Metodo_de_Newton

April 7, 2022

0.1 Método de Newton-Raphson

Nota: El demo está extraido de: https://computationalthinking.mit.edu/Spring21/newton_method/ pero pasado a Jupyter.

Recodemos que el método de Newton-Raphson se utiliza para encontrar raices de funciones (suficientemente suaves), es decir los valores $\{x^j\}$ donde $f(x^j) = 0$. El método requiere de un valor inicial x_0 y a partir del mismo define una sucesión de puntos $\{x_i^j\}$ que convergen a la raíz x^j. *De ahora en más nos olvidaremos de las otras raices y pensaremos en una sola, a la que llamaremos x y así no utilizaremos más el supra- indice j.*

El método de Newton-Raphson en realidad tiene una larga historia, *hay una buen síntesis en Wikipedia*, que comienza con algoritmos para calcular las raices cuandradas de números en la cultura Babilónica. La forma presente del mismo sin embargo es bastante moderna, 1740 y es debida a Thomas Simpson.

Encontrar las raices de una función es mucho más importante de lo que en principio parece ya que tiene muchas aplicaciones que van mucho más allá de lo que veremos aquí y que este método permite. Por ejemplo:

Ejemplo 1: Si tenemos una función suficientemente suave, $f(x) : R \to R$ podemos definir su inversa (generalmente sólo válida en una región), $g(y) : U \subset R \to V \subset R$, satisfaciendo $g(f(x)) = x \ \ \forall x \ \in V$.

Para ello, para cada $y \in U$ encontramos $\hat{x} \in V$ tal que $f(\hat{x}) - y = 0$.

Luego $g(y) := \hat{x}$ y por lo tanto $g(f(\hat{x})) = g(y) = \hat{x}$

Ejemplo 2: Si tenemos una función $F : R^n \to R^n$ también podemos encontrar los ceros. Y por lo tanto funciones inversas en dimensión arbitraria.

Ejemplo 3: Si tenemos funciones de funciones, también podemos encontrar los ceros. En este caso estamos resolviendo casi cualquier problema no lineal de la física!

$$\Delta\phi - \phi^3 = \rho \qquad \to \qquad \phi(\rho)$$

Es muy probable que utilicen este método en su vida profesional!

[23]:
```
using ForwardDiff, Plots, LaTeXStrings
```

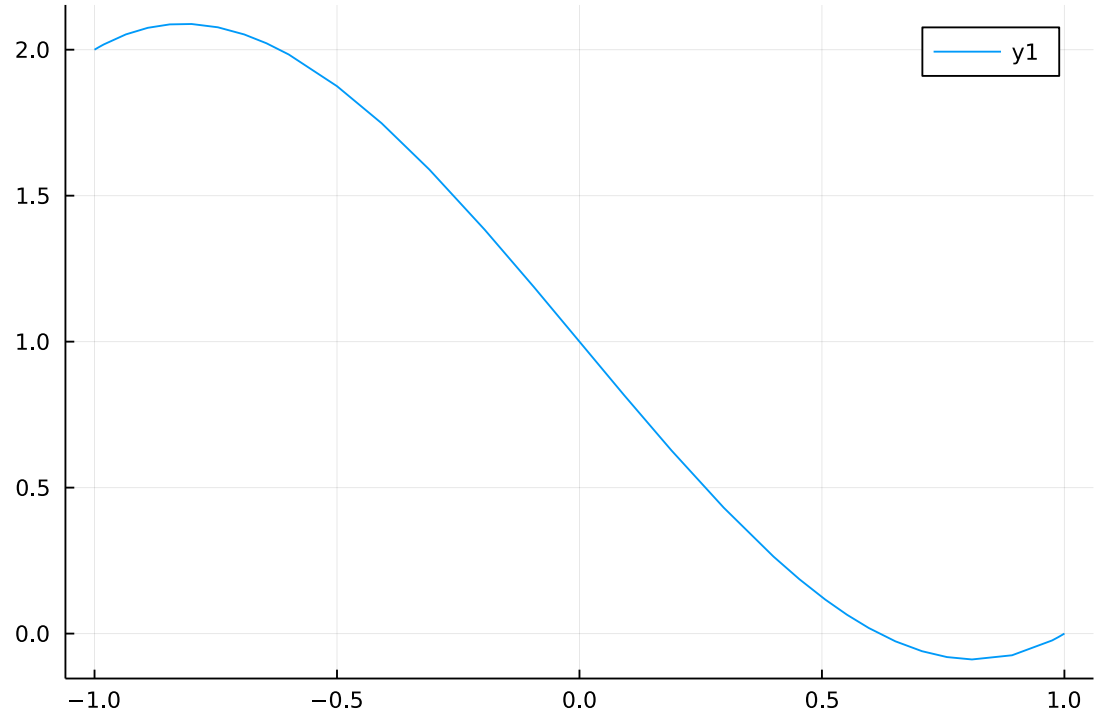
0.1.1 Versión Gráfica:

Vamos a hacer una versión gráfica del algoritmo de Newton-Raphson para ver como este converge a alguna de las raices de una función. Pero primero repasemos, a modo de justificación el método de la secante:

Gráfico Secante:

[24]:
```
f(x) = x^3 -2x + 1
plot(f,-1,1)
```

[24]:



[25]:
```
straight(x0, y0, x, m) = m*(x-x0) + y0
```

[25]:
```
straight (generic function with 1 method)
```

[26]:
```
function standard_Newton(f, #función a encontrar la raíz
                         n, # número de iteracones
                         x_range, # rango para graficar
                         x0, # valor inicial
                         ymin=-10, ymax=10 #límites de integración
                        )

    # aquí usamos una librería para calcular la derivada automáticamente
    f  = x -> ForwardDiff.derivative(f, x)
```

```
    #ploteamos la función
        p = plot(f, x_range, lw=3, ylim=(ymin, ymax), legend=:false)
    # el punto inicial
        scatter!([x0], [0], c="green", ann=(x0, -3, L"x_0", 10))
    # la línea del cero
        hline!([0.0], c="magenta", lw=3, ls=:dash)
    #comenzamos las iteraciones:
        for i in 1:n

                plot!([x0, x0], [0, f(x0)], c=:gray, alpha=0.5)
                scatter!([x0], [f(x0)], c=:red)
        #calculamos la tangente
                m = f (x0)
        # graficamos la recta tangente
                plot!(x_range, [straight(x0, f(x0), x, m) for x in x_range], c=:
↪blue, alpha=0.5, ls=:dash, lw=2)
        #iteramos el paso (o sea encontramos el punto donde la recta corta el␣
↪eje)
                x1 = x0 - f(x0) / m
                #scatter!([x1], [0], c="green", ann=(x1, -3, "x$i"))

                if i <= n
                        scatter!([x1], [0], c="green", ann=(x1, -3, L"x_%$i",␣
↪10))
                end

                x0 = x1

        end

        #p |> as_svg
    display(p)

end
```

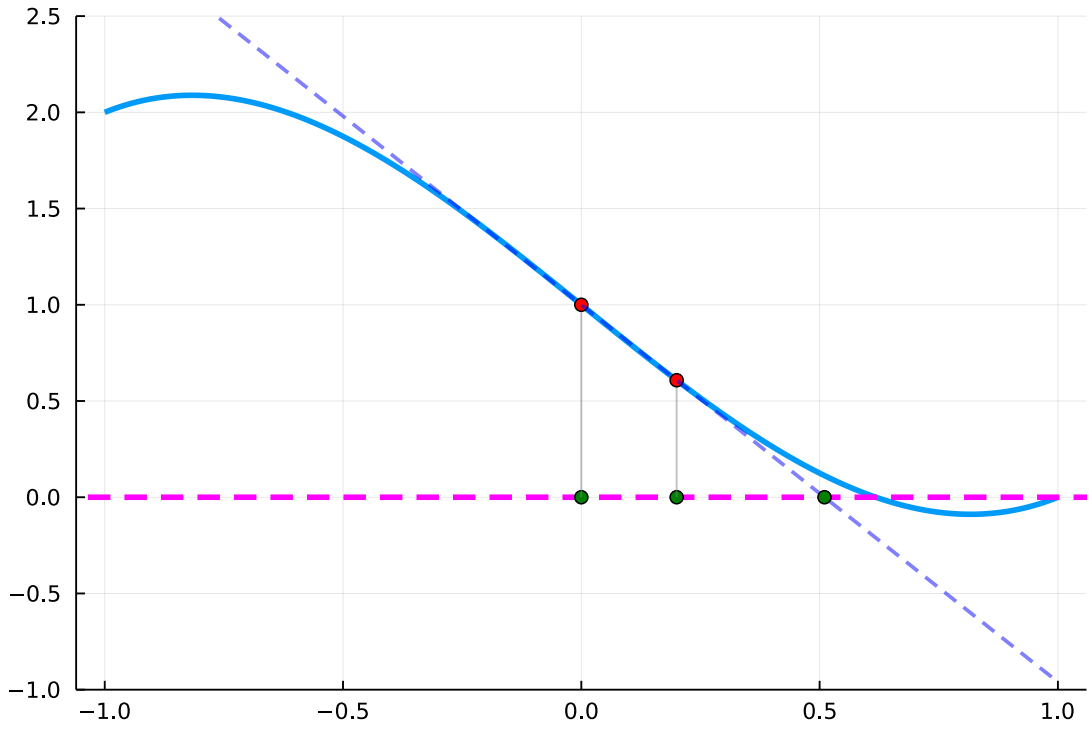
[26]: standard_Newton (generic function with 3 methods)

[27]:
```
function secante(f, n, x_range, x0, x1, ymin=-10, ymax=10)
    l = -2
        p = plot(f, x_range, lw=3, ylim=(ymin, ymax), legend=:false)

    hline!([0.0], c="magenta", lw=3, ls=:dash)

        scatter!([x0], [0], c="green", ann=(x0, l, L"x_0", 10))
        scatter!([x1], [0], c="green", ann=(x1, l, L"x_1", 10))

        for i in 1:n
```

