Homework 2

Two bisection procedures

For first (Bisection0) you input a function f, interval endpoints a,b and number of itterations.

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*) **(1)**

local *x*, *A*, *B*, *j*;
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."
elif *n* <= 0 **then**
  error "Your targeted number of iterations must be positive"
else
  for *j* **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

```
> Bisection := proc(f,a,b,epsilon)
  local n, 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
   (epsilon <=0) then error "Your targeted error must be positive";
  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;
fi;
od;
x;
fi;
end proc;
```

Bisection := **proc**(f, a, b, ϵ) (2)

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

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

```
> g:= x -> x^2-2.0;
```

$$g := x \rightarrow x^2 - 2.0 \quad (3)$$

```
> Bisection(g,1.0,2.0,0.05);
  abs((%-sqrt(2.0)));
```

1.406250000

0.007963562 (4)

```
> Bisection0(g,1.0,2.0,5);
```

1.406250000 (5)

Problem 1 on page 53

```
> f:= x -> sqrt(x) - cos(x);
```

$$f := x \rightarrow \sqrt{x} - \cos(x) \quad (6)$$

```
> Bisection0(f,0,1,3);
```

0.6250000000 (7)

Problem 2a on page 53

```
> f := x -> 3*(x+1)*(x-1/2)*(x-1);
```

$$f := x \rightarrow 3\ (x + 1) \left(x - \frac{1}{2}\right) (x - 1) \quad (8)$$

```
> Bisection0(f,-2,1.5,3);
```

-0.6875000000 (9)

```
> (1.5+2)/8.0;
  Bisection(f,-2,1.5,0.4375);
```

$$0.4375000000$$
$$-0.6875000000 \tag{10}$$

Problem 5b on page 53

```
> f := x -> exp(x) - x^2 + 3*x - 2;
```

$$f := x \rightarrow e^{x} - x^{2} + 3\,x - 2 \tag{11}$$

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);
```

$$-1.$$
$$2.718281828 \tag{12}$$

The smallest n with (1-0)/2^n < 10^(-5) is n = 14

```
> Bisection0(f,0.0,1.0,14);
```

$$0.2575073242 \tag{13}$$

```
> Bisection(f,0.0,1.0,10^(-5));
```

$$0.2575302124 \tag{14}$$

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));
```

$$x^{4} + 2\,x^{2} - x - 3$$
$$x^{4} + 2\,x^{2} - x - 3 \tag{15}$$

Let's check the real roots

```
> sol:= fsolve(x^4+2*x^2-x-3,x);
```

$$sol := -0.8760531158,\ 1.124123030 \tag{16}$$

```
> g3:= x -> sqrt((x+3)/(x^2+2));
```

$$g3 := x \rightarrow \sqrt{\frac{x+3}{x^{2}+2}} \tag{17}$$

```
> p0 := 1.0;
  for j from 1 to 4 do
  p||j := g3(p||(j-1));
  od;
  error3:=sol[2]-p4;
```

$$p0 := 1.0$$
$$p1 := 1.154700538$$
$$p2 := 1.116427410$$
$$p3 := 1.126052233$$
$$p4 := 1.123638885$$
$$error3 := 0.000484145 \tag{18}$$

```
> g4:= x -> (3*x^4+2*x^2+3)/(4*x^3+4*x-1);
```

$$g4 := x \rightarrow \frac{3\,x^{4} + 2\,x^{2} + 3}{4\,x^{3} + 4\,x - 1} \tag{19}$$

```
> p0 := 1.0;
  for j from 1 to 4 do
```

```
p||j := g4(p||(j-1));
od;
error4:=sol[2]-p4;
```

$$p0 := 1.0$$

$$p1 := 1.142857143$$

$$p2 := 1.124481690$$

$$p3 := 1.124123164$$

$$p4 := 1.124123031$$

$$error4 := -1.\ 10^{-9} \quad (20)$$

Iteration using 1d is considerably better than iteration using 1c.

pg 74: 6a

The first procedure newton0 prints out each iteration. The secod 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}| < tolerance.

