Newton

Relaxation

Secant

Bisection

$g(x) = x - f(x)/f'(x)$

$g(x) = x - \lambda f(x)$

$\lambda = 1/f'(\xi)$

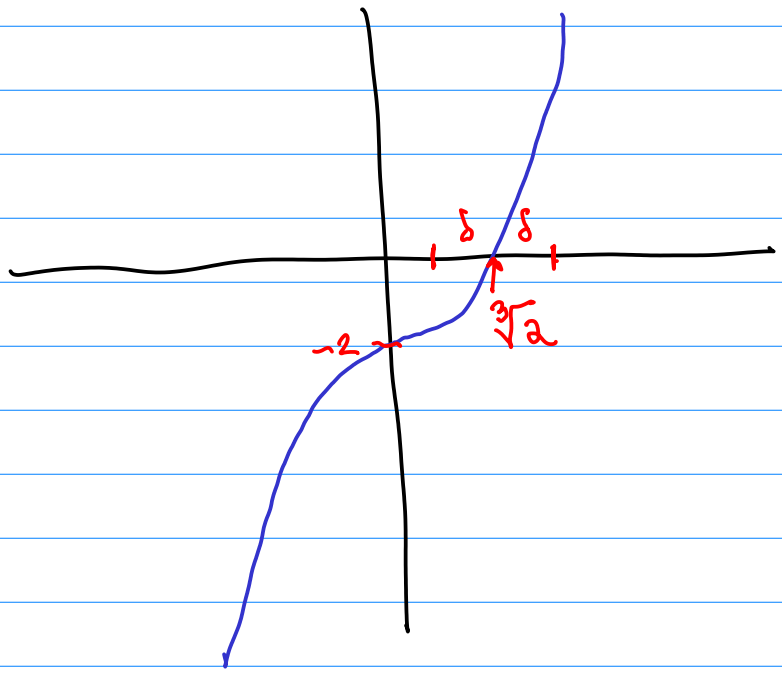
optimal choice but many others work.

Find $\sqrt[3]{2}$.

$f(x) = x^3 - 2$

$f'(x) = 3x^2$

Relaxation method




```
julia> include("relaxation.jl")
n=1 xn=2 en=0.7400789501051268
n=2 xn=-4 en=-5.259921049894873
n=3 xn=62 en=60.74007895010513
n=4 xn=-238264 en=-238265.2599210499
n=5 xn=13526183829105482 en=1.3526183829105482e16
n=6 xn=-87167673028280220 en=-8.7167673028280220e17
n=7 xn=1465923946645329446 en=1.4659239466453294e18
```

for $\lambda = 1$

λ is way too big!

```
julia> include("relaxation.jl")
n=1 xn=1.5 en=0.2400789501051268
n=2 xn=0.8125 en=-0.4474210498948732
n=3 xn=1.5443115234375 en=0.2843904735426268
n=4 xn=0.7027987287556243 en=-0.5571223211392489
n=5 xn=1.529233427632403 en=0.2693123777375303 6
n=6 xn=0.741135283122413 en=-0.5187857667724602
n=7 xn=1.5375893301908254 en=0.2776682802959522 5
```

$\lambda = 0.5$ still not working...

From last week

$$\lambda = \frac{2}{M+m} \quad \text{where} \quad m \le f'(x) \le M \quad \text{for} \quad x \in [\xi-\delta, \xi+\delta]$$

$$\lambda = \frac{1}{\left(\frac{M+m}{2}\right)}$$ ← average between the largest and smallest values of f' on $[\xi-\delta, \xi+\delta]$

```
julia> include("relaxation.jl")
n=1 xn=1.25 en=-0.00992104989487319
n=2 xn=1.26171875 en=0.0017977001051268093
n=3 xn=1.259575441479683 en=-0.0003456084151902683
n=4 xn=1.2599867929927655 en=6.574309789231236e-5
n=5 xn=1.2599085184114487 en=-1.2531483424504941e-5
n=6 xn=1.2599234376305355 en=2.387735662301438e-6
n=7 xn=1.2599205949045715 en=-4.5499030165707666e-7
n=8 xn=1.2599211365934386 en=8.669856543797039e-8
n=9 xn=1.259921033374386 en=-1.6520487200466505e-8
n=10 xn=1.2599210530428648 en=3.147991645136017e-9
n=11 xn=1.2599210492950208 en=-5.998523899819475e-10
n=12 xn=1.2599210500091755 en=1.1430234536646822e-10
n=13 xn=1.2599210498730926 en=-2.1780577341701246e-11
n=14 xn=1.2599210498990234 en=4.150235710653 76e-12
n=15 xn=1.2599210498940825 en=-7.907008381380365e-13
n=16 xn=1.2599210498950237 en=1.5054624213917123e-13
n=17 xn=1.2599210498948445 en=-2.864375403532904e-14
```

$\lambda = 0.25$

```
julia> include("relaxation.jl")
n=1 xn=1.2099868416491455 en=-0.04993420824572769 6
n=2 xn=1.257968161915301 en=-0.001952887979572271
n=3 xn=1.2599180244664132 en=-3.025428459979551e-6
n=4 xn=1.2599210498876083 en=-7.264855383937174e-12
n=5 xn=1.2599210498948732 en=0.0
n=6 xn=1.2599210498948732 en=0.0
```

$\lambda = 1/f'(\xi) \approx 0.209987\ldots$

almost as good as Newton's method...

```
julia> include("relaxation.jl")
n=1 xn=1.02 en=-0.2399210498948731 7
n=2 xn=1.03877584 en=-0.2211452098948731 8
n=3 xn=1.056357909613425 en=-0.2035631402814481 7
n=4 xn=1.072782282102464 en=-0.1871387677924167 8
n=5 xn=1.088089798606736 8 en=-0.1718312512881363 7
n=6 xn=1.102325150726602 en=-0.157595899168271 1
n=7 xn=1.115535987714931 en=-0.144385062179942 3
n=8 xn=1.127772069793016 3 en=-0.1321489801018569 4
n=9 xn=1.139084484122038 7 en=-0.1208365657728345 3
n=10 xn=1.149524935072704 1 en=-0.1103961148221690 4
n=11 xn=1.159145115904448 1 en=-0.1007759339904250 6
n=12 xn=1.167996164973735 6 en=-0.0919248849211376
n=13 xn=1.176128206242797 1 en=-0.0837928436520760 7
n=14 xn=1.183589971193561 1 en=-0.0763310787013120 6
n=15 xn=1.190428497251986 5 en=-0.0694925526428866 7
n=16 xn=1.196688896443154 1 en=-0.0632321534517190 6
n=17 xn=1.202414187151198 2 en=-0.0575068627436750
```

$\lambda = 0.2$ linear convergence but very slow

```
xn,xnm1=g(xn,xnm1),xn
```

evaluates this 2 tuple first and then assigns it to whatever on the left

Equivalent

$t \leftarrow g(x_n, x_{nm1})$

$x_{nm1} \leftarrow x_n$

$x_n \leftarrow t$

```
julia> include("secant.jl")
n=2 xn=1.2105263157894737 en=-0.04939473410539952
n=3 xn=1.2514085388756 73 en=-0.00851251101920014 2
n=4 xn=1.260265275839012 en=0.000344225944138854
n=5 xn=1.2599187140887995 en=-2.335806073672231e-6
n=6 xn=1.2599210492568176 en=-6.380556083485089e-10
n=7 xn=1.2599210498948743 en=1.1102230246251565e-15
```

ran out of digits of precision

```
julia> include("secant.jl")
n=2 xn=1.2105263157894737 en=-0.0493947341053995 2
n=3 xn=1.2514085388756733 en=-0.008512511019200142
n=4 xn=1.260265275839012 en=0.00034422594413885 4
n=5 xn=1.259918714088799 5 en=-2.335806073672231e-6
n=6 xn=1.2599210492568176 en=-6.380556083485089e-10
n=7 xn=1.2599210498948743 en=1.1102230246251565e-15
n=8 xn=1.2599210498948732 en=0.0
n=9 xn=1.2599210498948732 en=0.0
n=10 xn=NaN en=NaN
```

$$g(x_n, x_{n-1}) = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}$$

at the 10th this denominator was zero because the root was already found up to the number of significant digits available...

```
julia> 53*log(2)/log(10)
15.954589770191001
```

about 15 digits of precision...

- Sign bit: 1 bit
- Exponent: 11 bits
- Significand precision: 53 bits (52 explicitly stored)

Mantissa

```
n=6 xn=1.259921049256817617989224142489352215779932766434826097728551694279753969376789 en=-6.380555467779864647888761347903186982666818823534234178755457071371668901513247e-10
n=7 xn=1.259921049894874347679317808334646382541867721334166496656439017200410432036987 en=1.182912107201056418031971616256632658516574463905045110755523030674818965643774e-15
n=8 xn=1.259921049894873164767210008221936951781886384275792440149207675393224722853742 en=-5.990562913987883650804257155399327674367620749536602141107564739745305009345836e-25
n=9 xn=1.259921049894873164767210607278228350569689023951708841471806201346978118702051 en=-5.624407497991386101689108083215578119054230144036155768068374396135029166834383e-40
n=10 xn=1.259921049894873164767210607278228350570251464701507980081975112422724107381542 en=2.674244308675858806139229457982093029097629293406562745250266783237482311037566e-64
```

with more precision can see that 60% more digits are correct each iteration and the code is okay...

Next time is lab...