\ &= f(c) + f'(c)(x-c) + \frac{f''(c)}{2!} (x-c)^2 + \frac{f'''(c)}{3!} (x-c)^3 \end{aligned}$$

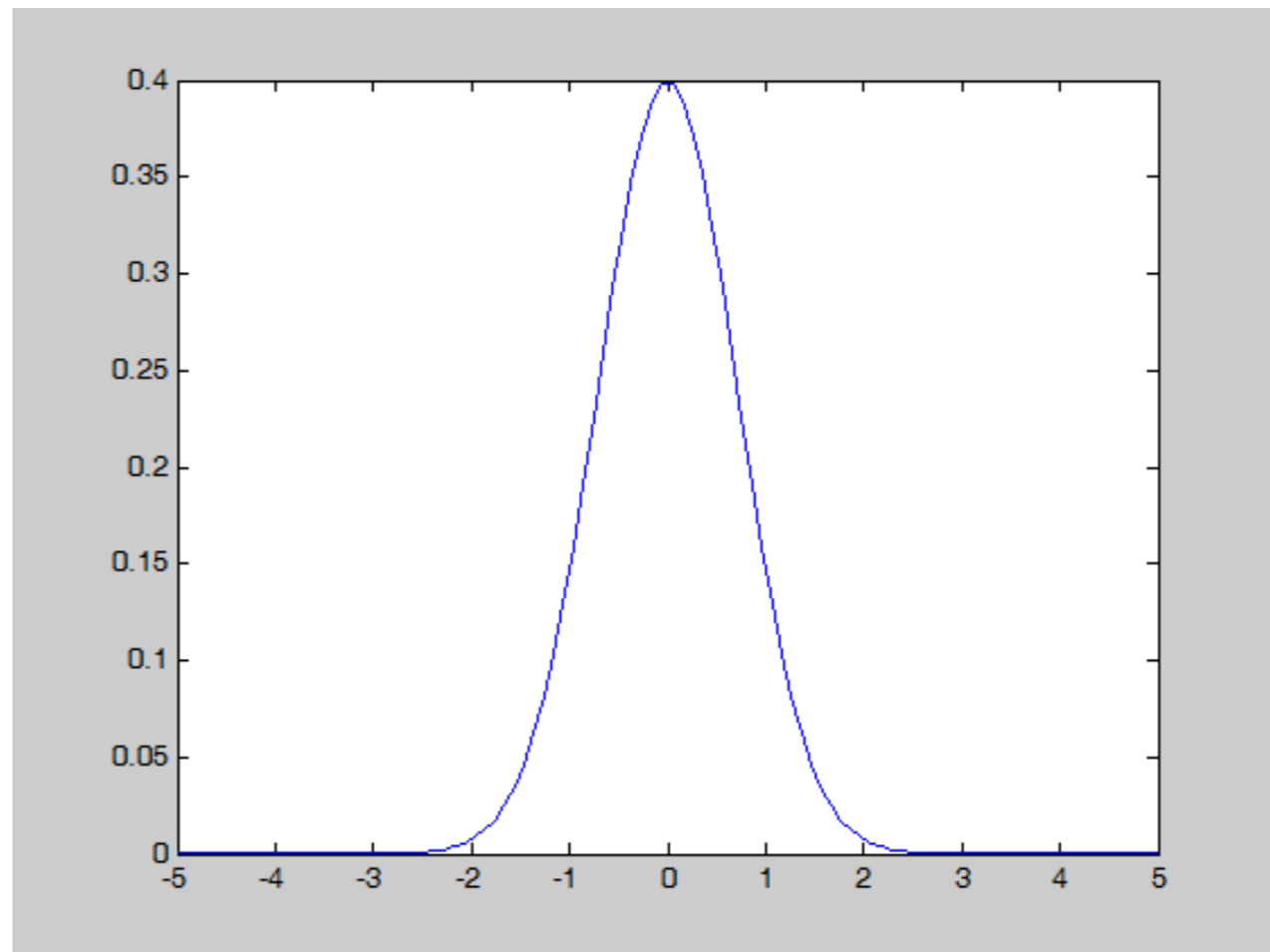
# Normal function

## source code

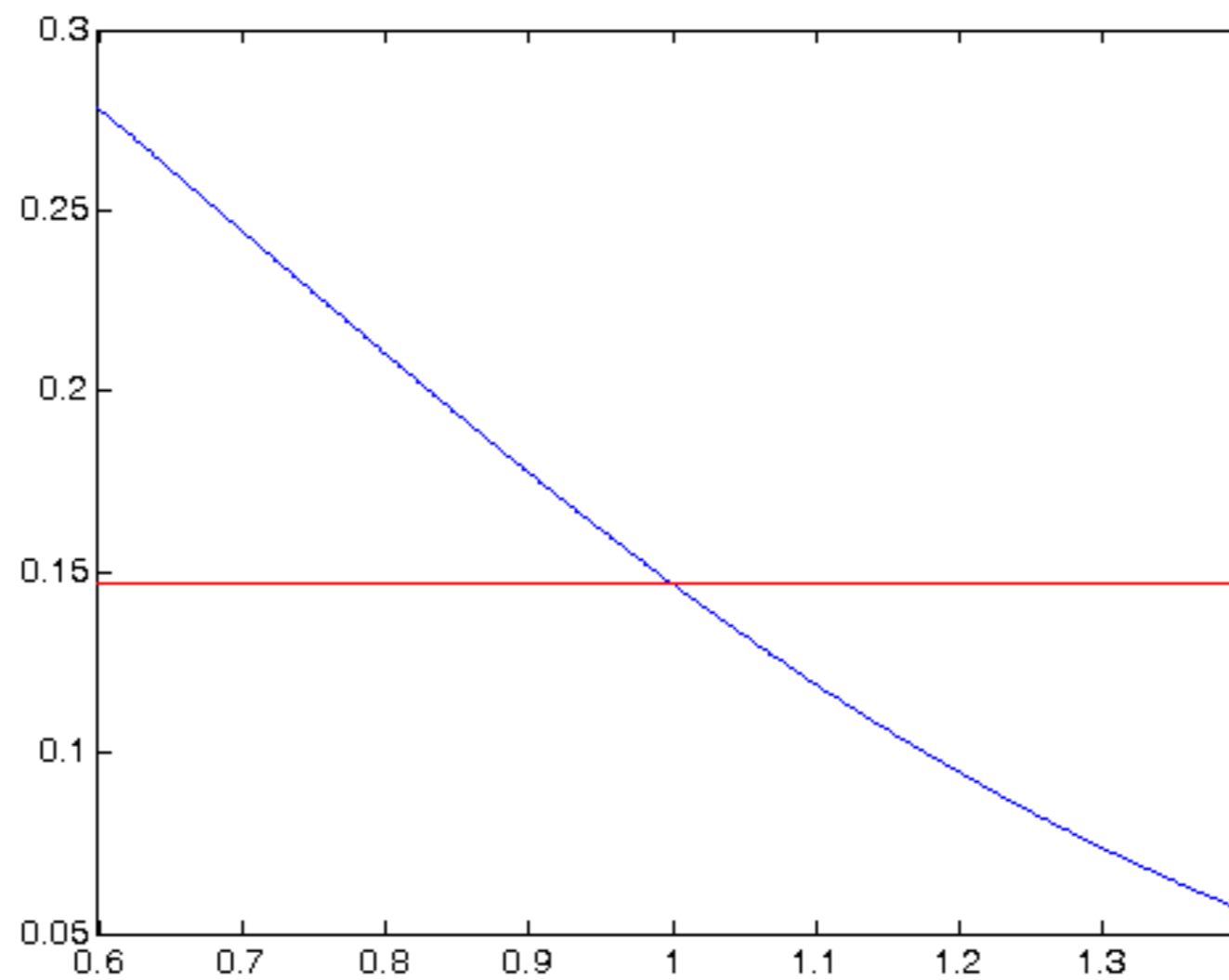
```
x=linspace(-5,5);
```

```
>> plot(x,f_normal(x))
```

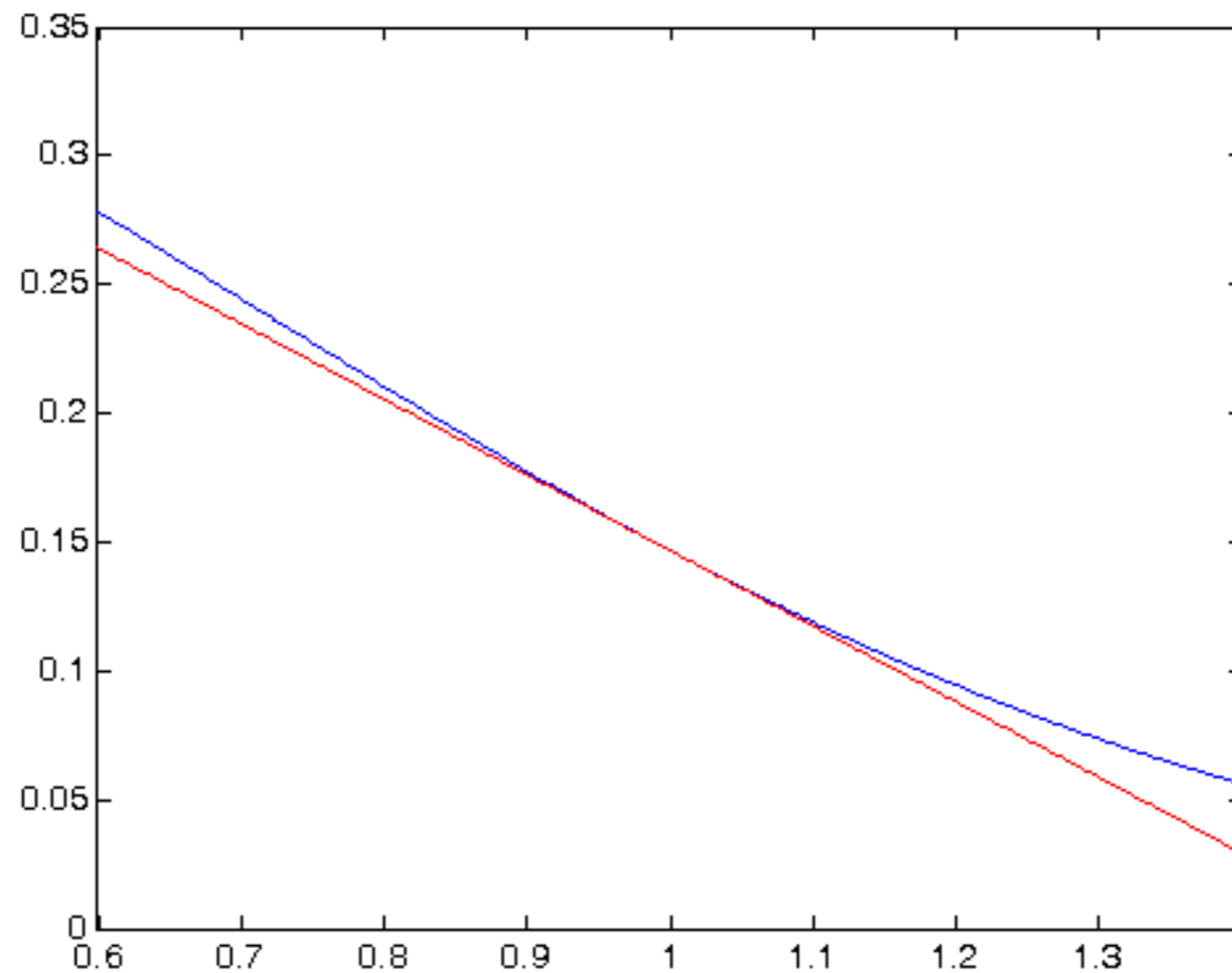
```
function y=f_normal(x)  
    y=1/sqrt(2*pi)*exp(-x.^2);  
return
```



