Introduction to Regular Expressions[1]

- A regular expression, also called a pattern, is an expression used to specify a set of strings required for a particular purpose.
  - Check this: `https://regexone.com`.

[1] See `https://en.wikipedia.org/wiki/Regular_expression`; also `https://www.mathworks.com/help/matlab/matlab_prog/regular-expressions.html`.

Example

```
>> text = 'bat cat can car coat court CUT ct ...
    CAT-scan';
>> pattern = 'c[aeiou]+t';
>> start_idx = regexp(text, pattern)

start_idx =

       5    17
```

- The pattern 'c[aeiou]+t' indicates a set of strings:
  - c must be the first character;
  - c must be followed by one of the characters in the brackets [aeiou], followed by t as the last character;
  - in particular, [aeiou] must occur one or more times, as indicated by the + operator.

Metacharacters[2]

<table>
<tr><th>Operator</th><th>Definition</th></tr>
<tr><td>|</td><td>Boolean OR.</td></tr>
<tr><td>*</td><td>0 or more times consecutively.</td></tr>
<tr><td>?</td><td>0 times or 1 time.</td></tr>
<tr><td>+</td><td>1 or more times consecutively.</td></tr>
<tr><td>{n}</td><td>exactly n times consecutively.</td></tr>
<tr><td>{m, }</td><td>at least m times consecutively.</td></tr>
<tr><td>{, n}</td><td>at most n times consecutively.</td></tr>
<tr><td>{m, n}</td><td>at least m times, but no more than n times consecutively.</td></tr>
</table>

[2] See `https://www.mathworks.com/help/matlab/ref/regexp.html`.

Operator	Definition
.	any single character, including white space.
$[c_1 c_2 c_3]$	any character contained within the brackets.
$[\wedge c_1 c_2 c_3]$	any character not contained within the brackets.
$[c_1\text{-}c_2]$	any character in the range of c_1 through c_2.
$\backslash$s	any white-space character.
$\backslash$w	a word; any alphabetic, numeric, or underscore character.
$\backslash$W	not a word.
$\backslash$d	any numeric digit; equivalent to [0-9].
$\backslash$D	no numeric digit; equivalent to $[\wedge 0\text{-}9]$.

Output Keywords

Keyword	Output
'start'	starting indices of all matches, by default
'end'	ending indices of all matches
'match'	text of each substring that matches the pattern
'tokens'	text of each captured token
'split'	text of nonmatching substrings
'names'	name and text of each named token

Examples

```
clear; clc;

text1 = {'Madrid, Spain', 'Romeo and Juliet', ...
    'MATLAB is great'};
tokens = regexp(text1, '\s', 'split')

text2 = 'EXTRA! The regexp function helps you ...
    relax.';
matches = regexp(text2, '\w*x\w*', 'match')
```

Exercise: Listing Filtered Files

```
clear; clc;

file_list = dir;
filenames = {file_list(:).name};
A = regexp(filenames, '.+\.m', 'match');
mask = cellfun(@(x) ~isempty(x), A);
cellfun(@(f) fprintf('%s\\%s\n', pwd, f{:}), A(mask))
```

Example: By Names

- You can associate names with tokens so that they are more easily identifiable.
- For example,

```
>> str = 'Here is a date: 01-Apr-2020';
>> expr = '(?<day>\d+)-(?<month>\w+)-(?<year>\d+)';
>> mydate = regexp(str, expr, 'names')

mydate =

        day: '01'
      month: 'Apr'
       year: '2020'
```

Exercise: Web Crawler

- Write a script which collects the names of html tags by defining a token within a regular expression.
- For example,

```
>> str = '<title>My Title</title><p>Here is some ...
   text.</p>';
>> pattern = '<(\w+).*>.*</\1>';
>> [tokens, matches] = regexp(str, pattern, ...
   'tokens', 'match')
```

More Regexp Functions

- See **regexpi**, **regexprep**, and **regexptranslate**.

```
>> Lecture 6
>>
>>      -- Special Topic: File Operations & other I/O
>>
```

Spreadsheets: Excel/CSV Files (Revisited)

- The command **xlsread**(*filename*) reads excel files, for example,

```
[~, ~, raw] = xlsread("2330.xlsx");
```

- By default, it returns a numeric matrix.
- The text part is the 2nd output, separated from the numeric part.
- You may consider the whole spreadsheet by using the 3rd output (stored in a cell array).
- Note that you can use $\sim$ to drop the output value.

