MATLAB Assignment

Submitted By

TAIWO ALARE

SBU ID # 115360301

Date: 09/21/2023

Question one

Creating a code that convert the base of a number from base n to base m

Number base conversion

```
% This code convert a number in base n to base m.
% base n and base m are between 2 and 10
% To do this, I converted base n to base 10 using "base2dec(num_base_n, n)" function
% and then converted from base 10 to base m using "dec2base(num_base_10, m)" function.
% But first, I verified that the inputed number is in base n and that base
% n and m are between 2 to 10.
% I want to ask the user to enter base 'n' and base 'm' but n and m must be 2 -10  (2 ≤ n ≤ 10)
disp('base n  and base m should be between 2 to 10');
n = input('Enter the base of  n (2 ≤ n ≤ 10): ');
m = input('Enter the base of  m (2 ≤ m ≤ 10): ');
% I want to check whether  base n and base m is between the specified range
if n < 2 || n > 10
   error('Invalid input for n. Please enter a value between 2 and 10.');
end
if m < 2 || m > 10
   error('Invalid input for m. Please enter a value between 2 and 10.');
end
% I want to ask the user for a number in base-n
disp('The number should be in the base you specified for n'),
num_base_n = input(['Enter a number in base-' num2str(n) ': '], 's');
% I want to check if the input number is valid in base-n, if not it would
% display an error feedback
valid_input_n = CheckBase(num_base_n, n);
if ~valid_input_n
   error('Invalid input for base-%d. Please enter a valid number in base-%d.', n, n);
end
% My first step, I want to convert the number from base-n to base-10
num_base_10 = base2dec(num_base_n, n);
```

```
% My second step, I want to convert the number from base-10 to base-m
num_base_m = dec2base(num_base_10, m);
% Display the results
fprintf('The number %s in base-%d is equivalent to %s in base-%d.\n', num_base_n, n, num_base_m, m);
function valid = CheckBase(num_str, n)
   %I am initializing a flag to determine if the input is valid in base-n
   valid = true;
   % Convert the input number to a string for processing
   num_digits = numel(num_str);
   % Iterate through each digit and check if it's within the base-n range
   for i = 1:num_digits
      digit = str2double(num_str(i));
      if isnan(digit) || digit >= n
         valid = false;
         return;
      end
   end
end
```

Running the code to convert 40443_5 to base 7

```
base n  and base m should be between 2 to 10
Enter the base of  n (2 ≤ n ≤ 10): 5
Enter the base of  m (2 ≤ m ≤ 10): 7
The number should be in the base you specified for n
Enter a number in base-5: 40443
```
The number 40443 in base-5 is equivalent to 10435 in base-7.

Testing the code

The code is saved in *Taiwo_AlareHW_1.m* file.

Comments

For simplicity in base conversion, the number in base n is first converted to decimal and then converted to base m from decimal.

Question two

Creating a function that solve matrix equation using Gaussian Elimination Method

```
function x = GaussElim(A, B)
   N = length(B);
   x = zeros(N,1);

   % Augmenting the matrix by joining the source vector B with the co-efficient matrix A
   Aug = [A, B];  % Use a comma to concatenate matrices horizontally

   % I make the augmented matrix an upper triangular matrix using this loop

   for j = 1:N-1
      for i = j+1:N
         if Aug(j,j) ~= 0
         m = Aug(i,j) / Aug(j,j);
         Aug(i,:) = Aug(i,:) - (m * Aug(j,:));
         else
            error('computational error') % dividing by zero will give syntax error
         end

   end

   % Displaying the upper triangular augmented matrix
   Aug

   x(N) = Aug(N,N+1) / Aug(N,N);
   for k = N-1:-1:1
      x(k) = (Aug(k,N+1) – Aug(k,k+1:N) * x(k+1:N)) / Aug(k,k);
   end
   x
end
```