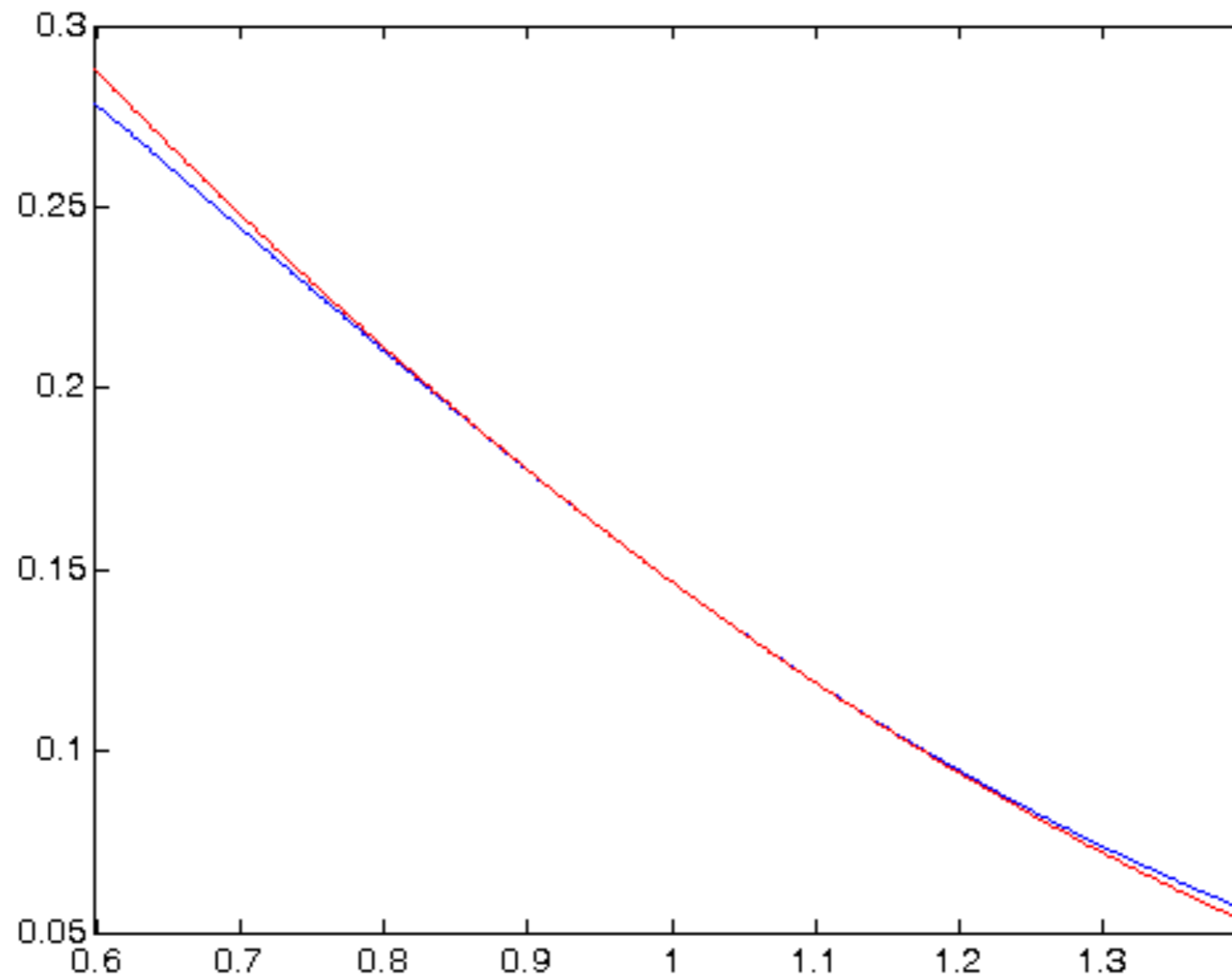
# Taylor expansion at $c=1$ with $k=0$



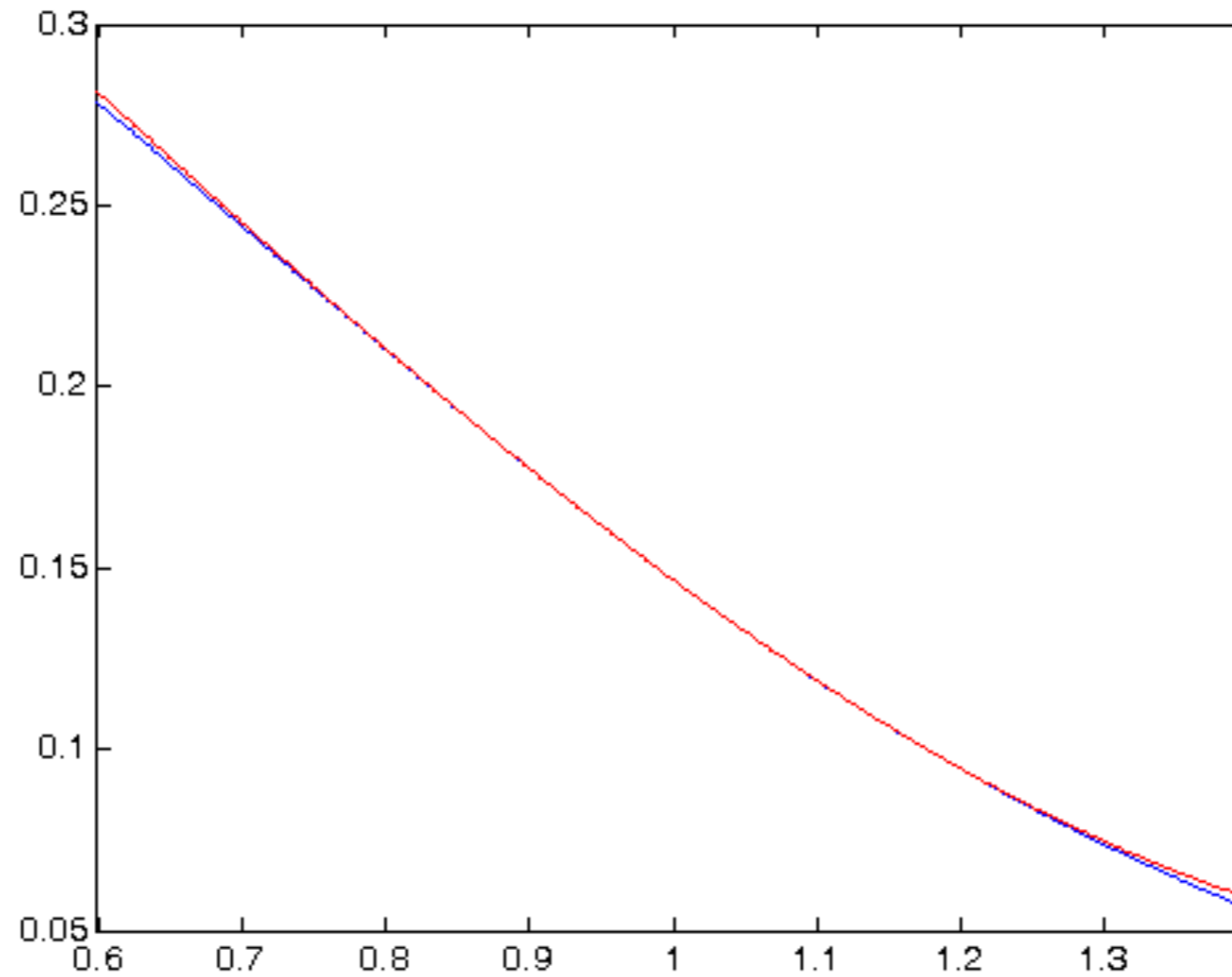
# Taylor expansion at $c=1$ with $k \leq 1$



# Taylor expansion at $c=1$ with $k \leq 2$



# Taylor expansion at $c=1$ with $k \leq 3$



# Taylor expansion

## Source code Taylor nor

```
% Taylor expansion
% f(x)=1/sqrt(2pi)*exp(-x^2);
x=linspace(0.6,1.4,500);
n=length(x);
c=1;
for i=1:n
    y1(i)=f_nm(c);
    y2(i)=-2*c*f_nm(c)*(x(i)-c);
    y3(i)=(4*c^2-2)*f_nm(c)/2*(x(i)-c)^2;
    y4(i)=(12*c-8*c^3)*f_nm(c)/6*(x(i)-c)^3;
end
plot(x,f_nm(x));hold on;plot(x,y1,'r'); title('one term');
figure; plot(x,f_nm(x));hold on;plot(x,y1+y2,'r');title('two terms');
figure; plot(x,f_nm(x));hold on;plot(x,y1+y2+y3,'r');title('three terms');
figure; plot(x,f_nm(x));hold on;plot(x,y1+y2+y3+y4,'r');title('four terms');
```

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2) \quad \text{manual}$$

$$f'(x) = \frac{df(x)}{dx} = -2xf(x)$$

$$\begin{aligned} f''(x) &= -2f(x) - 2xf'(x) = -2f(x) + 4x^2 f(x) \\ &= f(x)(4x^2 - 2) \quad f'(x)(4x^2 - 2) + 8xf(x) \end{aligned}$$

$$\begin{aligned} f'''(x) &= 4xf(x) + 8xf(x) - 8x^3 f(x) \\ &= f(x)(12x - 8x^3) \quad -2xf(x)(4x^2 - 2) + 8xf(x) = f(x)(-8x^3 + 12x) \end{aligned}$$

**differentiation**

```
>> x=sym('x');
```

```
>> diff(x.^2)
```

```
ans =
```

```
2*x
```

```
x=sym('x')
ss='x.^2';
inst=['diff(' ss ')']
eval(inst)
```

ans =

$2*x$

```
x=sym('x')
ss='x.^2';
inst= "diff("+ss+)"
eval(inst)
```

ans =

$2*x$

# First derivative

```
% input a string to specify a function
% find its derivative
ss='tanh(x)'
fx=inline(ss);
ss="diff("+ss+" )";
ss1=eval(ss);
fx1=inline(ss1)
```

## 2<sup>nd</sup> derivative

```
x= sym('x')
f=inline(tanh(x));
s1= diff(tanh(x))
f1 = inline(s1)
s2=diff(s1)
f2 = inline(s2)
s3=diff(s2)
f3 = inline(s3)
```

# Example

```
>> f_diff3  
function of x:tanh(x)
```

```
fx1 =
```

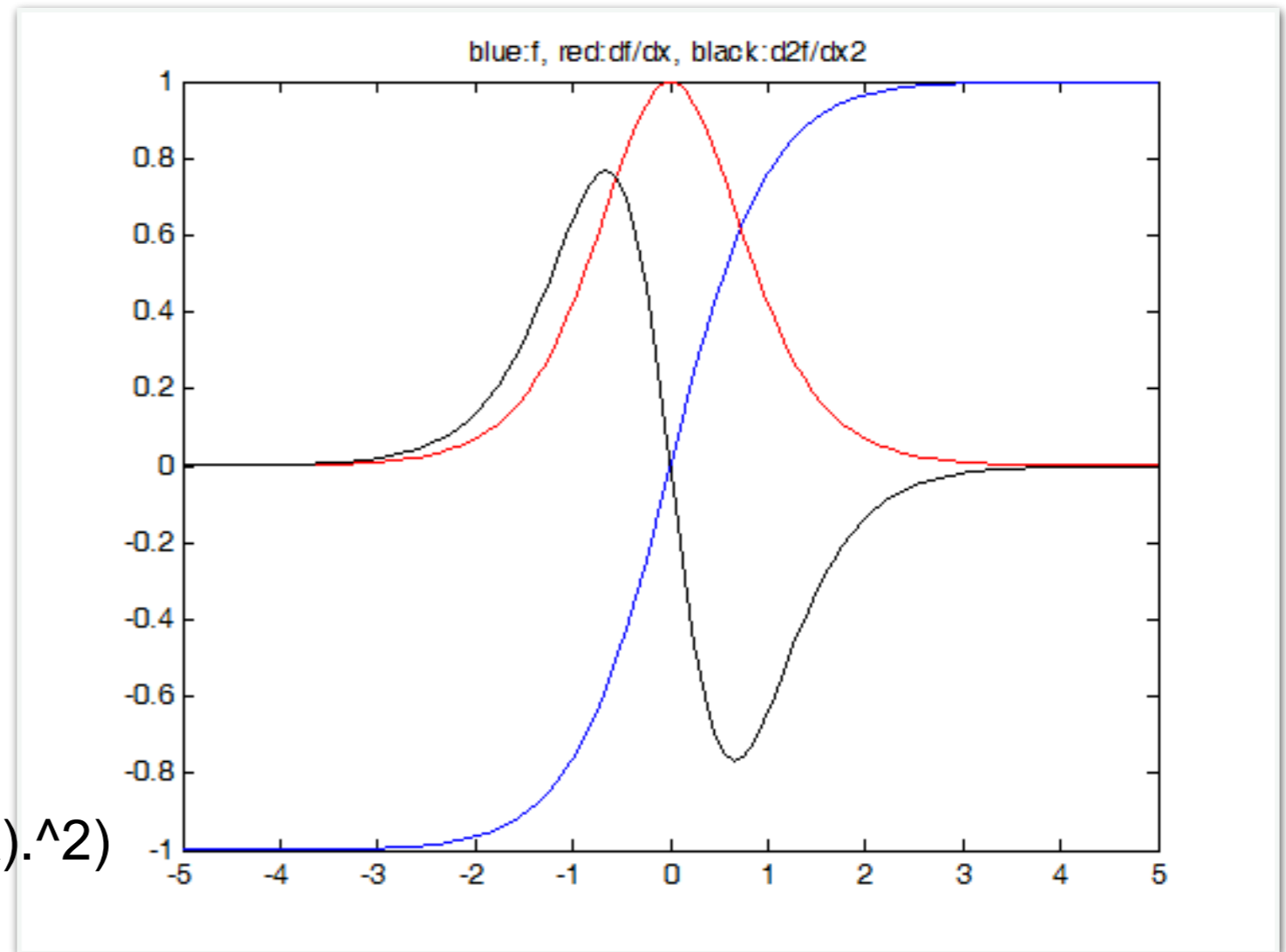
```
Inline function:
```

```
fx1(x) = 1-tanh(x).^2
```

```
fx2 =
```

```
Inline function:
```

```
fx2(x) = -2.*tanh(x).*(1-tanh(x).^2)
```



