```
pyplot (generic function with 1 method)
```

```julia
begin
    using Pkg
    Pkg.activate(mktempdir())
    Pkg.add("PyPlot")
    Pkg.add("PlutoUI")
    Pkg.add("GraphPlot");
    Pkg.add("LightGraphs");
    Pkg.add("Colors")

    using PlutoUI
    using PyPlot
    using LinearAlgebra
    using SparseArrays

    using GraphPlot, LightGraphs,Colors

    function pyplot(f;width=3,height=3)
        clf()
        f()
        fig=gcf()
        fig.set_size_inches(width,height)
        fig
    end


end
```

Eigenvalue analysis for more general matrices

For 1D heat conduction we had a very special regular structure of the matrix which allowed exact eigenvalue calculations.

We need a generalization to varying coefficients, nonsymmetric problems, unstructured grids $\ldots$\ $\Rightarrow$ what can be done for general matrices ?

The Gershgorin Circle Theorem

Theorem (Varga, Th. 1.11) Let A be an $n \times n$ (real or complex) matrix. Let Λ_i be the sum of the absolute values of the i-th row's off-diagonal entries:

$$\Lambda_i = \sum_{\substack{j=1\ldots n\\ j\neq i}} |a_{ij}|$$

If λ is an eigenvalue of A, then there exists r, $1 \leq r \leq n$ such that λ lies on the disk defined by the circle of radius Λ_r around a_{rr}:

$$|\lambda - a_{rr}| \leq \Lambda_r$$

Proof: Assume λ is an eigenvalue, $\vec{x} = (x_1 \ldots x_n)$ is a corresponding eigenvector. Assume $\vec{x}$ is normalized such that

$$\max_{i=1\ldots n} |x_i| = |x_r| = 1.$$

From $A\vec{x} = \lambda\vec{x}$ it follows that

$$\begin{aligned}
\lambda x_i &= \sum_{j=1\ldots n} a_{ij}x_j \\
(\lambda - a_{ii})x_i &= \sum_{\substack{j=1\ldots n\\ j\neq i}} a_{ij}x_j \\
|\lambda - a_{rr}| &= |\sum_{\substack{j=1\ldots n\\ j\neq r}} a_{rj}x_j| \leq \sum_{\substack{j=1\ldots n\\ j\neq r}} |a_{rj}||x_j| \leq \sum_{\substack{j=1\ldots n\\ j\neq r}} |a_{rj}| = \Lambda_r
\end{aligned}$$

□

Corollary Any eigenvalue $\lambda \in \sigma(A)$ lies in the union of the disks defined by the Gershgorin circles

$$\lambda \in \bigcup_{i=1\dots n} \{\mu \in \mathbb{C} : |\mu - a_{ii}| \leq \Lambda_i\}$$

Corollary The Gershgorin circle theorem allows to estimate the spectral radius $\rho(A)$:

$$\rho(A) \leq \max_{i=1\dots n} \sum_{j=1}^{n} |a_{ij}| = ||A||_\infty,$$

$$\rho(A) \leq \max_{j=1\dots n} \sum_{i=1}^{n} |a_{ij}| = ||A||_1.$$

Proof:

$$|\mu - a_{ii}| \leq \Lambda_i \quad \Rightarrow \quad |\mu| \leq \Lambda_i + |a_{ii}| = \sum_{j=1}^{n} |a_{ij}|$$

Furthermore, $\sigma(A) = \sigma(A^T)$.

□

This appears to be very easy to use, so let us try:

```
gershgorin_circles (generic function with 1 method)
```

```
function gershgorin_circles(A)
    t=0:0.01*π:2π

    # α is the trasnparency value.
    circle(x,y,r;α=0.3)=fill(x.+r.*cos.(t),y.+r.*sin.(t),alpha=α)

    n=size(A,1)
    for i=1:n
        Λ=0
        for j=1:n
            if j!=i
                Λ+=abs(A[i,j])
            end
        end
        circle(real(A[i,i]),imag(A[i,i]),Λ,α=0.5/n)
    end

    σ=eigvals(Matrix(A))
    scatter(real(σ),imag(σ),sizes=10*ones(n),color=:red)
end
```

```
n1 = 5
```