Creating a code that solve matrix equation using Gaussian Elimination Method

```
% Solving matrix equation Ax=B using Guassian Elimination Method
% A is the co-efficent matrix (n*n) and B is the source vector
disp('The co-efficent matrix should be in a bracket and the rows should be separated with ; [a b c; 1 2 3]')
disp('The source vector should be in a bracket and the rows should be separated with ; [a;b;c]')

A = input('Enter your coefficient matrix: ');
B = input('Enter the source vector: ');
x = GaussElim(A, B); % Calling the function x= GaussElim(A,B)
```

Running the code to solve

$$\begin{bmatrix} \mathbf{10} & \mathbf{3} & \mathbf{1} \\ \mathbf{3} & \mathbf{10} & \mathbf{2} \\ \mathbf{1} & \mathbf{2} & \mathbf{10} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \end{bmatrix} = \begin{bmatrix} \mathbf{19} \\ \mathbf{29} \\ \mathbf{35} \end{bmatrix}$$

```
The co-efficient matrix should be in a bracket and the rows should separated with ; [a b c; 1 2 3]
The source vector should be in a bracket and the rows should separated with ; [a;b;c]
Enter your coefficient matrix: [10 3 1; 3 10 2; 1 2 10]
Enter the source vector: [19; 29; 35]

Aug =

   10    3    1   19
    3   10    2   29
    1    2   10   35


Aug =

  10.0000    3.0000    1.0000   19.0000
         0    9.1000    1.7000   23.3000
         0         0    9.5824   28.7473


x =

   1.7951
   2.5986
   3.0000

>>
```

Testing the code

The function is saved in *GuassElim.m* file and the rest is saved in *Taiwo_Alare_HW_2.m* file. Both files should be opened first and running the *Taiwo_Alare_HW_2.m* file.

Question three

Creating a function that solve pi approximation using Monte Carlo Method

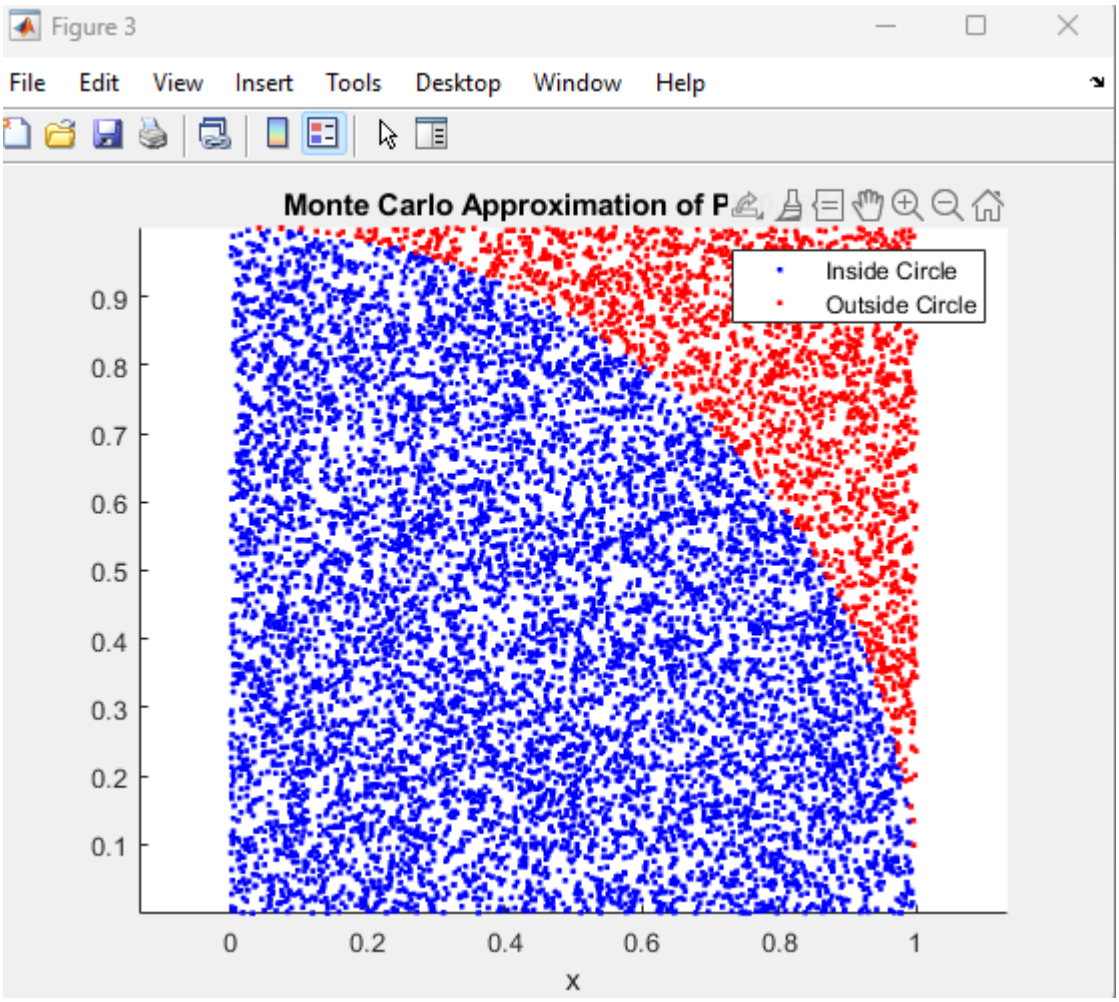
```
function y = MontePi(n)
   inside_circle = 0;

   for i = 1:n
      x = rand();
      y = rand();

      if x^2 + y^2 <= 1
         inside_circle = inside_circle + 1;
      end
   end
% pi_approx is the approxation of pie
   pi_approx = 4 * inside_circle / n;
   t = abs(pi - pi_approx); % t is the absolute error due pie approximation
   rel_error = t / pi;      % rel_error is the relavative error
end
```

Creating a code that solve pi approximation using Monte Carlo Method

```
% This code shows the approximation of Pi using Monte Carlo method
function y = MontePi(n) % I called the function y = MontePi.m



% first, I  time the execution for various values of n
n_values = [1000, 10000, 100000, 1000000]; % n_values are my given value of n
execution_times = zeros(size(n_values));

% I determined the time of execution of the length of n_values using tic and
% toc command for loop
for i = 1:length(n_values)
   n = n_values(i);

   tic;
   MontePi(n);
   execution_times(i) = toc;
end

figure;
plot(n_values, execution_times, 'o-');
xlabel('n');
zlabel('Execution Time (seconds)');
title('Execution Time vs. n');

% I also computed approximations of π and plot absolute errors
n_values = logspace(2, 6, 50);
```

```
t = zeros(size(n_values)); % t is the absolute error

for i = 1:length(n_values)
   n = round(n_values(i));
   y = MontePi(n);
   t(i) = y;
end

figure;
loglog(n_values, t, 'o-');
xlabel('n');
zlabel('Absolute Error');
title('Absolute Error vs. n');
xlim([10^2, 10^6]);

% I generated random points and display final value of π
n = 10000;
[x, z, color] = deal(zeros(n, 1));

figure;
axis equal;
hold on;

for i = 1:n
   x(i) = rand();
   z(i) = rand();

   if x(i)^2 + z(i)^2 <= 1
      color(i) = 1; % Inside the circle
   else
      color(i) = 2; % Outside the circle
   end
end

scatter(x(color == 1), z(color == 1), 'b.'); % blue color for outside the circle
scatter(x(color == 2), z(color == 2), 'r.'); % red color for outside the circle
xlabel('x');
zlabel('y');
title(['Monte Carlo Approximation of Pi: ', num2str(MontePi(n))]);

legend('Inside Circle', 'Outside Circle');
```

