CE 30
Second Semester, 2004-2005
Options for Project 1
by: Luisito L. Agustin

Option 1: Newton-Raphson Method
Quota: at most 6 students working individually or in groups

Option 2: Secant Method
Quota: at most 6 students working individually or in groups

Option 3: Bisection Method
Quota: at most 6 students working individually or in groups

Option 4: Method of False Position
Quota: at most 6 students working individually or in groups

Reference:

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

CE 30 Programming Exercise 001
Newton-Raphson Method

The main goals of this project are to implement the Newton-Raphson Method for obtaining solutions to equations of the form $f(x)=0$ and to demonstrate use of the method in getting the solutions of a predefined set of equations specified by given expressions for $f(x)$.

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

$$f_1(x)=x^3+x-1$$
$$f_2(x)=x-2\sin x$$
$$f_3(x)=x\cosh x-1$$
$$f_4(x)=\sin x-e^{-x}$$
$$f_5(x)=x^4-x-0.12$$

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. Also implement C/C++ functions for the first derivatives of these functions.

The names of functions are of type pointer to function. Place the functions in arrays 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 access to the $f_i(x)$ above and their derivatives shall be through the arrays of function pointers.

2. [20 pts] Implement the Newton-Raphson Method in a separate C/C++ function and place this function in a second source file. This source file shall only contain code related to the implementation of the Newton-Raphson Method. The implementation shall be based on sec 17.2 of [Kreyszig].

3. [20 pts] The Newton-Raphson function shall accept the following as parameters:
* a pointer to the function whose zeroes will be evaluated
* a pointer to the derivative of the function
* an initial guess for the solution
* an integer specifying the number of significant figures to which the solution would be obtained
* an integer specifying the maximum number of iterations that would be carried out
* other parameters that might be needed

The Newton-Raphson function provides a trace of its execution by printing out each successive estimate of the solution along with other information that would allow the correctness of the algorithm to be verified.

4. [20 pts] Implement the main( ) function and the user interface in a third source file. Provide a means for the user to choose one of the functions in part 1, so that the corresponding function pointers (the pointer to the function and the pointer to the derivative function) could be obtained by indexing into the arrays of pointers to functions and then passed as parameters to the Newton-Raphson function implemented in parts 2 and 3.

The user interface also provides a means for the user to provide an initial guess for the solution, to solve for other roots, and to change the chosen function.

5. [10 pts] The user interface informs the user whether a solution was found, possibly based on data returned by the Newton-Raphson function and/or based on parameters modified by it. If a solution was found the solution is printed with the required number of significant figures. If none was found, some feedback is given regarding what went wrong.

6. [10 pts] The user interface allows the user to change the maximum number of iterations to be performed and the number of significant figures to which a solution would be obtained. These are passed as parameters to the Newton-Raphson function. These parameters shall be set to reasonable default values when the program starts so that the user does not have to explicitly set them in order to get results.

7. [10 pts] In order to assist the user in making initial guesses as well as confirm results, provide a means for the user to evaluate a chosen function both at user-specified values of $x$ and at a user-specified number of equispaced points in an interval with user-specified bounds.

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 from sections 4, 5, 6 and 7 may be partially or totally forfeited if the implementation of the Newton-Raphson Method is incorrect. 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 002
Secant Method

The main goals of this project are to implement the Secant Method for obtaining solutions to equations of the form $f(x)=0$ and to demonstrate use of the method in getting the solutions of a predefined set of equations specified by given expressions for $f(x)$.

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

$$f_1(x)=e^{-x}-\tan x$$
$$f_2(x)=\cos x \cosh x-1$$
$$f_3(x)=\cos x-\sqrt{x}$$
$$f_4(x)=\sin x-e^{-x}$$
$$f_5(x)=x^4-x-0.12$$

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.

The names of functions are of type pointer to function. 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 access to the $f_i(x)$ above shall be through the array of function pointers.

2. [20 pts] Implement the Secant Method in a separate C/C++ function and place this function in a second source file. This source file shall only contain code related to the implementation of the Secant Method. The implementation shall be based on sec 17.2 of [Kreyszig].

3. [20 pts] The Secant Method function shall accept the following as parameters:
* a pointer to the function whose zeroes will be evaluated
* two initial guesses for the solution
* an integer specifying the number of significant figures to which the solution would be obtained
* an integer specifying the maximum number of iterations that would be carried out
* other parameters that might be needed

The Secant Method function provides a trace of its execution by printing out each successive estimate of the solution along with other information that would allow the correctness of the algorithm to be verified.

4. [20 pts] Implement the main( ) function and the user interface in a third source file. Provide a means for the user to choose one of the functions in part 1, so that the corresponding function pointer could be obtained by indexing into the array of function pointers and then passed as a parameter to the Secant Method function implemented in parts 2 and 3.

The user interface also provides a means for the user to provide initial guesses for the solution, to solve for other roots, and to change the chosen function.

5. [10 pts] The user interface informs the user whether a solution was found, possibly based on data returned by the Secant Method function and/or based on parameters modified by it. If a solution was found the solution is printed with the required number of significant figures. If none was found, some feedback is given regarding what went wrong.

6. [10 pts] The user interface allows the user to change the maximum number of iterations to be performed and the number of significant figures to which a solution would be obtained. These are passed as parameters to the Secant Method function. These parameters shall be set to reasonable default values when the program starts so that the user does not have to explicitly set them in order to get results.

