MATH50003 Numerical Analysis

III.1 Structured Matrices

Dr Sheehan Olver

Part III

Numerical Linear Algebra

square

1. Structured matrices such as banded
2. LU and PLU factorisations for solving linear systems
3. Cholesky factorisation for symmetric positive definite
4. Orthogonal matrices such as Householder reflections
5. QR factorisation for solving least squares

rectangular

III.1.1 Dense matrices

And their usage in matrix multiplication

Consider a matrix $\mathrm{A} \in \mathbb{F}^{m \times n}$ where $\mathbb{F}$ is a field ($\mathbb{R}$ or $\mathbb{C}$)

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = [\vec{a}_1 | \cdots | \vec{a}_n]$$

where $\vec{a}_k \in \mathbb{F}^m$

And a vector $\mathbf{x} \in \mathbb{F}^n$. We have ("multiplication by rows")

$$A\boldsymbol{x} := \begin{bmatrix} \sum_{j=1}^n a_{1j} x_j \\ \vdots \\ \sum_{j=1}^n a_{mj} x_j \end{bmatrix}$$

Or with floats: $$A\boldsymbol{x} \approx \begin{bmatrix} \bigoplus_{j=1}^n (a_{1j} \otimes x_j) \\ \vdots \\ \bigoplus_{j=1}^n (a_{mj} \otimes x_j) \end{bmatrix}$$

We can also write a matrix in terms of its columns:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = \left[\boldsymbol{a}_1 | \cdots | \boldsymbol{a}_n\right]$$

$$A\vec{x} = [\vec{a}_1 | \text{—} | \vec{a}_n] \begin{bmatrix} x_1 \\ \downarrow \\ x_n \end{bmatrix}$$

Alternative formula for multiplication (“multiplication by columns”):

$$A\boldsymbol{x} = x_1\boldsymbol{a}_1 + \cdots + x_n\boldsymbol{a}_n$$

Equivalent to mult-by-rows,

Even w/ floats.

But mystery: mult by columns is 4x faster.
See lab.

Computational complexity

How many floating point operations? Count the number of $\oplus$, $\otimes$, $\oslash$, $\ominus$

n-1 adds

1 mult, n times

$$A\boldsymbol{x} := \begin{bmatrix} \sum_{j=1}^n a_{1j} x_j \\ \vdots \\ \sum_{j=1}^n a_{mj} x_j \end{bmatrix} \approx \begin{bmatrix} \bigoplus_{j=1}^n (a_{1j} \otimes x_j) \\ \vdots \\ \bigoplus_{j=1}^n (a_{mj} \otimes x_j) \end{bmatrix}$$

Each has $n + n-1 = 2n-1$ operations
There are m rows, so

$m(2n-1)$ operations

$= O(nm)$ operations.

Eg for square matrices:

$A\tilde{x}$ take $O(n^2)$ operations (quadratic)

Doubling n increases cost $4\times$.

III.1.2 Triangular Matrices

Exploiting zero structure in a matrix

$$U = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ & \ddots & \vdots \\ & & u_{nn} \end{bmatrix}, \qquad L = \begin{bmatrix} \ell_{11} & & \\ \vdots & \ddots & \\ \ell_{n1} & \cdots & \ell_{nn} \end{bmatrix}$$

Multiplication takes roughly half the operations (still same complexity):

$$L\tilde{x} = \begin{bmatrix} l_{11} x_1 \\ l_{21} x_1 + l_{22} x_2 \\ \vdots \\ \sum_{j=1}^{n} l_{nj} x_j \end{bmatrix}$$

Now we have

1 mult +

2 mult + 1 add

3 mult + 2 add

|

n mult + n-1 add

$$= \sum_{k=1}^{n} (2k-1) \text{ operations}$$

$$= n^2 + 2n + 2 = O(n^2) \text{ operations}$$

(same quadratic complexity)

How to compute $A^{-1}\vec{b}$?

