UNIVERSITY OF BAHRAIN

College of Engineering

Department of Electrical & Electronics Engineering

EENG207: NUMERICAL METHODS AND COMPUTER APPLICATIONS

2nd Semester: 2023/2024

SAMPLE MIDTERM

Q1. *10 marks*

The Discrete Fourier Transform Algorithm
Understanding the FFT Algorithm: The Cooley-Tukey FFT algorithm

FAST: Computing the FAST Discrete Fourier Transform (DFT to FFT: Exploiting Symmetry):
For the algorithm given by Cooley and Tukey, they have shown that it's possible to divide the DFT computation into two smaller parts.
Therefore, from the definition of the DFT we have:

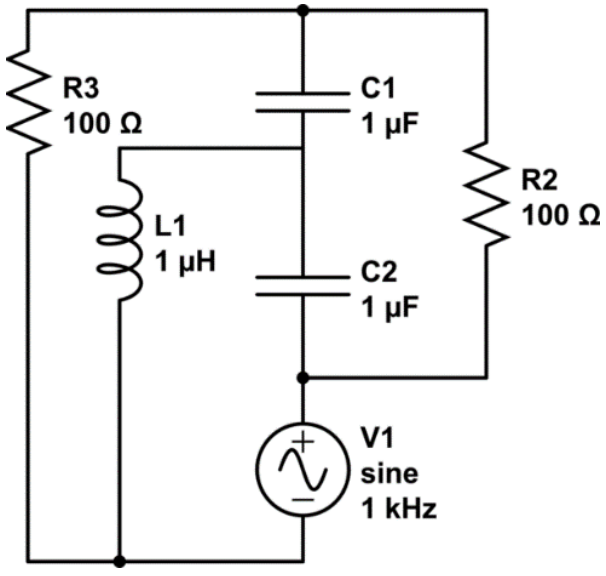
$$X_k = \sum_{m=0}^{N/2-1} x_{2m} . e^{-i2\pi km/(\frac{N}{2})} + e^{-i2\pi k/N} \sum_{m=0}^{N/2-1} x_{2m+1} . e^{-i2\pi km/(\frac{N}{2})}$$

- Write a computer algorithm using (*matlab, or python or c++*) to achieve the Discrete Fourier Transform Algorithm as expressed above.

Q2. 10 marks

For the circuit shown in below (*Resonance frequency of an RLC circuit*), .. write a computer code to plot the frequsency respsonse across the L1 and the V1.

- Show your hand caculations. (2 marks).
- Write a computer script for plaotting the Magniute and Phase as frqeeucney respsoes. (8 marks).

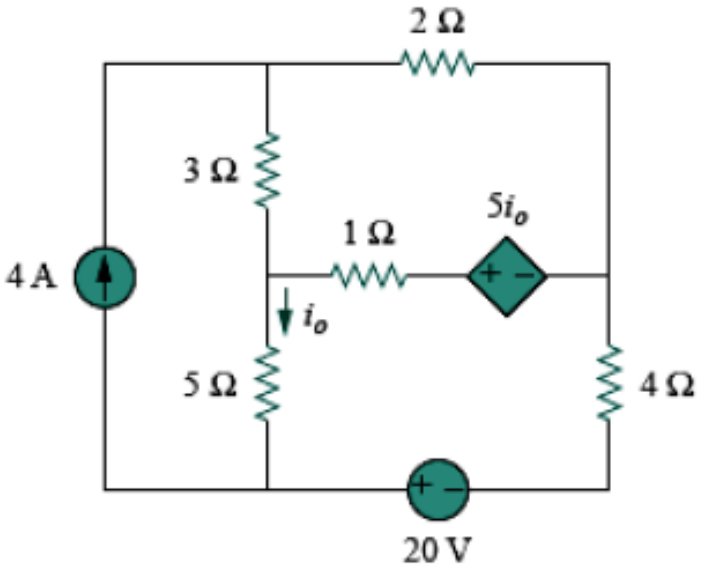


Quoted picture from:
https://electronics.stackexchange.com/questions/217772/resonance-frequency-of-an-rlc-circuit

Q3. 10 marks

PART A: For the below resistive circuit, … : (5 marks).