More Tips for Excel Files

- You can specify the range.
    - For example, the string argument `"B:B"` is used to import column B.
    - If you need a single value, say the cell B1, just use `"B1:B1"`.[3]
- You could specify the worksheet by the sheet name[4] or the sheet number.
- You could refer to the document for more details.[5]

---

[3] Contribution by Mr. Tsung-Yu Hsieh (MAT24409) on August 27, 2014.
[4] The default sheet name is "工作表".
[5] See https://www.mathworks.com/help/matlab/ref/xlsread.html.

Mat Files[6]

- Recall that I/O is costly.
- To save time, you may consider **save** matrices to the disk; for example,

```
data1 = rand(1, 10);
data2 = ones(10);
save('trial.mat', 'data1', 'data2');
```

- You can use **load** to fetch the data from mat files.

```
load('trial.mat');
```

[6] See https://www.mathworks.com/help/matlab/ref/save.html.

Selected Read/Write Functions

- For text data, see https://www.mathworks.com/help/matlab/text-files.html.
  - Try **dlmread**, **dlmwrite**, **csvread**, **csvwrite**, **textread**/**textscan**.
- For images, see https://www.mathworks.com/help/matlab/images_images.html.
- For video and audio, see https://www.mathworks.com/help/matlab/audio-and-video.html.

Selected File Operations[7]

cd	Change current folder.
pwd	Identify current folder.
ls	List folder contents by chars.
dir	List folder contents by structures.
exist	Check existence of variable, script, function, folder, or class.
mkdir	Make new folder.
visdiff	Compare two files or folders.

[7] See `https://www.mathworks.com/help/matlab/file-operations.html`.

Example: Pooling Data from Multiple Files[8]

```
clear; clc;

cd('./stocks'); % enter the folder
files = dir; % get all files in the current folder
files = files(3 :  end); % drop the first two
names = {files(:).name}; % get all file names
filter = endsWith(names, '.xlsx'); % filter by .xlsx
names = names(filter);

pool = cell(length(names), 2);
for i = 1 : length(names)
    [~, ~, raw] = xlsread(names{i});
    pool(i, :) = {names{i}(1 : 4), raw};
end
save('data_pool', 'pool');
```

[8] Download stocks.zip.

```
>> Lecture 7
>>
>>              -- Matrix Computation
>>
```

Vectors

- Let $\mathbb{R}$ be the set of all real numbers.
- $\mathbb{R}^n$ denotes the vector space of all m-by-1 column vectors:

$$u = (u_i) = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix}. \tag{1}$$

  - You can simply use the colon (:) operator to reshape any array in a column major, say $u(:)$.
- Similarly, the row vector v is

$$v = (v_i) = \begin{bmatrix} v_1 \cdots v_n \end{bmatrix}. \tag{2}$$

- We consider column vectors unless stated.

Matrices

- $\mathbf{M}_{m\times n}(\mathbb{R})$ denotes the vector space of all m-by-n real matrices, for example,

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

- Complex vectors/matrices[9] follow similar definitions and operations introduced later, simply with some care.

[9] Matlab treats a complex number as a single value.

Transposition

```
>> A = [1 i];
>> A' % Hermitian operator; see any textbook for ...
    linear algebra

ans =

   1.0000 + 0.0000i
   0.0000 - 1.0000i

>> A.' % transposition of A

ans =

   1.0000 + 0.0000i
   0.0000 + 1.0000i
```

Arithmetic Operations

- Let a_{ij} and b_{ij} be the elements of the matrices A and $B \in \mathbf{M}^{m \times n}(\mathbb{R})$ for $1 \leq i \leq m$ and $1 \leq j \leq n$.
- Then $C = A \pm B$ can be calculated by $c_{ij} = a_{ij} \pm b_{ij}$. (Try.)

Inner Product[10]

- Let $u, v \in \mathbb{R}^m$.
- Then the inner product, denoted by $u \cdot v$, is calculated by

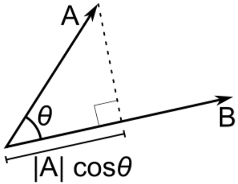
$$u \cdot v = u'v = [u_1 \cdots u_m] \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix}.$$

```
clear; clc;

u = [1; 2; 3];
v = [4; 5; 6];
u' * v     % normal way; orientation is important
dot(u, v) % using the built-in function
```

[10] Akaa dot product and scalar product.

