A system of nonlinear equations

Newton's method

Levenberg-Marquardt method

Outline

- Newton's method for nonlinear system solving
  - Updating rule
  - Matlab implementation

```
function F = myfun(x)
   F(1) =  x(1)^2 + x(2)^2-1;
   F(2) =  x(1)^2 - x(2)^2;
return
```

$$f_1(x_1, x_2) = x_1^2 + x_2^2 - 1$$

$$f_2(x_1, x_2) = x_1^2 - x_2^2$$

symbols

```
s1='x1^2+x2^2-1';
s2='x1^2-x2^2';
x1=sym('x1')
x2=sym('x2')
```

Inline Function

```
f=inline([str2sym(s1);str2sym(s2)]);
f(0,0)
```

fsolve

```
x=fsolve(@(x) [x(1)^2+x(2)^2-1 x(1)^2-x(2)^2],[1 1])
```

```
x =

    0.7071    0.7071
```

```
s1='x1^2+x2^2-1';
s2='x1^2-x2^2';

x1=sym('x1')
x2=sym('x2')
f=inline([str2sym(s1);str2sym(s2)]
f(x(1),x(2))
```

zeros

```
ans =

   1.0e-11 *

    0.2282
         0
```

Jacobian

```
A=jacobian([str2sym(s1);str2sym(s2)],[x1 x2]);
j=inline(A);
j(1,1)
```

$$f_1(x_1, x_2) = x_1^2 + x_2^2 - 1$$

$$f_2(x_1, x_2) = x_1^2 - x_2^2$$

```
A =

[ 2*x1,  2*x2]
[ 2*x1, -2*x2]
```

$$\begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} \end{bmatrix}$$

Nonlinear system

A system of nonlinear equations

$$F(x_1, x_2, \ldots, x_n) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \cdot \\ \cdot \\ \cdot \\ f_n(x_1, x_2, \ldots, x_n) \end{bmatrix}$$

$f_1, f_2, \ldots, f_n$ are coordinate functions of F

Example

$$3x_1 - \cos(x_2 x_3) - \frac{1}{2} = 0$$

$$x_1^2 - 81(x_2 + 0.1)^2 + \sin(x_3) + 1.06 = 0$$

$$e^{-x_1 x_2} + 20x_3 + \frac{1}{3}(10\pi - 3) = 0$$

myfun

```
function F = myfun(x)
    F(1) =  3*x(1)-cos(x(2)*x(3))-1/2;
    F(2) =  x(1).^2 -81*(x(2)+0.1).^2+sin(x(3))+1.06;
    F(3) = exp(-x(1)*x(2))+20*x(3)+1/3*(10*pi-3);
return
```

