Due date: Maple: Monday, February 14, at the beginning of the lab  
C++/Java: Monday, February 21, at the beginning of the lab

1 What this Lab is About

Our plans for this assignment are

- To implement the bisection and fixed point algorithms for the numerical resolution of nonlinear equations;
- to learn about deflation methods;
- To apply these techniques to the resolution of a simple practical problem.

2 Taking a first look at some NLEs

2.1 Nonlinear equations studied

We will study nonlinear equations of the form \( f(x) = 0 \), where \( f \) is one of the following two functions:

\[
\begin{align*}
    f_1(x) &= xe^x - 2e^{x/2}, \\
    f_2(x) &= x^2 \ln(\cos x + 2) - x \cos x - 1.
\end{align*}
\]

You should probably use Maple to plot these functions and identify (by looking at the plots) intervals on which to search for solutions of the NLEs.

2.2 The bisection algorithm

The bisection (dichotomy) algorithm that we saw in class theoretically allows us to get as close as me may desire to a solution \( x^* \) to the nonlinear equation. Of course, we shouldn’t expect to do better than the machine precision allowed by our data types and hardware.
What to do, Part I:
Implement the bisection algorithm (in an iterative form). Your procedure/method/function should take as parameters the function defining the NLE problem, the endpoints of the search segment and the desired tolerance of the search. We would like to get as output the value of the solution \( x^* \) found, as well as the number of iterations of the search.

### 2.3 The Fixed Point Iteration Algorithm

#### 2.3.1 Fixed point problems

You may have tried at some point\(^1\) to hit repeatedly the “cos” key of your calculator. If you did so, you found out that no matter where you start from, the iteration always converges to the same value, 0.7390851332 if your calculator is set to operate in radians (and 0.998477415 if angles are expressed in degrees). The “sin” key has 0 as its (boring) convergence point. The “\( \sqrt{\cdot} \)” and “\( x^2 \)” keys both define sequences that converge to 1—again, a pretty dull limit.

These are examples of fixed point problems. A fixed point of a function \( f \) is any \( x^* \) that verifies the equation

\[
f(x) = x.
\]

When you hit repeatedly the “cos” key of your calculator, you are defining a sequence

\[
x_0 = \text{whatever number you start from},
\]

\[
x_{k+1} = \cos(x_k), \quad k = 0, 1, 2, \ldots
\]

For this sequence to converge to a limit \( x^* \), that is, in order to have \( \lim_{n \to +\infty} x_k = x^* \), \( x^* \) must verify the equation

\[
x^* = \cos x^*,
\]

which is to say: \( x^* \) must be a fixed point of the \( \cos \) function.

#### 2.3.2 The NLE as a fixed point problem

Any NLE problem \( f(x) = 0 \) can be reformulated as a fixed point problem by defining a function \( g \) as follows:

\[
g : \mathbb{R} \setminus \{x_0\} \longrightarrow \mathbb{R}
\]

\[
x \mapsto f(x) + x.
\]

Obviously, for any \( x^* \) such that \( f(x^*) = 0 \), \( g(x^*) = x^* \). Any solution to the NLE defined by \( f \) is also a fixed point of \( g \), and vice-versa.

Now, there is not a single fixed point problem associated to a given NLE problem. In fact, there is an infinity of them. Let’s just look at a particular example:

\[
f : \mathbb{R} \setminus \{x_0\} \longrightarrow \mathbb{R}
\]

\[
x \mapsto \frac{\sqrt{x^2 + 1}}{1 + |x|} + \cos(x^2 - 2x + 1) - \frac{\sin x}{x}.
\]

---

\(^1\) Or maybe not: So far, to my utter surprise, I have not yet run into a CSC350 student who remembered having ever done that. So maybe I am weird after all.
We can simply “extract” the $x$ from the denominator under the $\sin x$ by writing

$$f(x) = 0 \implies \frac{\sin x}{x} = \frac{\sqrt{x^2 + 1}}{1 + |x|} + \cos(x^2 - 2x + 1)$$

$$\implies \frac{\sin x}{\frac{\sqrt{x^2 + 1}}{1 + |x|} + \cos(x^2 - 2x + 1)} = x.$$ 

All we have to do now is define

$$g_1 : \mathbb{R} \setminus \{x_0\} \rightarrow \mathbb{R}$$

$$x \mapsto g_1(x) = \frac{\sin x}{\frac{\sqrt{x^2 + 1}}{1 + |x|} + \cos(x^2 - 2x + 1)},$$

and we now have a new fixed point problem derived from our initial NLE problem.

We can define more fixed point problems by simple algebraic manipulation tricks. For example, we can “extract” the $x$ from the expression $\cos(x^2 - 2x + 1)$ to define a new fixed point problem, this time attached to the function

$$g_2 : \mathbb{R} \setminus \{x_0\} \rightarrow \mathbb{R}$$

$$x \mapsto g_2(x) = \frac{1}{2} \left( x^2 + 1 \right) - \frac{1}{2} \alpha \cos \left( \frac{\sin x}{x} - \frac{\sqrt{x^2 + 1}}{1 + |x|} \right),$$

The main difficulty in this reformulation consists in choosing a “good” function $g$. Performing this kind of algebraic manipulations automatically can also be quite challenging. It is one thing to do it in Maple with the expression of a function and quite another to do it with a Function1D object in C++ or Java. We will see in Subsection 2.3.4 a way to produce easily an entire family of fixed point problems corresponding to a given NLE problem.

