Math 4: Numerical Linear Algebra

Nathan Warner

Northern Illinois University

Computer Science
Northern Illinois University
United States

Contents

Numerical Linear Algebra

1.1 Introduction

- **Matrix Notation**: For a matrix $A \in \mathbb{R}^{m\times n}$, we say

$$A = (a_{ij}) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

  with $a_{ij} \in \mathbb{R}$.

- **Vector notation**: For a vector $x \in \mathbb{R}^n$ (or $\mathbb{R}^{n\times 1}$), we have

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

  for $x_i \in \mathbb{R}$.

- **Submatrix notation (rows)**:

$$A(i,:) \in \mathbb{R}^{1\times n} \iff A(i,:) = \begin{bmatrix} a_{i1} \; a_{i2} \; \cdots \; a_{in} \end{bmatrix}.$$

- **Submatrix notation (columns)**:

$$A(:,j) \in \mathbb{R}^{m\times 1} \iff A(:,j) = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix}.$$

- **Sparse Matrix**: A sparse matrix or sparse array is a matrix in which most of the elements are zero. There is no strict definition regarding the proportion of zero-value elements for a matrix to qualify as sparse but a common criterion is that the number of non-zero elements is roughly equal to the number of rows or columns.

- **Dense Matrix**: if most of the elements are non-zero, the matrix is considered dense

- **Sparsity**: The number of zero-valued elements divided by the total number of elements is sometimes referred to as the sparsity of the matrix.

- **Band Matrix**: a band matrix or banded matrix is a sparse matrix whose non-zero entries are confined to a diagonal band, comprising the main diagonal and zero or more diagonals on either side.

$$A(i_1:i_2,:) \in \mathbb{R}^{(i_2-i_1+1)\times n} \iff A(i_1:i_2,:) = \begin{bmatrix} a_{i_1 1} & a_{i_1 2} & \cdots & a_{i_1 n} \\ a_{i_1+1,1} & a_{i_1+1,2} & \cdots & a_{i_1+1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i_2 1} & a_{i_2 2} & \cdots & a_{i_2 n} \end{bmatrix}.$$

$$A(:,j_1:j_2) \in \mathbb{R}^{m\times(j_2-j_1+1)} \iff A(:,j_1:j_2) = \begin{bmatrix} a_{1j_1} & a_{1,j_1+1} & \cdots & a_{1j_2} \\ a_{2j_1} & a_{2,j_1+1} & \cdots & a_{2j_2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{mj_1} & a_{m,j_1+1} & \cdots & a_{mj_2} \end{bmatrix}.$$

Where

$$\begin{aligned} A(i_1:i_2,:) &: \text{ all rows between } i_1 \text{ and } i_2, \text{ across all columns,} \\ A(:,j_1:j_2) &: \text{ all columns between } j_1 \text{ and } j_2, \text{ across all rows.} \end{aligned}$$

- **Transposition**: $\mathbb{R}^{m\times n} \to \mathbb{R}^{n\times m}$

$$C = A^\top \iff c_{ij} = a_{ji}.$$

- **Addition** $(\mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n})$

$$C = A + B \implies c_{ij} = a_{ij} + b_{ij}.$$

- **Scalar-matrix Multiplication:** $(\mathbb{R} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n})$

$$C = \alpha A \implies c_{ij} = \alpha a_{ij}.$$

- **Matrix-matrix Multiplication:** $(\mathbb{R}^{m\times p} \times \mathbb{R}^{p\times n} \to \mathbb{R}^{m\times n})$

$$C = AB \implies c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}.$$

- **Matrix-vector Multiplication:** $(\mathbb{R}^{m\times n} \times \mathbb{R}^{n} \to \mathbb{R}^{m})$

$$y = Ax \implies y_i = \sum_{j=1}^{n} a_{ij} x_j.$$

- **Inner product (or dot product):** $(\mathbb{R}^{n} \times \mathbb{R}^{n} \to \mathbb{R})$

$$c = x^T y \implies c = \sum_{i=1}^{n} x_i y_i.$$

- **Outer product:** $(\mathbb{R}^{m} \times \mathbb{R}^{n} \to \mathbb{R}^{m\times n})$

$$C = xy^T \implies c_{ij} = x_i y_j.$$

- **Flops**: A flop is a floating-point operation between numbers stored in a floating-point format on a computer.

  If x and y are numbers stored in a floating point format, then the following operations are each one flop

$$x + y \quad x - y \quad xy \quad x/y.$$

1.2 Gaussian Elimination and its variants

1.2.1 Matrix Multiplication

- **Matrix Multiplication**: In general, if A is a real matrix with m rows and n columns, and x is a real vector with n entries, then

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \in \mathbb{R}^{m\times n} \quad \text{and} \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n.$$

If $b = Ax$, then $b \in \mathbb{R}^m$ and

$$b_i = \sum_{j=1}^{n} a_{ij}x_j = a_{i1}x_1 + \cdots + a_{in}x_n, \quad i = 1, \ldots, m.$$

Thus, b_i is the **inner-product** between the i-row of A,

$$A(i,:) = [a_{i1} \ \ \cdots \ \ a_{in}], \quad (i = 1, \ldots, m)$$

and the vector x.

