MATLAB 科学计算语言与应用

Lecture 3: Solving Equations, Curve Fitting, and Numerical Techniques

Outline

(1) Linear Algebra

(2)Polynomials

(3)Optimization

(4) Differentiation/Integration

(5) Differential Equations

Systems of Linear Equations

- Given a system of linear equations
  - x+2y-3z=5
  - -3x-y+z=-8
  - x-y+z=0
- Construct matrices so the system is described by Ax=b

```
» A=[1 2 -3;-3 -1 1;1 -1 1];
» b=[5;-8;0];
```

- And solve with a single line of code!

```
» x=A\b;
```

  - x is a 3x1 vector containing the values of x, y, and z

- **The \ will work with square or rectangular systems.**
- Gives least squares solution for rectangular systems.

Solve the following equations:

- System 1:

$$x + 4y = 34$$
$$-3x + y = 2$$

```
» A=[1 4;-3 1];
» b=[34;2];
» rank(A)
» x=inv(A)*b;
» x=A\b;
```

- System 2:

$$2x - 2y = 4$$
$$-x + y = 3$$
$$3x + 4y = 2$$

```
» A=[2 -2;-1 1;3 4];
» b=[4;3;2];
» rank(A)
```

- rectangular matrix

```
» x=A\b;
```

- gives least squares solution

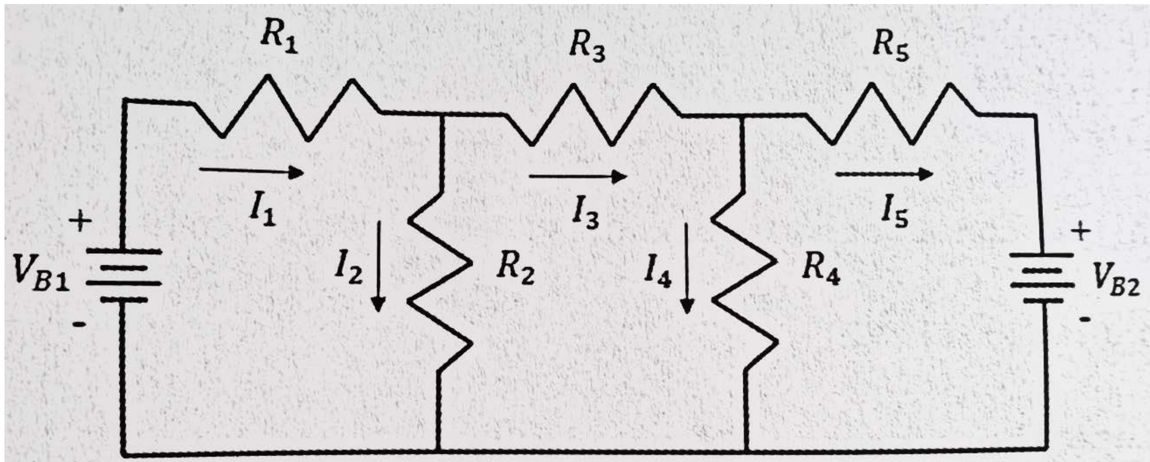
```
» error=abs(A*x1-b)
```

More Linear Algebra

- Given a matrix
  - » `mat=[1 2 -3;-3 -1 1;1 -1 1];`
- Calculate the rank of a matrix
  - » `r=rank(mat);`
    - ➢ the number of linearly independent rows or columns
- Calculate the determinant
  - » `d=det(mat);`
    - ➢ mat must be square; matrix invertible if det nonzero
- Get the matrix inverse
  - » `E=inv(mat);`
    - ➢ if an equation is of the form A*x=b with A a square matrix, x=A\b is (mostly) the same as x=inv(A)*b
- Get the condition number
  - » `c=cond(mat);` (or its reciprocal: `c = rcond(mat);`)
    - ➢ if condition number is large, when solving A*x=b, small errors in b can lead to large errors in x (optimal c==1)

Exercise3-1: 求解电路问题（1）

问题1：计算如图所示每个分支的电流



基尔霍夫定律：回路电压=0；节点电流=0；

$$
\begin{aligned}
I_1 &= I_2 + I_3\\
I_3 &= I_4 + I_5\\
V_{B1} &= R_1 I_1 + R_2 I_2\\
R_2 I_2 &= R_3 I_3 + R_4 I_4\\
V_{B2} &= -R_5 I_5 + R_4 I_4
\end{aligned}
$$

$$
\begin{aligned}
I_1 - I_2 - I_3 &= 0\\
I_3 - I_4 - I_5 &= 0\\
R_1 I_1 + R_2 I_2 &= V_{B1}\\
R_2 I_2 - R_3 I_3 - R_4 I_4 &= 0\\
R_4 I_4 - R_5 I_5 &= V_{B2}
\end{aligned}
$$

$$
\begin{bmatrix} 1 & -1 & -1 & 0 & 0\\ 0 & 0 & 1 & -1 & -1\\ R_1 & R_2 & 0 & 0 & 0\\ 0 & R_2 & -R_3 & -R_4 & 0\\ 0 & 0 & 0 & R_4 & -R_5 \end{bmatrix}
\begin{bmatrix} I_1\\ I_2\\ I_3\\ I_4\\ I_5 \end{bmatrix}
=
\begin{bmatrix} 0\\ 0\\ V_{B1}\\ 0\\ V_{B2} \end{bmatrix}
$$

