HW05 Fixed-point Iteration and Brent's Method

Problem 1 (5 points)

(Analytical) Use the theory of fixed-point iterations to determine under what conditions Newton's Method is exactly quadratically convergent.

Solution

We have exact quadratic convergence when $F'(\alpha) = 0, F''(\alpha) \neq 0$

$F(x) = x - \frac{f(x)}{f'(x)}$

$F'(x) = 1 - \frac{f'(x)^2 - f''(x)f(x)}{f'(x)^2}$

$= \frac{f'(x)^2 - f'(x)^2 + f''(x)f(x)}{f'(x)^2}$

$= \frac{f''(x)f(x)}{f'(x)^2}$

$f(\alpha) = 0 \Rightarrow F'(\alpha) = 0$

$F''(x) = \frac{(f''(x)f(x))'f'(x)^2 - 2f'(x)f''(x)(f''(x)f(x))}{f'(x)^4}$

$(f''(x)f(x))' = f'''(x)f(x) + f''(x)f'(x)$

$F''(x) = \frac{f(x)f'(x)^2 f'''(x) + f'(x)^3 f''(x) - 2f(x)f'(x)f''(x)^2}{f'(x)^4}$

$F''(\alpha) = \frac{0 \times f'(\alpha)^2 f'''(\alpha) + f'(\alpha)^3 f''(\alpha) - 2 \times 0 \times f'(\alpha)f''(\alpha)^2}{f'(\alpha)^4}$

$= \frac{f'(\alpha)^3 f''(\alpha)}{f'(\alpha)^4} = \frac{f''(\alpha)}{f'(\alpha)}$

Thus, for exact quadratic convergence we have $f'(\alpha) \neq 0, f''(\alpha) \neq 0$

Problem 2 (5 points)

(Julia) Use Brent's method in the `Roots.jl` package in Julia to find the roots of

1. $f(x) = x\cos(x)$

2. $f(x) = x^2 \ln(x)$
3. $f(x) = (x-1)^2$

Plot each curve to make sure your results are reasonable. If Brent's method fails, try another option. Give some thought to the accuracy.

Solution

In [1]:
```
using Roots
y = x-> x*cos.(x)
find_zero(y, (-(pi/2), (pi/2)), Roots.Brent())
```

Out[1]: 0.0

In [2]:
```
r = find_zero(y, (-3, -1), Roots.Brent())
#-pi/2
```

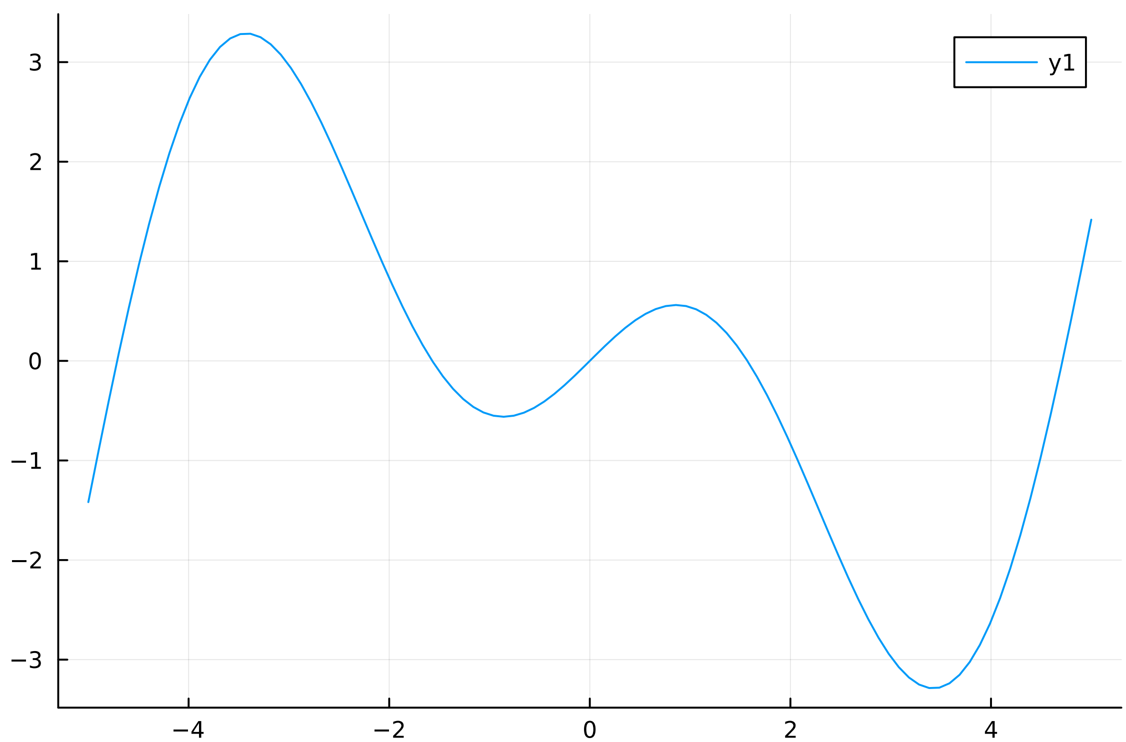
Out[2]: -1.5707963267948966

In [3]:
```
r = find_zero(y, (1, 3), Roots.Brent())
#pi/2
```

Out[3]: 1.5707963267948966

In [4]:
```
using Plots
x = range(-5, 5, length=100)
plot(x,y)
```

Out[4]:



$x\cos(x) = 0 \forall x = \frac{k\pi}{2}, k \in \mathbb{Z}$

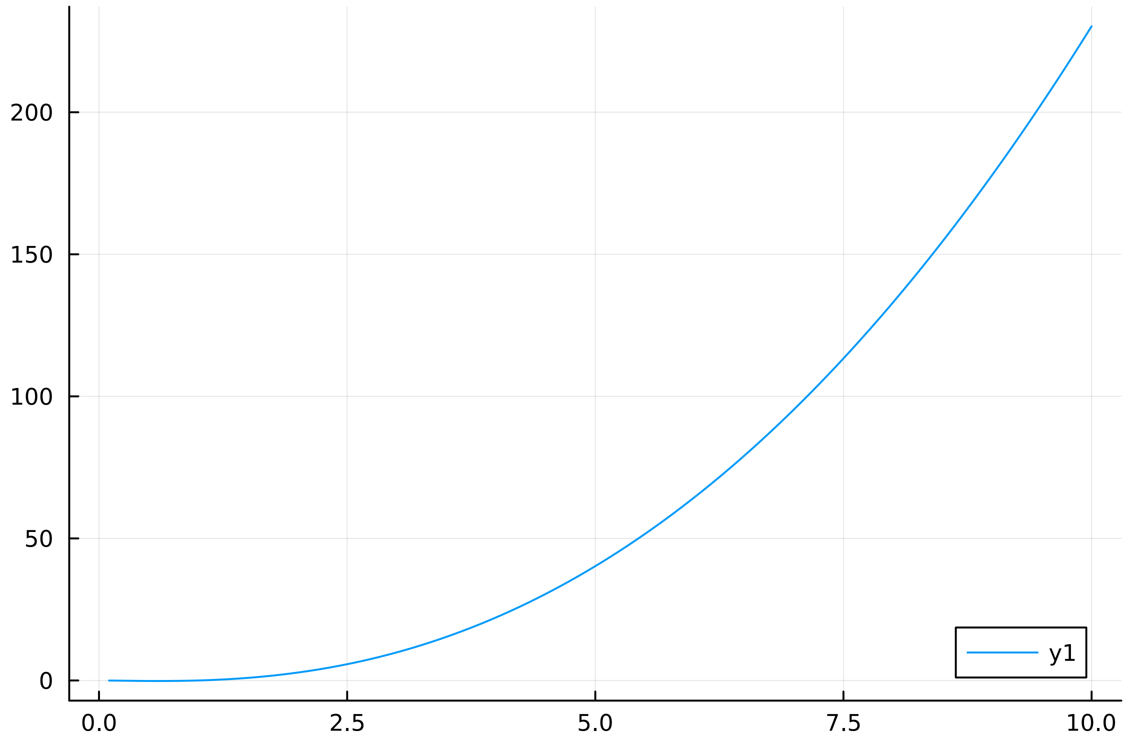
In [5]:
```
y = x -> x.^2 * log.(x)
```

Out[5]:
```
#5 (generic function with 1 method)
```

Brent's doesn't work here because y has $f(x) = x^2 \ln(x)$ has no roots. It is undefined when $x <= 0$, and positive when $x > 0$

In [6]:
```
x = range(0, 10, length=100)
plot(x,y)
```

Out[6]:



In [7]:
```
y = x -> (x - 1).^2
```

Out[7]:
```
#7 (generic function with 1 method)
```

In [8]:
```
find_zero(y, (0, 2), Roots.Brent())
```

```
ArgumentError: The interval [a,b] is not a bracketing interval.
You need f(a) and f(b) to have different signs (f(a) * f(b) < 0).
Consider a different bracket or try fzero(f, c) with an initial guess c.



Stacktrace:
  [1] assert_bracket
    @ ~/.julia/packages/Roots/BMiNe/src/Bracketing/bracketing.jl:52 [inlined]
  [2] init_state
    @ ~/.julia/packages/Roots/BMiNe/src/Bracketing/brent.jl:34 [inlined]
  [3] init_state(M::Roots.Brent, F::Roots.Callable_Function{Val{1}, Val{false}, var"
#7#8", Nothing}, x::Tuple{Int64, Int64})
    @ Roots ~/.julia/packages/Roots/BMiNe/src/Bracketing/bracketing.jl:6
  [4] #init#42
    @ ~/.julia/packages/Roots/BMiNe/src/find_zero.jl:299 [inlined]
  [5] init
    @ ~/.julia/packages/Roots/BMiNe/src/find_zero.jl:289 [inlined]
  [6] solve(FX::ZeroProblem{var"#7#8", Tuple{Int64, Int64}}, M::Roots.Brent, p::Not
hing; verbose::Bool, kwargs::Base.Pairs{Symbol, Roots.NullTracks, Tuple{Symbol}, Nam
edTuple{(:tracks,), Tuple{Roots.NullTracks}}})
    @ Roots ~/.julia/packages/Roots/BMiNe/src/find_zero.jl:491
  [7] find_zero(f::Function, x0::Tuple{Int64, Int64}, M::Roots.Brent, p′::Nothing; p
::Nothing, verbose::Bool, tracks::Roots.NullTracks, kwargs::Base.Pairs{Symbol, Union
{}, Tuple{}, NamedTuple{(), Tuple{}}})
    @ Roots ~/.julia/packages/Roots/BMiNe/src/find_zero.jl:220
  [8] find_zero
    @ ~/.julia/packages/Roots/BMiNe/src/find_zero.jl:210 [inlined]
  [9] find_zero(f::Function, x0::Tuple{Int64, Int64}, M::Roots.Brent)
    @ Roots ~/.julia/packages/Roots/BMiNe/src/find_zero.jl:210
 [10] top-level scope
    @ In[8]:1
```

Brent's doesn't work because you cannot have a bracketing interval, since $(x-1)^2 \geq 0 \forall x$

We can use Newton's Method

$f'(x) = 2(x-1)$ which has the same zero (the solution of $x-1=0$), which we pretend not to know...

Thus, we can use Newton's Method on $f'(x)$ with the same results.

In [9]:
```julia
y = x -> 2*(x-1)
yp = x -> 2
```

Out[9]:
```
#11 (generic function with 1 method)
```

In [10]:
```julia
function newton(x)
    return x - (y(x))/(yp(x))
end
```

Out[10]:
```
newton (generic function with 1 method)
```

In [11]:
```julia
a = 2
```

```
for i = 1:12
    a = newton(a)
end
a
```

Out[11]: 1.0

In [12]:
```
y = x->(x-1).^2
x = range(-5, 5, length=100)
plot(x,y)
```

Out[12]: