1. Create random vectors ‘a’ and ‘b’ ...

\[ a \times b \]

*a* represents matrix multiply. It requires that the number of columns in the first matrix must be equal to the number of rows in the second matrix. In the present case, since both \( a \) and \( b \) are 10 \times 1 vectors, this command will yield error message:

```plaintext
Error using ==> *
Inner matrix dimensions must agree.
```

\[ a \cdot b \]

*.* represents element by element multiplication of matrices, and requires that the dimensions of the two matrixes agree. The result \( r = a \cdot b \) will have the same dimension with entries \( r_{ij} = a_{ij} \cdot b_{ij} \).

\[ a / b \]

This is right matrix divide. If \( r = a / b \), then we have \( a = r \cdot b \), where ‘*’ is matrix multiplication.

\[ a ./ b \]

This performs a term by term division. It requires that the dimension of the two matrixes agree. The result \( r = a ./ b \) will have the same dimension with entries \( r_{ij} = a_{ij} \div b_{ij} \).

\[ a \ast b' \]

This is matrix multiply with the transpose ("(" is transpose). Since after the transposition, the number of columns of \( a \) is equal to the number of rows \( b^T \), which is 1, the command will yield a 10 \times 10 matrix with entry \( r_{ij} = a_{ij} \ast b_{ij} \), where \( N \) is the number of columns of \( a \), or number of rows of \( b^T \).

\[ a \setminus A \]

This is left matrix divide, roughly equal to \( a^\ast \ast A \), where is \( a^\ast \) the pseudo inverse matrix of \( a \) (\( a^\ast \ast a \setminus a \)). If \( r = a \setminus A \), then \( a^\ast \ast A = a^\ast \ast a \ast x \), or \( A^\ast \ast a = (a \ast x)^\ast \ast a \). Here the ‘*’ is matrix multiply.

An example of these commands is listed in the appendix.

2. Colon notation ...

\[ 1:10 \]

This yields an array \([1 \ 2 \ 3 \ \ldots \ 10] \).

\[ a[1:5] \]

This outputs the first five elements of \( a \).

\[ a(4:9) + b(1:6) \]

This adds the fourth through ninth elements of \( a \) to the first through sixth elements of \( b \) and produces a six dimensional vector.

\[ 0.1:0.1:100 \]

This yields a 1000 element array with entries beginning from 0.1 and ending at 100, i.e. \([0.1, 0.2, 0.3, \ldots, 100] \).

3. Boolean variables ...

These commands employ Boolean return those entries in \( b \) whose values are bigger than the correspondent entries in \( a \). They are explained in detail as follow:

\[ bga = b > a \]

The \( bga \) will be a logical array with entries \( bga_i = \begin{cases} 1, & \text{if } b_i > a_i, \\ 0, & \text{otherwise} \end{cases} \).

\[ Ib = \text{find}(bga) \]

The function \text{find} finds the indices of non-zero elements of \( bga \). In this example, array \( Ib \) contains the indices of those elements of \( b \) that are larger than the corresponding element of \( a \).

\[ b(Ib) \]

This command returns those entries of \( b \) with indices saves in \( Ib \). In this example, they are actually the entries in \( b \) whose values are bigger than the correspondent entries in \( a \).

4. Image

The output of the two commands are shown as follow:

\[ \text{image}(I) : \]

\[ \text{imshow}(I) : \]
image(I) displays matrix I as an image. Each element of I specifies the color of a rectilinear patch in the image. I can be a matrix of dimension M×N or M×N×3, and can contain double, uint8, or uint16 data. The image will always be displayed as a square image. When I is a 2-dimensional M×N matrix, the elements of I are used as indices into the current colormap to determine the color. image(I) places the center of element I(1,1) at (1,1) on the axes, and the center of element (M, N) at (M, N) on the axes, and draws each rectilinear patch as a square with 1 unit in width and height.

imshow(I) displays the intensity image I with N discrete levels of gray. If the N is omitted, imshow uses 256 gray levels on 24-bit displays, or 64 gray levels on other systems. The imshow shows the image to its true size instead of square by default.

* Reference: Matlab help file

5. Function

Please see the appendix for detail. The function [x1, x2] = rootsGQE(coef) returns the roots (real or complex) of a general quadratic equation with one unknown, given the coefficients of the equation. The coefficients are in the form of [a, b, c], corresponding to quadratic equation $ax^2 + bx + c = 0$. The roots are saved in x1 and x2. When a = 0, the equation degrades to a linear equation. When a = b = 0, the function returns -Inf (stands for infinity) as roots. When a = b = c = 0, the function returns NaN (stands for Not a Number) as roots.

6. Advantage of vectorization

Please see the appendix for detail.

7. Finally a bit of fun

The goal of this problem was to show you that many things affect the speed of algorithm execution, and that different computers do different things better.

In addition, as this result shows, computers can perform unexpectedly badly due to the OS or some other reason.

The machine used is
CPU: Pentium III 667MHz @ 133MHz bus
Storage: 128MB @ 133MHz bus memory, 20G HD
OS: Windows 98 second edition
Appendix: Scripts and examples¹.

