CE 30
Second Semester, 2004-2005
Options for Project 2
by: Luisito L. Agustin

Option 1:
CE 30 Programming Exercise 005
Numerical Integration using the Trapezoidal Rule

Quota: at most 4 students working individually or in groups

Option 2:
CE 30 Programming Exercise 006
Numerical Integration using Simpson's Rule

Quota: at most 4 students working individually or in groups

Option 3:
CE 30 Programming Exercise 007
Solution of Linear Systems using Gauss Elimination

Quota: at most 5 students working individually or in groups

Option 4:
CE 30 Programming Exercise 008
Matrix Inversion using Gauss-Jordan Elimination

Quota: at most 5 students working individually or in groups

Option 5:
CE 30 Programming Exercise 009
Solution of Linear Systems using LU Factorization by Doolittle's Method

Quota: at most 5 students working individually or in groups

Option 6:
CE 30 Programming Exercise 010
Solution of Linear Systems using LU Factorization by Crout's Method

Quota: at most 5 students working individually or in groups

Specifications for REALMATRIX and REALVEC files referred to in exercises 007 thru 010 may be found at the end of this document.

Reference:

[Kreyszig] Erwin Kreyszig: *Advanced Engineering Mathematics*, 8th ed, 1999.

CE 30 Programming Exercise 005
Numerical Integration using the Trapezoidal Rule

The main goal of this project is to implement the Trapezoidal Rule for computing definite integrals of functions of a single variable.

The specifications will require various parts of the implementation to be placed in separate source files for the sake of modularity. Hence header files and/or various declarations may be needed for data and functions in one source file to be accessible to code in another source file.

1. [5 pts] In a separate source file, define C/C++ functions implementing the following functions:

$$f_1(x) = \frac{1}{x}$$
$$f_2(x) = x\,e^{-x^2}$$
$$f_3(x) = \frac{1}{1+x^2}$$
$$f_4(x) = \frac{\sin x}{x}$$
$$f_5(x) = \cos\left(x^2\right)$$

Each function takes as parameter a floating point number and returns a floating point number corresponding to the value of the function evaluated at the input parameter. Place the functions in an array of pointers to functions defined in the same source file. The same source file shall only contain code related to the definition, description and evaluation of the functions specified above. All accesses to the $f_i(x)$ above shall be through the array of function pointers.

2. [20 pts] Implement the Trapezoidal Rule in a separate C/C++ function and place this function in a second source file. This source file shall only contain code and data related to the implementation of the Trapezoidal Rule. The implementation shall be based on sec 17.5 of [Kreyszig].

3. [10 pts] The Trapezoidal Rule function shall accept as parameters, among others, a pointer to one of the functions defined in part 1, and floating point numbers specifying the limits of integration.

4. [10 pts] Implement the main( ) function and the user interface in a third source file.

The user interface provides a means for the user to choose one of the functions defined in part 1 to be integrated, to specify the limits of integration and to invoke the Trapezoidal Rule function.

The user interface allows the user to modify various parameters of operation of the program.

The user interface provides appropriate feedback to the user. The user interface reports the results of the integration to the user.

5. [10 pts] The Trapezoidal Rule function shall report results to the user interface by returning appropriate data to the user interface, by modifying appropriate parameters or by a combination of both approaches.

6. [10 pts] Provide an option for the Trapezoidal Rule function to report intermediate results and provide a trace of its operation. This option is controlled thru the user interface and passed as a parameter to the Trapezoidal Rule function. By default, the Trapezoidal Rule function simply returns the results of its computations to the user interface as described in part 5, without printing anything to the console or to standard output. The user shall not be required to accept or confirm use of the current setting of this option.

7. [5 pts] The Trapezoidal Rule function computes the definite integral to a number of significant figures specified as a parameter to the function. The number of significant figures defaults to 4. The user interface provides a means for the user to modify this number. The user shall not be required to accept or confirm use of the current setting of this number.

8. [10 pts] The Trapezoidal Rule function achieves the desired accuracy by obtaining better estimates of the integral based on halving of intervals. It uses the basic error principle and relative error estimates based on the two latest estimates of the definite integral to determine if agreement up to the desired number of significant figures has been reached. The user interface reports the result of the integration using the required number of significant figures.