### 2.3.3 What to do, Part II

Propose a fixed point formulation for $f_1$ and $f_2$. Verify that your functions $g_1$ and $g_2$ satisfy the convergence criterion.

### 2.3.4 Improving the convergence rate

It is possible to improve the $g$ function in a fixed point problem $g(x) = x$ if we do the following.

For any real number $\lambda \neq 0$, we can add $\lambda x$ on both sides of the equation to obtain

$$f(x) + \lambda x = \lambda x.$$

We can then divide by $\lambda$ on both sides to get

$$x = G(x) = \frac{f(x) + \lambda x}{\lambda}.$$

Why should this new function $G$ perform any better than the original $g$? How could you choose $\lambda$?
3 Improving the Algorithms

3.1 Bracketing

The fixed point algorithm (and other algorithms that we will study next week) suffers from a serious problem: The iterated solution does not stay inside the bracket you specified initially, and instead venture wherever the shape of the function’s curve leads them. What we would like to do now is modify the algorithms so that once we have specified an initial bracket that contains a solution, the iteration stays inside the bracket.

This is really easy to do: If the \( x_{k+1} \) that you just computed is outside of the bracket, replace that value for \( x_{k+1} \) by the one given by the bisection algorithm and update the bracket.

What to do, Part III: Modify your fixed point procedure so that

- They all take as parameters two bracketing values, \( a \) and \( b \), a value for the tolerance, and a max number of iteration steps.
- Your iteration stays inside the bracket and updates the bracket at each iteration.

3.2 Deflation

Certainly it will not come as a major surprise to you that there is a deflation process for NLEs as well. Basically, the idea behind deflation for NLEs is as follows.

Let \( f \) be a function defined over the interval \([a, b]\) and that admits multiple zeros over that interval, \( x_1^*, x_2^*, \ldots x_n^* \). We can in theory find all these zeros by applying the following algorithm:

Initialize \( f_1 = f \)

\[ \text{for } i \text{ from 1 to } n \text{ and while not finished do} \]

find \( x_i^* \), a solution of \( f_i(x) = 0 \) on \([a, b]\)

define \( f_{i+1} : x \mapsto \frac{f_i(x)}{x - x_i^*} \)

\[ \text{end do} \]

To see how this works, let us consider the following function:

\[
\begin{align*}
f : \mathbb{R} & \rightarrow \mathbb{R} \\
x & \mapsto x^2 e^{-x} - \frac{1}{2}.
\end{align*}
\]

On the interval \([-1, 3]\), the graph of this function looks as shown in Figure 1.
The zeros of $f$ over $[-1, 3]$ are $-0.539835, 1.48796,$ and $2.61787$. Let’s say that the first zero we find is $x_1^* = 1.48796$. We would now define the function

$$f_2 : \mathbb{R} \rightarrow \mathbb{R}$$

$$x \mapsto \frac{x^2 e^{-x} - \frac{1}{2}}{x - x_1^*} = \frac{x^2 e^{-x} - \frac{1}{2}}{x - 1.48796}.$$

Figure 2 shows on the left the graph of $f_2$ over $[-1, 3]$ and on the right that graph superimposed on that of $f_1 = f$. We can observe that $x_1^*$ is not a zero of $f_2$, but that the other two zeros of $f$ are still there.

Of course, we will not stop here. We run our trusted nonlinear equation solver once more, and this
time we find \( x^*_2 = 2.61787 \). We can now define

\[
f_3 : \mathbb{R} \rightarrow \mathbb{R} \\
x \mapsto \frac{x^2 e^{-x} - \frac{1}{2}}{(x - x_1^*)(x - x_2^*)} = \frac{x^2 e^{-x} - \frac{1}{2}}{(x - 1.48796)(x - 2.61787)}.
\]

Figure 3 shows on the left the graph of \( f_3 \) over \([-1, 3]\) and on the right that graph superimposed on that of \( f_1 = f \) and \( f_2 \). This time, only one of the original three zeros is left.

One last time, we seek a zero of a nonlinear equation, this time, for \( f_3(x) = 0 \). We find, naturally, the third zero of the original equation, \( x_3^* = -0.539835 \). If we were to try applying the deflation process once more to the function, we would define

\[
f_4 : \mathbb{R} \rightarrow \mathbb{R} \\
x \mapsto \frac{x^2 e^{-x} - \frac{1}{2}}{(x - x_1^*)(x - x_2^*)(x - x_3^*)} = \frac{x^2 e^{-x} - \frac{1}{2}}{(x - 1.48796)(x - 2.61787)(x + 0.539835)}
\]

The graph of this function is shown on the left side of Figure 4, while the right side shows \( f \) and all its deflations. We can verify that all the zeros of the function over the interval \([-1, 3]\) have been found.

**What to do, Part IV:** Implement the deflation algorithm and test it with the following function over the interval \([0, 10]\):

\[
g : \mathbb{R} \rightarrow \mathbb{R} \\
x \mapsto 4 \left( (x - 1) \sqrt{x} - x \sqrt{x + 3} \cos \left( \sqrt{x} \ln x - \sin x \right) \right) - 1.
\]

**What to do, Part V:** You should have enough experience in numerical computation by now to see that the deflation algorithm is not without flaws. List and discuss the problems you see with this algorithm, and if possible illustrate them with an example.
4 Application: The Bouncing Ball

This application will give you a chance to re-use some of the Maple and C++/Java code that you have started developing.

4.1 The model

The problem we are trying to solve is as shown in Figure 5.

The “ground” is defined as curved, smooth surface (represented by an algebraic equation). We throw a ball from a position above the surface. The ball is submitted to gravity forces and bounces off the surface as shown in Figure 6.

As shown in the figure, the balls bounces off the surface so that its incoming and outgoing directions are symmetric relative to the local normal to the surface. We also impose that the surface absorbs some of the ball’s energy, so that the ball’s speed is reduced by a constant factor $\alpha \in (0, 1)$ ($\alpha$ is a characteristics of the surface).
4.2 The data

The data consist in

- A Function1D object providing the equation of the surface. Start with something simple, then go wild with a function that will provide lots of bumps for our ball to bounce around. Obviously, your function should return mostly negative values, that is, altitudes below the starting point \((0, 0)\). Otherwise there won’t be much bouncing around, only an endless fall.

- The speed \(V\) angle \(\gamma \in [0, \pi/2]\) at which the ball is initially thrown from point \((0, 0)\).

- The “reflection coefficient \(\alpha \in (0, 1)\)

5 Evaluation

5.1 What to hand in

Please read this part carefully. You will be penalized (see next subsection) if you don’t follow the rules and the TA ends up wasting time converting files just to be able to evaluate your work.

A: You should upload the following (the procedure to follow will be clarified by the end of this week):

1. Next week: the file of a well-commented Maple worksheet. Please make

2. Two weeks from now: A complete, cleaned-up CodeWarrior, gcc (mingw or cygwin), or Eclipse project folder.

- Complete means that everything needed to compile and execute the project is there: source files, project file or make file, header files (\(\text{.h}\) files) and precompiled header source files (\(\text{.pch}\) files) if you program in C++. Of course, your test data files should be there as well, but the TA might use other data sets to test your code.

- Cleaned-up means that files resulting from the compilation should be removed: \(\text{.exe}\) files, \(\text{.sym}\) files, \(\text{.o}\) object files (in C++), \(\text{.class}\) files (in Java), precompiled header
binary files (if in C++). If you use CodeWarrior, the entire <name of project> \texttt{data}
folder should be deleted as well (close the project before you throw away this folder).

3. Two weeks from now: Your report as a Word, HTML, or Acrobat (.pdf) document.

\textbf{B:} You should hand in printouts of your (complete) Maple worksheet and of your report on the day
the corresponding part of the assignment is due.

5.2 \textbf{Point distribution}

The maximum number of points is 100, but extra points could be awarded for excellent aspects of
the project or report. The point distribution for this assignment is as follows:

- \textbf{Maple Modelling}
  - Accomplishes what was demanded \hspace{1cm} 20 pts
  - Comments and analysis \hspace{1cm} 10 pts

- \textbf{C++/Java Code}
  - Accomplishes what was demanded \hspace{1cm} 20 pts
  - Good class design \hspace{1cm} 10 pts
  - General quality \& readability \hspace{1cm} 10 pts

- \textbf{Report}
  - Discussion and analysis of the results \hspace{1cm} 20 pts
  - General quality of the writing and presentation \hspace{1cm} 10 pts

5.3 \textbf{Various point penalties}

Hopefully we won’t have to apply many of these:

- Project folder incomplete or not properly cleaned up \hspace{1cm} -5 pts
- Report file missing from the project folder \hspace{1cm} -5 pts
- Maple file missing \hspace{1cm} 0 for that part

\textbf{Late penalties}

- Printed copy of the report, 1 day late \hspace{1cm} -5 pts
- Project folder (uploaded to EnVision server), per day late \hspace{1cm} -5%

If you submit a project late, then it is your responsibility to notify the TA (with CC. to me) that the
project is finally available for download on the EnVision server. If you fail to do so, then the “late
penalty clock” will keep ticking until the TA gets around to checking your folder on the EnVision
server and notices your project. Unless asked explicitly to do so, do \textbf{not} mail your project folder
as an attachment.

As I explained in class, I will post sometimes partial solutions to the assignments, to make sure
that nobody gets too far behind. Obviously, it will be impossible to get any point on a part after a
solution for it has been posted (this does not affect the late penalty count).