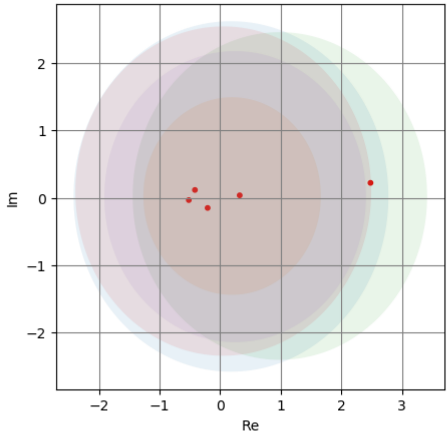
```
n1=5
```

```
A1 = 5×5 Array{Complex{Float64},2}:
 0.178502+0.0192365im  0.744411+0.093342im   …  0.392995+0.0961931im
 0.285649+0.0786766im  0.196577+0.0252502im     0.564553+0.0803397im
 0.169284+0.0438933im  0.755742+0.0518099im      0.55119+0.0852875im
 0.669057+0.0866883im  0.867834+0.0229806im     0.607358+0.0276727im
 0.480013+0.0704921im  0.581088+0.0177494im     0.247122+0.0167818im
```

```
A1=rand(n1,n1)+0.1*rand(n1,n1)*1im
```

```
Complex{Float64}[-0.518821-0.0347195im, -0.417636+0.115591im, -0.207874-0.151289im, 0.3
```

```
eigvals(A1)
```




```
pyplot(width=5, height=5) do
    gershgorin_circles(A1)
    xlabel("Re")
    ylabel("Im")
    PyPlot.grid(color=:gray)
end
```

So this is kind of cool! Let us try this out with our heat example and the Jacobi iteration matrix: $B = I - D^{-1}A$

```
heatmatrix1d (generic function with 1 method)
```

```
function heatmatrix1d(N;α=100)
    A=zeros(N,N)
    h=1/(N-1)
    A[1,1]=1/h+α
    for i=2:N-1
        A[i,i]=2/h
    end
    for i=1:N-1
        A[i,i+1]=-1/h
    end
    for i=2:N
        A[i,i-1]=-1/h
    end
    A[N,N]=1/h+α
    A
end
```

```
jacobi_iteration_matrix (generic function with 1 method)
```

```
jacobi_iteration_matrix(A)=I-inv(Diagonal(A))*A
```

```
N = 10
```

```
N=10
```

```
A2 = 10×10 Tridiagonal{Float64,Array{Float64,1}}:
 109.0  -9.0    ⋅     ⋅     ⋅     ⋅     ⋅     ⋅     ⋅      ⋅
  -9.0  18.0  -9.0    ⋅     ⋅     ⋅     ⋅     ⋅     ⋅      ⋅
    ⋅   -9.0  18.0  -9.0    ⋅     ⋅     ⋅     ⋅     ⋅      ⋅
    ⋅     ⋅   -9.0  18.0  -9.0    ⋅     ⋅     ⋅     ⋅      ⋅
    ⋅     ⋅     ⋅   -9.0  18.0  -9.0    ⋅     ⋅     ⋅      ⋅
    ⋅     ⋅     ⋅     ⋅   -9.0  18.0  -9.0    ⋅     ⋅      ⋅
    ⋅     ⋅     ⋅     ⋅     ⋅   -9.0  18.0  -9.0    ⋅      ⋅
    ⋅     ⋅     ⋅     ⋅     ⋅     ⋅   -9.0  18.0  -9.0     ⋅
    ⋅     ⋅     ⋅     ⋅     ⋅     ⋅     ⋅   -9.0  18.0   -9.0
    ⋅     ⋅     ⋅     ⋅     ⋅     ⋅     ⋅     ⋅   -9.0  109.0
```

```
A2=Tridiagonal(heatmatrix1d(N))
```

```
B2 = 10×10 Tridiagonal{Float64,Array{Float64,1}}:
 0.0  0.0825688   ⋅    ⋅    ⋅    ⋅    ⋅    ⋅    ⋅          ⋅
 0.5  0.0        0.5   ⋅    ⋅    ⋅    ⋅    ⋅    ⋅          ⋅
  ⋅   0.5        0.0  0.5   ⋅    ⋅    ⋅    ⋅    ⋅          ⋅
  ⋅    ⋅         0.5  0.0  0.5   ⋅    ⋅    ⋅    ⋅          ⋅
  ⋅    ⋅          ⋅   0.5  0.0  0.5   ⋅    ⋅    ⋅          ⋅
  ⋅    ⋅          ⋅    ⋅   0.5  0.0  0.5   ⋅    ⋅          ⋅
```

```
 ⋅  ⋅      ⋅  ⋅  ⋅  0.5  0.0  0.5  ⋅          ⋅
 ⋅  ⋅      ⋅  ⋅  ⋅   ⋅   0.5  0.0  0.5        ⋅
 ⋅  ⋅      ⋅  ⋅  ⋅   ⋅    ⋅   0.5  0.0        0.5
 ⋅  ⋅      ⋅  ⋅  ⋅   ⋅    ⋅    ⋅   0.0825688  0.0
```

```
B2=jacobi_iteration_matrix(A2)
```