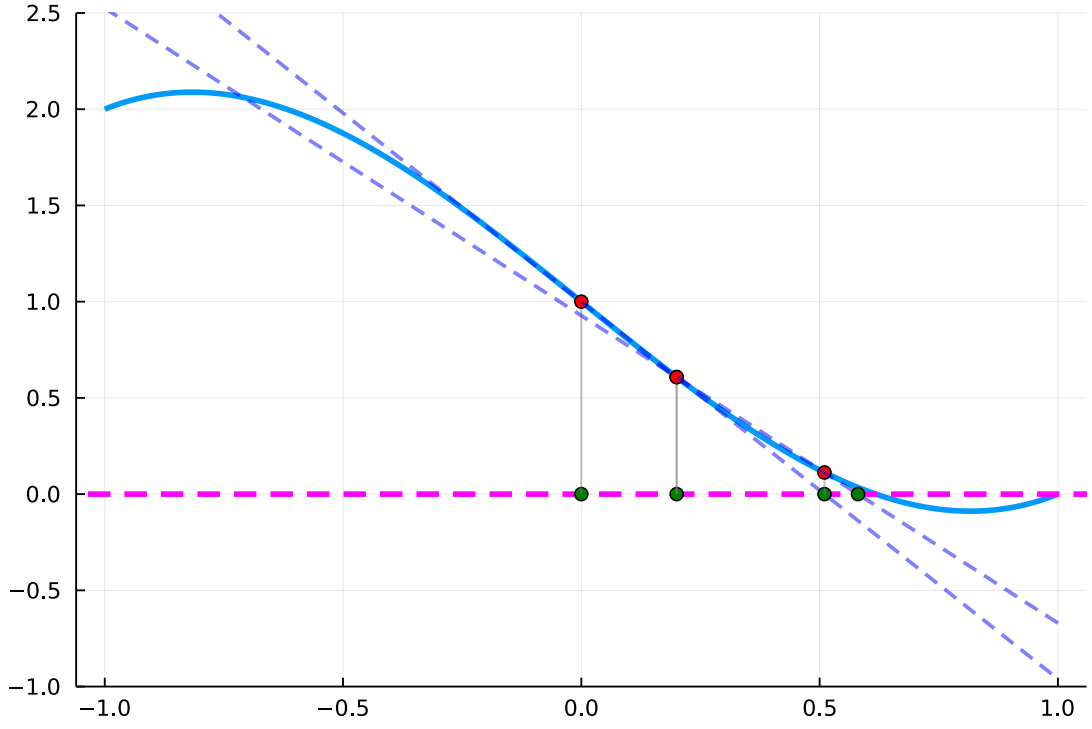
```
                plot!([x0, x0], [0, f(x0)], c=:gray, alpha=0.5)
        plot!([x1, x1], [0, f(x1)], c=:gray, alpha=0.5)
                scatter!([x0], [f(x0)], c=:red)
        scatter!([x1], [f(x1)], c=:red)

                m = (f(x1)-f(x0))/(x1-x0)

                plot!(x_range, [straight(x0, f(x0), x, m) for x in x_range], c=:
↪blue, alpha=0.5, ls=:dash, lw=2)

                x2 = x0 - f(x0) / m
                #scatter!([x1], [0], c="green", ann=(x1, -3, "x$i"))

                if i <= n
                        scatter!([x2], [0], c="green", ann=(x2, 1,␣
↪L"x_%$(i+1)", 10))
                end

                x0 = x1
        x1 = x2


        end

        #p |> as_svg
    display(p)

end
```

[27]:
```
secante (generic function with 3 methods)
```

[28]:
```
g(x) = x^3 - 2x + 1
x0 = 0.
x1 = 0.2
secante(g, 1, -1:0.01:1, x0, x1, -1, 2.5)
```