Exercise3-1: 求解电路问题（2）

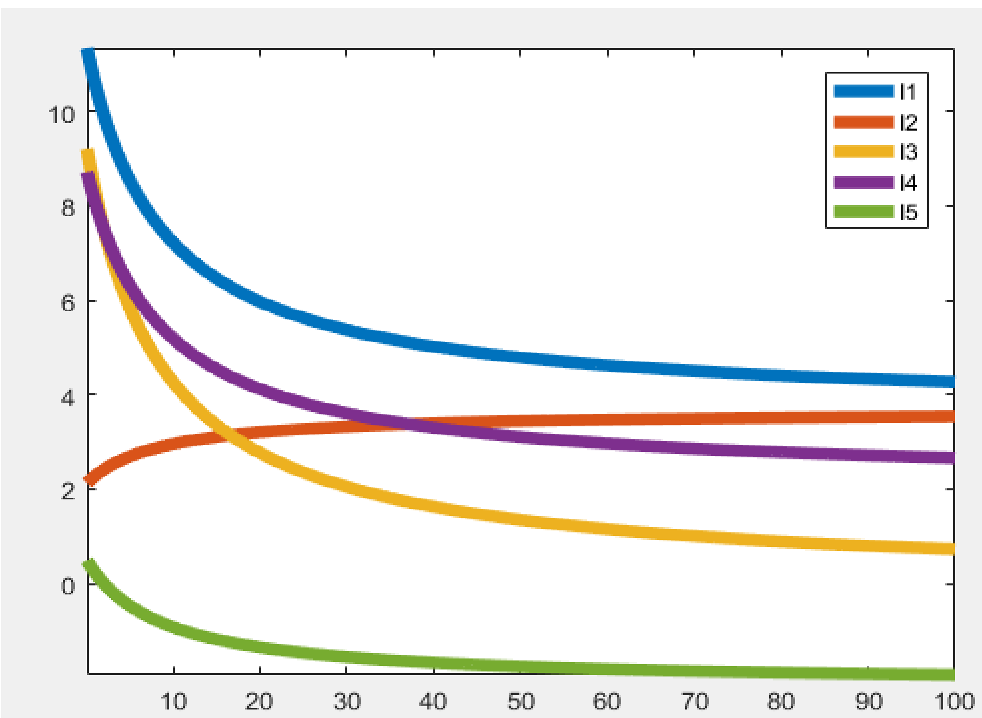
$$\begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \end{bmatrix} = \begin{bmatrix} 5 \\ 25 \\ 12 \\ 6 \\ 15 \end{bmatrix}$$

求解得：

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix} = \begin{bmatrix} 6.8778 \\ 3.0244 \\ 3.8533 \\ 4.8952 \\ -1.0419 \end{bmatrix}$$

其中，I_5=-1.0419，说明其实际流向与图中假设流向相反

问题2：把R3改成可变电阻（可变电阻在电路中常用来控制电流大小，如收音机流量）。观察R3在0.1Ω ~ 100 Ω变化时，各个电流变化。



当R3>30 Ω后，系统对R3变化的响应不明显；

从工程角度看，应选择0.1Ω ~ 30 Ω的可变电阻即可

Matrix Decompositions

- MATLAB has many built-in matrix decomposition methods

- The most common ones are
  - » `[V,D]=eig(X)`
    - ➢ Eigenvalue decomposition
  - » `[U,S,V]=svd(X)`
    - ➢ Singular value decomposition
  - » `[Q,R]=qr(X)`
    - ➢ QR decomposition
  - » `[L,U]=lu(X)`
    - ➢ LU decomposition
  - » `R=chol(X)`
    - ➢ Cholesky decomposition (X must be positive definite)

Exercise3-2: Matrix Operations

- 已知矩阵$A=\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 3\\ 1 & 4 & 9\end{bmatrix}$，求其行列式，转置，秩，逆，特征值、特征向量。

特征向量：

$$\begin{matrix}-0.1324 & -0.7300 & 0.5730\\ -0.3401 & -0.5645 & -0.7692\\ -0.9310 & 0.3853 & 0.2829\end{matrix}$$

特征值：

$$\begin{matrix}10.6031 & 0 & 0\\ 0 & 1.2454 & 0\\ 0 & 0 & 0.1515\end{matrix}$$

Outline

(1) Linear Algebra

(2)Polynomials

(3)Optimization

(4) Differentiation/Integration

(5) Differential Equations

Polynomials

- Many functions can be well described by a high-order polynomial

- MATLAB represents a polynomials by a vector of coefficients
  - if vector P describes a polynomial

    $$\underset{P(1)}{a}x^3+\underset{P(2)}{b}x^2+\underset{P(3)}{c}x+\underset{P(4)}{d}$$

- P=[1 0 -2] represents the polynomial x^2-2

- P=[2 0 0 0] represents the polynomial $2x^3$

Polynomial Operations

- P is a vector of length N+1 describing an N-th order polynomial
- To get the roots of a polynomial

  » `r=roots(P)`

  ➢ r is a vector of length N

- Can also get the polynomial from the roots

  » `P=poly(r)`

  ➢ r is a vector length N

- To evaluate a polynomial at a point

  » `y0=polyval(P,x0)`

  ➢ x0 is a single value; y0 is a single value

- To evaluate a polynomial at many points

  » `y=polyval(P,x)`

  ➢ x is a vector; y is a vector of the same size

Polynomial Operations

- 多项式的加减运算（略）
- 多项式乘法

  P=conv(p1,p2)   %p1,p2,p都是多项式的系数向量
- 多项式除法

  [q,r]=deconv(p1, p2)   %q,r分别是商式和余式的系数向量

  乘法和除法可逆：p1=conv(p2,q)+r;
- 多项式微分

  P=polyder(p1);   %多项式p1的导数

  P=polyder(p1,p2);   %多项式乘积p1*p2的导数

  [p,q]=polyder(p1,p2)%多项式相除p1/p2的导数，p,q分别为分子分母多项式系数向量

Polynomial Operations

- 多项式积分

  I = polyint(p,k)　　%以p为系数的多项式的积分，k为积分常数项, p=polyder(I)

- 多项式部分分式展开

$$\frac{B(s)}{A(s)}=\frac{r_1}{s-p_1}+\frac{r_2}{s-p_2}+\frac{r_3}{s-p_3}+\cdots+\frac{r_n}{s-p_n}+k(s)$$

  [r, p, k]=residue(B, A)

  % B, A分别为分子分母多项式系数行向量，r为零点列向量，p为极点列向量，k为余式多项式系数行向量；

  [B, A]= residue(r, p, k) % 部分分式展开式转换为多项式相除

1. 已知两个多项式为

$$p_1(\mathrm{x})=x^4-3x^3+x+2,$$

$$p_2(\mathrm{x})=x^3-2x^2+4$$

(1)求多项式p_1(x)的导数；

(2)求两个多项式乘积p_1(x)*p_2(x)的导数；

(3)求两个多项式相除p_2(x)/p_1(x)的导数

(4)求x=[0, 2, 4, 6, 8]时多项式p1的值；

(5)求多项式p1的根；

Polynomial Operations

- 数据插值（根据已知离散数据点，得到更多未知数据）
  - 一维插值

    yi=interp1(X, Y, xi, ‘method’)

    %已知数据点X~Y，用method规定的插值方法求xi处的值；

    %method，可以是线性，最近邻，三次样条，三次多项式等，请查手册；

    y=interpft(y1,n)

