MATH50003 Numerical Analysis

III.1 Structured Matrices

Dr Sheehan Olver

Part III

Numerical Linear Algebra

Software
Application
Theory

1. Structured matrices such as banded
2. Differential Equations via finite differences
3. LU and Cholesky factorisation for solving linear systems
4. Polynomial regression for approximating data via least squares
5. Orthogonal matrices such as Householder reflections
6. QR factorisation for solving rectangular least squares problems

III.1.1 Dense matrices

And their usage in matrix multiplication

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

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

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:

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

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

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

Computational complexity

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

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

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):

Can invert via back-substitution/forward elimination:

III.1.3 Banded Matrices

Matrices that are only non-zero near the diagonal

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

Definition 16 (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.*

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

Multiplication is linear complexity:

Back substitution/forward elimination are also linear complexity:

Definition 17 (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.