Unit II: Numerical Linear Algebra

Chapter II.1: Motivation
Almost everything in Scientific Computing relies on Numerical Linear Algebra!

We often reformulate problems as $Ax = b$, e.g. from Unit I:

- Interpolation (Vandermonde matrix) and linear least-squares (normal equations) are naturally expressed as linear systems
- Gauss-Newton/Levenberg-Marquardt involve approximating nonlinear problem by a sequence of linear systems

Similar themes will arise in remaining Units (Numerical Calculus, Optimization, Eigenvalue problems)
The goal of this Unit is to cover:

- key linear algebra concepts that underpin Scientific Computing
- algorithms for solving $Ax = b$ in a stable and efficient manner
- algorithms for computing factorizations of $A$ that are useful in many practical contexts (QR, SVD)

First, we discuss some practical cases where $Ax = b$ arises directly in mathematical modeling of physical systems.
Example: Electric Circuits

Ohm’s Law: Voltage drop due to a current $i$ through a resistor $R$ is $V = iR$

Kirchoff’s Law: The net voltage drop in a closed loop is zero
Example: Electric Circuits

Let $i_j$ denote the current in “loop $j$”

Then, we obtain the linear system:

\[
\begin{bmatrix}
(R_1 + R_3 + R_4) & R_3 & R_4 \\
R_3 & (R_2 + R_3 + R_5) & -R_5 \\
R_4 & -R_5 & (R_4 + R_5 + R_6)
\end{bmatrix}
\begin{bmatrix}
i_1 \\
i_2 \\
i_3
\end{bmatrix}
= 
\begin{bmatrix}
V_1 \\
V_2 \\
0
\end{bmatrix}
\]

Circuit simulators solve large linear systems of this type
Common in structural analysis to use a linear relationship between force and displacement, **Hooke’s Law**

Simplest case is the Hookean spring law

\[ F = kx, \]

- \( k \): spring constant (stiffness)
- \( F \): applied load
- \( x \): spring extension
Example: Structural Analysis

This relationship can be generalized to structural systems in 2D and 3D, which yields a linear system of the form

\[ Kx = F \]

- \( K \in \mathbb{R}^{n \times n} \): “stiffness matrix”
- \( F \in \mathbb{R}^{n} \): “load vector”
- \( x \in \mathbb{R}^{n} \): “displacement vector”
Example: Structural Analysis

Solving the linear system yields the displacement \((x)\), hence we can simulate structural deflection under applied loads \((F)\)

\[
Kx = F
\]

Unloaded structure

Loaded structure
Example: Structural Analysis

It is common engineering practice to use Hooke’s Law to simulate complex structures, which leads to large linear systems.

(From SAP2000, structural analysis software)
Example: Economics

Leontief awarded Nobel Prize in Economics in 1973 for developing linear input/output model for production/consumption of goods

Consider an “economy” in which $n$ goods are produced and consumed

- $A \in \mathbb{R}^{n \times n}$: $a_{ij}$ represents amount of good $j$ required to produce 1 unit of good $i$
- $x \in \mathbb{R}^{n}$: $x_i$ is number of units of good $i$ produced
- $d \in \mathbb{R}^{n}$: $d_i$ is consumer demand for good $i$

In general $a_{ii} = 0$, and $A$ may or may not be sparse
Example: Economics

The total amount of $x_i$ produced is given by the sum of consumer demand ($d_i$) and the amount of $x_i$ required to produce each $x_j$

$$x_i = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + d_i$$

production of other goods

Hence $x = Ax + d$ or,

$$(I - A)x = d$$

Solve for $x$ to determine the required amount of production of each good

If we consider many goods (e.g. an “entire economy”), then we get a large linear system
Matrix computations arise all over the place!

Numerical Linear Algebra algorithms provide us with a toolbox for performing these computations in an efficient and stable manner.

In most cases, can use these tools as “black boxes”, e.g. use “backslash” in Matlab to solve $Ax = b$ for you.

But it’s important to understand what the linear algebra “black boxes” do:

- Pick the right algorithm for a given situation (e.g. exploit structure in a problem: symmetry, bandedness, etc)
- Understand how/when the “black box” can fail