- Write the loops equations, then write down a computer scrip/ code that solves for the loop's currents:



Quoted picture from:
https://www.electronics-tutorials.ws/accircuits/parallel-circuit.html

PART B: Translate the below to a computer code: : (5 marks).

Tabulate the functions

$$y = (x^2 + 3)\sin \pi x^2$$

and

$$z = \sin^2 \pi x/(x^{-2} + 3)$$

for $x = 0, 0.2, \ldots, 10$. *Hence, tabulate the function*

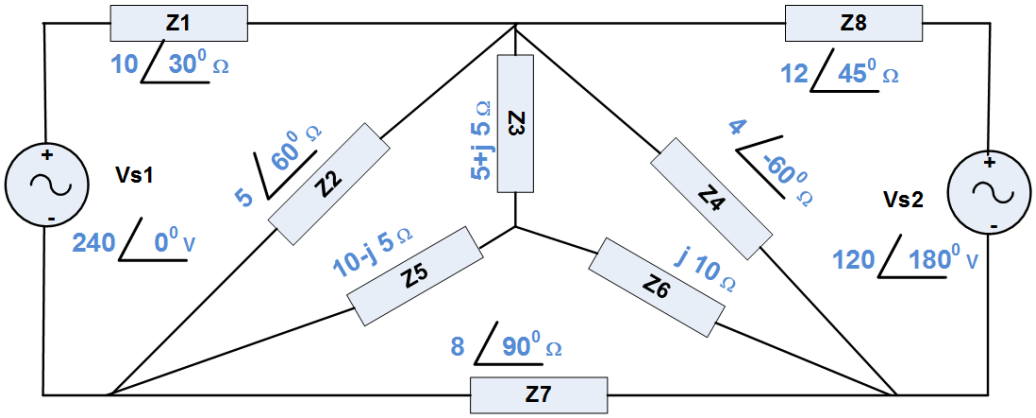
$$w = \frac{(x^2 + 3)\sin \pi x^2 \sin^2 \pi x}{(x^{-2} + 3)}.$$

Plot a graph of w *over the range* $0 \leq x \leq 10$.

Q4. *10 marks*

For the circuit shown in below, write a code and MeshAnalysis to:

a) Find the real and reactive powers (indicate whether delivered or absorbed) by the voltage source Vs2.
b) Find the real and reactive powers consumed by the impedance Z7



Q5. *10 marks*

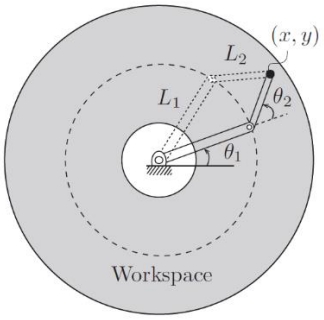
Using the concept of matrices (inversion), write down MATLAB computer script to solve the following system of three linear equations:

$$4x - 2.5y + 3z = 22.2$$
$$6x - 4y - 2z = 16.6$$
$$2x + 4y - 5.5z = -11.3$$

Hint: *use the matrix method.*

Q6. *10 marks*

Robotics Inverse Kinematics Using Newton Raphson Algorithm. Use the Newton Raphson Algorithm, to find the Inverse Kinematics for the following robotics system, that is moving in planner path.



$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \\ L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \end{pmatrix} = f(\theta_1 + \theta_2)$$

- Show your hand calculations, for only three steps (including the initial value) (5)
- Write a computer script code, (your written code to solve this robotics inverse kinematics problem).

Q7. *10 marks*

Using GJ iterative method, solve for the $f(\omega, \delta)$, and for any given - nonzero - initial values,

$f(\omega, \delta) = \omega + e^{-\omega} + \delta^3$,

$$g(\omega, \delta) = \omega^2 + 2\omega\delta - \delta^2 + \tan(\omega)$$

- Find the solution within 4 steps. Show your numerical hand steps. (5 marks)
- Write a script-code (*matlab, c++, or python*) to show how to solve the same problem numerically. (5 marks)

Q8. *10 marks*

For initial point chosen by you, use the secant algorithm to find a solution for $f(\beta) = \beta e^{\beta} - \pi r^2 0.84$, r=1.007. (5)
Write computer code for the Secant algorithm. (5)