    %一维快速傅里叶插值，n为傅里叶逆变换点数。

    yi=spline(x,y,xi)

    %相当于 yi=interp1(x, y, xi, ‘spline’);
  - 二维插值

Polynomial Operations

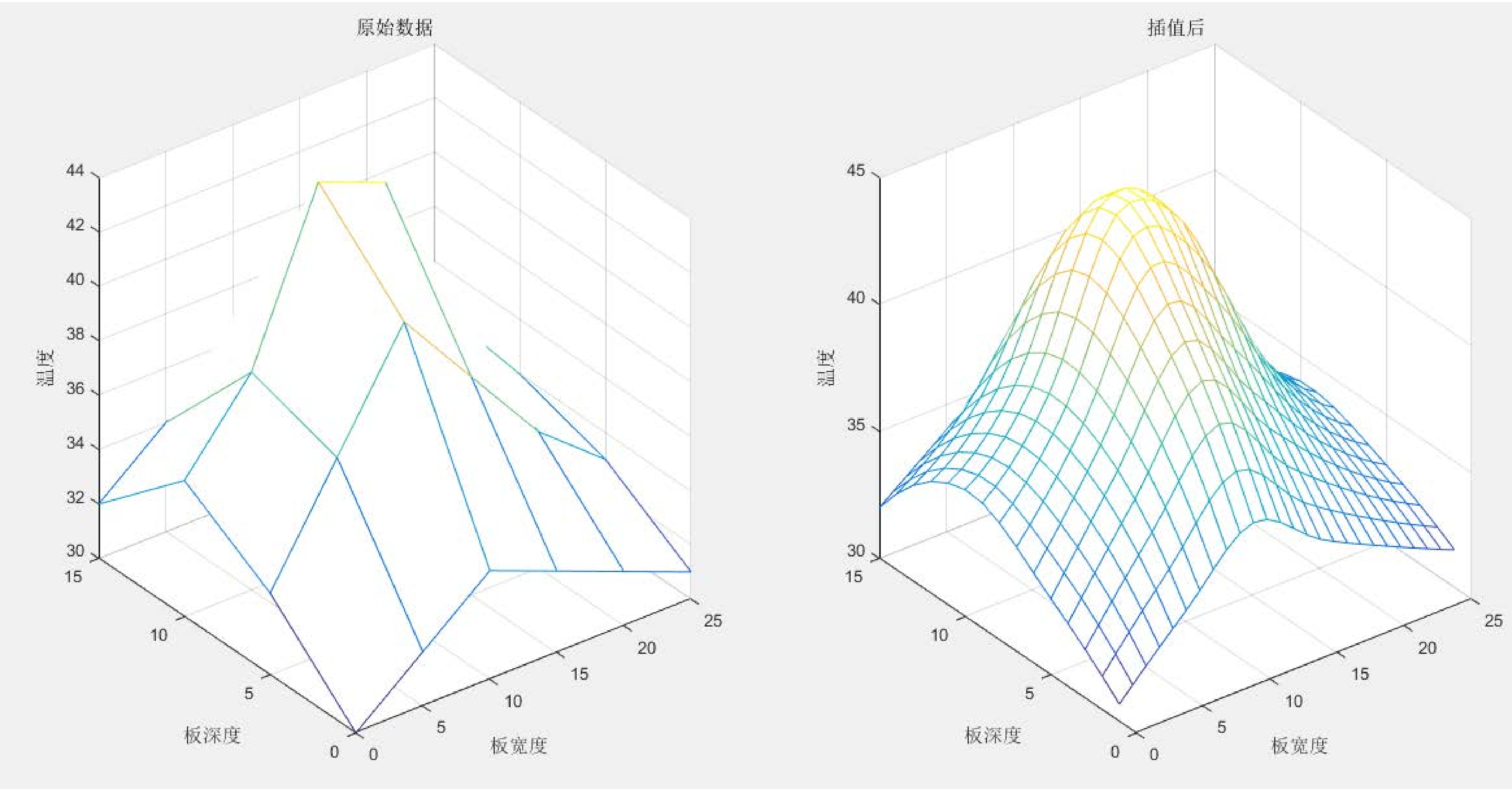
- 数据插值（根据已知离散数据点，得到更多未知数据）
  - 二维插值

  Z1=interp2(X, Y, Z, X1, Y, ‘method’);

Exercise3-3: 实验室对电脑主板温度做测试，测量结果如下表所示。X和y表示主板的宽度和深度，表中值为温度。

y \ x	0	5	10	15	20	25
0	30	32	34	33	32	31
5	33	37	41	38	35	33
10	35	38	44	43	37	34
15	32	34	36	35	33	32

Exercise3-3：主板宽度和深度每隔1cm的温度分布



练习：主板宽度每隔0.2cm和深度每隔0.1cm的温度分布

插值可得主板宽度2.6，深度7.2处的温度：41.2176

Polynomial Fitting

- MATLAB makes it very easy to fit polynomials to data

- Given data vectors X=[-1 0 2] and Y=[0 -1 3]
  - » `p2=polyfit(X,Y,2);`
    - ➢ finds the best (least-squares sense) second-order polynomial that fits the points (-1,0),(0,-1), and (2,3)
    - ➢ see **help polyfit** for more information
  - » `plot(X,Y,'o', 'MarkerSize', 10);`
  - » `hold on;`
  - » `x = -3:.01:3;`
  - » `plot(x,polyval(p2,x), 'r--');`

Exercise3-4: Polynomial Fitting

- Evaluate $y = x^2$ for x=-4:0.1:4.

