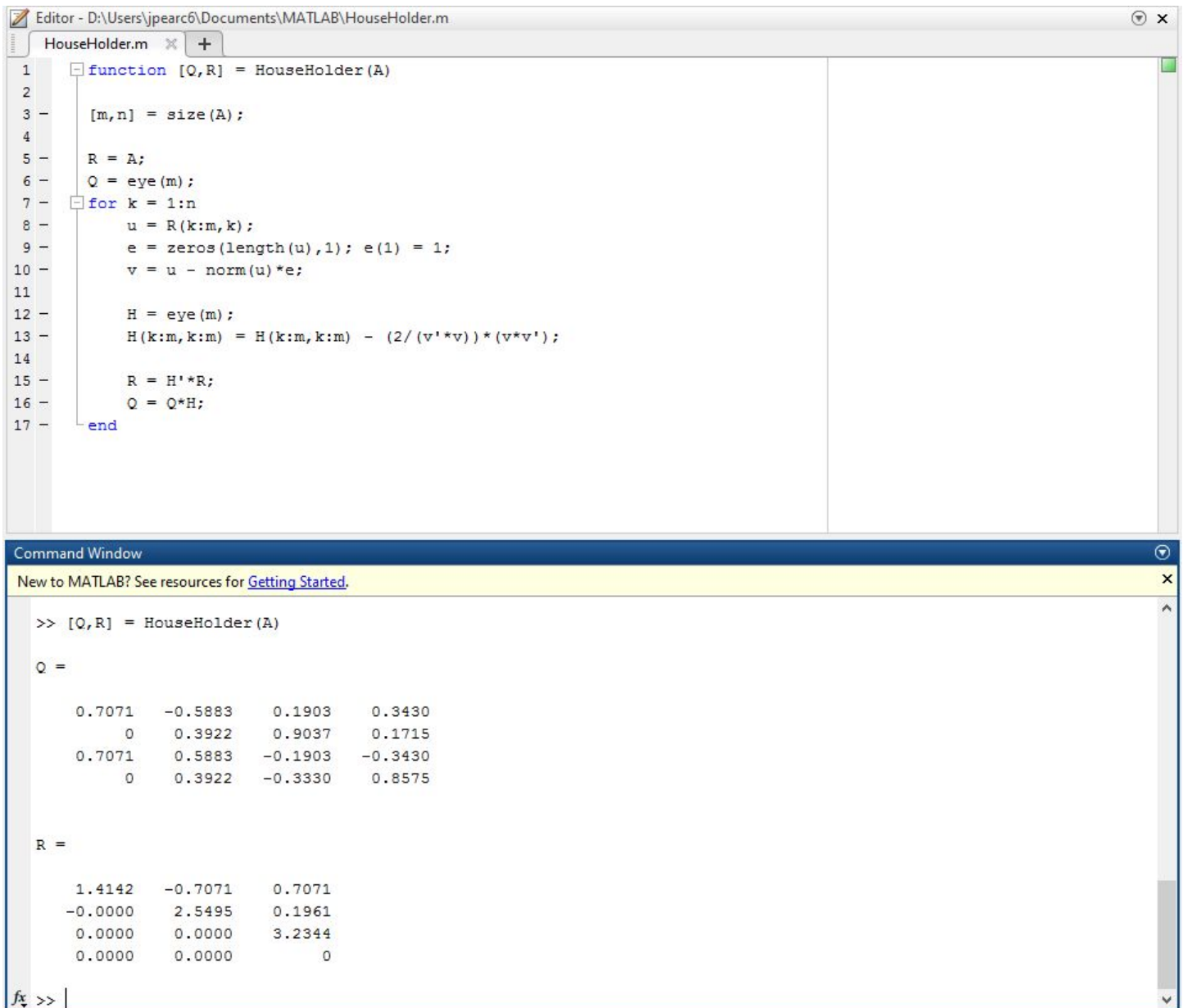


## 2ab) Explicit Version



The image shows a MATLAB environment with two windows. The top window is the MATLAB Editor, displaying a function named `HouseHolder.m`. The function takes a matrix `A` as input and returns two matrices, `Q` and `R`. The code implements the Householder transformation algorithm. The bottom window is the Command Window, showing the execution of the function with a sample matrix `A`. The output displays the matrices `Q` and `R`.

```
1 function [Q,R] = HouseHolder(A)
2
3 [m,n] = size(A);
4
5 R = A;
6 Q = eye(m);
7 for k = 1:n
8     u = R(k:m,k);
9     e = zeros(length(u),1); e(1) = 1;
10    v = u - norm(u)*e;
11
12    H = eye(m);
13    H(k:m,k:m) = H(k:m,k:m) - (2/(v'*v))*(v*v');
14
15    R = H'*R;
16    Q = Q*H;
17 end
```

