Lecture 29

Instructor Lecture and Matlab Demonstration:

Use of variable step size RK45 method via ode45() (see fishh.m and ypfishh.m)

\[
y0 = [10]; \quad \% \text{ initial population} \\
t0 = 0; \quad \% \text{ initial time} \\
tf = 10; \quad \% \text{ final time} \\
[t \ y] = \text{ode45('ypfishh',}[t0 \ tf],y0); \\
\text{plot}(t,y)
\]

where

\[
\text{function output = ypfishh}(t,y) \\
\quad \text{output}(1) = (.01)*y(1)*(100 - y(1)) - 5.0;
\]

Generates:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>45x1</td>
<td>360</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>t0</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>tf</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>45x1</td>
<td>360</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>y0</td>
<td>1x1</td>
<td>8</td>
<td>double</td>
<td></td>
</tr>
</tbody>
</table>

![Fish Population with Harvesting](image)
Compiled Code and Matlab Code:

Fortran requires extra steps of gathering memory, subroutine libraries and creating executable files *.exe. It does not have many of the built in Matlab features such as graphics, symbolic computation and other toolboxes, but the execution time for most compiled code is significantly faster.

Fortran syntax is different from Matlab:

- `!` is used in place of `%` for “comments”
- The data types must be declared such as
  ```fortran
  real, dimension (1:21,1:151) :: u
  real, dimension (1:151) :: times
  real, dimension (1:21) :: x
  real :: L,T,dt,dx,vel,decay,a,pi
  integer :: maxk,n,i,k
  ```
- The loops and if-else are a little different as in
  ```fortran
  do i = 1,n+1 ! initial concentration
    x(i) = (i-1)*dx
    if (i <= n/2+1) then
      u(i,1) = sin(pi*x(i)*2)
    else
      u(i,1) = 0.2
    end if
  end do
  ```
- The output uses formatted statements to files as in
  ```fortran
  open(6,file = 'c:\bob\outflow1d')
  do k = 1,maxk ! times loop
    do i = 2,n+1 ! space loop
      u(i,k+1) = a*u(i,k) + vel*dt/dx*u(i-1,k)
    end do
    write(6,'(21f12.4)') (u(i,k+1),i=1,n+1)
    write(*,'(21f12.4)') (u(i,k+1),i=1,n+1)
  end do
  ```

Student Matlab Computations:

Examine the flow1d.f90 and flow1d.m, which are similar to poll1d.m

Go to moodle, login, choose ma116 and review 192 multiple choice questions.