- Inner product is also called projection for emphasizing its geometric significance.



- Recall that we know

$$u \cdot v = 0$$

if and only if these two are orthogonal to each other, denoted by

$$u \perp v.$$

Generalization of Inner Product

- Let $x \in \mathbb{R}$, $f(x)$ and $g(x)$ be real-valued functions.
- In particular, assume that $g(x)$ is a basis function.[11]
- Then we can define the inner product of f and g on $[a, b]$ by

$$\langle f, g \rangle = \int_a^b f(x)g(x)dx.$$

[11] See https://en.wikipedia.org/wiki/Basis_function, https://en.wikipedia.org/wiki/Eigenfunction, and https://en.wikipedia.org/wiki/Approximation_theory.

- For example, Fourier transform is widely used in engineering and science.
  - Fourier integral[12] is defined as

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

    where $f(t)$ is a square-integrable function.
  - The Fast Fourier transform (FFT) algorithm computes the discrete Fourier transform (DFT) in $O(n \log n)$ time.[13,14]

[12] See https://en.wikipedia.org/wiki/Fourier_transform.
[13] Cooley and Tukey (1965).
[14] See https://en.wikipedia.org/wiki/Fast_Fourier_transform.

Matrix Multiplication

- Let $A \in \mathbf{M}_{m \times q}(\mathbb{R})$ and $B \in \mathbf{M}_{q \times n}(\mathbb{R})$.
- Then $C = AB$ is given by

$$c_{ij} = \sum_{k=1}^{q} a_{ik} \times b_{kj}. \tag{3}$$

- For example,

$$\overset{4\times 2 \text{ matrix}}{\begin{bmatrix} a_{11} & a_{12} \\ \cdot & \cdot \\ a_{31} & a_{32} \\ \cdot & \cdot \end{bmatrix}} \overset{2\times 3 \text{ matrix}}{\begin{bmatrix} \cdot & b_{12} & b_{13} \\ \cdot & b_{22} & b_{23} \end{bmatrix}} = \overset{4\times 3 \text{ matrix}}{\begin{bmatrix} \cdot & x_{12} & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & x_{33} \\ \cdot & \cdot & \cdot \end{bmatrix}}$$

Example

```
clear; clc;

A = randi(10, 5, 4); % 5-by-4
B = randi(10, 4, 3); % 4-by-3
C = zeros(size(A, 1), size(B, 2));
for i = 1 :  size(A, 1)
    for j = 1 : size(B, 2)
        for k = 1 : size(A, 2)
            C(i, j) = C(i, j) + A(i, k) * B(k, j);
        end
    end
end
C % display C
```

- Time complexity: $O(n^3)$.
- Strassen (1969): $O(n^{log_2 7})$.

Matrix Exponentiation

- Raising a matrix to a power is equivalent to repeatedly multiplying the matrix by itself.
  - For example, $A^2 = AA$.
- The matrix exponential[15] is a matrix function on square matrices analogous to the ordinary exponential function, more explicitly,

$$e^A = \sum_{n=0}^{\infty} \frac{A^n}{n!}.$$

- However, it is not allowed to perform A^B.

[15] See matrix exponentials and Pauli matrices.

Determinants

- Consider the matrix
$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$
- Then $\det(A) = ad - bc$ is called the determinant of A.
  - The method of determinant calculation in high school is a wrong way but produces correct answers for all 3×3 matrices.
- Let's try the minor expansion formula for det(A).[16]

[16] See http://en.wikipedia.org/wiki/Determinant.

Recursive Algorithm for Minor Expansion Formula

```
function y = myDet(A)

    [r, ~] = size(A);

    if r == 1
        y = A;
    elseif r == 2
        y = A(1, 1) * A(2, 2) - A(1, 2) * A(2, 1);
    else
        y = 0;
        for i = 1 : r
            B = A(2 : r, [1 : i - 1, i + 1 : r]);
            cofactor = (-1) ^ (i + 1) * myDet(B);
            y = y + A(1, i) * cofactor;
        end
    end
end
```

- It needs $n!$ terms in the sum of products, so this algorithm runs in $O(n!)$ time!
- Use **det** for determinants, which can be done in $O(n^3)$ time by using LU decomposition or alike.[17]

[17] See https://en.wikipedia.org/wiki/LU_decomposition. Moreover, various decompositions are used to implement efficient matrix algorithms in numerical analysis. See https://en.wikipedia.org/wiki/Matrix_decomposition.