Example

$$x_1^2 + x_2^2 + x_3^2 = 4$$

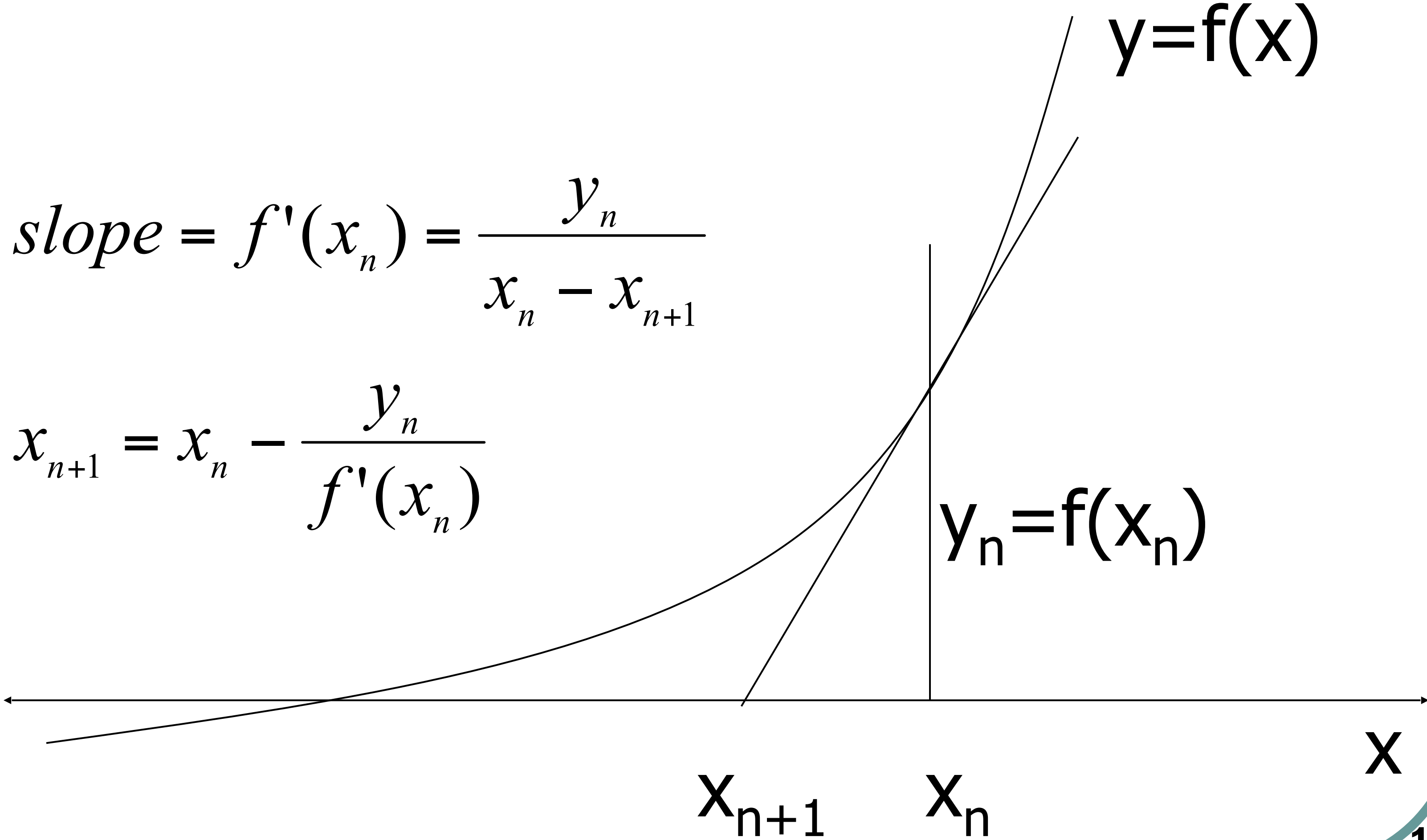
$$2x_1 - x_2 + x_3 = 1$$

$$x_1 + 3x_2 - x_3 = 3$$

Newton’s method -Tangent line

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

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



Updating rule

x : scalar

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

x : vector

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \left[J(\mathbf{x}_n)\right]^{-1} F(\mathbf{x}_n)$$

Taylor series

- Second order expansion at x= x_n

$$F(\mathbf{x}+\Delta\mathbf{x}) \approx F(\mathbf{x}) + J(\mathbf{x})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T H(\mathbf{x})\Delta\mathbf{x}$$

Jacobi matrix

Hessian matrix

$$\mathbf{x} \leftarrow \mathbf{x}_n, \quad \Delta\mathbf{x} \leftarrow \mathbf{x} - \mathbf{x}_n,$$

$$F(\mathbf{x}) \approx F(\mathbf{x}_n) + J(\mathbf{x}_n)(\mathbf{x}-\mathbf{x}_n) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_n)^T H(\mathbf{x}_n)(\mathbf{x}-\mathbf{x}_n)$$

Hessian Matrix

- $$H(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f_1(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f_1(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f_1(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f_2(\mathbf{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f_2(\mathbf{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f_2(\mathbf{x})}{\partial x_1 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f_n(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 f_n(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f_n(\mathbf{x})}{\partial x_n^2} \end{bmatrix}$$

Jacobi matrix

-

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

$$x_1^2 + x_2^2 + x_3^2 = 4$$

$$2x_1 - x_2 + x_3 = 1$$

$$x_1 + 3x_2 - x_3 = 3$$

$$J(\mathbf{x}) = \begin{bmatrix} 2x_1 & 2x_2 & 2x_3 \\ 2 & -1 & 1 \\ 1 & 3 & -1 \end{bmatrix}$$

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

$$x_1^2 + x_2^2 + x_3^2 = 4$$
$$2x_1 - x_2 + x_3 = 1$$
$$x_1 + 3x_2 - x_3 = 3$$

```
s1='x1^2+x2^2+x3^2-4';
s2='2*x1-x2+x3-1';
s3='x1+3*x2-x3+3';

x1=sym('x1')
x2=sym('x2')
x3=sym('x3')
A=jacobian([str2sym(s1);str2sym(s2);str2sym(s3)],[x1 x2 x3]);
j=inline(A);
```

```
A =

[ 2*x1, 2*x2, 2*x3]
[    2,   -1,    1]
[    1,    3,   -1]
```

$$J(\mathbf{x}) = \begin{bmatrix} 2x_1 & 2x_2 & 2x_3 \\ 2 & -1 & 1 \\ 1 & 3 & -1 \end{bmatrix}$$