Q9. *10 marks*

For initial point chosen by you, use the bisction algorithm to find a solution for $f(\beta) = \beta e^{\beta} - \pi r^2 0.84$, r=1.007. (5)
Write computer code for the Secant algorithm. (5)

NUMERICAL ANALYSIS

$$b_n - a_n = \frac{b_0 - a_0}{2^n}$$

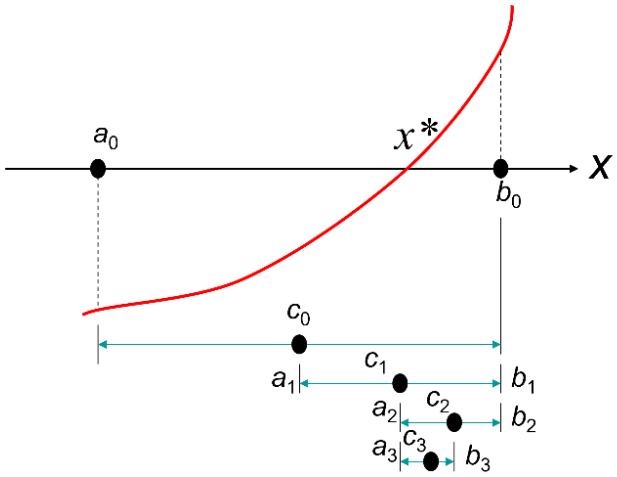
$$\left|c_n - x^*\right| \le c_n - a_n$$

$$\le \frac{b_0 - a_0}{2^{n+1}}$$

For the previous example

$$\frac{0.4}{2^{n+1}} < 10^{-4}$$

$$n = 11$$



Number of Iterations

$$\left|c_n - x^*\right| \le c_n - a_n \qquad \le \frac{b_0 - a_0}{2^{n+1}}$$

Newton Raphson Method

$$x_n = x_{n-1} - \frac{f\left(x_{n-1}\right)}{f'\left(x_{n-1}\right)}$$

Secant Method

$$x_n = \frac{x_{n-1} f\left(x_{n-2}\right) - x_{n-2} f\left(x_{n-1}\right)}{f\left(x_{n-2}\right) - f\left(x_{n-1}\right)}$$

False Position

$$x_n = \frac{x_{n-1} f\left(x_{n-2}\right) - x_{n-2} f\left(x_{n-1}\right)}{f\left(x_{n-2}\right) - f\left(x_{n-1}\right)}$$

Newton Raphson Method

(multivariable)

$$\begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} = \begin{bmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \end{bmatrix} - \underline{J}^{-1}\left(x_1^{(k-1)}, x_2^{(k-1)}\right) \begin{bmatrix} f_1\left(x_1^{(k-1)}, x_2^{(k-1)}\right) \\ f_2\left(x_1^{(k-1)}, x_2^{(k-1)}\right) \end{bmatrix}$$

$$\underline{x}^{(k)} = \underline{x}^{(k-1)} - \underline{J}^{-1}\left(\underline{x}^{(k-1)}\right) \underline{f}\left(\underline{x}^{(k-1)}\right)$$

ITERATIVE METHODS (I): Gauss-Jacobi Method

$$\begin{aligned}
x_1^{(k)} &= \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \cdots\cdots - a_{1n}x_n^{(k-1)}\right) \\
x_2^{(k)} &= \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(k-1)} - a_{23}x_3^{(k-1)} - \cdots\cdots - a_{2n}x_n^{(k-1)}\right) \\
x_3^{(k)} &= \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^{(k-1)} - a_{32}x_2^{(k-1)} - \cdots\cdots - a_{3n}x_n^{(k-1)}\right) \\
&\vdots \\
x_n^{(k)} &= \frac{1}{a_{nn}}\left(b_n - a_{n1}x_1^{(k-1)} - a_{n2}x_2^{(k-1)} - \cdots\cdots - a_{n,n-1}x_{n-1}^{(k-1)}\right)
\end{aligned}$$

