Homework 2

Two bisection procedures
For first (Bisection0) you input a function f, interval endpoints a,b and number of iterations.
For first (Bisection0) you input a function f, interval endpoints a,b and desired maximal error.

> restart;

> Bisection0 := proc(f,a,b,n)
local x,A,B,j; #a assumed < b
A:=evalf(a);
B:=evalf(b);
if (f(A)*f(B)>=0) then error "Your function does not have opposite signs on the interval endpoints.";
elif (n <= 0) then error "Your targeted number of iterations must be positive";
else
for j from 1 to n do
x:=(A+B)/2.0;
if (f(A)*f(x)<0) then B:= x; else A:= x;
fi;
od;
x;
fi;
end proc;

Bisection0 := proc(f,a,b,n)  
local x,A,B,j;  #a assumed < b
A := evalf(a);
B := evalf(b);
if 0 <= f(A)*f(B) then
   error "Your function does not have opposite signs on the interval endpoints."
else
   n := ceil(ln(epsilon/(B-A))/ln(0.5));
   for j from 1 to n do
      x := (A+B)/2.0; if f(A)*f(x) < 0 then B := x else A := x end if
   end do;
x;
end if;
end proc
\[ x := (A + B) / 2.0; \]
\[ \text{if } (f(A) * f(x) < 0) \text{ then } B := x; \text{ else } A := x; \]
\[ \text{fi;} \]
\[ \text{od;} \]
\[ x; \]
\[ \text{fi;} \]
\[ \text{end proc;} \]

\textit{Bisection} := \textbf{proc}(f, a, b, \epsilon) \tag{2} \]
\[
\text{local } n, x, A, B, f; \\
A := \text{evalf}(a); \\
B := \text{evalf}(b); \\
\text{if } 0 \leq f(A) \times f(B) \text{ then} \\
\quad \text{error } \text{"Your function does not have opposite signs on the interval endpoints."} \\
\text{elif } \epsilon \leq 0 \text{ then} \\
\quad \text{error } \text{"Your targeted error must be positive"} \\
\text{else} \\
\quad n := \text{ceil}(\ln(\epsilon / (B - A)) / \ln(0.5)); \\
\quad \text{for } j \text{ to } n \text{ do} \\
\quad\quad x := (A + B) / 2.0; \text{ if } f(x) \times f(A) < 0 \text{ then } B := x \text{ else } A := x \text{ end if} \\
\quad\quad \text{end do;} \\
\quad x \\
\text{end if} \\
\text{end proc} \]

Let's try the routines out. Remember we need the number of iterations \(n\) equal to the smallest positive integer satisfying \((b-a)/2^n < \text{desired error}\)

\[ x := (A + B) / 2.0; \]
\[ \text{if } (f(A) \times f(x) < 0) \text{ then } B := x; \text{ else } A := x; \]
\[ \text{fi;} \]
\[ \text{od;} \]
\[ x; \]
\[ \text{fi;} \]
\[ \text{end proc;} \]

\text{Problem 1 on page 53}
\[ f := x \rightarrow \sqrt{x} - \cos(x); \]
\[ f := x \rightarrow \sqrt{x} - \cos(x) \tag{6} \]

\text{Problem 2a on page 53}
\[ f := x \rightarrow 3 \times (x+1) \times (x-1/2) \times (x-1); \]
\[ f := x \rightarrow 3 \times (x+1) \left( x - \frac{1}{2} \right) (x - 1) \tag{8} \]

\[ Bisection0(f,-2,1.5,3); \]
\[-0.6875000000 \tag{9} \]
> (1.5+2)/8.0;
Bisection(f,-2,1.5,0.4375);
> f := x -> exp(x) - x^2 + 3*x - 2;
> f(0.0); f(1.0);
> Bisection0(f,0.0,1.0,14);
> Bisection(f,0.0,1.0,10^(-5));
Problem 5b on page 53
To see if bisection works, we first check that f(x) has values with different signs at the endpoints 0, 1
> f(0.0); f(1.0);
The smallest n with (1-0)/2^n < 10^(-5) is n = 14
> Bisection0(f,0.0,1.0,14);
> Bisection(f,0.0,1.0,10^(-5));
Problems 1c and 1d on page 63
> expand((x^2+2)*x^2-(x+3));
> expand(x*(4*x^3+4*x-1)-(3*x^4+2*x^2+3));
Let's check the real roots
> sol:= fsolve(x^4+2*x^2-x-3,x);
> g3:= x -> sqrt((x+3)/(x^2+2));
p0 := 1.0;
for j from 1 to 4 do
p||j := g3(p||(j-1));
od;
error3:=sol[2]-p4;
g4:= x -> (3*x^4+2*x^2+3)/(4*x^3+4*x-1);
p0 := 1.0;
for j from 1 to 4 do
Iteration using 1d is considerably better than iteration using 1c.