First order expansion

- *Set* zero to

$$F(\mathbf{x}) \approx F(\mathbf{x}_n) + J(\mathbf{x}_n)(\mathbf{x} - \mathbf{x}_n)$$

$$F(\mathbf{x}_n) + J(\mathbf{x}_n)(\mathbf{x} - \mathbf{x}_n) = 0 \;\; \Rightarrow \;\; \mathbf{x} = \mathbf{x}_n - J^{-1}(\mathbf{x}_n)F(\mathbf{x}_n)$$

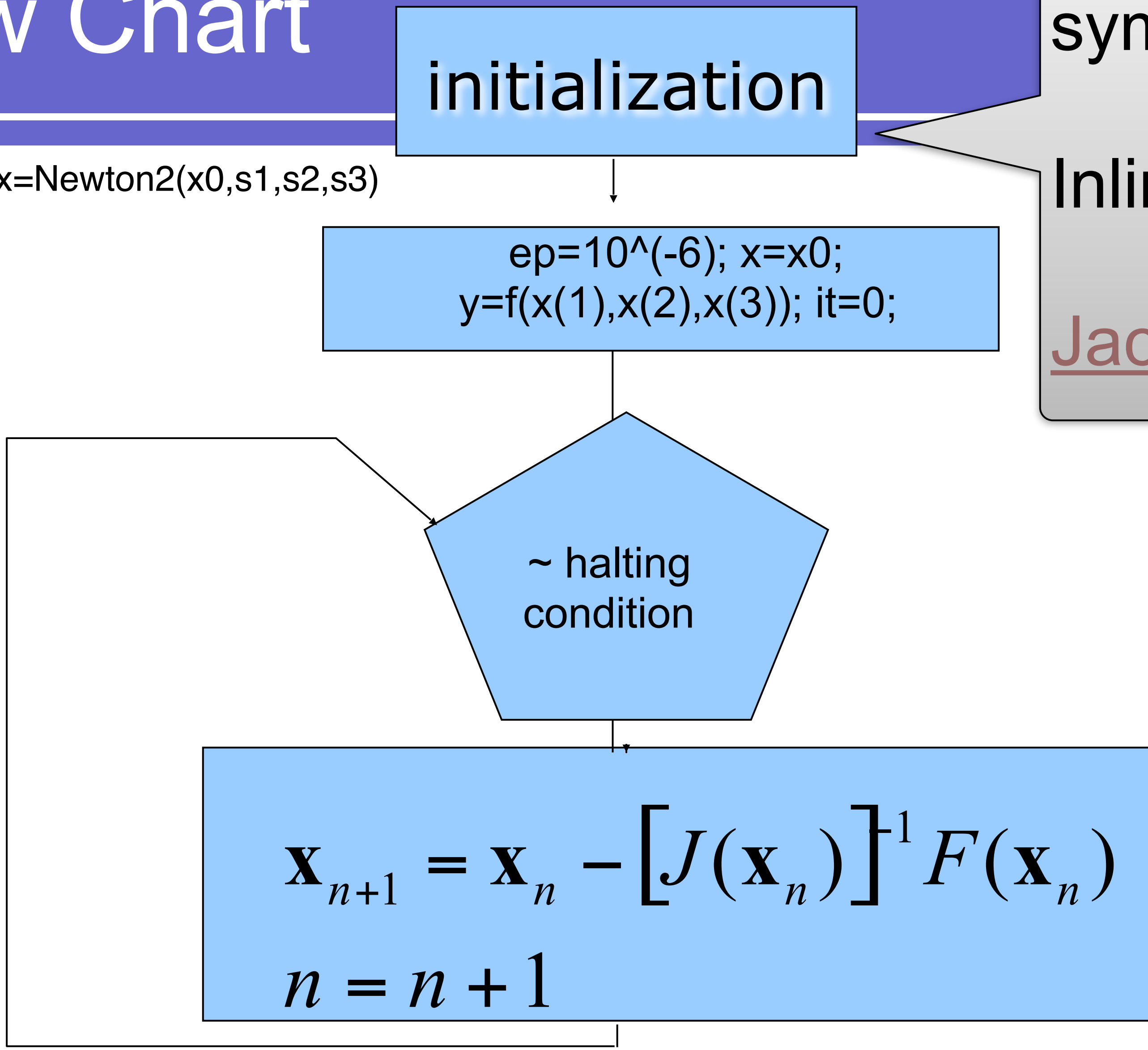
Newton's method

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \left[J(\mathbf{x}_n)\right]^{-1} F(\mathbf{x}_n)$$

