Ejercicio 1.

Método de Bisección	Método de Newton.

program ej1pr2;
uses wincrt;
var
i:integer;
error:real;
a,b,c:real;
solucion:boolean;
procedure cambia(var a,b:real);
var
aux:real;
begin
aux:=a;
a:=b;
b:=aux
end;
function f(x:real):real;
begin
f:=cos(x)/x
end;
begin
solucion:=false;
repeat
write ('Dame el valor de a y b: ');
readln (a,b);
	program ej1bpr2;
	uses wincrt;
	var
	i,k:integer;
	error:real;
	a,b:real;
	solucion:boolean;
	procedure cambia(var a,b:real);
	var
	aux:real;
	begin
	aux:=a;
	a:=b;
	b:=aux
	end;
	function f(x:real):real;
	begin
	f:=cos(x)/x;
	end;
	function f_prima(x:real):real;
	begin
	f_prima:=((−sin(x)*x)−cos(x))/(x*x);
	end;
	begin

```
if a>b then cambia(a,b);

until (f(a)*f(b))<0;

write ('Dame el valor del error:');

readln(error);

while not(solucion) do

begin

c:=(a+b)/2;

solucion:=(abs(b−c)<error) or (f(c)=0);

if not solucion then

if f(b)*f(c)<0 then a:=c

else b:=c

end;

writeln;

writeln('La raiz del polinomio es ',c)

end.
```

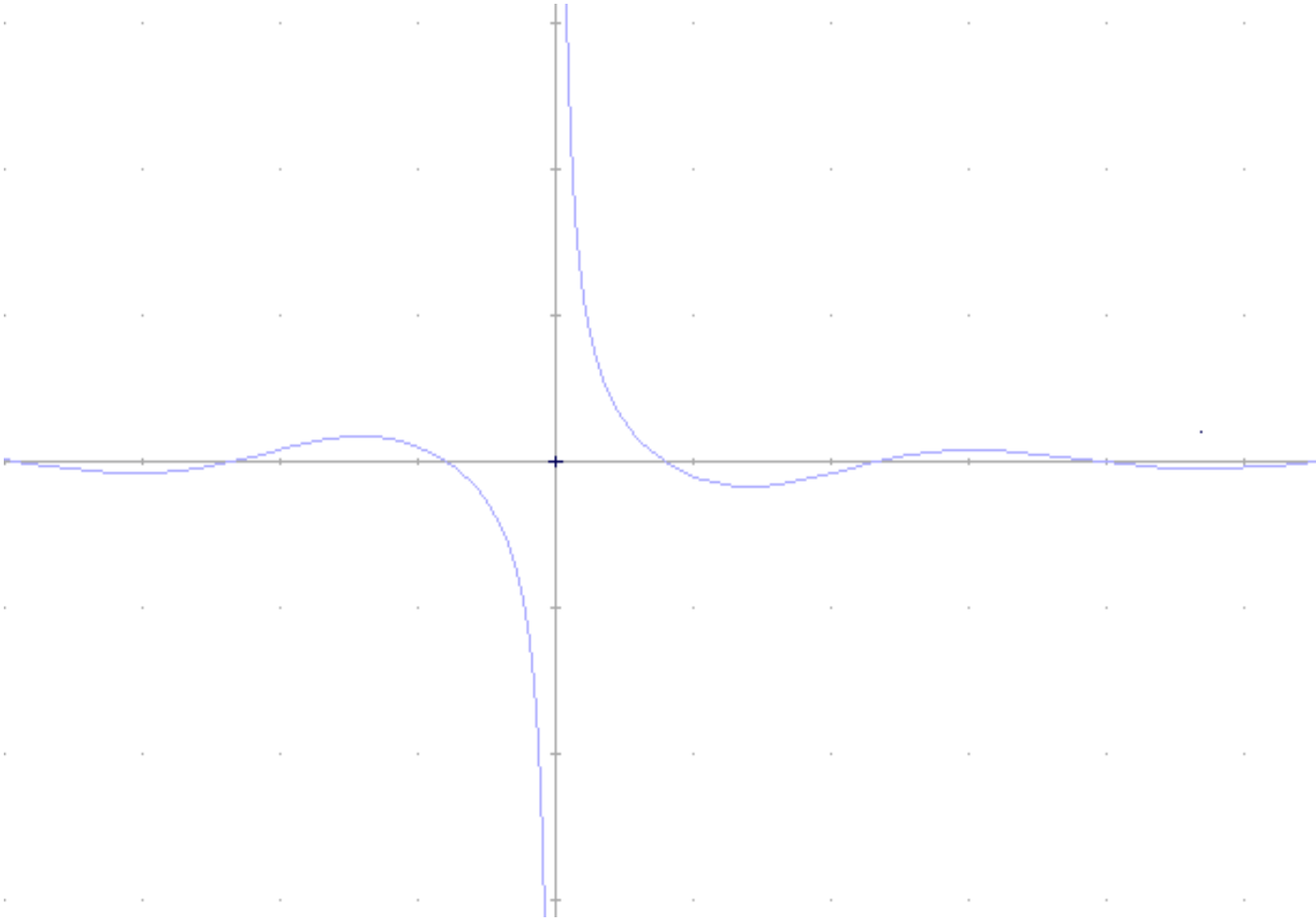
```
write ('Dame el valor de a: ');

readln (a);

write ('Dame el valor del error:');

readln(error);

write ('Dame el número de iteraciones:');

readln(i);

k:=1;

b:=a−(f(a)/f_prima(a));

while (abs(a−b)>error) and (i>k) do

begin

a:=b;

b:=a−(f(a)/f_prima(a));

k:=k+1;

end;

if k>i then writeln ('Lo siento pelanas, pero no converge.')

else writeln('La raiz del polinomio es ',b)

end.
```