The first procedure newton0 prints out each iteration. The second just gives the newton estimate.

At this stage, we are using the crude stopping criteria for newton's method of $|x_n - x_{n+1}| < \text{tolerance}$.

```plaintext
> newton0 := proc(p,tol,N,StartValue)
local x,j,A,B,dp,g;
   dp:= unapply(diff(p(x),x),x);
g:= x -> x - p(x)/dp(x);
   A:=StartValue;
   B:= g(A);
   print(1,B);
   for j from 2 to N while (abs(A - B) >= tol) do
      A:= B:
      B:= g(A);
      print(j,B);
   od:
end proc;
```

```plaintext
> newton0 := proc(p,tol,N,StartValue)
local x,j,A,B,dp,g;
   dp := unapply(diff(p(x),x),x);
g := x -> x - p(x)/dp(x);
   A := StartValue;
   B := g(A);
   print(1,B);
   for j from 2 to N while abs(A - B) >= tol do
      A := B;
      B := g(A);
      print(j,B);
   od:
end proc;
```

```plaintext
> p:= x -> exp(x) + 2.0^(-x) + 2.0*cos(x) - 6.0;
```

```plaintext
> plot(p(x),x = -4..2.5);
```
For start values, I will take 2.0; 1.5; and finally 1.0.
Note the difference between taking 1 and 2 as starting values.

> `newton0(p,10^(-5),10,2.0);`

1, 1.850521336
2, 1.829751202
3, 1.829383715
4, 1.829383602

> `newton0(p,10^(-5),10,1.50);`

1, 1.956489721
2, 1.841533061
3, 1.829506013
4, 1.829383615
5, 1.829383602

> `newton0(p,10^(-5),10,1.0);`

1, 3.469798013
2, 2.726126471
The difference in number of iterations mainly comes from \( x = 2 \) being closer to the solution. Let's start far away.

```maple
> newton0(p,10^(-5),15,0.2);
1, 9.028292387
2, 8.029140543
3, 7.030566191
4, 6.033364960
5, 5.044214323
6, 4.090000311
7, 3.231730850
8, 2.545996298
9, 2.088208818
10, 1.874834972
11, 1.831047530
12, 1.829385920
13, 1.829383602
```

```maple
> newton := proc(p,tol,N,StartValue)
local x,j,A,B,dp,g;
dp:= unapply(diff(p(x),x),x);
g:= x -> x - p(x)/dp(x);
A:=StartValue;
B:= g(A);
for j from 2 to N while (abs(A - B) >= tol) do
A:= B:
B:= g(A);
end;
if abs(A-B) >= tol then
print(B,abs(A-B));
error "tolerance has not been achieved in your requested maximum number of iterations"  else B
fi;
end proc;
```

```maple
newton := proc(p, tol, N, StartValue)
  local x, j, A, B, dp, g;
  dp := unapply(diff(p(x), x), x);
  g := x -> x - p(x)/dp(x);
  A := StartValue;
  B := g(A);
  for j from 2 to N while tol <= abs(A - B) do A := B; B := g(A) end do;
  if tol <= abs(A - B) then
```

Problem 6a on page 75: we start with a procedure for the secant method

```maple
> secant0 := proc(p,tol,N,StartValue0,StartValue1)
local x,j,A,B,C,g;
g := (x,previous_x) -> x - p(x)*(x-previous_x)/(p(x)-p(previous_x));
A := StartValue0;
B := StartValue1;
C := g(A,B);
print(1,C);
for j from 2 to N while (abs(C - B) >= tol) do
A:= B:
B:= C;
C:= g(A,B);
print(j,C);
end do:
end proc;

secant0 := proc(p, tol, N, StartValue0, StartValue1)
    local x, j, A, B, C, g;
g := (x, previous_x) -> x - p(x)*(x - previous_x)/(p(x) - p(previous_x));
A := StartValue0;
B := StartValue1;
C := g(A, B);
print(1, C);
for j from 2 to N while tol <= abs(A - B) do
    A := B; B := C; C := g(A, B); print(j, C)
end do
end proc

What start values? I will do 2, 2.1; 1, 2; and finally 0.9, 1.1
> secant0(p, 10^(-5), 20, 2.0, 2.1);

1, 1.861612511
2, 1.835794637
3, 1.829553861
4, 1.829384514
5, 1.829383602
6, 1.829383602
```

