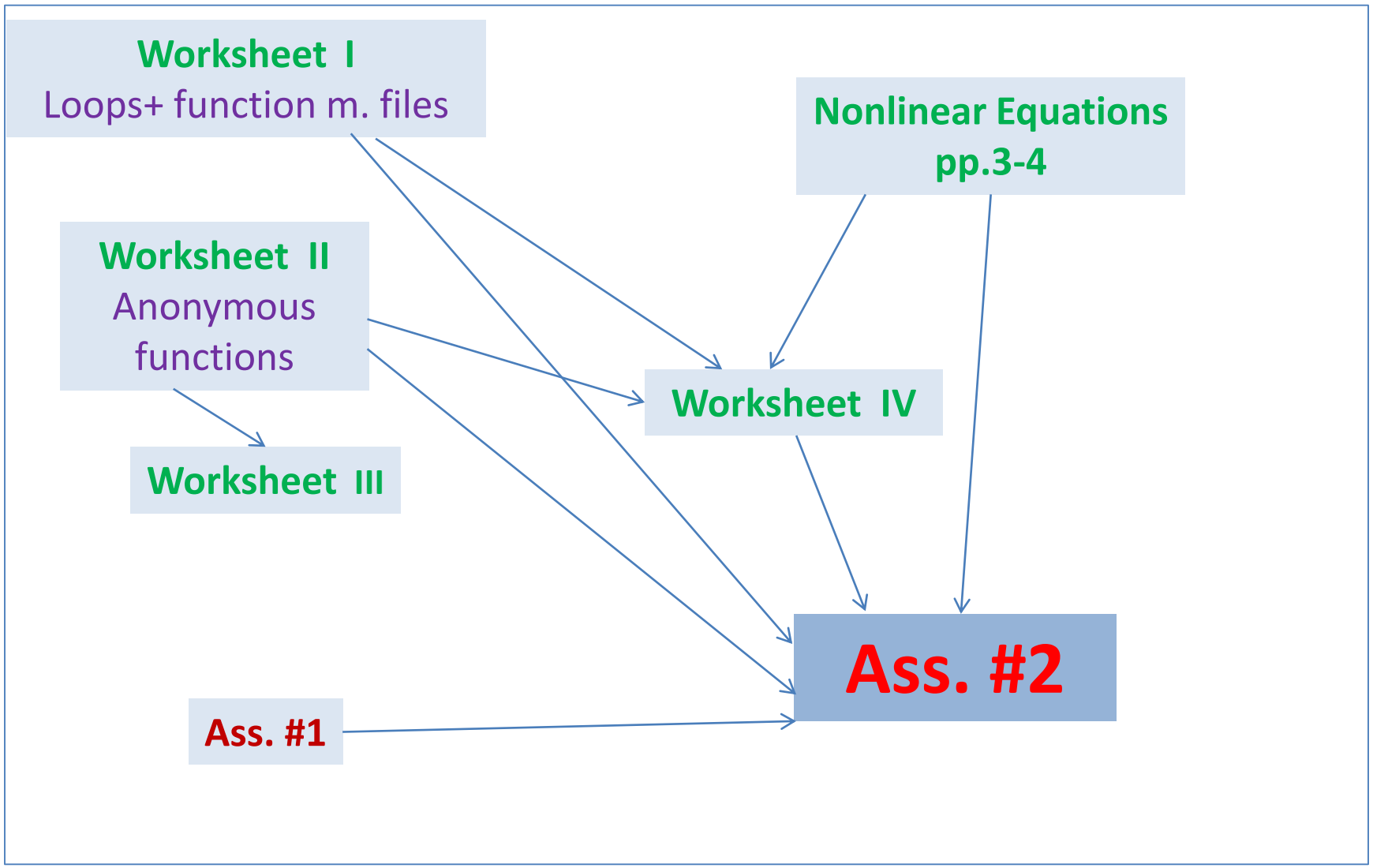
MS6021 Scientific Computing

Lecture/Worksheet #2

+

Assignment #2 (due end of Week 6)


Worksheet I
Loops+ function m. files
Nonlinear Equations
pp.3-4
Worksheet II
Anonymous
functions
Worksheet IV
Worksheet III
Ass. #2
Ass. #1

Computational Maths:

NON-Linear Equations (and Systems)

- Fixed-point iteration:
  —Rewrite the equation $f(x) = 0$ in the form $x = g(x)$
  —Choose some initial guess x^0
  —Generate the sequence by $x^{n+1} = g(x^n)$

- Newton's Method (also called Newton-Raphson method) – discuss for single equations and systems
  - Choose some initial guess x^0
  - Generate the sequence by $x^{n+1} = x^n + v^{n+1}$ where $f(x^n) + v^{n+1} f'(x^n) = 0$

- Damped Newton Method: similar, but $x^{n+1} = x^n + \alpha^{n+1} v^{n+1}$,
  where $0 < \alpha^{n+1} \leq 1$ is "optimal"…

READING:

- Natalia's MA4402 notes
  http://www.staff.ul.ie/natalia/MA4402/MA442_lec_Numerics.pdf
  (**part 4** on Fixed-Point Iteration and Newton's Method)
- *https://ocw.mit.edu/courses/chemical-engineering/10-34-numerical-methods-applied-to-chemical-engineering-fall-2015/lecture-notes/MIT10_34F15_Lec08.pdf*
  *(**pp. 1-8** on Newton's method; **pp. 23-27** on Damped Newton's method…)*
- *Programming for Computations - MATLAB/Octave*
  *(**Chapter 6** on Newton's and other methods + MatLab implementation)*

Stopping Criterion

- One can use either

$$| x^{n+1} - x^n | < TOL$$

or

$$|f(x^{n+1})| < TOL$$

- HOWEVER, I recommend to STOP after BOTH

$$| x^{n+1} - x^n | < TOL \quad \text{and} \quad |f(x^{n+1})| < TOL$$

Initial Guess: x^0

NOTE: most methods for non-linear equations are SENSITIVE w.r.t. the initial guess…

(In particular, Newton's method…)

Further MatLab skills

- *Worksheet-I*
  An Introduction to Matlab by David F. Griffiths:
  Sections **18+19+20+22+21.3+21.5+23**
  — now, among other things, we shall employ
  LOOPs and **function m. files**

- *Worksheet-II*
  Anonymous functions (see below; for easy creation of function handles…)

Worksheet II on **Anonymous functions**

What Are Anonymous Functions?

An anonymous function is a function that is *not* stored in a program file, but is associated with a variable whose data type is **function_handle.** Anonymous functions can accept inputs and return outputs, just as standard functions do.

EXAMPLE: create a handle to an anonymous function that finds the square of a number:

```
sqr = @(x) x.^2;
```

Variable sqr is a function handle.
The @ operator creates the handle, and the parentheses () immediately after the @ operator include the function input arguments.
This anonymous function accepts a single input x, and implicitly returns a single output, an array the same size as x that contains the squared values.

Find the square of a particular value (5) by passing the value to the function handle, just as you would pass an input argument to a standard function.

```
a = sqr(5)
    a = 25
```

Many MATLAB® functions accept function handles as inputs so that you can evaluate functions over a range of values. You can create handles either for anonymous functions or for functions in program files.

EXAMPL:E: find the integral of the sqr function from 0 to 1 by passing the function handle to the integral function:

```
q = integral(sqr,0,1);
```

You do not need to create a variable in the workspace to store an anonymous function. Instead, you can create a temporary function handle within an expression, such as this call to the integral function:

```
q = integral(@(x) x.^2,0,1);
```

Variables in the Expression

EXAMPLE: create a function handle to an anonymous function that requires coefficients a, b, and c.

```
a = 1.3; b = .2; c = 30;
parabola = @(x) a*x.^2 + b*x + c;
```

Because a, b, and c are available at the time you create parabola, the function handle includes those values. The values persist within the function handle even if you clear the variables:

```
clear a b c
x = 1; y = parabola(x)
        y = 31.5000
```

To supply different values for the coefficients, you must create a new function handle:

```
a = -3.9; b = 52; c = 0;
parabola = @(x) a*x.^2 + b*x + c; x = 1;
y = parabola(1)
        y = 48.1000
```

Multiple Anonymous Functions

The expression in an anonymous function can include another anonymous function.

EXAMPLE: you can solve the equation for varying values of c by combining two anonymous functions:

```
g = @(c) (integral(@(x) (x.^2 + c*x + 1),0,1));
```

Here is how to derive this statement:

Write the integrand as an anonymous function,

```
@(x) (x.^2 + c*x + 1)
```

Evaluate the function from zero to one by passing the function handle to integral,

```
integral(@(x) (x.^2 + c*x + 1),0,1)
```

Supply the value for c by constructing an anonymous function for the entire equation,

```
g = @(c) (integral(@(x) (x.^2 + c*x + 1),0,1));
```

The final function allows you to solve the equation for any value of c. For example:

```
g(2)
        ans = 2.3333
```

Functions with No Inputs

If your function does not require any inputs, use empty parentheses when you define and call the anonymous function. EXAMPLE:

```
t = @() datestr(now);
d = t()
```

```
d = 26-Jan-2012 15:11:47
```

NOTE: Omitting the parentheses in the assignment statement creates another function handle, and does not execute the function:

```
d = t
```

```
d = @() datestr(now)
```

Functions with Multiple Inputs

Anonymous functions require that you explicitly specify the input arguments as you would for a standard function, separating multiple inputs with commas.