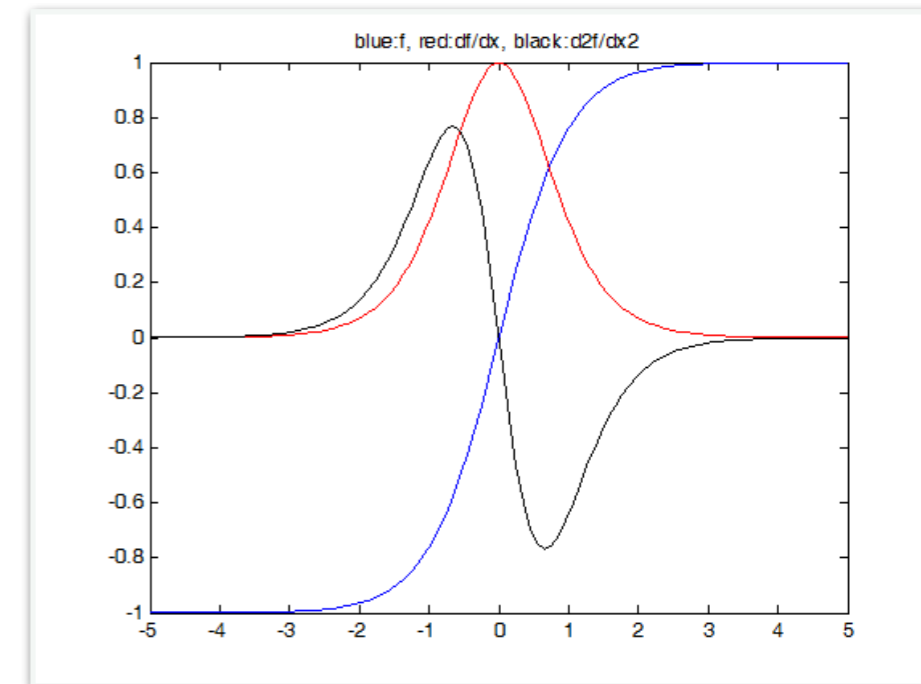
$$f(x) = \sum_{k=0}^3 \frac{f^{(k)}(c)}{k!} (x-c)^k$$

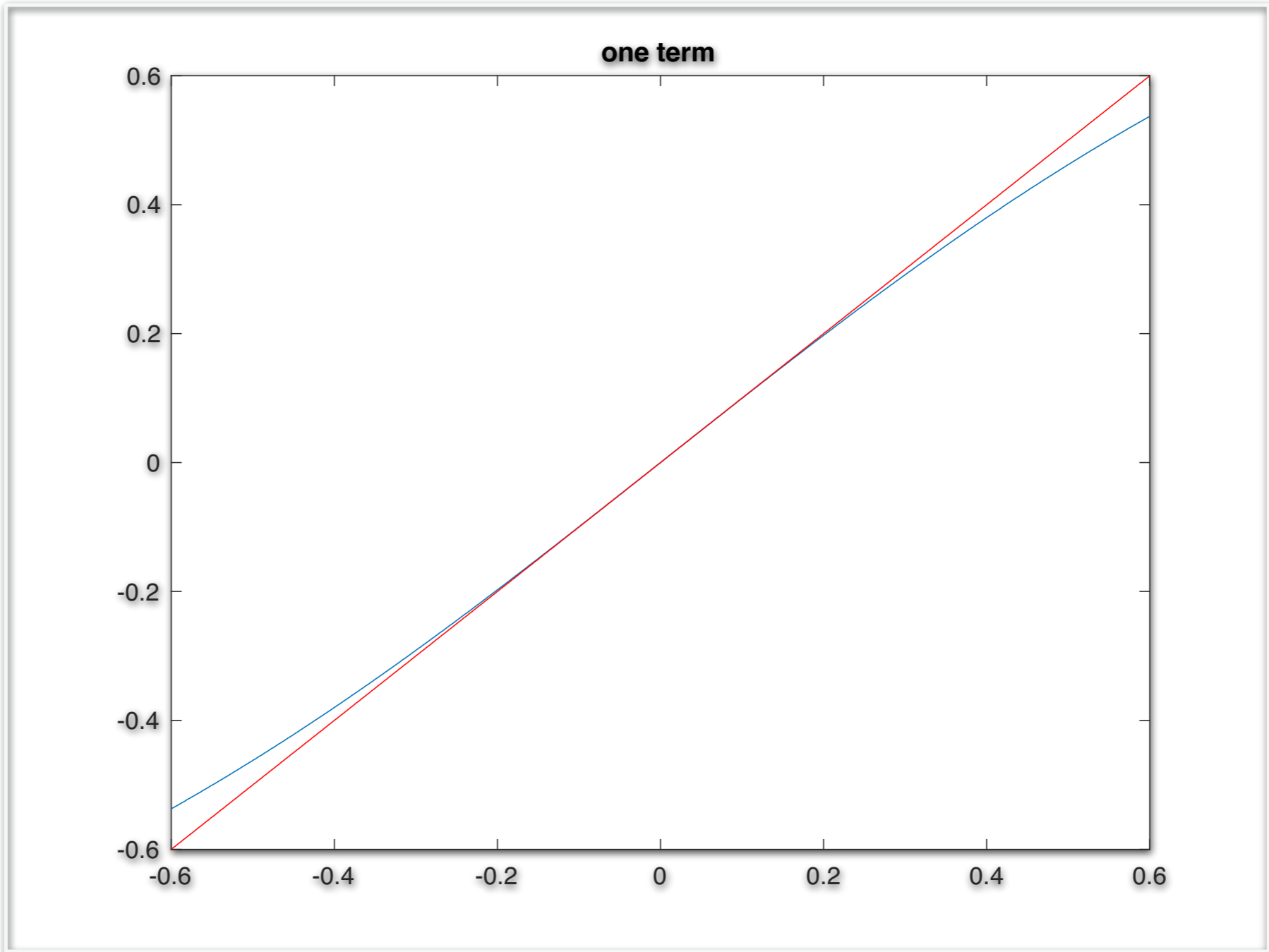
$$= f(c) + f'(c)(x-c) + \frac{f''(c)}{2!} (x-c)^2 + \frac{f'''(c)}{3!} (x-c)^3$$