7. [10 pts] In order to assist the user in making initial guesses as well as confirm results, provide a means for the user to evaluate a chosen function both at user-specified values of $x$ and at a user-specified number of equispaced points in an interval with user-specified bounds.

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 from sections 4, 5, 6 and 7 may be partially or totally forfeited if the implementation of the Secant Method is incorrect. 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 003
Bisection Method

The main goals of this project are to implement the Bisection Method for obtaining solutions to equations of the form $f(x)=0$ and to demonstrate use of the method in getting the solutions of a predefined set of equations specified by given expressions for $f(x)$.

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

$$f_1(x)=x+\ln x-2$$
$$f_2(x)=\cos x \cosh x-1$$
$$f_3(x)=\cos x-\sqrt{x}$$
$$f_4(x)=e^{-x}-\ln x$$
$$f_5(x)=e^{x}+x^{4}+x-2$$

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.

The names of functions are of type pointer to function. Place the functions in an array of function pointers 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 access to the $f_i(x)$ above shall be through the array of function pointers.

2. [20 pts] Implement the Bisection Method in a separate C/C++ function and place this function in a second source file. This source file shall only contain code related to the implementation of the Bisection Method. The implementation shall be based on exercise 25, page 848, sec 17.2 of [Kreyszig].

3. [20 pts] The Bisection function shall accept the following as parameters:
* a pointer to the function whose zeroes will be evaluated
* two initial guesses for the solution
* an integer specifying the number of significant figures to which the solution would be obtained
* an integer specifying the maximum number of iterations that would be carried out
* other parameters that might be needed

The Bisection Method function provides a trace of its execution by printing out each successive estimate of the solution along with other information that would allow the correctness of the algorithm to be verified.

4. [20 pts] Implement the main( ) function and the user interface in a third source file. Provide a means for the user to choose one of the functions in part 1, so that the corresponding function pointer could be obtained by indexing into the array of pointers to functions and then passed as a parameter to the Bisection Method function implemented in parts 2 and 3.

The user interface also provides a means for the user to provide initial guesses for the solution, to solve for other roots, and to change the chosen function.

5. [10 pts] The user interface informs the user whether a solution was found, possibly based on data returned by the Bisection Method function and/or based on parameters modified by it. If a solution was found the solution is printed with the required number of significant figures. If none was found, some feedback is given regarding what went wrong.

6. [10 pts] The user interface allows the user to change the maximum number of iterations to be performed and the number of significant figures to which a solution would be obtained. These are passed as parameters to the Bisection Method function. These parameters shall be set to reasonable default values when the program starts so that the user does not have to explicitly set them in order to get results.

7. [10 pts] In order to assist the user in making initial guesses as well as confirm results, provide a means for the user to evaluate a chosen function both at user-specified values of $x$ and at a user-specified number of equispaced points in an interval with user-specified bounds.

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 from sections 4, 5, 6 and 7 may be partially or totally forfeited if the implementation of the Bisection Method is incorrect. 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 004
Method of False Position

The main goals of this project are to implement the Method of False Position for obtaining solutions to equations of the form $f(x)=0$ and to demonstrate use of the method in getting the solutions of a predefined set of equations specified by given expressions for $f(x)$.

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

$$f_1(x)=x+\ln x-2$$
$$f_2(x)=x^3+x-1$$
$$f_3(x)=x-2\sin x$$
$$f_4(x)=e^{-x}-\ln x$$
$$f_5(x)=e^x+x^4+x-2$$

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.

The names of functions are of type pointer to function. 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 access to the $f_i(x)$ above shall be through the array of function pointers.

2. [20 pts] Implement the Method of False Position in a separate C/C++ function and place this function in a second source file. This source file shall only contain code related to the implementation of the Method of False Position. The implementation shall be based on sec 17.2 of [Kreyszig].

3. [20 pts] The False Position function shall accept the following as parameters:
* a pointer to the function whose zeroes will be evaluated
* two initial guesses for the solution
* an integer specifying the number of significant figures to which the solution would be obtained
* an integer specifying the maximum number of iterations that would be carried out
* other parameters that might be needed

The False Position function provides a trace of its execution by printing out each successive estimate of the solution along with other information that would allow the correctness of the algorithm to be verified.

4. [20 pts] Implement the main( ) function and the user interface in a third source file. Provide a means for the user to choose one of the functions in part 1, so that the corresponding function pointer could be obtained by indexing into the array of function pointers and then passed as parameters to the False Position function implemented in parts 2 and 3.

The user interface also provides a means for the user to provide initial guesses for the solution, to solve for other roots, and to change the chosen function.

5. [10 pts] The user interface informs the user whether a solution was found, possibly based on data returned by the False Position function and/or based on parameters modified by it. If a solution was found the solution is printed with the required number of significant figures. If none was found, some feedback is given regarding what went wrong.

6. [10 pts] The user interface allows the user to change the maximum number of iterations to be performed and the number of significant figures to which a solution would be obtained. These are passed as parameters to the False Position function. These parameters shall be set to reasonable default values when the program starts so that the user does not have to explicitly set them in order to get results.

7. [10 pts] In order to assist the user in making initial guesses as well as confirm results, provide a means for the user to evaluate a chosen function both at user-specified values of $x$ and at a user-specified number of equispaced points in an interval with user-specified bounds.

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 from sections 4, 5, 6 and 7 may be partially or totally forfeited if the implementation of the Method of False Position is incorrect. 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.