The Golden Section Search Method

1. Derivation of the Method
   - optimization with interval reduction
   - solving a minimax problem

2. Writing a Julia Function
   - input/output specification
   - documenting and defining
   - running the function

3. Analysis of the Method
   - cost and convergence

MCS 471 Lecture 7
Numerical Analysis
Jan Verschelde, 7 September 2022
The Golden Section Search Method

1. Derivation of the Method
   - optimization with interval reduction
   - solving a minimax problem

2. Writing a Julia Function
   - input/output specification
   - documenting and defining
   - running the function

3. Analysis of the Method
   - cost and convergence
Definition

A continuous function \( f \) over an interval \([a, b]\) is unimodal if \( f \) has only one minimum in \([a, b]\).

The above definition applies also to the maximum.

We want to find the minimum of a function \( f \) that is unimodal to narrow \([a, b]\) that contains the minimum, similar to bisection.

Two goals:

1. we want an optimal reduction factor for the interval,
2. with a minimum number of function calls.
our running example \( f(x) = \exp(x) - 2x \) over \([0, 1]\)
try to bisect
compare with the midpoint

In the bisection method we evaluate at the midpoint. Consider \( m = \frac{a + b}{2} \) and a small \( \delta > 0 \).

- Let \( x_1 = m - \frac{\delta}{2} \) and \( x_2 = m - \frac{\delta}{2} \).
- If \( f(x_1) < f(x_2) \) then continue with \([a, x_1]\)
  - else continue with \([x_2, b]\)

Two conditions on \( \delta \):
1. \( \delta \) is small enough for the minimum not in \([x_1, x_2]\), and
2. \( \delta \) is large enough for \( f(x_1) \neq f(x_2) \).

**Problem:** we need two function evaluations in each step.

*Can we do better?*
The Golden Section Search Method

1. Derivation of the Method
   - optimization with interval reduction
   - solving a minimax problem

2. Writing a Julia Function
   - input/output specification
   - documenting and defining
   - running the function

3. Analysis of the Method
   - cost and convergence
We have a minimax problem:

- **minimize** the number of function evaluations;
- **maximize** the accuracy of the optimal value.

To solve this problem, we will require a constant reduction factor of each interval. If \([a_k, b_k]\) is the interval in step \(k\), then we want

\[
b_{k+1} - a_{k+1} = c(b_k - a_k),
\]

for some constant factor \(c < 1\).
two cases

For \( x_1 \) and \( x_2 \) somewhere in \([a, b]\), there are two cases:

1. If \( f(x_1) < f(x_2) \), then \([a, b] := [a, x_2]\), with interval size reduction

\[
x_2 - a = c(b - a) \quad \Rightarrow \quad x_2 = a + cb - ca
\]
\[
\Rightarrow \quad x_2 = (1 - c)a + cb.
\]

2. If \( f(x_1) > f(x_2) \), then \([a, b] := [x_1, b]\), with interval size reduction

\[
b - x_1 = c(b - a) \quad \Rightarrow \quad -x_1 = cb - ca - b
\]
\[
\Rightarrow \quad x_1 = ca + (1 - c)b.
\]

If we know the value for \( c \),
then we know the location of \( x_1 \) and \( x_2 \).
fixed positions of $x_1$ and $x_2$

The constant reduction factor $c$ fixes $x_1$ and $x_2$. 

\[
\begin{align*}
x_1 &= a + c(b - a) \\
x_2 &= b - c(b - a)
\end{align*}
\]
two simplifications

*Can we make this any simpler?*

1. Without loss of generality, focus on \( f(x_1) < f(x_2) \).
2. For ease of calculations, set \([a, b] = [0, 1] \).

Let us apply the simplifications.

If \( f(x_1) < f(x_2) \), then \([0, x_2]\) is the new interval.

The condition of a constant reduction factor \( c \) (and \([a, b] = [0, 1] \)): \[
\begin{align*}
x_1 &= b - c(b - a) = 1 - c \\
x_2 &= (1 - c)a + cb = c.
\end{align*}
\]

In the new interval \([0, c]\),

1. we recycle \( x_1 = 1 - c \),
2. we do one new function evaluation, at \( x_2 \).