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
>> [Q,R] = HouseHolder(A)

Q =

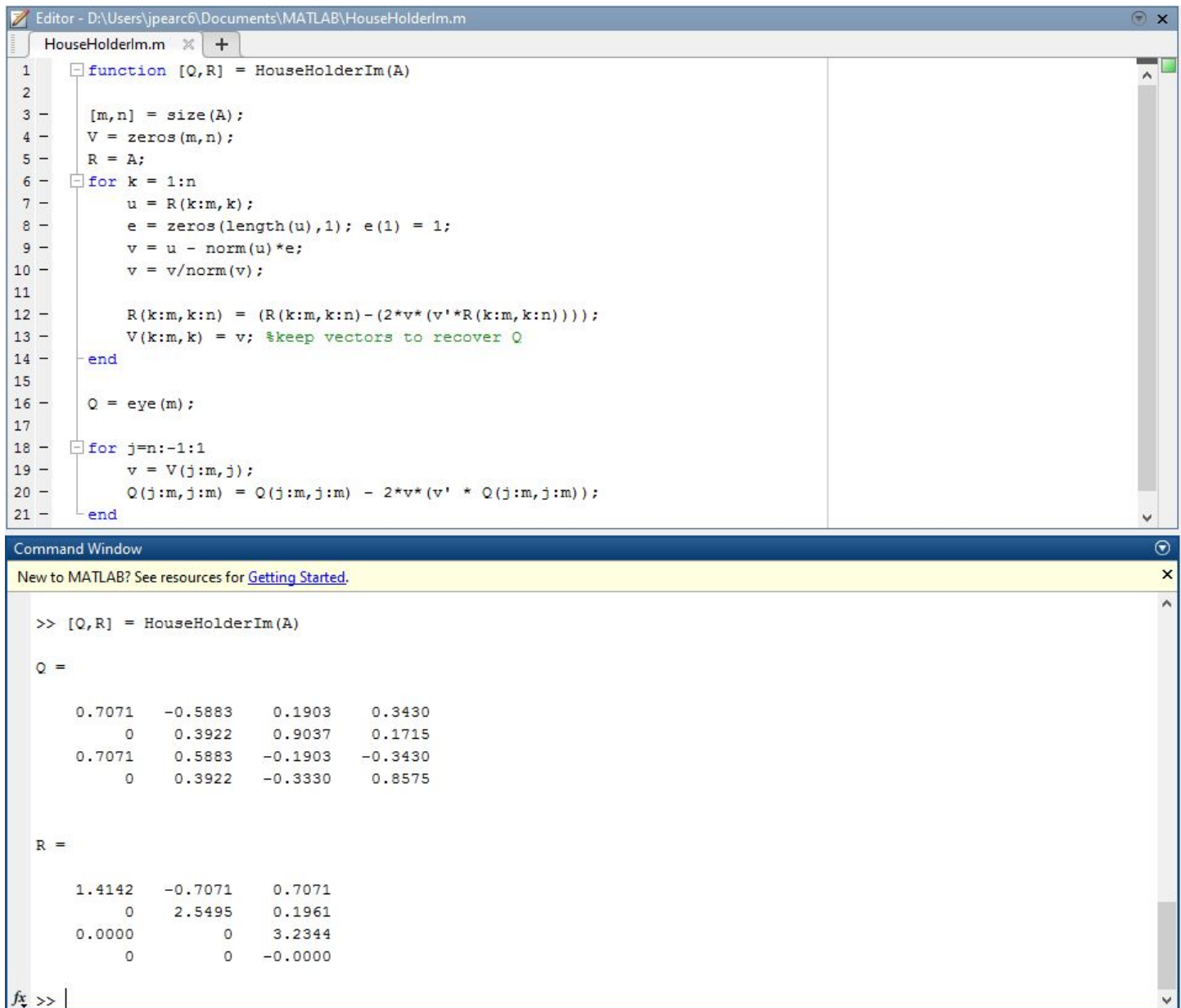
    0.7071   -0.5883    0.1903    0.3430
         0    0.3922    0.9037    0.1715
    0.7071    0.5883   -0.1903   -0.3430
         0    0.3922   -0.3330    0.8575

R =

    1.4142   -0.7071    0.7071
   -0.0000    2.5495    0.1961
    0.0000    0.0000    3.2344
    0.0000    0.0000         0
```

fx >> |

## 2ab) Implicit Version



The image shows a MATLAB Editor window with a script named `HouseHolderIm.m` and a Command Window below it.

**HouseHolderIm.m**

```
1 function [Q,R] = HouseHolderIm(A)
2
3 [m,n] = size(A);
4 V = zeros(m,n);
5 R = A;
6 for k = 1:n
7     u = R(k:m,k);
8     e = zeros(length(u),1); e(1) = 1;
9     v = u - norm(u)*e;
10    v = v/norm(v);
11
12    R(k:m,k:n) = (R(k:m,k:n) - (2*v*(v'*R(k:m,k:n))));
13    V(k:m,k) = v; %keep vectors to recover Q
14 end
15
16 Q = eye(m);
17
18 for j=n:-1:1
19     v = V(j:m,j);
20     Q(j:m,j:m) = Q(j:m,j:m) - 2*v*(v' * Q(j:m,j:m));
21 end
```

**Command Window**

New to MATLAB? See resources for [Getting Started](#).

```
>> [Q,R] = HouseHolderIm(A)

Q =

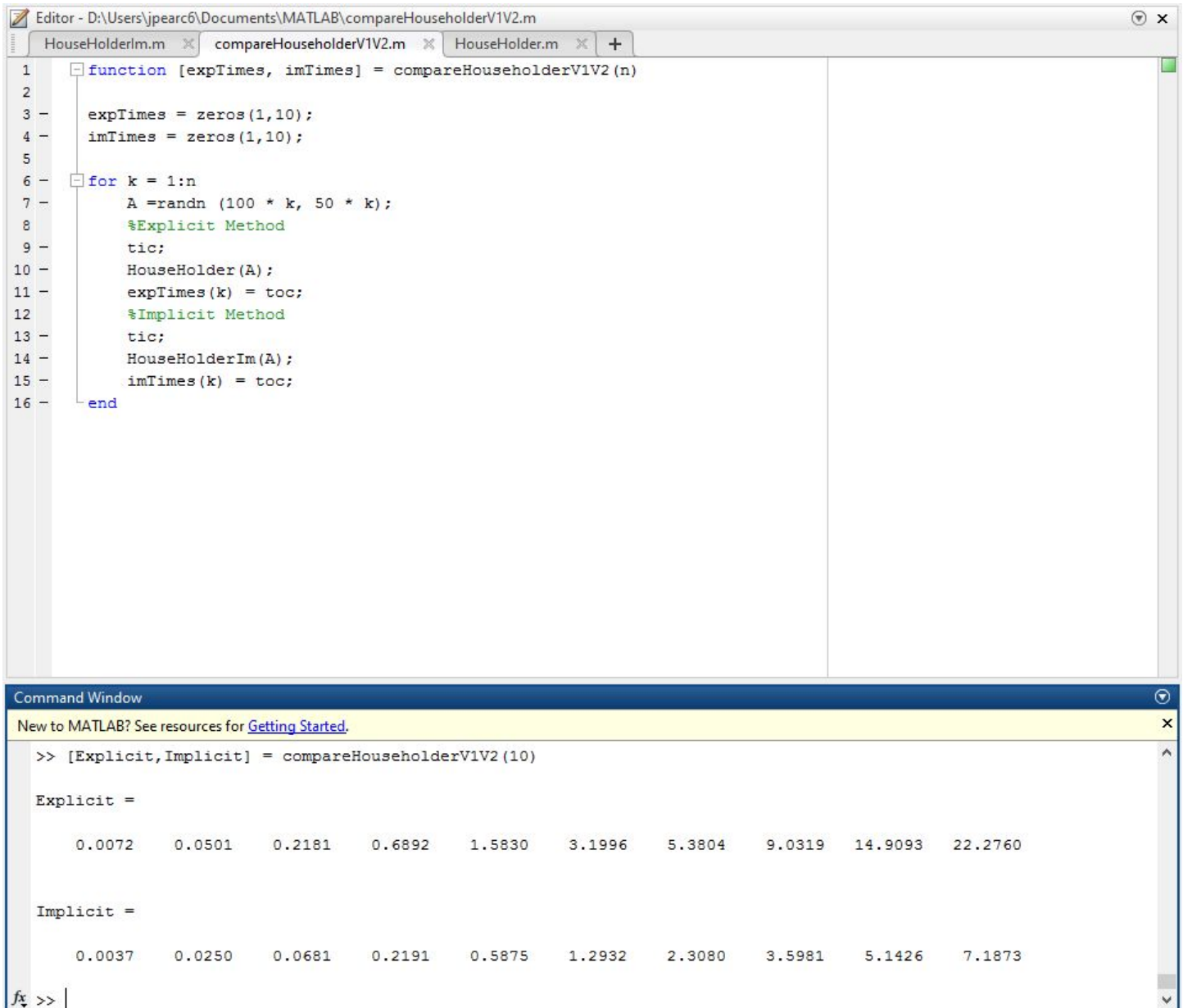
    0.7071    -0.5883     0.1903     0.3430
         0     0.3922     0.9037     0.1715
    0.7071     0.5883    -0.1903    -0.3430
         0     0.3922    -0.3330     0.8575

R =

    1.4142    -0.7071     0.7071
         0     2.5495     0.1961
    0.0000         0     3.2344
         0         0    -0.0000
```

`fx >>`

## 2c) Code and Output



The image shows a MATLAB Editor window with a script named `compareHouseholderV1V2.m` and a Command Window below it.

**Editor - D:\Users\jpearc6\Documents\MATLAB\compareHouseholderV1V2.m**

```
1 function [expTimes, imTimes] = compareHouseholderV1V2(n)
2
3     expTimes = zeros(1,10);
4     imTimes = zeros(1,10);
5
6     for k = 1:n
7         A = randn(100 * k, 50 * k);
8         %Explicit Method
9         tic;
10        HouseHolder(A);
11        expTimes(k) = toc;
12        %Implicit Method
13        tic;
14        HouseHolderIm(A);
15        imTimes(k) = toc;
16    end
```

**Command Window**

New to MATLAB? See resources for [Getting Started](#).

```
>> [Explicit,Implicit] = compareHouseholderV1V2(10)

Explicit =

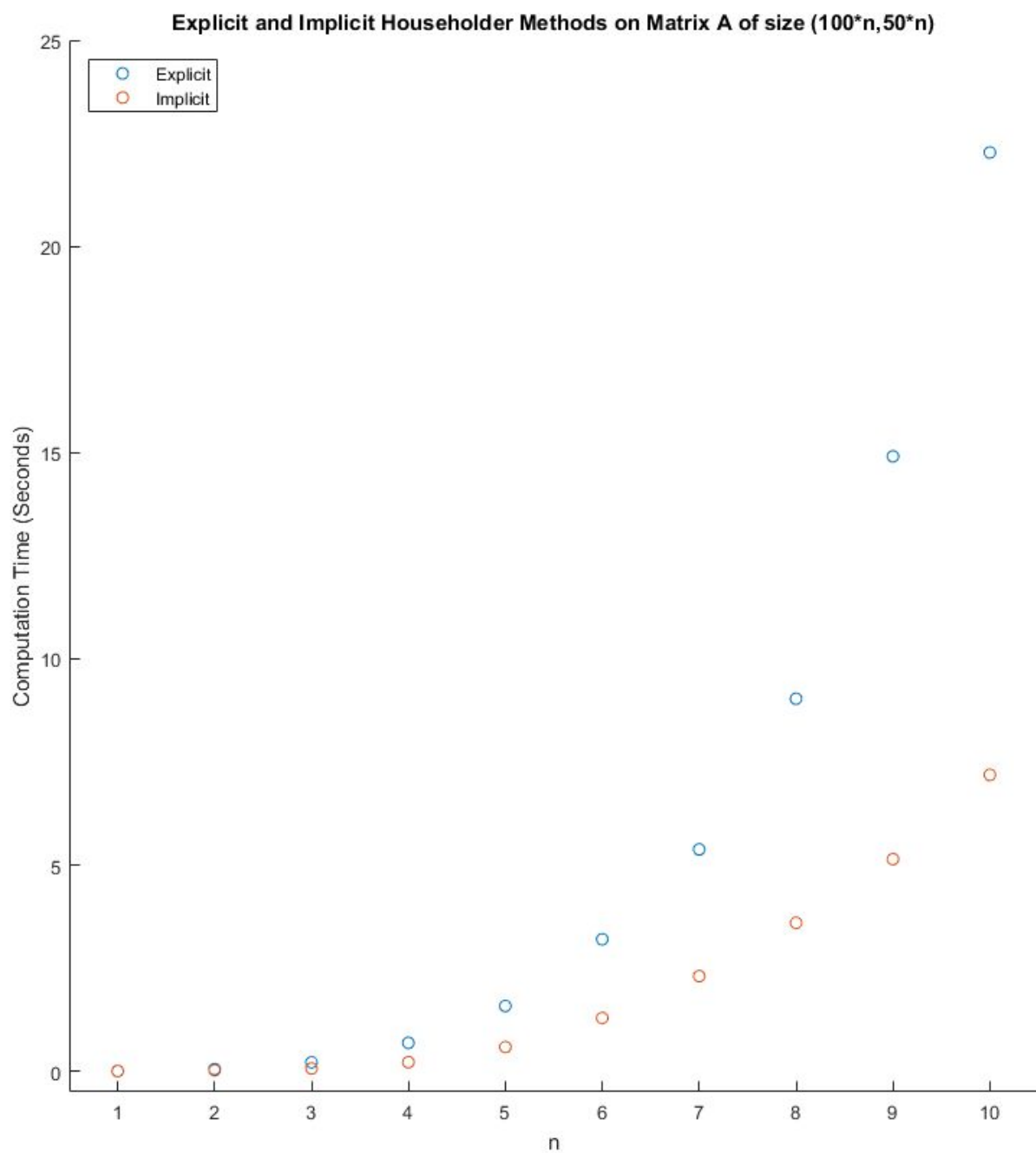
    0.0072    0.0501    0.2181    0.6892    1.5830    3.1996    5.3804    9.0319   14.9093   22.2760

