Summary of eigenvalue/vector methods so far

1. Methods for finding 1 or a few eigenvalues/vectors.

   Power method and its variations (shifting, inverse iteration, deflation, Rayleigh quotient iteration)

2. Methods for finding all eigenvalues and perhaps eigenvectors.

   Apply similarity transformations to the matrix that bring it closer to diagonal. For symmetric matrices, orthogonal transformations (rotations, reflections).

   Jacobi method: apply rotations again and again gradually reducing off-diagonal matrix elements “indiscriminately”.

   Or: first reduce the matrix to tridiagonal, which can be done in a finite number of iterations by applying a specific series of rotations (Givens) or reflections (Householder).

3. Tridiagonal matrices.

An iterative procedure for the characteristic polynomial and its roots.

The QL algorithm where a similarity transformation is done by decomposing the matrix into a QL product and then by inverting the order (LQ) a similar matrix is produced that is used at the next stage. These iterations converge rapidly to a diagonal matrix (although there is no exact convergence in a finite number of steps).

Different methods can be combined. For example, while it is possible to keep track of the similarity transformations and then apply the inverse transformation to get the eigenvectors, this is rather computationally intense. If only a small fraction of eigenvectors are required (the rule of thumb is <25%), then it is better to obtain only the eigenvalues and then use the inverse iteration with a shift to obtain the corresponding eigenvector for each of them.

A few words about the QL method.

Start with $\boldsymbol{A}_0=\boldsymbol{A}$. A sequence of transformations:

$$\boldsymbol{A}_s=\boldsymbol{Q}_s\boldsymbol{L}_s \quad \text{(decomposition)}$$

The product of an orthogonal and a lower (or left) triangular matrix

Form new matrix $\boldsymbol{A}_{s+1}=\boldsymbol{L}_s\boldsymbol{Q}_s$

This sequence actually converges to a lower triangular matrix in general; in our case when the original matrix is tridiagonal symmetric, this would be a diagonal matrix.

1) Can just as well do the QR decomposition working with upper (or **R**ight) triangular matrices

2) What is the meaning of the QR (or QL) decomposition?

If $n\times n$ matrix $\boldsymbol{A}$ is non-degenerate (aka invertible), its columns are n linearly independent vectors and form a basis in n-dimensional space. Can be orthonormalized.

$$\boldsymbol{A}=(\vec{a}_1,\vec{a}_2,\ldots,\vec{a}_n) \qquad \vec{q}_1=\vec{a}_1/\|\vec{a}\| \;\Rightarrow\; \vec{a}_1=\|\vec{a}\|\vec{q}_1=r_{11}\vec{q}_1$$

$$\vec{q}_2\propto\vec{a}_2-d_{21}\vec{q}_1 \qquad \vec{q}_1\cdot\vec{q}_2\propto\vec{q}_1\cdot\vec{a}_2-d_{21}\vec{q}_1\cdot\vec{q}_1=0 \;\Rightarrow\; d_{21}=\frac{\vec{q}_1\cdot\vec{a}_2}{\vec{q}_1\cdot\vec{q}_1}$$

$$\vec{q}_2=\left(\vec{a}_2-\frac{\vec{q}_1\cdot\vec{a}_2}{\vec{q}_1\cdot\vec{q}_1}\vec{q}_1\right)\Big/\left\|\vec{a}_2-\frac{\vec{q}_1\cdot\vec{a}_2}{\vec{q}_1\cdot\vec{q}_1}\vec{q}_1\right\| \;\Rightarrow\; \vec{a}_2=r_{12}\vec{q}_1+r_{22}\vec{q}_2$$

$$\vec{q}_3\propto\vec{a}_3-d_{31}\vec{q}_1-d_{32}\vec{q}_2 \qquad \vec{q}_1\cdot\vec{q}_3=\vec{q}_2\cdot\vec{q}_3=0 \;\Rightarrow\; d_{31},\ d_{32}$$

$$\vec{a}_3=r_{13}\vec{q}_1+r_{23}\vec{q}_2+r_{33}\vec{q}_3$$

Gram-Schmidt process

$$\vec{a}_k=\Sigma_{i=1}^{k}r_{ik}\vec{q}_i \qquad \boldsymbol{Q}=(\vec{q}_1,\vec{q}_2,..,\vec{q}_n)\ \text{(orthogonal)} \qquad a_{jk}=\Sigma_{i=1}^{k}r_{ik}q_{ji}$$

$$\boldsymbol{A}=\boldsymbol{QR} \qquad \boldsymbol{R}=(r_{ij})=\begin{pmatrix} r_{11} & r_{12} & r_{13} & \ldots & r_{1n} \\ 0 & r_{22} & r_{23} & \ldots & r_{2n} \\ 0 & 0 & r_{33} & \ldots & r_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & r_{nn} \end{pmatrix}$$

