Chapter 4: Eigenvalue Problems

Eigenvalues and Eigenvectors

- Standard *eigenvalue problem: Given $n \times n$ matrix $\mathbf{A}$, find scalar λ and nonzero vector $\mathbf{x}$ such that*

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

- λ is the *eigenvalue* and $\mathbf{x}$ is the *eigenvector*

- λ (and $\mathbf{x}$) may be complex even if $\mathbf{A}$ is real

- *Spectrum* of $\mathbf{A}$ = set of all eigenvalues $\lambda(\mathbf{A})$

- *Spectral radius* $\rho(\mathbf{A}) = \max\{|\lambda| : \lambda \in \lambda(\mathbf{A})\}$

Geometric Interpretation

- The matrix-vector product $\hat{\mathbf{v}} = A\mathbf{v}$ stretches or shrinks any vector $\mathbf{v}$ lying in direction of eigenvector $\mathbf{x}$
- Scalar expansion or contraction factor is given by corresponding λ
- Eigenvalues and eigenvectors lead to simple interpretation of general linear transformations (e.g., as represented by matrix-vector products)
- They are particularly useful when considering iterative processes that can be cast as a sequence of matrix-vector products, such as

$$\mathbf{x}_1 = \mathbf{A}\mathbf{x}_0, \;\; \mathbf{x}_2 = \mathbf{A}\mathbf{x}_1, \; \dots \;, \mathbf{x}_k = \mathbf{A}^k\mathbf{x}_0,$$

- Such sequences are in fact at the core of most of the algorithms used to find the eigenpairs $(\lambda, \mathbf{x})$ of $\mathbf{A}$

Examples: Eigenvalues and Eigenvectors

- $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}: \quad \lambda_1 = 1,\ \mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \lambda_2 = 2,\ \mathbf{x}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

- $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}: \quad \lambda_1 = 1,\ \mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \lambda_2 = 2,\ \mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

- $\mathbf{A} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}: \quad \lambda_1 = 2,\ \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \lambda_2 = 4,\ \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

Examples: Eigenvalues and Eigenvectors

- $\mathbf{A} = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$: $\quad \lambda_1 = 2,\ \mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \lambda_2 = 1,\ \mathbf{x}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

- $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$: $\quad \lambda_1 = i,\ \mathbf{x}_1 = \begin{bmatrix} 1 \\ i \end{bmatrix}, \quad \lambda_2 = -i,\ \mathbf{x}_2 = \begin{bmatrix} i \\ 1 \end{bmatrix}$

Classic Eigenvalue Problem

- Consider the coupled pair of differential equations:

$$\begin{aligned}\frac{dv}{dt} &= 4v - 5w, \quad v = 8 \text{ at } t = 0,\\ \frac{dw}{dt} &= 2v - 3w, \quad w = 5 \text{ at } t = 0.\end{aligned}$$

- This is an ***initial-value problem***.
- With the coefficient matrix,

$$A = \begin{bmatrix} 4 & -5 \\ 2 & -3 \end{bmatrix},$$

we can write this as,

$$\frac{d}{dt}\begin{pmatrix} v(t) \\ w(t) \end{pmatrix} = \begin{bmatrix} 4 & -5 \\ 2 & -3 \end{bmatrix}\begin{pmatrix} v(t) \\ w(t) \end{pmatrix}.$$

- Introducing the *vector unknown*, $\mathbf{u}(t) := [v(t)\ w(t)]^T$ with $\mathbf{u}(0) = [8\ 5]^T$, we can write the system in vector form,

$$\frac{d\mathbf{u}}{dt} = A\mathbf{u}, \quad \text{with } \mathbf{u} = \mathbf{u}(0) \text{ at } t = 0.$$

- How do we find $\mathbf{u}(t)$?

- If we had a 1×1 matrix $A = a$, we would have a scalar equation:

$$\frac{du}{dt} = a\,u \text{ with } u = u(0) \text{ at } t = 0.$$

  The solution to this equation is a pure exponential:

$$u(t) = e^{at}\,u(0),$$

  which satisfies the initial condition because $e^0 = 1$.

- The derivative with respect to t is $ae^{at}u(0) = au$, so it satisfies the scalar initial value problem.

- The constant a is critical to how this system behaves.
  - If $a > 0$ then the solution grows in time.
  - If $a < 0$ then the solution decays.
  - If $a \in Im$ then the solution is oscillatory.
    (More on this later...)

- Coming back to our system, suppose we again look for solutions that are pure exponentials in time, e.g.,

$$\begin{aligned} v(t) &= e^{\lambda t} y \\ w(t) &= e^{\lambda t} z. \end{aligned}$$

- If this is to be a solution to our initial value problem, we require

$$\begin{aligned} \frac{dv}{dt} &= \lambda e^{\lambda t} y = 4e^{\lambda t} y - 5e^{\lambda t} z \\ \frac{dw}{dt} &= \lambda e^{\lambda t} z = 2e^{\lambda t} y - 3e^{\lambda t} z. \end{aligned}$$

- The $e^{\lambda t}$ cancels out from each side, leaving:

$$\begin{aligned} \lambda y &= 4y - 5z \\ \lambda z &= 2y - 3z, \end{aligned}$$

which is the eigenvalue problem.

$$\boxed{\begin{bmatrix} 4 & -5 \\ 2 & -3 \end{bmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \begin{pmatrix} y \\ z \end{pmatrix}}$$

- In vector form, $\mathbf{u}(t) \;=\; e^{\lambda t}\mathbf{x}$, yields

$$\frac{d\mathbf{u}}{dt} \;=\; A\mathbf{u} \;\;\Longleftrightarrow\;\; \lambda e^{\lambda t}\mathbf{x} \;=\; A(e^{\lambda t}\mathbf{x})$$

  which gives the eigenvalue problem in matrix form:

$$\lambda\mathbf{x} \;=\; A\mathbf{x} \quad \text{or}$$

$$A\mathbf{x} \;=\; \lambda\mathbf{x}.$$

- As in the scalar case, the solution behavior depends on whether λ has
  - positive real part $\longrightarrow$ a growing solution,
  - negative real part $\longrightarrow$ a decaying solution,
  - an imaginary part $\longrightarrow$ an oscillating solution.
- Note that here we have two unknowns: λ and $\mathbf{x}$.
- We refer to $(\lambda, \mathbf{x})$ as an eigenpair, with *eigenvalue* λ and *eigenvector* $\mathbf{x}$.

Solving the Eigenvalue Problem

- The eigenpair satisfies

$$(A - \lambda I)\,\mathbf{x} = 0,$$

which is to say,

  - $\mathbf{x}$ is in the null-space of $A - \lambda I$
  - λ is chosen so that $A - \lambda I$ has a null-space.

- We thus seek λ such that $A - \lambda I$ is singular.
- Singularity implies $det(A - \lambda I)$=0.
- For our example:

$$0 = \begin{vmatrix} 4-\lambda & -5 \\ 2 & -3-\lambda \end{vmatrix} = (4-\lambda)(-3-\lambda) - (-5)(2),$$

or

$$\lambda^2 - \lambda - 2 = 0,$$

which has roots $\lambda = -1$ or $\lambda = 2$.

Finding the Eigenvectors

- For the case $\lambda = \lambda_1 = -1$, $(A - \lambda_1 I)\mathbf{x}_1$ satisfies,

$$\begin{bmatrix} 5 & -5 \\ 2 & -2 \end{bmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

which gives us the eigenvector $\mathbf{x}_1$

$$\mathbf{x}_1 = \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

- **Note that any nonzero multiple of $\mathbf{x}_1$ is also an eigenvector.**
- Thus, $\mathbf{x}_1$ defines a *subspace* that is invariant under multiplication by A.

Because $Ax_1 = \lambda\, x_1$, i.e., it is simply a stretching of x_1

- For the case $\lambda = \lambda_2 = 2$, $(A - \lambda_2 I)\mathbf{x}_2$ satisfies,

$$\begin{bmatrix} 2 & -5 \\ 2 & -5 \end{bmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

which gives us the second eigenvector as any multiple of

$$\mathbf{x}_2 = \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} 5 \\ 2 \end{pmatrix}.$$

Return to Model Problem

- Note that our model problem $\frac{d\mathbf{u}}{dt} = A\mathbf{u}$, is *linear* in the unknown $\mathbf{u}$.
- Thus, if we have two solutions $\mathbf{u}_1(t)$ and $\mathbf{u}_2(t)$ satisfying the differential equation, their sum $\mathbf{u} := \mathbf{u}_1 + \mathbf{u}_2$ also satisfies the equation:

$$\begin{array}{rl} & \dfrac{d\mathbf{u}_1}{dt} = A\mathbf{u}_1 \\[2ex] + & \dfrac{d\mathbf{u}_2}{dt} = A\mathbf{u}_2 \\ \hline & \dfrac{d}{dt}(\mathbf{u}_1 + \mathbf{u}_2) = A(\mathbf{u}_1 + \mathbf{u}_2) \\[2ex] & \dfrac{d\mathbf{u}}{dt} = A\mathbf{u} \end{array}$$

- Take $\mathbf{u}_1 = c_1 e^{\lambda_1 t}\mathbf{x}_1$:

$$\begin{aligned} \frac{d\mathbf{u}_1}{dt} &= c_1\lambda_1 e^{\lambda_1 t}\mathbf{x}_1 \\ A\mathbf{u}_1 &= A\left(c_1 e^{\lambda_1 t}\mathbf{x}_1\right) \\ &= c_1 e^{\lambda_1 t} A\mathbf{x}_1 \\ &= c_1 e^{\lambda_1 t}\lambda_1\mathbf{x}_1 \\ &= \frac{d\mathbf{u}_1}{dt}. \end{aligned}$$

- Similarly, for $\mathbf{u}_2 = c_2 e^{\lambda_2 t}\mathbf{x}_2$: $\quad \dfrac{d\mathbf{u}_2}{dt} = A\mathbf{u}_2.$

- Thus,

$$\begin{aligned} \frac{d\mathbf{u}}{dt} &= \frac{d}{dt}(\mathbf{u}_1 + \mathbf{u}_2) = A(\mathbf{u}_1 + \mathbf{u}_2) \\ \mathbf{u} &= c_1 e^{\lambda_1 t}\mathbf{x}_1 + c_2 e^{\lambda_2 t}\mathbf{x}_2. \end{aligned}$$

- The only remaining part is to find the coefficients c_1 and c_2 such that $\mathbf{u} = \mathbf{u}(0)$ at time $t = 0$.

- This initial condition yields a 2×2 system,

$$\begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 \end{bmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 8 \\ 5 \end{pmatrix}.$$

- Solving for c_1 and c_2 via Gaussian elimination:

$$\begin{bmatrix} 1 & 5 \\ 1 & 2 \end{bmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 8 \\ 5 \end{pmatrix}$$

$$\begin{bmatrix} 1 & 5 \\ 0 & -3 \end{bmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 8 \\ -3 \end{pmatrix}$$

$$\begin{aligned} c_2 &= 1 \\ c_1 &= 8 - 5c_1 = 3. \end{aligned}$$

- So, our solution is $$\begin{aligned} \mathbf{u}(t) &= \mathbf{x}_1 c_1 e^{\lambda_1 t} + \mathbf{x}_2 c_2 e^{\lambda_2 t} \\ &= \begin{pmatrix} 1 \\ 1 \end{pmatrix} 3e^{-t} + \begin{pmatrix} 5 \\ 2 \end{pmatrix} e^{2t}. \end{aligned}$$

- Clearly, after a long time, the solution is going to look like a multiple of $\mathbf{x}_2 = [5\ 2]^T$ because the component of the solution parallel to $\mathbf{x}_1$ will decay.

- (More precisely, the component parallel to $\mathbf{x}_1$ will not grow as fast as the component parallel to $\mathbf{x}_2$.)

Example Summary

- Model problem, $\mathbf{u} \in \mathcal{R}^n$,

$$\frac{d\mathbf{u}}{dt} = A\mathbf{u}, \quad \mathbf{u} = \mathbf{u}(0) \text{ at time } t = 0.$$

- Assuming A has n *linearly independent* eigenvectors, can express

$$\mathbf{u}(t) = \sum_{j=1}^{n} \mathbf{x}_j c_j e^{\lambda_j t}.$$

- Coefficients c_j determined by initial condition:

$$X\mathbf{c} = \sum_{j=1}^{n} \mathbf{x}_j c_j = \mathbf{u}(0) \iff \mathbf{c} = X^{-1}\mathbf{u}(0).$$

- Eigenpairs $(\lambda_j, \mathbf{x}_j)$ satisfy

$$A\mathbf{x}_j = \lambda_j \mathbf{x}_j.$$

Growing / Decaying Modes

- Our model problem,

$$\frac{d\mathbf{u}}{dt} = A\mathbf{u} \longrightarrow \mathbf{u}(t) = \mathbf{x}_1 c_1 e^{\lambda_1 t} + \mathbf{x}_2 c_2 e^{\lambda_2 t}$$

  leads to growth/decay of components.

- Also get growth/decay through matrix-vector products.
- Consider $\mathbf{u} = c_1\mathbf{x}_1 + c_2\mathbf{x}_2$.

$$\begin{aligned} A\mathbf{u} &= c_1 A\mathbf{x}_1 + c_2 A\mathbf{x}_2 \\ &= c_1\lambda_1\mathbf{x}_1 + c_2\lambda_2\mathbf{x}_2 \end{aligned}$$

$$\begin{aligned} A^k\mathbf{u} &= c_1\lambda_1^k\mathbf{x}_1 + c_2\lambda_2^k\mathbf{x}_2 \\ &= \lambda_2^k \left[c_1 \left(\frac{\lambda_1}{\lambda_2}\right)^k \mathbf{x}_1 + c_2\mathbf{x}_2 \right]. \end{aligned}$$

$$\lim_{k\longrightarrow\infty} A^k\mathbf{u} = \lambda_2^k \left[c_1 \cdot 0 \cdot \mathbf{x}_1 + c_2\mathbf{x}_2\right] = c_2\lambda_2^k\mathbf{x}_2.$$

- So, repeated matrix-vector products lead to emergence of eigenvector associated with the eigenvalue λ that has largest modulus.
- This is the main idea behind the ***power method***, which is a common way to find the eigenvector associated with $\max|\lambda|$.

Characteristic Polynomial

- Equation $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ is equivalent to

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$$

  which has nonzero solution $\mathbf{x}$ *iff* matrix $(\mathbf{A} - \lambda\mathbf{I})$ is singular

- Eigenvalues of $\mathbf{A}$ are roots λ_i of *characteristic polynomial*

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

  of degree n in λ

- *Fundamental Theorem of Algebra* implies that $n \times n$ matrix $\mathbf{A}$ always has n eigenvalues, but they need not be real nor distinct

- Complex eigenvalues of real matrix occur in complex conjugate pairs: If $\lambda = \alpha + i\beta$ is an eigenvalue of a real matrix then so is $\alpha - i\beta$, where $i = \sqrt{-1}$

Example: Characteristic Polynomial

- Evaluate $\det(\mathbf{A} - \lambda\mathbf{I})$ of earlier example

$$
\begin{aligned}
|\mathbf{A} - \lambda\mathbf{I}| &= \det\left(\begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \\
&= \det\left(\begin{bmatrix} 3-\lambda & -1 \\ -1 & 3-\lambda \end{bmatrix}\right) \\
&= (3-\lambda)(3-\lambda) - (-1)(-1) = \lambda^2 - 6\lambda + 8 = 0
\end{aligned}
$$

- Eigenvalues are

$$
\lambda = \frac{6 \pm \sqrt{36 - 32}}{2}, \quad \text{or} \quad \lambda_1 = 2, \quad \lambda_2 = 4
$$

Companion Matrix

- Monic polynomial

$$p(\lambda) = c_0 + c_1\lambda + \cdots + c_{n-1}\lambda^{n-1} + \lambda^n$$

is characteristic polynomial of *companion matrix*

$$\mathbf{C}_n = \begin{bmatrix} 0 & 0 & \cdots & 0 & -c_0 \\ 1 & 0 & \cdots & 0 & -c_1 \\ 0 & 1 & \cdots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -c_{n-1} \end{bmatrix}$$

- Roots of polynomial degree > 4 cannot always be computed in finite number of steps

- So in general, computation of eigenvalues of matrices of order > 4 requires a (theoretically infinite) iterative process

Example: Companion Matrix, $n = 3$

- Consider companion matrix

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & -c_0 \\ 1 & 0 & -c_1 \\ 0 & 1 & -c_2 \end{bmatrix}$$

- Evaluate determinant of $\mathbf{C} - \lambda\mathbf{I}$

$$|\mathbf{C} - \lambda\mathbf{I}| = \begin{vmatrix} -\lambda & 0 & -c_0 \\ 1 & -\lambda & -c_1 \\ 0 & 1 & -(c_2+\lambda) \end{vmatrix}$$

$$= -c_0 \begin{vmatrix} 1 & -\lambda \\ 0 & 1 \end{vmatrix} + c_1 \begin{vmatrix} -\lambda & 0 \\ 0 & 1 \end{vmatrix} - (c_2+\lambda) \begin{vmatrix} -\lambda & 0 \\ 1 & -\lambda \end{vmatrix}$$

$$= -c_0 - c_1\lambda - c_2\lambda^2 - \lambda^3 = 0$$

- Roots of resultant monic polynomial, $p(\lambda) = c_0 + c_1\lambda + c_2\lambda^2 + \lambda^3 = 0$, are the 3 eigenvalues, λ_1, λ_2, and λ_3

Characteristic Polynomial, continued

- Computing eigenvalues using characteristic polynomial is *not recommended* because of
  - work in computing coefficients of characteristic polynomial
  - sensitivity of coefficients of characteristic polynomial
  - work in solving for roots of characteristic polynomial
- Characteristic polynomial is a powerful theoretical tool but usually not useful computationally
- In fact, in many cases we use eigenvalue solvers to find the roots of poynomials

Example: Characteristic Polynomial

- Consider $\mathbf{A} = \begin{bmatrix} 1 & \epsilon \\ \epsilon & 1 \end{bmatrix}$

  with $\epsilon_M < \epsilon < \sqrt{\epsilon_M}$

- Exact eigenvalues of $\mathbf{A}$ are $1 + \epsilon$ and $1 - \epsilon$

- Computing characteristic polynomial in float point arithmetic leads to

$$\det(\mathbf{A} - \lambda \mathbf{I}) = \lambda^2 - 2\lambda + (1 - \epsilon^2) = \lambda^2 - 2\lambda + 1$$

  which as 1 as a double root

- Thus, eigenvalues cannot be resolved by this method even though they are distinct to working precision

Multiplicity and Diagonalizability

- *Multiplicity* is number of times root appears when polynomial is written as product of linear factors (e.g., $(1-\lambda)^3(2-\lambda)^2(5-\lambda)$) ***Algebraic Multiplicity***

- Eigenvalue with multiplicity 1 is *simple*

- *Defective* matrix has eigenvalue of multiplicity $k > 1$ with fewer than k linearly independent corresponding eigenvectors ***Geometric Multiplicity***

- Nondefective matrix $\mathbf{A}$ has n linearly independent eigenvectors, so it is *diagonalizable*

$$\mathbf{X}^{-1}\mathbf{A}\mathbf{X} = \mathbf{D}$$

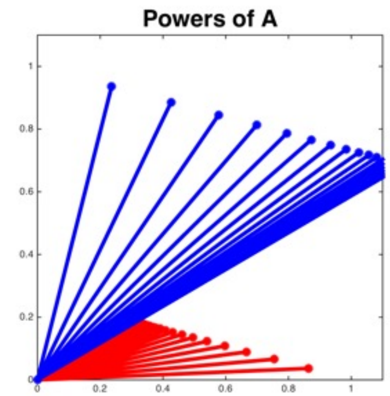
where $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \cdots \mathbf{x}_n]$ is nonsingular matrix of eigenvectors

- Note: every matrix is ϵ away from being diagonalizable

Diagonalization

- The real merit of eigenvalue decomposition is that it simplifies powers of a matrix.
- Consider

$$
\begin{aligned}
X^{-1}AX &= D, \ \textit{diagonal} \\
AX &= XD \\
A &= XDX^{-1} \\
A^2 &= \left(XDX^{-1}\right)\left(XDX^{-1}\right) \\
&= XD^2X^{-1} \\
A^k &= \left(XDX^{-1}\right)\left(XDX^{-1}\right)\cdots\left(XDX^{-1}\right) \\
&= XD^kX^{-1} \\
&= X \begin{bmatrix} \lambda_1^k & & & \\ & \lambda_2^k & & \\ & & \ddots & \\ & & & \lambda_n^k \end{bmatrix} X^{-1}
\end{aligned}
$$



pow_a.m

- High powers of A tend to be dominated by largest eigenpair $(\lambda_1, \underline{x}_1)$, assuming $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$.

Matrix Powers Example

- Consider our 1D finite difference example introduced earlier.

$$-\frac{d^2u}{dx^2} \;=\; f(x) \;\longrightarrow\; -\frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} \;\approx\; f(x_i).$$

where $u(0) = u(1) = 0$ and $\Delta x = 1/(n+1)$.

- In matrix form,

$$A\mathbf{u} = \frac{1}{\Delta x^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_m \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_m \end{pmatrix}$$

- Eigenvectors and eigenvalues have closed-form expression:

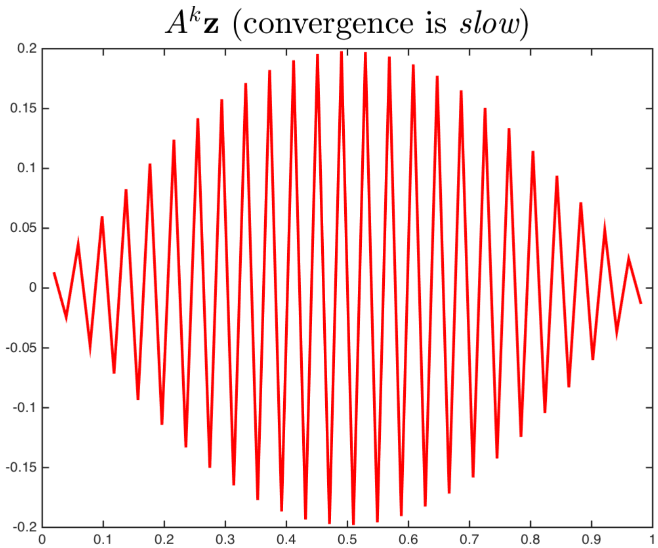
$$(\mathbf{z}_k)_i \;=\; \sin k\pi x_i \;=\; \sin k\pi i\Delta x \qquad \lambda_k \;=\; \frac{2}{\Delta x^2}\,(1 - \cos k\pi\Delta x)$$

- Eigenvalues are in the interval $\sim\ [\pi^2,\ 4(n+1)^2]$.