Implicit =

    0.0037    0.0250    0.0681    0.2191    0.5875    1.2932    2.3080    3.5981    5.1426    7.1873

fx >>
```

## 2c) Computation Time Chart



3)a)

As in class the least squares problem can be represented as follows:

$$\min_{\alpha} \|y - A\alpha\|_2$$

Where,

$$y = (0.0131, 0.0542, 0.1652, 0.5614, 1.6412, 2.9546, 4.9099, 8.3626, 10.9059, 15.4361)^T$$

$$n = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)^T$$

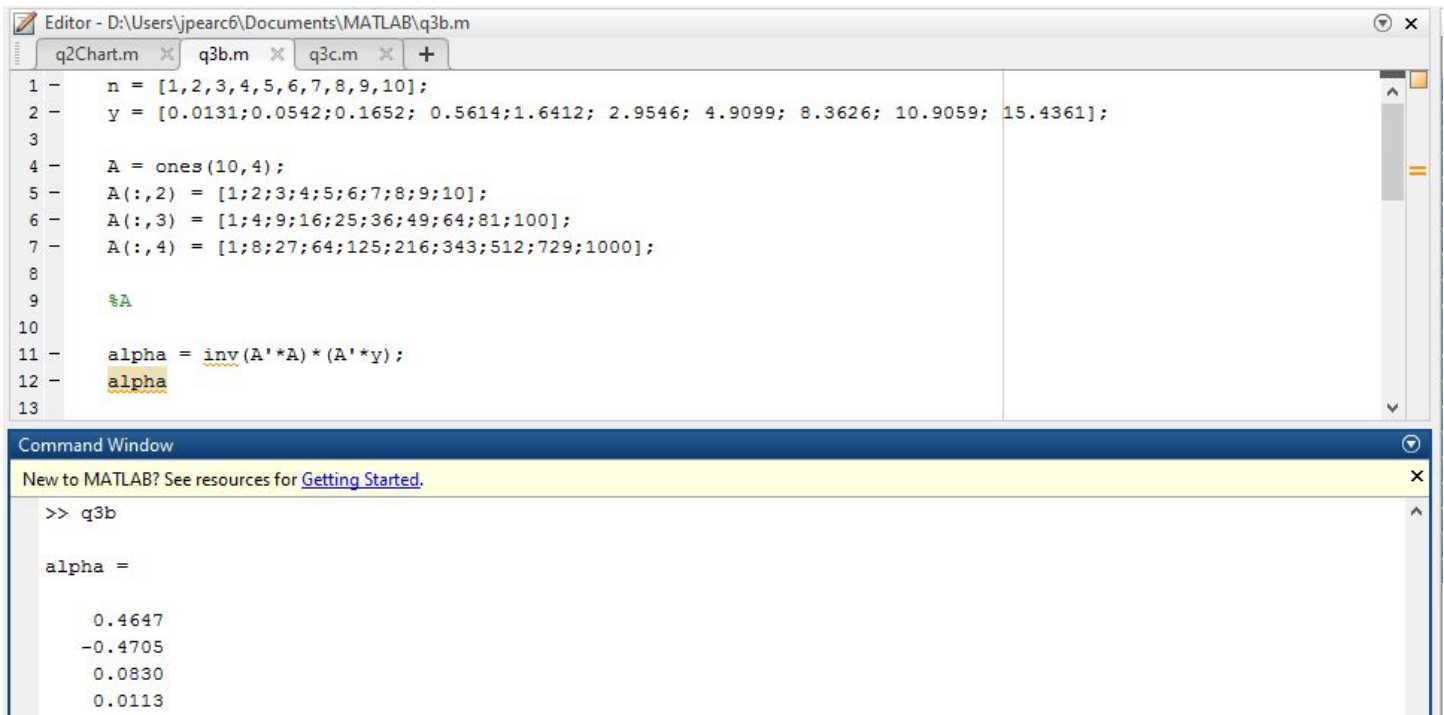
$$A = (n^0, n^1, n^2, n^3)$$

$$\alpha = (w_0, w_1, w_2, w_3)^T$$

3)b)

$$\alpha = (A^T A)^{-1} (A^T y)$$

Using Matlab to solve for  $\alpha$ :



The screenshot shows the MATLAB Editor window with a script named q3b.m. The script defines the vector n, the vector y, and the matrix A. It then calculates the least squares solution alpha using the normal equations. The Command Window shows the output of the script, displaying the value of alpha as a column vector.

```
1 - n = [1,2,3,4,5,6,7,8,9,10];
2 - y = [0.0131;0.0542;0.1652; 0.5614;1.6412; 2.9546; 4.9099; 8.3626; 10.9059; 15.4361];
3
4 - A = ones(10,4);
5 - A(:,2) = [1;2;3;4;5;6;7;8;9;10];
6 - A(:,3) = [1;4;9;16;25;36;49;64;81;100];
7 - A(:,4) = [1;8;27;64;125;216;343;512;729;1000];
8
9 %A
10
11 - alpha = inv(A'*A) * (A'*y);
12 - alpha
13
```

Command Window

New to MATLAB? See resources for [Getting Started.](#)

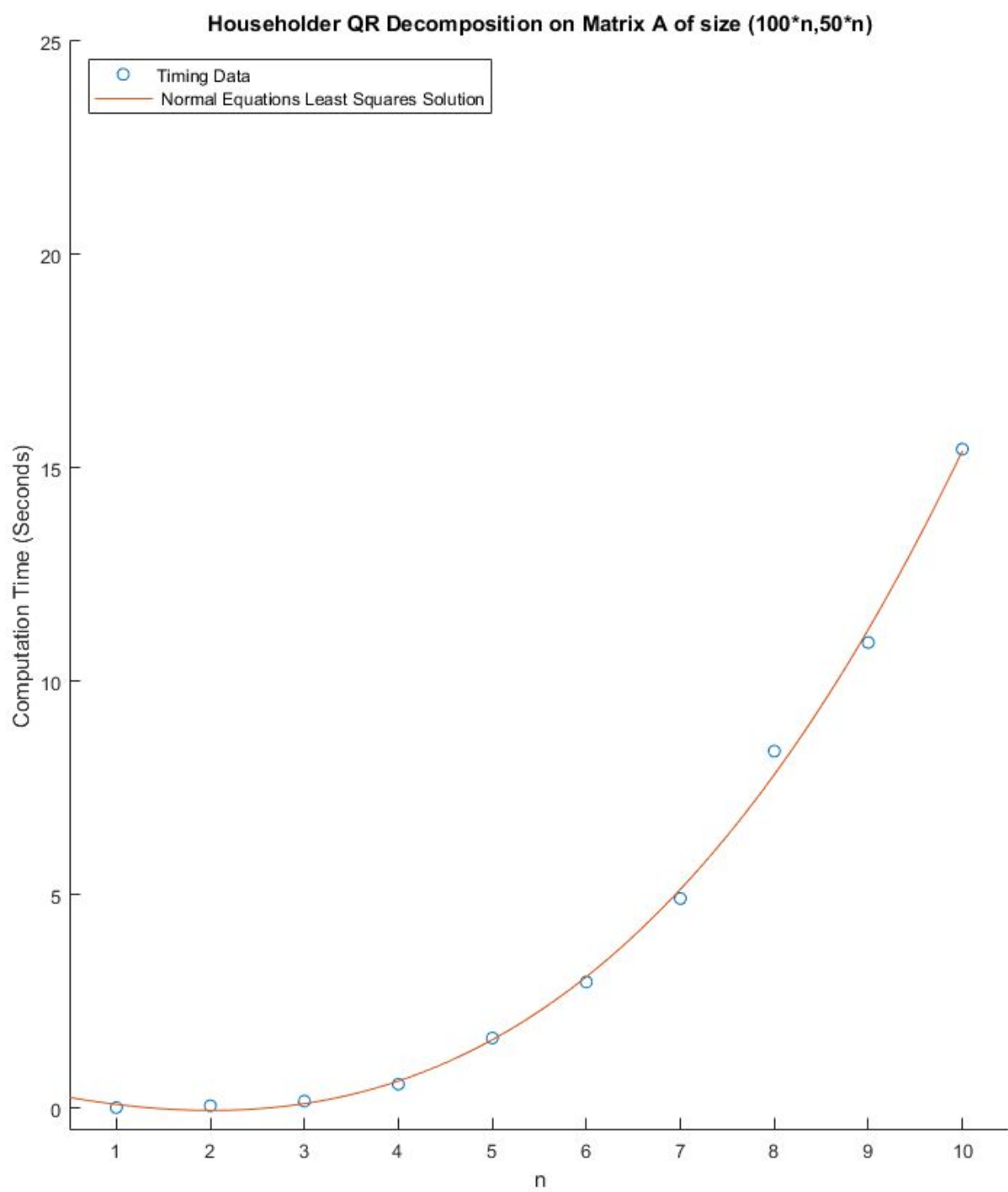
```
>> q3b

