CS@CU

COMS W3101-2

Programming Languages: MATLAB

Lecture 5

Spring 2010

Instructor: Michele Merler

http://www1.cs.columbia.edu/~mmerler/comsw3101-2.html

Quiz Review

- 1. Generate the pythagoric table (do not insert the values manually)

1	2	3	4	5	6	7	8	9	10	11	12
2	4	6	8	10	12	14	16	18	20	22	24
3	6	9	12	15	18	21	24	27	30	33	36
4	8	12	16	20	24	28	32	36	40	44	48
5	10	15	20	25	30	35	40	45	50	55	60
6	12	18	24	30	36	42	48	54	60	66	72
7	14	21	28	35	42	49	56	63	70	77	84
8	16	24	32	40	48	56	64	72	80	88	96
9	18	27	36	45	54	63	72	81	90	99	108
10	20	30	40	50	60	70	80	90	100	110	120
11	22	33	44	55	66	77	88	99	110	121	132
12	24	36	48	60	72	84	96	108	120	132	144

Quiz Review

- 1. Generate the pythagoric table (do not insert the values manually)
  - `r = 1:12;`
  - `PT = r'*r;`

Quiz Review

- 2. Compute the series $S = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right) \to \pi$
    - Using 10 elements (S_{10})
    - Using 100 elements (S_{100})
    - Compare the S_{10} and S_{100} with the value of the limit. Which one is closer to the limit? Display the answer to command window

Quiz Review

- 2. Compute the series $S = 4\cdot\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots\right) \to \pi$
  - Using 10 elements (S_{10}) $\mapsto$ N = 9
  - Using 100 elements (S_{100}) $\mapsto$ N = 99
  - Compare the S_{10} and S_{100} with the value of the limit. Which one is closer to the limit? Display the answer to command window

$$S = 4\cdot\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots\right) \to \pi$$

$$= 4\cdot\sum_{n=0}^{N=\infty}(-1)^n\left(\frac{1}{2n+1}\right)$$

Quiz Review

- 2. Compute the series $S = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right) \rightarrow \pi$
  - Using 10 elements (S_{10}) $\mapsto$ N = 9
  - Using 100 elements (S_{100}) $\mapsto$ N = 99
  - Compare the S_{10} and S_{100} with the value of the limit. Which one is closer to the limit? Display the answer to command window

$$S = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right) = 4 \cdot \sum_{n=0}^{N=\infty} (-1)^n \left(\frac{1}{2n+1}\right) \rightarrow \pi$$

  - `fracVec = ones(1,100)./[1:2:199];`

Quiz Review

- 2. Compute the series $S = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right) \to \pi$
  - Using 10 elements (S_{10}) $\mapsto$ N = 9
  - Using 100 elements (S_{100}) $\mapsto$ N = 99
  - Compare the S_{10} and S_{100} with the value of the limit. Which one is closer to the limit? Display the answer to command window

$$S = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right) = 4 \cdot \sum_{n=0}^{N=\infty} (-1)^n \left(\frac{1}{2n+1}\right) \to \pi$$

  - `fracVec = ones(1,100)./[1:2:199];`
  - `signs =(-1).^[0:99];`

Quiz Review

- 2. Compute the series $S = 4\cdot\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots\right)\rightarrow\pi$
  - Using 10 elements (S_{10}) $\mapsto$ N = 9
  - Using 100 elements (S_{100}) $\mapsto$ N = 99
  - Compare the S_{10} and S_{100} with the value of the limit. Which one is closer to the limit? Display the answer to command window

$$S = 4\cdot\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\cdots\right) = 4\cdot\sum_{n=0}^{N=\infty}(-1)^n\left(\frac{1}{2n+1}\right)\rightarrow\pi$$

```
fracVec = ones(1,100)./[1:2:199];
signs =(-1).^[0:99];

S10 = 4 * sum(fracVec(1:10).*signs(1:10));
diff10 = pi - S10;

S100 = 4 * sum(fracVec.*signs);
diff100 = pi - S100;
```

Quiz Review

- 2. Compute the series $S = 4\cdot\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right) \to \pi$
  - Using 10 elements (S_{10}) $\mapsto$ N = 9
  - Using 100 elements (S_{100}) $\mapsto$ N = 99
  - Compare the S_{10} and S_{100} with the value of the limit. Which one is closer to the limit? Display the answer to command window

$$S = 4\cdot\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right) = 4\cdot\sum_{n=0}^{N=\infty}(-1)^n\left(\frac{1}{2n+1}\right) \to \pi$$

```
fracVec = ones(1,100)./[1:2:199];
signs =(-1).^[0:99];

S10 = 4 * sum(fracVec(1:10).*signs(1:10));
diff10 = pi - S10;

S100 = 4 * sum(fracVec.*signs);
diff100 = pi - S100;

if(diff10>diff100)
    disp('S100 is closer to the limit')
else
    disp('S100 is closer to the limit')
end
```

Quiz Review

- 2. Compute the series $S = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right) \to \pi$
  - Using 10 elements (S_{10})
  - Using 100 elements (S_{100})
  - Compare the S_{10} and S_{100} with the value of the limit. Which one is closer to the limit? Display the answer to command window

```
varSign = -1;
S10 = 4;
for in=3:2:19
    S10 = S10 + 4 * varSign * 1/in;
    varSign = -1 * varSign;
end
```

Quiz Review

- 2. Compute the series $S = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \cdots\right) \to \pi$
  - Using 10 elements (S_{10})
  - Using 100 elements (S_{100})
  - Compare the S_{10} and S_{100} with the value of the limit. Which one is closer to the limit? Display the answer to command window

```
varSign = -1;
S100 = 4;
for in=3:2:199
    S100 = S100 + 4 * varSign * 1/in;
    varSign = -1 * varSign;
end
```

Quiz Review

- 3. Compute sin(x) and cos(x) in the interval x = [0, 2π] (choose the number of elements in x so that the functions can be plotted smoothly). You have to plot 3 graphs in the same figure:

  - Plot sin(x) in blue in the specified interval, label the axes, assign a title to the figure
  - Plot cos(x) in red in the specified interval, label the axes, assign a title to the figure and display the legend
  - Compute the maximum between the two functions at each point in x, then plot the function called maxSinCos(x) , with line width 2 and color magenta, together with sin(x) and cos(x) in the specified interval. Label the axes, assign a title to the figure and display the legend

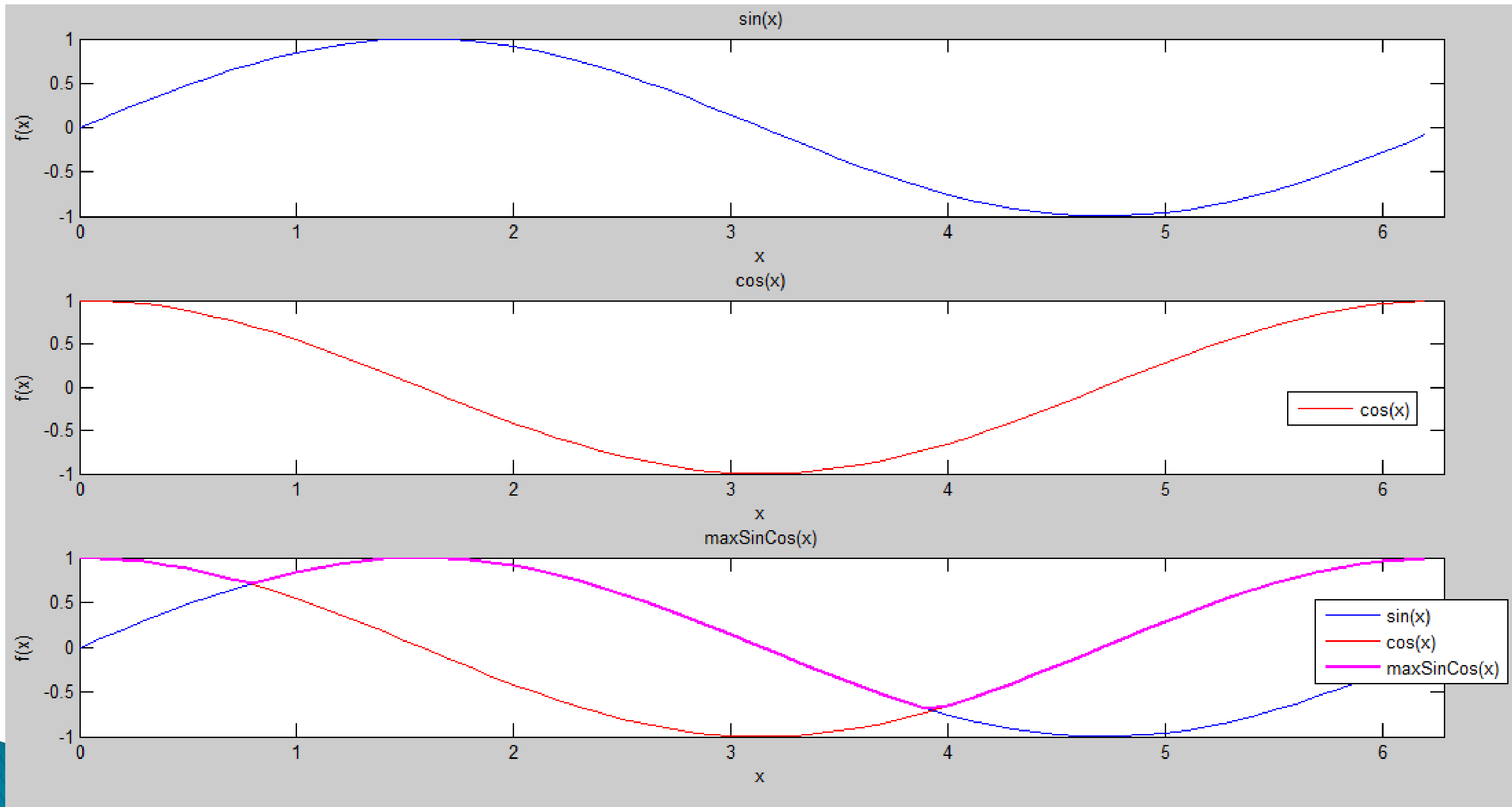
Quiz Review

- 3. Compute sin(x) and cos(x) in the interval x = [0, 2π] (choose the number of elements in x so that the functions can be plotted smoothly). You have to plot 3 graphs in the same figure:
  - Plot sin(x) in blue in the specified interval, label the axes, assign a title to the figure
  - Plot cos(x) in red in the specified interval, label the axes, assign a title to the figure and display the legend
  - Compute the maximum between the two functions at each point in x, then plot the function called maxSinCos(x) , with line width 2 and color magenta, together with sin(x) and cos(x) in the specified interval. Label the axes, assign a title to the figure and display the legend

```
x = [0:0.1:2*pi];

sX = sin(x);
cX = cos(x);

figure
subplot(3,1,1)
plot(x,sX);
xlim([0 2*pi])
title('sin(x)');
xlabel('x'); ylabel('f(x)')

subplot(3,1,2)
plot(x,cX,'r');
xlim([0 2*pi])
title('cos(x)');
xlabel('x'); ylabel('f(x)'); legend('cos(x)');

maxSinCosX = max(sin(x),cos(x));

subplot(3,1,3)
plot(x,sX);
hold on
plot(x,cX,'r');
plot(x,maxSinCosX,'m','linewidth',2);
hold off
xlim([0 2*pi])
title('maxSinCos(x)');
xlabel('x'); ylabel('f(x)')
legend('sin(x)','cos(x)','maxSinCos(x)')
```

Quiz Review

3. Output

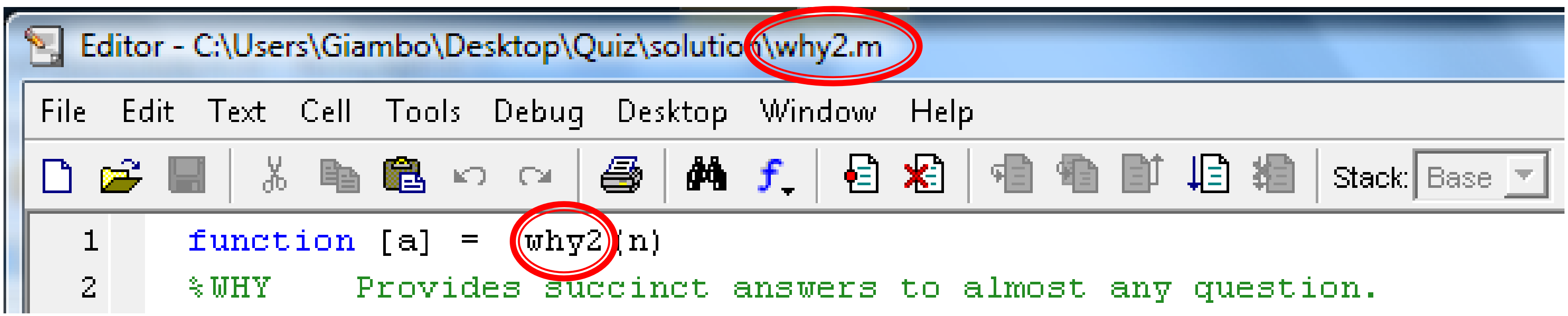


Quiz Review

- 4. Playing with the ‘why’ function.
  - Open the ‘why’ function, copy the full code into a file named **why2.m** and save it. Modify line 1 of **why2.m** so that the function returns the variable ‘a’.
  - Write function called ‘countWhy’ that takes a *filename.txt* as input. In the function, implement a loop which invokes the ‘why2’ command at every iteration. Stop the loop when the message returned by ‘why2’ is a repetition of a message already seen. Write to *filename.txt* all the ‘why2’ messages seen, and display to command window the number of iterations achieved.

Quiz Review

- 4. Playing with the ‘why’ function.
  - Open the ‘why’ function, copy the full code into a file named **why2.m** and save it. Modify line 1 of **why2.m** so that the function returns the variable ‘a’.



Quiz Review

- Write function called ‘countWhy’ that takes a *filename.txt* as input. In the function, implement a loop which invokes the ‘why2’ command at every iteration. Stop the loop when the message returned by ‘why2’ is a repetition of a message already seen. Write to *filename.txt* all the ‘why2’ messages seen, and display to command window the number of iterations achieved.

```
numTimes = countWhy('whyAnswers.txt');
disp(numTimes)
```

Quiz Review

```
function [count] = countWhy(filename)

fid = fopen(filename,'w');
count = 0;
in = 1;
control = 0;

while 1
    v{in} = why2;
    fprintf(fid,'%s\n',v{in});

    control = 0;
    if(in>1)
        for in2 = 1:in-1
            if(strcmp(v{in},v{in2}))
                control=1;
                break
            end
        end
    end
    if(control==1)
            break
        else
            in = in+1;
        end
end

fclose(fid)

count = in
```

Functions Handles

- Handle = another type of variables in MATLAB
- An identifier for a function
  - `h = @sin;`
  - `handlename = @functionname;`
  - `x = h(pi/2);`
  - `x = sin(pi/2);`

- Can be inserted in structs and cells, not arrays
  - `S.a = @sin; S.b = @cos; S.c = @tan;`
  - `C = {@sin, @cos, @tan};`
  - ~~`A = [@sin, @cos, @tan];`~~

Functions Handles

- Handle to anonymous function
  - `h = @(x,y) x + y;`
  - `handlename = @(v1,v2,…) body;`

Example 1

```
h = @(x,y) x + y;
x = h(1,2);
```

Example 2

```
h = @(fun,x,y) fun(x) + y;
val = h(@sin, pi/2, 3);
```

Example 3

```
function [val] = myfun(fun,x,y)
val = fun(x) + y;

myfun(@sin, pi/2, 3);
```

Functions Handles

- Handle to anonymous function
  - `h = @(x,y) x + y;`
  - *`handlename`* `= @(`*`v1,v2,…`*`)` *`body`*`;`

Example 1

- `h = @(x,y) x + y;`
- `x = h(1,2);`

Note that the function does not have a specified name, but it is identified only trough the function handle ***h*** !