ITERATIVE METHODS (I): Gauss-Seidel Method

$$\begin{aligned}
x_1^{(k)} &= \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^{(k-1)} - a_{13}x_3^{(k-1)} - \cdots\cdots - a_{1n}x_n^{(k-1)}\right) \\
x_2^{(k)} &= \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k-1)} - \cdots\cdots - a_{2n}x_n^{(k-1)}\right) \\
x_3^{(k)} &= \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)} - \cdots\cdots - a_{3n}x_n^{(k-1)}\right) \\
&\vdots \\
x_n^{(k)} &= \frac{1}{a_{nn}}\left(b_n - a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} - \cdots\cdots - a_{n,n-1}x_{n-1}^{(k)}\right)
\end{aligned}$$

Convergence of Iterative Methodz; Gauss-Jacobi

$\underline{x}(k) = \underline{D}\text{-}1\underline{B} - \underline{D}\text{-}1(\underline{L}+\underline{U})\,\underline{x}(k\text{-}1)$ $\qquad$ $\underline{T} = -\,\underline{D}\text{-}1(\underline{L}+\underline{U})$ $\qquad$ $\underline{C} = \underline{D}\text{-}1\underline{B}$

Gauss-Seidel

$$\underline{x}^{(k)} = (\underline{L}+\underline{D})^{-1}\underline{B} - (\underline{L}+\underline{D})^{-1}\underline{U}\,\underline{x}^{(k-1)} \qquad \underline{T} = -(\underline{L}+\underline{D})^{-1}\underline{U} \qquad \underline{C} = (\underline{L}+\underline{D})^{-1}\underline{B}$$

$$\sqrt{\rho\left(\underline{T}^{T}\,\underline{T}\right)} < 1$$

Optimal Relaxation Factor

Gauss-Jacobi

$$\underline{T} = \underline{I} - \omega\,\underline{D}^{-1}\,\underline{A}$$

Gauss-Seidel

$$\underline{T} = (\underline{D} + \omega\,\underline{L})^{-1}\,[(1-\omega)\underline{D} - \omega\underline{U}\,]$$

$$\omega_{opt} \approx \frac{2}{1+\sqrt{1-\left[\rho\left(\underline{T}_j\right)\right]^2}}$$

Gauss-Jacobi Method

$$\underline{x}^{(k)} = \underline{D}^{-1}\,\underline{B} - \underline{D}^{-1}\,(\underline{L}+\underline{U})\,\underline{x}^{(k-1)}$$

Gauss-Seidel Method

$$\underline{x}^{(k)} = (\underline{D}+\underline{L})^{-1}\,\underline{B} - (\underline{D}+\underline{L})^{-1}\,\underline{U}\,\underline{x}^{(k-1)}$$

Curve-Fitting

$$\begin{bmatrix} \sum_{k=1}^{N} 1 & \sum_{k=1}^{N} x_k & \sum_{k=1}^{N} x_k^2 & \cdots & \sum_{k=1}^{N} x_k^n \\ \sum_{k=1}^{N} x_k & \sum_{k=1}^{N} x_k^2 & \sum_{k=1}^{N} x_k^3 & \cdots & \sum_{k=1}^{N} x_k^{n+1} \\ \sum_{k=1}^{N} x_k^2 & \sum_{k=1}^{N} x_k^3 & \sum_{k=1}^{N} x_k^4 & \cdots & \sum_{k=1}^{N} x_k^{n+2} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \sum_{k=1}^{N} x_k^n & \sum_{k=1}^{N} x_k^{n+1} & \sum_{k=1}^{N} x_k^{n+2} & \cdots & \sum_{k=1}^{N} x_k^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{N} y_k \\ \sum_{k=1}^{N} y_k x_k \\ \sum_{k=1}^{N} y_k x_k^2 \\ \vdots \\ \sum_{k=1}^{N} y_k x_k^n \end{bmatrix}$$

LAGRANGE APPROXIMATION

$$P(x)=\sum_{i=0}^{n} L_i(x)\, y_i \qquad L_k(x)=\prod_{\substack{i=0\\ i\neq k}}^{n}\frac{x-x_i}{x_k-x_i}$$

INTERPOLATION AND POLYNOMIAL APPROXIMATION

Newton Polynomial

Special case:

When $x_0, x_1, \ldots, x_n$ are arranged consecutively with equal spacing $h = x_{n+1} - x_n$

x	f	1st Divided differences	2nd Divided differences	3rd Divided differences	4th Divided differences
x_0	f_0	$\frac{f_1-f_0}{h}=\frac{\Delta f_0}{h}$	$\frac{\frac{\Delta f_1}{h}-\frac{\Delta f_0}{h}}{2h}=\frac{\Delta^2 f_0}{2h^2}$	$\frac{\frac{\Delta^2 f_1}{2h^2}-\frac{\Delta^2 f_0}{2h^2}}{3h}=\frac{\Delta^3 f_0}{6h^3}$	$\frac{\Delta^4 f_0}{24h^4}$ .....
x_1	f_1	$\frac{f_2-f_1}{h}=\frac{\Delta f_1}{h}$	$\frac{\frac{\Delta f_2}{h}-\frac{\Delta f_1}{h}}{2h}=\frac{\Delta^2 f_1}{2h^2}$	⋮	⋮
x_2	f_2	$\frac{f_3-f_2}{h}=\frac{\Delta f_2}{h}$	⋮		
x_3	f_3				
⋮	⋮	⋮	⋮	⋮	⋮

$$P(x)=a_0+a_1(x-x_0)+a_2(x-x_0)(x-x_1)+\cdots\cdots+a_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})$$

Table A.2: **Array operators**

Character	Description
.*	Array multiplication
./	Array (right) division
.^	Array power
.\	Array (left) division
.′	Array (nonconjugated) transpose

Table A.3: **Relational and logical operators**

Character	Description
$<$	Less than
$\leq$	Less than or equal to
$>$	Greater than
$\geq$	Greater than or equal to
$==$	Equal to
$\sim=$	Not equal to
&	Logical or element-wise AND
|	Logical or element-wise OR
&&	Short-circuit AND
||	Short-circuit OR

Table A.4: **Managing workspace and file commands**

Command	Description
`cd`	Change current directory
`clc`	Clear the Command Window
`clear (all)`	Removes all variables from the workspace
`clear x`	Remove x from the workspace
`copyfile`	Copy file or directory
`delete`	Delete files
`dir`	Display directory listing
`exist`	Check if variables or functions are defined
`help`	Display help for MATLAB functions
`lookfor`	Search for specified word in all help entries
`mkdir`	Make new directory
`movefile`	Move file or directory
`pwd`	Identify current directory
`rmdir`	Remove directory
`type`	Display contents of file
`what`	List MATLAB files in current directory
`which`	Locate functions and files
`who`	Display variables currently in the workspace
`whos`	Display information on variables in the workspace

Table A.5: **Predefined variables and math constants**

Variable	Description
`ans`	Value of last variable (answer)
`eps`	Floating-point relative accuracy
`i`	Imaginary unit of a complex number
`Inf`	Infinity (∞)
`eps`	Floating-point relative accuracy
`j`	Imaginary unit of a complex number
`NaN`	Not a number
`pi`	The number π (3.14159...)

Table A.6: **Elementary matrices and arrays**

Command	Description
`eye`	Identity matrix
`linspace`	Generate linearly space vectors
`ones`	Create array of all ones
`rand`	Uniformly distributed random numbers and arrays
`zeros`	Create array of all zeros

Table A.7: **Arrays and Matrices: Basic information**

Command	Description
`disp`	Display text or array
`isempty`	Determine if input is empty matrix
`isequal`	Test arrays for equality
`length`	Length of vector
`ndims`	Number of dimensions
`numel`	Number of elements
`size`	Size of matrix

Table A.8: **Arrays and Matrices: operations and manipulation**

Command	Description
`cross`	Vector cross product
`diag`	Diagonal matrices and diagonals of matrix
`dot`	Vector dot product
`end`	Indicate last index of array
`find`	Find indices of nonzero elements
`kron`	Kronecker tensor product
`max`	Maximum value of array
`min`	Minimum value of array
`prod`	Product of array elements
`reshape`	Reshape array
`sort`	Sort array elements
`sum`	Sum of array elements
`size`	Size of matrix