[29]:
```
secante(g, 2, -1:0.01:1, x0, x1, -1, 2.5)
```



[30]:
```
secante(g, 4, 0:0.01:1, x0, x1, -0.1, 0.2)
```

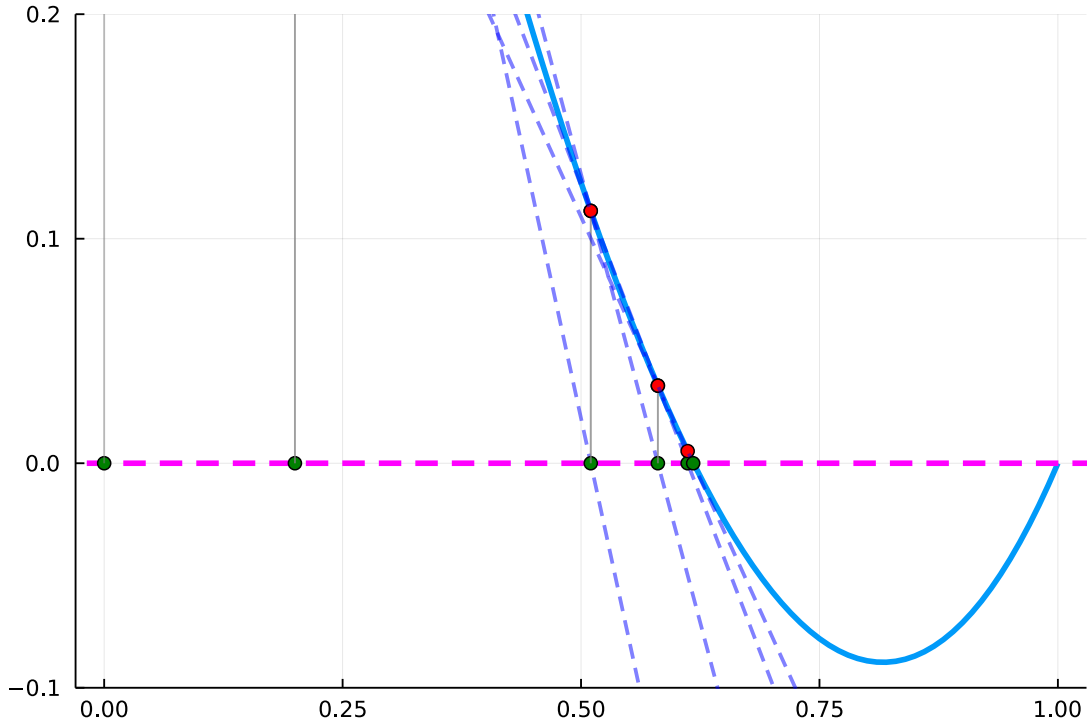
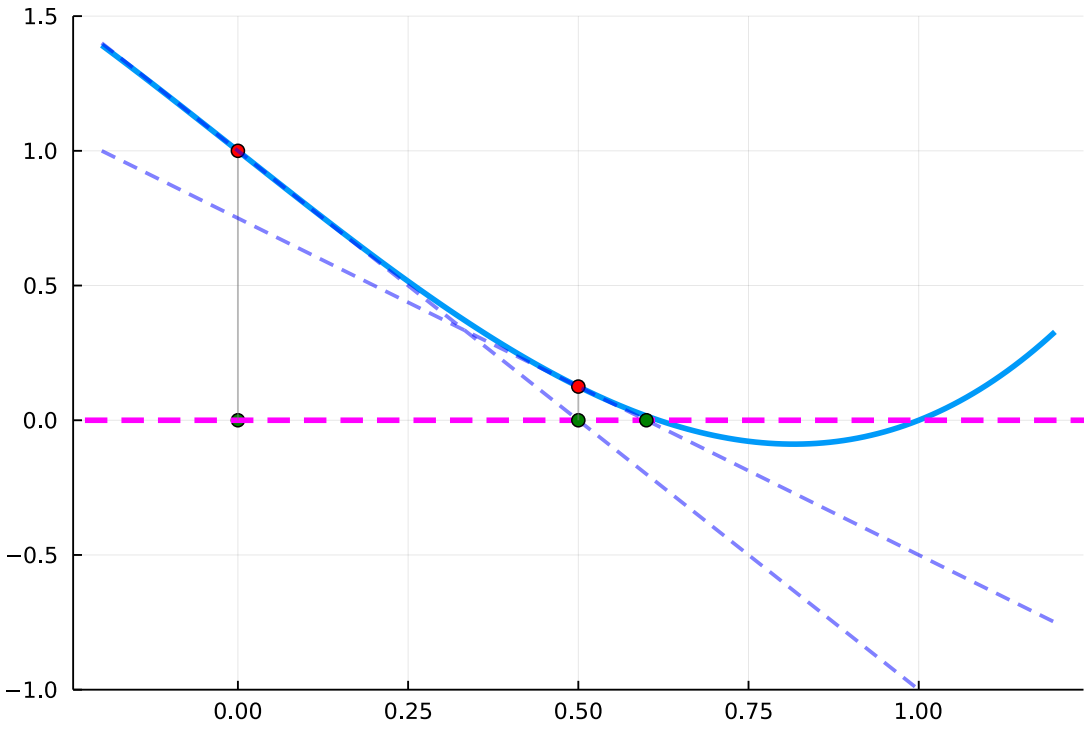
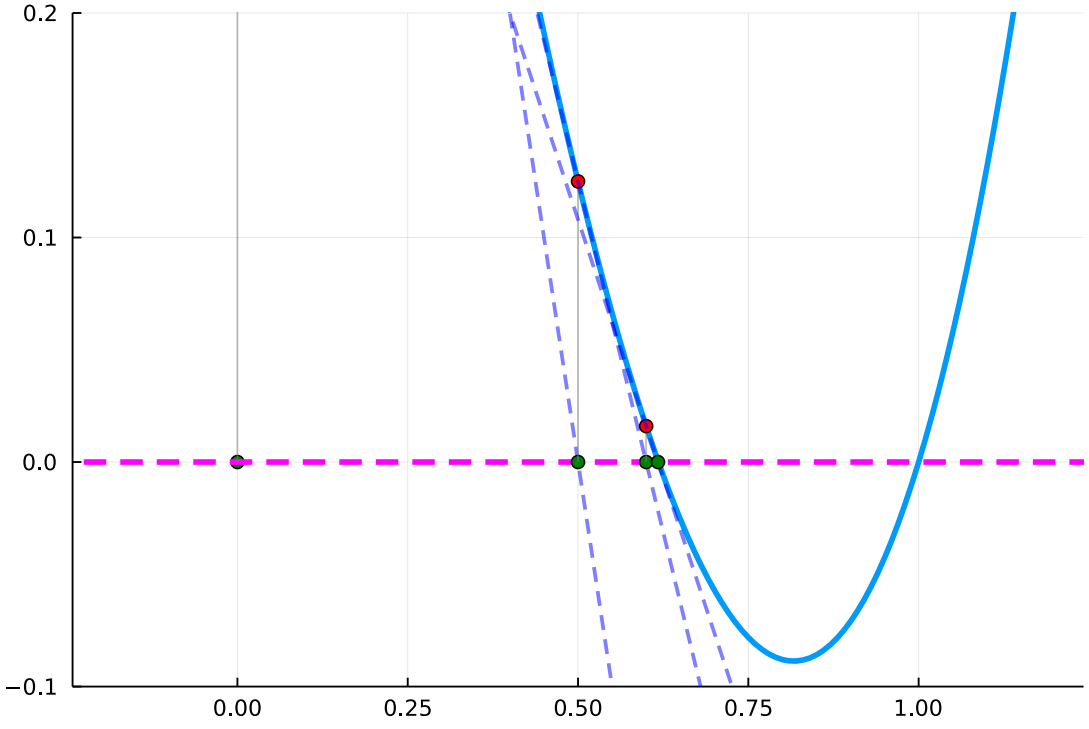