Matlab Example: heat_demo.m

- Repeatedly applying A to a random input vector reveals the eigenvalue of maximum modulus.

- This idea leads to one of the most common (but not most efficient) ways of finding an eigenvalue/vector pair, called the power method.




```
hdr

n = 50;
h = 1/(n+1);
e = ones(n,1);
A = spdiags([-e 2*e -e],-1:1, n,n)/(h*h);
x = 1:n; x=h*x';

z=sin(pi*x);
for k=1:3000;
     z=A*z; z=z/norm(z);
     plot(x,z,'r-',lw,2);
     title('$A^k {\bf z}$ (convergence is {\em slow})',intp,ltx,fs,24);
     drawnow;
     k
end;
```

```
hdr

n=100;


h = 1/(n+1); e = ones(n,1); A = spdiags([-e 2*e -e],-1:1, n,n)/(h*h);

x=1:n; x=h*x';

z=rand(n,1); hold off;
z=sin(pi*x);
for k=1:2000;
    z=A*z; z=z/norm(z);
    plot(x,z,'r-',lw,2); pause;
    k
end;
if -min(z) > max(z); z=-z; end; plot(x,z,'r-'); pause;

[Z,D]=eig(full(A)); zn=Z(:,n);


hold on
zn=zn/norm(zn); if -min(zn) > max(zn); zn=-zn; end;
plot(x,zn,'kx')
sn = sin(n*pi*x); sn=sn/norm(sn);
plot(x,sn,'go')
```

Diagonalization

- Note that if we define $A^0 = I$, we have any polynomial of A defined as

$$p_k(A)\underline{x} = X \begin{bmatrix} p_k(\lambda_1) & & & \\ & p_k(\lambda_2) & & \\ & & \ddots & \\ & & & p_k(\lambda_n) \end{bmatrix} X^{-1}\underline{x}.$$

- We can further extend this to other functions,

$$f(A)\underline{x} = X \begin{bmatrix} f(\lambda_1) & & & \\ & f(\lambda_2) & & \\ & & \ddots & \\ & & & f(\lambda_n) \end{bmatrix} X^{-1}\underline{x}.$$

- For example, the solution to $f(A)\underline{x} = \underline{b}$ is would be

$$\underline{x} = X\left[f(D)\right]^{-1} X^{-1}\underline{b}.$$

- The diagonalization concept is very powerful because it transforms *systems* of equations into scalar equations.

Stopped Here

Eigenspaces and Invariant Subspaces

- Eigenvectors can be scaled arbitrarily: if $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, then $\mathbf{A}(\gamma\mathbf{x}) = \lambda(\gamma\mathbf{x})$ for any scalar γ, so $\gamma\mathbf{x}$ is also eigenvector corresponding to λ

- Eigenvectors are usually *normalized* by requiring some norm of eigenvector to be 1 (2-norm is most favored...)

- *Eigenspace* $= \mathcal{S}_\lambda = \{\mathbf{x} : \mathbf{A}\mathbf{x} = \lambda\mathbf{x}\}$

- Subspace $\mathcal{S}$ of $\mathbb{R}^n$ (or $\mathbb{C}^n$) is *invariant* if $\mathbf{A}\mathcal{S} \subseteq \mathcal{S}$

- For eigenvectors $\mathbf{x}_1 \cdots \mathbf{x}_p$ span$([\mathbf{x}_1 \cdots \mathbf{x}_p])$ is invariant subspace

- **Q:** When might invariance fail?

  A: In floating-point arithmetic, because of round-off error

Relevant Properties of Matrices

- Properties of matrix $\mathbf{A}$ relevant to eigenvalue problems

Property	Definition
diagonal	$a_{ij} = 0$ for $i \neq j$
tridiagonal	$a_{ij} = 0$ for $\|i - j\| > 1$
triangular	$a_{ij} = 0$ for $i > j$ (upper)
	$a_{ij} = 0$ for $i < j$ (lower)
Hessenberg	$a_{ij} = 0$ for $i > j + 1$ (upper)
	$a_{ij} = 0$ for $i < j - 1$ (lower)
orthogonal	$\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I}$
unitary	$\mathbf{A}^H\mathbf{A} = \mathbf{A}\mathbf{A}^H = \mathbf{I}$ $(\mathbf{A} \in \mathbb{C}^{n\times n})$
symmetric	$\mathbf{A} = \mathbf{A}^T$
skew-symmetric	$\mathbf{A} = -\mathbf{A}^T$
Hermitian	$\mathbf{A} = \mathbf{A}^H$
normal	$\mathbf{A} = \mathbf{A}^H$
normal	$\mathbf{A}^H\mathbf{A} = \mathbf{A}\mathbf{A}^H$

Upper Hessenberg (from Chap03…)

- A is upper Hessenberg – *A is upper triangular with one additional nonzero diagonal below the main one:* $A_{ij} = 0 \text{ if } i > j+1$

0.1967	0.2973	0.0899	0.3381	0.5261	0.3965	0.1279
0.0934	0.0620	0.0809	0.2940	0.7297	0.0616	0.5495
0	0.2982	0.7772	0.7463	0.7073	0.7802	0.4852
0	0	0.9051	0.0103	0.7814	0.3376	0.8905
0	0	0	0.0484	0.2880	0.6079	0.7990
0	0	0	0	0.6925	0.7413	0.7343
0	0	0	0	0	0.1048	0.0513

- Requires only n Givens rotations, instead of $O(n^2)$, to effect QR factorization.

Examples: Matrix Properties

- Transpose $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$

- Conjugate transpose $\begin{bmatrix} 1+i & 1+2i \\ 2-i & 2-2i \end{bmatrix}^H = \begin{bmatrix} 1-i & 2+i \\ 1-2i & 2+2i \end{bmatrix}$

- Symmetric: $\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$ $\quad (\mathbf{A} = \mathbf{A}^T)$

- Skew-Symmetric: $\begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix}^T = -\begin{bmatrix} 0 & -2 \\ 2 & 0 \end{bmatrix}$, $\quad (\mathbf{A} = -\mathbf{A}^T)$

Examples, continued

- Nonsymmetric: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

- Hermitian $\begin{bmatrix} 1 & 1+i \\ 1-i & 2 \end{bmatrix}$

- NonHermitian $\begin{bmatrix} 1 & 1+i \\ 1+i & 2 \end{bmatrix}$

Examples, continued

- Orthogonal: $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$, $\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$

- Unitary: $\begin{bmatrix} \frac{i\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & -i\frac{\sqrt{2}}{2} \end{bmatrix}$

- Nonorthogonal $\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$

- Normal $\begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 2 \\ 1 & 0 & 1 \end{bmatrix}$

- Nonnormal $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ ← *"canonical non-normal matrix"*
  Defective – has only one eigenvector.

Normal Matrices

Normal matrices have orthogonal eigenvectors, so $\underline{x}_i^H \underline{x}_j = \delta_{ij}$

$$X^T = X^{-1}$$

$$A = XDX^H$$

Normal matrices include

- symmetric ($A = A^T$)
- skew-symmetric ($A = -A^T$)
- unitary ($U^H U = I$)
- circulant (periodic+Toeplitz)
- others ...

$$\begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 2 \\ 2 & 0 & 1 \end{bmatrix}$$

Normal Matrices

Normal matrices have orthogonal eigenvectors, so $\underline{x}_i^H \underline{x}_j = \delta_{ij}$

Beware!

- If A is normal, it *has* orthogonal eigenvectors.
- That does not mean that all eigensolvers will *return* orthogonal eigenvectors.
- In particular, if two or more eigenvectors share the same eigenvalue, then they needn't be orthogonal to each other.
- You probably need to orthogonalize them yourself.

Normal ma

- symme
- skew-symmetric ($A = -A^T$)
- unitary ($U^H U = I$)
- circulant (periodic+Toeplitz)
- others ...

$$\begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 2 \\ 2 & 0 & 1 \end{bmatrix}$$

Properties of Eigenvalue Problems

Properties of eigenvalue problem affecting choice of algorithm and software

- Are all eigenvalues needed, or only a few?
- Are only eigenvalues needed, or are corresponding eigenvectors also needed?
- Is the matrix real or complex?
- Is the matrix relatively small and dense, or large and sparse?
- Does the matrix have any special properties such as symmetry, or is it a general matrix?

Sparsity

- Sparsity, either direct or implied, is a big driver in choice of eigenvalue solvers.
- Typically, only O(n) entries in entire matrix, where $n \sim 10^9$—10^{18} might be anticipated.
- Examples include Big Data (e.g., google page rank) and physics simulations (fluid, heat transfer, electromagnetics, fusion, etc.).
- Usually, need only a few ($k \ll n$) eigenvectors / eigenvalues.
- Often, there are special properties of A that make it difficult to create A. Instead, work strictly with matrix-vector products

$$\boldsymbol{y} = \boldsymbol{A}\,\boldsymbol{x}$$

Conditioning of Eigenvalue Problems

- Condition of eigenvalue problem is sensitivity of eigenvalues and eigenvectors to changes in matrix

- Condition of eigenvalue problem is *not* same as conditioning of solution to linear system for same matrix

  - Finding $\lambda = 0$ is a common situation in eigenvalue problems, but indicates a singularity when trying to solve $\mathbf{Ax} = \mathbf{b}$ sensitivity of coefficients of characteristic polynomial

- Different eigenvalues and eigenvectors are not necessarily equally sensitive to perturbations in matrix

Conditioning of Eigenvalues

- If μ is eigenvalue of $\mathbf{A}+\mathbf{E}$ of nondefective matrix $\mathbf{A}$, then

$$|\mu - \lambda_k| \;\leq\; \text{cond}_2(\mathbf{X})\|\mathbf{E}\|_2$$

  where λ_k is closest eigenvalue of $\mathbf{A}$ to μ and $\mathbf{X}$ is the nonsingular matrix of eigenvectors of $\mathbf{A}$

- Absolute condition number of eigenvalues is condition number of matrix of eigenvectors with respect so solving linear equations (e.g., $\mathbf{Xc} = \mathbf{b}$)

- Eigenvalues may be sensitive if eigenvectors are nearly linearly dependent (i.e., matrix is nearly defective)

- For *normal* matrix ($\mathbf{A}\mathbf{A}^H = \mathbf{A}^H\mathbf{A}$), eigenvectors are orthogonal, so eigenvalues are well-conditioned

Conditioning of Eigenvalues

- If $(\mathbf{A}+\mathbf{E})(\mathbf{x}+\Delta\mathbf{x}) = (\lambda+\Delta\lambda)(\mathbf{x}+\Delta\mathbf{x})$, where λ is a simple eigenvalue of $\mathbf{A}$, then

$$|\Delta\lambda| \lesssim \frac{\|\mathbf{y}\|_2 \cdot \|\mathbf{x}\|_2}{|\mathbf{y}^H\mathbf{x}|}\|\mathbf{E}\|_2 = \frac{1}{\cos\theta}\|\mathbf{E}\|_2$$

  where $\mathbf{x}$ and $\mathbf{y}$ are corresponding right and left eigenvectors and θ is the angle between them

- For symmetric or Hermitian matrix right and left eigenvectors are same so $\cos\theta = 1$ and eigenvalues are inherently well-conditioned

- Eigenvalues of nonnormal matrices may be sensitive

- For multiple or closely clustered eigenvalues, corresponding eigenvectors may be sensitive

Problem Transformations

- *Shift*: If $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ and σ is any scalar, then $(\mathbf{A} - \sigma\mathbf{I})\mathbf{x} = (\lambda - \sigma)\mathbf{x}$, so eigenvalues of shifted matrix are shifted eigenvalues of $\mathbf{A}$

- *Inversion*: If $\mathbf{A}$ is nonsingular and $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ with $\mathbf{x} \neq \mathbf{0}$, then $\lambda \neq 0$ and $\mathbf{A}^{-1}\mathbf{x} = (1/\lambda)\mathbf{x}$, so eigenvalues of inverse are reciprocals of $\lambda(\mathbf{A})$

- *Powers*: If $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, then $\mathbf{A}^k\mathbf{x} = \lambda^k\mathbf{x}$, so eigenvalues of power of matrix are λ^k

- *Polynomial*: If $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, and $p(t)$ is a polynomial, then $p(\mathbf{A})\mathbf{x} = p(\lambda)\mathbf{x}$, so eigenvalues of polynomial in $\mathbf{A}$ are $p(\lambda)$.

Similarity Transformation

- $\mathbf{B}$ is *similar* to $\mathbf{A}$ if there exists a nonsingular matrix $\mathbf{T}$ such that

$$\mathbf{B} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}$$

- Then,

$$\mathbf{B}\mathbf{y} = \lambda\mathbf{y} \implies \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\mathbf{y} = \lambda\mathbf{y} \implies \mathbf{A}\mathbf{T}\mathbf{y} = \lambda\mathbf{T}\mathbf{y}$$

  so $\mathbf{A}$ and $\mathbf{B}$ have the same eigenvalues, and if $\mathbf{y}$ is eigenvector of $\mathbf{B}$, then $\mathbf{x} = \mathbf{T}\mathbf{y}$ is eigenvector of $\mathbf{A}$

- Similarity transformations preserve eigenvalues and eigenvectors are easily recovered

Example: Similarity Transformation

- From eigenvalues and eigenvectors for previous example,

$$\begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$$

  and hence

$$\begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$$

- So original matrix is similar to diagonal matrix, and eigenvectors form columns of similarity transformation matrix

Diagonal Form

- Eigenvalues of diagonal matrix are diagonal entries, eigenvectors are $\mathbf{X} = \mathbf{I}$
- Diagonal form is desirable in simplifying eigenvalue problems for general matrices by similarity transformations
- But not all matrices are diagonalizable
- Closest one can get, in general, is Jordan form, which is nearly diagonal but may have som nonzero entries on first superdiagonal corresponging to one or more multiple eigenvalues

Simple non-diagonalizable example, 2 x 2 Jordan block:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\begin{vmatrix} 1-\lambda & 1 \\ 0 & 1-\lambda \end{vmatrix} = (1-\lambda)^2 = 0$$

Only one eigenvector: $\underline{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

3×3 **Non-Diagonalizable Example**

$$A = \begin{bmatrix} 2 & & \\ & 2 & \\ & & 2 \end{bmatrix}, \qquad B = \begin{bmatrix} 2 & 1 & \\ & 2 & 1 \\ & & 2 \end{bmatrix}.$$

- Characteristic polynomial is $(\lambda - 2)^3$ for both A and B.
- Algebraic multiplicity is 3.
- For A, three eigenvectors. Say, $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$.
- For B, only one eigenvector ($\alpha\mathbf{e}_1$), so geometric multiplicity of B is 1.

Triangular Form

- Every matrix can be transformed into triangular, *Schur*, form by similarity and diagonal entries of triangular matrix are the eigenvalues

- Eigenvectors are less obvious but straightforward to compute

- If

$$\mathbf{A} - \lambda\mathbf{I} = \begin{bmatrix} \mathbf{U}_{11} & \mathbf{u} & \mathbf{U}_{13} \\ \mathbf{0} & 0 & \mathbf{v}^T \\ \mathbf{O} & \mathbf{0} & \mathbf{U}_{33} \end{bmatrix}$$

is triangular, then $\mathbf{U}_{11}\mathbf{y} = \mathbf{u}$ can be solved for $\mathbf{y}$ so that

$$\mathbf{x} = \begin{bmatrix} \mathbf{y} \\ -1 \\ \mathbf{0} \end{bmatrix}$$

is corresponding eigenvector

Eigenvectors / Eigenvalues of Upper Triangular Matrix

- Suppose $\mathbf{A}$ is upper triangular

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{u} & \mathbf{U}_{13} \\ 0 & \lambda & \mathbf{v}^T \\ O & 0 & \mathbf{A}_{33} \end{bmatrix}$$

- Then

$$0 = (\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \underbrace{\begin{bmatrix} \mathbf{U}_{11} & \mathbf{u} & \mathbf{U}_{13} \\ \mathbf{0} & 0 & \mathbf{v}^T \\ \mathbf{O} & \mathbf{0} & \mathbf{U}_{33} \end{bmatrix}}_{(\mathbf{A} - \lambda\mathbf{I})} \underbrace{\begin{bmatrix} \mathbf{y} \\ -1 \\ 0 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{U}_{11}\mathbf{y} - \mathbf{u} \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}}$$

- Because $\mathbf{U}_{11}$ is nonsingular, can solve $\mathbf{U}_{11}\mathbf{y} = \mathbf{u}$ to find eigenvector $\mathbf{x}$.

Block Triangular Form

- If

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1p} \\ & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2p} \\ & & \ddots & \vdots \\ & & & \mathbf{A}_{pp} \end{bmatrix}$$

  with *square* diagonal blocks, then

$$\lambda(\mathbf{A}) = \bigcup_{j=1}^{p} \lambda(\mathbf{A}_{jj}),$$

  so eigenvalue problem breaks into p smaller eigenvalue problems

- *Real* Schur form has 1×1 diagonal blocks corresponding to real eigenvalues and 2×2 diagonal blocks corresponding to complex conjugate eigenvalue pairs

Similarity Transformations

- Given
$$\begin{aligned} B &= T^{-1}\,A\,T \\ A &= T\,B\,T^{-1} \end{aligned}$$
- If A is normal ($AA^H = A^H A$),
$$A = Q\,\Lambda\,Q^H$$
  B is diagonal, T is unitary ($T^{-1} = T^H$).
- If A is symmetric real,
$$A = Q\,\Lambda\,Q^T$$
  B is diagonal, T is orthogonal ($T^{-1} = T^T$).
- If B is diagonal, T is the matrix of eigenvectors.

Forms Attainable by Similarity: $\mathbf{B} = \mathbf{T}^{-1}\mathbf{AT}$

A	**T**	**B**
distinct eigenvalues	nonsingular	diagonal
real symmetric	orthogonal	real diagonal
complex Hermitian	unitary	real diagonal
normal	unitary	diagonal
arbitrary real	orthogonal	real block triangular (real Schur)
arbitrary	unitary	upper triangular (Schur)
arbitrary	nonsingular	almost diagonal (Jordan)

Always exists

- Given matrix $\mathbf{A}$ with indicated property, $\mathbf{B}$ and $\mathbf{T}$ exist with indicated properties such that $\mathbf{B} = \mathbf{T}^{-1}\mathbf{AT}$
- if $\mathbf{B}$ is diagonal or triangular, eigenvalues are its diagonal entries
- if $\mathbf{B}$ is diagonal, eigenvectors are columns of $\mathbf{T}$

Computing Eigenpairs via Various (Sophisticated!) Forms of Power Iteration

Power Iteration

- Simplest method for computing one eigenvalue-eigenvector pair is *power iteration*, which repeatedly multiplies matrix times initial starting vector

- Assume $\mathbf{A}$ has unique eigenvalue of maximum modulus, say λ_1, with corresponding eigenvector $\mathbf{v}_1$

- Starting from nonzero vector $\mathbf{x}_0$, iteration scheme

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1}$$

  converges to multiple of eigenvector $\mathbf{v}_1$ corresponding to *dominant* eigenvalue λ_1

```
format shorte

D=eye(3); D(1,1) = 4; D(2,2) = 3; D(3,3) = 2;
V=[ 3  4  2 ;
    4  3  2 ;
    0  0  1];

A = V*D*inv(V)

% Power iteration

x=[ 1 ; 1 ; 1 ];

disp(['       k            x1            x2            x3          x1/x1_old']);
disp(['-----------------------------------------------------------------']);

for k=1:20;
    xo = x;
    x=A*x;
    lambda = x(1)/xo(1);
    disp([k x' lambda])
end;
```

k	x1	x2	x3	x1/x1_old
1.0000e+00	5.7143e-01	4.2857e-01	2.0000e+00	5.7143e-01
2.0000e+00	-4.0000e+00	-5.0000e+00	4.0000e+00	-7.0000e+00
3.0000e+00	-2.6857e+01	-3.2143e+01	8.0000e+00	6.7143e+00
4.0000e+00	-1.2400e+02	-1.4900e+02	1.6000e+01	4.6170e+00
5.0000e+00	-5.1371e+02	-6.2529e+02	3.2000e+01	4.1429e+00
6.0000e+00	-2.0440e+03	-2.5250e+03	6.4000e+01	3.9789e+00
7.0000e+00	-8.0154e+03	-1.0044e+04	1.2800e+02	3.9214e+00
8.0000e+00	-3.1324e+04	-3.9749e+04	2.5600e+02	3.9080e+00
9.0000e+00	-1.2257e+05	-1.5721e+05	5.1200e+02	3.9130e+00
1.0000e+01	-4.8108e+05	-6.2244e+05	1.0240e+03	3.9249e+00
1.1000e+01	-1.8947e+06	-2.4686e+06	2.0480e+03	3.9384e+00
1.2000e+01	-7.4857e+06	-9.8065e+06	4.0960e+03	3.9509e+00
1.3000e+01	-2.9656e+07	-3.9015e+07	8.1920e+03	3.9616e+00
1.4000e+01	-1.1774e+08	-1.5541e+08	1.6384e+04	3.9704e+00
1.5000e+01	-4.6831e+08	-6.1965e+08	3.2768e+04	3.9773e+00
1.6000e+01	-1.8652e+09	-2.4726e+09	6.5536e+04	3.9828e+00
1.7000e+01	-7.4363e+09	-9.8722e+09	1.3107e+05	3.9870e+00
1.8000e+01	-2.9672e+10	-3.9434e+10	2.6214e+05	3.9901e+00
1.9000e+01	-1.1847e+11	-1.5757e+11	5.2429e+05	3.9926e+00
2.0000e+01	-4.7321e+11	-6.2978e+11	1.0486e+06	3.9944e+00

Convergence of Power Iteration

- To see why power iteration converges to dominant eigenvector, express starting vector $\mathbf{x}_0$ as linear combination Starting from nonzero vector $\mathbf{x}_0$, iteration scheme

$$\mathbf{x}_0 \;=\; \sum_{j=1}^{n} c_j \mathbf{v}_j$$

  where $\mathbf{v}_j$ are eigenvectors of $\mathbf{A}$

- Then

$$\begin{aligned}\mathbf{x}_k \;&=\; \mathbf{A}\mathbf{x}_{k-1} \;=\; \mathbf{A}^2\mathbf{x}_{k-2} \;=\; \cdots \;=\; bA^k\mathbf{x}_0 \\ &=\; \sum_{j=1}^{n} \lambda_j^k c_j \mathbf{v}_j \;=\; \lambda_1^k \left[c_1\mathbf{v}_1 + \sum_{j=2}^{n} \left(\frac{\lambda_j^k}{\lambda_1} \right)^k c_j \mathbf{v}_j \right]\end{aligned}$$

- Because $|\lambda_j/\lambda_1| < 1$ for $j > 1$, successively higher powers go to zero, leaving only $\mathbf{v}_1$ component

2 x 2 Example

$$A = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$

$$D = \begin{bmatrix} 1 & \\ & 2 \end{bmatrix} \qquad X = \begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

Example: Power Iteration

- Ratio of values of given component of $\boldsymbol{x}_k$ from one iteration to next converges to dominant eigenvalue λ_1
- For example, if $\boldsymbol{A} = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$ and $\boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, we obtain

k	$\boldsymbol{x}_k^T$		ratio
0	0.0	1.0	
1	0.5	1.5	1.500
2	1.5	2.5	1.667
3	3.5	4.5	1.800
4	7.5	8.5	1.889
5	15.5	16.5	1.941
6	31.5	32.5	1.970
7	63.5	64.5	1.985
8	127.5	128.5	1.992

- Ratio is converging to dominant eigenvalue, which is 2

Limitations of Power Iteration

- Power iteration can fail for various reasons

- Starting vector may have *no* component in dominant eigenvector (i.e., $c_1 = 0$). Not a problem in practice because rounding error usually introduces such a component in any case

- There may be more than one eigenvalue having same maximum modulus, in which case iteration may converge to linear combination of corresponding eigenvectors

- For real matrix and starting vector, iteration can never converge to a complex eigenvector

Normalized Power Iteration

- Geometric growth of components at each iteration risks eventual overflow (or underflow if $|\lambda_1| < 1$)

- Approximate eigenvector should be normalized at eath iteration, say, by requiring its largest component to be 1 in modulus, giving iteration scheme

$$\begin{aligned} \mathbf{y}_k &= \mathbf{A}\mathbf{x}_{k-1} \\ \mathbf{x}_k &= \mathbf{y}_k / \|\mathbf{y}_k\|_\infty \end{aligned}$$

- With normalization, $\|\mathbf{y}_k\|_\infty \longrightarrow |\lambda_1|$, and $\mathbf{x}_k \longrightarrow \mathbf{v}_1 / \|\mathbf{v}_1\|_\infty$

```
format shorte
D=eye(3); D(1,1) = 4; D(2,2) = 3; D(3,3) = 2;
V=[ 3  4  2 ;
    4  3  2 ;
    0  0  1];
A = V*D*inv(V)

disp(['      k           x1            x2            x3           x1/x1_old']);
disp(['-----------------------------------------------------------------------']);

x=[ 1 ; 1 ; 1 ];
for k=1:20;          % Normalized Power iteration
    y=A*x;
    abs_lambda = norm(y,Inf);
    x=y/abs_lambda;
    disp([k x' abs_lambda])
end;
```

k	x1	x2	x3	x1/x1_old
1.0000e+00	2.8571e-01	2.1429e-01	1.0000e+00	2.0000e+00
2.0000e+00	-8.0000e-01	-1.0000e+00	8.0000e-01	2.5000e+00
3.0000e+00	-8.3556e-01	-1.0000e+00	2.4889e-01	6.4286e+00
4.0000e+00	-8.3221e-01	-1.0000e+00	1.0738e-01	4.6356e+00
5.0000e+00	-8.2157e-01	-1.0000e+00	5.1177e-02	4.1965e+00
6.0000e+00	-8.0950e-01	-1.0000e+00	2.5347e-02	4.0382e+00
7.0000e+00	-7.9807e-01	-1.0000e+00	1.2744e-02	3.9777e+00
8.0000e+00	-7.8804e-01	-1.0000e+00	6.4404e-03	3.9577e+00
9.0000e+00	-7.7967e-01	-1.0000e+00	3.2568e-03	3.9550e+00
1.0000e+01	-7.7289e-01	-1.0000e+00	1.6451e-03	3.9594e+00
1.1000e+01	-7.6753e-01	-1.0000e+00	8.2963e-04	3.9659e+00
1.2000e+01	-7.6334e-01	-1.0000e+00	4.1768e-04	3.9726e+00
1.3000e+01	-7.6011e-01	-1.0000e+00	2.0997e-04	3.9784e+00
1.4000e+01	-7.5764e-01	-1.0000e+00	1.0543e-04	3.9833e+00
1.5000e+01	-7.5576e-01	-1.0000e+00	5.2881e-05	3.9872e+00
1.6000e+01	-7.5434e-01	-1.0000e+00	2.6505e-05	3.9903e+00
1.7000e+01	-7.5326e-01	-1.0000e+00	1.3277e-05	3.9926e+00
1.8000e+01	-7.5245e-01	-1.0000e+00	6.6477e-06	3.9944e+00
1.9000e+01	-7.5184e-01	-1.0000e+00	3.3273e-06	3.9958e+00
2.0000e+01	-7.5138e-01	-1.0000e+00	1.6650e-06	3.9969e+00

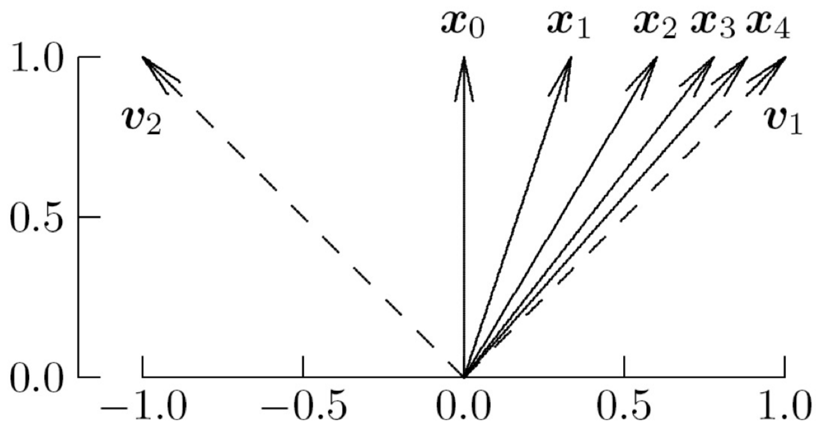
Example: Normalized Power Iteration

- Repeating previous example with normalized scheme,

k	$\boldsymbol{x}_k^T$		$\|\boldsymbol{y}_k\|_\infty$
0	0.000	1.0	
1	0.333	1.0	1.500
2	0.600	1.0	1.667
3	0.778	1.0	1.800
4	0.882	1.0	1.889
5	0.939	1.0	1.941
6	0.969	1.0	1.970
7	0.984	1.0	1.985
8	0.992	1.0	1.992

Geometric Interpretation

- Behavior of power iteration depicted geometrically



- Initial vector $\mathbf{x}_0 = \mathbf{v}_1 + \mathbf{v}_2$ contains equal components in $\mathbf{v}_1$ and $\mathbf{v}_2$ (dashed arrows)

- Repeated multiplication by $\mathbf{A}$ causes component in $\mathbf{v}_1$ (corresponding to larger eigenvalue 2) to dominate, so sequence of vectors $\mathbf{x}_k$ converges to $\mathbf{v}_1$

Convergence Rate of Power Iteration

Convergence rate of power iteration depends on relative separation of λ_1 and λ_2.

Assuming $c_1 \neq 0$ and $|\lambda_1| > |\lambda_j|$, $j > 1$, we have

$$\begin{aligned}
A^k \underline{x} &= \sum_{j=1}^{n} \underline{x}_j \, \lambda_j^k \, c_j \\
&= \lambda_1^k \, c_1 \left[\underline{x}_1 + \sum_{j=2}^{n} \underline{x}_j \, \frac{\lambda_j^k}{\lambda_1^k} \, \frac{c_j}{c_1} \right] \\
&\sim \underline{x}_1 \, \lambda_1^k \, c_1 \text{ as } k \longrightarrow \infty \\
&\sim \lambda_1^k \, c_1 \left[\underline{x}_1 + \underline{x}_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k \frac{c_2}{c_1} \right]
\end{aligned}$$

In preceding text examples, ratio is ½, so error is reduced by ½ with each iteration.

demo10/err_norm.m

```
format shorte
D=eye(3); D(1,1) = 4; D(2,2) = 3; D(3,3) = 2;
V=[ 3  4  2 ;
    4  3  2 ;
    0  0  1];
A = V*D*inv(V)

lam_max = 4;

disp(['       k           x1           x2           x3         x1/x1_old']);
disp(['----------------------------------------------------------------------']);

x=[ 1 ; 1 ; 1 ]; est=0;
for k=1:40;           % Normalized Power iteration
    y=A*x;
    abs_lambda = norm(y,Inf);
    x=y/abs_lambda;
    err = abs(4-abs_lambda);
    disp([k x' abs_lambda err est])
    est = 0.75*err;
end;
```

