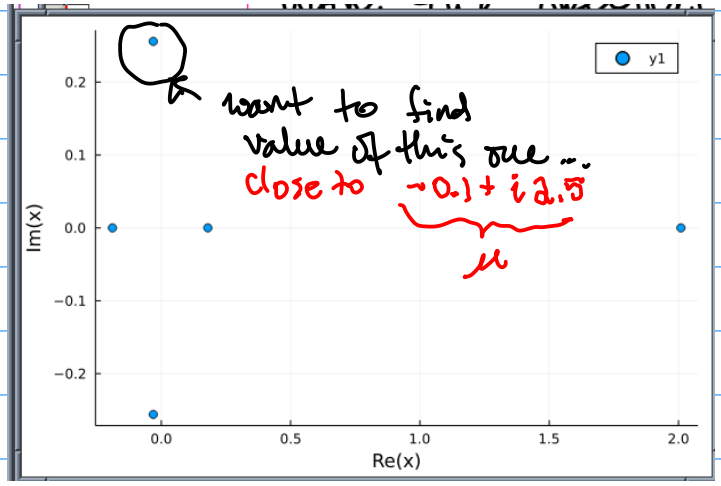
Using the spectral mapping theorem to find other eigenvalues of a matrix.

```
julia> using LinearAlgebra

julia> A=rand(5,5)
5×5 Matrix{Float64}:
 0.272674  0.0585042  0.536515  0.0727685  0.293427
 0.21885   0.314038   0.49899   0.0375723  0.158291
 0.332157  0.501655   0.648209  0.115574   0.319924
 0.598795  0.871543   0.507402  0.105962   0.337564
 0.633303  0.824295   0.920495  0.949212   0.597828

julia> eigvals(A)
5-element Vector{ComplexF64}:
   -0.1888173161842402 + 0.0im
 -0.030973371225072897 - 0.25593363499612165im
 -0.030973371225072897 + 0.25593363499612165im
   0.18004675007438675 + 0.0im
     2.009429401835247 + 0.0im
```

```
julia> scatter(eigvals(A))
```




```
julia> v0=rand(5); v0=v0/norm(v0)
5-element Vector{Float64}:
 0.5295496223141816
 0.0610295821869298 25
 0.09775794270487649
 0.5252666964930642
 0.6560418201596373
```

$(A-\mu I)^{-1}$

```
julia> v=v0
       for k=1:7
          w=A*v
          v=w/norm(w)
          lambda=v'*A*v
          println("lambda=",lambda)
       end
lambda=1.905495751832909
lambda=2.0011920163045867
lambda=2.0105876250367216
lambda=2.0094156179139953
lambda=2.0094050227221447
lambda=2.0094294772884713
lambda=2.009429756839256
```

$$B = (A-\mu I)^{-1} = (A-(-.1+i2.5)I)^{-1}$$

```
julia> mu=-0.1+2.5im
-0.1 + 2.5im

julia> B=inv(A-mu*I)
5×5 Matrix{ComplexF64}:
 0.0414189+0.379536im   …  0.0311536-0.0199753im
 0.0202004-0.0167821im     0.0122099-0.0153435im
 0.0286597-0.0274724im     0.0297935-0.0253626im
 0.0675683-0.0349037im     0.0292338-0.0291479im
 0.0485651-0.0669159im     0.0664626+0.347461im
```

```
julia> v=v0
       for k=1:20
          w=B*v
          v=w/norm(w)
          lambda=v'*B*v
          println("lambda=",lambda)
       end
lambda=0.14360299922506325 + 0.29488574885395324im
lambda=0.11419545298401862 + 0.31550441714008437im
lambda=0.07794262145676142 + 0.33575201003963 97im
lambda=0.04152049883772445 + 0.35920340834965514im
lambda=0.01576354267895501 + 0.39082100930431757im
lambda=0.010676908886473623 + 0.42512253455880433im
lambda=0.02100815333773875 + 0.44740078757507845im
lambda=0.03174322852932889 + 0.45382187773098936im
lambda=0.03600297499937477 + 0.45178227049403397im
lambda=0.03475012020587065 + 0.44800229963772 04im
lambda=0.03075572243763898 + 0.44548333296493237im
lambda=0.026265546728197003 + 0.44485372145060 09im
lambda=0.022617742979713792 + 0.44561243868804 57im
lambda=0.020290102860268375 + 0.44689244916761195im
lambda=0.019116195456253746 + 0.44796589694603745im
lambda=0.01862907583421483 + 0.44848742600746 91im
lambda=0.018379815279169445 + 0.44846748248104024im
lambda=0.01809847144787949 + 0.44810091241407424im
lambda=0.017699282979998496 + 0.44760287620333 91im
lambda=0.017210489808444113 + 0.44712207662060 22im
```

eigenvalue of Largest Magnitude for B

global

$$Bx = \lambda x$$

$$(A-\mu I)^{-1}x = \lambda x$$

$$x = \lambda (A - \mu I) x$$

$$x = \lambda A x - \lambda \mu x$$

$$(1 + \lambda\mu) x = \lambda A x$$

$$Ax = \frac{1+\lambda\mu}{\lambda} x$$

The corresponding eigenvalue of A is

$$\frac{1+\lambda\mu}{\lambda} \approx$$

```
julia> (1+lambda*mu)/lambda
-0.014039679945544859 + 0.26678309347930085im
```

trying to find...

```
julia> eigvals(A)
5-element Vector{ComplexF64}:
   -0.1888173161842402 + 0.0im
 -0.030973371225072897 - 0.25593363499612165im
 -0.030973371225072897 + 0.25593363499612165im
    0.18004675007438675 + 0.0im
      2.009429401835247 + 0.0im
```

Try with more iterations

```
julia> v=v0
       for k=1:20000
          w=B*v
          v=w/norm(w)
          global lambda=v'*B*v
          # println("lambda=",lambda)
       end
```

Much better...

differ by a few digits

```
julia> (1+lambda*mu)/lambda
-0.030973371225072748 + 0.25593363499612154im
```

Question: How to make this converge faster

Algorithm 27.3. Rayleigh Quotient Iteration

$v^{(0)}$ = some vector with $\|v^{(0)}\| = 1$
$\lambda^{(0)} = (v^{(0)})^T A v^{(0)}$ = corresponding Rayleigh quotient
for $k = 1, 2, \ldots$

Solve $(A - \lambda^{(k-1)} I) w = v^{(k-1)}$ for w	apply $(A - \lambda^{(k-1)} I)^{-1}$
$v^{(k)} = w / \|w\|$	normalize
$\lambda^{(k)} = (v^{(k)})^T A v^{(k)}$	Rayleigh quotient

Idea: Rather that iterating $(A-\mu I)^{-1}$ where μ is close to the eigenvalue we're looking for. Update μ with better approximation as we iterate...

```
julia> v=v0
       for k=1:7
          global w=(A-mu*I)\v
          global v=w/norm(w)
          global lambda=v'*A*v
          println("lambda=",lambda)
       end
       for k=1:7
          global w=(A-lambda*I)\v
          global v=w/norm(w)
          global lambda=v'*A*v
          println("lambda=",lambda)
       end
lambda=1.3804714995096456 - 0.0036616064869412 49im
lambda=1.1317469909072775 - 0.029366627245914767im
lambda=0.8350748210977686 - 0.08346745604754946im
lambda=0.5092715004455244 - 0.12454773923576323im
lambda=0.19811090584014973 - 0.0790261784305102im
lambda=-0.0021924870315643136 + 0.07508627886664 44im
lambda=-0.04328289238809693 + 0.24169464241065455im
lambda=-0.03685180175720287 + 0.25001832965643 82im
lambda=-0.030855616188898913 + 0.25587536067923666im
lambda=-0.030973376547080114 + 0.25593936308729 6106im
lambda=-0.030973371225073473 + 0.2559336349961221im
lambda=-0.030973371225072766 + 0.25593363499612165im
lambda=-0.030973371225072655 + 0.25593363499612165im
lambda=-0.030973371225072807 + 0.25593363499612165im
```

Inverse iteration

Rayleigh Quotient iteration

7 inverse iteration with μ close to the eigenvalue I'm searching for

7 Rayleigh quotient iterations.

Did inverse iteration first so that the starting vector for the Rayliegh quotent iteration was close enough to an eigenvector for the eigenvalue being searched for...

```
julia> v
5-element Vector{ComplexF64}:
  0.40175506267255817 - 0.1440508323676813im
 -0.06136538507756637 - 0.16671969860356545im
  0.09306872558672342 - 0.05463495493639986im
 -0.23596735787386827 - 0.2754956014220091im
  -0.3895219265545326 + 0.70094831570389l6im

julia> A*v
5-element Vector{ComplexF64}:
 0.024423744454368096 + 0.10728437347408745im
   0.0445698713414453 - 0.010541594950457853im
 0.011100270427680109 + 0.025511645984976933im
  0.07781729526980435 - 0.05185895610737126im
  -0.1673314431510271 - 0.12140249496568006im

julia> lambda*v
5-element Vector{ComplexF64}:
 0.024423744454368117 + 0.10728437347408741im
 0.044569871341445334 - 0.010541594950457837im
   0.0111002704276801 + 0.025511645984976915im
  0.07781729526980441 - 0.05185895610737l235im
 -0.16733144315102694 - 0.12140249496568001im
```

```
julia> v=v0
       lambda=v'*A*v
       for k=1:7
          global w=(A-lambda*I)\v
          global v=w/norm(w)
          global lambda=v'*A*v
          println("lambda=",lambda)
       end
lambda=2.039775703445323
lambda=2.0099365779537264
lambda=2.0094295401006557
lambda=2.0094294018352548
lambda=2.0094294018352445
lambda=2.0094294018352445
lambda=2.0094294018352445
```

If one leaves out the "warmup" using inverse iteration, it could converge to any eigenvalue.

- In this case the one with largest magnitude, but which one depends on v_0.