Really easy if triangular.

Can invert via back-substitution/forward ~~elimination~~ substitution:

Solve $L\vec{x} = \vec{b}$, ie

$$\begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$\Rightarrow$ Row 1 $\quad l_{11} x_1 = b_1 \Rightarrow x_1 = b_1 / l_{11}$

Row 2 $\quad l_{21} x_1 + l_{22} x_2 = b_2 \Rightarrow x_2 = \dfrac{b_2 - l_{21} x_1}{l_{22}}$

$\vdots$

(Row n) $\quad x_n = \dfrac{b_n - \sum_{j=1}^{n-1} l_{nj} x_j}{\quad}$

ℓ_{nn}

Total: $\sum_{k=1}^{n} O(k) = O(n^2)$ operations (quadratic)

III.1.3 Banded Matrices

Matrices that are only non-zero near the diagonal

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1,u+1} & & & \\ \vdots & a_{22} & \ddots & a_{2,u+2} & & 0 \\ a_{1+l,1} & \ddots & \ddots & \ddots & \ddots & \\ & a_{2+l,2} & \ddots & \ddots & \ddots & a_{n-u,n} \\ 0 & & \ddots & \ddots & \ddots & \vdots \\ & & & a_{n,n-l} & \cdots & a_{nn} \end{bmatrix}$$

l, u are lower/upper bandwidths.

Definition 13 (Bidiagonal)**.** If a square matrix has bandwidths $(l, u) = (1, 0)$ it is *lower-bidiagonal* and if it has bandwidths $(l, u) = (0, 1)$ it is *upper-bidiagonal*.

$$L = \begin{bmatrix} \ell_{11} & & & 0 \\ \ell_{21} & \ell_{22} & & \\ & \ddots & \ddots & \\ 0 & & \ell_{n,n-1} & \ell_{nn} \end{bmatrix} \qquad U = \begin{bmatrix} u_{11} & u_{12} & & \\ & u_{22} & \ddots & \\ & & \ddots & u_{n-1,n} \\ & & & u_{nn} \end{bmatrix}$$

Multiplication is linear complexity:

$$L\vec{x} = \begin{bmatrix} l_{11} x_1 \\ l_{21} x_1 + l_{22} x_2 \\ l_{32} x_2 + l_{33} x_3 \\ \vdots \\ l_{n,n-1} x_{n-1} + l_{nn} x_n \end{bmatrix}$$

$1 + (n-1)(3) = O(n)$ operations (linear complexity)

Doubling n doubles cost.

Back substitution/forward ~~elimination~~ substitution are also linear complexity:

$L\vec{x} = \vec{b} \Leftrightarrow$

$$\begin{bmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & & \\ & l_{32} & l_{33} & & \\ & & \ddots & \ddots & \\ & & & l_{n,n-1} & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Then

$$x_1 = \frac{b_1}{l_{11}}$$

$$x_2 = \frac{b_2 - l_{21} x_1}{l}$$

l_{22}

$$x_3 = \frac{b_3 - l_{32} x_2}{l_{33}}$$

$$\vdots$$

$$x_n = \frac{b_n - l_{n,n-1} x_{n-1}}{l_{nn}}$$

Total $\sum_{k=1}^{n} O(1) = O(n)$ operations (linear).

Definition 14 (Tridiagonal)**.** If a square matrix has bandwidths $l = u = 1$ it is *tridiagonal.*

$$A = \begin{bmatrix} a_{11} & a_{12} & & & \\ a_{21} & a_{22} & a_{23} & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ & & & a_{n,n-1} & a_{nn} \end{bmatrix}$$

Multiplication is linear complexity.
We will see later inversion is also linear complexity.

Compare w/ dense A:

multiplication is quadratic $O(n^2)$

$\underbrace{\text{linear solves}}$ is cubic $O(n^3)$ (to be seen)

HOW?