Método de la Bisección:

a	b	error	raiz
4	7	10−6	4.7123882771
4	7	10−6	4.7123889803

Método de Newton:

a	error	iteraciones	raiz
4	10−6	25	4.7123889804
5	10−6	25	4.7123889804
6	10−6	25	−1.5707963268
7	10−6	25	7.853981634



Como se puede ver en la gráfica, el método de Newton, para valores iniciales cercanos, da saltos en la busqueda de raices, haciendo que no siempre nos dé como resultado la misma raiz. Esto se debe a que el corte de la derivada de la función con el eje x se separa mucho cuando esta se hace cerca a un punto mínimo o máximo de la misma.

Ejercicio 2.

La raiz obtenida con el programa que abajo se incluye, con un error de 10−6 y un número máximo de 25 iteraciones es **raiz=2.4142135382**.

program ej2pr2;	begin {*** Programa Principal ***}
uses wincrt;	solucion:=false;
var	repeat
i,k:integer;	write ('Dame el valor de a y b: ');
error:real;	readln (a,b);
a,b,c:real;	if a>b then cambia(a,b);
solucion:boolean;	until (f(a)*f(b))<0;

```
procedure cambia(var a,b:real);
var
aux:real;
begin
aux:=a;
a:=b;
b:=aux
end;
function f(x:real):real;
begin
f:=ln((x*x)/(2*x+1));
end;
function f_prima(x:real):real;
begin
f_prima:=((2*x*x+1)*(2*x+1))/(x*x);
end;
```

```
write ('Dame el valor del error:');
readln(error);
write ('Dame el número de iteraciones:');
readln(i);
while not(solucion) do
begin
c:=(a+b)/2;
solucion:=(abs(b−c)<error) or (f(c)=0);
if not solucion then
if f(b)*f(c)<0 then a:=c
else b:=c;
end;
solucion:=false;
k:=1;
a:=c;
b:=a−(f(a)/f_prima(a));
solucion:=abs(a−b)>error;
while not(solucion) and (i>k) do
begin
a:=b;
b:=a−(f(a)/f_prima(a));
k:=k+1;
solucion:=abs(a−b)>error;
end;
if k>i then writeln ('Lo siento pelanas, pero no
converge.')
else writeln('La raiz del polinomio es ',b);
```

```
end.
```

