TDT4127 Programming and Numerics
Week 40

Gaussian elimination

Important note

- Next week: **Reference group meeting**

  - Information on **Blackboard** about who is in the reference group
  - Contact them and give them feedback
    - They will in turn inform Guttorm and me in the meeting

- We appreciate both positive and negative feedback!

Learning goals

- Goals
  - Solving linear systems
  - Algorithm:
    - ***Gaussian elimination***
- Curriculum
  - Exercise 6 & 7

Remembering vectors

- A **vector** is an n-dimensional variable

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

- It is a nice and compact way of writing n variables
- It makes it easier for us to consider functions of more than one variable

$$f(\boldsymbol{x}) \text{ vs } f(x_1, x_2, \dots, x_n)$$

Lists and vectors

- Next week in programming: **Lists**
  - Lists are Python's way of representing vectors

    `x = [1,2,3]` is equivalent to $\boldsymbol{x} = [1,2,3]^T$
  - To access elements in a list, use the *delimiter* `[]`

    `print(x[0]) outputs 1`

    `print(x[2]) outputs 3`
  - *Elements* are numbered from 0: called zero-based numbering
    - Therefore, we write $x_0, x_1, \ldots, x_{n-1}$ for mathematics in this lecture.
  - Lists, unlike mathematical vectors, can contain other stuff too
    - Text strings, numbers, bools, etc.
    - *More about this next week*

Lists and vectors

- Lists allow us to handle large amounts of data easily
  - If we have three values, it is cleaner to write

    ```
    x = [1,2,3]
    ```

    instead of

    ```
    x0 = 1
    x1 = 2
    x2 = 3
    ```

  - Even more noticeable if we have a thousand variables
  - When making functions, we only need to pass one variable

    ```
    def f(x):
    ```

    instead of

    ```
    def f(x1,x2,x3):
    ```

Remembering matrices

- *Linear transformations* are special ***vector-valued*** functions

$$\boldsymbol{y} = f(\boldsymbol{x})$$

- The *linear* structure of f means it is of the form

$$\begin{aligned} y_0 &= a_{00}x_0 + a_{01}x_1 + a_{02}\mathrm{x}_2 \\ y_1 &= a_{10}x_0 + a_{11}x_1 + a_{12}x_2 \\ y_2 &= a_{20}x_0 + a_{21}x_1 + a_{22}x_2 \end{aligned}$$

  - **<u>Note:</u>** All the information about what f does lies in the coefficients a_{ij}
  - This means we can represent f using a matrix

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{21} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}, \qquad \boldsymbol{y} = A\boldsymbol{x}$$

Solving linear systems

- The linear equation $f(\boldsymbol{x}) = \boldsymbol{b}$ can be solved for $\boldsymbol{x}$ exactly!
  - Unlike the equations we used Newton on, where we got better and better results but never the mathematically ***exact*** solution.
- **Example**:

$$4x_0 + x_1 = 5$$
$$x_0 + x_1 = 2$$

Subtract the second equation from the first to find

$$3x_0 = 3$$

Can easily solve and find

$$x_0 = 1, x_1 = 1$$

- The general principle: add/subtract/swap equations to isolate one unknown, then *back substitute*.

Gaussian elimination

- Use row operations (swap rows/ add multiples of rows), go from

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & | & b_0 \\ a_{10} & a_{11} & a_{12} & | & b_1 \\ a_{20} & a_{21} & a_{22} & | & b_2 \end{bmatrix}$$

to

$$\begin{bmatrix} \tilde{a}_{00} & \tilde{a}_{01} & \tilde{a}_{02} & | & \tilde{b}_0 \\ 0 & \tilde{a}_{11} & \tilde{a}_{12} & | & \tilde{b}_1 \\ 0 & 0 & \tilde{a}_{22} & | & \tilde{b}_2 \end{bmatrix}$$

- A system where we can *back substitute* easily!

