Example: Newton’s root finding algorithm

Use Newton’s method to find roots of \( f(x) = 0 \) where

\[
f(x) = 9x^4 - 42x^3 - 1040x^2 + 5082x - 5929.
\]

Solution:

Create a function to find the root of

```c
// function to find root of
double poly(double x){
    return ((9.*x - 42.)*x - 1040.)*x + 5082.*x -5929;
}
```

its derivative

```c
// derivative function to find root of
double poly_deriv(double x){
    return (36.*x - 126.)*x - 2080.)*x + 5082.;
}
```

and a simple root finding function defined by:

```c
// find the root of a function
// On input: x is initial guess, On return x is the root
// maxiter is the max no of its, tol is the accuracy of the root
// 0 will be returned if no root is found
// 1 if it is successful
int find_root(double& x, int maxiter, double tol){
    // find root
}
```

The algorithm to find the root is given by:

\[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}
\]

so we may write the code inside the function as
```cpp
// find root
int iter;
for(iter=0; iter<maxiter; iter++){
    x = x - poly(x)/poly_deriv(x);
    if(std::abs(poly(x))<tol) break;
}
if(iter==maxiter) return 0;
else return 1;
```

Example use of the program is

```cpp
int iter=100;
double x, tol=1.e-6;
std::cout << "Enter initial guess\n";
std::cin >> x;
if(find_root(x, iter, tol))
    std::cout << "There is a root at " << x << "\n";
else
    std::cout << "No root found.\n";
```

Functions and Loops

2.1 Repeat the root finding algorithm using the Secant method:

\[ x_n = x_{n-1} - \frac{f(x_{n-1})(x_{n-1} - x_{n-2})}{f(x_{n-1}) - f(x_{n-2})} \]

2.2 Write a function to compute the value of \( \sin(x) \) using the first \( N \) terms of the power series expansion (\( N \) should be an argument to the function):

\[ \sin(x) \approx \sum_{k=0}^{N} (-1)^{k} \frac{x^{2k+1}}{(2k+1)!} \]

How many terms of the series are required to agree with the built-in \( \sin \) function. Produce output of the two functions side by side for the values 0, \( \pi/6 \), \( \pi/4 \), \( \pi/2 \), 2\( \pi/3 \) and \( \pi \).

2.3 Rewrite the above function for \( \sin \) so that terms are added \textbf{until} the result is accurate to 10 decimal places. Think about what condition is satisfied for this to happen – i.e. what is the truncation error?

2.4 Write a function that takes the arguments \( a \), \( b \) and \( c \), computes and then displays the roots of the quadratic equation

\[ ax^2 + bx + c = 0. \]

You will need to identify whether the roots are real or complex. If the roots are complex, display the results in the form \( A + Bi \).
2.5 Write functions to evaluate the following expressions.

(i) \( f(\gamma) = 2 \tan^{-1} \left( \left( \frac{1+\gamma}{1-\gamma} \right)^{\frac{1}{2}} \sin \gamma \right) \)

(ii) \( g(x) = \frac{1}{\sqrt{2\pi}} \log(x^3 + 1)e^{-x^2} \)

Using \( x = 0.14, a = 0.235 \) and \( b = 1.763 \), evaluate \( y = f(g(x)) \) and \( y = g(f(x + a) + g(b)) \).

2.6 Write a function to calculate the \( n \)th Fibonacci number. Recall that the Fibonacci numbers, \( f_n \), are defined by

\( f_0 = 0, f_1 = 1, f_n = f_{n-1} + f_{n-2} \)

Use your function to calculate and display the Fibonacci quotient, \( q_n = f_n/f_{n-1} \), for a number of different values of \( n \). You should find that as \( n \) increases \( q_n \) converges to the golden mean, \( (1 + \sqrt{5})/2 \approx 1.618 \).

2.7 The recurrence relation

\( p_0 = 1, p_1 = \frac{1}{3}, p_n = \frac{19}{3}p_{n-1} - 2p_{n-2}, \quad (n \geq 2) \)

has the unique solution \( p_n = \left( \frac{1}{3} \right)^n, n \geq 0 \).

Write a C++ program to compute the terms \( n = 0 \) up to \( n = 20 \) of the series using the data type \texttt{float} to store variables. Output the results to a file with columns showing the following quantities:

\[ n, \ p_n \ (\text{analytical}), \ p_n \ (\text{numerical}), \ \text{relative error}, \ \text{absolute error} \]

2.8 Run your program again this time using \texttt{double}. Is there any difference between the results? Can you think of a reason why the results are different?

Pointers

2.9 Create an integer array of dimension 10. Print out the memory addresses of each element in the array. What do you notice?

2.10 Create two versions of a function that returns the cube of a floating point number. Pass the variable itself in one version and the pointer to the variable in the other. If you overwrite the variable in the function does it matter? If so, why?

2.11 Write a function to swap two numbers.

2.12 Write a function that calculates the maximum value of an array of numbers.
2.13 Write a function passing x as an argument. Try declaring (in the argument list) as

(i) double x
(ii) double& x
(iii) const double x
(iv) const double& x

What happens when you try to assign x a new value inside the function?

2.14 Use a pointer to create a new double array with n elements (where n is specified somewhere else in the code):

```cpp
double *array;
array = new double[n]
```

Assign the values 0 through to 99 to the elements. What happen if you write the code:

```cpp
std::cout << "an element outside the array" << array[200];
```

Don’t forget to delete the storage when you’re finished

```cpp
delete array[];
```