k	x1	x2	x3	x1/x1_old		
1.0000e+00	2.8571e-01	2.1429e-01	1.0000e+00	2.0000e+00	2.0000e+00	0
2.0000e+00	-8.0000e-01	-1.0000e+00	8.0000e-01	2.5000e+00	1.5000e+00	1.5000e+00
3.0000e+00	-8.3556e-01	-1.0000e+00	2.4889e-01	6.4286e+00	2.4286e+00	1.1250e+00
4.0000e+00	-8.3221e-01	-1.0000e+00	1.0738e-01	4.6356e+00	6.3556e-01	1.8214e+00
5.0000e+00	-8.2157e-01	-1.0000e+00	5.1177e-02	4.1965e+00	1.9655e-01	4.7667e-01
6.0000e+00	-8.0950e-01	-1.0000e+00	2.5347e-02	4.0382e+00	3.8154e-02	1.4741e-01
7.0000e+00	-7.9807e-01	-1.0000e+00	1.2744e-02	3.9777e+00	2.2348e-02	2.8615e-02
8.0000e+00	-7.8804e-01	-1.0000e+00	6.4404e-03	3.9577e+00	4.2344e-02	1.6761e-02
9.0000e+00	-7.7967e-01	-1.0000e+00	3.2568e-03	3.9550e+00	4.4979e-02	3.1758e-02
1.0000e+01	-7.7289e-01	-1.0000e+00	1.6451e-03	3.9594e+00	4.0631e-02	3.3734e-02
1.1000e+01	-7.6753e-01	-1.0000e+00	8.2963e-04	3.9659e+00	3.4076e-02	3.0473e-02
1.2000e+01	-7.6334e-01	-1.0000e+00	4.1768e-04	3.9726e+00	2.7436e-02	2.5557e-02
1.3000e+01	-7.6011e-01	-1.0000e+00	2.0997e-04	3.9784e+00	2.1555e-02	2.0577e-02
1.4000e+01	-7.5764e-01	-1.0000e+00	1.0543e-04	3.9833e+00	1.6673e-02	1.6166e-02
1.5000e+01	-7.5576e-01	-1.0000e+00	5.2881e-05	3.9872e+00	1.2768e-02	1.2505e-02
1.6000e+01	-7.5434e-01	-1.0000e+00	2.6505e-05	3.9903e+00	9.7127e-03	9.5762e-03
1.7000e+01	-7.5326e-01	-1.0000e+00	1.3277e-05	3.9926e+00	7.3552e-03	7.2845e-03
1.8000e+01	-7.5245e-01	-1.0000e+00	6.6477e-06	3.9944e+00	5.5531e-03	5.5164e-03
1.9000e+01	-7.5184e-01	-1.0000e+00	3.3273e-06	3.9958e+00	4.1839e-03	4.1649e-03
2.0000e+01	-7.5138e-01	-1.0000e+00	1.6650e-06	3.9969e+00	3.1479e-03	3.1380e-03
2.1000e+01	-7.5104e-01	-1.0000e+00	8.3298e-07	3.9976e+00	2.3661e-03	2.3609e-03
2.2000e+01	-7.5078e-01	-1.0000e+00	4.1668e-07	3.9982e+00	1.7773e-03	1.7746e-03
2.3000e+01	-7.5058e-01	-1.0000e+00	2.0841e-07	3.9987e+00	1.3344e-03	1.3330e-03
2.4000e+01	-7.5044e-01	-1.0000e+00	1.0423e-07	3.9990e+00	1.0016e-03	1.0008e-03
2.5000e+01	-7.5033e-01	-1.0000e+00	5.2125e-08	3.9992e+00	7.5156e-04	7.5116e-04
2.6000e+01	-7.5025e-01	-1.0000e+00	2.6066e-08	3.9994e+00	5.6388e-04	5.6367e-04
2.7000e+01	-7.5019e-01	-1.0000e+00	1.3034e-08	3.9996e+00	4.2302e-04	4.2291e-04
2.8000e+01	-7.5014e-01	-1.0000e+00	6.5177e-09	3.9997e+00	3.1733e-04	3.1727e-04
2.9000e+01	-7.5010e-01	-1.0000e+00	3.2590e-09	3.9998e+00	2.3803e-04	2.3799e-04
3.0000e+01	-7.5008e-01	-1.0000e+00	1.6296e-09	3.9998e+00	1.7854e-04	1.7852e-04
3.1000e+01	-7.5006e-01	-1.0000e+00	8.1483e-10	3.9999e+00	1.3391e-04	1.3390e-04
3.2000e+01	-7.5004e-01	-1.0000e+00	4.0742e-10	3.9999e+00	1.0044e-04	1.0043e-04
3.3000e+01	-7.5003e-01	-1.0000e+00	2.0372e-10	3.9999e+00	7.5332e-05	7.5329e-05
3.4000e+01	-7.5002e-01	-1.0000e+00	1.0186e-10	3.9999e+00	5.6500e-05	5.6499e-05
3.5000e+01	-7.5002e-01	-1.0000e+00	5.0930e-11	4.0000e+00	4.2376e-05	4.2375e-05
3.6000e+01	-7.5001e-01	-1.0000e+00	2.5465e-11	4.0000e+00	3.1783e-05	3.1782e-05
3.7000e+01	-7.5001e-01	-1.0000e+00	1.2733e-11	4.0000e+00	2.3837e-05	2.3837e-05
3.8000e+01	-7.5001e-01	-1.0000e+00	6.3664e-12	4.0000e+00	1.7878e-05	1.7878e-05
3.9000e+01	-7.5001e-01	-1.0000e+00	3.1832e-12	4.0000e+00	1.3409e-05	1.3409e-05
4.0000e+01	-7.5000e-01	-1.0000e+00	1.5916e-12	4.0000e+00	1.0056e-05	1.0056e-05

Power Iteration with Shift

- Convergence rate of power iteration depends on ratio $|\lambda_2/\lambda_1|$, where λ_2 is eigenvalue having second largest modulus

- May be possible to choose shift, $\mathbf{A} - \sigma\mathbf{I}$ such that

$$\left|\frac{\lambda_2 - \sigma}{\lambda_1 - \sigma}\right| < \left|\frac{\lambda_2}{\lambda_1}\right|$$

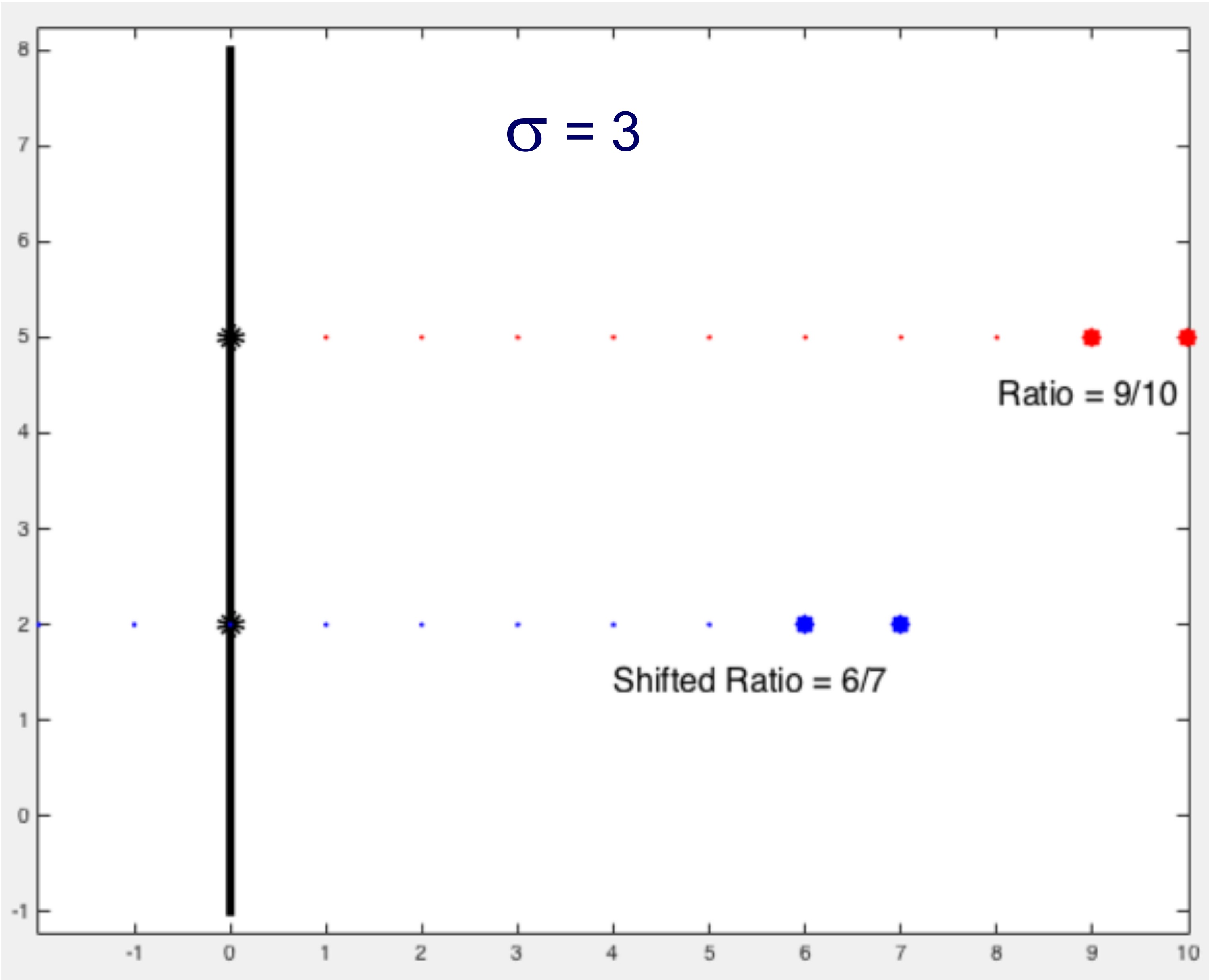
  so convergence is accelerated

- Shift σ must then be added to result to obtain eigenvalue of original matrix

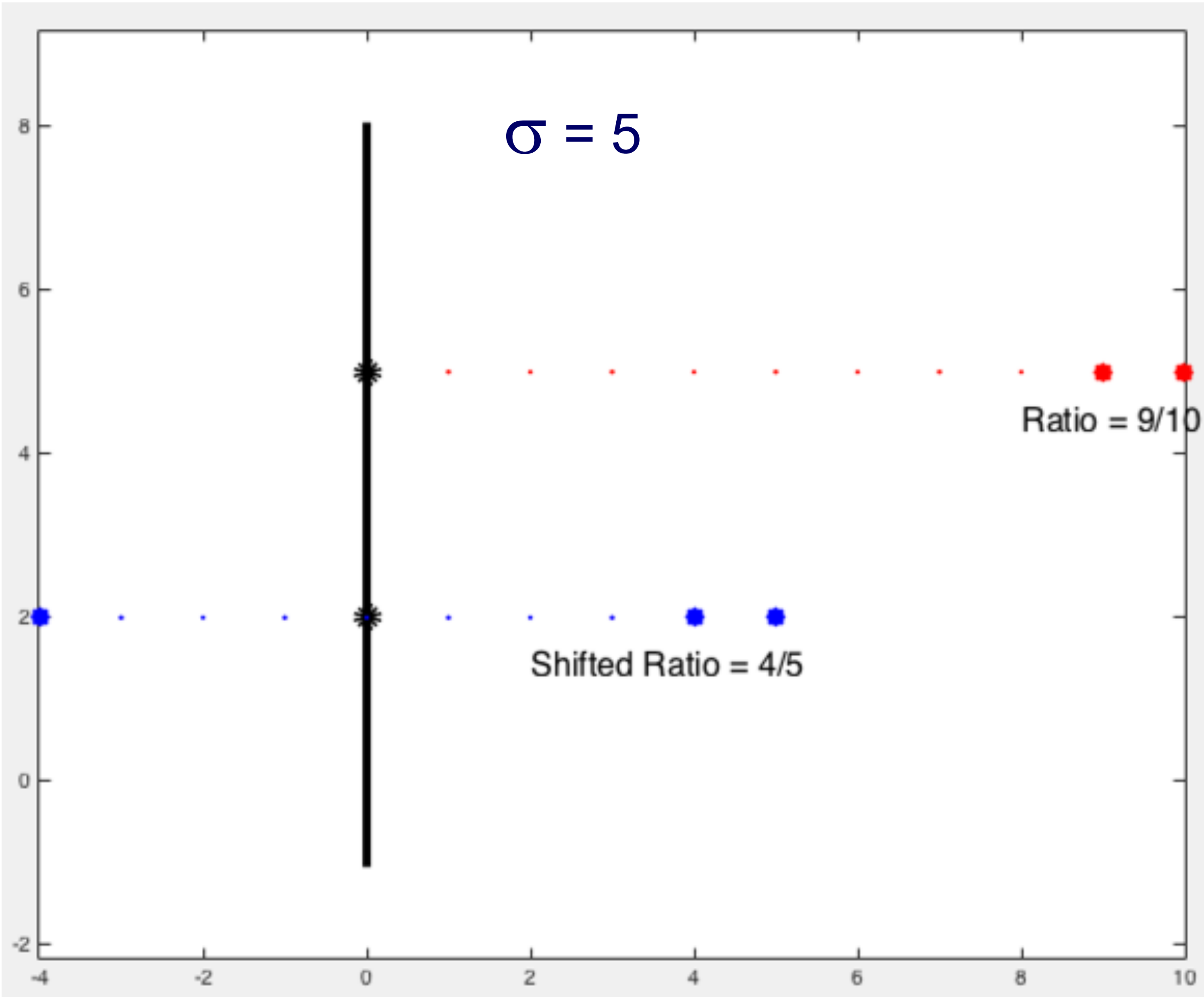
- In earlier example, if we pick $\sigma = 1$ (equal to other eigenvalue), ratio becomes zero and method converges in one iteration

