EL2310 – Scientific Programming

Lecture 5: Programming in Matlab

Yasemin Bekiroglu
(yaseminb@kth.se)

Royal Institute of Technology – KTH

Overview

Wrap Up

- Last time
  - `for` and `while` loops
  - `if` and `switch` branching
  - `nargin` and `nargout`
- Today
  - Last lecture on MATLAB

`nargin` and `nargout`

- Can check how many input and output arguments were given
- `nargin`: number of inputs arguments
- `nargout`: number of output arguments
- Typically:
  - Let `nargin` and `nargout` define what is done
  - Check `nargin` and give default values if not given

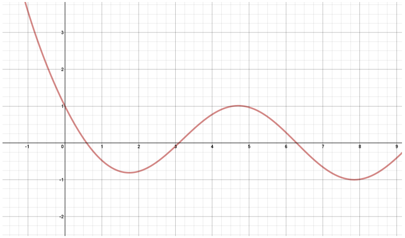
Skip to next iteration

- Sometimes you want to break out of a repetition
- Use `break` command
- Will continue after the end statement of the `for/while` loop

- Sometimes you want to start the next iteration
- Use `continue` command
- Will go up to the `for/while` statement again

Task 4.4

- Write a function that finds a solution to: $f(x) = e^{-x} - sin(x) = 0$



- Newton's method: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
- Assume initial guess x_0 is given
- Iterate at most `maxit` time
- Stop if $|x_n - x_{n-1}| \leq tol$

Task 1

- Modify the Newton method from Task 4.4 to break when we are done.

Subfunctions

- Can have many functions in an m-file
- Only one function is the *primary function*
- Subfuctions begin with a new function header
- Subfunctions cannot be called from outside, only from other subfunctions and the primary function
- Only the primary function can be called from outside

Why subfunctions?

- Can make the code easier to read/write
- Can have everything in one file
- Encapsulation
- Remember that only primary function can be called from outside

Task 2

- Rewrite the Newton code so that the code to calculate f(x) and f’(x) are in a subfunction.

Passing functions as arguments

- In the Newton method task from last time we would have to write a new primary function for every new function we would like to solve
- Can be avoided by instead passing a function name as an argument

‘Old style’ syntax

- Call function `B` with function `A` as argument
- Old style: `B('A')` - passing function name

```
function A(x)
  <commands>
end

function B(fcn)
  feval(fcn, <args>)
end
```

'New style' syntax

- New style: `B(@A)` - passing function handle

```
function A(x)
  <commands>
end

function B(fcn)
  fcn(<args>)
end
```

Task 3

- Re-implement the Newton function (Task 4.4) with function as argument
- Now we can solve any $f(x) = 0$ assuming we define a function that returns evaluation of $f(x)$ and $f'(x)$

Symbolic manipulation

- Matlab (with the right toolbox) can also do symbolic calculations
- Declare symbol with e.g. `syms t`
- Example

  `syms t` - Declare symbolic variable

  `f = t*sin(t)`

  `diff(f,'t')` - Differentiation

  `subs(f,'t',3)` - Substitution

Profiling

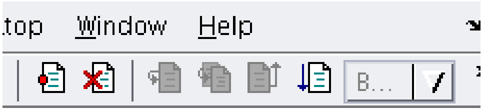
- Often useful to be able to tell what takes time in your program
- Can use `profile`
- `profile on` - Starts profiler
- `profile off` - Stops profiler
- `profile viewer` - Displays results
- For more info do `help profile`
- Use `fcn_busy` as a function in our Newton task and profile the code!

Debugging

- Very rare that you get everything right immediately
- Debugging often accomplished by printing intermediate results
- Compare outputs with expected values

Debugging continued

- The MATLAB editor has debugging support so that you can step through the code to see what happens
- You can set ‘breakpoint(s)’
  - Program will stop at the breakpoint
  - which will allow you to check variables, etc.
- Step line by line
- Step in/out of functions
- …



Making movies with MATLAB

- Check out
  - `frame=getframe(figure_handle)` Get a movie frame
  - `movie(frames)` Play movie frames
  - `movie2avi(frames)` Make a movie file
  - `avi=avifile(filename)` Create AVI file
  - `addframe(avi, frame)` Add a frame to the AVI file

Task 4 (let's see how)

- Make a movie for `surf(X,Y,Z)` with `Z=sin(X-x0)` for varying `x0`

Movie making example

```
[X,Y] = meshgrid(0:0.1:10,0:0.1:10);
n = 0;
for x0 = 0:0.1:10
  Z = sin(X-x0);
  surf(X,Y,Z)
  n = n + 1;
  F(n) = getframe(gcf);
  drawnow
end
movie2avi(F, 'sinmovie.avi','FPS',10)
```