Gráfico Newton-Raphson:

[31]:
```
#f(x) = 2x^3 - 2x + 1 #(con un polo doble)
n = 2
x0 = 0.
standard_Newton(f, n, -0.2:0.01:1.2, x0, -1, 1.5)
```



[32]:
```
standard_Newton(f, 3, -0.2:0.01:1.2, x0, -0.1, 0.2)
```



0.1.2 Deducción del algoritmo:

Dado un valor x_i veamos como encontramos el próximo elemento de la sucesión:

Del gráfico vemos que vamos desde x_i a un punto en el gráfico, $(x_i, f(x_i))$, y desde allí trasamos una recta que tiene la misma pendiente (tangente) que la curva en ese punto. La intersección de esa recta con la recta $y = 0$, $(x_{i+1}, 0)$ nos dará el siguiente punto en la iteración.

La recta viene dada por:

$$y = f'(x_i) * (x - x_i) + f(x_i) \qquad f'(x_i) = \frac{df}{dx}|_{x=x_i}$$

Efectivamente vemos que es una recta ya que es una función lineal de x y para $x = x_i$ pasa por el punto $(x_i, f(x_i))$. La recta toca la abscisa $(y = 0)$ cuando

$$f'(x_i) * (x - x_i) + f(x_i) = 0.$$

Por lo tanto el algoritmo definiendo la sucesión es:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}.$$

A primera vista pareciera que el algoritmo es más costoso en término de cálculos, en cada iteración debemos evaluar f y $\frac{1}{f'}$. Mientras que en los otros métodos sólo necesitabamos evaluar a f en cada iteración. **Pero este método converge mucho más rápido!**

0.1.3 Ejemplo: Raíz de 2

Para aproximar la raíz de 2 resolveremos:

$$x^2 - 2 = 0$$

Es decir $f(x) = x^2 - 2$ y por lo tanto $f'(x) = 2x$.

El algoritmo es entonces:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^2 - 2}{2x_i} = \frac{x_i}{2} + \frac{1}{x_i}$$

Iniciando con $x_0 = 1$ obtenemos:

$$x_1 = \frac{1}{2} + 1 = \frac{3}{2} = 1.5 \qquad x_1^2 = 2.25 \tag{1}$$

$$x_2 = \frac{3}{4} + \frac{2}{3} = \frac{17}{12} = 1.4166666666666667 \qquad x_2^2 = 2.0069444444444446 \tag{2}$$

$$x_3 = \frac{17}{24} + \frac{12}{17} = \frac{577}{408} = 1.4142156862745099 \qquad x_3^2 = 2.000006007304883 \tag{3}$$

[33]:
```
Float64(17//24 + 12//17)^2
```

[33]:
```
2.000006007304883
```

Hagamos un algoritmo simple para calcular raices por el método de Newton-Raphson:

[34]:
```
function NR_S(f,df,x ,n_iter=10)
    x = x
    iter = 0
    while iter < n_iter
        iter = iter + 1
        x = x - f(x)/df(x)
    end
    return (f(x), x)
end
```

[34]:
```
NR_S (generic function with 2 methods)
```

Lo aplicamos al ejemplo anterior:

[35]:
```
fr2(x) = x^2 - 2
dfr2(x) = 2x
```

[35]:
```
dfr2 (generic function with 1 method)
```

[36]:
```
NR_S(fr2,dfr2,1,3)
```

[36]:
```
(6.007304882871267e-6, 1.4142156862745099)
```

Aquí les dejamos una versión más completa para que la investiguen y la usen para jugar con sus funciones favoritas.