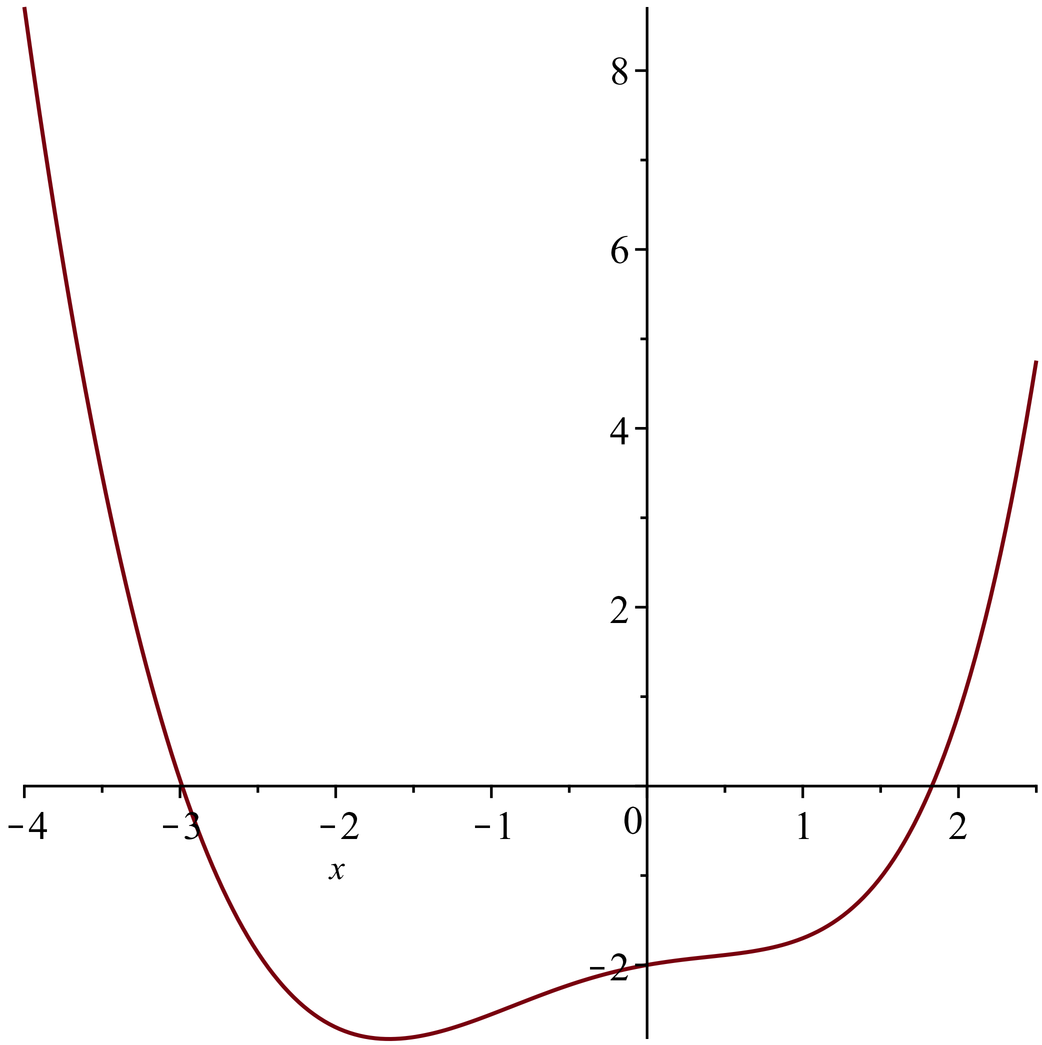
```
> newton0 := proc(p,tol,N,StartValue)
local x,j,A,B,dp,g;
dp:= unapply(diff(p(x),x),x);
g:= x -> x - p(x)/dp(x);
A:=StartValue;
B:= g(A);
print(1,B);
for j from 2 to N while (abs(A - B) >= tol) do
A:= B:
B:= g(A);
print(j,B);
od:
end proc;
```

```
newton0 := proc(p, tol, N, StartValue)                                  (21)
    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 tol <= abs(A − B) do
        A := B; B := g(A); print(j, B)
    end do
end proc
```

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

$$p := x \rightarrow e^{x} + 2.0^{-x} + 2.0\cos(x) - 6.0 \quad (22)$$

```
> plot(p(x),x = -4..2.5);
```