ρ2 = 0.9421026930965894

```
ρ2=maximum(abs.(eigvals(Matrix(B2))))
```

We have $b_{ii}=0$, $\Lambda_i = \begin{cases} \frac{1}{1+\alpha h}, & i=1,n \\ 1 & i=2\dots n-1 \end{cases}$

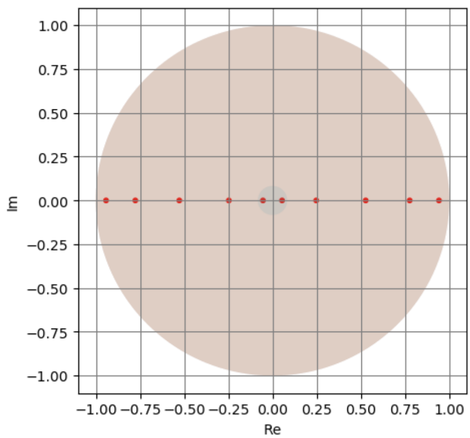
We see two circles around 0: one with radius 1 and one with radius $\frac{1}{1+\alpha h}$

$\Rightarrow$ estimate $|\lambda_i| \leq 1$

We can also caculate the value from the estimate: Gershgorin circles of $B2$ are centered in the origin, and the spectral radius estimate just consists in the maximum of the sum of the absolute values of the row entries.

ρ2_gershgorin = 1.0

```
ρ2_gershgorin=maximum([sum( abs.(B2[i,:])) for i=1:size(B2,1)])
```




```
pyplot(width=5, height=5) do
    gershgorin_circles(B2)
        xlabel("Re")
    ylabel("Im")

    PyPlot.grid(color=:gray)
end
```

So the estimate from the Gershgorin Circle theorem is very pessimistic... Can we improve this ?

Matrices and Graphs

- Permutation matrices are matrices which have exactly one non-zero entry in each row and each column which has value 1.
- There is a one-to-one correspondence permutations π of the the numbers $1\dots n$ and $n\times n$ permutation matrices $P=(p_{ij})$ such that

$$p_{ij} = \begin{cases} 1, & \pi(i)=j \\ 0, & \text{else} \end{cases}$$

- Permutation matrices are orthogonal, and we have $P^{-1}=P^T$

- $A \to PA$ permutes the rows of A
- $A \to AP^T$ permutes the columns of A

Define a directed graph from the nonzero entries of a matrix $A = (a_{ik})$:

- Nodes: $\mathcal{N} = \{N_i\}_{i=1\ldots n}$
- Directed edges: $\mathcal{E} = \{\overrightarrow{N_k N_l} | a_{kl} \neq 0\}$
- Matrix entries $\equiv$ weights of directed edges
- 1:1 equivalence between matrices and weighted directed graphs

Create a bidirectional graph (digraph) from a matrix in Julia. Create edge labels from off-diagonal entries and node labels combined from diagonal entries and node indices.

```
create_graph (generic function with 1 method)
```

```
function create_graph(matrix)
    @assert size(matrix,1)==size(matrix,2)
    n=size(matrix,1)
    g=LightGraphs.SimpleDiGraph(n)
    elabel=[]
    nlabel=Any[]
    for i in 1:n
        push!(nlabel,"""$(i) \n $(matrix[i,i])""")
        for j in 1:n
            if i!=j && matrix[i,j]>0
                add_edge!(g,i,j)
                push!(elabel,matrix[i,j])
            end
        end
     end
    g,nlabel,elabel
end
```

```
rndmatrix (generic function with 1 method)
```