[37]:
```
function NR(f,df,x ,tol =1.e-7,tol_f=1.e-7,max_iter=100)#x ::tipo, tol ::tipo=1.
 ↪e-7,tol_f::tipo=1.e-7,max_iter::Int64=100)
    x = x
    iter = 0
    Er = zeros(max_iter)
    dx = 1.
    while (abs(dx/x) > tol ) && (abs(f(x)) > tol_f) && iter < max_iter
        iter = iter + 1
```

```
        dx = f(x)/df(x)
        x = x - dx
        Er[iter] = abs(dx/x)
    end
    return (f(x), x, Er)
end
```

[37]: NR (generic function with 4 methods)

0.2 Tarea:

1. Entienda en detalle la función anterior.

0.2.1 Defina varias funciones y juegue con el método.

1. Cambie el número de iteraciones y el punto inicial.
2. Pruebe con raices simples y vea desde que puntos el algoritmo convege a cada una. Esa regiones se llaman las bases de atracción de la raíz de este método.
3. Vea que sucede si pone una raíz doble o de orden mayor.

Ejemplo: calcule la inversa de una función. Nota: aquí deberemos utilizar funciones con inversas globales (por ejemplo monótonas crecientes). De lo contrario el problema se complicará mucho.

[38]:
```
"""
inversa(f, df, y, x0=1.0)
Calcula la inversa de la función f(x) : R \to R.
Debemos dar la función, su derivada, el punto donde la queremos evaluar
y un valor semilla para el valor de la inversa.
"""
function inversa(f, df, y, x0=1.0)
    g(x) = y - f(x)
    dg(x) = -df(x)
    (Ef, x, Er) = NR(g,dg,x0)
    return x
end
```

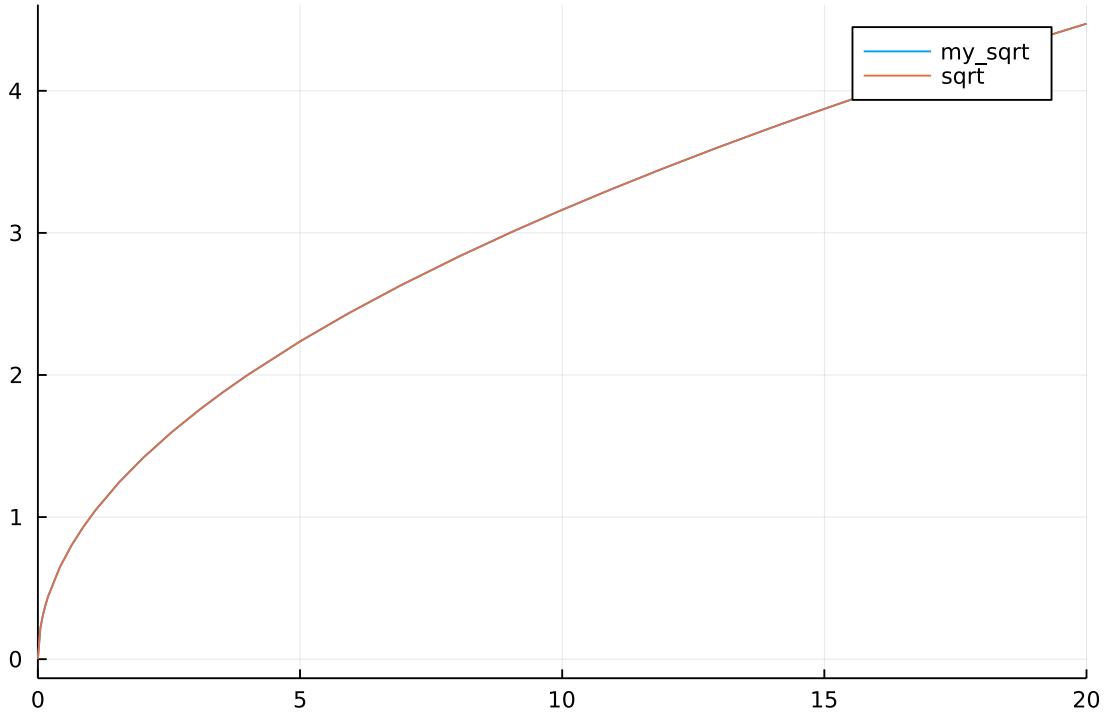
[38]: inversa

[39]:
```
fx2(x) = x^2
dfx2(x) = 2x
my_sqrt(x) = inversa(fx2,dfx2, x)

s = 2
my_sqrt(s) - sqrt(s)
```