alpha =

    0.4647
   -0.4705
    0.0830
    0.0113
```

Normal Equations Least Squares Cubic Polynomial Solution:

$$time(x) = 0.4647 - 0.4705x + 0.0830x^2 + 0.0113x^3$$

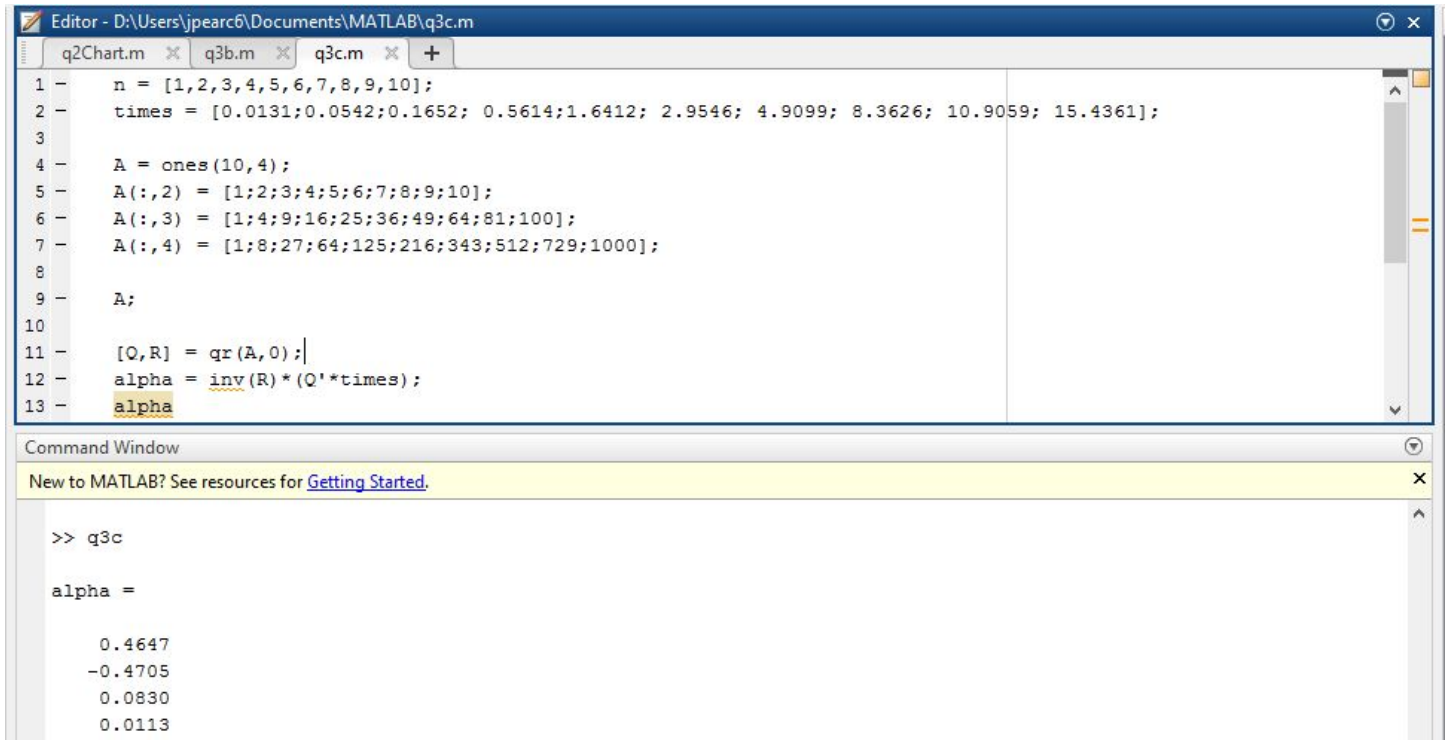


3)c)

Given the reduced QR decomposition of A,

$$\alpha = R^{-1}(Q^T y)$$

Using Matlab to solve for  $\alpha$ :



The image shows a MATLAB Editor window with a script named q3c.m. The script defines a vector n, a vector times, and a matrix A. It then performs a QR decomposition of A using the qr function, and calculates alpha using the formula alpha = inv(R) \* (Q' \* times). The Command Window shows the output of the script, which is a column vector alpha with four elements: 0.4647, -0.4705, 0.0830, and 0.0113.

