Due date: Monday, May 05, at the beginning of the class.

1 Objectives of this Assignment

Our plans for this assignment are

- To implement optimization algorithms in 1D and n-D;
- To apply these techniques to a simple practical problem.

For this assignment, I impose again specifications for the Java/C++ implementation. When grading your labs, the TA will execute a program that we have written.

2 Optimization in 1D

2.1 Bracketing

When we search for a local minimum $x^*$ of a function $f : x \rightarrow f(x)$, we need to bracket that minimum, just as we had to bracket the zero of a nonlinear equation. As we saw in class, it is not enough to give the two endpoints, $a$ and $b$, of an interval, because we have no way to verify that a minimum is encountered on $[a, b]$ if the only things we know are the values taken by $f$ at $a$ and $b$. When we were seeking a zero of the function, it was sufficient to check that $f(a) f(b) < 0$. 

![Optimization Bracket](image)

Figure 1: An optimization bracket $(a, c, b)$ for a function $f$. 

A minimisation bracket is a triplet \((a, c, b)\) such that \(f(a) > f(c)\) and \(f(b) > f(c)\), as shown in Figure 1. If these conditions are verified, then we know that at least one local minimum must exist over \([a, b]\), that is, an \(x^*\) such that

There exists a (possibly small) neighborhood \(V\) of \(x^*\) such that for all \(x\) in \(V\), \(f(x) > f(x^*)\).

Or, in more rigorous mathematical terms,

\[
\exists \varepsilon > 0, \forall x \in [a, b], |x - x^*| < \varepsilon \implies f(x) > f(x^*).
\]

To refine our estimate of the location of the minimum, we need to subdivide our bracket. If we pick a new point \(d \in [c, b]\), then we have two possible cases, based on the value of \(f(d)\):

- If \(f(d) > f(c)\), then a “safe” interpretation of the known values taken by \(f\) over \([a, b]\) gives us a generic graph that looks like the one on the left side of Figure 2, and our new bracket should be \((a, c, d)\). Note that it does not matter whether \(f(d) > f(b)\), as shown in the figure, or not.

- If \(f(d) < f(c)\), then a “safe” interpretation of the known values taken by \(f\) over \([a, b]\) gives us a generic graph that looks like the one on the right side of Figure 2, and our new bracket should be \((c, d, b)\).

\[
\begin{align*}
\text{f(d) > f(c)} & \quad \text{f(d) < f(c)} \\
\text{a} & \quad \text{a} \\
\text{c} & \quad \text{c} \\
\text{d} & \quad \text{d} \\
\text{b} & \quad \text{b}
\end{align*}
\]

Figure 2: Subdivision of the bracket.

2.2 Where to cut

We could simply say “cut the segment in the middle,” like we did for the bisection algorithm. The problem is that the next time we subdivide, we will not be able to keep the \(1/2 - 1/2\) proportion, as shown in Figure 3.

\[
\begin{align*}
\text{2/3} & \quad \text{1/3} \\
\text{a} & \quad \text{b} \\
\text{c} & \quad \text{d}
\end{align*}
\]

Figure 3: Further subdivision of the bracket: We cannot always keep the \(1/2 - 1/2\) proportion.
If our initial bracket is \((a, c, b)\), with \(c\) the midpoint of \([a, b]\) (on the left in Figure 3), then the next subdivision will take place at \(d\), which should be either the midpoint or \([a, c]\) or that of \([c, b]\). Let’s say that we select the latter, as shown in the central part of Figure 3. The problem with this subdivision is, of course that we may end up with \((a, d, c)\) as our new bracket, in which case the “central” point is not in the center at all.

To select the best proportions for our bracket, let us start from a bracket \((a, c, b)\) such that \([a, c]\) represents a fraction \(\alpha < 1/2\) of the entire segment, \([a, b]\). Of course, \([c, b]\) represents a fraction \(1 - \alpha\) of \([a, b]\) (left in Figure 4). Let us now select our next point \(d\) in segment \([c, d]\) so that \([b, d]\) represents a fraction \(\beta\) of \([a, b]\).

![Diagram of further subdivision](image)

**Figure 4: Further subdivision of the bracket: We cannot always keep the 1/2 – 1/2 proportion.**

The first thing we can deduce from the proportions of the new bracket, as shown on the right side of Figure 4, is that we should take \(\alpha = \beta\). Next, if we want the new bracket to have the same proportions as the old one, we need to select \(\alpha\) such that

\[
\frac{1 - 2\alpha}{1 - \alpha} = \alpha,
\]

which, after you develop it, gives you a nice polynomial equation in terms of \(\alpha\):

\[
\alpha^2 - 3\alpha + 1 = 0.
\]

The solution of this equation that lies between 0 and 1 is \(\alpha = (3 - \sqrt{5})/2\), so that \(1 - \alpha = (\sqrt{5} - 1)/2\), also known as the golden ratio.

### 2.3 What to do, Part I

implement the golden section search minimisation algorithm, as given in your textbook, on Page 271.

### 3 Optimization in dimension \(n\)

This time we are dealing with a real function of \(n\) variables:

\[
f : \mathbb{R}^n \longrightarrow \mathbb{R}
\]

\((x_1, \ldots, x_n) \mapsto f(x_1, \ldots, x_n).

Our objective is still to find a local minimum \(X^*\) for \(f\). Let us consider the current “point” \(X\) in the vicinity of the minimum
3.1 A naive algorithm

The simplest way to implement this algorithm is just to compute a new gradient at each step and advance in the direction opposite to the gradient by a small amount (the step of the iteration). Ideally, you would want to reduce the length of the step as you get closer to the minimum.

3.2 Somewhat better: Steepest descent

The somewhat improved version of this algorithm consists in computing a direction of the gradient and then compute the minimum along the direction of the gradient. To do this, you just have to apply the golden section search algorithm.

3.3 Conjugate directions

If the gradient is not available, we can still locate the minimum of our function if we operate as follows. First we need to define a set of $n$ directions $u_1, u_2, \ldots, u_n$ that span the entire space. Good initial values for these directions would be the $n$ axes of our space, the $e_i$.

Then we can repeat the following until convergence is obtained

starting from a point $X_0$, for $i$ from 1 to $n$ do

Starting from $X_{i-1}$, minimize $f$ along the direction $u_i$. This gives a new point $X_i$

end do

update the directions $u_i$

A simple way to “update” the $u_i$ is simply to do

$u_i = u_{i+1}$ for $i = 1 \ldots n - 1$

$u_n = X_n - X_0$.

3.4 Maple implementation

Implement all three algorithms.

4 C++/Java Implementation

I am a bit short on time right now. I will post the C++ and Java specifications tonight (Monday 04/28) on the web site.

5 Evaluation

5.1 What to hand in

A: You should upload to the EnVision server a folder containing the following:
1. A complete, cleaned-up CodeWarrior project folder.
   - Complete means that everything needed to compile and execute the project is there: source files, project file, header files (.h files) and precompiled header source files (.pch files) if you program in C++, test data files.
   - Cleaned-up means that files resulting from the compilation should be removed: .exe files, .sym files, precompiled header binary files (if in C++), and the entire `<name of project>.data` folder (close the project before you throw away this folder).

2. Maple worksheets if you did any prototyping using Maple (either remove the results from the worksheet or export the worksheet as text first, please).


**B:** You should hand in hard copies of your report and of your Maple/muPad worksheet.

### 5.2 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:

<table>
<thead>
<tr>
<th>Map/Maple Modelling</th>
<th>Accomplishes what was demanded</th>
<th>25 pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comments and analysis</td>
<td></td>
<td>10 pts</td>
</tr>
<tr>
<td><strong>C++/Java Code</strong></td>
<td>Accomplishes what was demanded</td>
<td>15 pts</td>
</tr>
<tr>
<td></td>
<td>Good class design</td>
<td>10 pts</td>
</tr>
<tr>
<td></td>
<td>General quality &amp; readability</td>
<td>10 pts</td>
</tr>
<tr>
<td><strong>Report</strong></td>
<td>Discussion and analysis of the results</td>
<td>20 pts</td>
</tr>
<tr>
<td></td>
<td>General quality of the writing and presentation</td>
<td>10 pts</td>
</tr>
</tbody>
</table>

### 5.3 Various point penalties

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

- Project folder incomplete or not properly cleaned up -5 pts
- Report file missing from the project folder -5 pts
- Maple file missing 0 for that part

**Late penalties**

- Printed copy of the report, 1 day late -5 pts
- Project folder (uploaded to EnVision server), per day late -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 not mail your project folder as an attachment.