Also,

$$b = A(:,1)x_1 + \cdots + A(:,n)x_n,$$

so b is a **linear combination** of the columns of A, i.e.,

$$A(:,j) = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix}, \quad j = 1, \ldots, n.$$

- **Matrix-Matrix Multiplication**: Let $A \in \mathbb{R}^{m\times n}$ and $X \in \mathbb{R}^{n\times p}$.

If $B = AX$ then $B \in \mathbb{R}^{m\times p}$ and

$$b_{ij} = \sum_{k=1}^{n} a_{ik}x_{kj}, \quad i = 1, \ldots, m, \ \ j = 1, \ldots, p.$$

That is, b_{ij} is the inner-product between row i of A and column j of X.

Also, each column of B is a linear combination of the columns of A.

Total flops required for matrix multiplication is

$$\sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{n} 2 = 2mnp.$$

If $A, X \in \mathbb{R}^{n\times n}$, then computing $B = AX$ requires $2n^3 = O(n^3)$ flops.

We can see this by describing the algorithm for Matrix-Matrix multiplication

```
for i = 1:m
    for j = 1:n
        for k = 1:p
            C[i,j] += A[i,k]B[k,j]
        end
    end
end
```

The multiplication $A[i,j]B[k,j]$ is one flop, followed by the addition. Therefore, two flops per iteration of the innermost loop.

- **Block Matrices**: Partition $A \in \mathbb{R}^{m\times n}$ and $X \in \mathbb{R}^{n\times p}$ into blocks:

$$A = \begin{array}{c c} & \begin{array}{cc} n_1 & n_2 \end{array} \\ \begin{array}{c} m_1 \\ m_2 \end{array} & \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \end{array}, \qquad X = \begin{array}{c c} & \begin{array}{cc} p_1 & p_2 \end{array} \\ \begin{array}{c} n_1 \\ n_2 \end{array} & \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \end{array}$$

where $n = n_1 + n_2$, $m = m_1 + m_2$, and $p = p_1 + p_2$.

If $B = AX$, and

$$B = \begin{array}{c c} & \begin{array}{cc} p_1 & p_2 \end{array} \\ \begin{array}{c} m_1 \\ m_2 \end{array} & \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \end{array},$$

then

$$\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = B = AX = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$$

$$= \begin{bmatrix} A_{11}X_{11} + A_{12}X_{21} & A_{11}X_{12} + A_{12}X_{22} \\ A_{21}X_{11} + A_{22}X_{21} & A_{21}X_{12} + A_{22}X_{22} \end{bmatrix}$$

That is,

$$B_{ij} = \sum_{k=1}^{2} A_{ik}X_{kj}, \qquad i,j = 1,2.$$

1.2.2 Systems of Linear Equations

- **Systems of linear equations**: Let $A \in \mathbb{R}^{n\times n},\ b \in \mathbb{R}^m$, our goal is to find $x \in \mathbb{R}^m$ such that $Ax = b$
- **Singularity**: A **singular matrix** is a square matrix that does not have an inverse.

  A **nonsingular** matrix is a square matrix that does have an inverse.

  The following are equivalent

  - $Ax = b$ has a unique solution
  - $\det(A) \neq 0$
  - A^{-1} exists
  - There is no nonzero vector $y \in \mathbb{R}^m$ such that $Ay = 0$

  If any one of the following are true, they all are true, and A is non-singular
- **Solution to** $Ax = b$: If A is nonsingular, then A^{-1} exists, and

$$x = A^{-1}b.$$

  Which is the unique solution to $Ax = b$

  Note: Practically, it is not wise to compute A^{-1}, as this can be expensive.

1.2.3 Triangular systems

- **Upper triangular**: A square matrix $A = U$ of the form

$$A = U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

  is called **upper triangular**

- **Lower triangular**: A square matrix $A = L$

$$A = L = \begin{bmatrix} \ell_{11} & 0 & \cdots & 0 \\ \ell_{21} & \ell_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{nn} \end{bmatrix}$$

  is called **lower triangular**

- **Solutions to triangular systems**: Consider the system

$$\begin{bmatrix} \ell_{11} & 0 & \cdots & 0 \\ \ell_{21} & \ell_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

  So,

$$\begin{aligned} \ell_{11}x_1 &= b_1 \\ \ell_{21}x_1 + \ell_{22}x_2 &= b_2 \\ &\vdots \\ \ell_{n1}x_1 + \ell_{n2}x_2 + \cdots + \ell_{nn}x_n &= b_n. \end{aligned}$$

  Then,

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

  and,

$$\begin{aligned} \ell_{22}x_2 &= b_2 - \ell_{21}x_1 \\ \implies x_2 &= \frac{b_2 - \ell_{21}x_1}{\ell_{22}}. \end{aligned}$$

  In general, we have

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} \ell_{ij}x_j}{\ell_{ii}}$$

  for $i = 1, 2, ..., n$. This method is called **Forward Substitution**.