For QL, $\vec{a}_n=l_{nn}\vec{q}_n$ $\qquad \vec{a}_{n-1}=l_{n-1,n-1}\vec{q}_{n-1}+l_{n,n-1}\vec{q}_n$ $\qquad \vec{a}_k=\Sigma_{i=k}^{n}l_{ik}\vec{q}_i$

$$\boldsymbol{A}=\boldsymbol{QL} \qquad \boldsymbol{L}=(l_{ij})$$ is lower triangular

3) Why does the QL algorithm work?

Relation to the power method. $A^k \vec{v}^{(0)}$ converges to the dominant eigenvector. Suppose we want to start with *n* different initial vectors $\vec{v}_i^{(0)}$ and iterate them to make them converge to all *n* distinct eigenvectors. If we simply calculate $A^k \vec{v}_i^{(0)}$, all will converge to the same dominant one. Need to orthonormalize during the process.

Start with $\vec{e}_i = (0 \quad \ldots \quad 0 \quad \underset{i}{1} \quad 0 \quad \ldots \quad 0)^T$. $\quad A(\vec{e}_1 \quad \ldots \quad \vec{e}_n) = A$

To orthonormalize, do QL decomposition: $A = Q_0 L_0$. The orthonormalized vectors are the columns of $\boldsymbol{Q}_0$. Act on them with $\boldsymbol{A}$ again:

$AQ_0 = Q_0 L_0 Q_0 = Q_0 Q_1 L_1$ So after orthonormalization, we get $\boldsymbol{Q}_0\boldsymbol{Q}_1$.

$$AQ_0 Q_1 = Q_0 L_0 Q_0 Q_1 = Q_0 Q_1 L_1 Q_1 = Q_0 Q_1 Q_2 L_2$$

After *m* iterations, get $AQ_0 Q_1 \ldots Q_{m-1} = Q_0 Q_1 \ldots Q_{m-1} Q_m L_m$.

If the sequence $Q_0,\ Q_0 Q_1,\ Q_0 Q_1 Q_2, \ldots$ converges towards the set of eigenvectors, then $\boldsymbol{Q}_m\boldsymbol{L}_m$ converges towards a diagonal matrix with eigenvalues on the diagonal.

Complex Hermitian matrices

For matrix $\boldsymbol{A}=(a_{jk})$, its Hermitian conjugate is $\boldsymbol{A}^H=(a_{kj}^*)$. Also denoted $\boldsymbol{A}^\dagger$.

Matrix ***A*** is Hermitian if $\boldsymbol{A}^H = \boldsymbol{A}$. Important in quantum mechanics, since operators of all observables (including the Hamiltonian) have to be self-adjoint and the corresponding matrices in any basis are Hermitian.

Example: momentum operator in 1D is $\hat{p}=-i\hbar\frac{d}{dx}$. Consider an arbitrary function *f*(*x*) on interval [0,1]. Discretize it on the grid with a constant step $b=1/n$: $f_j = f(bj)$, $j = 0,\ldots,n-1$. So the function is represented as a vector of length *n*. Assume periodic boundary conditions [$f(x+1) = f(x)$]. Then

$$\hat{p}\,f(x)=-i\hbar\frac{df(x)}{dx} \qquad [\hat{p}\,f(x)]_j\approx-i\hbar\frac{f_{(j+1)\bmod n}-f_{(j-1)\bmod n}}{2b}$$

$$\hat{p}\,f(x)\to\boldsymbol{P}\vec{f} \qquad \boldsymbol{P}=\begin{pmatrix} 0 & -i\hbar/2b & 0 & 0 & \ldots & 0 & 0 & i\hbar/2b \\ i\hbar/2b & 0 & -i\hbar/2b & 0 & \ldots & 0 & 0 & 0 \\ 0 & i\hbar/2b & 0 & -i\hbar/2b & \ldots & 0 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & i\hbar/2b & 0 & -i\hbar/2b \\ -i\hbar/2b & 0 & 0 & 0 & \ldots & 0 & i\hbar/2b & 0 \end{pmatrix}$$

Complex Hermitian matrices

Have many of the properties of real symmetric matrices: they are diagonalizable, *n* distinct eigenvectors that are mutually orthogonal or orthogonalizable, and the eigenvalues are real (even if the matrix elements are complex).