Where should \( x_2 \) be? To the left of \( 1 - c \) or to the right of \( 1 - c \)?
try $x_2$ to the right of $x_1$

Suppose we place a new function evaluation at the right of $x_1 = 1 - c$, then $x_1$ is the left point of the interval $[0, c]$, and we write $x_1$ in two ways (once inside $[0, 1]$, once inside $[0, c]$):

$$x_1 = 1 - c = c0 + (1 - c)c \quad \Rightarrow \quad (1 - c)^2 = 0$$

The (double) root of this equation is 1, which gives no reduction, so we exclude this possibility.
try \( x_2 \) to the left of \( x_1 \)

Suppose we place a new function evaluation at the left of \( x_1 = 1 - c \), then \( x_1 \) is the right point of the interval \([0, c]\), and we write \( x_1 \) in two ways (once inside \([0, c]\), we set \( x_2 = x_1 \)),

\[
1 - c = (1 - c)0 + cc \quad \Rightarrow \quad c^2 + c - 1 = 0
\]

The positive root leads to \( c = \frac{-1 + \sqrt{5}}{2} \), which equals approximately 0.6180.
the golden section

We found \( c = \frac{-1 + \sqrt{5}}{2} \), as the positive root of \( c^2 + c - 1 = 0 \).

The constant \( c \) satisfies

\[
\begin{align*}
  c^2 + c - 1 &= 0 \quad \Rightarrow \quad c^2 + c = 1, \\
  \Rightarrow c + 1 &= 1 / c.
\end{align*}
\]

Adding 1 to \( c \) gives

\[
\frac{-1 + \sqrt{5}}{2} + 1 = \frac{1 + \sqrt{5}}{2} = \varphi, \quad \text{the golden ratio}.
\]
The Golden Section Search Method

1. Derivation of the Method
   - optimization with interval reduction
   - solving a minimax problem

2. Writing a Julia Function
   - input/output specification
   - documenting and defining
   - running the function

3. Analysis of the Method
   - cost and convergence
**input/output specification**

The essential input arguments are

- a function \( f(x) \) in one variable \( x \), and
- the bounds \( a \) and \( b \) of the interval \( [a, b] \).

What are the numerical parameters?

1. \( N \) is an upper bound on the number of function evaluations.
2. \( \delta \) is the tolerance on the forward error: \( b - a < \delta \).
3. \( \epsilon \) is the tolerance on the backward error: \( |f(x_1) - f(x_2)| < \epsilon \).

For the numerical parameters we provide default values which are well suited for an illustrative example.

The specifying, documenting, defining, and running are presented here as four distinct stages, one after the other. In practice, the development intermingles those four stages.
The Golden Section Search Method

1. Derivation of the Method
   - optimization with interval reduction
   - solving a minimax problem

2. Writing a Julia Function
   - input/output specification
   - documenting and defining
   - running the function

3. Analysis of the Method
   - cost and convergence
documenting the function

using Printf

""
Runs the golden section search on the function f to approximate the minimum of f over an interval [a, b].

Assumed is that f is continuous on [a, b] and that f has only one minimum in [a, b].

No more than N function evaluations are done.

The iteration stops when b - a < dxtol, or when |f(x1) - f(x2)| < fxtol.
documentation continued

ON ENTRY:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>is a function in one variable, unimodal on the interval ([a, b])</td>
</tr>
<tr>
<td>a</td>
<td>is the left bound of an interval</td>
</tr>
<tr>
<td>b</td>
<td>is the right bound of an interval</td>
</tr>
<tr>
<td>dxtol</td>
<td>is the tolerance on the forward error</td>
</tr>
<tr>
<td>fxtol</td>
<td>is the tolerance on the backward error</td>
</tr>
<tr>
<td>N</td>
<td>is maximum number of function evaluations</td>
</tr>
</tbody>
</table>

ON RETURN: \((a, b, \text{fail})\)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>is the new left of ([a, b]) enclosing the minimum</td>
</tr>
<tr>
<td>b</td>
<td>is the new right of ([a, b]) enclosing the minimum</td>
</tr>
<tr>
<td>fail</td>
<td>is true if the accuracy requirements are not met, false otherwise</td>
</tr>
</tbody>
</table>

EXAMPLE:

\[
f(x) = \exp(x) - 2*x;
\]

\[
(a, b, \text{fail}) = \text{gss}(f, 0.0, 1.0)
\]

```c
function gss(f::Function, a::Float64, b::Float64,
            dxtol::Float64=1.0e-2, fxtol::Float64=1.0e-04, N::Int64=20)
```

```c