1. Scripts and examples for question no. 1

Script:

function hw_1

% CMSC 828D: Fundamentals of Computer Vision
% Homework 1
% Instructors: Larry Davis, Raman Duraiswami, Daniel DeMenthon, and Yiannis Aloimonos
% Student: Haiying Liu
% Date: Aug. 30, 2000
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a = rand(10, 1);
b = rand(10, 1);
A = rand(10, 10);
r1 = a * b;
disp(['a * b = ', num2str(r1)]);
r2 = a .* b;
disp(['a .* b = ', num2str(r2)]);
r3 = a / b;
disp(['a / b = ', num2str(r3)]);
r4 = a ./ b;
disp(['a ./ b = ', num2str(r4)]);
r5 = a .* b';
disp(['a .* b' = ', num2str(r5)]);
r6 = a \ A;
disp(['a \ A = ', num2str(r6)]);

Examples:

hw1_1

r6 = a \ A;
disp(['a \ A = ', num2str(r6)]);

Examples:

hw1_1

1. All scripts are written as functions to prevent them from interfering each other.
2. Scripts and examples for question no. 2

Script:

```matlab
function hw_2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%===============================================================================
%= Qustion no.2
%= Excecute the following commands and explain.
a  = rand(10, 1);
b  = rand(10, 1);
r1 = 1:10;
disp(' ');
disp('1:10 = ');
disp(num2str(r1));
r2 = a(1:5);
disp(' ');
disp('a(1:5) = ');
disp(num2str(r2));
r3 = a(4:9) + b(1:6);
disp(' ');
disp('a(4:9) + b(1:6) = ');
disp(num2str(r3));

a \ A =

0.48763  0.76342  0.51268  0.83627  0.54667  0.52898
0.57729  0.54086  0.78753  0.7749

```

Examples:

```matlab
hw1_2

1:10 =
1 2 3 4 5 6 7 8 9 10
```

```matlab
a(1:5) =
0.27946
0.88505
0.43355
0.37712
0.22666
```

```matlab
a(4:9) + b(1:6) =
0.87988
0.49369
0.99302
0.97128
```
0.62111
1.5469
0.1:0.1:100 =
0.1 0.2 0.3 ... 100
diary off

3. Scripts and examples for question no. 3

**Script:**

```matlab
function hw_3
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%= Question no.3
%= Executing the following commands and explain.
% a = rand(10, 1)
% b = rand(10, 1)
% bga = b > a
% Ib = find(bga)
% b(Ib)

Example:

hw1_3

a =
```
0.7463
0.4629
0.4730
0.9297
0.5494
0.0268
0.4832
0.0426
0.7744
0.5816
```

b =
```
0.0300
0.8845
0.6577
0.9891
0.7825
0.0678
0.6423
0.2053
0.7586
0.9711
```

bga =
```
0
1
```
4. Scripts and examples for question no. 4

Scripts:

function hw_4

% CMSC 828D: Fundamentals of Computer Vision
% Homework1
% 
% Qustion no.4
%= Create a random gray level image I (0-255) of size 256x128 using the rand
% command. Convert this image to type uint8. Display it using the commands
%= image(I)
%= imshow(I)
% What differences do you see?

I = rand(258, 128) * 255;
I = uint8(I);
figure;
image(I);
figure;
imshow(I);

diary off
5. Scripts and examples for question no. 5

**Script:**

```matlab
function [x1, x2] = rootsGQE(coefficients)
    % Syntax [x1, x2] = rootsGQE(coefficients)
    % coefficients - coefficients of the general quadratic equation
    % x1, x2 - two roots of the equation
    % Description: Compute the roots of a general quadratic equation
    % expressed by its coefficients
    % Date : Aug. 30, 2000
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %= Question no.5
    %= Write a function that will return the roots of a general quadratic equation
    %= given the coefficients of the equation. Make your program as general as you
    %= can.
    a = coefficients(1);
    b = coefficients(2);
    c = coefficients(3);
    if a ~= 0
        temp = sqrt(b .* b - 4 * a * c);
        x1 = (-b + temp) / (a + a);
        x2 = (-b - temp) / (a + a);
    else
        x1 = -c / b;
        x2 = x1;
    end

    % Examples:
    [x1, x2] = rootsGQE(rand(3, 1))
    x1 =
    -0.1750
    x2 =
    -0.8530
```

---

[x1, x2] = rootsGQE([9, 1, 3])
x1 =
-0.0556 + 0.5747i
x2 =
-0.0556 - 0.5747i

[x1, x2] = rootsGQE([0, 1, 3])
x1 =
-3
x2 =
-3

[x1, x2] = rootsGQE([0, 0, 3])
Warning: Divide by zero.
> In G:\Course\CMSC828D\rootsGQE.m at line 42
x1 =
-Inf
x2 =
-Inf

[x1, x2] = rootsGQE([0, 0, 0])
Warning: Divide by zero.
> In G:\Course\CMSC828D\rootsGQE.m at line 42
x1 =
NaN
x2 =
NaN

diary off
6. Scripts and examples for question no. 6

Scripts:

```matlab
function ans = hw1_6
  % Date : Aug. 30, 2000
  ||=%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ||%============================================================================
    ||=%= Question no.6
    ||=%= For problem 3, write a function that explicitly uses for loops to achieve
    ||=%= the same result
    ||= a = rand(10, 1)
    ||= b = rand(10, 1)
    ||= nRow_a = size(a);
    ||= nCol_b = size(b);
    ||= if nRow_a ~= nCol_b
    ||= error('The degree of a must be equal to b.');
    ||= end
    ||= idx_Ib = 0;
    ||= for idx = 1:nRow_a
        ||= if b(idx) > a(idx)
            ||= bga(idx) = 1;
            ||= idx_Ib = idx_Ib + 1;
            ||= Ib(idx_Ib) = idx;
            ||= bb(idx_Ib) = b(idx);
        ||= end
    ||= end
    ||= bga = bga'
    ||= Ib = Ib'
    ||= ans = bb';
    ||= ans =

Example:

hw1_6

a =
  0.4102
  0.4401
  0.2975
  0.2978
  0.4444
  0.9955
```

```matlab
diary off
```

Example:

```
hw1_6

a =
  0.3967
  0.7363
  0.3665
  0.2783
  0.6961
  0.5942
  0.1874
  0.1653
  0.1057
  0.0114
  0.0398
  0.8772
  0.5112
  0.6458
```