# Linear Algebra – brief review

Many good long textbooks

DO NOT CODE – use excellent free packages

Nonlinear fluids $\rightarrow$ many linear sub-problems,
e.g. Poisson problem, e.g. linear stability

Questions

- "matrix inversion": $Ax = b$
- eigenvalues: $Ae = \lambda e$

Matrices

- dense or sparse
- symmetric, positive definite, banded,...

# LAPACK

Free packages. Download library.

Search engine to find correct routine for you

- linear equations or linear least squares,
  or eigenvalues, singular decomposition, generalised
- precision: single/double, real/complex
- matrix type: symmetric, SPD, banded

Driver routine, calls computational routines, calls auxiliary (BLAS)

Real, single, general matrix, linear equations
SGESV($N, Nrhs, A, LDA, IPIV, B, LBD, info$)
where matrix $A$ is $N \times N$, with $Nrhs$ $b$'s in $B$.

# Solving linear simultaneous equations

1. Gaussian elimination

$$\begin{array}{ccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1 \\
a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2 \\
\vdots & & & & & & & & \vdots \\
a_{n1}x_1 & + & a_{n2}x_2 & + & \dots & + & a_{nn}x_n & = & b_n
\end{array}$$

Divide 1st eqn by $a_{11}$, so coef $x_1$ is 1

Subtract 1st eqn $\times a_{k1}$ from $k$th eqn, so coef $x_1$ becomes 0

Repeat on $(n-1) \times (n-1)$ subsystem of eqn $2 \to n$

Repeat on even smaller subsystems

Finally back-solve

$$\begin{array}{rcccc}
 & a_{nn}x_n & = b_n & \to & x_n \\
a_{n-1\,n-1}x_{n-1} + & a_{n-1\,n}x_n & = b_{n-1} & \to & x_{n-1} \\
 & \vdots & & & \\
 & & & \to & x_1
\end{array}$$

# LU decomposition – rephrase Gaussian elimination

Lower and Upper triangular

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ . & 1 & 0 & 0 \\ . & . & 1 & 0 \\ . & . & . & 1 \end{pmatrix} \qquad U = \begin{pmatrix} . & . & . & . \\ 0 & . & . & . \\ 0 & 0 & . & . \\ 0 & 0 & 0 & . \end{pmatrix}$$

Step $k = 1 \to n$:

$u_{kj} = a_{kj}$ for $j = k \to n$

$\ell_{ik} = a_{ik}/a_{kk}$ for $i = k \to n$

$a_{ij} \leftarrow a_{ij} - \ell_{ik} u_{kj}$ for $i = k+1 \to n$, for $j = k+1 \to n$

For a dense matrix $\frac{1}{3}n^3$ multiplies

For a tridiagonal matrix, avoiding zeros $2n$ multiplies

Solve $LUx = b$ by

Forward $Ly = b$

$$\begin{array}{ccccccc} \ell_{11} y_1 & & & = & b_1 & \rightarrow & y_1 \\ \ell_{21} y_1 & + & \ell_{22} y_2 & = & b_1 & \rightarrow & y_2 \\ & & & & & \vdots & \\ & & & & & \rightarrow & y_n \end{array}$$

Backward $Ux = y$

$$\begin{array}{ccccccc} & & u_{nn} x_n & = & y_n & \rightarrow & x_n \\ u_{n-1\,n-1} x_{n-1} & + & u_{n-1\,n} x_n & = & y_{n-1} & \rightarrow & x_{n-1} \\ & & & & & \vdots & \\ & & & & & \rightarrow & x_1 \end{array}$$

Finding $LU$ is $O(n^3)$
but solving $LUx = b$ for a new $b$ is only $O(n^2)$

# LU: pivoting

Problem at step $k$ if $a_{kk} = 0$
Find largest $a_{jk}$ in $j = k \to n$, say at $j = \ell$
Swap rows $k$ and $\ell$ – use index mapping (permutation matrix)

Partial pivoting = swapping rows
Full pivoting = swap rows and columns – rarely better

- Note $\det A = \Pi_i u_{ii}$
- Symmetric $A$: $A = LDL^T$ with diagonal $D$
- Sym & positive definite: $A = (LD^{1/2})(LD^{1/2})^T$ Cholesky
- Tridiagonal $A$: $L$ diagonal and one under, $U$ diagonal and one above.

# Errors $Ax = b$

Small $\epsilon$ error in $b$ could become $\epsilon/\lambda_{\min}$ error in solution,
while worst solution is $b/\lambda_{\max}$
Thus relative error in solution could increase by factor

$$K = \frac{\lambda_{\max}}{\lambda_{\min}} = \text{condition number of } A$$

Theoretically $LU$ decomposition gives bigger errors, but not often

# $QR$ decomposition

$$A = QR$$

- $R$ upper triangular
- $Q$ orthogonal, $QQ^T = I$, i.e. columns orthonormal
  So at no cost $Q^{-1} = Q^T$
- May not stretch/increase errors like $LU$
- Used for eigenvalues
- $\det A = \Pi_i r_{ii}$

$Q$ not unique

3 methods: Gram-Schmidt, Givens, Householder

# $QR$ Gram-Schmidt

Columns of $A$ $\quad \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$

$$
\begin{array}{llllll}
\mathbf{q}_1' & = \mathbf{a}_1 & & & \mathbf{q}_1 = \mathbf{q}_1'/|\mathbf{q}_1'| \\
\mathbf{q}_2' & = \mathbf{a}_2 & -(\mathbf{a}_2 \cdot \mathbf{q}_1)\mathbf{q}_1 & & \mathbf{q}_2 = \mathbf{q}_2'/|\mathbf{q}_2'| \\
\mathbf{q}_3' & = \mathbf{a}_3 & -(\mathbf{a}_3 \cdot \mathbf{q}_1)\mathbf{q}_1 & -(\mathbf{a}_3 \cdot \mathbf{q}_2)\mathbf{q}_2 & \mathbf{q}_3 = \mathbf{q}_3'/|\mathbf{q}_3'| \\
\vdots & & & &
\end{array}
$$

$Q =$ matrix with columns $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n$

Let

$$r_{ii} = |\mathbf{q}_i'|, \quad \text{and} \quad r_{ij} = \mathbf{a}_j \cdot \mathbf{q}_i, \quad i < j$$

Then

$$\mathbf{a}_j = \sum_{i=1}^{j} \mathbf{q}_j r_{ij} \qquad \text{i.e.} \quad A = QR$$

Better: when produce $\mathbf{q}_i$ project it out of $\mathbf{a}_j$ $j > i$

# $QR$ Givens rotation

$Q =$ product of many rotations

$$G_{ij} = \begin{pmatrix} & & i & & & j & & \\ 1 & & & & & & & \\ & 1 & & & & & & \\ i & & \cos\theta & & & \sin\theta & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ j & & -\sin\theta & & & \cos\theta & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{pmatrix}$$

$G_{ij}A$ alters rows and columns $i$ and $j$

Choose $\theta$ to zero an off-diagonal

  Strategy to avoid filling previous zeros

Can parallelise

# $QR$ Householder

$Q =$ product of many reflections

$$H = \left( I - 2\frac{\mathbf{h}\mathbf{h}^T}{\mathbf{h}\cdot\mathbf{h}} \right)$$

Take $\quad \mathbf{h}_1 = \mathbf{a}_1 + (\alpha_1, 0, \ldots, 0)^T$ with $\alpha_1 = |\mathbf{a}_1| sign(a_{11})$

So

$$\mathbf{h}_1 \cdot \mathbf{a}_1 = |\mathbf{a}_1|^2 + |a_{11}||\mathbf{a}_1| \quad \text{and} \quad \mathbf{h}_1 \cdot \mathbf{h}_1 = \text{twice}$$

Hence

$$H_1 \mathbf{a}_1 = (-\alpha_1, 0, \ldots, 0)^T$$

Now work on $(n-1) \times (n-1)$ subsystem in same way

Note $H\mathbf{x}$ is $O(n)$ operations, not $O(n^2)$

Hence forming $Q$ is $O(n^3)$

# Sparse matrices

Do not store all $A$, just non-zero elements in "packed" form

Evaluating $A\mathbf{x}$ cheaper than $O(n^2)$
e.g. Poisson on $N \times N$ grid, $A$ is $N^2 \times N^2$ with $5N^2$ non-zero, so $A\mathbf{x}$ is $5N^2$ not $N^4$

$LU$ and $QR$ "direct methods" for dense (faster if banded)

Use iterative method for sparse $A$
i.e.

$$A = B + C \quad \rightarrow \quad \mathbf{x}_{n+1} = B^{-1}(\mathbf{b} - C\mathbf{x}_n)$$

converges if $|B^{-1}C| < 1$, e.g. SOR

# Conjugate gradients – $A$ symmetric, positive definite

– actually a direct method, but usually converges well before $n$ steps

Solve $Ax = b$ by minimising quadratic

$$f(x) = \frac{1}{2}(Ax - b)^T A^{-1}(Ax - b) = \frac{1}{2}x^T Ax - x^T b + \frac{1}{2}b^T Ab$$

with

$$\nabla f = Ax - b$$

From $\mathbf{x}_n$ look in direction $\mathbf{u}$ for minimum

$$f(\mathbf{x}_n + \alpha\mathbf{u}) = f(\mathbf{x}_n) + \alpha\mathbf{u} \cdot \nabla f_n + \frac{1}{2}\alpha^2\mathbf{u}^T A\mathbf{u}$$

i.e. minimum at $\alpha = -\mathbf{u} \cdot \nabla f_n / \mathbf{u}^T A\mathbf{u}$

Choose $\mathbf{u}$? steepest descent $\mathbf{u} = \nabla f$? NO

# GC not steepest descent $\nabla f$

Steepest descent $\rightarrow$ rattle from side to side across steep valley with no movement along the valley floor

Need new direction $\mathbf{v}$ which does not reset $\mathbf{u}$ minimisation

$$f(\mathbf{x}_n + \alpha\mathbf{u} + \beta\mathbf{v}) = f(\mathbf{x}_n) + \alpha\mathbf{u}\cdot\nabla f_n + \tfrac{1}{2}\alpha^2\mathbf{u}^T A\mathbf{u}$$
$$+\alpha\beta\mathbf{u}^T A\mathbf{v} + \beta\mathbf{v}\cdot\nabla f_n + \tfrac{1}{2}\beta^2\mathbf{v}^T A\mathbf{v}$$

Hence need $\mathbf{u}^T A\mathbf{v} = 0$ "conjugate directions"

# Conjugate Gradient Algorithm

Start $x_0$ and $u_0$
Residual $r_n = Ax_n - b = \nabla f_n$
Iterate

$$x_{n+1} = x_n + \alpha u_n$$
$$r_{n+1} = r_n + \alpha A u_n$$
$$u_{n+1} = r_{n+1} + \beta u_n$$

Minimising $\alpha = -\frac{u_n^T r_n}{u_n^T A u_n}$

Conj grad $\beta = -\frac{r_{n+1}^T A u_n}{u_n^T A u_n}$

Note only one matrix evaluation per iteration – good sparse

Can show $u_{n+1}$ conjugate all $u_i$ $i = 1, 2, \ldots, n$

Can show $\alpha = \frac{r_n^T r_n}{u_n^T A u_n}, \qquad \beta = \frac{r_{n+1}^T r_{n+1}}{r_n^T r_n}$

## Precondition

$Ax = b$ same solution as $B^{-1}Ax = B^{-1}b$
Choose $B$ with easy inverse and $B^{-1}A$ sparse
Typical $ILU$ = incomplete $LU$, few large elements

## Non-symmetric $A$

GMRES minimises $(Ax - b)^T(Ax - b)$
– but condition number $K^2$
GMRES($n$) restart after $n$ – avoids large storage