Table A.9: **Arrays and Matrices: matrix analysis and linear equations**

Command	Description
`cond`	Condition number with respect to inversion
`det`	Determinant
`inv`	Matrix inverse
`linsolve`	Solve linear system of equations
`lu`	LU factorization
`norm`	Matrix or vector norm
`null`	Null space
`orth`	Orthogonalization
`rank`	Matrix rank
`rref`	Reduced row echelon form
`trace`	Sum of diagonal elements

Appendix B

Release notes for Release 14 with Service Pack 2

B.1 Summary of changes

MATLAB 7 Release 14 with Service Pack 2 (R14SP2) includes several new features. The major focus of R14SP2 is on *improving* the quality of the product. This document doesn't attempt to provide a complete specification of every single feature, but instead provides a brief introduction to each of them. For full details, you should refer to the MATLAB documentation (Release Notes).

The following key points may be relevant:

1. **Spaces before numbers** - For example: `A* .5`, you will typically get a mystifying message saying that A was previously used as a variable. There are two workarounds:

   (a) Remove all the spaces:

   ```
   A*.5
   ```

   (b) Or, put a zero in front of the dot:

   ```
   A * 0.5
   ```

2. **RHS empty matrix** - The right-hand side must literally be the empty matrix [ ]. It cannot be a variable that has the value [ ], as shown here:

   ```
   rhs = [];
   A(:,2) = rhs
   ??? Subscripted assignment dimension mismatch
   ```

3. **New format option** - We can display MATLAB output using two *new* formats: `short eng` and `long eng`.

   - `short eng` – Displays output in *engineering* format that has at least 5 digits and a power that is a multiple of three.

     ```
     >> format short eng
     >> pi
     ans   =
           3.1416e+000
     ```

   - `long eng` – Displays output in *engineering* format that has 16 significant digits and a power that is a multiple of three.

     ```
     >> format long eng
     >> pi
     ans   =
           3.14159265358979e+000
     ```

4. **Help** - To get help for a *subfunction*, use

   ```
   >> help function_name>subfunction_name
   ```

   In previous versions, the syntax was

   ```
   >> help function_name/subfunction_name
   ```

   This change was introduced in R14 (MATLAB 7.0) but was not documented. Use the MathWorks Web site search features to look for the latest information.

5. **Publishing** - Publishing to LaTeXnow respects the image file type you specify in preferences rather than always using EPSC2-files.

   - The Publish image options in Editor/Debugger preferences for Publishing Images have changed slightly. The changes prevent you from choosing invalid formats.
   - The files created when publishing using cells now have more natural extensions. For example, JPEG-files now have a .jpg instead of a .jpeg extension, and EPSC2-files now have an .eps instead of an .epsc2 extension.
   - Notebook will no longer support Microsoft Word 97 starting in the next release of MATLAB.

6. **Debugging** - Go directly to a subfunction or using the enhanced **Go To** dialog box. Click the **Name** column header to arrange the list of function alphabetically, or click the **Line** column header to arrange the list by the position of the functions in the file.

B.2 Other changes

1. There is a new command `mlint`, which will scan an M-file and show inefficiencies in the code. For example, it will tell you if you've defined a variable you've never used, if you've failed to pre-allocate an array, etc. These are common mistakes in EA1 which produce runnable but inefficient code.

2. You can comment-out a block of code without putting `% at the beginning of each line`. The format is

```
%{

  Stuff you want MATLAB to ignore...

%}
```

   The delimiters `%{` and `%}` must appear on lines by themselves, and it may not work with the comments used in functions to interact with the help system (like the H1 line).

3. There is a new function `linsolve` which will solve $Ax = b$ but with the user's choice of algorithm. This is in addition to left division $x = A\backslash b$ which uses a default algorithm.

4. The `eps` constant now takes an optional argument. `eps(x)` is the same as the old `eps*abs(x)`.

5. You can break an M-file up into named cells (blocks of code), each of which you can run separately. This may be useful for testing/debugging code.

6. Functions now optionally end with the `end` keyword. This keyword is mandatory when working with nested functions.

B.3 Further details

1. You can *dock* and *un-dock* windows from the main window by clicking on an icon. Thus you can choose to have all Figures, M-files being edited, help browser, command window, etc. All appear as panes in a single window.