- Add random noise to these samples. Use **randn**. Plot the noisy signal with **.** markers

- Fit a 2nd degree polynomial to the noisy data

- Plot the fitted polynomial on the same plot, using the same x values and a red line

Exercise3-5: Polynomial Fitting

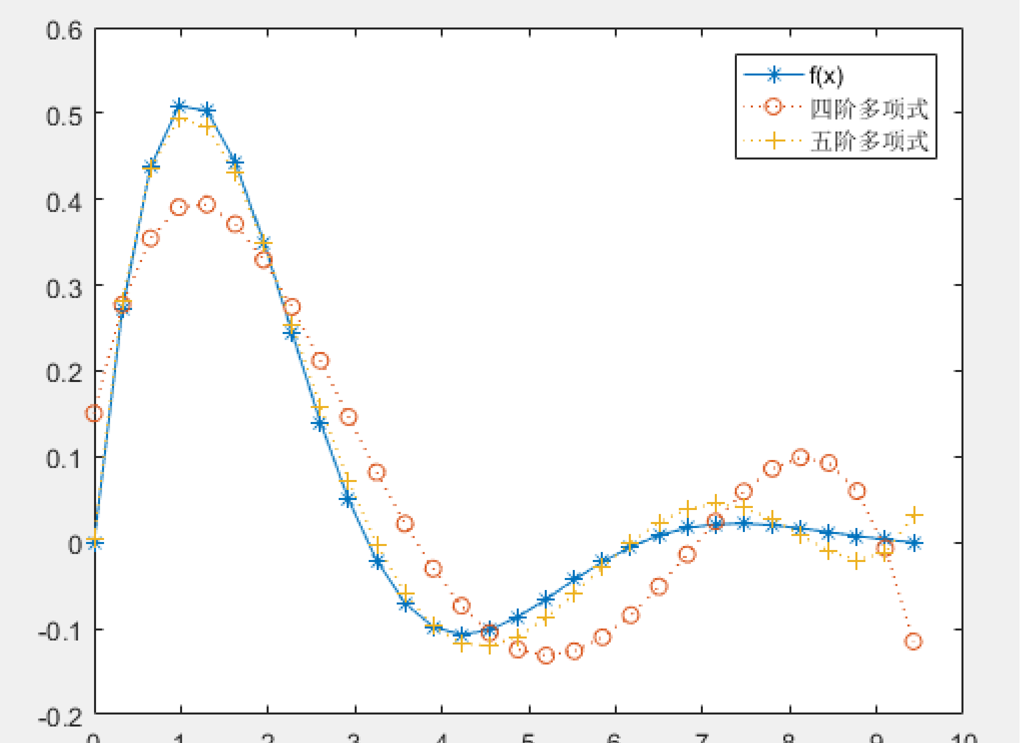
- 分别用四阶和五阶多项式在区间[0, 3π]内拟合函数

$$f(x) = e^{-0.5x}\sin(x)$$

- 绘图比较拟合的四阶多项式、五阶多项式、f(x)的区别

四阶多项式：-0.002378 x^4 + 0.04625 x^3 - 0.27815 x^2 + 0.476 x + 0.15048

五阶多项式： 0.00071166 x^5 - 0.019146 x^4 + 0.18564 x^3 - 0.75929 x^2 + 1.0826 x + 0.0045771



Outline

(1) Linear Algebra

(2)Polynomials

(3)Optimization

(4) Differentiation/Integration

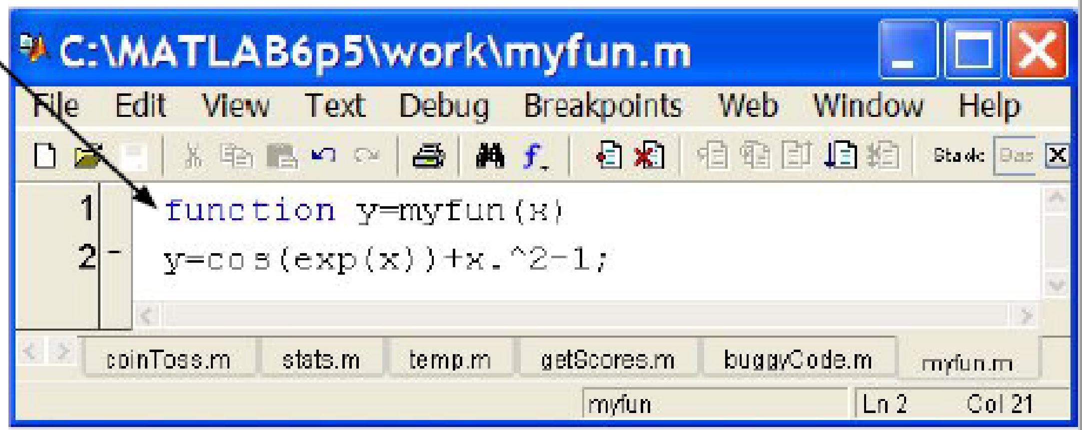
(5) Differential Equations

Nonlinear Root Finding

- Many real-world problems require us to solve f(x)=0
- Can use **fzero** to calculate roots for *any* arbitrary function

- **fzero** needs a function passed to it.
- We will see this more and more as we delve into solving equations.

- Make a separate function file
  - » `x=fzero('myfun',1)`
  - » `x=fzero(@myfun,1)`
    - ➢ 1 specifies a point close to where you think the root is



Minimizing a Function

- **fminbnd**: minimizing a function over a bounded interval

```
» x=fminbnd('myfun',-1,2);
```

  - myfun takes a scalar input and returns a scalar output
  - myfun(x) will be the minimum of myfun for $-1 \leq x \leq 2$

- **fminsearch**: unconstrained interval

```
» x=fminsearch('myfun',.5)
```

  - finds the local minimum of myfun starting at x=0.5

- Maximize g(x) by minimizing f(x)=-g(x)

- Solutions may be local!

Anonymous Functions

- You do not have to make a separate function file

```
» x=fzero(@myfun,1)
```

  - ➢ What if myfun is really simple?

- Instead, you can make an anonymous function

```
» x=fzero(@(x)(cos(exp(x))+x.^2-1),  1  );
```

input function to evaluate

```
» x=fminbnd(@(x) (cos(exp(x))+x.^2-1),-1,2);
```

- Can also store the function handle

```
» func=@(x) (cos(exp(x))+x.^2-1);
» func(1:10);
```