[39]: 1.5947243525715749e-12

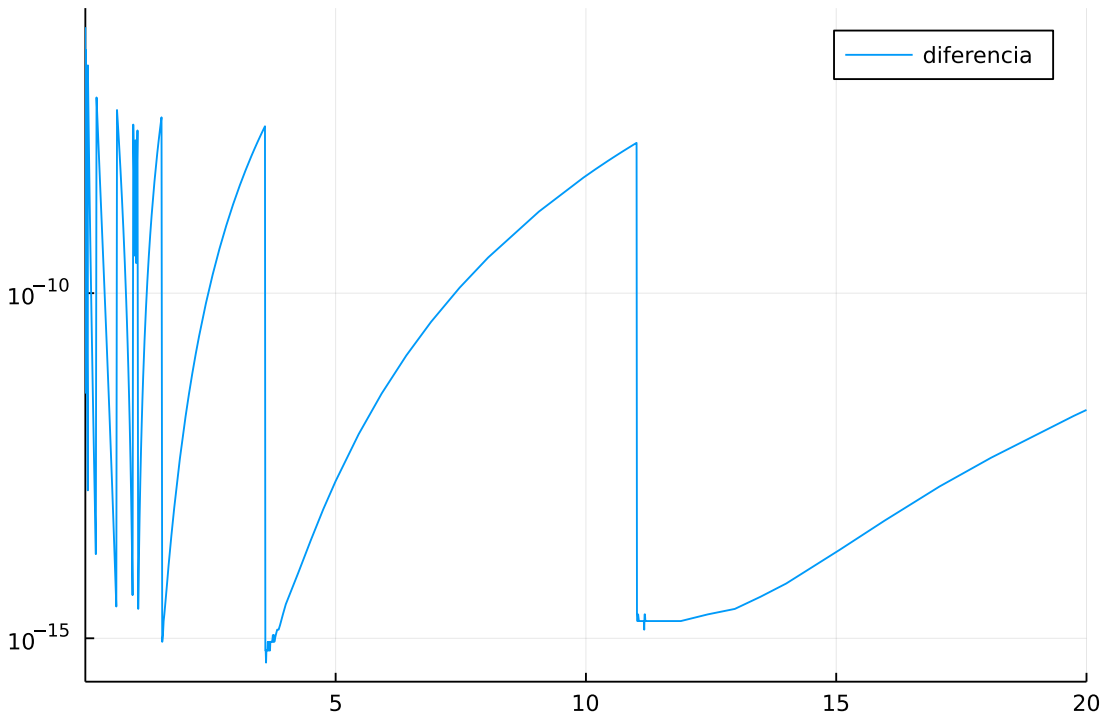
[40]:
```
plot(my_sqrt, xlim=(0,20), label="my_sqrt")
plot!(sqrt, label="sqrt")
```

[40]:



[41]:
```
plot(x -> abs(my_sqrt(x)-sqrt(x) + eps(x)), xlim=(0.0001,20), label="diferencia"
    , yscale=:log10
    )
```

[41]:



0.2.2 Convergencia

Analizaremos el porqué de la convergencia tan rápida del método.

Teorema de convergencia del método de Newton-Raphson:

Sea $f(x)$ y $[a,b]$ un intervalo tal que:

1. *$f(x)$ está definida en dicho intervalo y es dos veces contínuamente diferenciable en el mismo. (Derivada segunda existe y es contínua). $f \in C^2[a,b]$.*
2. *Existe un cero de f en dicho intervalo, es decir, $\exists p \in [a,b]$ t.q. $f(p) = 0$.*
3. *En dicho cero la derivada de f no se anula, es decir, $f'(p) \neq 0$*

Entonces existe $\delta > 0$ tal que si $x_0 \in [p-\delta, p+\delta]$ luego la sucesión generada por el método de Newton-Raphson con valor inicial x_0, $\{x_i\}$ converge a p. (es decir dado $\epsilon > 0$ arbitrario, existe n tal que $|x_i - p| < \epsilon \ \ \forall\ i > n$)

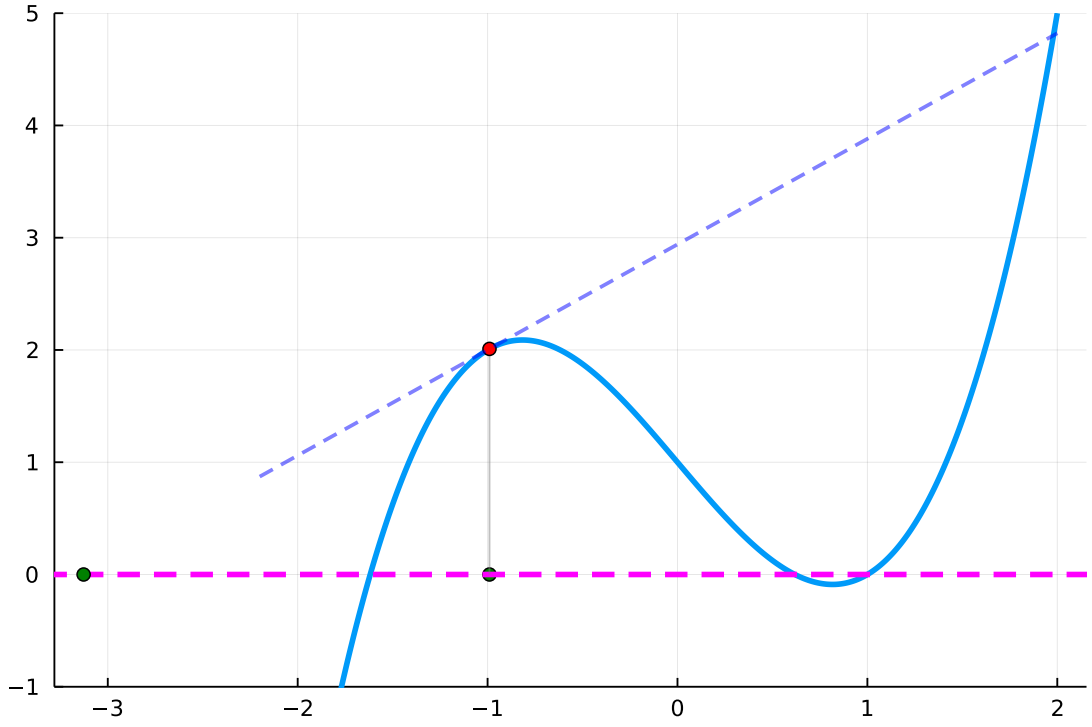
Notas:

1. Hay dos intervalos, el intervalo *grande* donde sabemos que está el cero, $[a,b]$ y un intervalo *menor* $[p-\delta, p+\delta]$ que será un intervalo que eligiremos de forma tal que la derivada de f no sea demasiado pequeña. Ese intervalo existe pues hemos supuesto que $f(p) \neq 0$ y por lo tanto, por continuidad de esa función existe todo un intervalo alrededor de p (de tamaño δ) donde estará acotada por debajo, $|f'(x)| > c\ \forall x \in [p-\delta, p+\delta]$

2. El intervalo $[p-\delta, p+\delta]$ también nos servirá para seleccionar un valor inicial que nos asegurará convergencia, si elegimos un valor demasiado alejado de p puede suceder que nuestra iteración no converja, o lo haga a otra raíz. Ver gráfico más abajo.

3. Como hemos supuesto que $f''(x)$ es contínua en el intervalo $[a, b]$ existirá una constante M tal que $|f''(x)| < M \ \forall x \in [a, b]$.

[42]:
```
n = 2
x0 = -0.99
standard_Newton(f, 1, -2.2:0.01:2, x0, -1, 5)
```



Prueba: Comenzamos notando que del teorema del valor medio tenemos que si p es una raíz:

$$-f(x_i) = f(p) - f(x_i) = f'(x_i)(p - x_i) + \frac{f''(\zeta_i)}{2}(p - x_i)^2$$

donde $\zeta_i \in [p, x_i]$ es un valor que no conocemos.

Sumando de cada lado p en la fórmula de la iteración de Newton-Raphson obtenemos,

$$(p - x_{i+1}) = (p - x_i) + \frac{f(x_i)}{f'(x_i)} \tag{4}$$

$$= (p - x_i) - \frac{f'(x_i)(p - x_i) + \frac{f''(\zeta_i)}{2}(p - x_i)^2}{f'(x_i)} \tag{5}$$

$$= (p - x_i) - \frac{f'(x_i)(p - x_i)}{f'(x_i)} - \frac{f''(\zeta_i)(p - x_i)^2}{2f'(x_i)} \tag{6}$$

$$= -\frac{f''(\zeta_i)(p - x_i)^2}{2f'(x_i)} \tag{7}$$

Por lo tanto, definiendo el error cometido en el paso de la iteración como: $e_i = |p - x_i|$ obtenemos una *cota para el error:*

$$e_{i+1} \leq \frac{M}{2c} e_i^2$$

Donde recordemos que $M = \max_{x \in [a,b]} |f''(x)|$, y $c = min_{x \in [p-\delta, p+\delta]} |f'(x)|$.

Este es un caso de convergencia cuadrática. Veamos que significa. Primero podemos absorber las constantes redefiniendo e, Sea $\tilde{e} = \frac{M}{2c} e$, entonces la relación resulta:

$$\tilde{e}_{i+1} \leq \tilde{e}_i^2$$

La solución de esta relación es:

$$\tilde{e}_i = \tilde{e}_0^{2^i}$$

Si $\tilde{e}_0 < 1$ (estamos cerca de la solución) la sucesión converge muy rápido. Si $\tilde{e}_0 > 1$ la sucesión puede crece sin cota.

Por lo tanto, para completar la prueba del teorema solo resta elegir δ lo suficientemente chico para que $\tilde{e}_0 = \frac{M}{2c} e_0 < 1$, es decir, $e_0 < \delta < \frac{2c}{M}$.

Cuán rápido converge?

[43]:
```
e0 = 0.5
E = [e0^(2^i) for i in 1:7]
```

[43]:
```
7-element Vector{Float64}:
 0.25
 0.0625
 0.00390625
 1.52587890625e-5
 2.3283064365386963e-10
 5.421010862427522e-20
 2.938735877055719e-39
```

Vemos por lo tanto que típicamente 4 o 5 iteraciones de Newton-Raphson son suficientes para la mayoría de las aplicaciones prácticas. La convergencia cuandrática nos dice que en cada paso duplicamos la cantidad de cifras correctas!

Algunas consideraciones:

1. En la práctica no conocemos p sino sólo una región aproximada de su ubicación. Por lo tanto, aunque conozcamos δ, si p no conocemos x_0. En problemas complicados, multidimensionales, puede ser difícil encontrar un x_0 apropiado.
2. Si la derivada es muy chica o tiende a cero en la raíz el método puede dar problemas de precisión y convergencia. Incluso puede dar un Inf debido a una división por cero. Para estos casos hay métodos especiales.
3. Si la derivada no está acotada también tenemos problemas, el caso típico para ver esta situación de con funciones tales como: $f(x) = |x|^\alpha, \quad \alpha \in (0, 1/2]$ (El teorema no aplica por falta de diferenciabilidad en el cero.)

[ ]: