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

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

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

   ◦ \( r = 1:12; \)
   ◦ \( PT = r'*r; \)
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 \( (S_{10}) \)
- 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
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 ($S_{10}$) $\rightarrow N = 9$
- Using 100 elements ($S_{100}$) $\rightarrow N = 99$

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

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

$$= 4 \cdot \sum_{n=0}^{N=\infty} (-1)^n \left(\frac{1}{2n+1}\right)$$
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 (\( S_{10} \)) \( \rightarrow N = 9 \)
- Using 100 elements (\( S_{100} \)) \( \rightarrow N = 99 \)

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:

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

- \( \text{fracVec} = \text{ones}(1, 100) ./ [1:2:199] ; \)
2. Compute the series \( S = 4 \cdot \left( \frac{1}{3} - \frac{1}{5} + \frac{1}{7} - \frac{1}{9} + \frac{1}{11} + \cdots \right) \rightarrow \pi \)

- Using 10 elements \( (S_{10}) \rightarrow N = 9 \)
- Using 100 elements \( (S_{100}) \rightarrow N = 99 \)

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

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

- \( \text{fracVec} = \text{ones}(1,100)./\{1:2:199\}; \)
- \( \text{signs} = (-1).^[0:99]; \)
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 \( S_{10} \) \( \rightarrow \) \( N = 9 \)
- Using 100 elements \( S_{100} \) \( \rightarrow \) \( N = 99 \)

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

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

- \( \text{fracVec} = \text{ones}(1,100) /[1:2:199]; \)
- \( \text{signs} = (-1).^(0:99); \)
- \( S_{10} = 4 \cdot \text{sum}(\text{fracVec}(1:10) .* \text{signs}(1:10)); \)
- \( \text{diff}_{10} = \text{pi} - S_{10}; \)
- \( S_{100} = 4 \cdot \text{sum}(\text{fracVec} .* \text{signs}); \)
- \( \text{diff}_{100} = \text{pi} - S_{100}; \)
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 \( (S_{10}) \rightarrow N = 9 \)
- Using 100 elements \( (S_{100}) \rightarrow N = 99 \)

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

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

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

- \( S_{10} = 4 \times \text{sum} (\text{fracVec}(1:10).*\text{signs}(1:10)); \)
- \( \text{diff10} = \pi - S_{10}; \)

- \( S_{100} = 4 \times \text{sum} (\text{fracVec}.*\text{signs}); \)
- \( \text{diff100} = \pi - S_{100}; \)

- \( \text{if} (\text{diff10}>\text{diff100}) \)
  - \( \text{disp}(’S_{100} \text{ is closer to the limit’}) \)
- \( \text{else} \)
  - \( \text{disp}(’S_{100} \text{ is closer to the limit’}) \)
- \( \text{end} \)
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 (\( S_{10} \))
- 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.

- \( \text{varSign} = -1; \)
- \( S_{10} = 4; \)
- \( \text{for } \text{in}=3:2:19 \)
- \( \quad S_{10} = S_{10} + 4 \times \text{varSign} \times 1/\text{in}; \)
- \( \quad \text{varSign} = -1 \times \text{varSign}; \)
- \( \text{end} \)
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 \((S_{10})\)
- 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

- \(\text{varSign} = -1;\)
- \(S_{100} = 4;\)
- \(\text{for } \text{in}=3:2:199\)
- \(S_{100} = S_{100} + 4 \times \text{varSign} \times 1/\text{in};\)
- \(\text{varSign} = -1 \times \text{varSign};\)
- \(\text{end}\)
3. Compute \( \sin(x) \) and \( \cos(x) \) in the interval \( x = [0, 2\pi] \) (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 \( \text{maxSinCos}(x) \) , 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
3. Compute $\sin(x)$ and $\cos(x)$ in the interval $x = [0, 2\pi]$ (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

```matlab
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)');
```
3. Output
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.
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’.

```matlab
function [a] = why2(n)

% WHY Provides succinct answers to almost any question.
```
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)
```
function [count] = countWhy(filename)

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

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

  - \( h = @\text{sin}; \)
  - \( \text{handle name} = @\text{function name}; \)
  
  - \( x = h(\pi/2); \)
  - \( x = \text{sin}(\pi/2); \)

- Can be inserted in structs and cells, **not arrays**
  
  - \( S.a = @\text{sin}; S.b = @\text{cos}; S.c = @\text{tan}; \)
  - \( C = \{@\text{sin}, @\text{cos}, @\text{tan}\}; \)
  
  - \( A = [@\text{sin}, @\text{cos}, @\text{tan}]; \)
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
- \( h = @(\text{fun},x,y) \, \text{fun}(x) + y; \)
- \( \text{val} = h(@\text{sin}, \, \pi/2, \, 3); \)

Example 3
- function [val] = myfun(fun,x,y)
- \( \text{val} = \text{fun}(x) + y; \)
- \( \text{myfun}(\@\text{sin}, \, \pi/2, \, 3); \)
Functions Handles

- Handle to anonymous function

  o \( h = @(x,y) \ x + y \);  
  o \( handlename = @(v1,v2,...) \ body; \)

Example 1

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

Example 2

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

Example 3

  o \( function \ [val] = myfun(fun,x,y) \)
  o \( val = fun(x) + y \);
  o \( myfun(@sin, \pi/2, 3) \);

Note that the function does not have a specified name, but it is identified only through the function handle \( h \)!

In these cases the variable \( fun \) is a function handle itself!
Find zeros of functions

- `fzero()` - finds a zero of a function $f(x)$ the closest possible to a specified point $x_1$

  - $x_Z = \text{fzero}(@\sin,x_1)$;  
  - $\text{res} = \text{fzero}(\text{funHandle},\text{startPoint})$;

Example 1: find the zero of the function $\sin(x)$ closest to $\pi/3$

- $x = [-\pi:0.1:\pi]$;
- $x_1 = \pi/3$;
- $x\text{Zero} = \text{fzero}(\text{@sin},x_1)$;
- plot($x$, $\sin(x)$);
- hold on
- plot($x$, $0.*x$,'k');
- plot($x_1$, $\sin(x_1)$,'g*');
- plot($x\text{Zero}$, $\sin(x\text{Zero})$,'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);`

Example 2: 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]);`
Find zeros of functions

- `fzero()` – finds a zero of a function $f(x)$ inside a specified range $x1$
  
  \[
  xZ = \text{fzero}(@\sin, x1); \quad res = \text{fzero}(\text{funHandle}, \text{range});
  \]

Example 2: find the zero of the function $\sin(x)$ between $-\pi/3$ and $\pi/3$

\[
\begin{align*}
  x &= [-\pi:0.1:\pi]; \\
  x1 &= [-\pi/3 \, \pi/3]; \\
  xZero &= \text{fzero}(@\sin, x1); \\
  \text{plot}(x, \sin(x)); \\
  \text{hold on} \\
  \text{plot}(x, 0.*x, 'k'); \\
  \text{plot}(x1, \sin(x1), 'g*'); \\
  \text{plot}(xZero, \sin(xZero), 'rd'); \\
  \text{hold off} \\
  \text{xlim}([-\pi \, \pi]);
\end{align*}
\]

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

Sign(fun(x1(1))) $\sim=\text{Sign}(\text{fun}(x1(2)))$ !!!
Find roots of polynomials

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

Example: \[ x^2 - 5x + 6 = 0 \iff (x - 3)(x - 2) = 0 \]

- \( p = [1, -5, 6]; \)
- \( res = \text{roots}(vec); \)
- \( res = \text{roots}(p); \)
- \([3; 2]\)
Suppose we have the following system of linear equations:

\[
\begin{align*}