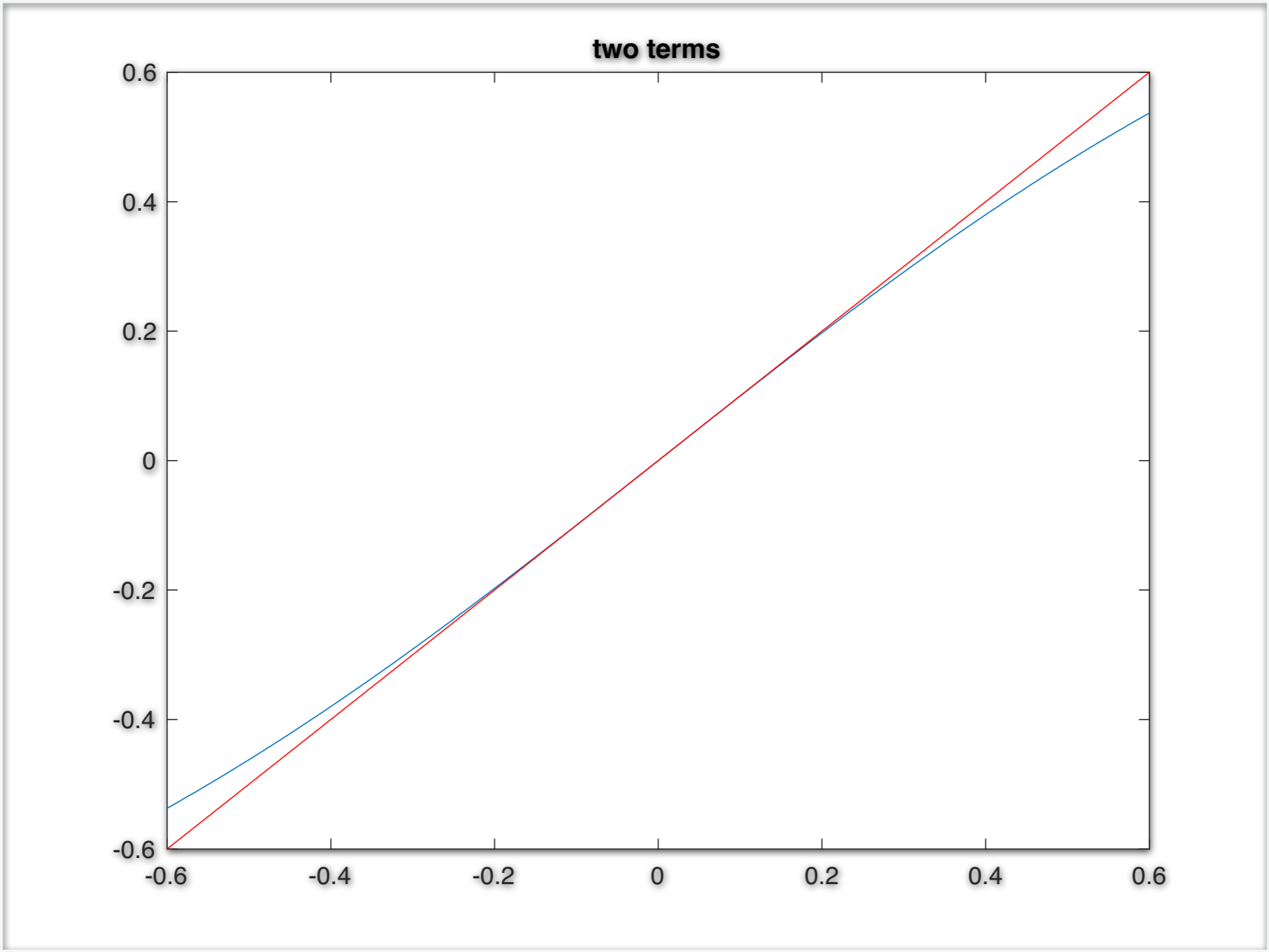
```

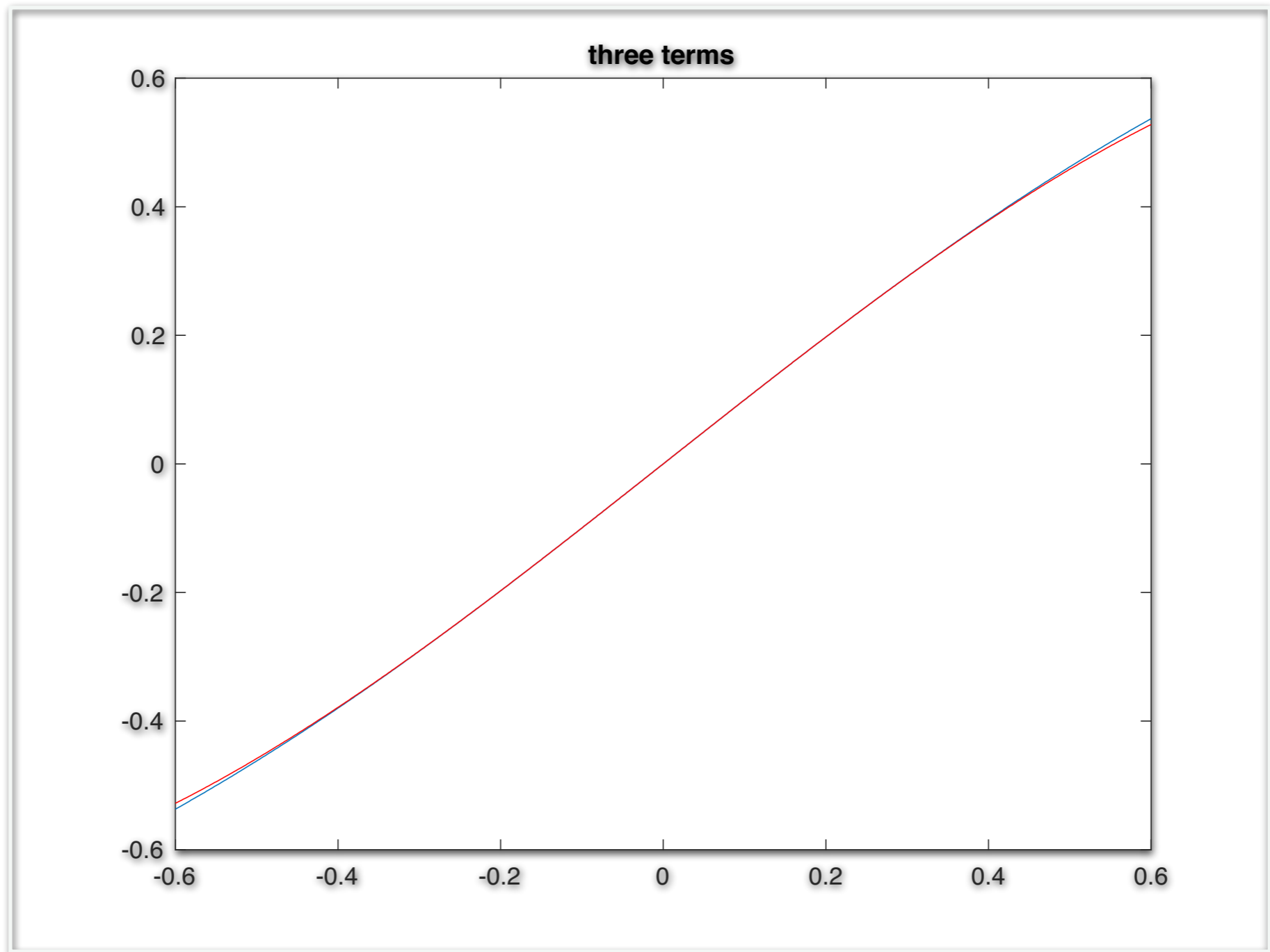
x=linspace(-0.6,0.6,500);
n=length(x);
c=0;
for i=1:n
    y(i)=f(c);
    y1(i)=f1(c)*(x(i)-c);
    y2(i)=f2(c)/2*(x(i)-c)^2;
    y3(i)=f3(c)/6*(x(i)-c)^3;
end
plot(x,f(x));hold on;
plot(x,y+y1,'r'); title('two term');
figure;
plot(x,f(x));hold on;
plot(x,y+y1+y2,'b');title('three terms');
figure; plot(x,f(x));hold on;
plot(x,y+y1+y2+y3,'g');title('four terms');

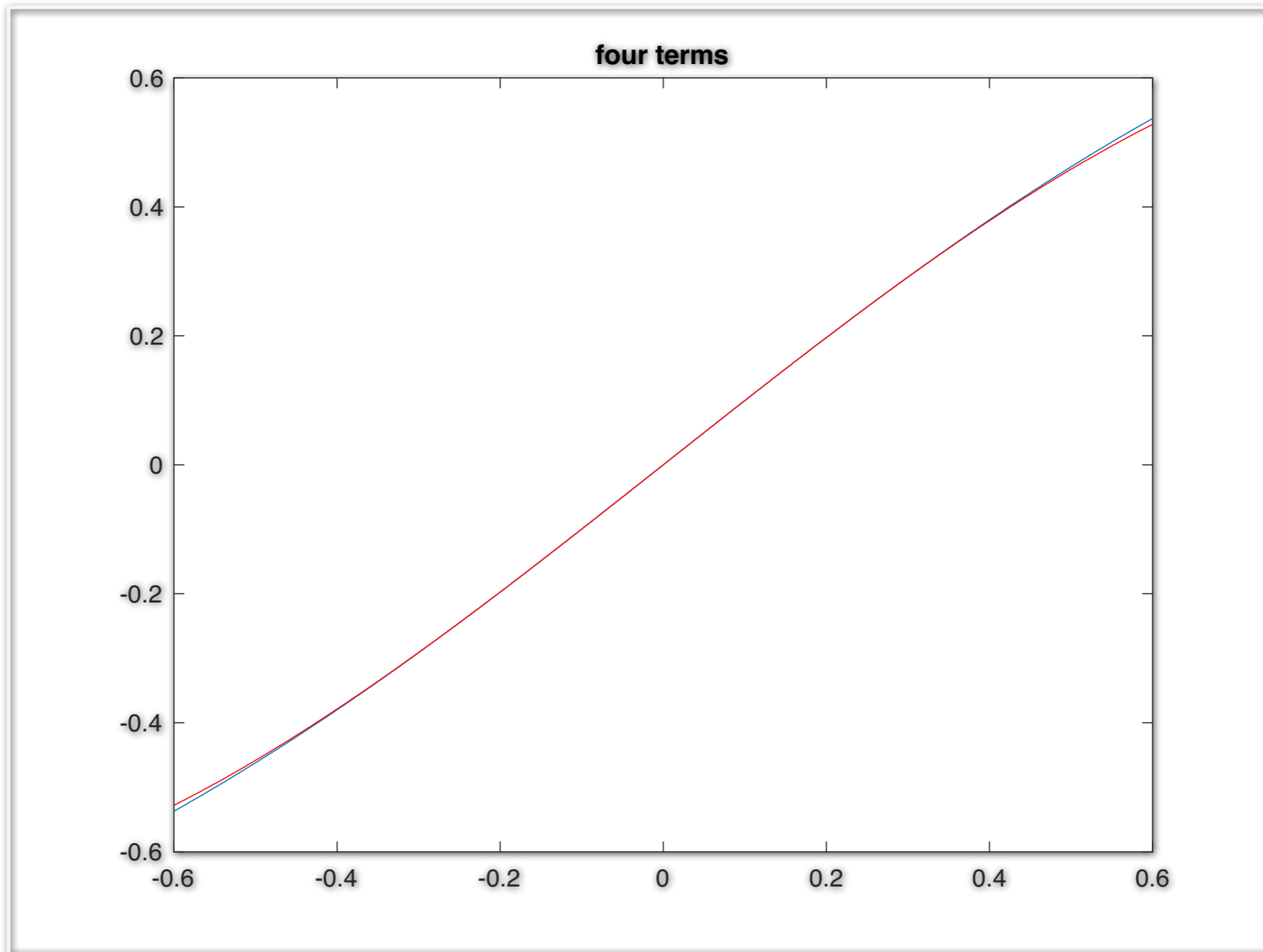
```











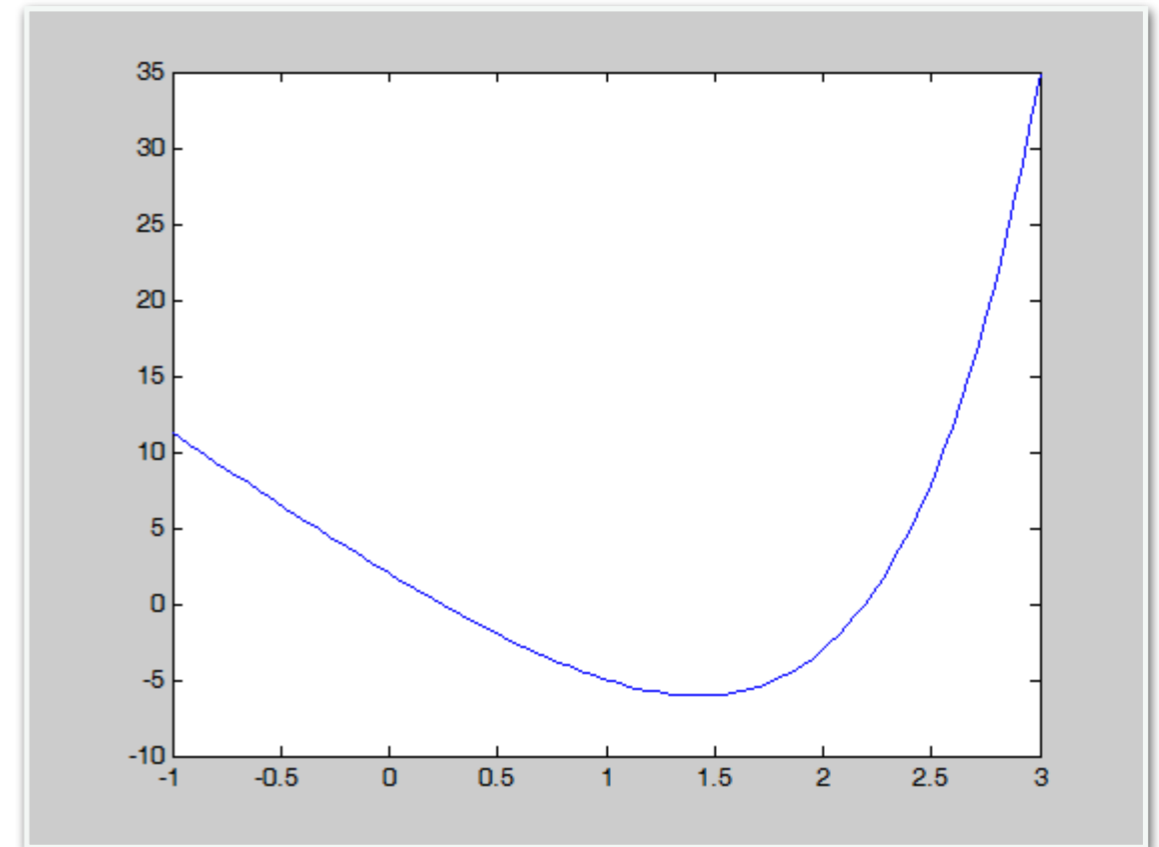
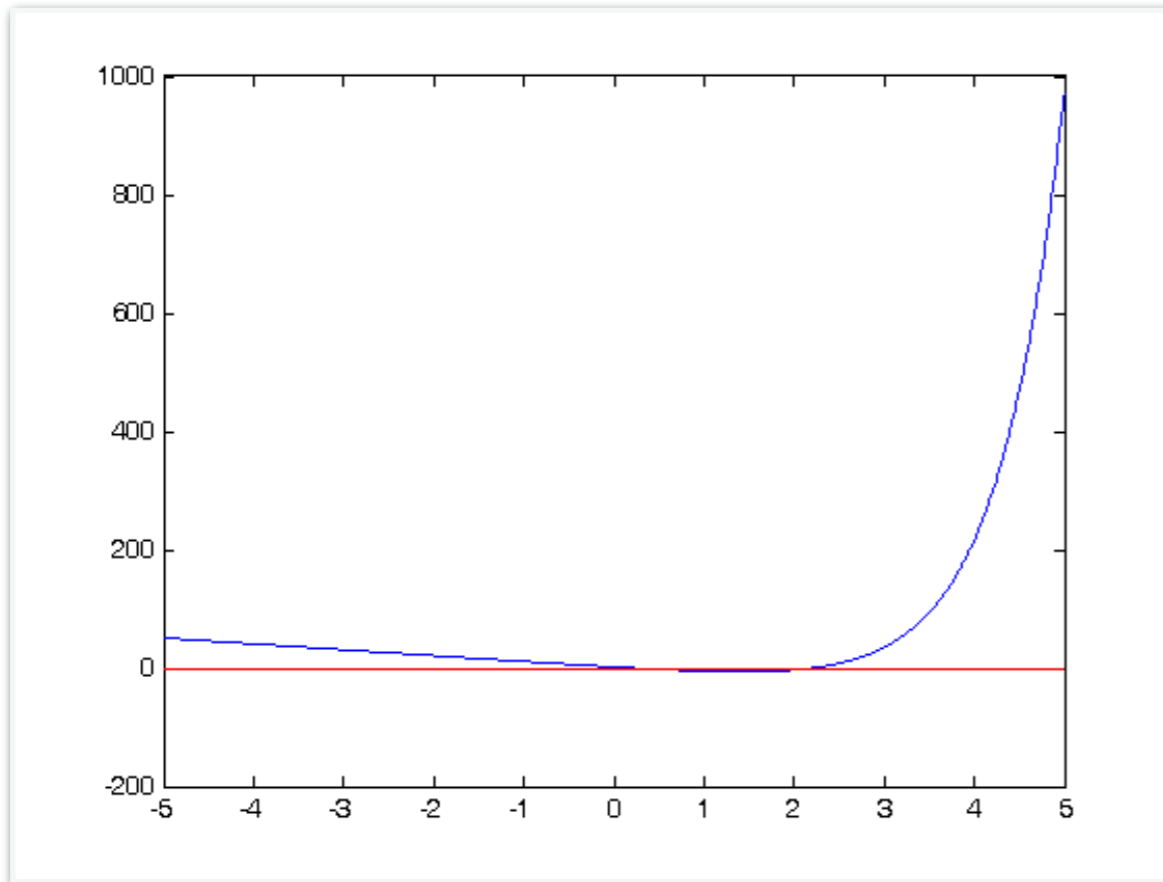
## Problem: **Root finding**