2. Error messages in the command window resulting from running an M-file now include a clickable link to the offending line in the editor window containing the M-file.

3. You can customize figure interactively (labels, line styles, etc.) and then automatically generate the code which reproduces the customized figure.

4. `feval` is no longer needed when working with function handles, but still works for backward compatibility. For example, `x=@sin; x(pi)` will produce `sin(pi)` just like `feval(x,pi)` does, but faster.

5. You can use function handles to create anonymous functions.

6. There is support for nested functions, namely, functions defined within the body of another function. This is in addition to sub-functions already available in version 6.5.

7. There is more support in arithmetic operations for numeric data types other than double, e.g. `single, int8, int16, uint8, uint32,` etc.

   Finally, please visit our webpage for other details:

   http://computing.mccormick.northwestern.edu/matlab/

Appendix C

Main characteristics of MATLAB

C.1 History

- Developed primarily by Cleve Moler in the 1970's
- Derived from FORTRAN subroutines LINPACK and EISPACK, linear and eigenvalue systems.
- Developed primarily as an interactive system to access LINPACK and EISPACK.
- Gained its popularity through word of mouth, because it was not officially distributed.
- Rewritten in C in the 1980's with more functionality, which include plotting routines.
- The MathWorks Inc. was created (1984) to market and continue development of MATLAB.

According to Cleve Moler, three other men played important roles in the origins of MATLAB: J. H. Wilkinson, George Forsythe, and John Todd. It is also interesting to mention the authors of LINPACK: Jack Dongara, Pete Steward, Jim Bunch, and Cleve Moler. Since then another package emerged: LAPACK. LAPACK stands for Linear Algebra Package. It has been designed to supersede LINPACK and EISPACK.

C.2 Strengths

- MATLAB may behave as a *calculator* or as a *programming language*
- MATLAB combine nicely calculation and graphic plotting.
- MATLAB is relatively easy to learn

- MATLAB is interpreted (not compiled), errors are easy to fix
- MATLAB is optimized to be relatively fast when performing matrix operations
- MATLAB does have some object-oriented elements

C.3 Weaknesses

- MATLAB is not a *general* purpose programming language such as C, C++, or FORTRAN
- MATLAB is designed for scientific computing, and is not well suitable for other applications
- MATLAB is an interpreted language, slower than a compiled language such as C++
- MATLAB commands are specific for MATLAB usage. Most of them do not have a direct equivalent with other programming language commands

C.4 Competition

- One of MATLAB's competitors is **Mathematica**, the *symbolic* computation program.
- MATLAB is more convenient for *numerical analysis* and *linear algebra*. It is frequently used in *engineering community*.
- Mathematica has superior symbolic manipulation, making it popular among *physicists*.
- There are other competitors:
  - **Scilab**
  - **GNU Octave**
  - **Rlab**

Bibliography

[1] The MathWorks Inc. *MATLAB 7.0 (R14SP2)*. The MathWorks Inc., 2005.

[2] S. J. Chapman. *MATLAB Programming for Engineers*. Thomson, 2004.

[3] C. B. Moler. *Numerical Computing with MATLAB*. Siam, 2004.

[4] C. F. Van Loan. *Introduction to Scientific Computing*. Prentice Hall, 1997.

[5] D. J. Higham and N. J. Higham. *MATLAB Guide*. Siam, second edition edition, 2005.

[6] K. R. Coombes, B. R. Hunt, R. L. Lipsman, J. E. Osborn, and G. J. Stuck. *Differential Equations with MATLAB*. John Wiley and Sons, 2000.

[7] A. Gilat. *MATLAB: An introduction with Applications*. John Wiley and Sons, 2004.

[8] J. Cooper. *A MATLAB Companion for Multivariable Calculus*. Academic Press, 2001.

[9] J. C. Polking and D. Arnold. *ODE using MATLAB*. Prentice Hall, 2004.

[10] D. Kahaner, C. Moler, and S. Nash. *Numerical Methods and Software*. Prentice-Hall, 1989.

[11] J. W. Demmel. *Applied Numerical Linear Algebra*. Siam, 1997.

[12] D. Houcque. Applications of MATLAB: Ordinary Differential Equations. *Internal communication, Northwestern University*, pages 1–12, 2005.