Ejercicio 4.

```
program ej3pr2;
{$N+}
uses crt,dos;
var
a,e,newt,stef:double;
function pot(x:double;n:integer):double;
begin
if n<>1 then pot:=pot(x,n−1)*x
else pot:=x;
end;
function f(x:double):double;
begin
f:=pot(x,6)−x−1;
end;
function fprima(x:double):double;
begin
fprima:=(6*pot(x,5))−1;
end;
function g(x:double):double;
begin
g:=pot(x,6)−1;
end;
function Newton(e,a:double):double;
const
```

```
function Steffen(e,a:double):double;
var
x:array [0..2] of double;
Ix:array [0..2] of double;
xN:array [0..2] of double;
R1:double;
aux:integer;
begin
writeln;
writeln('M todo de Steffensen.');
writeln;
x[0]:=a;
{ Metodo de Steffensen }
aux:=0;
repeat
aux:=aux+1;
x[1]:=g(x[0]);
x[2]:=g(x[1]);
Ix[0]:=x[1]−x[0];
Ix[1]:=x[2]−x[1];
R1:=Ix[1]/Ix[0];
xN[2]:=x[1]+Ix[1]/(1−R1);
x[0]:=xN[2];
writeln(g(xN[2]):10:8,' para xN[2]=',xN[2]:10:8);
```

```
nitera=50;

var

aux:integer;

b:double;

begin

writeln;

writeln('M todo de Newton.');

writeln;

aux:=1;

b:=a−(f(a)/fprima(a));

while ((abs(b−a)>e) and (nitera>aux)) do

begin

a:=b;

b:=a−(f(a)/fprima(a));

aux:=aux+1;

writeln('Iteraci¢n :',aux,' x=',b);

end;

writeln('En ',aux,' iteraciones, con el m todo de Newton.');

Newton:=b;

end;

until abs(xN[2]−x[2])<e;

writeln('En ',aux,' iteraciones, con el m todo de Stenffensen.');

Steffen:=xN[2];

end;

begin {*** Programa principal ***}

clrscr;

write('Dame el valor del error:');

readln(e);

{M todo de Newton}

write('Dame el valor de x:');

readln(a);

newt:=Newton(e,a);

stef:=Steffen(e,a);

writeln;

writeln('Raiz por Newton =',newt);

writeln('Raiz por Steffensen =',stef);

writeln;

repeat until keypressed;

end.
```

La comparación de los algoritmos de Steffensen y Newton, empleando en ambos un error de 10−8, ha dado como resultado:

Valor de x	**Método de Newton.**	**Método de Steffensen.**
1	6 iteraciones	6 iteraciones
0.7	34 iteraciones	8 iteraciones
0.6	11 iteraciones	8 iteraciones