- Find  $x$  such  $f(x)=0$  for a given  $f$
- $f$  denotes an arbitrary differentiable function

$$f(x) = 2^{x^2} - 10x + 1$$

```
>> x=linspace(-1,3);
```

```
>> plot(x,2.^x.^2-10*x+1)
```



## Root finding

- Input: a string that specifies a function
- Plot the given function
- Find at least a root

# Tangent line

1

$$\text{slope} = f'(x_n) = \frac{y_n}{x_n - x_{n+1}}$$

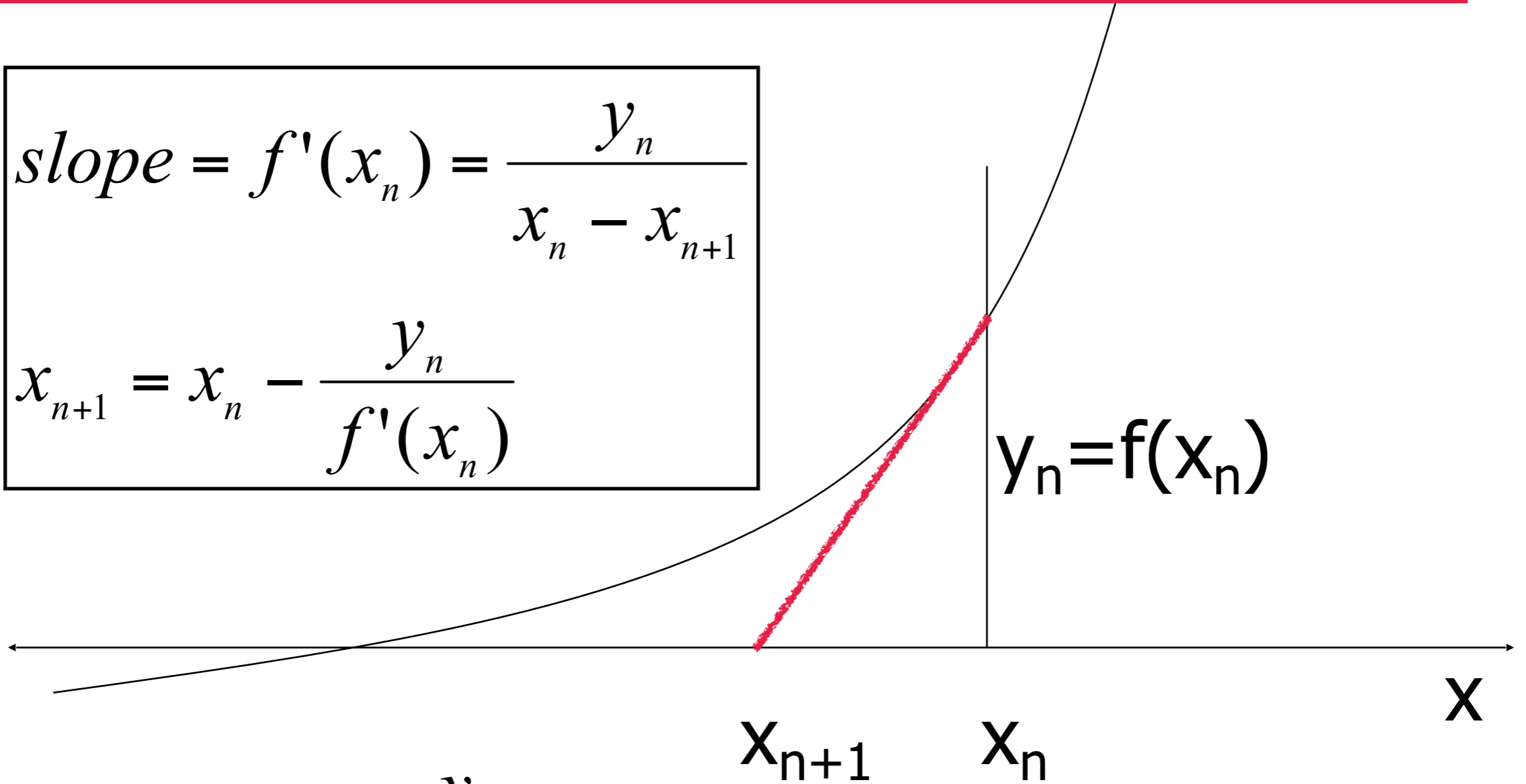
3

$$x_{n+1} = x_n - \frac{y_n}{f'(x_n)}$$

$$y_n = f(x_n)$$

2

$$x_n - x_{n+1} = \frac{y_n}{f'(x_n)}$$



## Iterative approach

- Start at a random guess  $x_0$
- Refine current guess by
  - Find the tangent line that passes  $(x_n, f(x_n))$
  - Set  $x_{n+1}$  to the intersection of the tangent line to horizontal axis

$$x_{n+1} = x_n - \frac{y_n}{f'(x_n)}$$

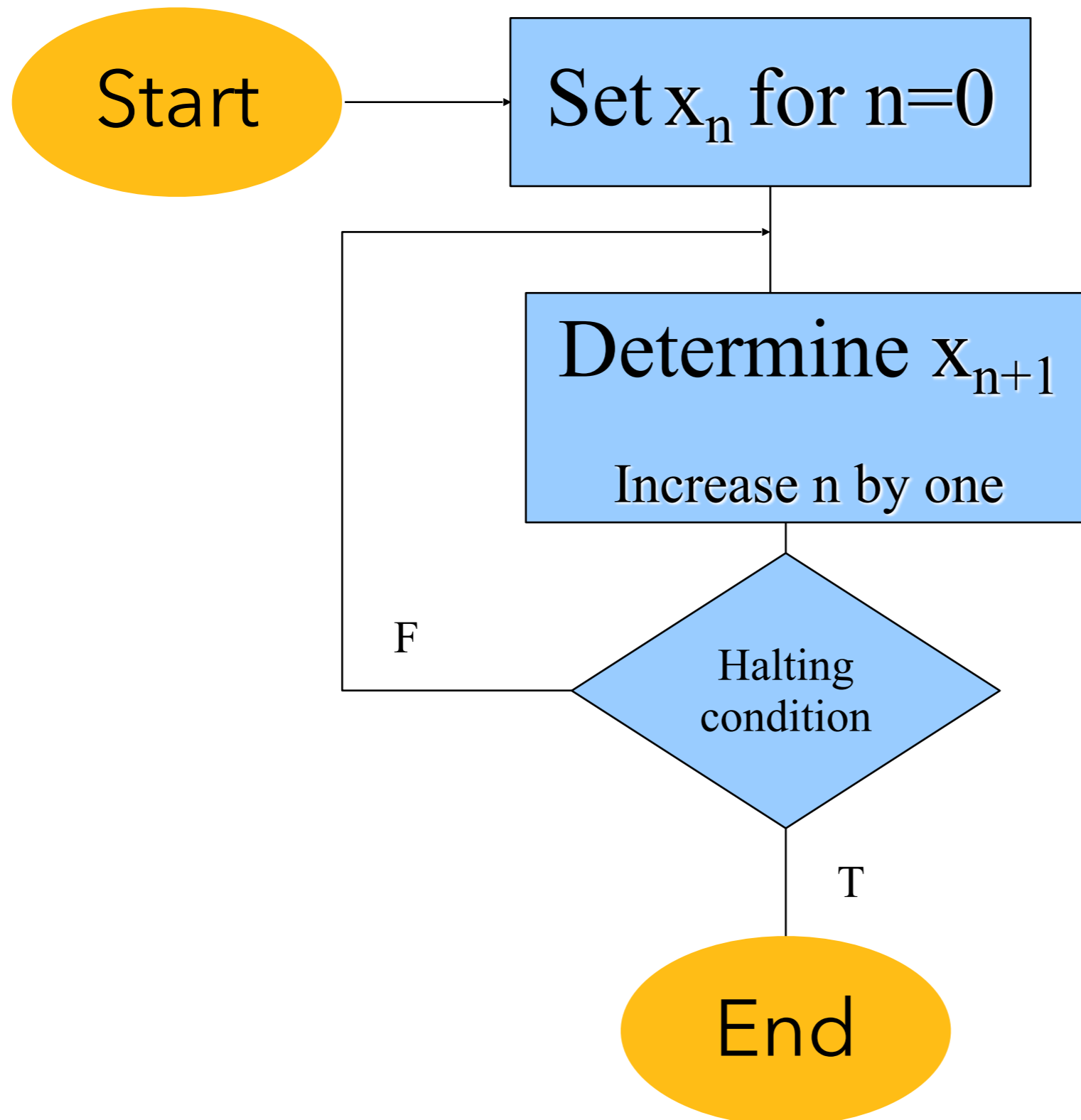
## Updating rule

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

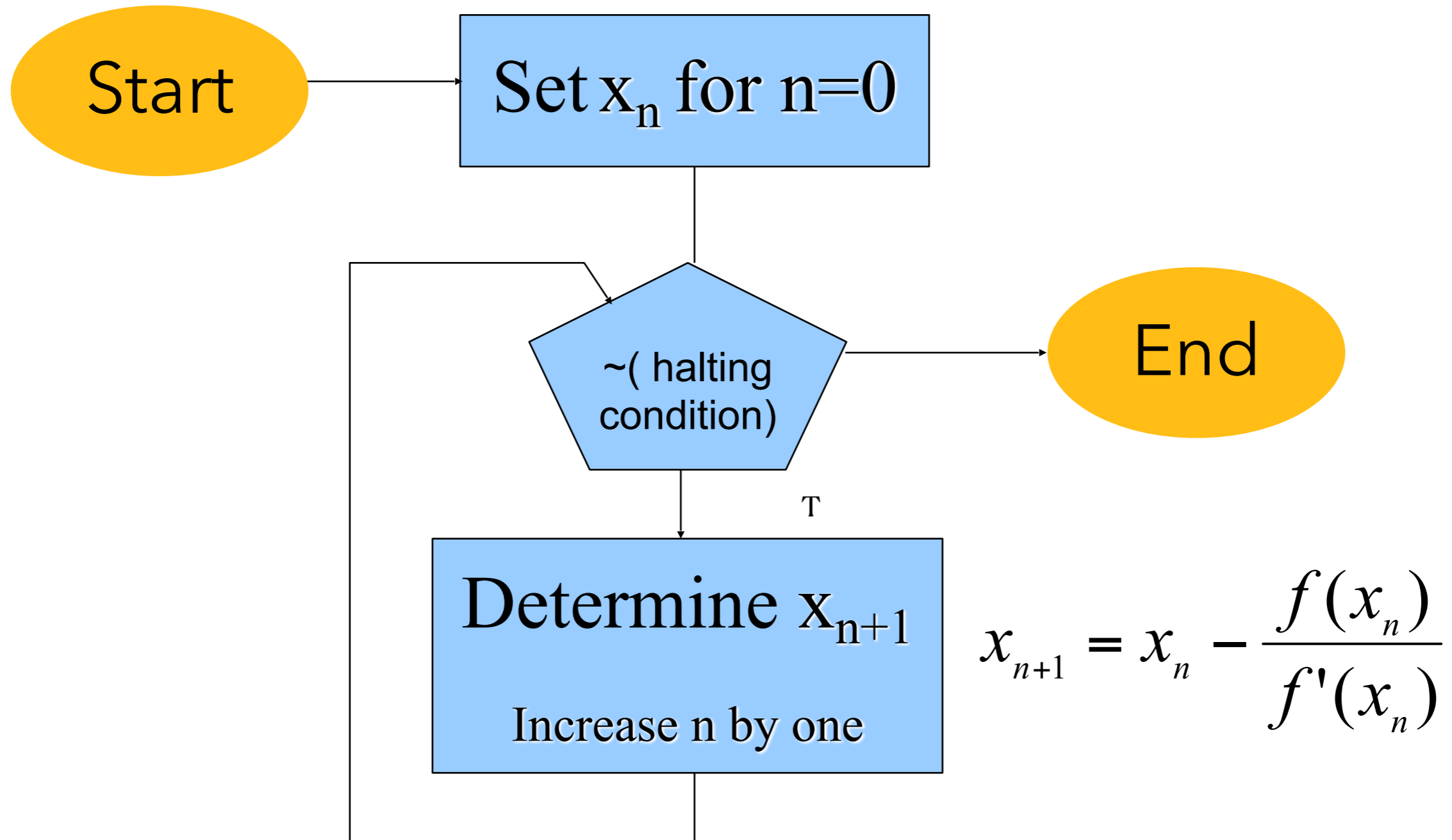
# Newton method

- Iterative approach
  1. Random guess
  2. Refine current guess according to an updating rule
  3. If halting condition does not hold, go to step 2 otherwise exit