Example 2

- `h = @(fun,x,y) fun(x) + y;`
- `val = h(@sin, pi/2, 3);`

In these cases the variable ***fun*** is a function handle itself!

Example 3

```
function [val] = myfun(fun,x,y)
val = fun(x) + y;
```

- `myfun(@sin, pi/2, 3);`

Find zeros of functions

- `fzero()` – finds a zero of a function f(x) the closest possible to a specified point x1

  - `xZ = fzero(@sin,x1);`
  - *`res`* `= fzero(`*`funHandle,startPoint`*`);`

  Example1: find the zero of the function sin(x) closest to π/3

```
x = [-pi:0.1:pi];
x1 = pi/3;
xZero = fzero(@sin,x1);
plot(x, sin(x));
hold on
plot(x, 0.*x,'k');
plot(x1, sin(x1),'g*');
plot(xZero, sin(xZero),'rd');
hold off
xlim([-pi pi]);
```

Find zeros of functions

- `fzero()` – finds a zero of a function f(x) inside a specified range x1
  - `xZ = fzero(@sin,x1);`
  - `res = fzero(funHandle,range);`

Example2: find the zero of the function sin(x) between −π/3 and π/3

```
x = [-pi:0.1:pi];
x1 = [-pi/3 pi/3];
xZero = fzero(@sin,x1);
plot(x, sin(x));
hold on
plot(x, 0.*x,'k');
plot(x1, sin(x1),'g*');
plot(xZero, sin(xZero),'rd');
hold off
xlim([-pi pi]);
```

Find zeros of functions

- `fzero()` – finds a zero of a function f(x) inside a specified range x1
  - `xZ = fzero(@sin,x1);`
  - *`res`* `= fzero(`*`funHandle,range`*`);`

Example2: find the zero of the function sin(x) between $-\pi/3$ and $\pi/3$

```
x = [-pi:0.1:pi];
x1 = [-pi/3 pi/3];
xZero = fzero(@sin,x1);
plot(x, sin(x));
hold on
plot(x, 0.*x,'k');
plot(x1, sin(x1),'g*');
plot(xZero, sin(xZero),'rd');
hold off
xlim([-pi pi]);
```

If x1 is a range, the function *fun()* referenced by *funHandle* MUST change sign between x1(1) and x1(2) !

Sign(fun(x1(1))) ~= Sign(fun(x1(2))) !!!

Find roots of polynomials

- `roots()` – returns a vector whose elements are the roots of a polynomial

Example $x^2 - 5x + 6 = 0 \Leftrightarrow (x-3)(x-2) = 0$

- `p = [1 -5 6];`
- `res = roots(p);`

  `[3;2]`

- `res = roots(vec);`

Solving Equations

- Suppose we have the following system of linear equations:

$$\begin{cases} x + y - 2z = 4 \\ 3x + 5y + z = -2 \\ -2x + 3y - 10z = 7 \end{cases}$$

  - We want to solve it and find the values of x, y and z
  - We can store the equations in the following way:
    - $AX = b$, with

$$A = \begin{bmatrix} 1 & 1 & -2 \\ 3 & 5 & 1 \\ -2 & 3 & -10 \end{bmatrix} \quad X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad b = \begin{bmatrix} 4 \\ -2 \\ 7 \end{bmatrix}$$

Solving Equations

- Suppose we have the following system of linear equations:

$$\begin{cases} x + y - 2z = 4 \\ 3x + 5y + z = -2 \\ -2x + 3y - 10z = 7 \end{cases}$$

  - We know from linear algebra the solution is
    - $X = \text{inv}(A) * b$
  - So with MATLAB we can solve it in 2 ways:
    - `X = inv(A) * b;`
    - `X = A \ b;`

Solving Equations

- Suppose we have the following system of linear equations:

$$\begin{cases} x + y - 2z = 4 \\ 3x + 5y + z = -2 \\ -2x + 3y - 10z = 7 \end{cases}$$

  - We know from linear algebra the solution is
    - $X = \text{inv}(A) * b$
  - So with MATLAB we can solve it in 2 ways:
    - `X = inv(A) * b;`
    - `X = A \ b;`

NOTE 1 : the \ operator works for square systems. For **rectangular systems** it gives the **least squares** solution.