```
1 - n = [1,2,3,4,5,6,7,8,9,10];
2 - times = [0.0131;0.0542;0.1652; 0.5614;1.6412; 2.9546; 4.9099; 8.3626; 10.9059; 15.4361];
3
4 - A = ones(10,4);
5 - A(:,2) = [1;2;3;4;5;6;7;8;9;10];
6 - A(:,3) = [1;4;9;16;25;36;49;64;81;100];
7 - A(:,4) = [1;8;27;64;125;216;343;512;729;1000];
8
9 - A;
10
11 - [Q,R] = qr(A,0);
12 - alpha = inv(R) * (Q'*times);
13 - alpha
```

Command Window

New to MATLAB? See resources for [Getting Started](#).

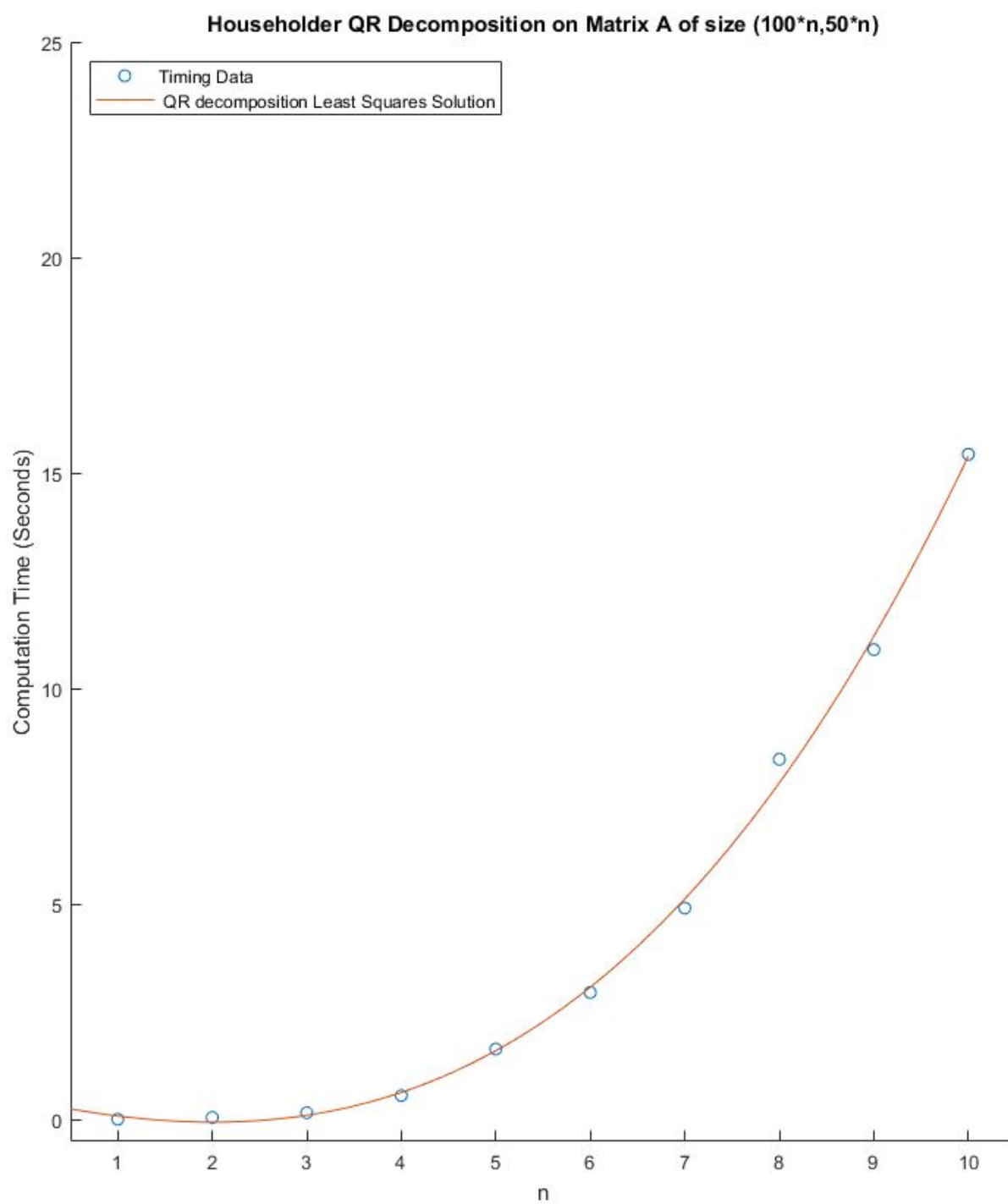
```
>> q3c

alpha =

    0.4647
   -0.4705
    0.0830
    0.0113
```

QR Decomposition Least Squares Cubic Polynomial Solution:

$$time(x) = 0.4647 - 0.4705x + 0.0830x^2 + 0.0113x^3$$





3d)i)

750 x 375 Matrix. Therefore  $x = \frac{750}{100} = 7.5$

$$time(7.5) = 0.4647 - 0.4705(7.5) + 0.0830(7.5)^2 + 0.0113(7.5)^3 = 6.372$$

According to my model, I expect this Householder algorithm to run for 6.372 seconds for a 750 x 375 size matrix

3d)ii)

10000 x 5000 Matrix. Therefore  $x = \frac{10000}{100} = 100$

$$time(100) = 0.4647 - 0.4705(100) + 0.0830(100)^2 + 0.0113(100)^3 = 12083.415$$

According to my model, I expect this Householder algorithm to run for 12083.415 seconds for a 10000 x 5000 size matrix