# Iterative approach



# Iterative approach



# Iterative approach

Start

```
s='x.^2-5*x+6'  
f=inline(s); x=sym('x')  
ss=['diff(' s ')'];  
s1=eval(ss);  
f1=inline(s1); x_zero=rand;
```

$\sim(\text{abs}(f(x\_zero)) < 10^{-6})$

End

```
x_zero=x_zero-f(x_zero)/f1(x_zero)
```

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

## Halting condition

- Let  $x_n$  denote the guess at iteration  $n$
- The absolute value of  $f(x_n)$  is expected close enough to zero for some  $n$
- Halting Condition  
 $\text{abs}(f(x_n)) < \text{epsilon}$ 
  - epsilon denotes a predefined positive small number

# Initialization

- Random initialization
- Input from users

## Demo\_newton

source code demo\_newton

```
fstr=input('input a function:', 's');  
x_ini=input('guess its zero:');  
x_zero=newton(fstr,x_ini);
```

newton.m

## Main Program

```
1 fstr=input('input a function:', 's');
2 x_ini=input('guess its zero:');
3 range=3;
4 x_zero=newton(fstr,x_ini);
5 fx=inline(fstr);x=linspace(-range,range);plot(x,fx(x));hold on;
6 plot(x_zero,fx(x_zero),'ro');
7 plot([-range range],[0 0],'r');
```

## Newton method

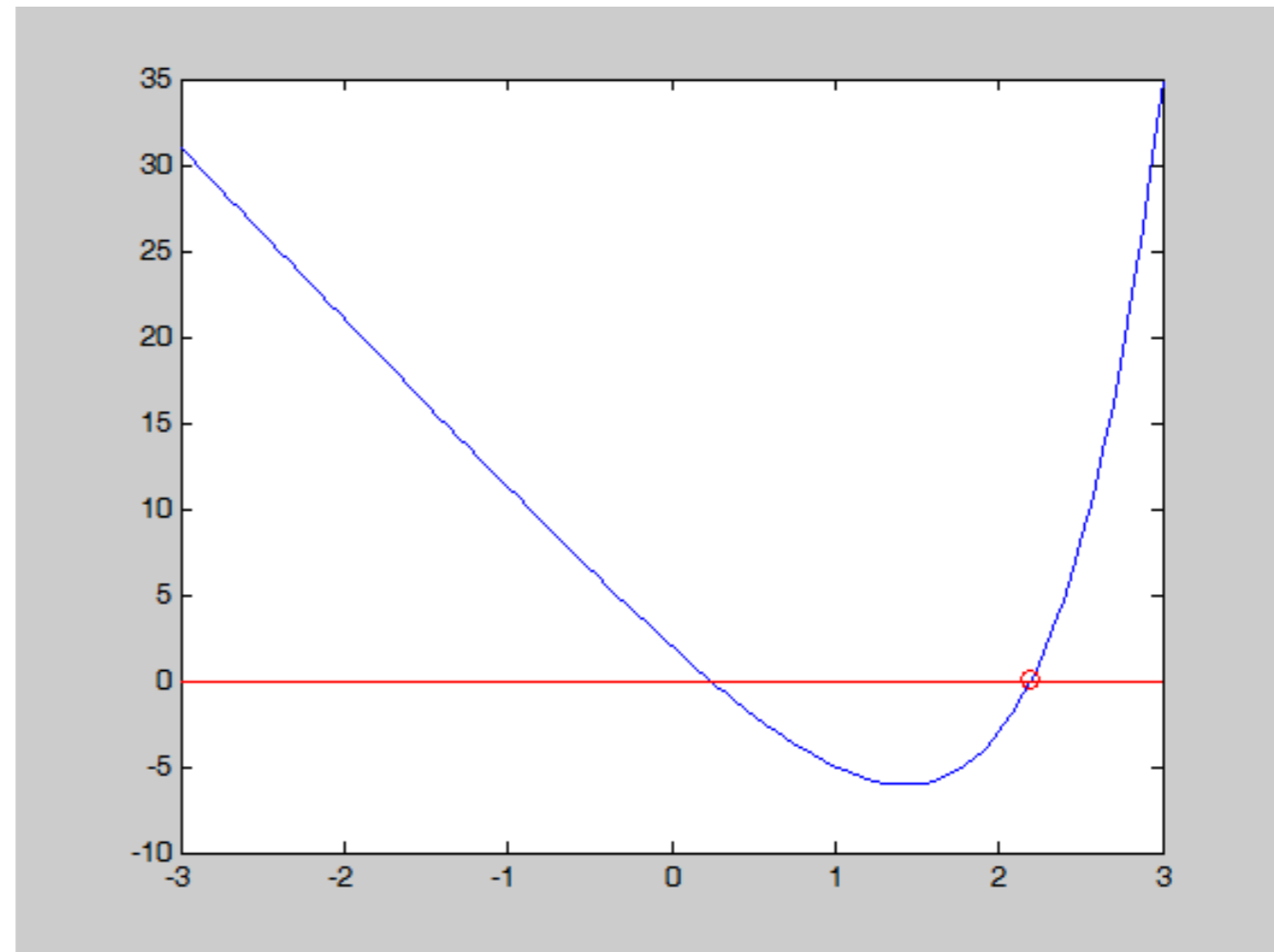
```
8  ss=['diff(' fstr ')'];
9  x=sym('x');
10 fstr1=eval(ss);
11 fx1=inline(fstr1);
12 xzero=x_ini;
13 it=0;
14 while abs(fx(xzero)) > ep
15     it=it+1;
16     if abs(fx1(xzero)) < ep
17         fprintf('zero derivative\n');
18         return
19     end
20     xzero=xzero-fx(xzero)/fx1(xzero);
21     fprintf(' iter=%0d x=%0f fx=%0f\n',it,xzero,fx(xzero));
22 end
```

```
>> demo_newton
input a function:cos(x)
guess its zero:2
iter=1 x=1.542342 fx=0.028450
iter=2 x=1.570804 fx=-0.000008
iter=3 x=1.570796 fx=0.000000
>>
```

# Example

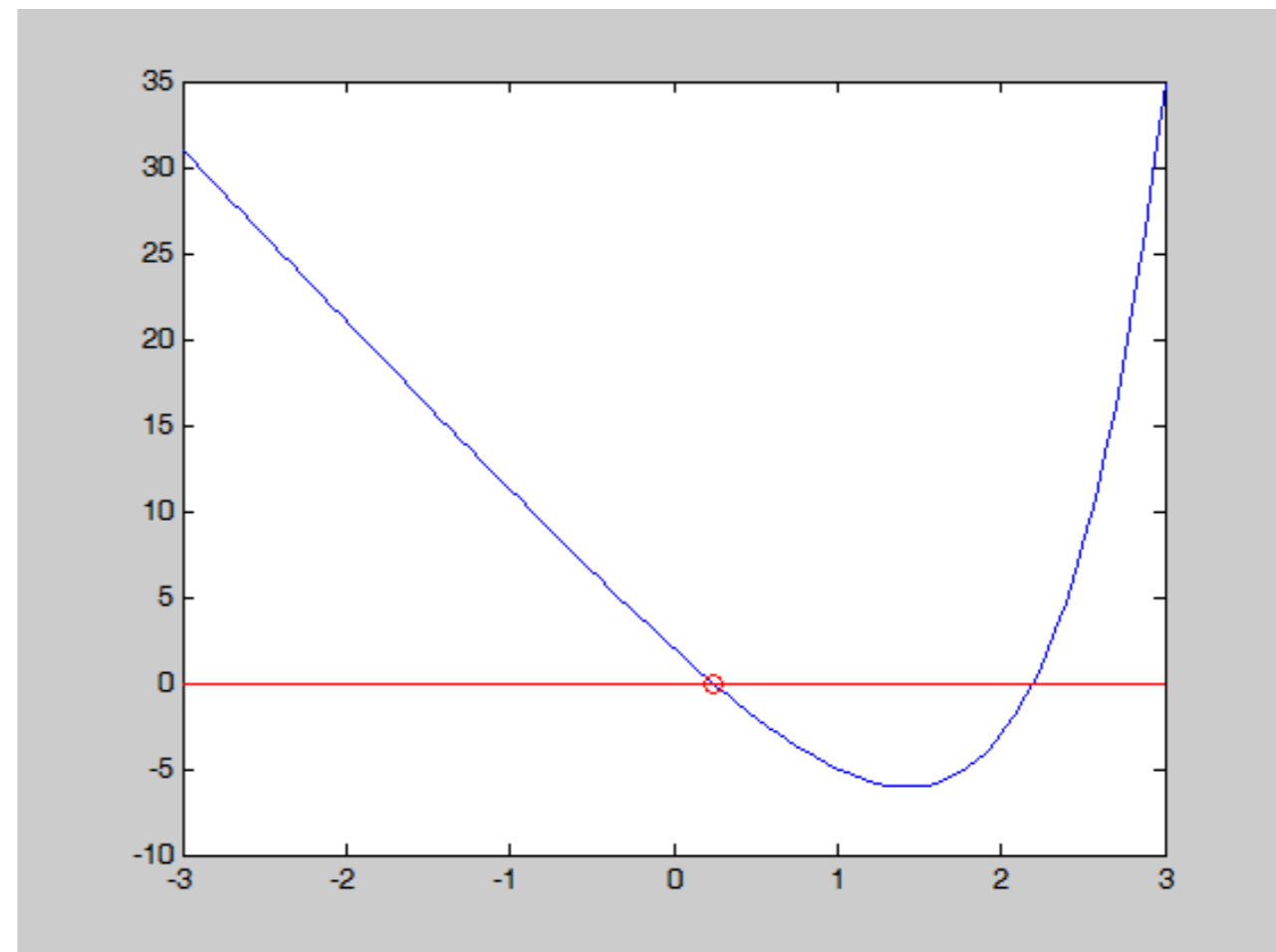
demo\_newton  
input a function:  $2x^2 - 10x + 1$   
guess its zero: 1.5

$$2x^2 - 10x + 1$$



# Example

```
>> demo_newton  
input a function:2.^x.^2-10*x+1  
guess its zero:0.5
```

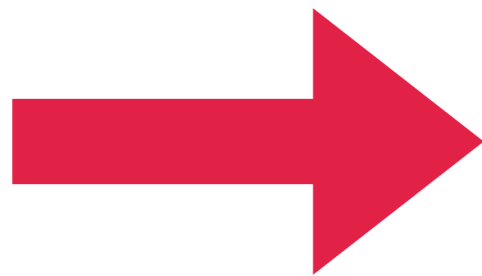


非限制式最佳化  
**Unconstrained  
Optimization**

# Unconstrained Optimization

- Given a differentiable function,  $y=h(x)$ , unconstrained optimization aims to find the minimum of  $h(x)$
- Let  $x$  denote a minimum and

$$f(x) = \frac{dh(x)}{dx}$$



$$f(x) = \frac{dh(x)}{dx} = 0$$

## Optimization by Newton method

- 步驟一

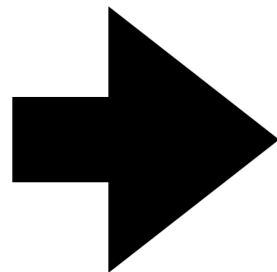
Use symbolic differentiation to find the first derivative of a given function

- 步驟二

Apply the Newton method to find zeros of the first derivative

$$f(x) = \frac{dh(x)}{dx}$$

求h的極值  
(minima)