9. [10 pts] The integrand, the function from part 1 that is to be integrated, will have to be evaluated at various points. The Trapezoidal Rule function shall keep to a minimum the number of times the integrand is evaluated. The integrand should not have to be evaluated repeatedly at the same point as the Trapezoidal Rule function increases the number of intervals.

Results of previous evaluations of the integrand may be stored in dynamically allocated or dynamically resized arrays. These arrays are allocated by the Trapezoidal Rule function and freed before it returns.

10. [10 pts] The user interface shall make the program easy and convenient to use.

Grading: The point distribution is as indicated, giving a total of 100 points. Penalties may be imposed for inefficient and/or poorly designed code. Points may be forfeited if submitted code cannot be satisfactorily explained during the project evaluation. For students working in groups, all group members are responsible for knowing all parts of the code. Students working in groups may be given different grades.

CE 30 Programming Exercise 006
Numerical Integration using Simpson's Rule

The main goal of this project is to implement Simpson's Rule for computing definite integrals of functions of a single variable.

The specifications will require various parts of the implementation to be placed in separate source files for the sake of modularity. Hence header files and/or various declarations may be needed for data and functions in one source file to be accessible to code in another source file.

1. [5 pts] In a separate source file, define C/C++ functions implementing the following functions:

$$f_1(x)=\frac{1}{x}$$
$$f_2(x)=x e^{-x^2}$$
$$f_3(x)=\frac{1}{1+x^2}$$
$$f_4(x)=\frac{\sin x}{x}$$
$$f_5(x)=\cos\left(x^2\right)$$

Each function takes as parameter a floating point number and returns a floating point number corresponding to the value of the function evaluated at the input parameter. Place the functions in an array of pointers to functions defined in the same source file. The same source file shall only contain code related to the definition, description and evaluation of the functions specified above. All accesses to the $f_i(x)$ above shall be through the array of function pointers.

2. [20 pts] Implement Simpson's Rule in a separate C/C++ function and place this function in a second source file. This source file shall only contain code and data related to the implementation of Simpson's Rule. The implementation shall be based on sec 17.5 of [Kreyszig].

3. [10 pts] The Simpson's Rule function shall accept as parameters, among others, a pointer to one of the functions defined in part 1, and floating point numbers specifying the limits of integration.

4. [10 pts] Implement the main( ) function and the user interface in a third source file.

The user interface provides a means for the user to choose one of the functions defined in part 1 to be integrated, to specify the limits of integration and to invoke the Simpson's Rule function.

The user interface allows the user to modify various parameters of operation of the program.

The user interface provides appropriate feedback to the user. The user interface reports the results of the integration to the user.

5. [10 pts] The Simpson's Rule function shall report results to the user interface by returning appropriate data to the user interface, by modifying appropriate parameters or by a combination of both approaches.

6. [10 pts] Provide an option for the Simpson's Rule function to report intermediate results and provide a trace of its operation. This option is controlled thru the user interface and passed as a parameter to the Simpson's Rule function. By default, the Simpson's Rule function simply returns the results of its computations to the user interface as described in part 5, without printing anything to the console or to standard output. The user shall not be required to accept or confirm use of the current setting of this option.

7. [5 pts] The Simpson's Rule function computes the definite integral to a number of significant figures specified as a parameter to the function. The number of significant figures defaults to 4. The user interface provides a means for the user to modify this number. The user shall not be required to accept or confirm use of the current setting of this number.

8. [10 pts] The Simpson's Rule function achieves the desired accuracy by obtaining better estimates of the integral based on halving of intervals. It uses the basic error principle and relative error estimates based on the two latest estimates of the definite integral to determine if agreement up to the desired number of significant figures has been reached. The user interface reports the result of the integration using the required number of significant figures.

9. [10 pts] The integrand, the function from part 1 that is to be integrated, will have to be evaluated at various points. The Simpson's Rule function shall keep to a minimum the number of times the integrand is evaluated. The integrand should not have to be evaluated repeatedly at the same point as the Simpson's Rule function increases the number of intervals.

Results of previous evaluations of the integrand may be stored in dynamically allocated or dynamically resized arrays. These arrays are allocated by the Simpson's Rule function and freed before it returns.