Linear Systems (Transformation/Mapping)[18]

- A linear system is a mathematical model of a system based on linear operators satisfying the property of superposition.
  - For simplicity, $Ax = y$ for any input x associated with the output y.
  - Then A is a linear operator if and only if

$$A(ax_1 + bx_2) = aAx_1 + bAx_2 = ay_1 + by_2$$

    for $a, b \in \mathbb{R}$.
  - For example, $\frac{d(x^2+3x)}{dx} = \frac{dx^2}{dx} + 3\frac{dx}{dx} = 2x + 3$.
- Linear systems typically exhibit features and properties that are much simpler than the nonlinear case.
  - What about nonlinear cases?

[18] See https://en.wikipedia.org/wiki/Linear_system.

First-Order Approximation: Local Linearization

- Let $f(x)$ be any nonlinear function.
- Assume that $f(x)$ is infinitely differentiable at x_0.
- By Taylor's expansion[19], we have

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + O\left((x - x_0)^2\right),$$

where $O\left((x - x_0)^2\right)$ is the collection of higher-order terms, which can be neglected as $x - x_0 \to 0$.

- Then we have a first-order approximation

$$f(x) \approx f'(x_0)x + k,$$

with $k = f(x_0) - x_0 f'(x_0)$, a constant.

[19] See https://en.wikipedia.org/wiki/Taylor_series.

Two Observations

- We barely feel like the curvature of the ground; however, we look at Earth on the moon and agree that Earth is a sphere.
- Newton's kinetic energy is a low-speed approximation (classical limit) to Einstein's total energy.
  - Let m be the rest mass and v be the velocity relative to the inertial coordinate.
  - The resulting total energy is

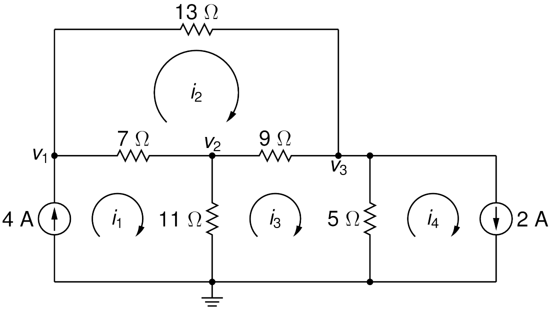
$$E = \frac{mc^2}{\sqrt{1 - (v/c)^2}}.$$

  - By applying the first-order approximation,

$$E \approx mc^2 + \frac{1}{2}mv^2.$$

Example: Kirchhoff's Laws[20]

- The algebraic sum of currents in a network of conductors meeting at a point is zero.
- The directed sum of the potential differences (voltages) around any closed loop is zero.



[20]See `https://en.wikipedia.org/wiki/Kirchhoff's_circuit_laws`.

General Form of Linear Equations[21]

- Let n be the number of unknowns and m be the number of constraints.
- A general system of m linear equations with n unknowns is

$$\begin{cases} a_{11}x_1 & +a_{12}x_2 & \cdots & +a_{1n}x_n & = & y_1 \\ a_{21}x_1 & +a_{22}x_2 & \cdots & +a_{2n}x_n & = & y_2 \\ \vdots & \vdots & \ddots & \vdots & = & \vdots \\ a_{m1}x_1 & +a_{m2}x_2 & \cdots & +a_{mn}x_n & = & y_m \end{cases}$$

where $x_1, \ldots, x_n$ are unknowns, $a_{11}, \ldots, a_{mn}$ are the coefficients of the system, and $y_1, \ldots, y_m$ are the constant terms.

[21] See https://en.wikipedia.org/wiki/System_of_linear_equations.

Matrix Equation

- Hence we can rewrite the aforesaid equations as follows:

$$Ax = y.$$

where

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

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \text{ and } y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}.$$

- Finally, x can be done by $x = A^{-1}y$, where A^{-1} is called the inverse of A.

Inverse Matrices[22]

- For simplicity, let $A \in \mathbf{M}_{n \times n}(\mathbb{R})$ and $x, y \in \mathbb{R}^n$.
- Then A is called invertible if there exists $B \in \mathbf{M}_{n \times n}(\mathbb{R})$ such that

$$AB = BA = I_n,$$

  where I_n denotes a $n \times n$ identity matrix.
  - We use A^{-1} to denote the inverse of A.
  - You can use **eye**(n) to generate an identity matrix I_n.
- Use **inv**(A) to calculate the inverse of A.