Ejercicio 4

El programa implementado queda como sigue:

```
program ej4p2;
uses crt;
const
{grado_pol=5; { Grado del polinomio+1. }
grado_factor=3; { Grado del factor+1. }
n=10;
type
tipo_solucion=record
sol:real;
imag:boolean;
end;
vector=array[1..n] of real;
vector_solucion=array [1..n] of tipo_solucion;
var
grado_pol:word;
xp,p,q2:vector;
coc,res:vector;
aux:vector;
solucion:vector_solucion;
r,s,Ar,As:real;
i:integer;
b1,b0,Yb1,Yb0,Zb1,Zb0:real;
error:real;
comun:real;
cont_sol:integer;
```

```
salir:boolean;

num_iter:integer;

c:char;

{−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
}

{ PROCEDIMIENTO div_pol }

{−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
}

{ }

{ Este procedimiento divide dos polinommios. }

{ }

{ PARAMETROS DE ENTRADA: }

{ }

{ coc: Vector donde se devolver  el cociente de la divisi¢n. }

{ res: Vector donde se devolver  el resto de la divisi¢n. }

{ p: Vector donde se halla el dividendo. }

{ q2: Vector donde se halla el divisor. }

{ grado_pol: Grado del dividendo. }

{ grado_factor: Grado del divisor. }

{ }

{−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
}

procedure div_pol (var coc,res:vector; p,q2:vector;

grado_pol,grado_factor:integer);

var

j,k:integer;

begin

{ Se inicializan el vector resto y el vector cociente. }
```

```
for j:=1 to grado_pol do
begin
res[j]:=p[j];
coc[j]:=0
end;
{ A partir de aqu¡ se realiza la divisi¢n. }
for k:=(grado_pol−grado_factor) downto 0 do
{ grado_pol−grado_factor es el grado que tendr  el vector cociente. }
begin
{ Como los vectores est n definidos de 1..n, entonces el elemento del
vector que en ese momento calculamos ser  siempre uno m s del que se
nos da k. }
coc[k+1]:=res[grado_factor+k]/q2[grado_factor];
{ Ahora se realiza la resta entre el cociente parcial*divisor y los
elementos del polinomio dividendo correspondientes. }
for j:=(grado_factor+k−1) downto k+1 do
res[j]:=res[j]−coc[k+1]*q2[j−k]
end;
res[grado_factor]:=0
end;
procedure montar_factor (r,s:real; var q2:vector);
begin
q2[3]:=1; q2[2]:=r; q2[1]:=s
end;
procedure pto8;
begin
```

```
end;

procedure pto7;

begin

{ Pto 7 }

div_pol (coc,res,p,q2,grado_pol,grado_factor);

for i:=grado_pol−grado_factor+1 downto 1 do aux[i]:=coc[i];

b1:=res[2];

b0:=res[1];

end;

procedure pto6;

begin

{ Pto 6 }

r:=r+Ar;

s:=s+As;

montar_factor (r,s,q2);

pto7;

end;

procedure pto5;

begin

{ Pto 5 }

comun:=(Yb1*Zb0)−(Zb1*Yb0);

Ar:=((−b1*Zb0)−(Zb1*(−b0)))/comun;

As:=((Yb1*(−b0))−(−b1*Yb0))/comun;

if num_iter=0 then writeln ('Incremento de r=',Ar,' Incremento de s=',As);

pto6;

end;
```

```
procedure pto4;
begin
{ Pto 4 }
div_pol (coc,res,aux,q2,(grado_pol−grado_factor+1),grado_factor);
Zb1:=−res[2]; { ëb1/ës }
Zb0:=−res[1]; { ëb0/ës }
if num_iter=0 then writeln ('ëb1/ës=',Zb1,' ëb0/ës=',Zb0);
pto5;
end;
procedure pto3;
begin
{ Pto 3 }
for i:=(grado_pol−grado_factor+1)+1 downto 2 do
xp[i]:=aux[i−1];
xp[1]:=0;