## If tough, then $SVD$ = singular value decomposition

$$A = USV = \sum_i u_i^T \lambda_i v_i$$

with $v$ and $u$ eigenvectors and adjoints, $\lambda_i$ eigenvalues

# Eigenproblems $Ae = \lambda e$ and generalised $Ae = \lambda Be$

- No finite/direct method – must iterate
- $A$ real & symmetric – nice orthogonal evectors
- $A$ not symmetric – possible degenerate cases
  also non-normal modes (& pseud-spectra. . . )

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -1 & k^2 \\ 0 & -1-k \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \quad \text{IC} \quad x(0) = 0,\ y(0) = 1$$

has solution $x = k(e^{-t} - e^{(1+k)t})$
which eventually decays but before is $k$ larger than IC.

Henceforth $A$ real and symmetric

# Power iteration – for largest evalue

Start random $x_0$
Iterate a few times $x_{n+1} = Ax_n = A^n x_0$

$x_n$ becomes dominated by evector with largest evalue, so

$$\lambda_{\mathrm{approx}} = |Ax_n|/|x_n|, \qquad e_{\mathrm{approx}} = Ax_n/|Ax_n|$$

With this crude approximation invert

$$(A - \lambda_{\mathrm{approx}} I)^{-1}$$

which has one very large evalue $\quad 1/(\lambda_{\mathrm{correct}} - \lambda_{\mathrm{approx}})$,
so power iteration on this converges very rapidly

Find other evalues with $\mu$-shifts $(A - \mu I)^{-1}$

# Jacobi – small $A$ only

Find maximum off-diagonal $a_{ij}$

Givens rotation $G_{ij}$ with $\theta$ to zero $a_{ij}$, and $a_{ji}$ by symmetry

$$A' = GAG^T \quad \text{has same evalues}$$

Does fill in previous zeros,
but sum of off-diagonals squared decreases by $a_{ij}^2$

Hence converges to diagonal (=evalues) form

# Main method

Step 1: reduce to Hessenberg $H$, upper triangular plus one below diagonal

Arnoldi (GS on Kyrlov space $q_1, Aq_1, A^2q_1, \ldots$)
Given unit $q_1$, step $k = 1 \to n-1$

$$v = Aq_k$$
$$\text{for } j = 1 \to k\text{: } H_{jk} = q_j \cdot v;\ v \leftarrow v - H_{jk}q_j$$
$$H_{k+1\,k} = |v|$$
$$q_{k+1} = v/H_{k+1\,k}$$

Hence

original $\quad v = Aq_k = H_{k+1\,k}q_{k+1} + H_{kk}q_k + \ldots + H_{1k}q_1$

i.e. $\quad A\,(q_1, q_2, \ldots, q_n) = (q_1, q_2, \ldots, q_n)\,H$

i.e. $\quad AQ = QH \quad$ or $\quad H = Q^T AQ \quad$ with same evalues as $A$

Cost $O(n^2)$ if dense

$$H = Q^T A Q \quad \text{Hessenberg}$$

$A$ symmetric $\to$ $H$ symmetric, hence tridiagonal

Hence reduce 'for $j = 1 \to k$' to 'for $j = k-1, k$',

Cost $\to O(n^2)$ (Lanzcos)

NB: making $q_{k+1}$ orthogonal to $q_k$ & $q_{k-1}$

gives $q_{k+1}$ orthogonal to $q_j$ $j = k, k-1, k-2, \ldots, 1$

cf conjugate gradient

# Main method, step 2

a. $QR$ Find $QR$ decomposition of $H$
Set $H' = RQ = Q^T AQ$
– remains Hessenberg/Tridiagonal
– off-diagonals reduced by $\lambda_i/\lambda_j$
$\rightarrow$ converges to diagonal, of evalues

b. Power iteration – quick when tridiagonal

c. Root solve $\det(A - \lambda I) = 0$ – quick if tridiagonal

BUT USE PACKAGES