[22]See `https://en.wikipedia.org/wiki/Invertible_matrix#The_invertible_matrix_theorem`.

- However, **inv**(A) may return a weird result even if A is ill-conditioned, indicates how much the output value of the function can change for a small change in the input argument.[23]
- For example, calculate the inverse of the matrix

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

  - Recall the Cramer's rule[24]: A is invertible iff $\mathbf{det}(A) \neq 0$. (Try.)
- If these constraints cannot be eliminated by row reduction, they are linearly independent.

[23] You may refer to the condition number of a function with respect to an argument. Also try **rcond**.

[24] See https://en.wikipedia.org/wiki/Cramer's_rule.

Linear Independence

- Let $K = \{a_1, a_2, \ldots, a_n\}$ for each $a_i \in \mathbb{R}^m$.
- Now consider this linear superposition

$$x_1 a_1 + x_2 a_2 + \cdots + x_n a_n = 0,$$

  where $x_1, x_2, \ldots, x_n \in \mathbb{R}$ are the weights.
- Then K is linearly independent iff

$$x_1 = x_2 = \cdots = x_n = 0.$$

Example: $\mathbb{R}^3$

- Let
$$K_1 = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}.$$
- It is clear that K_1 is linearly independent.
- Moreover, you can represent all vectors in $\mathbb{R}^3$ if you collect all linear superpositions from K_1.
- We call this new set a span of K_1, denoted by $\text{Span}(K_1)$.[25]
- Clearly, $\text{Span}(K_1) = \mathbb{R}^3$.

[25] See https://en.wikipedia.org/wiki/Linear_span.

- Now let
$$K_2 = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \right\}.$$
- Then K_2 is not a linearly independent set. (Why?)
- If you take one or more vectors out of K_2, then K_2 becomes linearly independent.

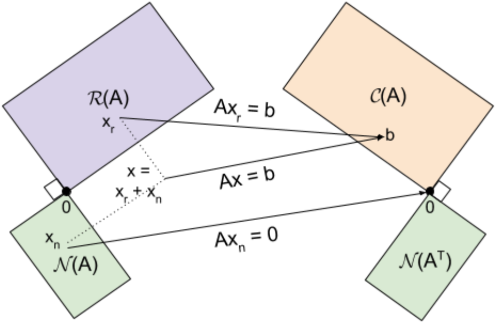
Basis of Vector Space & Its Dimension[26]

- However, you can take only one vector out of K_2 if you want to represent all vectors in $\mathbb{R}^3$. (Why?)
  - The dimension of $\mathbb{R}^3$ is exactly the size (element number) of K_2.
- We say that the basis of $\mathbb{R}^n$ is a maximally linearly independent set of size n.
- Note that the basis of $\mathbb{R}^3$ is not unique.
  - For example, K_1 could be also a basis of $\mathbb{R}^3$.

[26] See https://en.wikipedia.org/wiki/Basis_(linear_algebra), https://en.wikipedia.org/wiki/Vector_space, and https://en.wikipedia.org/wiki/Dimension_(vector_space).

Linear Transformation (Revisited)[27]

Matrix A converts n-tuples into m-tuples $\mathbb{R}^n \to \mathbb{R}^m$.
That is, linear transformation T_A is a map between rows and columns



Fundamental Subspaces

$\mathcal{C}$(A): Column space (image)
$\mathcal{R}$(A): Row space (coimage)
$\mathcal{N}$(A): Null space (kernel)
$\mathcal{N}(A^T)$: Left null space (cokernel)

Identities

$\dim(\mathcal{C}) \equiv \text{rank}(A)$
$\dim(\mathcal{N}) \equiv \text{nullity}(A)$

Theorems

$\dim(\mathcal{C}) + \dim(\mathcal{N}) = n$
$\dim(\mathcal{R}) = \dim(\mathcal{C})$

[27] See https://en.wikipedia.org/wiki/Linear_map; also see https://kevinbinz.com/2017/02/20/linear-algebra/.