{ En xp tenemos xQn−2. }
div_pol (coc,res,xp,q2,(grado_pol−grado_factor+1)+1,grado_factor);
Yb1:=−res[2]; { ëb1/ër }
Yb0:=−res[1]; { ëb0/ër }
if num_iter=0 then writeln ('ëb1/ër=',Yb1,' ëb0/ër=',Yb0);
pto4;
end;
procedure pto2;
begin
{ Pto 2 }
div_pol (coc,res,p,q2,grado_pol,grado_factor);
```

```
{ En coc se devuelve Qn−2. Su grado es grado_pol−grado_factor. Para
representarlo en el ordenador hay que sumarle 1, por estar definido el
vector de 1..n }
{ Guardamos en aux Qn−2. }
for i:=grado_pol−grado_factor+1 downto 1 do aux[i]:=coc[i];
b1:=res[2];
b0:=res[1];
if num_iter=0 then
begin
for i:=grado_pol−grado_factor+1 downto 1 do
writeln ('Qn−2[',i,']=',aux[i]);
writeln ('b1=',b1,' b0=',b0);
end;
pto3;
end;
procedure calcular_raices (pol:vector;var solucion:vector_solucion;
var cont_sol:integer);
var
a,b,c:real;
discr:real;
begin
a:=pol[3];
b:=pol[2];
c:=pol[1];
discr:=sqr(b)−4*a*c;
if discr>0 then
```

```
begin
solucion[cont_sol].imag:=false;
solucion[cont_sol+1].imag:=false;
end
else
begin
{Raices complejas}
solucion[cont_sol].imag:=true;
solucion[cont_sol+1].imag:=true;
end;
solucion[cont_sol].sol:=(−b+sqrt (abs(discr)))/(2*a);
solucion[cont_sol+1].sol:=(−b+sqrt (abs(discr)))/(2*a);
cont_sol:=cont_sol+2;
end;
procedure pantalla;
var
c:char;
begin
writeln ('r=',r,' s=',s);
for i:=grado_pol−grado_factor+1 downto 1 do
writeln ('Qn−2[',i,']=',aux[i]);
writeln ('b1=',b1,' b0=',b0);
writeln ('ëb1/ër=',Yb1,' ëb0/ër=',Yb0);
writeln ('ëb1/ës=',Zb1,' ëb0/ës=',Zb0);
writeln ('Incremento de r=',Ar,' Incremento de s=',As);
writeln
('−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−');
```

```
c:=readkey;

writeln;

end;

begin

clrscr;

write ('Dame el error admitido: '); readln (error);

write ('Dame el grado del polinomio: '); readln (grado_pol);

grado_pol:=grado_pol+1;

for i:=grado_pol downto 1 do

begin

write ('P[',i−1,']='); readln (p[i]);

end;

num_iter:=0;

write ('Dame el coeficiente del t rmino de primer grado: ');

readln (r);

write ('Dame el t rmino independiente: ');

readln (s);

writeln
('−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−');

salir:=false;

writeln ('INICIO');

montar_factor (r,s,q2);

pto2;

writeln
('−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−');

c:=readkey;

cont_sol:=1;
```

```
repeat
num_iter:=num_iter+1;
writeln ('ITERACION ',num_iter);
pantalla;
if abs(b1)+abs(b0)<error then
begin
calcular_raices (q2,solucion,cont_sol);
if (grado_pol−grado_factor)<3 then
begin
calcular_raices (aux,solucion,cont_sol);
salir:=true
end
else
begin
for i:=1 to grado_pol do p[i]:=0;
for i:=1 to (grado_pol−grado_factor+1) do p[i]:=aux[i];
pto2
end;
end
else pto3;
until salir;
writeln;
writeln ('Hay un total de ',cont_sol−1,' soluciones, que son: ');
for i:=1 to cont_sol−1 do
begin
if not solucion[i].imag then
```

```
write (i,'.) ',solucion[i].sol:3:2,' ')
else
write (i,'.) ',solucion[i].sol:3:2,'I ');
end;
c:=readkey;
end.
```