Running the code



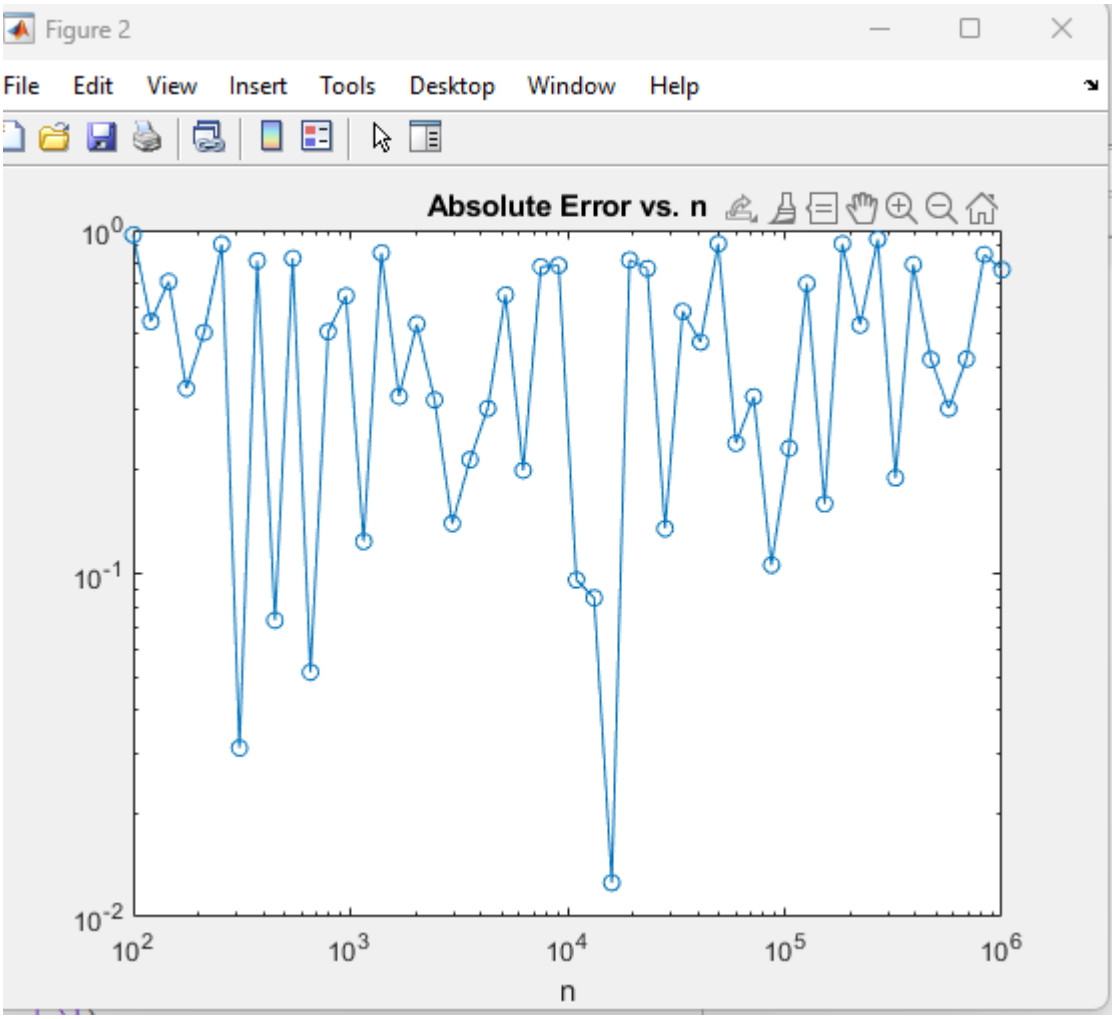
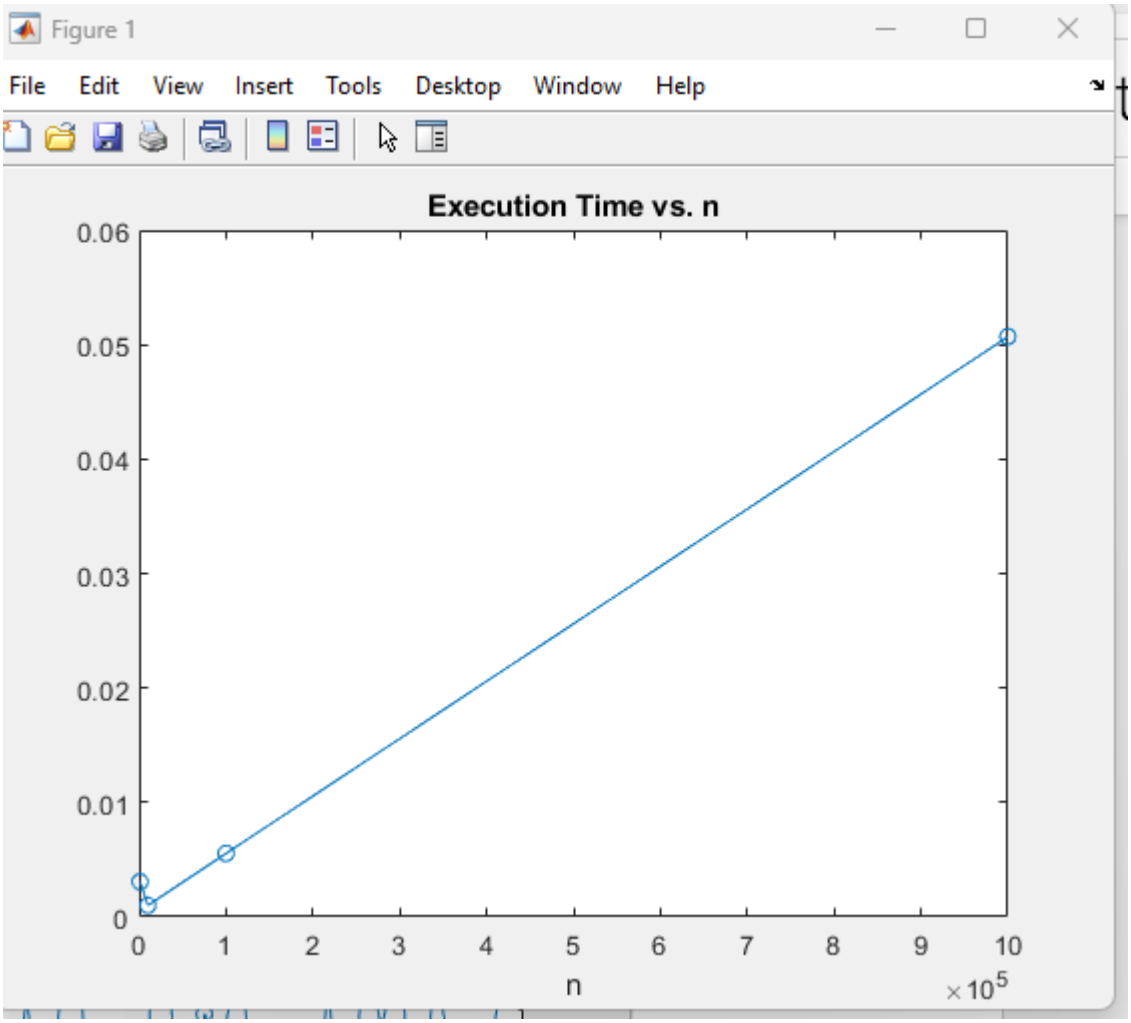


Figure 2
File Edit View Insert Tools Desktop Window Help
Absolute Error vs. n
10^0
10^{-1}
10^{-2}
10^2
10^3
10^4
10^5
10^6
n



Testing the code

The function is saved in *Monte.m* file and the rest is saved in *Taiwo_Alare_HW_3.m* file. Both files should be opened first and running the *Taiwo_Alare_HW_2.m* file should produce the animation