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Flow Chart

function x=Newton2(x0,s1,s2,s3)



symbols

Inline function

Jacobian

```
s1='3*x1-cos(x2*x3)-1/2';
s2='x1^2 -81*(x2+0.1)^2+sin(x3)+1.06';
s3='exp(-x1*x2)+20*x3+1/3*(10*pi-3)';
x1=sym('x1')
x2=sym('x2')
X3=sym('x3')
```

symbols

inline Function

```
f=inline([str2sym(s1);str2sym(s2) ;str2sym(s3)]);
f(0,0,0)
```

Inline function

Jaconian

```
A=jacobian([str2sym(s1);str2sym(s2) ;str2sym(s3)],[x1 x2 x3]);
j=inline(A);
j(1,1,1)
```

Jacobian

Symbols, inline and Jacobian

```
s1='3*x1-cos(x2*x3)-1/2';
s2='x1^2 -81*(x2+0.1)^2+sin(x3)+1.06';
s3='exp(-x1*x2)+20*x3+1/3*(10*pi-3)';
x1=sym('x1');x2=sym('x2');x3=sym('x3');
f=inline([str2sym(s1);str2sym(s2) ;str2sym(s3)]);
A=jacobian([str2sym(s1);str2sym(s2) ;str2sym(s3)],[x1 x2 x3]);
j=inline(A);
```

symbols

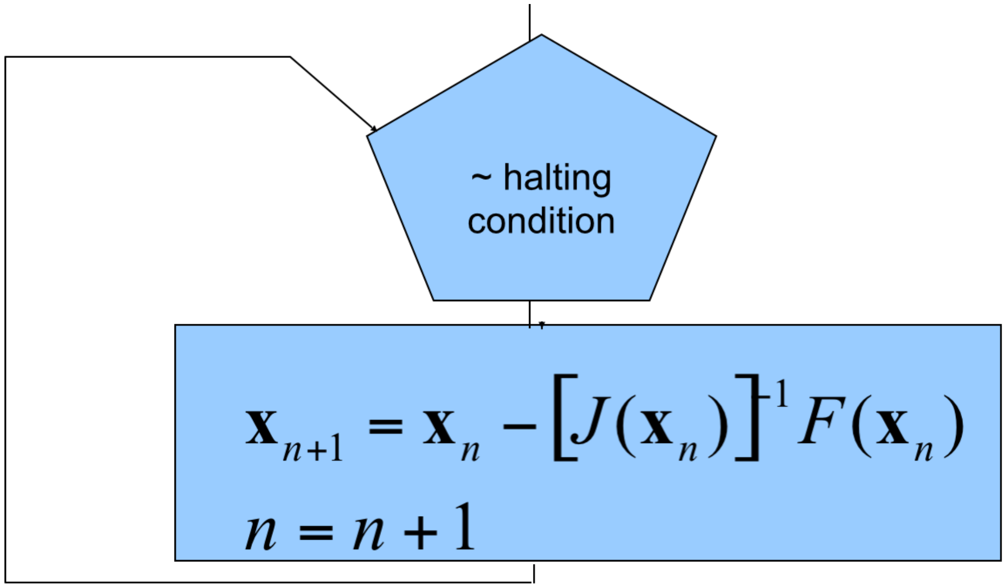
Inline function

Jacobian

Init

```
ep=10^(-6); x=x0;
y=f(x(1),x(2),x(3)); it=0;
```

```
while sum(abs(y)) > ep & it < 100
x=x-inv(j(x(1),x(2),x(3)))*y;
y=f(x(1),x(2),x(3))
it=it+1
end
x
```



- Implement the Newton's method for solving a three-variable nonlinear system
- Test your matlab codes with the following nonlinear system

$$3x_1 - \cos(x_2 x_3) - \frac{1}{2} = 0$$

$$x_1^2 - 81(x_2 + 0.1)^2 + \sin(x_3) + 1.06 = 0$$

$$e^{-x_1 x_2} + 20x_3 + \frac{1}{3}(10\pi - 3) = 0$$

$$x_1^2 + x_2^2 + x_3^2 = 4$$

$$2x_1 - x_2 + x_3 = 1$$

$$x_1 + 3x_2 - x_3 = 3$$