Optimization Toolbox

- If you are familiar with optimization methods, use the optimization toolbox

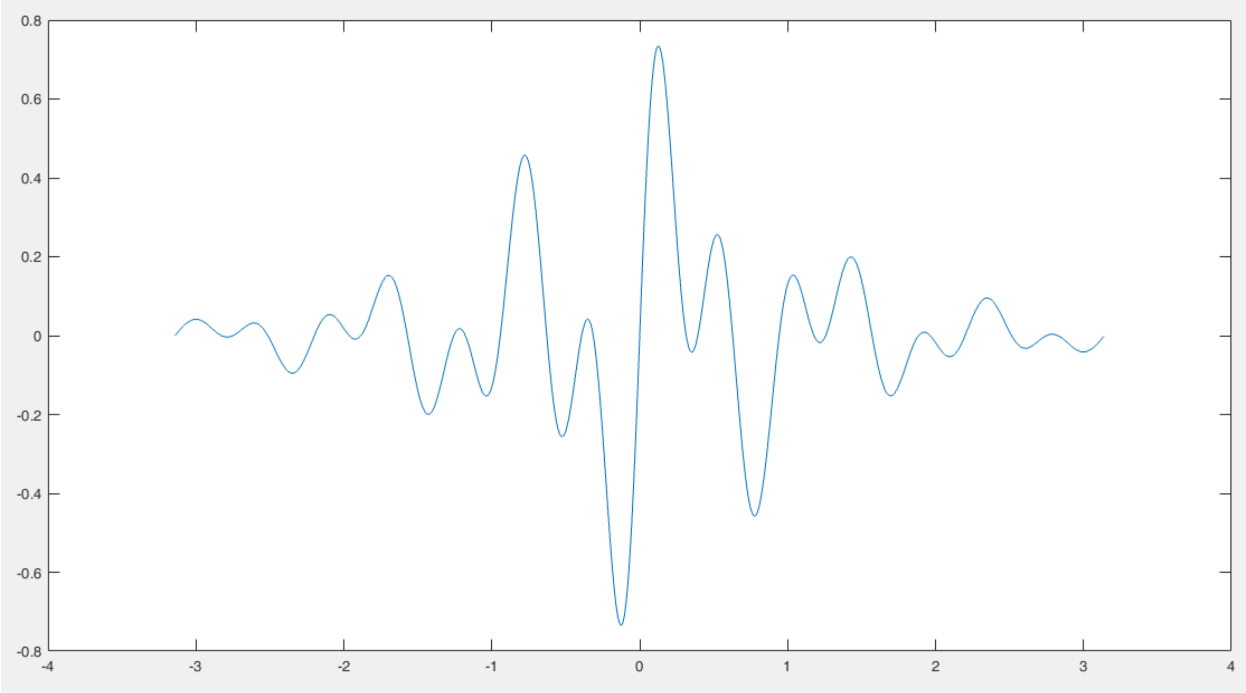
- Useful for larger, more structured optimization problems

- Sample functions (see `help` for more info)
  - » `linprog`
    - ➢ linear programming using interior point methods
  - » `quadprog`
    - ➢ quadratic programming solver
  - » `fmincon`
    - ➢ constrained nonlinear optimization

Exercise3-4: Min-Finding

- Find the minimum of the function $f(x) = \cos(4x)\sin(10x)e^{-|x|}$ over the range −π to π. Use `fminbnd`.
- Plot the function on this range to check that this is the minimum.

```
>> func=@(x)(cos(4*x).*sin(10*x).*exp(-abs(x)));
>> minx=fminbnd(func,-pi,pi)
>> func(minx)
>> x=-pi:0.01:pi;
>> plot(x,func(x));
```



Numerical Issues

- Many techniques in this lecture use floating point numbers
- **This is an approximation!**

- Examples:

```
» sin(pi) = ?
» sin(2 * pi) = ?
» sin(10e16 * pi) = ?
```

  - ➢ Both sin and pi are approximations!

```
» A = (10e13)*ones(10) + rand(10)
```

  - ➢ A is nearly singular, poorly conditioned (see `cond(A)`)

```
» inv(A)*A = ?
```

- MATLAB knows no fear!

- Give it a function, it optimizes / differentiates / integrates
  - ➢ That's great! It's so powerful!

- Numerical techniques are powerful **but** not magic

- **Beware of overtrusting the solution!**
  - ➢ You will get an answer, but it may not be what you want

- Analytical forms may give more intuition
  - ➢ Symbolic Math Toolbox

Outline

(1) Linear Algebra

(2)Polynomials

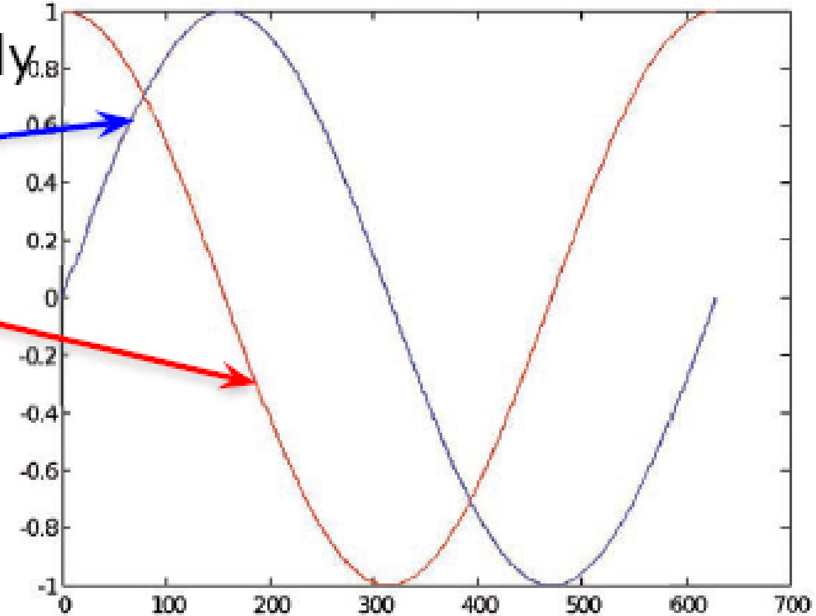
(3)Optimization

(4) Differentiation/Integration

(5) Differential Equations

Numerical Differentiation

- MATLAB can 'differentiate' numerically
  - » `x=0:0.01:2*pi;`
  - » `y=sin(x);`
  - » `dydx=diff(y)./diff(x);`
    - ➢ diff computes the first difference



- Can also operate on matrices
  - » `mat=[1 3 5;4 8 6];`
  - » `dm=diff(mat,1,2)`
    - ➢ first difference of mat along the 2nd dimension, dm=[2 2;4 -2]
    - ➢ The opposite of `diff` is the cumulative sum `cumsum`

- 2D gradient
  - » `[dx,dy]=gradient(mat);`

Numerical Integration

- MATLAB contains common integration methods

- Adaptive Simpson's quadrature (input is a function)
  - » `q=quad('myFun',0,10)`
    - q is the integral of the function `myFun` from 0 to 10
  - » `q2=quad(@(x) sin(x).*x,0,pi)`
    - q2 is the integral of `sin(x).*x` from 0 to pi
- Trapezoidal rule (input is a vector)
  - » `x=0:0.01:pi;`
  - » `z=trapz(x,sin(x))`
    - z is the integral of sin(x) from 0 to pi
  - » `z2=trapz(x,sqrt(exp(x))./x)`
    - z2 is the integral of $\sqrt{e^x}/x$ from 0 to pi

