

## 34. PROGRAMMING

Topics:

- 

Objectives:

- 

### 34.1 Overview

- brief summary of topic(s)
- category: review / fundamental / application / research

### 34.2 Introduction

- Some useful program elements
  - 'clear' empties the screen
  - 'exec' executes a file as a script.
  - 'chdir' changes a working directory
  - 'getf('functionfile') - gets a function from a working directory
  - pwd - returns the current working directory
  - 'xbasc() - clears a graphics window

- For-loops in Scilab,

```
for i=0:0.1:6 // 0 to 6 in 0.1 steps
    // loop code goes here
end
```

```
// OR using vectors
V=[0:0.1:6];
for i=V
    // loop code goes here
end
```

- While-loops in Scilab,

```
i = 0;
while(i <= 5),
    i = i + 1;
    i
end
i
```

- If-then in Scilab,

```
if(__logical__),
    // some code
else
    // other code
end
```

```
// >, >=, <, <=, ==, ~= Basic comparisons
// &, |, ~ Basic Boolean operators
```

- Functions in Scilab,

```
function returnval = functionname(arg1, arg2)
    returnval = arg1 + arg2;
    if(returnval < 0),
        returnval = 0;
    return
end
endfunction

functionname(5, 3)
```

- Note that when a function is defined in Scilab it is not actually run. However when the function is called it will be interpreted.
- Cases in Scilab,

```
select val,
    case 0,
        // code
    case 1,
        // more code
    else
        break
end
```

- Printing in Scilab,

```
mprintf("format string goes here %d %f\n", arg_int, arg_float)

// %d - integer
// %f - float
// \n - end of line
```

- Inputting values in Scilab,

```
a = input("input a value for a");
```

### **34.3 Examples**

- A simple Scilab Program,

```

//
// test.sce
//
// A simple program to integrate an accelerating mass
//
// To run this in Scilab use 'File' then 'Exec'.
//
// by: H. Jack Aug 27, 2002
//

// Set the time length and step size
steps = 100;
t_end = 10;
delta_t = t_end / steps;

// Initial conditions for the motion
pos=[0];
vel=[0];
t=[0];
acc=9.81;

// Loop to integrate the motion
for i=1:steps,
    t = [t; i * delta_t];
    vel = [vel; vel($) + acc];
    pos = [pos; pos($) + vel($) * delta_t + 0.5 * acc * delta_t^2];
end

// Dump the values to the screen
//[t vel pos]

// Graph the values
plot2d(t, [pos vel], [-2, -5], leg="position@velocity");
// leg - the legend titles
// style - draw lines with marks
// nax - grid lines for the graph

xtitle('Time (s)');

// Write values to a file (delete the existing file first)
unix("del c:\temp\data.txt");
write("c:\temp\data.txt", [t vel pos]);

```

- A simple numerical integration of the equations of motion (ME/PDM),

```

// first_order.sce
// A first order integration of an accelerating mass
// To run this in Scilab use 'File' then 'Exec'.
// by: H. Jack Sept., 16, 2002

// System component values
mass = 10;
force = 100;

x0 = 8;                // initial conditions
v0 = 12;
X=[x0, v0];

// define the state matrix function
// the values returned are [x, v]
function foo=f(state,t)
    foo = [ state($, 2), force/mass]; // d/dt x = v, d/dt v = F/M
endfunction

// Set the time length and step size for the integration
steps = 100;
t_start = 1;
t_end = 100;
h = (t_end - t_start) / steps;

// Loop for integration
for i=1:steps,
    X = [X ; X($,:) + h*f(X, i*h)];
end
printf("The value at the end of first order integration is (x, v) = (%f, %f)\n", ...
    X($,1),          ...
    X($,2));

// Explicit equation
function x=position(x0, v0, a0, t)
    x = (0.5 * a0 * t^2) + (v0 * t) + x0;
endfunction

function v=velocity(v0, a0, t)
    v = (a0 * t) + v0;
endfunction

```

- Integration continued,

```
printf("The value with integration is (x, v) = (%f, %f)\n", ...
      position(x0, v0, force/mass, t_end), ...
      velocity(v0, force/mass, t_end));

//
// The results should be
//   first order integration = (49710, 1002)
//   explicit                = (51208, 1012)
//
// The difference is 1498 for position and 10 for velocity. This is relatively small, but
// shows a clear case of the innacuracy of the numerical solutions.
//
// Note: increasing the number of steps increases the accuracy
//
```

- A simple Bode plot example Program (for electricals),

```

steps_per_dec = 6;
decades = 6;
start_freq = 0.1;

// The transfer function
function foo=G(w)
    D = %i * w;
    foo = (D + 5) / (D^2 + 100*D + 10000);
endfunction

fd = mopen("data.txt", "w");
for step = 0:(steps_per_dec * decades),
    f = start_freq * 10 ^ (step/steps_per_dec); // calculate the next frequency
    w = f * (2 * %pi); // convert the frequency to radians
    [gain, phase] = polar(G(w)); // get the result and convert to magnitude/angle
    gaindb = 20*log10(gain); // convert to dB
    phasedeg = 180 * phase / %pi; // convert to degrees
    fprintf(fd, "%f, %f, %f\n", f, gaindb, phasedeg);
end
fclose(fd);

// To graph it directly
D=poly(0,'D');
h=syslin('c', (D + 5) / (D^2 + 100*D + 10000) );
bode(h, 0.1, 1000, 'Sample Transfer Function');

```

## 34.4 Summary

## 34.5 Problems

1. Write a program that chooses a random number between 1 and 100. A user will then have to guess the value. Each guess will be given a hint for higher/lower. At the end the game should report the number of guesses required.

## 34.6 Challenge Problems