$\boldsymbol{A}=\boldsymbol{B}+i\boldsymbol{C}$, where $\boldsymbol{B}$ and $\boldsymbol{C}$ are real, $\boldsymbol{B}$ is symmetric, $\boldsymbol{C}$ is antisymmetric.

$$(\boldsymbol{B}+i\boldsymbol{C})(\vec{u}+i\vec{v})=\lambda(\vec{u}+i\vec{v})$$

$$\boldsymbol{B}\vec{u}-\boldsymbol{C}\vec{v}+i(\boldsymbol{C}\vec{u}+\boldsymbol{B}\vec{v})=\lambda\vec{u}+i\lambda\vec{v}$$

Separately for real and imaginary parts:

$$\boldsymbol{B}\vec{u}-\boldsymbol{C}\vec{v}=\lambda\vec{u}$$
$$\boldsymbol{C}\vec{u}+\boldsymbol{B}\vec{v}=\lambda\vec{v}$$

$$\begin{pmatrix}\boldsymbol{B} & -\boldsymbol{C}\\ \boldsymbol{C} & \boldsymbol{B}\end{pmatrix}\begin{pmatrix}\vec{u}\\ \vec{v}\end{pmatrix}=\lambda\begin{pmatrix}\vec{u}\\ \vec{v}\end{pmatrix}$$

$$\boldsymbol{D}=\begin{pmatrix}\boldsymbol{B} & -\boldsymbol{C}\\ \boldsymbol{C} & \boldsymbol{B}\end{pmatrix}$$

$$d_{jk}=b_{jk}=b_{kj}=d_{kj} \qquad d_{j+n,k+n}=b_{jk}=b_{kj}=d_{k+n,j+n} \qquad d_{j+n,k}=c_{jk}=-c_{kj}=d_{k,j+n}$$

$$j,k=1,\ldots,n$$

Hermitian eigenvalue problem of size *n* is reduced to a real symmetric problem of size 2*n*.

$$\begin{pmatrix} \boldsymbol{B} & -\boldsymbol{C} \\ \boldsymbol{C} & \boldsymbol{B} \end{pmatrix}\begin{pmatrix} \vec{u} \\ \vec{v} \end{pmatrix}=\lambda\begin{pmatrix} \vec{u} \\ \vec{v} \end{pmatrix}$$

$2n$ equations, so $2n$ eigenvalues/vectors, but the original problem has n?

Suppose $\begin{pmatrix} \vec{u} \\ \vec{v} \end{pmatrix}$ is an eigenvector, so $\begin{aligned} \boldsymbol{B}\vec{u}-\boldsymbol{C}\vec{v}&=\lambda\vec{u} \\ \boldsymbol{C}\vec{u}+\boldsymbol{B}\vec{v}&=\lambda\vec{v} \end{aligned}$ Consider $\begin{pmatrix} -\vec{v} \\ \vec{u} \end{pmatrix}$.

$$\begin{pmatrix} \boldsymbol{B} & -\boldsymbol{C} \\ \boldsymbol{C} & \boldsymbol{B} \end{pmatrix}\begin{pmatrix} -\vec{v} \\ \vec{u} \end{pmatrix}=\begin{pmatrix} -\boldsymbol{B}\vec{v}-\boldsymbol{C}\vec{u} \\ -\boldsymbol{C}\vec{v}+\boldsymbol{B}\vec{u} \end{pmatrix}=\lambda\begin{pmatrix} -\vec{v} \\ \vec{u} \end{pmatrix}$$

So it's also an eigenvector with the same λ.

But $-\vec{v}+i\vec{u}=i\left(\vec{u}+i\vec{v}\right)$. So while $\begin{pmatrix} -\vec{v} \\ \vec{u} \end{pmatrix}$ is distinct from $\begin{pmatrix} \vec{u} \\ \vec{v} \end{pmatrix}$ for the real $(2n)\times(2n)$ problem, they represent the same eigenvector of the original complex $n\times n$ problem. This is why the numbers of eigenvectors differ by a factor of 2.

