1 Programa, mediante un bloque, un método de Newton-Raphson en el que las entradas sean la función, la variable, el punto de partida, un error absoluto y un error relativo. Pon criterios de parada que eviten bucles infinitos si no hay convergencia.

```
nr(expr,var,ini,err):=block(
   [x0:ini, x1, dfx0, j, tol:10^(−10)],
   numer: true,
   local(f,df),
   define(f(x),subst(x,var,expr)),
   define(df(x),diff(f(x),x)),

   if sign(f(x0))=sign(f(x1)) then error("la función no cambia de signo en los extremos"),

   for i:1 thru 25 do(
      j:i,
      dfx0:df(x0),
      if abs(df(x0))<err then
         error("escoja otro valor inicial"),
      x1:x0−f(x0)/dfx0,
      if abs(x0−x1)<tol then return(),
      x0:x1
   ),
   if j=25 then error("escoja otro valor inicial") else x1
)$
```

```
nr(3·x+5,x,−10,10^−5);
```

$$-1.666666666666667$$

```
solve([3·x+5], [x]);
```

rat: replaced 1.666666666666667 by 5/3 = 1.666666666666667

$$[x=-1.666666666666667]$$

2 Programa, mediante un bloque, un método de la secante en el que las entradas sean la función, la variable, los dos puntos de partida, un error absoluto y un error relativo. Pon criterios de parada que eviten bucles infinitos si no hay convergencia.

Función recta secante que pasa por (a,f(a)), (b,f(b)): (x-a)/(b-a) = (y-f(a))/(f(b)-f(a)), la cual se acaba simplificando a x*f(b)-x*f(a)-a*f(b)+b*f(a))/(b-a). Antes de hacer la función vamos a desarrollar un ejemplo para comprobar que esto es correcto:

```
f(x):=x·cos(x)+3;
define(secante(x,a,b),(x·f(b)−x·f(a)−a·f(b)+b·f(a))/(b−a));
```

$$f(x) := x\cos(x)+3$$

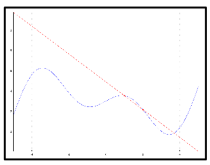
$$\text{secante}(x,a,b) := (b\cos(b)+3)\,x-(a\cos(a)+3)\,x-a(b\cos(b)+3)+(a\cos(a)+3)\,b/$$

Vamos a sacar la recta secante que une los puntos (1,f(1)) y (2,f(2)) y graficarlo todo:

```
secante(x,1,2);
```

$$4.912898284830565-1.372595978962424\,x$$

```
wxdraw2d(
  color= blue,
  explicit(f(x), x, −5, 5),
  color=red,
  explicit(4.912898284830565−1.372595978962424·x,x,−5,5),
  point_type=7,
  points([[1,f(1)],[2,f(2)]]),
  yaxis= true,
  xaxis= true,
  grid= true
);
```



Hemos obtenido correctamente la secante. Ahora definiremos el bloque y la probaremos entre 2 y 3.5 (para pillar un cambio de signo).

```
secante(expr,var,ini,fin,err):=block(
   [j:0, x0:ini, x1:fin, x2, fx0, fx1, tol:err],
   numer: true,
   ratprint: false,
   define(f(x),subst(x,var,expr)),
   define(secante_(x,a,b),(x·f(b)−x·f(a)−a·f(b)+b·f(a))/(b−a)),
   fx0:f(x0), fx1:f(x1),
   if sign(fx0)=sign(fx1) then error("Los puntos no tienen distinto signo"),
   while j < 15 and abs(x0−x1)>tol do(
      j:j+1,
      g(x):=secante_(x,x0,x1),
      sol:solve(g(x)=0,x),
      x2:rhs(part(sol,1)),
      x0:x1, x1:x2
   ),
   [x2]
)$

secante(x·cos(x)+3,x,2,3.5,10^−5);
```

[3.021693598847618]

```
find_root(f(x)=0,x,2,3.5);
```

3.021693598959826

3 *Modifica los bloques de los ejercicios anteriores para que en la salida, además de la solución aproximada, obtengas también el número de iteraciones necesarias para alcanzarla. Compara con distintas entradas para ver qué método es más rápido.*

```
nr_2(expr,var,ini):=block(
   [x0:ini, x1, dfx0, j, tol:10^(−10)],
   numer: true,
   local(f,df),
   define(f(x),subst(x,var,expr)),
   define(df(x),diff(f(x),x)),

   for i:1 thru 15 do(
      j:i,
      dfx0:df(x0),
      if abs(df(x0))<10^(−5) then
         error("escoja otro valor inicial"),
      x1:x0−f(x0)/dfx0,
      if abs(x0−x1)<tol then return(),
      x0:x1
   ),
   if j=15 then error("escoja otro valor inicial") else [x1,j]
)$

secante_2(expr,var,ini,fin,err):=block(
   [j:0, x0:ini, x1:fin, x2, fx0, fx1, tol:err],
   numer: true,
   ratprint: false,
   define(f(x),subst(x,var,expr)),
   define(secante_(x,a,b),(x·f(b)−x·f(a)−a·f(b)+b·f(a))/(b−a)),
   fx0:f(x0), fx1:f(x1),
   if sign(fx0)=sign(fx1) then error("Los puntos no tienen distinto signo"),
   while j < 15 and abs(x0−x1)>tol do(
      j:j+1,
      g(x):=secante_(x,x0,x1),
      sol:solve(g(x)=0,x),
      x2:rhs(part(sol,1)),
      x0:x1, x1:x2
   ),
   [x2,j]
)$
```

4 *Encuentra, usando el método de Newton-Raphson, una solución aproximada de la ecuación 2*%e^(x²+x−1)−%e^x²=2 en el intervalo [−0.3,1] con una exactitud menor que 10−6.*

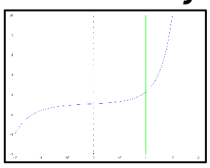
```
expr(x):=2·%e^(x^2+x−1)−%e^(x^2)−2;
```

$$\text{expr}(x) := 2\ \%e^{x^2+x-1} - \%e^{x^2} - 2$$

Vamos primero a graficarla:

```
wxdraw2d(
   color= blue,
   explicit(expr(x), x, −1.5, 1.5),
   color=green,
   implicit(x=−0.3+y·0,x,−1,1,y,−15,20),
   implicit(x=1+y·0,x,−1,2,y,−15,20),
   yaxis= true,
   xaxis= true,
   grid= true
);
```

rat: replaced 0.3 by 3/10 = 0.3



Para el método de Newton Raphson lo más conveniente es escoger puntos cuya derivada no sea muy cercana a cero. Como vemos, entre el punto -0.3 y 1 (marcados en verde) estamos en un punto de silla. El punto más propicio es el que más pendiente tenga, en este caso, el 1 aparentemente.

```
find_root(2·%e^(x^2+x−1)−%e^(x^2)−2, x, −0.3, 1);
```

0.9234093449813873

```
nr(2·%e^(x^2+x−1)−%e^(x^2)−2,x,0,10^(−6));
```

0.9234093449813873

Para poder calcularlo en este rango con un error absoluto menor a 10^-6 ha sido necesario aumentar el número de iteraciones (en el guión se sugieren 15, yo he aumentado arbitrariamente a 25).