給定h，求f

給定f，求根

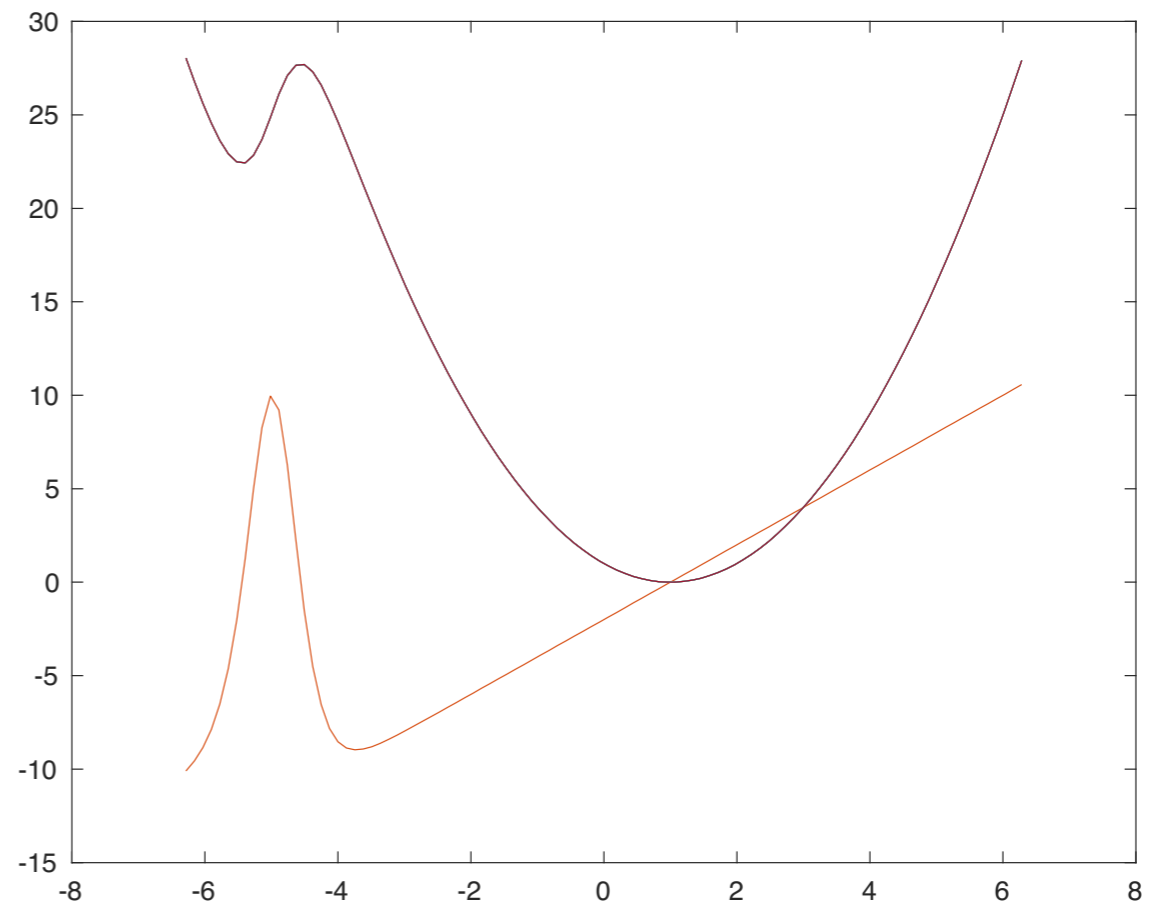
$$f(x) = \frac{dh(x)}{dx}$$

$$h(x) = (x - \tanh(2x + 10))^2$$

$$f(x) = ?$$

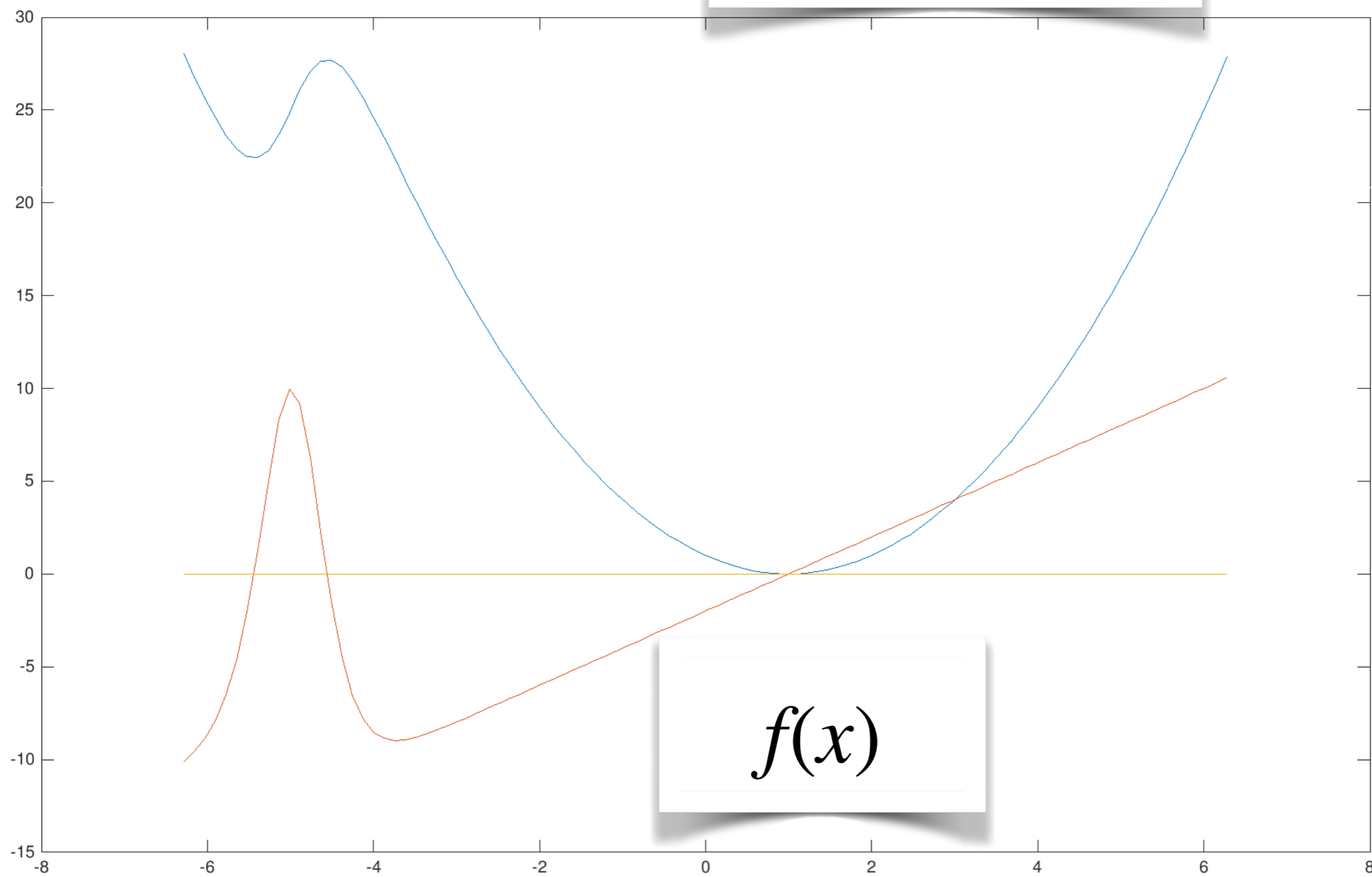
# First derivative

```
s = "(x-tanh(2*x+10)).^2"  
h = inline(s);  
  
syms x;  
ss = "diff(" + s + ")";  
s = eval(ss);  
f = inline(s);  
  
z = linspace(-2*pi,2*pi);  
plot(z,h(z));hold on;  
plot(z,f(z));
```



h的極值，是f的根

$$f(x) = \frac{dh(x)}{dx}$$



$f(x)$

函數字串與  
inline函數

執行符號微分  
指令的字串

f代表h的導數

```
s = "(x-tanh(2*x+10)).^2"  
h = inline(s);
```

```
syms x;  
ss = "diff(" + s + ")";  
s = eval(ss);  
f = inline(s);
```

```
z = linspace(-2*pi,2*pi);  
plot(z,h(z));hold on;  
plot(z,f(z));
```

```
>> h
```

```
h =
```

```
Inline function:
```

```
h(x) = (x-tanh(2*x+10)).^2
```

```
>> f
```

```
f =
```

```
Inline function:
```

```
f(x) = (tanh(x.*2.0+1.0e+1).^2.*2.0-1.0).*(x-tanh(x.*2.0+1.0e+1)).*2.0
```

結合Newton方法，求函  
數的極值

產生的函數字串

```
s = "(x-tanh(2*x+10)).^2"  
h = inline(s);
```

```
syms x;  
ss = "diff(" + s + ")";  
s = eval(ss);  
f = inline(s);
```

```
z = linspace(-2*pi,2*pi);  
plot(z,h(z));hold on;  
plot(z,f(z));
```

# Iterative approach

輸入h的函數字串  
產生f的函數字串

產生f的inline函數  
產生f1的inline函數  
設定x\_zero的初始值

$\sim(\text{abs}(f(x\_zero)) < 10^{-6})$

End

$x\_zero = x\_zero - f(x\_zero) / f1(x\_zero)$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

# Iterative approach

```
s = "(x-tanh(2*x+10)).^2"  
syms x;  
ss = "diff(" + s + ")";  
s = eval(ss);
```

```
%s='x.^2-5*x+6'  
f=inline(s);  
ss = "diff(" + string(s) + ")";  
s1 = eval(ss);  
f1=inline(s1); x_zero=rand;
```

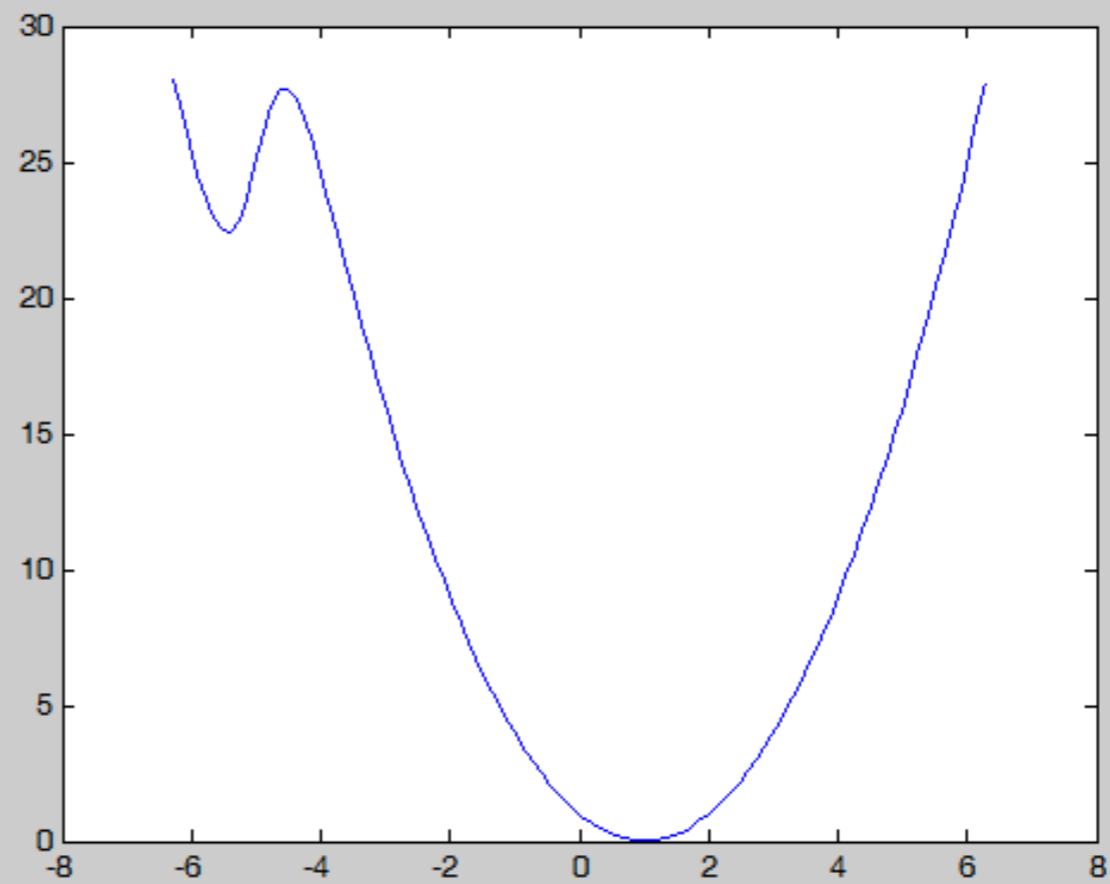
~( abs(f(x\_zero)) < 10^-6)