While this approach works, dealing with the $n\times n$ complex matrix directly can save about a factor of 2 in memory and CPU time.

For real symmetric matrices, similarity transforms $\boldsymbol{Q}^{-1}\boldsymbol{A}\boldsymbol{Q}$ involved an orthogonal matrix, $\boldsymbol{Q}^T = \boldsymbol{Q}^{-1}$. The analog for Hermitian matrices is unitary matrices: $\boldsymbol{U}^H = \boldsymbol{U}^{-1}$. $\boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{U} = \boldsymbol{U}^H\boldsymbol{A}\boldsymbol{U}$.

For Jacobi rotation we had:

$$\boldsymbol{P}_{pq}(\theta) = \begin{pmatrix} 1 & \dots & 0 & \dots & \dots & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & c & \dots & 0 & \dots & s & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & 1 & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & -s & \dots & \dots & \dots & c & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & \dots & \dots & 0 & \dots & 1 \end{pmatrix} \begin{matrix} \\ \\ p \\ \\ \\ \\ q \\ \\ \\ \end{matrix}$$

$$\begin{matrix} & & p & & & & q & & \end{matrix}$$

$$c = \cos\theta, \; s = \sin\theta$$

Analogous complex transform:

$$\boldsymbol{R}_{pq}(\theta) = \begin{pmatrix} 1 & \dots & 0 & \dots & \dots & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \frac{e^{-i\theta}}{\sqrt{2}} & \dots & 0 & \dots & \frac{e^{i\theta}}{\sqrt{2}} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & 1 & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \frac{-e^{-i\theta}}{\sqrt{2}} & \dots & \dots & \dots & \frac{e^{i\theta}}{\sqrt{2}} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \dots & \dots & \dots & 0 & \dots & 1 \end{pmatrix} \begin{matrix} \\ \\ p \\ \\ \\ \\ q \\ \\ \\ \end{matrix}$$

$$\begin{matrix} & & p & & & & q & & \end{matrix}$$

To make a particular matrix element zero, need 2 parameters.

For ordinary rotations, $\boldsymbol{P}_{pq}(\theta_2)\boldsymbol{P}_{pq}(\theta_1)=\boldsymbol{P}_{pq}(\theta_1+\theta_2)$. Not the case for $\boldsymbol{R}_{pq}$.

$$\boldsymbol{R}_{12}(\theta_2)\boldsymbol{R}_{12}(\theta_1)=\begin{pmatrix} \frac{e^{-i\theta_2}}{\sqrt{2}} & \frac{e^{i\theta_2}}{\sqrt{2}} \\ \frac{-e^{-i\theta_2}}{\sqrt{2}} & \frac{e^{i\theta_2}}{\sqrt{2}} \end{pmatrix}\begin{pmatrix} \frac{e^{-i\theta_1}}{\sqrt{2}} & \frac{e^{i\theta_1}}{\sqrt{2}} \\ \frac{-e^{-i\theta_1}}{\sqrt{2}} & \frac{e^{i\theta_1}}{\sqrt{2}} \end{pmatrix}=\begin{pmatrix} -ie^{-i\theta_1}\sin\theta_2 & e^{i\theta_1}\cos\theta_2 \\ -e^{-i\theta_1}\cos\theta_2 & ie^{i\theta_1}\sin\theta_2 \end{pmatrix}$$

Has 2 real parameters, can be used to eliminate both the real and the imaginary parts of the desired matrix element.

Householder transformation

$$\boldsymbol{P}=\boldsymbol{I}-2\,\vec{w}\otimes\vec{w}^{*}$$

It is possible to make the resulting tridiagonal matrix real by doing an additional transform $\boldsymbol{DTD}^{-1}$, where $\boldsymbol{D}$ = diag(τ_1,...,τ_n), $\tau_1 = 1$, $\tau_{j+1} = \tau_j \exp(-i\Phi_j)$, where Φ_j is the phase of $t_{j+1,j}$. In this transformation, element t_{jk} is multiplied by τ_j/τ_k.