NOTE 2 : we have to check if the system is **over** or **underdetermined**

Linear Algebra

- `rank()`
  - Computes the rank of a matrix (the number of linearly independent rows or columns)
  - `R = rank(M);`
- `det()`
  - Computes the determinant of a matrix, which must be square
  - NOTE: if determinant is nonzero, matrix is invertible
  - `d = det(M);`
- `trace()`
  - Computes the trace of a matrix (the sum of its diagonal elements)
  - `R = trace(M);`
- `inv()`
  - Computes the inverse of a matrix
  - `Ainv = inv(M)`

Computing the determinant: cofactor expansion

http://comp.uark.edu/~jjrencis/femur/Learning-Modules/Linear-Algebra/solving/determinant/cofactor_expansion.html

Matrix Decomposition

- Eigenvalues, Eigenvectors
- eig()
  - `[eigVect eigVal] = eig(M);`
  - NOTE: M must be square
  - RESULT: 2 matrices of same dimension as M,
    - a diagonal matrix eigVal whose diagonal elements are the eigenvalues of M
    - a matrix eigVec whose columns are the eigenvectors of M
  - eigVal*M = eigVal*eigVec

- Singular Value Decomposition
- svd()
  - `[U, S, V] = svd(M);`
  - NOTE: M does not have to be square
  - RESULT: a diagonal matrix S of the same dimension as M, with nonnegative diagonal elements in decreasing order, and unitary matrices U and V so that
  - M = U*S*V‘.

Differentiation

- `diff()` 1D
  - `x = [1:2:11];`
  - `diff(x)/2;`

$$f(x) \rightarrow \frac{df(x)}{dx}$$

- `gradient()` 2D
  - `x = 1:12;`
  - `M = x'*x`
  - `[dx dy] = gradient(M);`

$$f(x,y) \rightarrow \begin{matrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{matrix}$$

Integration 1

- Using the trapezoidal rule
  - `x = 0:0.01:pi;`
  - `intTX = trapz(x,sin(x));`

$$\text{int } X = \int_0^{\pi} \sin(x)\, dx$$

Integration 2

- Using recursive adaptive Simpson quadrature

$$\operatorname{int} X = \int_0^{\pi} \sin(x)\,dx$$

  - `intQX = quad(@sin,0,pi)`
  - `q = quad(@sin,0,pi,tol);`

Polynomial Fitting

- Fit an n^{th} degree polynomial to predefined data

- `polyfit()`
- `polyval()`

- Example (fit 2nd degree polynomial to noisy data)
  - `x = [-4:0.1:4];`
  - `y = x.^2;`
  - `yNoisy = y + randn(size(y));`
  - `plot(x,yNoisy,'.');`

  - `polY = polyfit(x,yNoisy,2);`

Polynomial Fitting

- Fit an n^{th} degree polynomial to predefined data

- polyfit()
- polyval()

- Example (fit 2nd degree polynomial to noisy data)

```
x = [-4:0.1:4];
y = x.^2;
yNoisy = y + randn(size(y));
plot(x,yNoisy,'.');

polY = polyfit(x,yNoisy,2);

hold on;
plot(x,polyval(polY,x),'r');
hold off;
```

Polynomial Fitting

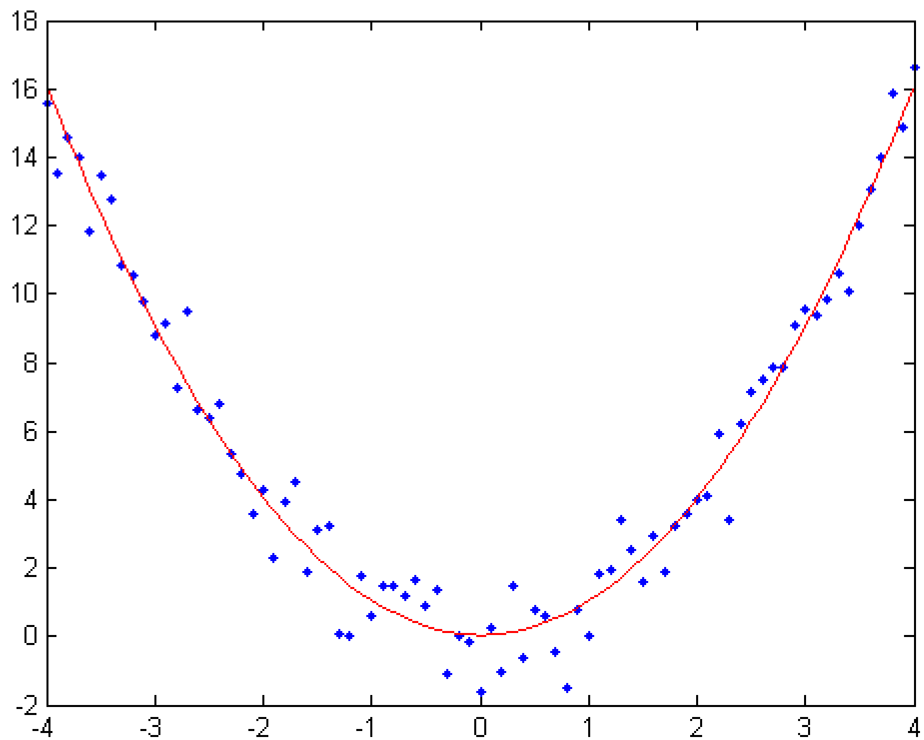
- Fit an n^{th} degree polynomial to predefined data

- `polyfit()`
- `polyval()`

- Example (fit 2nd de
  - `x = [-4:0.1:4];`
  - `y = x.^2;`
  - `yNoisy = y + randn`
  - `plot(x,yNoisy,'.')`

  - `polY = polyfit(x,y`

  - `hold on;`
  - `plot(x,polyval(pol`
  - `hold off;`



Exercises in class !

Exercises in class 1/2

```
bought = dlmread('groceries.txt',',',0,1);
fid = fopen('groceries.txt');
food = textscan(fid,'%s');
fclose(fid);
food = strtok(food{1},',');

M = eye(length(food),length(food));
for in1=1:length(food)
    for in2=1:length(food)
        if(in1 ~= in2)
            M(in1,in2) = ...
            coBought(food,bought,food{in1},food{in2}) / ...
            numBought(food,bought,food{in1});
        end
    end
end

imagesc(M);
colormap('hot');
colorbar
```

Exercises in class 1/2

```
function [num] = numBought(names,mat,index)

for in=1:length(names)
    if(strcmp(names{in},index))
        break
    end
end
num = sum(mat(in,:));
```

```
function [num] = coBought(names,mat,index1,index2)

for in=1:length(names)
    if(strcmp(names{in},index1))
        in1 = in;
    end
    if(strcmp(names{in},index2))
        in2 = in;
    end
end
num = sum( mat(in1,:).* mat(in2,:) );
```

Exercises in class 2/2

```
% series 1
n = 1:10^5;
S1 = cumsum( 2 ./ (n.^2 + 2.*n) );

figure
subplot(1,2,1)
plot(S1)

% series 2
S2 = cumsum((-1).^n .* n./(n+1));

subplot(1,2,2)
plot(S2)
```

Homeworks policy

- Due at beginning of class, no exceptions
- Put your code (.m files) and additional files in a single folder, name it *youruni*_hw_*X* and zip it
- Upload the zipped folder to CourseWorks
- Bring a printout of your code to class
- Good luck and have fun, it's the last one !