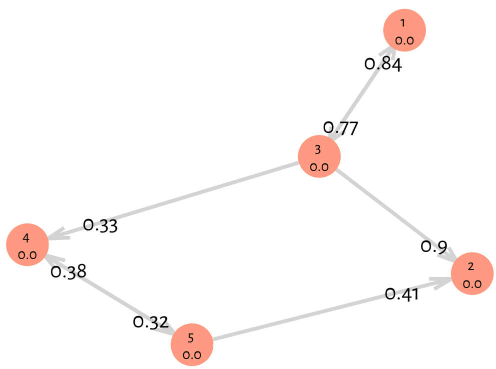
```
# sparse random matrix with entries with limited numbers decimal values
rndmatrix(n,p)=rand(0:0.01:1,n,n).*Matrix(sprand(Bool,n,n,p))
```

```
A3 = 5×5 Array{Float64,2}:
      0.0   0.0   0.77  0.0   0.0
      0.0   0.0   0.0   0.0   0.0
      0.84  0.9   0.0   0.33  0.0
      0.0   0.0   0.0   0.0   0.32
      0.0   0.41  0.0   0.38  0.0
```

```
A3=rndmatrix(5,0.3)
```

```
({5, 7} directed simple Int64 graph, Any["1 \n 0.0", "2 \n 0.0", "3 \n 0.0", "4 \n 0.0
```

```
graph3,nlabel3,elabel3=create_graph(A3)
```




```
GraphPlot.gplot(graph3,
    nodelabel=nlabel3,
    edgelabel=elabel3,
    nodefillc=RGB(1.0,0.6,0.5),
```

```
    EDGELABELSIZE=6.0,
    NODESIZE=0.1,
    EDGELINEWIDTH=1
)
```

- Matrix graph of $A3$ is strongly connected: **false**
- Matrix graph of $A3$ is weakly connected: **true**

Definition A square matrix A is *reducible* if there exists a permutation matrix P such that

$$PAP^T = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

A is *irreducible* if it is not reducible.

Theorem (Varga, Th. 1.17): A is irreducible $\Leftrightarrow$ the matrix graph is strongly connected, i.e. for each *ordered* pair (N_i, N_j) there is a path consisting of directed edges, connecting them.

Equivalently, for each i, j there is a sequence of consecutive nonzero matrix entries $a_{ik_1}, a_{k_1k_2}, a_{k_2k_3}\dots, a_{k_{r-1}k_r}a_{k_rj}$.

The Taussky theorem

Theorem (Varga, Th. 1.18) Let A be irreducible. Assume that the eigenvalue λ is a boundary point of the union of all the disks

$$\lambda \in \partial \bigcup_{i=1\dots n} \{\mu \in \mathbb{C} : |\mu - a_{ii}| \leq \Lambda_i\}$$

Then, all n Gershgorin circles pass through λ, i.e. for $i = 1\dots n$,

$$|\lambda - a_{ii}| = \Lambda_i$$

Proof Assume λ is eigenvalue, $\vec{x}$ a corresponding eigenvector, normalized such that $\max_{i=1\dots n}|x_i| = |x_r| = 1$. From $A\vec{x} = \lambda\vec{x}$ it follows that

$$\begin{aligned}(\lambda - a_{rr})x_r &= \sum_{\substack{j=1\dots n\\ j\neq r}} a_{rj}x_j \\ |\lambda - a_{rr}| &\leq \sum_{\substack{j=1\dots n\\ j\neq r}} |a_{rj}|\cdot|x_j| \\ &\leq \sum_{\substack{j=1\dots n\\ j\neq r}} |a_{rj}| = \Lambda_r \quad (*)\end{aligned}$$

λ is boundary point $\Rightarrow |\lambda - a_{rr}| = \sum_{\substack{j=1\dots n\\ j\neq r}} |a_{rj}|\cdot|x_j| = \Lambda_r$

$\Rightarrow$ For all $p \neq r$ with $a_{rp} \neq 0$, $|x_p| = 1$.

Due to irreducibility there is at least one p with $a_{rp} \neq 0$. For this p, $|x_p| = 1$ and equation (*) is valid (with p in place of r) $\Rightarrow |\lambda - a_{pp}| = \Lambda_p$

Due to irreducibility, this is true for all $p = 1\dots n$. $\square$

Apply this to the Jacobi iteration matrix for the heat conduction problem: We know that $|\lambda_i| \leq 1$, and we can see that the matrix graph is strongly connected.

Assume $|\lambda_i| = 1$. Then λ_i lies on the boundary of the union of the Gershgorin circles. But then it must lie on the boundary of both circles with radius $\frac{1}{1+\alpha h}$ and 1 around 0.

Contradiction! $\Rightarrow |\lambda_i| < 1$, $\rho(B) < 1!$

```
α = 1
```

```
α=1
```

```
N4 = 5
```

```
N4=5
```

```
A4 = 5×5 Tridiagonal{Float64,Array{Float64,1}}:
  5.0  -4.0    ⋅     ⋅     ⋅
 -4.0   8.0  -4.0    ⋅     ⋅
   ⋅   -4.0   8.0  -4.0    ⋅
   ⋅     ⋅   -4.0   8.0  -4.0
   ⋅     ⋅     ⋅   -4.0   5.0
```

```
A4=Tridiagonal(heatmatrix1d(N4,α=α))
```

```
B4 = 5×5 Tridiagonal{Float64,Array{Float64,1}}:
 0.0  0.8   ⋅    ⋅    ⋅
 0.5  0.0  0.5   ⋅    ⋅
  ⋅   0.5  0.0  0.5   ⋅
  ⋅    ⋅   0.5  0.0  0.5
  ⋅    ⋅    ⋅   0.8  0.0
```

```
B4=jacobi_iteration_matrix(A4)
```

```
ρ4 = 0.9486832980505135
```

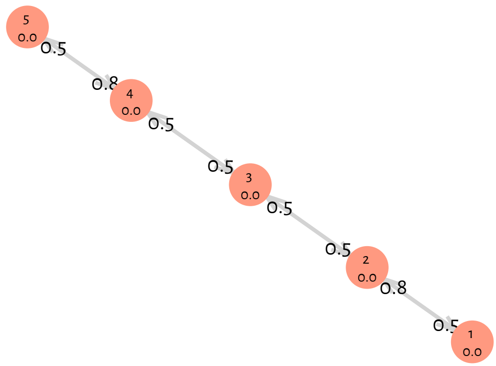
```
ρ4=maximum(abs.(eigvals(Matrix(B4))))
```

```
ρ4_gershgorin = 1.0
```

```
ρ4_gershgorin=maximum([sum( abs.(B4[i,:])) for i=1:size(B4,1)])
```

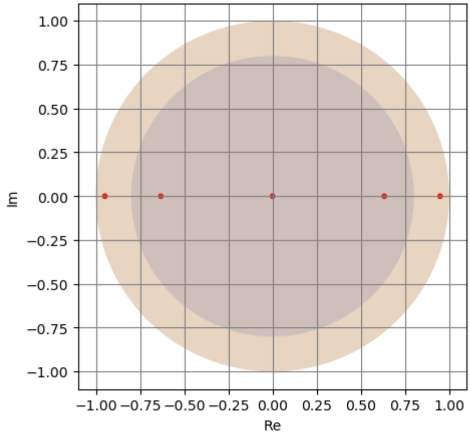
```
({5, 8} directed simple Int64 graph, Any["1 \n 0.0", "2 \n 0.0", "3 \n 0.0", "4 \n 0.0
```

```
graph4,nlabel4,elabel4=create_graph(B4)
```




```
GraphPlot.gplot(graph4,
    # rand(n),rand(n),
    nodelabel=nlabel4,
    edgelabel=elabel4,
    nodefillc=RGB(1.0,0.6,0.5),
    EDGELABELSIZE=6.0,
    NODESIZE=0.1,
    EDGELINEWIDTH=1
)
```

- Matrix graph is strongly connected: **true**
- Matrix graph is weakly connected: **true**




```
pyplot(width=5, height=5) do
    gershgorin_circles(B4)
        xlabel("Re")
    ylabel("Im")

    PyPlot.grid(color=:gray)
end
```

- Unfortunately, we don't get a quantitative estimate here.
- Advantage: we don't need to assume symmetry of A or spectral equivalence estimates