  x + y - 2z &= 4 \\
  3x + 5y + z &= -2 \\
  -2x + 3y - 10z &= 7
\end{align*}
\]

- We want to solve it and find the values of \(x, y\) and \(z\)
- We can store the equations in the following way:

\[
AX = b, \quad \text{with}
\]

\[
A = \begin{bmatrix}
  1 & 1 & -2 \\
  3 & 5 & 1 \\
  -2 & 3 & -10
\end{bmatrix}, \quad
X = \begin{bmatrix}
  x \\
  y \\
  z
\end{bmatrix}, \quad
b = \begin{bmatrix}
  4 \\
  -2 \\
  7
\end{bmatrix}
\]
Suppose we have the following system of linear equations:

\[
\begin{align*}
  x + y - 2z &= 4 \\
  3x + 5y + z &= -2 \\
  -2x + 3y - 10z &= 7
\end{align*}
\]

We know from linear algebra the solution is

\[X = \text{inv}(A) \times b\]

So with MATLAB we can solve it in 2 ways:

\[X = \text{inv}(A) \times b;\]
\[X = A \backslash b;\]
Solving Equations

- Suppose we have the following system of linear equations:
  \[
  \begin{align*}
  x + y - 2z &= 4 \\
  3x + 5y + z &= -2 \\
  -2x + 3y - 10z &= 7
  \end{align*}
  \]
- We know from linear algebra the solution is
  \[
  X = \text{inv}(A) \times b
  \]
- So with MATLAB we can solve it in 2 ways:
  \[
  \begin{align*}
  X &= \text{inv}(A) \times b \\
  X &= A \backslash b
  \end{align*}
  \]

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)
  
  \[ R = \text{rank}(M); \]

- **`det()`**
  - Computes the determinant of a matrix, which must be square
  - NOTE: if determinant is nonzero, matrix is invertible

  \[ d = \text{det}(M); \]

- **`trace()`**
  - Computes the trace of a matrix (the sum of its diagonal elements)

  \[ R = \text{trace}(M); \]

- **`inv()`**
  - Computes the inverse of a matrix

  \[ A_{\text{inv}} = \text{inv}(M) \]
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()**
    - \([\text{eigVect} \ \text{eigVal}] = \text{eig}(M);\)
    - **NOTE:** \(M\) must be square
    - **RESULT:** 2 matrices of same dimension as \(M\),
      - a diagonal matrix \(\text{eigVal}\) whose diagonal elements are the eigenvalues of \(M\)
      - a matrix \(\text{eigVec}\) whose columns are the eigenvectors of \(M\)
    - \(\text{eigVal}^*M = \text{eigVal}^*\text{eigVec}\)

- **Singular Value Decomposition**
  - **svd()**
    - \([U, \ S, \ V] = \text{svd}(M);\)
    - **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**
  - \( x = [1:2:11]; \)
  - \( \text{diff}(x)/2; \)

- **gradient() 2D**
  - \( x = 1:12; \)
  - \( M = x'*x \)
  - \( [dx \ dy] = \text{gradient}(M); \)
Using the trapezoidal rule

\[ x = 0:0.01:\pi; \]
\[ \text{intTX} = \text{trapz}(x, \sin(x)); \]

\[
\int_{0}^{\pi} \sin(x) \, dx
\]
Using recursive adaptive Simpson quadrature

\[ \int_0^\pi \sin(x) \, dx \]

\[ \text{intQX} = \text{quad}(@\sin,0,\pi) \]

\[ q = \text{quad}(@\sin,0,\pi,\text{tol}); \]
Fit an $n^{th}$ degree polynomial to predefined data

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

Example (fit 2nd degree polynomial to noisy data)

- $x = [-4:0.1:4]$;
- $y = x.^2$;
- $y_{Noisy} = y + \text{randn(size}(y))$;
- `plot(x,y_{Noisy},'.')`;

- `polY = polyfit(x,y_{Noisy},2);`
Polynomial Fitting

- Fit an $n^{th}$ degree polynomial to predefined data
  
  - `polyfit()`
  - `polyval()`

Example (fit 2nd degree polynomial to noisy data)

- $x = [-4:0.1:4]$;
- $y = x.^2$;
- $y\text{Noisy} = y + \text{randn(size}(y))$;
- `plot(x,y\text{Noisy},'.'`);

- `polY = polyfit(x,y\text{Noisy},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 degree polynomial to noisy data)
  - $x = [-4:0.1:4]$;
  - $y = x.^2$;
  - $y\text{Noisy} = y + \text{randn}(\text{size}(y))$;
  - $\text{plot}(x,y\text{Noisy},'.'')$;
  - $\text{polyY} = \text{polyfit}(x,y\text{Noisy},2)$;
  - $\text{hold on;}$
  - $\text{plot}(x,\text{polyval(polyY,x)},'r')$
  - $\text{hold off;}$
Exercises in class!
Exercises in class 1/2

```matlab
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

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

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

```matlab
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,:));
```
% 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!