- In general, we would not be able to make such fortuitous choice, but shifts can be extremely useful in some contexts, particularly with *inverse iteration*, which we will see later

Eigenvalue Ratios: Shifted and Unshifted



Eigenvalue Ratios: Shifted and Unshifted



Eigenvalue Ratios: Shifted and Unshifted

```
%
%  Shifted power demo
%

n=10;
X=rand(n,n); [X,R]=qr(X);
Lam=[n:-1:1]; L=diag(Lam);

A=X*L*X';


z=[1:n]'; z=z/norm(z); zu=z; zs=z;

s=4; I=eye(n);
H=A-s*I; err_u = 1; err_s = 1;
for k=1:400;

    yu = A*zu; lam_u = norm(yu,Inf); zu=yu/lam_u;

    ys = H*zs; lam_s = norm(ys,Inf)+s; zs=ys/norm(ys,Inf);

    err_ul=err_u; err_u = abs(Lam(1)-lam_u)/(Lam(1)); ru = err_ul/err_u;
    err_sl=err_s; err_s = abs(Lam(1)-lam_s)/(Lam(1)); rs = err_sl/err_s;

    disp([k err_u err_s ru rs]), pause

end;
```

Power Iteration with Shift

$$(A - \sigma I)\underline{x} \;\;=\;\; \lambda\underline{x} - \sigma\underline{x} \;=\; (\lambda - \sigma)\underline{x} \;=\; \mu\underline{x}$$

If $\lambda_k \in \{1 \ .9 \ \ldots \ .1\}$, then

$$\frac{\lambda_2}{\lambda_1} = 0.9$$

If $\sigma = 0.4$, then $\mu_k \in \{.5 \ .4 \ \ldots \ -.4\}$ and

$$\frac{\mu_2}{\mu_1} = 0.8,$$

so about twice the convergence rate.

Shifted power iteration, however, is somewhat limited.

The real power derives from *inverse* power iterations with shifts.

Inverse Iteration

Very Important!

- If smallest eigenvalue of matrix is required rather than largest, can make use of fact that eigenvalues of $\mathbf{A}^{-1}$ are reciprocals of $\lambda(\mathbf{A})$, so smallest eigenvalue of $\mathbf{A}$ is largest eigenvalue of $\mathbf{A}^{-1}$

- This leads to *inverse iteration* scheme,

$$
\begin{aligned}
\mathbf{A}\mathbf{y}_k &= \mathbf{x}_{k-1} \\
\mathbf{x}_k &= \mathbf{y}_k / \|\mathbf{y}_k\|_\infty
\end{aligned}
$$

  which is equivalent to power iteration applied to $\mathbf{A}^{-1}$

- Inverse of $\mathbf{A}$ not computed explicitly, but factorization of $\mathbf{A}$ used to solve system of equations at each iteration

- Can of course reuse the LU factors so that cost of successive iterations is relatively low because we only need to solve triangular systems $\mathbf{L}\mathbf{z} = \mathbf{x}_{k-1}$ and $\mathbf{U}\mathbf{y}_k = \mathbf{z}$

Inverse Iteration, continued

- Inverse iteration converges to eigenvector corresponding to *smallest* eigenvalue of $\mathbf{A}$

- Eigenvalue obtained is dominant eigenvalue of $\mathbf{A}^{-1}$ and hence its reciprocal is smallest eigenvalue of $\mathbf{A}$ in modulus

Example: Inverse Iteration

- Applying inverse iteration to previous example to compute smallest eigenvalue yields sequence

k	$\boldsymbol{x}_k^T$		$\|\boldsymbol{y}_k\|_\infty$
0	0.000	1.0	
1	−0.333	1.0	0.750
2	−0.600	1.0	0.833
3	−0.778	1.0	0.900
4	−0.882	1.0	0.944
5	−0.939	1.0	0.971
6	−0.969	1.0	0.985

which is indeed converging to 1 (which is its own reciprocal in this case)

Again, the error is reduced by ~½ on each iteration.

Inverse Iteration with Shift

- As before, shifting strategy, working with $\mathbf{A} - \sigma\mathbf{I}$ for some scalar σ, can greatly improve performance

- Inverse iteration is particularly useful for computing eigenvector corresponding to approximate eigenvalue because it converges rapidly when applied to shifted matrix $\mathbf{A} - \hat{\lambda}\mathbf{I}$, where $\hat{\lambda}$ is approximate eigenvalue

- Inverse iteration is also useful for computing eigenvalue closest to given value β because, if β is used as shift, then desired eigenvalue corresponds to smallest eigenvalue of shifted matrix

- Power Iteration: $\mathbf{x} = A^k\mathbf{x} \longrightarrow c\mathbf{x}_1$

- Normalized Power Iteration: $\left.\begin{array}{l} \mathbf{y} = A\mathbf{x} \\ \mathbf{x} = \mathbf{y}/||\mathbf{y}|| \end{array}\right\} \begin{array}{l} ||\mathbf{y}|| \longrightarrow |\lambda_1| \\ \mathbf{x} \longrightarrow \mathbf{x}_1 \end{array}$

- Inverse Iteration: $\left.\begin{array}{l} \mathbf{y} = A^{-1}\mathbf{x} \\ \mathbf{x} = \mathbf{y}/||\mathbf{y}|| \end{array}\right\} \begin{array}{l} ||\mathbf{y}|| \longrightarrow |\lambda_n|^{-1} \\ \mathbf{x} \longrightarrow \mathbf{x}_n \end{array}$

- Inverse Iteration with shift: $\left.\begin{array}{l} M = A - \sigma I \\ \mathbf{y} = M^{-1}\mathbf{x} \\ \mathbf{x} = \mathbf{y}/||\mathbf{y}|| \end{array}\right\} \begin{array}{l} ||\mathbf{y}|| \longrightarrow |\mu_k| = \max|\lambda_k - \sigma|^{-1} \\ \mathbf{x} \longrightarrow \mathbf{x}_k \end{array}$

Inverse iteration with shift can be arbitrarily fast since separation ratio can be 0.

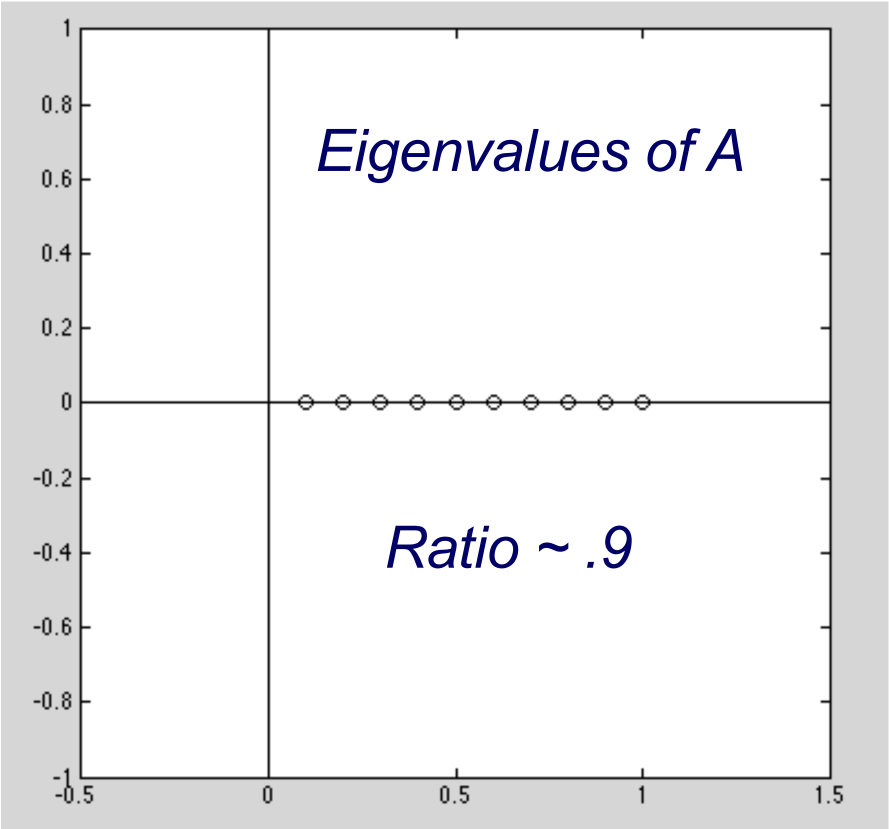
- Inverse Iteration:

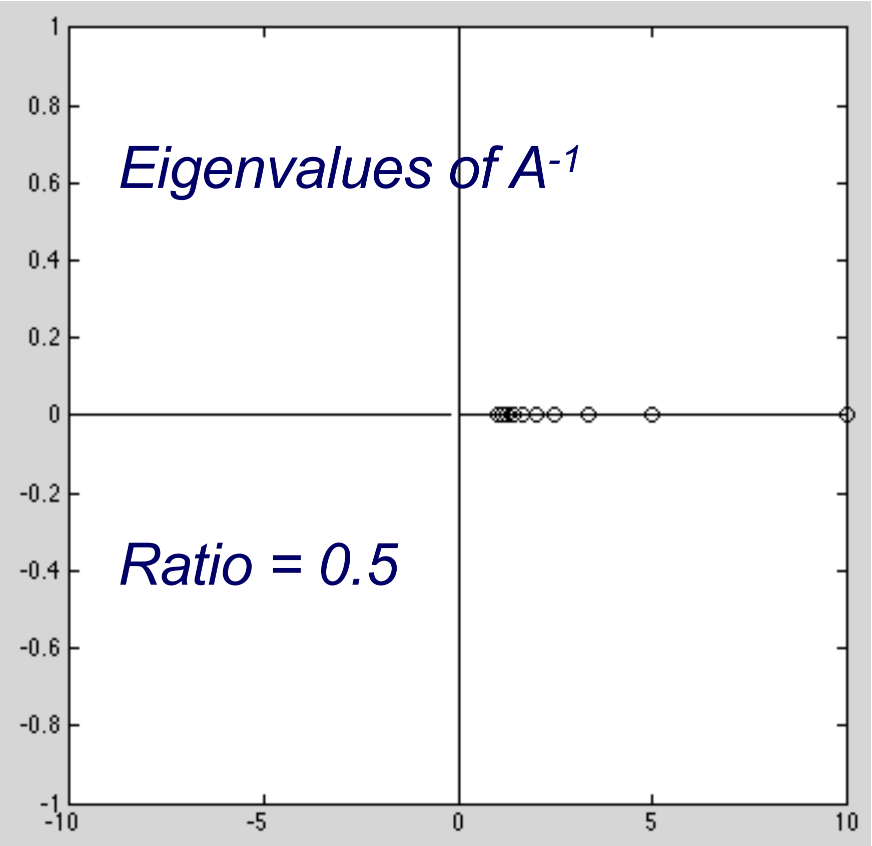
$$
\begin{aligned}
\mathbf{x}^k &= A^{-k}\mathbf{x}^0 \\
&= \sum_{j=1}^{n} \mathbf{x}_j \left(\frac{1}{\lambda_j}\right)^k c_j \\
&= \left(\frac{1}{\lambda_n}\right)^k c_n \left[\mathbf{x}_n + \sum_{j=1}^{n-1} \mathbf{x}_j \left(\frac{\lambda_n}{\lambda_j}\right)^k c_j\right].
\end{aligned}
$$

eig_inv_demo.m

Inverse Iteration Illustration

- With shift and invert, can get significant ratios of dominant eigenvalue





- Inverse Iteration with Shift: Let

$$M := A - \sigma I$$

$$\mu_j := \lambda_j - \sigma, \text{ and}$$

$$l \text{ such that } |\sigma - \mu_l| < |\sigma - \mu_j|, \; j \neq l.$$

Then,

$$\begin{aligned}
\mathbf{x}^k &= M^{-k}\mathbf{x}^0 \\
&= \sum_{j=1}^{n} \mathbf{x}_j \left(\frac{1}{\mu_j}\right)^k c_j \\
&= \left(\frac{1}{\mu_l}\right)^k c_l \left[\mathbf{x}_l + \sum_{j \neq l} \mathbf{x}_j \left(\frac{\mu_l}{\mu_j}\right)^k c_j\right].
\end{aligned}$$

- Using current approximation to λ_l can select $|\sigma - \lambda_l| = |\mu_l|$ to be small. (Cannot do the same with shifted power iteration.)
- Blow-up is contained by normalizing after each iteration.

  eig_shift_invert.m

Rayleigh Quotient

- Given approximate eigenvector $\mathbf{x}$ for real matrix $\mathbf{A}$, determining best estimate for corresponding λ can be considered as a $n \times 1$ LLSQ approximation problem,

$$\mathbf{x}\lambda \approx \mathbf{A}\mathbf{x}$$

- From normal equation, $\mathbf{x}^T\mathbf{x}\lambda = \mathbf{x}^T\mathbf{A}\mathbf{x}$, LLSQ solution is

$$\lambda = \frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$$

- This quantity, known as the *Rayleigh Quotient* has many useful properties

Example: Rayleigh Quotient

- Rayleigh quotient can accelerate convergence of iterative methods such as power iteration because Rayleigh quotient $\mathbf{x}_k^T\mathbf{A}\mathbf{x}_k/\mathbf{x}_k^T\mathbf{x}_k$ gives better approximation to λ that basic power iteration

- For previous example from text using power iteration, value of Rayleigh quotient at each iteration is shown below

k	$\boldsymbol{x}_k^T$		$\\|\boldsymbol{y}_k\\|_\infty$	$\boldsymbol{x}_k^T\boldsymbol{A}\boldsymbol{x}_k/\boldsymbol{x}_k^T\boldsymbol{x}_k$
0	0.000	1.0		
1	0.333	1.0	1.500	1.500
2	0.600	1.0	1.667	1.800
3	0.778	1.0	1.800	1.941
4	0.882	1.0	1.889	1.985
5	0.939	1.0	1.941	1.996
6	0.969	1.0	1.970	1.999

Some Rayleigh Quotient Properties

- View $r(\mathbf{x})$ as a function of $\mathbf{x} \in \mathbb{R}^n$,

$$r(\mathbf{x}) \;=\; \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}}.$$

- Then *gradient* of r is

$$\begin{aligned} \frac{\partial}{\partial x_j} r(\mathbf{x}) &= \frac{2}{\mathbf{x}^T \mathbf{x}} \left(A\mathbf{x} \,-\, r(\mathbf{x})\mathbf{x} \right)_j \\ \nabla r(\mathbf{x}) &= \frac{2}{\mathbf{x}^T \mathbf{x}} \left(A\mathbf{x} \,-\, r(\mathbf{x})\mathbf{x} \right). \end{aligned}$$

- If $\mathbf{x}$ is an eigenvector of A then $r(\mathbf{x}) = \lambda$.
- Moreover, $\nabla r(\mathbf{x}) \;=\; 0$.

- **Main result:** If $\mathbf{q}_j$ is the jth eigenvector of A, then

$$|r(\mathbf{x}) \,-\, r(\mathbf{q}_j)| \;=\; O(\|\mathbf{x} - \mathbf{q}_j\|^2) \;\text{ as } \mathbf{x} \longrightarrow \mathbf{q}_j.$$

Rayleigh Quotient Iteration

- For approximate eigenvector, Rayleigh quotient yields good estimate for corresponding λ

- Moreover, inverse iteration converges rapidly to eigenvector if approximate eigenvalue is used as shift

- Combining these two ideas lead to *Rayleigh Quotient Iteration*,

$$\begin{aligned}\sigma_k &= \mathbf{x}_k^T \mathbf{A} \mathbf{x}_k / \mathbf{x}_k^T \mathbf{x}_k \\ (\mathbf{A} - \sigma_k \mathbf{I}) \mathbf{y}_{k+1} &= \mathbf{x}_k \\ \mathbf{x}_{k+1} &= \mathbf{y}_{k+1} / \|\mathbf{y}_{k+1}\|_2\end{aligned}$$

- If we start from $\mathbf{x}_0$ with $\|\mathbf{x}_0\|_2 = 1$, the normalization $\mathbf{x}_k^T \mathbf{x}$ is superfluous.

Rayleigh Quotient Iteration, continued

- Rayleigh quotient iteration is especially effective for symmetric matrices and usually converges very rapidly (e.g., just one or two iterations may suffice)

- Using different shift at each iteration means matrix must be refactored each time to solve linear system, so cost per iteration is high unless matrix has special form that makes factorization easy

- Factorization overhead can be amortized by taking several iterations by reusing **LU** factors between refactorizations. Fixed point iteration will still be convergent

- If $\mathbf{A}$ is tridiagonal, factorization and solve cost is only $\approx 8n$ operations

Example: Rayleigh Quotient Iteration

- Using same matrix as previous examples and randomly chosen starting vector $\boldsymbol{x}_0$, Rayleigh quotient iteration converges in two iterations

k	$\boldsymbol{x}_k^T$		σ_k
0	0.807	0.397	1.896
1	0.924	1.000	1.998
2	1.000	1.000	2.000

Matlab Demo: eig_shifted_rq.m

Deflation

- After eigenvalue λ_1 and corresponding $\mathbf{x}_1$ has been computed, additional eigenvalues can be computed by *deflation*, which effectively removes known eigenvalue

- Let $\mathbf{H}$ be any nonsingular matrix such that $\mathbf{H}\mathbf{x}_1 = \alpha\mathbf{e}_1$, scalar multiple of first column of identity matrix (Householder transformation is a good candidate for $\mathbf{H}$)

- Then similarity transformation determined by $\mathbf{H}$ transforms $\mathbf{A}$ into form

$$\mathbf{H}\mathbf{A}\mathbf{H}^{-1} = \begin{bmatrix} \lambda_1 & \mathbf{b}^T \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

  where $\mathbf{B}$ is order $n-1$ matrix having eigenvalues $\lambda_2, \ldots, \lambda_n$

Deflation, continued

- Thus, we can work with $\mathbf{B}$ to compute next eigenvalue λ_2
- Moreover, if $\mathbf{y}_2$ is eigenvector of $\mathbf{B}$ corresponding to λ_2, then

$$\mathbf{x}_2 = \mathbf{H}^{-1} \begin{bmatrix} \alpha \\ \mathbf{y}_2 \end{bmatrix}, \quad \text{where } \alpha = \frac{\mathbf{b}^T \mathbf{y}_2}{\lambda_2 - \lambda_1}$$

  is eigenvector corresponding to λ_2 for original matrix $\mathbf{A}$, provided $\lambda_1 \neq \lambda_2$
- Process can be repeated to find additional eigenpairs

Deflation – Finding Second Eigenpair

- Choose H to be elementary Householder matrix such that $H\mathbf{x}_1 = \mathbf{e}_1$.
- Consider

$$
\begin{aligned}
A\mathbf{x}_1 &= \lambda_1\,\mathbf{x}_1 \\
AH^{-1}H\mathbf{x}_1 &= \lambda_1\,H^{-1}H\mathbf{x}_1 \\
AH^{-1}\mathbf{e}_1 &= \lambda_1\,H^{-1}\mathbf{e}_1 \\
HAH^{-1}\mathbf{e}_1 &= \lambda_1\,\mathbf{e}_1 \qquad\qquad\qquad\qquad *
\end{aligned}
$$

$$
\begin{aligned}
A_2 &:= HAH^{-1} = HAH^{-1}\,I \\
&= HAH^{-1}\,[\,\mathbf{e}_1\;\mathbf{e}_2\;\ldots\;\mathbf{e}_n\,] \\
&= \begin{bmatrix} \lambda_1 & \mathbf{b}^T \\ 0 & \\ 0 & B \\ 0 & \end{bmatrix}
\end{aligned}
$$

- Apply method of choice to B to find λ_2.

Matrix-Free Deflation – Symmetric A

Deflation via Projection.

for $k = 1,\ 2, \ldots$

$$\begin{aligned} \mathbf{y} &= A\mathbf{x} \\ \mathbf{y} &= \mathbf{y} - \mathbf{x}_1 \frac{\mathbf{x}_1^T \mathbf{y}}{\mathbf{x}_1^T \mathbf{x}_1} = \mathbf{y} - \mathbf{x}_1 \mathbf{x}_1^T \mathbf{y}^{*} \\ \mathbf{x} &= \mathbf{y} / ||\mathbf{y}||. \end{aligned}$$

- Guarantees that $\mathbf{x}$ is devoid of any component of $\mathbf{x}_1$ prior to start of each application of power method.
- Do not require knowledge of A.
- Only need a routine that provides $\mathbf{y} \longleftarrow A\mathbf{x}$.
- Convenient for large sparse matrices when trying to avoid computing known eigenpairs.

*assuming $\mathbf{x}_1$ has unit 2-norm.

Deflation on the Fly – *Subspace Iteration*

Can effect deflation on the fly — *Subspace Iteration.*

- Take two independent vectors $Y = (\mathbf{y}_1\ \mathbf{y}_2)$.

$$\text{for } k = 1,\ 2, \ldots$$

$$\begin{aligned} Z &= AY \\ \mathbf{y}_1 &= \mathbf{z}_1/||\mathbf{z}_1||. \\ \mathbf{y}_2 &= \mathbf{z}_2 - \mathbf{y}_1\mathbf{y}_1^T\mathbf{z}_2 \\ \mathbf{y}_2 &= \mathbf{y}_2/||\mathbf{y}_2|| \end{aligned}$$

y_2* orthogonal to *y_1

- $\mathbf{y}_1$ converges to $\mathbf{x}_1$ (*standard power iteration*).
- $(\mathbf{y}_1\ \mathbf{y}_2)$ converge to $span(\mathbf{x}_1\ \mathbf{x}_2)$.
- $\mathbf{y}_2$ converges to $\mathbf{x}_2$ if A is symmetric or $\mathbf{x}_2 \perp \mathbf{x}_1$.

subspace.m

- $\mathbf{z}_1^T\mathbf{x}_1 \longrightarrow \lambda_1$ and $\mathbf{z}_2^T\mathbf{x}_2 \longrightarrow \lambda_2$.