General matrices (very briefly)

As in the symmetric case, it makes sense to do matrix transformation in two stages. In the first stage, transform into the upper Hessenberg form:

$$\begin{pmatrix}
a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & \dots & a_{1,n-3} & a_{1,n-2} & a_{1,n-1} & a_{1,n} \\
a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & \dots & a_{2,n-3} & a_{2,n-2} & a_{2,n-1} & a_{2,n} \\
0 & a_{32} & a_{33} & a_{34} & a_{35} & \dots & a_{3,n-3} & a_{3,n-2} & a_{3,n-1} & a_{3,n} \\
0 & 0 & a_{43} & a_{44} & a_{45} & \dots & a_{4,n-3} & a_{4,n-2} & a_{4,n-1} & a_{4,n} \\
0 & 0 & 0 & a_{54} & a_{55} & \dots & a_{5,n-3} & a_{5,n-2} & a_{5,n-1} & a_{5,n} \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & 0 & 0 & 0 & \dots & a_{n-2,n-3} & a_{n-2,n-2} & a_{n-2,n-1} & a_{n-2,n} \\
0 & 0 & 0 & 0 & 0 & \dots & 0 & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\
0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & a_{n,n-1} & a_{n,n}
\end{pmatrix}$$

Can be done using Householder transforms or by a variant of Gaussian elimination. Then make closer to upper triangular iteratively, likewise using Householder transforms.

Prior to these transformations, balance the matrix by applying a diagonal transform, making sure that elements in the row and corresponding column are of the same order of magnitude.

Back to symmetric matrices. In addition, require them to be positive definite.

Another way of transforming into a tridiagonal matrix: Lanczos method

Particularly attractive because a few extreme eigenvalues can be obtained without bringing the iterations to completion. Especially advantageous for large sparse matrices.

Closely related to the conjugate gradient method for linear systems.

In the conjugate gradient method, we started with $\vec{r}^{(0)}$ (which was the residual at the starting point, but it does not matter – can be an arbitrary vector).

$$\beta_k = \frac{\vec{r}^{(k)} \cdot \vec{r}^{(k)}}{\vec{r}^{(k-1)} \cdot \vec{r}^{(k-1)}} \text{ for } k>0 \text{ or } \beta_0 = 0 \qquad \vec{p}^{(k)} = \vec{r}^{(k)} + \beta_k \vec{p}^{(k-1)}$$

$$\alpha_k = \frac{\vec{r}^{(k)} \cdot \vec{r}^{(k)}}{\vec{p}^{(k)} \cdot \boldsymbol{A} \cdot \vec{p}^{(k)}} \qquad \vec{r}^{(k+1)} = \vec{r}^{(k)} - \alpha_k \boldsymbol{A} \vec{p}^{(k)}$$

This generates a sequence of orthogonal vectors $\vec{r}^{(k)}$ $[r^{(j)} \cdot r^{(k)} = 0,\ j \neq k]$

and a sequence of conjugate vectors $\vec{p}^{(k)}$ $[\vec{p}^{(j)} \cdot \boldsymbol{A} \vec{p}^{(k)} = 0,\ j \neq k]$.

$$\vec{r}^{(k)} = \vec{p}^{(k)} - \beta_k \vec{p}^{(k-1)} \qquad \boldsymbol{R}_k = (\vec{r}^{(0)}, \ldots, \vec{r}^{(k-1)}) \qquad \boldsymbol{P}_k = (\vec{p}^{(0)}, \ldots, \vec{p}^{(k-1)})$$

$$\boldsymbol{R}_k = \boldsymbol{P}_k \boldsymbol{B}_k, \text{ where } \boldsymbol{B}_k = \begin{pmatrix} 1 & -\beta_1 & 0 & \ldots & 0 \\ 0 & 1 & -\beta_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & -\beta_{k-1} \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix}.$$

$$\boldsymbol{R}_k=\left(\vec{r}^{(0)},\ldots,\vec{r}^{(k-1)}\right) \qquad \boldsymbol{P}_k=\left(\vec{p}^{(0)},\ldots,\vec{p}^{(k-1)}\right) \qquad \boldsymbol{R}_k=\boldsymbol{P}_k\boldsymbol{B}_k$$

$$\boldsymbol{B}_k=\begin{pmatrix} 1 & -\beta_1 & 0 & \ldots & 0 \\ 0 & 1 & -\beta_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & -\beta_{k-1} \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix}.$$

$$\boldsymbol{R}_k^T\boldsymbol{A}\boldsymbol{R}_k=\boldsymbol{B}_k^T\boldsymbol{P}_k^T\boldsymbol{A}\boldsymbol{P}_k\boldsymbol{B}_k$$

$$\vec{p}^{(j)}\cdot\boldsymbol{A}\,\vec{p}^{(k)}=0$$

$$\boldsymbol{P}_k^T\boldsymbol{A}\boldsymbol{P}_k=\begin{pmatrix} \vec{p}^{(0)}\boldsymbol{A}\,\vec{p}^{(0)} & 0 & \ldots & 0 & 0 \\ 0 & \vec{p}^{(1)}\boldsymbol{A}\,\vec{p}^{(1)} & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \vec{p}^{(k-2)}\boldsymbol{A}\,\vec{p}^{(k-2)} & 0 \\ 0 & 0 & \ldots & 0 & \vec{p}^{(k-1)}\boldsymbol{A}\,\vec{p}^{(k-1)} \end{pmatrix}$$

Therefore $\boldsymbol{R}_k^T\boldsymbol{A}\boldsymbol{R}_k$ is a tridiagonal $k\times k$ matrix.

$\boldsymbol{Q}_k=\boldsymbol{R}_k\boldsymbol{D}^{-1}$, where $\boldsymbol{D}=\mathrm{diag}\left(\|\vec{r}^{(0)}\|,\ldots,\|\vec{r}^{(k-1)}\|\right)$.

$\boldsymbol{Q}_k^T\boldsymbol{A}\boldsymbol{Q}_k$ is, of course, still tridiagonal.

After n iterations (k=n), $\boldsymbol{Q}_n$ is an orthogonal $n\times n$ matrix.

$\boldsymbol{Q}_k^T \boldsymbol{A} \boldsymbol{Q}_k$ is a similarity transform of matrix $\boldsymbol{A}$, and the result is tridiagonal.

$$\vec{p}^{(k)} = \vec{r}^{(k)} + \beta_k \vec{p}^{(k-1)} \qquad \vec{r}^{(k+1)} = \vec{r}^{(k)} - \alpha_k \boldsymbol{A} \vec{p}^{(k)}$$

$$\boldsymbol{A}\vec{p}^{(k)} = \boldsymbol{A}\vec{r}^{(k)} + \beta_k \boldsymbol{A}\vec{p}^{(k-1)} \Rightarrow \boldsymbol{A}\vec{r}^{(k)} = \boldsymbol{A}\vec{p}^{(k)} - \beta_k \boldsymbol{A}\vec{p}^{(k-1)}$$

$$= \frac{1}{\alpha_k}\left(\vec{r}^{(k)} - \vec{r}^{(k+1)}\right) - \frac{\beta_k}{\alpha_k}\left(\vec{r}^{(k-1)} - \vec{r}^{(k)}\right)$$

$$\vec{r}^{(k+1)} = -\alpha_k \left(\boldsymbol{A}\vec{r}^{(k)} - \frac{1+\beta_k}{\alpha_k}\vec{r}^{(k)} - \frac{\beta_k}{\alpha_k}\vec{r}^{(k-1)} \right)$$

$$\boldsymbol{Q}^T \boldsymbol{A}\boldsymbol{Q} = \boldsymbol{T}_n \Rightarrow \boldsymbol{A}\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{T}_n$$

$$\boldsymbol{T}_n = \begin{pmatrix} \gamma_0 & \delta_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ \delta_1 & \gamma_1 & \delta_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & \delta_2 & \gamma_2 & \delta_3 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & \delta_{n-2} & \gamma_{n-2} & \delta_{n-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & \delta_{n-2} & \gamma_{n-1} \end{pmatrix}$$

$$\boldsymbol{Q} = (\vec{q}_0, \vec{q}_1, \dots, \vec{q}_{n-1})$$

$$\boldsymbol{A}\vec{q}_j = \delta_{j-1}\vec{q}_{j-1} + \gamma_j \vec{q}_j + \delta_j \vec{q}_{j+1}$$

$$\vec{q}_j \boldsymbol{A} \vec{q}_j = \gamma_j$$

δ_j found from normalization

$\vec{q}_0 = \text{random with norm 1}$ $\quad \vec{q}_{-1} = 0 \quad \delta_0 = 0$

For j = 0,1,...,n-2

$\vec{w}_j = \boldsymbol{A}\vec{q}_j$

$\gamma_j = \vec{w}_j \cdot \vec{q}_j$

$\vec{w}_j = \vec{w}_j - \gamma_j \vec{q}_j - \delta_j \vec{q}_{j-1}$

$\delta_{j+1} = \|\vec{w}_j\|$

$\vec{q}_{j+1} = \vec{w}_j / \delta_{j+1}$

End loop

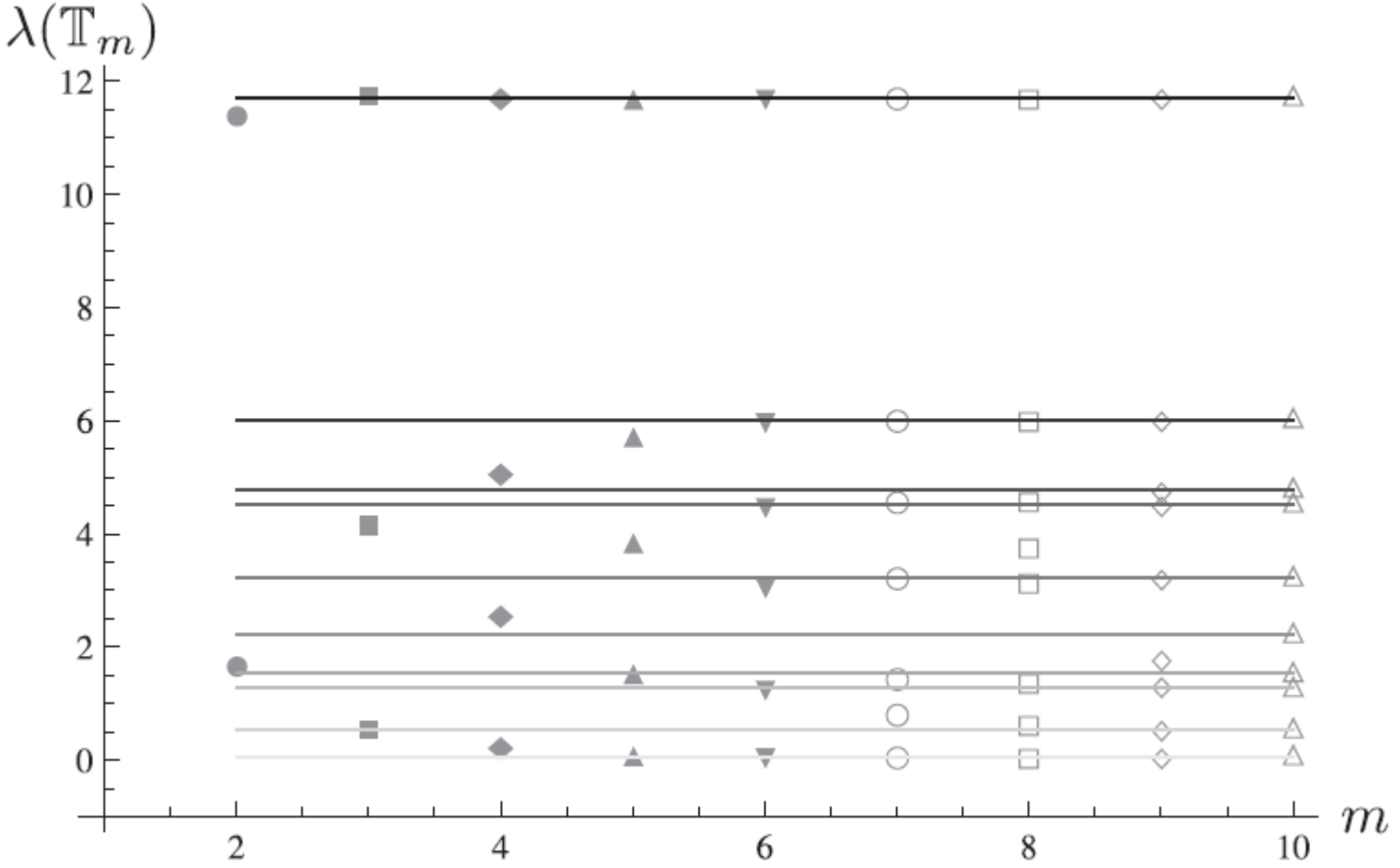
$\vec{w}_{n-1} = \boldsymbol{A}\vec{q}_{n-1} \quad \gamma_{n-1} = \vec{w}_{n-1} \cdot \vec{q}_{n-1}$

$$\boldsymbol{T}_n = \begin{pmatrix} \gamma_0 & \delta_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ \delta_1 & \gamma_1 & \delta_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & \delta_2 & \gamma_2 & \delta_3 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & \delta_{n-2} & \gamma_{n-2} & \delta_{n-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & \delta_{n-2} & \gamma_{n-1} \end{pmatrix}$$

Doing all n iterations is actually problematic: loss of orthogonality.