```

Numerical Analysis (MCS 471) Golden Section Search Method L-7  7 September 2022 19/28
statements before the loop

c = (-1+sqrt(5))/2
x1 = c*a + (1-c)*b
fx1 = f(x1)
x2 = (1-c)*a + c*b
fx2 = f(x2)
dfx = abs(fx2 - fx1)
stri = @sprintf("%3d", 0)
strx1 = @sprintf("%.4e", x1)
strx2 = @sprintf("%.4e", x2)
strfx1 = @sprintf("%.4e", fx1)
strfx2 = @sprintf("%.4e", fx2)
strbma = @sprintf("%.4e", b-a)
title = " x1 x2 f(x1)"
println("$title f(x2) b - a")
println("$stri : $strx1 $strx2 $strfx1 $strfx2 $strbma")
the first part of the loop

```
for i = 1:N-2
    if fx1 < fx2
        b = x2
        x2 = x1
        fx2 = fx1
        x1 = c*a + (1-c)*b
        fx1 = f(x1)
    else
        a = x1
        x1 = x2
        fx1 = fx2
        x2 = (1-c)*a + c*b
        fx2 = f(x2)
    end
```

**Exercise 1:** The statements following the `if` test were justified in the derivation of the method. Justify the statements in the `else` case, when \( f(x_1) > f(x_2) \).
the stop criteria

```
stri = @sprintf("%3d", i)
strx1 = @sprintf("%.4e", x1)
strx2 = @sprintf("%.4e", x2)
strfx1 = @sprintf("%.4e", fx1)
strfx2 = @sprintf("%.4e", fx2)
strbma = @sprintf("%.4e", b-a)
println("$stri : $strx1 $strx2 $strfx1 $strfx2 $strbma")
if (abs(b-a) < dxtol) | (abs(fx2 - fx1) < fxtol)
    stri = string(i+2)
    println("succeeded after $stri function evaluations")
    return (a, b, false);
end
end
strN = string(N)
println("failed requirements after $strN function evaluations")
return (a, b, true)
end
```

Look at the posted program! goldensection.jl.
The Golden Section Search Method

1. Derivation of the Method
   - optimization with interval reduction
   - solving a minimax problem

2. Writing a Julia Function
   - input/output specification
   - documenting and defining
   - running the function

3. Analysis of the Method
   - cost and convergence
running the function

julia> include("goldensection.jl");

help?> gss

... read the documentation ...

julia> f(x) = exp(x) - 2*x; (a, b, fail) = gss(f,0.0,1.0)

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>f(x1)</th>
<th>f(x2)</th>
<th>b - a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 : 3.8197e-01</td>
<td>6.1803e-01</td>
<td>7.0123e-01</td>
<td>6.1921e-01</td>
<td>1.0000e+00</td>
</tr>
<tr>
<td>6 : 6.9505e-01</td>
<td>7.0820e-01</td>
<td>6.1371e-01</td>
<td>6.1393e-01</td>
<td>5.5728e-02</td>
</tr>
</tbody>
</table>

succeeded after 9 function evaluations
(0.6737620787507361, 0.7082039324993692, false)
adjusting the tolerance

Often we may have to change the default value of the tolerances to take into account the shape of the function.

Exercise 2: Consider the functions $f(x) = x^2, x^4, x^8$ over $[-0.5, 0.5]$.

1. Run `gss` on $f(x) = x^2, x^4, x^8$ and report the number of function evaluations and the accuracy of the result. Write one concluding sentence.

2. We want the same results on $f(x) = x^4$ and on $f(x) = x^8$ as on the results on $f(x) = x^2$ with the default tolerances.

Adjust the default tolerance on the backward error for runs on $x^4$ and $x^8$ (if necessary) so the number of function evaluations and the accuracy is the same for all three functions.

What are the new tolerances for $f(x) = x^4$ and $x^8$?
The Golden Section Search Method

1. Derivation of the Method
   - optimization with interval reduction
   - solving a minimax problem

2. Writing a Julia Function
   - input/output specification
   - documenting and defining
   - running the function

3. Analysis of the Method
   - cost and convergence
The interval that encloses the minimum is reduced by the constant factor \( c \approx 0.6180 \), at the cost of one new function evaluation.

**Exercise 3:** Suppose we have an interval enclosing the minimum of a unimodal function of length equal to \( 1.0 \times 10^{-1} \). How many function evaluations does the golden section search need to reduce the length of the enclosing interval to \( 1.0 \times 10^{-4} \)? Justify!

*For further reading:*
five introductory lectures on root finding

The sentences below summarize the essence of each lecture.

1. Bisection gives one bit of accuracy per function evaluation. The convergence of fixed-point iterations is determined by the derivatives of the fixed-point equation at the fixed point.

2. The numerical conditioning depends on the difficulty to evaluate, the size of the root, and the derivative at the root.

3. Newton’s method converges quadratically to a regular root and geometrically to a multiple root, when sufficiently close to a root.

4. The secant method converges superlinearly, when sufficiently close to a root.

5. The golden section search method reduces the interval that encloses the optimum by a constant factor in each step with one new function evaluation.