10. [10 pts] The user interface shall make the program easy and convenient to use.

Grading: The point distribution is as indicated, giving a total of 100 points. Penalties may be imposed for inefficient and/or poorly designed code. Points may be forfeited if submitted code cannot be satisfactorily explained during the project evaluation. For students working in groups, all group members are responsible for knowing all parts of the code. Students working in groups may be given different grades.

CE 30 Programming Exercise 007
Solution of Linear Systems using Gauss Elimination

The main goal of this project is to implement Gauss Elimination for solving linear systems of equations. The system of equations shall be specified by data from REALMATRIX files representing the coefficient matrix, and REALVEC files representing the vector of constants.

The specifications will require various parts of the implementation to be placed in separate source files for the sake of modularity. Hence header files and/or various declarations may be needed for data and functions in one source file to be accessible to code in another source file.

1. [20 pts] In a separate source file, implement functions that will parse REALMATRIX and REALVEC files, placing the data into dynamically allocated arrays. These functions take FILE pointers as parameters and return appropriate data structures, or pointers to data structures, containing results of the parsing, including the matrix or vector data themselves, the dimensions of the matrix or vector, and indications of any errors that might have occurred. As an alternative, functions may return information thru the modification of parameters passed to them.

Note that these functions should work regardless of the dimensions of the matrices or vectors, subject only to the limits of available memory. Matrices represented in REALMATRIX files are not necessarily square.

The source file should only contain functions and data related to the parsing of REALMATRIX and REALVEC files.

2. [20 pts] Implement Gauss Elimination and back substitution in a separate source file. This source file shall only contain code and data related to the solution of systems of equations using Gauss Elimination. The implementation shall be based on sec 18.1 of [Kreyszig]. Use partial pivoting as described in sec 18.1 of [Kreyszig]. Augmented matrices shall be specified by the data in matrices and vectors obtained from REALMATRIX and REALVEC files. These data and the dimension of the system of equations shall be among the parameters passed to the Gauss Elimination function.

3. [10 pts] Implement the main( ) function and user interface in a separate source file.

The user interface allows the user to specify REALMATRIX and REALVEC files to be parsed by the functions in part 1 and to invoke the Gauss Elimination function on the data obtained from parsed REALMATRIX and REALVEC files.

Before being passed to the Gauss Eliminatin function, the dimensions of the matrix and vector are checked to ensure that the matrix and vector represent a valid system of equations.

The user interface allows the user to modify various parameters of operation of the program.

The user interface provides appropriate feedback to the user. The user interface reports the results of the Gauss Elimination to the user.

4. [10 pts] The Gauss Elimination function shall report results to the user interface by returning appropriate data to the user interface, by modifying appropriate parameters or by a combination of both approaches.

5. [10 pts] Provide an option for the Gauss Elimination function to report intermediate results and provide a trace of its operation. This option is controlled thru the user interface and passed as a parameter to the Gauss Elimination function. By default, the Gauss Elimination function simply returns the results of its computations to the user interface as described in part 4, without printing anything to the console or to standard output. The user shall not be required to accept or confirm use of the current setting of this option.

6. [10 pts] In the case where a unique solution is obtained, the user interface prints the solution. The user is also given the option of having the solution written to a REALVEC file.

7. [10 pts] By default, entries in the solution vector are printed with 4 significant figures. The user interface provides a means for the user to modify this number. The user shall not be required to accept or confirm use of the current setting of this number.

8. [10 pts] The user interface shall make the program easy and convenient to use.

Grading: The point distribution is as indicated, giving a total of 100 points. Penalties may be imposed for inefficient and/or poorly designed code. Points may be forfeited if submitted code cannot be satisfactorily explained during the project evaluation. For students working in groups, all group members are responsible for knowing all parts of the code. Students working in groups may be given different grades.

CE 30 Programming Exercise 008
Matrix Inversion using Gauss-Jordan Elimination

The main goal of this project is to implement Gauss-Jordan Elimination for computing the inverse of a matrix. The matrix shall be specified by data from REALMATRIX files.

The specifications will require various parts of the implementation to be placed in separate source files for the sake of modularity. Hence header files and/or various declarations may be needed for data and functions in one source file to be accessible to code in another source file.