EXAMPLE: this function accepts two inputs, x and y:

```
myfunction = @(x,y) (x^2 + y^2 + x*y);
x = 1; y = 10;
z = myfunction(x,y)
```

```
z = 111
```

MatLab routine for finding a zero of a function of one variable:

fzero

- *Worksheet-III*

See pp. 164-167 in
[Higham & Higham, MatLab Guide]

Implementation of methods for nonlinear singe equations

Worksheet-IV

- Write a function m. file implementing the Fixed-Iteration method for x=g(x) with the following inputs and outputs:
  [result, iterations_performed, flag] = FixedIter (g , x0, TOL, k_MAX)
  where

  g = function handle ;
  x0 = initial guess ;
  TOL = the tolerance to be used in the stopping criterial ;
  k_MAX = the maximum number of iterations

  flag =1 if TOL was attained (i.e. success), 0 otherwise.

  Write **2 versions** of this function: using a (i) **while** loop; (ii) **for** loop.

- Write a function m. file implementing the Newton's method for f(x)=0 with the following inputs and outputs:
  [result, iterations_performed, flag] = myNewton (f , f1, x0, TOL, k_MAX)

  f = function handle for f

  f1 = function handle for f'

NOTE: you may consult (NOT COPY) *Programming for Computations - MATLAB/Octave (Chapter 6 on Newton's and other methods + MatLab implementation) --- ideally AFTER you have your own code.*

Worksheet-IV (continued)

- Use the above m. functions to solve the examples from Natalia's MA4402 notes
  http://www.staff.ul.ie/natalia/MA4402/MA442_lec_Numerics.pdf
  (**part 4** on Fixed-Point Iteration and Newton's Method)
  Compare the numbers of iterations for various values of x0 and TOL.

- Exercise 6.1 from
  [*Programming for Computations - MATLAB/Octave]*
  Would Damped Newton's method work better?

- Exercise 6.6 from
  [*Programming for Computations - MATLAB/Octave]*

Assignement #2, 20%, by end of Week 6

- Consider the semi-linear problem

$$-u'' + f(x,u) = 0, \quad u(0)=u(1)=0.$$

- Here we consider 3 particular cases:

(A) $$f(x,u) = \frac{u-\sin(x)}{5-u} - \exp(-3x)$$

(B) $$f(x,u) = 2(u^4 - 1) - x^2 e^{5x}$$

(C) $$f(x,u) = 50(u^4 - 1) - x^2 e^{5x}$$

NOTE: exact solutions are unknown (unlike the problem in Assignment #1, part I)

- We discretize **as in Assignment #1**, BUT now we have a **non-linear term**!
- To deal with the **non-linear part** in $-u'' + f(x,u) = 0$, use

  (1) a version of the Fixed-point iteration :

$$-(u^{n+1})'' + f(x,u^n) = 0, \qquad u^{n+1}(0)= u^{n+1}(1)=0.$$

  (2) Newton's method:

$$-(u^{n+1})'' + f(x,u^n) + f_u(x,u^n)(u^{n+1}-u^n) = 0, \qquad u^{n+1}(0)= u^{n+1}(1)=0.$$

  NOTE: it is advantageous to implement (2) using the following equivalent form with v^{n+1} *:*

$$-(v^{n+1})'' + f_u(x,u^n)\, v^{n+1} -(u^n)'' + f(x,u^n) = 0, \quad v^{n+1}(0)= v^{n+1}(1)=0$$

$$u^{n+1}=u^n + v^{n+1}$$

  (3) Damped Newton's method:

$$u^{n+1}=u^n + \alpha^{n+1} v^{n+1},$$

  where α^{n+1} *is, e.g., in* $\{0.1,\ 0.2, \ldots, 0.9, 1\}$*, such that a discrete version of*

$$\max_{0<x<1} |-(u^n + \alpha^{n+1}v^{n+1})'' + f(x,\, u^n + \alpha^{n+1}v^{n+1})| \text{ attains its } \mathbf{MIN}.$$

- *INITIAL guess :* $u^0(x) = 0$ *for all x.*

[continued on next page

Assignement #2 (continued)

SUBMISSION:

- Description of the methods (details of implementation hand-written/a file form)
- m. file(s)
- Discussion of the computed solutions and performance of the methods:
  - Graphs of computed solutions (for a few values of N)
  - How many iterations are required for each of the 3 problems with each of the 3 methods to attain TOL= 10^{-6} with N=100 and N=1000.
    For example, you may present your results in the form of a Table:

	Problem A	Problem B	Problem C	Problem A	Problem B	Problem C
Fixed Iteration						
Newton						
Damped Newton						
	N=100			N=1000		