MATLAB: Program Flow

1. Overview
2. In-class exercise

MATLAB: Program Flow

Overview

MATLAB Programming

- MATLAB provides capabilities for developing customized programs (e.g. scripts and functions) to accomplish various tasks
- These programs (also called codes) often require the use of certain programming motifs
- Command – a single statement that can be executed at the MATLAB prompt
- Conditional statements – execute a series of commands if a certain relational condition is satisfied
- Loops – execute a series of commands repeatedly until a certain relational condition is satisfied

Relational Operators

- Less than: A < B
- Greater than: A > B
- Less than or equal: A <= B
- Greater than or equal: A >= B
- Equal: A == B
- Not equal: A ~= B
- Relational operators perform element-by-element comparisons between two arrays. They return a logical array of the same size, with elements set to logical 1 (true) where the relation is true, and elements set to logical 0 (false) where it is not.

Relational Operators Examples

```
>> A=1; B=2; A < B
ans =
    1
>> A = [1 2 3]; B = [3 2 1]; B >= A
ans =
    1     1     0
>> A = [1 0 0; 0 1 0; 0 1 0]; A == 1
ans =
    1     0     0
    0     1     0
    0     1     0
```

If Statement

```
if expression
        statements
elseif expression
        statements
else
        statements
end
```

- Evaluates an expression, and executes a group of statements when the expression is true.
- The elseif and else blocks are optional. The statements execute only if previous expressions in the if...end block are false. An if block can include multiple elseif blocks.

If Statement Example

```
function [x] = quadratic_mod(coeff)
a = coeff(1);
b = coeff(2);
c = coeff(3);
discrim = b^2-4*a*c;
if discrim >=0
    x1 = -(b+sqrt(discrim))/(2*a);
    x2 = -(b-sqrt(discrim))/(2*a);
    x = [x1 x2];
else
    x = [NaN NaN];
    display('Roots not real');
end
```

If Statement Example

```
>> x = quadratic_mod([2 4 1])

x =

   -1.7071   -0.2929

>> x = quadratic_mod([2 2 1])

Roots not real

x =

   NaN   NaN
```

For and While Loops

```
for index = values
        statements
end
```

- Executes a group of statements in a loop a specified number of times.

```
while expression
        statements
end
```

- Evaluates an expression, and repeats the execution of a group of statements in a loop while the expression is true.

For Loop Example

- Function that constructs a *nxn* identify matrix

```
function [A] = identity(n)
for i = 1:n
    for j = 1:n
        if i == j
            A(i,j) = 1;
        else
            A(i,j) = 0;
        end
    end
end
```

For Loop Example

```
>> n = 2;
>> A = identity(n)
A =
     1     0
     0     1
>> n = 3;
>> identity(n)
ans =
     1     0     0
     0     1     0
     0     0     1
>> eye(n) ← built-in function
ans =
     1     0     0
     0     1     0
     0     0     1
```

While Loop Example

- Function to calculate *n* factorial

```
function f = fact(n)
f = n;
while n > 1
    n = n-1;
    f = f*n;
end
>> n = 6;
>> fact(n)
ans =
   720
>> factorial(n) ← built-in function
ans =
   720
```

MATLAB: Program Flow

In-class Exercise

Calculate Hilbert Matrix

- A Hilbert matrix is a *n*x*n* matrix with the following structure:

$$H=\begin{bmatrix} 1 & \frac{1}{2} & \cdots & \frac{1}{n-1} & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} & \frac{1}{n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{n-1} & \frac{1}{n} & \cdots & \frac{1}{2n-3} & \frac{1}{2n-2} \\ \frac{1}{n} & \frac{1}{n+1} & \cdots & \frac{1}{2n-2} & \frac{1}{2n-1} \end{bmatrix} \xrightarrow{n=3} H=\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

- Write a function that creates a Hilbert matrix for any size *n*