0.069802075 **(23)**

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 **(24)**

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

1, 1.956489721
2, 1.841533061
3, 1.829506013
4, 1.829383615
5, 1.829383602 **(25)**

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

1, 3.469798013
2, 2.726126471

$$3, 2.197294485$$
$$4, 1.914273084$$
$$5, 1.834995797$$
$$6, 1.829409874$$
$$7, 1.829383603$$
$$8, 1.829383602 \tag{26}$$

The difference in number of iterations mainly comes from x = 2 being closer to the solution. Let's start far away.

```
> newton0(p,10^(-5),15,0.2);
```

$$1, 9.028292387$$
$$2, 8.029140543$$
$$3, 7.030566191$$
$$4, 6.033364960$$
$$5, 5.044214323$$
$$6, 4.090000311$$
$$7, 3.231730850$$
$$8, 2.545996298$$
$$9, 2.088208818$$
$$10, 1.874834972$$
$$11, 1.831047530$$
$$12, 1.829385920$$
$$13, 1.829383602 \tag{27}$$

```
> 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);
  od:
  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;
```

newton := **proc**(*p*, *tol*, *N*, *StartValue*) **(28)**

local *x*, *j*, *A*, *B*, *dp*, *g*;

$dp := unapply(diff(p(x), x), x);$

$g := x \rightarrow x - p(x) / dp(x);$

$A := StartValue;$

$B := g(A);$

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**

$print(B, \mathrm{abs}(A - B));$
error
"tolerance has not been achieved in your requested maximum number of iterations"
else
B
end if
end proc

```
> newton(p,10^(-5),8,1.0);
```

$$1.829383602 \tag{29}$$

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

```
> 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);
  od:
  end proc;
```

secant0 := **proc**(*p*, *tol*, *N*, *StartValue0*, *StartValue1*) **(30)**
local *x*, *j*, *A*, *B*, *C*, *g*;
$g := (x, previous\_x) \rightarrow x - p(x) * (x - previous\_x) / (p(x) - p(previous\_x));$
$A := StartValue0;$
$B := StartValue1;$
$C := g(A, B);$
$print(1, C);$
for j **from** 2 **to** N **while** $tol <= \mathrm{abs}(A - B)$ **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 **(31)**

```
> 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 **(32)**

```
> 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 **(33)**

```
> 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;
```

$secant := \textbf{proc}(p, tol, N, StartValue0, StartValue1)$ **(34)**

$\textbf{local}\ x, j, A, B, C, g;$

$g := (x, previous\_x) \rightarrow x - p(x) * (x - previous\_x) / (p(x) - p(previous\_x));$

$A := StartValue0;$

$B := StartValue1;$

$C := g(A, B);$

$\textbf{for}\ j\ \textbf{from}\ 2\ \textbf{to}\ N\ \textbf{while}\ tol <= \mathrm{abs}(A - B)\ \textbf{do}\ A := B;\ B := C;\ C := g(A, B)\ \textbf{end do};$

$\textbf{if}\ tol <= \mathrm{abs}(A - B)\ \textbf{then}$

```
        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);
```