1. [20 pts] In a separate source file, implement functions that will parse REALMATRIX files, placing the data into dynamically allocated arrays. These functions take FILE pointers as parameters and return appropriate data structures, or pointers to data structures, containing results of the parsing, including the matrix itself, the dimensions of the matrix, and indications of any errors that might have occurred. As an alternative, functions may return information thru the modification of parameters passed to them.

Note that these functions should work regardless of the dimensions of the matrices, subject only to the limits of available memory. Matrices represented in REALMATRIX files are not necessarily square.

The source file should only contain functions and data related to the parsing of REALMATRIX files.

2. [20 pts] Implement the matrix inversion algorithm based on Gauss-Jordan Elimination in a separate source file. This source file shall only contain code and data related to matrix inversion based on Gauss-Jordan Elimination. The implementation shall be based on sec 18.2 of [Kreyszig]. Use partial pivoting as described in sec 18.1 of [Kreyszig]. The matrix whose inverse is to be found shall be specified by the data obtained from a REALMATRIX file. These data and the dimension of the matrix shall be among the parameters passed to the matrix inversion function.

3. [10 pts] Implement the main( ) function and user interface in a separate source file.

The user interface allows the user to specify REALMATRIX files to be parsed by the functions in part 1 and to invoke the matrix inversion function on the data obtained from parsed REALMATRIX files. The matrix inversion function is called only if the matrix is square.

The user interface allows the user to modify various parameters of operation of the program.

The user interface provides appropriate feedback to the user. The user interface reports the results of the matrix inversion to the user.

4. [10 pts] The matrix inversion function shall report results to the user interface by returning appropriate data to the user interface, by modifying appropriate parameters or by a combination of both approaches.

5. [10 pts] Provide an option for the matrix inversion function to report intermediate results and provide a trace of its operation. This option is controlled thru the user interface and passed as a parameter to the matrix inversion function. By default, the matrix inversion function simply returns the results of its computations to the user interface as described in part 4, without printing anything to the console or to standard output. The user shall not be required to accept or confirm use of the current setting of this option.

6. [10 pts] In the case where an inverse is obtained, the user interface prints the inverse The user is also given the option of having the inverse matrix written to a REALMATRIX file.

7. [10 pts] By default, entries in the inverse matrix are printed with 4 significant figures. The user interface provides a means for the user to modify this number. The user shall not be required to accept or confirm use of the current setting of this number.

8. [10 pts] The user interface shall make the program easy and convenient to use.

Grading: The point distribution is as indicated, giving a total of 100 points. Penalties may be imposed for inefficient and/or poorly designed code. Points may be forfeited if submitted code cannot be satisfactorily explained during the project evaluation. For students working in groups, all group members are responsible for knowing all parts of the code. Students working in groups may be given different grades.

CE 30 Programming Exercise 009
Solution of Linear Systems using LU Factorization by Doolittle's Method

The main goal of this project is to implement Doolittle's Method of LU Factorization for solving linear systems of equations. The system of equations shall be specified by data from REALMATRIX files representing the coefficient matrix, and REALVEC files representing the vector of constants.

The specifications will require various parts of the implementation to be placed in separate source files for the sake of modularity. Hence header files and/or various declarations may be needed for data and functions in one source file to be accessible to code in another source file.

1. [20 pts] In a separate source file, implement functions that will parse REALMATRIX and REALVEC files, placing the data into dynamically allocated arrays. These functions take FILE pointers as parameters and return appropriate data structures, or pointers to data structures, containing results of the parsing, including the matrix or vector data themselves, the dimensions of the matrix or vector, and indications of any errors that might have occurred. As an alternative, functions may return information thru the modification of parameters passed to them.

Note that these functions should work regardless of the dimensions of the matrices or vectors, subject only to the limits of available memory. Matrices represented in REALMATRIX files are not necessarily square.

The source file should only contain functions and data related to the parsing of REALMATRIX and REALVEC files.

2. [20 pts] Implement Doolittle's Method for obtaining an LU decomposition in a separate source file. This source file shall only contain code and data related to the solution of systems of equations using Doolittle's Method. The implementation shall be based on sec 18.2 of [Kreyszig]. Interchange rows as needed and keep track of these row interchanges. Note that the LU decomposition requires only the coefficient matrix. Information regarding row interchanges will be needed when the L and U matrices are used to solve the system of equations.

In the same source file implement a function, or functions, taking as parameters the L and U matrices, the vector of constants, the size of the system and other parameters that might be needed. The function obtains a solution to the system, and returns this solution to the calling function.

3. [10 pts] Implement the main( ) function and user interface in a separate source file.

The user interface allows the user to specify REALMATRIX and REALVEC files to be parsed by the functions in part 1 and to pass the data to the funtions implemented in part 2 for solving the system of equations.

The dimensions of the matrix and vector are checked to ensure that the matrix and vector represent a valid system of equations.

The user interface allows the user to modify various parameters of operation of the program.

The user interface provides appropriate feedback to the user. The user interface reports the solution to the user.

4. [10 pts] The LU decomposition functions in part 2 shall report results to the user interface by returning appropriate data to the user interface, by modifying appropriate parameters or by a combination of both approaches.

5. [10 pts] Provide an option for the user to export the L and U matrices to REALMATRIX files.

6. [5 pts] In the case where a unique solution is obtained, the user interface prints the solution. The user is also given the option of having the solution written to a REALVEC file.

7. [10 pts] Note that since the LU decomposition depends only on the coefficient matrix, different systems having the same coefficient matrix and differing only in the vector of constants, can all be solved based on a single LU decomposition. That is, the LU decomposition has to be done only once.

The user interface should allow the system to be modified by replacement of the current vector by data from another REALVEC file of the same dimension. It should be possible to solve the new system without performing a new LU decomposition.

8. [5 pts] By default, entries in the solution vector and in the L and U matrices are printed with 4 significant figures. The user interface provides a means for the user to modify this number. The user shall not be required to accept or confirm use of the current setting of this number.

9. [10 pts] The user interface shall make the program easy and convenient to use.

Grading: The point distribution is as indicated, giving a total of 100 points. Penalties may be imposed for inefficient and/or poorly designed code. Points may be forfeited if submitted code cannot be satisfactorily explained during the project evaluation. For students working in groups, all group members are responsible for knowing all parts of the code. Students working in groups may be given different grades.

CE 30 Programming Exercise 010
Solution of Linear Systems using LU Factorization by Crout's Method

The main goal of this project is to implement Crout's Method of LU Factorization for solving linear systems of equations. The system of equations shall be specified by data from REALMATRIX files representing the coefficient matrix, and REALVEC files representing the vector of constants.

The specifications will require various parts of the implementation to be placed in separate source files for the sake of modularity. Hence header files and/or various declarations may be needed for data and functions in one source file to be accessible to code in another source file.

1. [20 pts] In a separate source file, implement functions that will parse REALMATRIX and REALVEC files, placing the data into dynamically allocated arrays. These functions take FILE pointers as parameters and return appropriate data structures, or pointers to data structures, containing results of the parsing, including the matrix or vector data themselves, the dimensions of the matrix or vector, and indications of any errors that might have occurred. As an alternative, functions may return information thru the modification of parameters passed to them.

Note that these functions should work regardless of the dimensions of the matrices or vectors, subject only to the limits of available memory. Matrices represented in REALMATRIX files are not necessarily square.

The source file should only contain functions and data related to the parsing of REALMATRIX and REALVEC files.

2. [20 pts] Implement Crout's Method for obtaining an LU decomposition in a separate source file. This source file shall only contain code and data related to the solution of systems of equations using Crout's Method. The implementation shall be based on sec 18.2 of [Kreyszig]. Interchange rows as needed and keep track of these row interchanges. Note that the LU decomposition requires only the coefficient matrix. Information regarding row interchanges will be needed when the L and U matrices are used to solve the system of equations.

In the same source file implement a function, or functions, taking as parameters the L and U matrices, the vector of constants, the size of the system and other parameters that might be needed. The function obtains a solution to the system, and returns this solution to the calling function.

3. [10 pts] Implement the main( ) function and user interface in a separate source file.

The user interface allows the user to specify REALMATRIX and REALVEC files to be parsed by the functions in part 1 and to pass the data to the funtions implemented in part 2 for solving the system of equations.

The dimensions of the matrix and vector are checked to ensure that the matrix and vector represent a valid system of equations.

The user interface allows the user to modify various parameters of operation of the program.

The user interface provides appropriate feedback to the user. The user interface reports the solution to the user.

4. [10 pts] The LU decomposition functions in part 2 shall report results to the user interface by returning appropriate data to the user interface, by modifying appropriate parameters or by a combination of both approaches.

5. [10 pts] Provide an option for the user to export the L and U matrices to REALMATRIX files.

6. [5 pts] In the case where a unique solution is obtained, the user interface prints the solution. The user is also given the option of having the solution written to a REALVEC file.

7. [10 pts] Note that since the LU decomposition depends only on the coefficient matrix, different systems having the same coefficient matrix and differing only in the vector of constants, can all be solved based on a single LU decomposition. That is, the LU decomposition has to be done only once.

The user interface should allow the system to be modified by replacement of the current vector by data from another REALVEC file of the same dimension. It should be possible to solve the new system without performing a new LU decomposition.

8. [5 pts] By default, entries in the solution vector and in the L and U matrices are printed with 4 significant figures. The user interface provides a means for the user to modify this number. The user shall not be required to accept or confirm use of the current setting of this number.

9. [10 pts] The user interface shall make the program easy and convenient to use.

Grading: The point distribution is as indicated, giving a total of 100 points. Penalties may be imposed for inefficient and/or poorly designed code. Points may be forfeited if submitted code cannot be satisfactorily explained during the project evaluation. For students working in groups, all group members are responsible for knowing all parts of the code. Students working in groups may be given different grades.

REALMATRIX File Format

Luisito L. Agustin

A REALMATRIX file is a text file meant for representing matrices with real-valued entries.

Each line of text shall not be more than 2048 characters long. A double slash, "//", starts a comment lasting up to the end of a line.Unexpected text initiates a comment lasting up to the end of a line. White spaces and end-of-line characters are not significant except possibly for separating tokens.

In the absence of comments, the file format is as follows:

```
REALMATRIX
dim nrows x ncols
{
a_11
a_12
...
a_1,ncols
a_21
...
a_2,ncols
a_31
...
a_nrows,1
...
a_nrows,ncols
}
```

"REALMATRIX" is a case-sensitive keyword indicating that the file is a REALMATRIX file. "dim" is a case-sensitive keyword. nrows is an integer indicating the number of rows in the matrix. ncols is an integer indicating the number of columns in the matrix. The "x" between nrows and ncols is a keyword. The a_i,j are floating point numbers in general, with the first subscript indicating the row number and the second indicating the column number.

Applications that read REALMATRIX files shall do so in a single pass.

Applications may parse XYDAT files by reading each line of text one at a time and discarding comments before parsing for expected tokens. If unexpected text is found, the unexpected text is treated as starting a comment lasting up to the end of the line, and the comment discarded. Applications shall ignore text beyond the 2048th character in each line.

Applications that produce REALMATRIX files shall have the REALMATRIX keyword on a line by itself, and shall ensure that all lines of text do not exceed 2048 characters.

REALVEC File Format

Luisito L. Agustin

A REALVEC file is a text file meant for representing vectors with real-valued components.

Each line of text shall not be more than 2048 characters long. A double slash, “//”, starts a comment lasting up to the end of a line.Unexpected text initiates a comment lasting up to the end of a line. White spaces and end-of-line characters are not significant except possibly for separating tokens.

In the absence of comments, the file format is as follows:

```
REALVEC
dim n
{
a_1
a_2
...
a_n
}
```

“REALVEC” is a case-sensitive keyword indicating that the file is a REALVEC file. “dim” is a case-sensitive keyword. n is a positive integer indicating the number of components of the vector. The a_i are floating point numbers in general.

Applications that read REALVEC files shall do so in a single pass.

Applications may parse REALVEC files by reading each line of text one at a time and discarding comments before parsing for expected tokens. If unexpected text is found, the unexpected text is treated as starting a comment lasting up to the end of the line, and the comment discarded. Applications shall ignore text beyond the 2048th character in each line.

Applications that produce REALVEC files shall have the REALVEC keyword on a line by itself, and shall ensure that all lines of text do not exceed 2048 characters.