Example: Vector Projection ($\mathbb{R}^3 \to \mathbb{R}^2$)

- Let $u \in \mathbb{R}^3$ and $v \in \mathbb{R}^2$.
- We consider the projection matrix (operator),

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

  so that $Au = v$.
- For example,

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

Solution Set to System of Linear Equations[28]

- Recall that m is the number of constraints and n is the number of unknowns.
- Now consider the following cases.
- If $m = n$, then there exists a unique solution.
- If $m > n$, then it is called an overdetermined system and there is no solution.
  - Fortunately, we can find a least-squares error solution such that $\| Ax - y \|^2$ is minimal, shown later.
- If $m < n$, then it is called a underdetermined system which has infinitely many solutions.
  - Become an optimization problem?
- For all cases,

$$x = A \setminus y.$$

[28] See https://www.mathworks.com/help/matlab/ref/mldivide.html.

Case 1: $m = n$

- For example,

$$\begin{cases} 3x + 2y - z = 1 \\ x - y + 2z = -1 \\ -2x + y - 2z = 0 \end{cases}$$

```
>> A = [3 2 -1; 1 -1 2; -2 1 -2];
>> b = [1; -1; 0];
>> x = A \ b

     1
    -2
    -2
```

Case 2: $m > n$

- For example,

$$\begin{cases} 2x - y = 2 \\ x - 2y = -2 \\ x + y = 1 \end{cases}$$

```
>> A = [2 -1; 1 -2; 1 1];
>> b = [2; -2; 1];
>> x = A \ b

      1
      1
```

Case 3: $m < n$

- For example,

$$\begin{cases} x + 2y + 3z = 7 \\ 4x + 5y + 6z = 8 \end{cases}$$

```
>> A = [1 2 3; 4 5 6];
>> b = [7; 8];
>> x = A \ b

     -3
     0
     3.333
```

- Note that this solution is a basic solution, one of infinitely many.
- How to find the directional vector? (Try **cross**.)

Gaussian Elimination Algorithm[29]

- First we consider the linear system is represented as an augmented matrix $[\,A\,|\,y\,]$.
- We then transform A into an upper triangular matrix

$$[\,\bar{A}\,|\,y\,] = \left[\begin{array}{cccc|c} 1 & \bar{a}_{12} & \cdots & \bar{a}_{1n} & \bar{y}_1 \\ 0 & 1 & \cdots & \bar{a}_{2n} & \bar{y}_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \bar{y}_n \end{array}\right].$$

  where $\bar{a}_{ij}$'s and $\bar{y}_i$'s are the resulting values after elementary row operations.
- This matrix is said to be in reduced row echelon form.

[29]See https://en.wikipedia.org/wiki/Gaussian_elimination.

- The solution can be done by backward substitution:

$$x_i = \bar{y}_i - \sum_{j=i+1}^{n} \bar{a}_{ij} x_j,$$

where $i = 1, 2, \cdots, n$.
- Time complexity: $O(n^3)$.

Exercise

```
clear; clc;

A = [3 2 -1; 1 -1 2; -2 1 -2];
b = [1; -1; 0];
A \ b % check the answer

for i = 1 : 3
    for j = i : 3
        b(j) = b(j) / A(j, i); % why first?
        A(j, :) = A(j, :) / A(j, i);
    end
    for j = i + 1 : 3
        A(j, :) = A(j, :) - A(i, :);
        b(j) = b(j) - b(i);
    end
end
x = zeros(3, 1);
```

```
for i = 3 : -1 : 1
    x(i) = b(i);
    for j = i + 1 : 1 : 3
        x(i) = x(i) - A(i, j) * x(j);
    end
end
x
```

Selected Functions of Linear Algebra[30]

- Matrix properties: **norm**, **null**, **orth**, **rank**, **rref**, **trace**, **subspace**.
- Matrix factorizations: **lu**, **chol**, **qr**.

[30] See `https://www.mathworks.com/help/matlab/linear-algebra.html`.

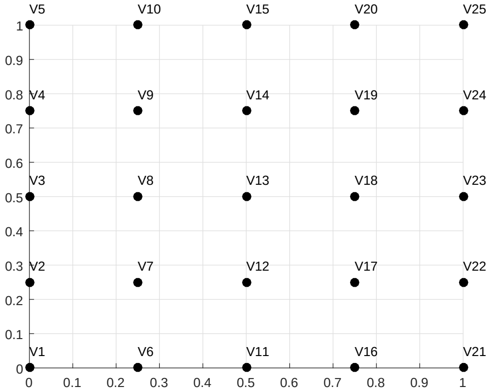
Numerical Example: 2D Laplace's Equation

- A partial differential equation (PDE) is a differential equation that contains unknown multivariable functions and their partial derivatives.[31]
- Let $\Phi(x, y)$ be a scalar field on $\mathbb{R}^2$.
- Consider Laplace's equation[32] as follows:

$$\nabla^2 \Phi(x, y) = 0,$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the Laplace operator.
- Consider the system shown in the next page.