End

```
x_zero=x_zero-f(x_zero)/f1(x_zero)
```

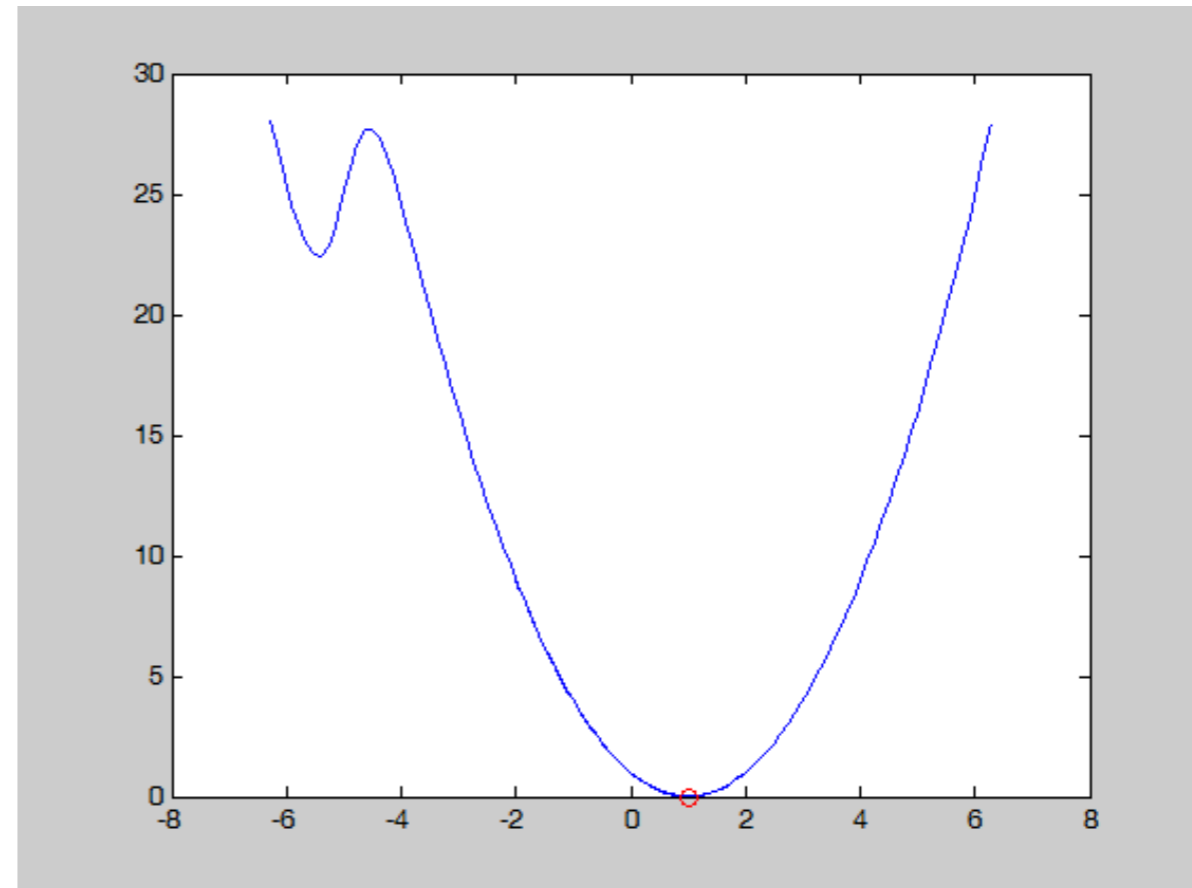
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

```
>> x=linspace(-2*pi,2*pi);  
>> plot(x,(x-tanh(2*x+10)).^2)
```



# Local minimum

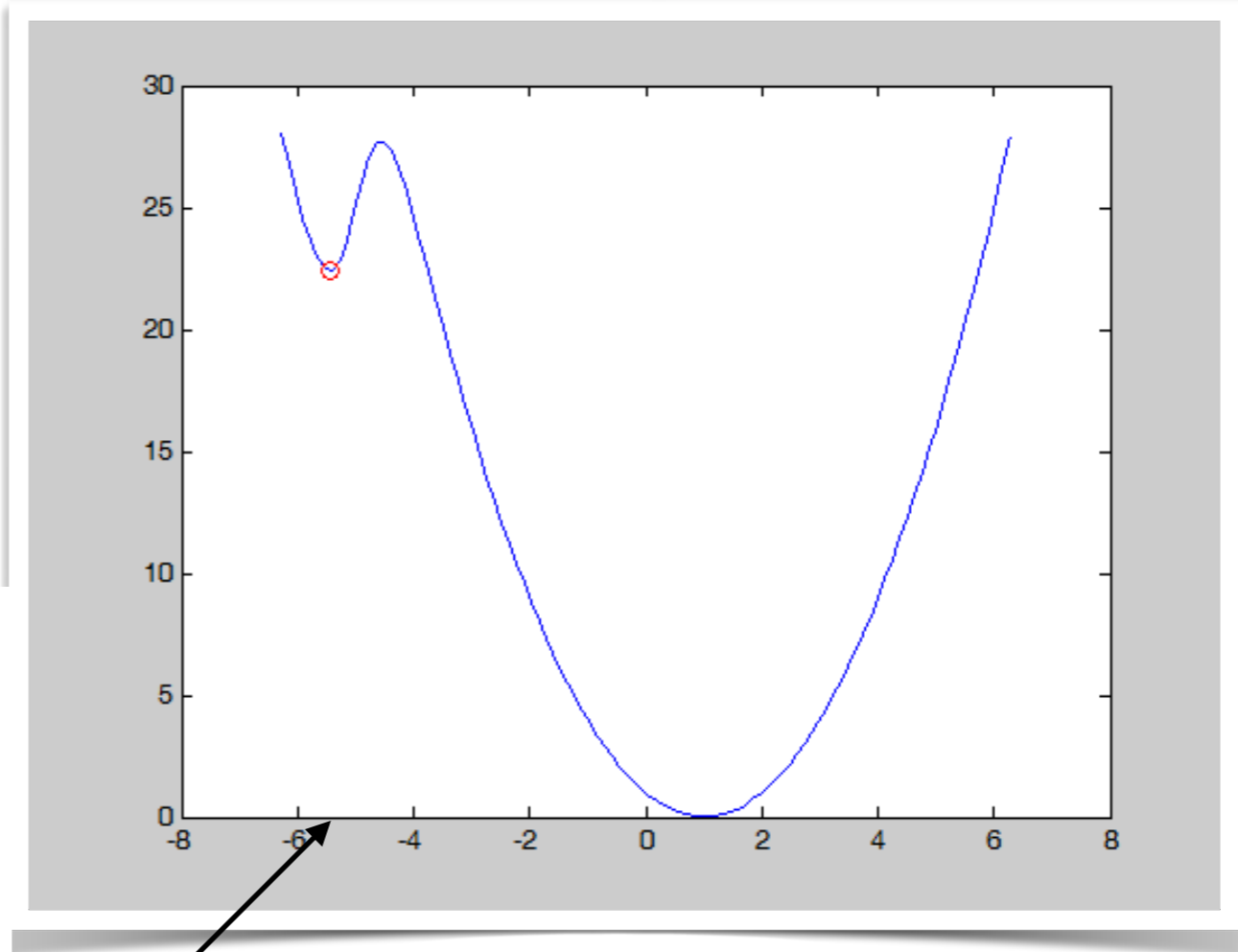
```
>> demo_min  
input a function:(x-tanh(2*x+10)).^2  
guess its minimum:4
```



# Local Minimum

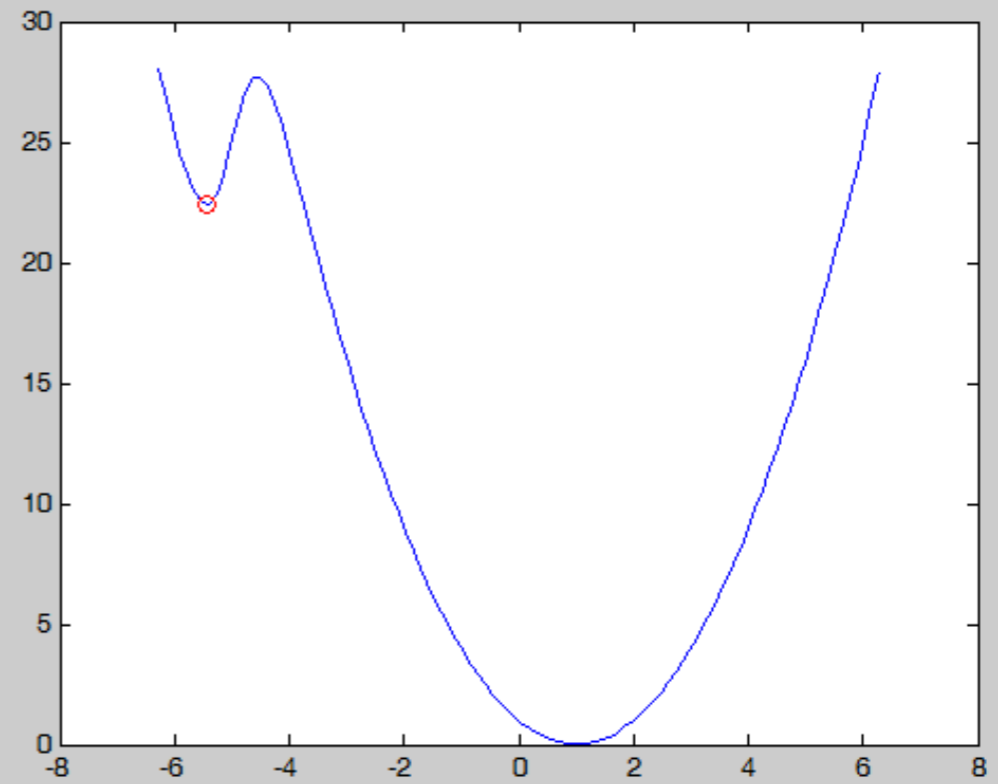
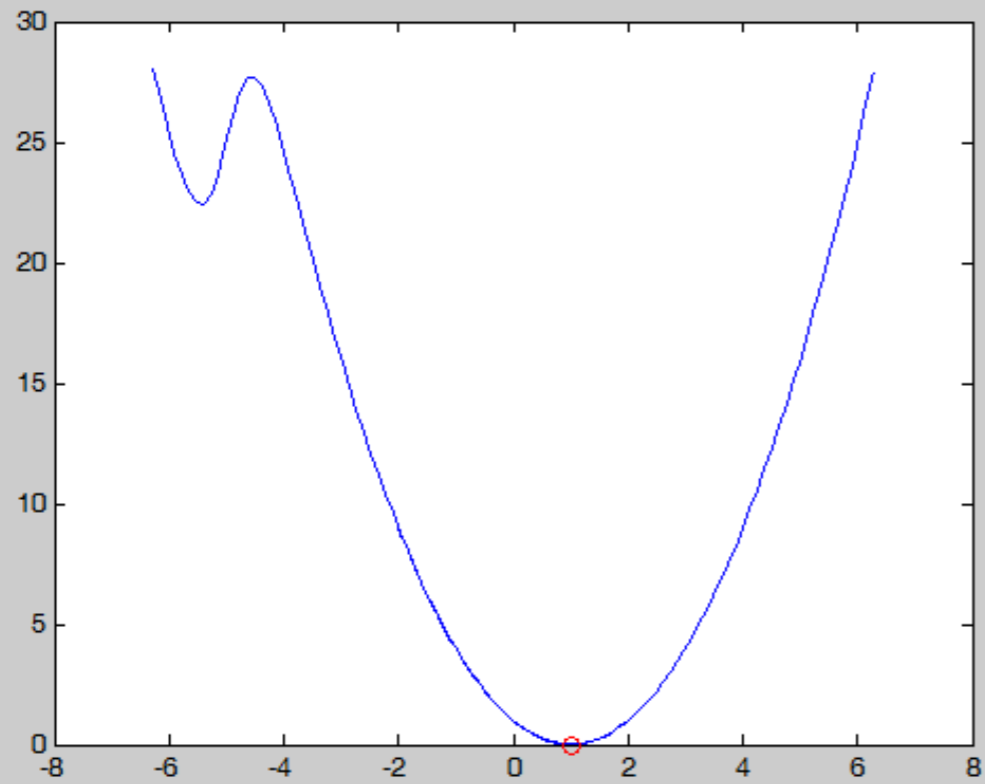
```
>> demo_min  
input a function:(x-tanh(2*x+10)).^2  
guess its minimum:-4
```

```
x_zero = -6.166606 f(x) = -9.604612  
x_zero = -4.132548 f(x) = -7.768752  
x_zero = -5.134212 f(x) = 8.404744  
x_zero = -5.543811 f(x) = -2.537791  
x_zero = -5.429992 f(x) = 0.288497  
x_zero = -5.440610 f(x) = 0.002064  
x_zero = -5.440687 f(x) = 0.000000
```



-5.440687

# Global minimum



```
>> demo_min  
input a function:0.2*x.^3-3*x+cos(x)  
guess its minimum:2
```