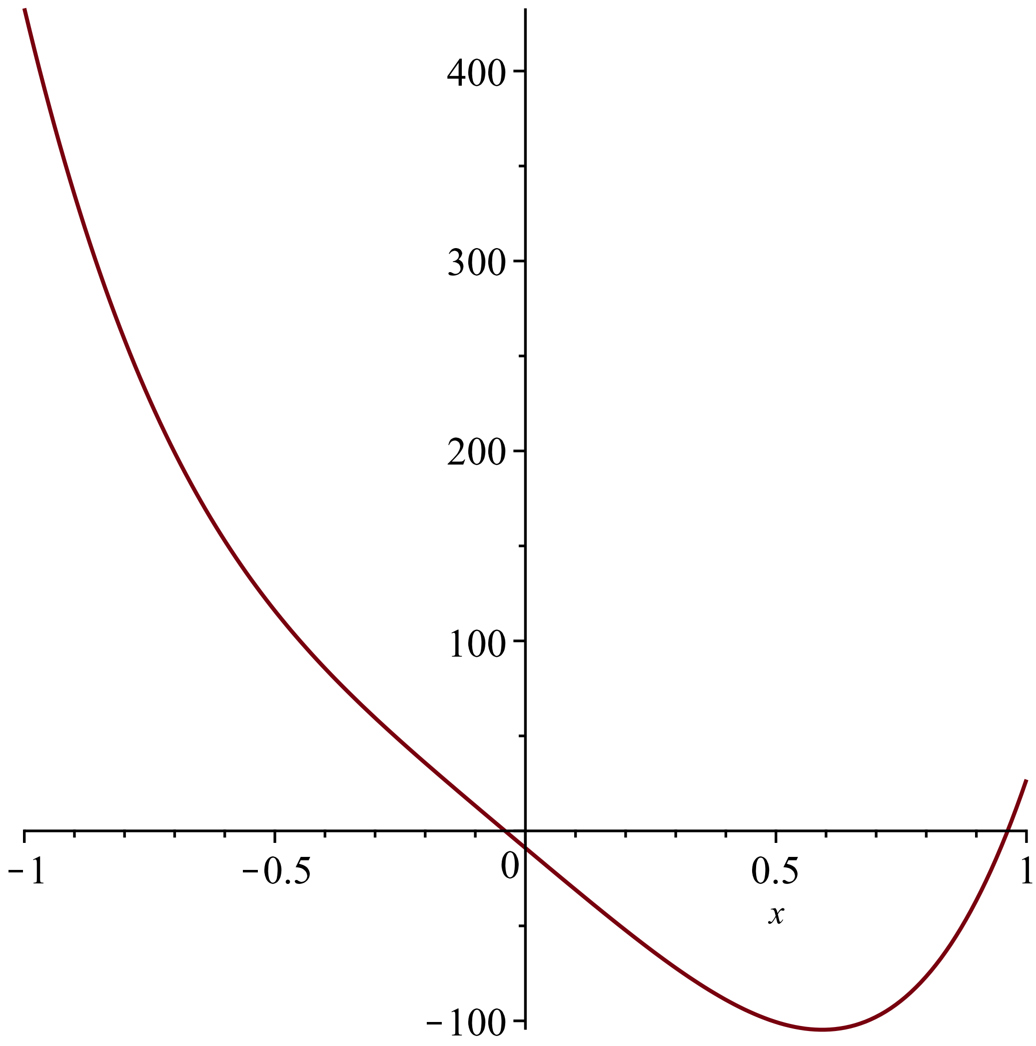
$$1.829383602 \tag{35}$$

13bc

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

$$f := x \rightarrow 230\,x^4 + 18\,x^3 + 9\,x^2 - 221\,x - 9 \tag{36}$$

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




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

1, -0.0203619910
2, -0.04069125644
3, -0.04065926258

$$4, -0.04065928832$$
$$5, -0.04065928832 \tag{37}$$

```
> newton0(f,10^(-6),20,-0.5);
```

$$1, -0.1504524887$$
$$2, -0.0418168139$$
$$3, -0.04065934350$$
$$4, -0.04065928831 \tag{38}$$

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

$$1, 0.2500000000$$
$$2, 0.7737627651$$
$$3, -1.285417781$$
$$4, 0.5945955195$$
$$5, 0.394641103$$
$$6, -0.6693181205$$
$$7, 0.0497143925$$
$$8, -0.0207541529$$
$$9, -0.04073533288$$
$$10, -0.04065922825$$
$$11, -0.04065928832$$
$$12, -0.04065928832 \tag{39}$$

```
> newton0(f,10^(-6),20,0.5);
```

$$1, -0.705089820$$
$$2, -0.3237911140$$
$$3, -0.0646031310$$
$$4, -0.04068615117$$
$$5, -0.04065928834$$
$$6, -0.04065928832 \tag{40}$$

Note we always seem to get the negative solution! Let's try closer to 1.

```
> secant0(f,10^(-6),20,0.5,1.0);
```

$$1, 0.8942213516$$
$$2, 0.9570463537$$
$$3, 0.9632104450$$
$$4, 0.9623896322$$
$$5, 0.9623984043$$
$$6, 0.9623984189$$
$$7, 0.9623984189 \tag{41}$$

```
> newton0(f,10^(-6),20,0.8);
```

$$1, 1.056240803$$
$$2, 0.9765537980$$
$$3, 0.9627864090$$
$$4, 0.9623987210$$
$$5, 0.9623984188 \tag{42}$$