Gaussian elimination

- Start on first row, first column. The pivot element belongs to both the pivot row and the pivot column.

$$\begin{bmatrix} 0 & 1 & 2 & | & 3 \\ 4 & 5 & 6 & | & 7 \\ 8 & 9 & 1 & | & 2 \end{bmatrix}$$

1) If the current pivot element is 0, swap the pivot row for one **below** with a **nonzero** element

Gaussian elimination

- Start on first row, first column. The pivot element belongs to both the pivot row and the pivot column.

$$\begin{bmatrix} 4 & 5 & 6 & | & 7 \\ 0 & 1 & 2 & | & 3 \\ 8 & 9 & 1 & | & 2 \end{bmatrix}$$

1) If the current pivot element is 0, swap the pivot row for one **below** with a **nonzero** element

NTNU

Gaussian elimination

$$\left[\begin{array}{ccc|c} 0 & 1 & 2 & 3 \\ 0 & 5 & 6 & 7 \\ 0 & 9 & 1 & 2 \end{array}\right]$$

1a) If all elements **below** the pivot element are 0, shift the pivot column to the right

Gaussian elimination

$$\left[\begin{array}{ccc|c} 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 \\ 0 & -1 & -11 & -12 \end{array}\right]$$

2) Add multiples of the pivot row to the rows **below** such that they are **zeroed** out in the pivot column

Gaussian elimination

$$\left[\begin{array}{ccc|c} 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 \\ 0 & -1 & -11 & -12 \end{array}\right]$$

3) Move the pivot row down and the pivot column to the right. If on the **last row** or the augmented column, **stop**.

Gaussian elimination

$$\left[\begin{array}{ccc|c} 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & -9 & -9 \end{array}\right]$$

2) Add multiples of the pivot row to the rows **below** such that they are **zeroed** out in the pivot column

Gaussian elimination

$$\left[\begin{array}{ccc|c} 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & -9 & -9 \end{array}\right]$$

3) Move the pivot row down and the pivot column to the right. If on the **last row** or the augmented column, **stop**.

Back substitution

With the augmented matrix in triangular/echelon form:

$$\begin{bmatrix} 4 & 5 & 6 & | & 7 \\ 0 & 1 & 2 & | & 3 \\ 0 & 0 & -9 & | & -9 \end{bmatrix},$$

we can interpret this as a linear system

$$\begin{aligned} 4x_0 + 5x_1 + 6x_2 &= 7 \\ 0x_0 + 1x_1 + 2x_2 &= 3 \\ 0x_0 + 0x_1 - 9x_2 &= -9 \end{aligned}$$

and backsubstitute

$$x_2 = 1$$

$$x_1 = 3 - 2x_2 = 1$$

$$x_0 = \frac{7 - 5x_2 - 6x_1}{4} = -1$$

A few remarks

- A detailed explanation of Gaussian elimination is found in **exercise set 6**.
- If the pivot element is 0, which row do we swap with?
  - Swap with the row with the largest entry in the pivot column
  - Used in **exercise set 6**, explained next week
- In fact, swap rows even if the pivot element is *nonzero*!
  - This is to avoid numerical instability (next week).
- What about over/underdetermined systems?
  - We could *throw an exception*, programming week 43
  - For now, don't bother with this; most important to get used to programming with matrices!

Programming with matrices

- A matrix in Python is a *list of lists* (vector of vectors)

```
A = [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 1, 0]]
```

  gives the matrix from the examples above
- `A[i][j]` in Python corresponds to a_{ij} in mathematics
  - Writing `A[i][j] = 2` assigns the value $a_{ij} = 2$
- `A[i]` gives you the list that makes up the `i`'th row of `A`

```
A[1] = [4, 5, 6, 7]
```

- No command for getting the columns of `A`

Programming with matrices

- Some useful tips for **exercise set 6**:
  - `len(A)` gives you the number of rows in `A`
  - `len(A[1])` gives you the number of columns in `A`

- To loop over the *rows* in a matrix (fixed column `j`):

```
j = 1
for i in range(0,len(A))
   A[i][j] = …
```

Programming with lists

- Functions with lists as input can make changes ***in-place***
  - Change the values of input variables *outside* the function

```
def multList(x,y):
    for i in range(0,len(x)):
        x[i] = x[i]*y[i]
    return

x = [1, 2, 3]
y = [4, 5, 6]
multList(x,y)
print(x)
```

```
Prints: [4, 10, 18]
```

Summary

- Gaussian elimination can be used to solve linear systems
  - Together with back substitution
- Requires programming with lists and matrices
  - Treated as one variable, can look up/change values
  - Some programming tricks involving in-place functions and `len`
- Next week:
  - Partial pivoting (changing the pivot row for the largest) – why?
  - Plotting in Python

Questions?