However, after only a few iterations the highest and lowest eigenvalues converge. In fact, loss of orthogonality occurs when convergence occurs.

To continue, orthogonalize new vectors with respect to the converged ones.



J. Franklin, Computational Methods for Physics, Cambridge University Press, 2013.

Matrix ***B***, 10×10, filled with random entries between -1 and 1. $\boldsymbol{B}\boldsymbol{B}^T$.

Generalizations to non-symmetric matrices

Arnoldi method

$\boldsymbol{Q}^T \boldsymbol{AQ} = \boldsymbol{H}$ (Hessenberg) $\qquad \boldsymbol{AQ} = \boldsymbol{QH} \qquad \boldsymbol{A}\vec{q}_j = \sum_{i=1}^{j+1} h_{ij}\vec{q}_i,\ 1 \le j \le n-1$

$h_{ij} = \vec{q}_i \boldsymbol{A}\vec{q}_j,\ i = 1,\ldots,j$

$\vec{q}_{j+1} = \vec{r}_{j+1} / \|\vec{r}_{j+1}\|$

$h_{j+1,j}\vec{q}_{j+1} = \boldsymbol{A}\vec{q}_j - \sum_{i=1}^{j} h_{ij}\vec{q}_i$

$\vec{r}_{j+1} = \boldsymbol{A}\vec{q}_j - \sum_{i=1}^{j} h_{ij}\vec{q}_i$

Start with arbitrary normalized $\vec{q}_k$

For *k*=2,3,...

$\vec{q}_k = \boldsymbol{A}\vec{q}_{k-1}$

For *j*=1,...,*k*-1

$h_{j,k-1} = \vec{q}_j \cdot \vec{q}_k$

$\vec{q}_k = \vec{q}_k - h_{j,k-1}\vec{q}_j$

End loop

$h_{k,k-1} = \|\vec{q}_k\|$

$\vec{q}_k = \vec{q}_k / h_{k,k-1}$

End loop

Explicit orthogonalization with respect to all previous vectors.

Back to solving linear systems. So far, assumed that we have as many equations as unknowns and the matrix is not degenerate, so there is a unique solution.

Underdetermined systems: fewer equations than unknowns. Usually infinitely many solutions.

Overdetermined systems: more equations than unknowns. Usually no solutions, but a solution can be found in the least-squares sense, i.e., minimize the norm of the residual.

$$f(\vec{x}) = (\boldsymbol{A}\vec{x} - \vec{b}) \cdot (\boldsymbol{A}\vec{x} - \vec{b}) = \vec{x} \cdot \boldsymbol{A}^T \boldsymbol{A} \cdot \vec{x} - 2\vec{x} \cdot \boldsymbol{A}^T \cdot \vec{b} + \vec{b} \cdot \vec{b}$$

$$\nabla f(\vec{x}) = 2\boldsymbol{A}^T \boldsymbol{A} \cdot \vec{x} - 2\vec{A}^T \vec{b} = 0$$ Method of normal equations

Works if the rank is equal to the number of columns.

A universal approach: singular value decomposition (SVD)

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$$

Matrix ***A*** is m×n (m>n), ***U*** is m×n, ***V*** is n×n, ***D*** is n×n and diagonal – its diagonal elements are singular values. ***U*** and ***V*** have orthonormal columns.

For square matrices (n×n):

Nullspace or kernel of ***A***: set of $\vec{y}$ such that $\boldsymbol{A}\vec{y} = 0.$

Range of ***A***: set of $\vec{c}$ such that $\exists \vec{x}: \ \boldsymbol{A}\vec{x} = \vec{c}$

Columns of ***U*** with nonzero corresponding elements of ***D*** span the range.

Columns of ***V*** with nonzero corresponding elements of ***D*** span the nullspace.