  A similar process is used on upper triangular matrices and is called **Backward Substitution**.

- **Counting flops for the forward substitution method**: We have

```
for i = 1:n
    for j=1:i-1
        b[i] = b[i] - ell[i,j]b[j]
    end
    b[i] = b[i] / ell[i,i]
end
```

Thus, the count of flops is

$$\begin{aligned} n + \sum_{i=1}^{n} 2(i-1) = n + 2\sum_{i=1}^{n}(i-1) &= n + 2\left(\sum_{i=1}^{n} i - \sum_{i=1}^{n} 1\right) \\ &= n + 2\left(\sum_{i=1}^{n} i - n\right) = n + 2\left(\frac{n(n+1)}{2} - n\right) \\ &= n + n^2 - n = n^2 \end{aligned}$$

So, forward substitution is $\mathcal{O}(n^2)$

- **Column oriented forward substitution**: Suppose we have $Lx = b$ when L is lower triangular, we split the matrix into the following blocks

$$\begin{bmatrix} \ell_{11} & 0 \\ \hat{\ell} & \hat{L} \end{bmatrix} \begin{bmatrix} x_1 \\ \hat{x} \end{bmatrix} = \begin{bmatrix} b_1 \\ \hat{b} \end{bmatrix}.$$

With $\hat{\ell} \in \mathbb{R}^{n-1}$, $\hat{L} \in \mathbb{R}^{n-1 \times n-1}$, $\hat{x} \in \mathbb{R}^{n-1}$, $\ell_{11}, x_1, b_1 \in \mathbb{R}$. Note that $\hat{L}$ is also lower triangular.

We have

$$\begin{aligned} \ell_{11}x_1 = b_1 &\implies x_1 = \frac{b_1}{\ell_{11}} \\ \hat{\ell}x_1 + \hat{L}\hat{x} = \hat{b} &\implies \hat{L}\hat{x} = \hat{b} - \hat{\ell}x_1 \end{aligned}$$

Thus, we reduced the dimension by one. We repeat this process for the remaining x_i. The process is

1. Compute $x_1 = \frac{b_1}{\ell_{11}}$
2. Compute $\hat{b} - \hat{\ell}x_1 = \tilde{b} \in \mathbb{R}^{n-1}$
3. Find $\hat{L}x = \tilde{b}$

- **Counting flops for column oriented forward substitution**: Let f_n be the flop count, we have

$$f_n = 1 + 2(n-1) + f_{n-1}.$$

With

$$f_{n-1} = 1 + 2(n-2) + f_{n-2}.$$

Until

$$f_1 = 1 + 2 + f_0$$

with $f_0 = 0$

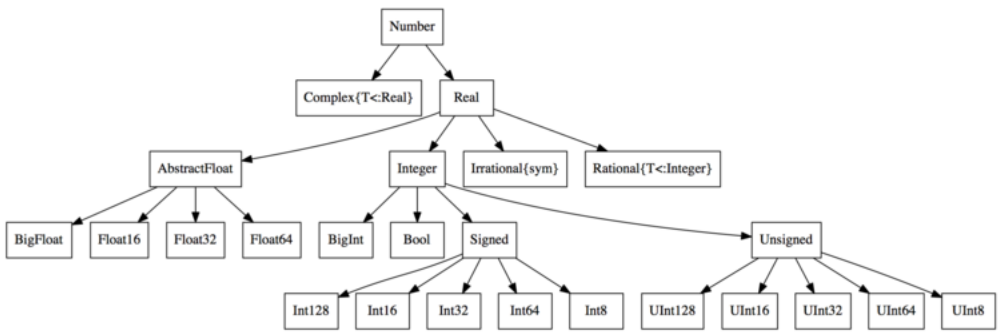
So,

$$
\begin{aligned}
f_n &= 1 + 2(n-1) + 1 + 2(n-2) + ... + 1 + 2(n-(n-1)) \\
&= \sum_{i=1}^{n-1} 1 + 2(n-1) = n + 2n^2 - 2\sum_{i=1}^{n} i \\
&= ... = n^2.
\end{aligned}
$$

Thus, column oriented forward substitution is also $\mathcal{O}(n^2)$

Julia

2.1 Types



- **Subtype constraint <:** $A <: B$ means A is a subtype of B

```
Int <: Number #true
```

2.2 Functions

2.3 Linear Algebra

2.3.1 Matrix creation and operations

- **Array constructors**:
  - Array{T}(undef, dims...)
  - Vector{T}(undef, n)
  - Matrix{T}(undef, m, n)
- **Zeros/ones/fills**
  - zeros(n), zeros(m,n)
  - ones(n), ones(m,n)
  - fill(x, dims...)
- **Uniform ranges**:
  - collect(1:n) → vector
  - collect(1:m, 1:n) → matrix (grid)