Los resultados obtenidos por pantalla son:

```
Error admitido: 1.0000000000E−04
Polinomio:
P[4]=1.0 P[3]=−6.0 P[2]=12.0 P[1]=−19.0 P[0]=12.0
Coeficiente del término de primer grado: 0.0000000000E+00
Término independiente: 0.0000000000E+00
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
INICIO
Qn−2[3]= 1.0000000000E+00
Qn−2[2]=−6.0000000000E+00
Qn−2[1]= 1.2000000000E+01
b1=−1.9000000000E+01 b0= 1.2000000000E+01
b1/r=−1.2000000000E+01 b0/r= 0.0000000000E+00
b1/s= 6.0000000000E+00 b0/s=−1.2000000000E+01
Incremento de r=−1.0833333333E+00 Incremento de s= 1.0000000000E+00
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
ITERACION 1
r=−1.0833333333E+00 s= 1.0000000000E+00
Qn−2[3]= 1.0000000000E+00
Qn−2[2]=−4.9166666667E+00
```

Qn−2[1]= 5.6736111111E+00

b1=−7.9369212963E+00 b0= 6.3263888889E+00

b1/r=−1.2000000000E+01 b0/r= 0.0000000000E+00

b1/s= 6.0000000000E+00 b0/s=−1.2000000000E+01

Incremento de r=−1.0833333333E+00 Incremento de s= 1.0000000000E+00

−−

ITERACION 2

r=−1.8331268647E+00 s= 2.9686270866E+00

Qn−2[3]= 1.0000000000E+00

Qn−2[2]=−4.1668731353E+00

Qn−2[1]= 1.3929658273E+00

b1=−4.0766244643E+00 b0= 7.8648039143E+00

b1/r=−5.2083333333E−01 b0/r=−3.8333333333E+00

b1/s= 3.8333333333E+00 b0/s=−4.6736111111E+00

Incremento de r=−7.4979353134E−01 Incremento de s= 1.9686270866E+00

−−

ITERACION 3

r=−8.5730329716E−01 s= 2.2677934645E+00

Qn−2[3]= 1.0000000000E+00

Qn−2[2]=−5.1426967028E+00

Qn−2[1]= 5.3233556958E+00

b1=−2.7736956372E+00 b0=−7.2271256387E−02

b1/r= 5.8537142434E+00 b0/r=−6.9280223924E+00

b1/s= 2.3337462707E+00 b0/s= 1.5756612593E+00

Incremento de r= 9.7582356751E−01 Incremento de s=−7.0083362208E−01

−−

ITERACION 4

r=−1.0782639849E+00 s= 2.9469189730E+00

Qn−2[3]= 1.0000000000E+00

Qn−2[2]=−4.9217360151E+00

Qn−2[1]= 3.7461503387E+00

b1=−4.5670376472E−01 b0= 9.6039849124E−01

b1/r= 6.1831966506E−01 b0/r=−9.7183871584E+00

b1/s= 4.2853934057E+00 b0/s=−3.0555622313E+00

Incremento de r=−2.2096068775E−01 Incremento de s= 6.7912550847E−01

−−−

ITERACION 5

r=−9.9685637916E−01 s= 2.9948942331E+00

Qn−2[3]= 1.0000000000E+00

Qn−2[2]=−5.0031436208E+00

Qn−2[1]= 4.0176901326E+00

b1=−1.1053984206E−02 b0=−3.2557008599E−02

b1/r= 3.3450461015E+00 b0/r=−1.1326400648E+01

b1/s= 3.8434720302E+00 b0/s=−7.9923136564E−01

Incremento de r= 8.1407605755E−02 Incremento de s= 4.7975260132E−02

−−−

ITERACION 6

r=−1.0000039987E+00 s= 2.9999875369E+00

Qn−2[3]= 1.0000000000E+00

Qn−2[2]=−4.9999960013E+00

Qn−2[1]= 3.9999964683E+00

b1=−6.1848346377E−05 b0= 6.0447302531E−05

b1/r= 2.9708970942E+00 b0/r=−1.1998406556E+01

b1/s= 4.0062872417E+00 b0/s=−1.0227958994E+00

Incremento de r=−3.1476195305E−03 Incremento de s= 5.0933037728E−03

−−−

ITERACION 7

r=−9.9999999999E−01 s= 3.0000000000E+00

Qn−2[3]= 1.0000000000E+00

Qn−2[2]=−5.0000000000E+00

Qn−2[1]= 4.0000000000E+00

b1= 5.8207660913E−11 b0=−1.4551915228E−10

b1/r= 2.9999990660E+00 b0/r=−1.1999926156E+01

b1/s= 3.9999920026E+00 b0/s=−1.0000089314E+00

Incremento de r= 3.9986999158E−06 Incremento de s= 1.2463087509E−05

−−−

Hay un total de 4 soluciones que son:

2.6i (doble)

4 (doble)