[31] See
`https://en.wikipedia.org/wiki/Partial_differential_equation`.
[32] Pierre-Simon Laplace (1749–1827).



- Consider the boundary condition:
  - $V_1 = V_2 = \cdots = V_4 = 0.$
  - $V_{21} = V_{22} = \cdots = V_{24} = 0.$
  - $V_1 = V_6 = \cdots = V_{16} = 0.$
  - $V_5 = V_{10} = \cdots = V_{25} = 1.$

An Simple Approximation[33]

- As you can see, we partition the region into many subregions by applying a proper mesh generation.
- Then $\Phi(x, y)$ can be approximated by

$$\Phi(x, y) \approx \frac{\Phi(x + h, y) + \Phi(x - h, y) + \Phi(x, y + h) + \Phi(x, y - h)}{4},$$

where h is small enough.

[33] See https://en.wikipedia.org/wiki/Finite_difference_method#Example:_The_Laplace_operator.

Matrix Formation

- By collecting all constraints, we have $Ax = b$ where

$$A = \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}$$

and

$$b = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}^T.$$

Dimension Reduction by Symmetry

- As you can see, $V_7 = V_{17}$, $V_8 = V_{18}$ and $V_9 = V_{19}$.
- So we can reduce A to A'

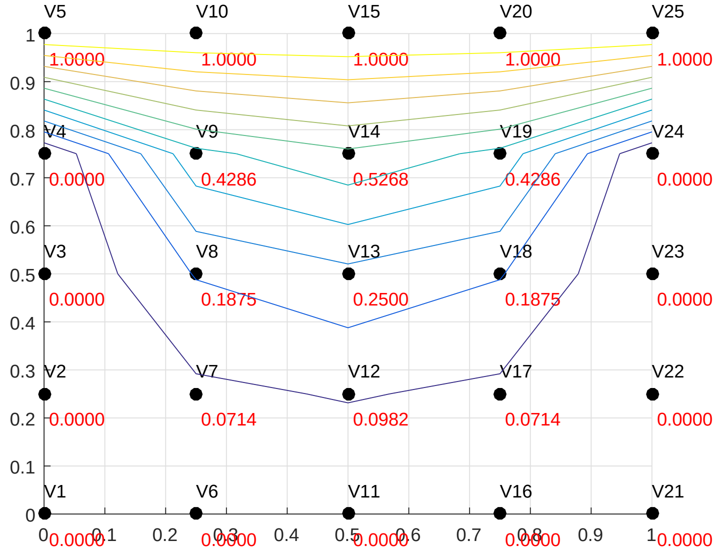
$$A' = \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 \\ -2 & 0 & 0 & 4 & -1 & 0 \\ 0 & -2 & 0 & -1 & 4 & -1 \\ 0 & 0 & -2 & 0 & -1 & 4 \end{bmatrix}$$

and

$$b' = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}^T.$$

- The dimensions of this problem are cut to 6 from 9.
- This trick helps to alleviate the curse of dimensionality.[34]

[34] See https://en.wikipedia.org/wiki/Curse_of_dimensionality.


V5
V10
V15
V20
V25
1.0000
1.0000
1.0000
1.0000
1.0000
V4
V9
V14
V19
V24
0.0000
0.4286
0.5268
0.4286
0.0000
V3
V8
V13
V18
V23
0.0000
0.1875
0.2500
0.1875
0.0000
V2
V7
V12
V17
V22
0.0000
0.0714
0.0982
0.0714
0.0000
V1
V6
V11
V16
V21
0.0000
0.0000
0.0000
0.0000
0.0000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Remarks

- This is a toy example for numerical methods of PDEs.
- We can use the PDE toolbox for this case. (Try.)
  - You may consider the finite element method (FEM).[35]
  - The mesh generation is also crucial for numerical methods.[36]
  - You can use the Computational Geometry toolbox for triangular mesh.[37]

[35] See https://en.wikipedia.org/wiki/Finite_element_method.

[36] See https://en.wikipedia.org/wiki/Mesh_generation.

[37] See https://www.mathworks.com/help/matlab/computational-geometry.html.