多重数值积分

- 二重积分

Q2=dblquad(func, xmin, xmax, ymin, ymax)

- 三重积分

Q3=triplequad(func, xmin, xmax, ymin, ymax, zmin, zmax)

Outline

(1) Linear Algebra

(2)Polynomials

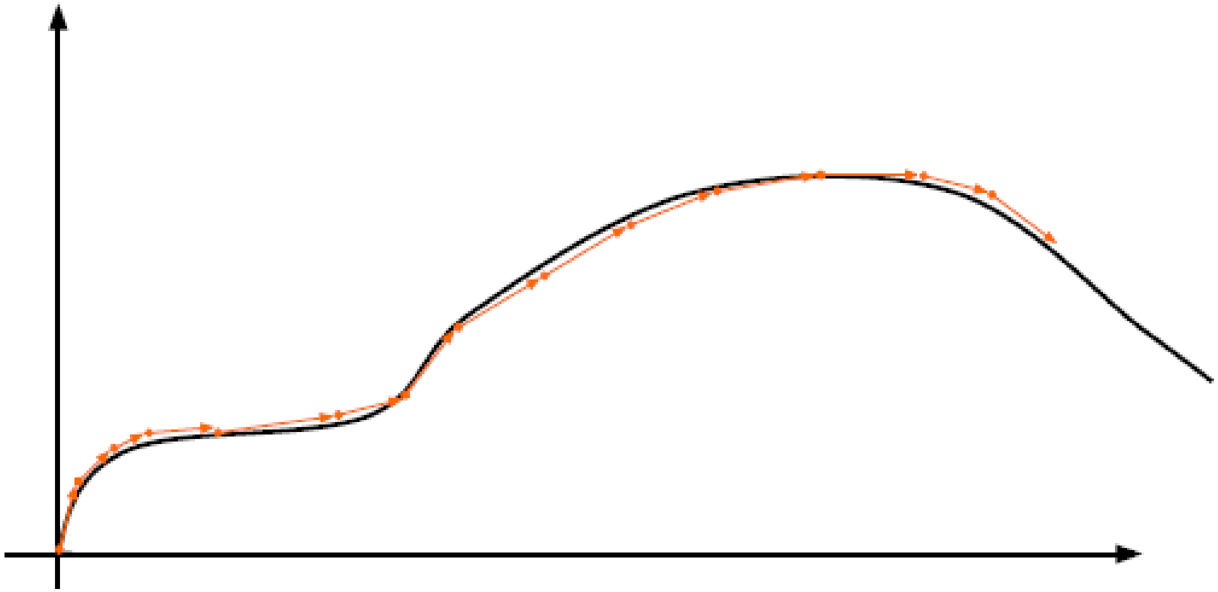
(3)Optimization

(4) Differentiation/Integration

(5) Differential Equations

ODE Solvers: Method

- Given a differential equation, the solution can be found by integration:



  - ➢ Evaluate the derivative at a point and approximate by straight line
  - ➢ Errors accumulate!
  - ➢ Variable timestep can decrease the number of iterations

ODE Solvers: MATLAB

- MATLAB contains implementations of common ODE solvers

- Using the correct ODE solver can save you lots of time and give more accurate results
  - `ode23`
    - Low-order solver. Use when integrating over small intervals or when accuracy is less important than speed
  - `ode45`
    - High order (Runge-Kutta) solver. High accuracy and reasonable speed. Most commonly used.
  - `ode15s`
    - Stiff ODE solver (Gear's algorithm), use when the diff eq's have time constants that vary by orders of magnitude

ODE Solvers: Standard Syntax

- To use standard options and variable time step

```
» [t,y]=ode45('myODE',[0,10],[1;0])
```

ODE integrator:
23, 45, 15s

ODE function

Time range

Initial conditions

- Inputs:
  - ➢ ODE function name (or anonymous function). This function should take inputs (t,y), and returns dy/dt
  - ➢ Time interval: 2-element vector with initial and final time
  - ➢ Initial conditions: $y0$
  - ➢ Make sure all inputs are in the same (variable) order

- Outputs:
  - ➢ t contains the time points
  - ➢ y contains the corresponding values of the variables

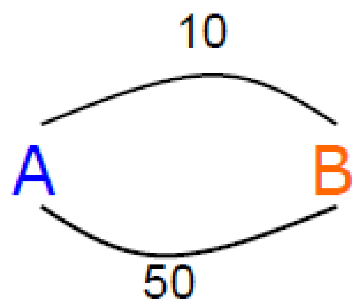
ODE Function

- The ODE function must return the value of the derivative at a given time and function value

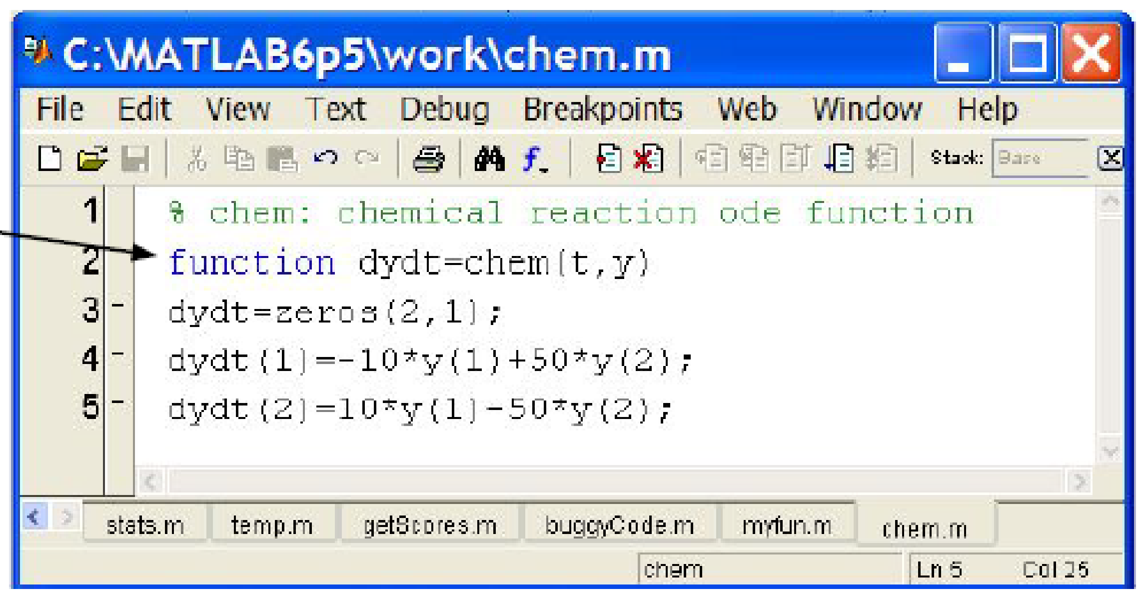
- Example: chemical reaction
  - ➢ Two equations

$$\frac{dA}{dt} = -10A + 50B$$

$$\frac{dB}{dt} = 10A - 50B$$



  - ➢ ODE file:
    - – y has [A;B]
    - – dydt has [dA/dt;dB/dt]



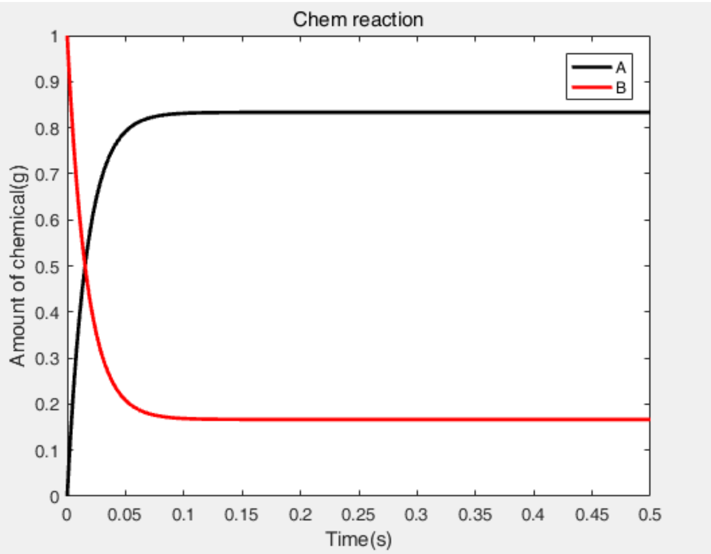
ODE Function: viewing results

- To solve and plot the ODEs on the previous slide:

```
» [t,y]=ode45('chem',[0 0.5],[0 1]);
```

   ➢ assumes that only chemical B exists initially

```
» plot(t,y(:,1),'k','LineWidth',1.5);
» hold on;
» plot(t,y(:,2),'r','LineWidth',1.5);
» legend('A','B');
» xlabel('Time (s)');
» ylabel('Amount of chemical (g)');
» title('Chem reaction');
```



Higher Order Equations

- Must make into a system of first-order equations to use ODE solvers
- Nonlinear is OK!
- Pendulum（摆锤） example:

$$\ddot{\theta} + \frac{g}{L} sin(\theta) = 0$$

$$\ddot{\theta} = -\frac{g}{L} sin(\theta)$$

$$let \quad \dot{\theta} = \gamma$$

$$\dot{\gamma} = -\frac{g}{L} sin(\theta)$$

$$\vec{y} = \begin{bmatrix} \theta \\ \gamma \end{bmatrix}$$

$$\frac{d\vec{y}}{dt} = \begin{bmatrix} \dot{\theta} \\ \dot{\gamma} \end{bmatrix}$$

```
function dydt=pendulum(t,y)
L=1;
theta = y(1);
gamma = y(2);

dtheta = gamma;
dgamma = -(9.8/L)*sin(theta);

dydt = zeros(2,1);

dydt(1)=dtheta;
dydt(2)=dgamma;
```

Plotting the Output

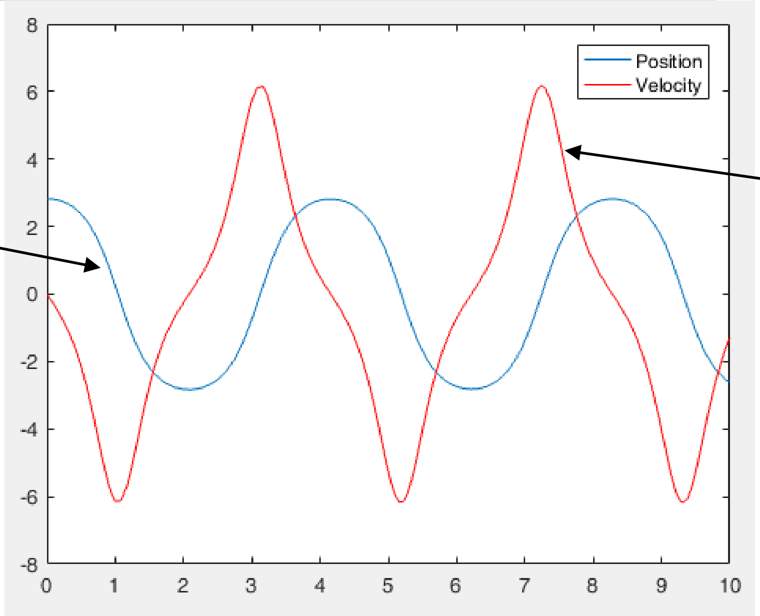
- We can solve for the position and velocity of the pendulum:

```
>> [t,y]=ode45('pendulum',[0 10],[0.9*pi 0]);
```

➢ assume pendulum is almost static

```
>> plot(t,y(:,1));
>> hold on;
>> plot(t,y(:,2),'r');
>> legend('Position','Velocity');
```

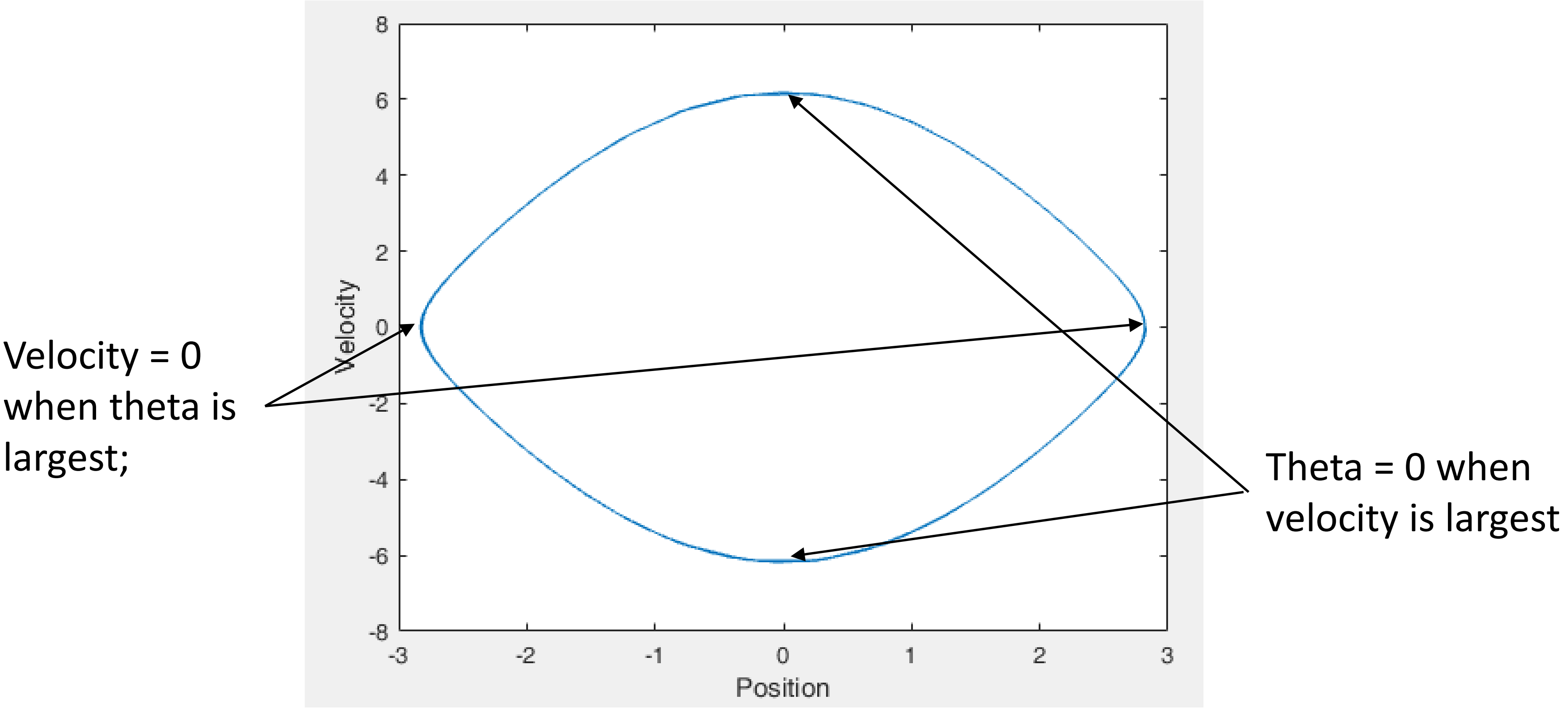
Position in terms of angle(rad)



Velocity(m/s)

Plotting the Output

- Or we can plot in the phase plane:
- The phase plane is just a plot of one variable versus the other:



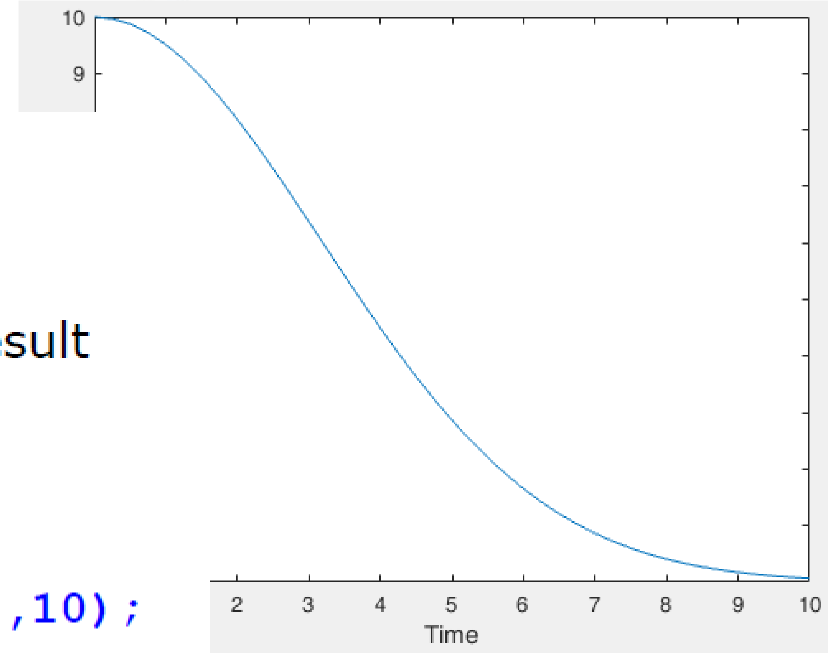
ODE Solvers: Custom Options

- MATLAB's ODE solvers use a variable timestep
- Sometimes a fixed timestep is desirable
  - » `[t,y]=ode45('chem',[0:0.001:0.5],[0 1]);`
    - ➢ Specify timestep by giving a vector of (increasing) times
    - ➢ The function value will be returned at the specified points

- You can customize the error tolerances using odeset
  - » `options=odeset('RelTol',1e-6,'AbsTol',1e-10);`
  - » `[t,y]=ode45('chem',[0 0.5],[0 1],options);`
    - ➢ This guarantees that the error at each step is less than `RelTol` times the value at that step, and less than `AbsTol`
    - ➢ Decreasing error tolerance can considerably slow the solver
    - ➢ See doc odeset for a list of options you can customize

Exercise3-5: ODE

- Use `ode45` to solve for $y(t)$ on the range t=[0 10], with initial condition $y(0)=10$ and $dy/dt=-t\,y/10$
- Plot the result.



- Make the following function

```
» function dydt=odefun(t,y)
» dydt=-t*y/10;
```

- Integrate the ODE function and plot the result

```
» [t,y]=ode45('odefun',[0 10],10);
```

- Alternatively, use an anonymous function

```
» [t,y]=ode45(@(t,y) -t*y/10,[0 10],10);
```

- Plot the result

```
» plot(t,y);xlabel('Time');ylabel('y(t)');
```

Exercise3-6: ODE

范德波尔（荷兰）方程：描述电子电路中三极管的震荡效应

$$\ddot{y} - \mu(1 - y^2)\dot{y} + y = 0 \qquad \mu\text{为常数}$$

用ODE方法求解$\mu = 0$、$\mu = 1$时的范德波尔方程（y~t），以及相位关系（y'~y）。

设初始条件$y(0) = 2, y'(0)$=0.

$\mu = 10$