Note: The above code snippet is a Maple procedure for the secant method. It finds the root of a function `p` within a specified tolerance `tol` and maximum number of iterations `N`, starting with two initial guesses `StartValue0` and `StartValue1`. The output provides the iteration count and the corresponding root approximation.
secant0 := proc(p,tol,N,StartValue0,StartValue1)
local x,j,A,B,C,g;
g := (x,previous_x) -> x - p(x)*(x-previous_x)/(p(x)-p(previous_x));
A := StartValue0;
B := StartValue1;
C := g(A,B);
for j from 2 to N while (abs(A - B) >= tol) do
A:= B;
B:= C;
C:= g(A,B);
od:
if abs(A-B) >= tol then
error "tolerance has not been achieved in your requested maximum number of iterations" else C;
fi;
end proc;

secant := proc(p, tol, N, StartValue0, StartValue1) 
local x,j,A,B,C,g;
g := (x,previous_x) -> x - p(x)*(x-previous_x)/(p(x)-p(previous_x));
A := StartValue0;
B := StartValue1;
C := g(A,B);
for j from 2 to N while tol <= abs(A - B) do A := B; B := C; C := g(A, B) end do;
if tol <= abs(A - B) then

> secant0(p,10^(-5),20,1.0,2.0);
1, 1.678308485
2, 1.808102877
3, 1.832298464
4, 1.829331173
5, 1.829383474
6, 1.829383602
7, 1.829383602

> secant0(p,10^(-5),20,0.9,1.0);
1, 3.826865984
2, 1.119796739
3, 1.228222072
4, 2.597286536
5, 1.504383199
6, 1.665017395
7, 1.886197421
8, 1.821189172
9, 1.829001742
10, 1.829386235
11, 1.829383601
12, 1.829383602

secant := proc(p, tol, N, StartValue0, StartValue1) 
local x,j,A,B,C,g;
g := (x,previous_x) -> x - p(x)*(x-previous_x)/(p(x)-p(previous_x));
A := StartValue0;
B := StartValue1;
C := g(A,B);
for j from 2 to N while (abs(A - B) >= tol) do
A:= B;
B:= C;
C:= g(A,B);
od:
if abs(A-B) >= tol then
error "tolerance has not been achieved in your requested maximum number of iterations" else C;
fi;
end proc;

secant := proc(p, tol, N, StartValue0, StartValue1) 
local x,j,A,B,C,g;
g := (x,previous_x) -> x - p(x)*(x-previous_x)/(p(x)-p(previous_x));
A := StartValue0;
B := StartValue1;
C := g(A,B);
for j from 2 to N while tol <= abs(A - B) do A := B; B := C; C := g(A, B) end do;
if tol <= abs(A - B) then
error
"tolerance has not been achieved in your requested maximum number of iterations"
else
C
end if
end proc
> secant(p,10^-5,20,1.0,2.0);
1.829383602

(35)

13bc
> f := x -> 230*x^4 + 18*x^3 + 9*x^2 - 221*x - 9;

(36)

> plot(f(x),x=-1..1);

> secant0(f,10^-6,20,-1.0,0.0);

1, -0.0203619910
2, -0.04069125644
3, -0.04065926258
Note we always seem to get the negative solution! Let's try closer to 1.

> \texttt{secant0} (f, 10^{-6}, 20, 0.5, 1.0);

<table>
<thead>
<tr>
<th>\text{Iteration}</th>
<th>\text{Result}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.705089820</td>
</tr>
<tr>
<td>2</td>
<td>-0.323791140</td>
</tr>
<tr>
<td>3</td>
<td>-0.0646031310</td>
</tr>
<tr>
<td>4</td>
<td>-0.04068615117</td>
</tr>
<tr>
<td>5</td>
<td>-0.04065928834</td>
</tr>
<tr>
<td>6</td>
<td>-0.04065928832</td>
</tr>
</tbody>
</table>

> \texttt{newton0} (f, 10^{-6}, 20, 0.8);

<table>
<thead>
<tr>
<th>\text{Iteration}</th>
<th>\text{Result}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.056240803</td>
</tr>
<tr>
<td>2</td>
<td>0.9765537980</td>
</tr>
<tr>
<td>3</td>
<td>0.9627864090</td>
</tr>
<tr>
<td>4</td>
<td>0.9623987210</td>
</tr>
<tr>
<td>5</td>
<td>0.9623984188</td>
</tr>
</tbody>
</table>