CHEMICAL ENGINEERING 2E04

Chapter 1 – Linear Algebraic Equations

Module 1A: Introduction to Linear Systems

Dr. Jake Nease

Kieran McKenzie

Steven Karolat

Cynthia Pham

Chemical Engineering

McMaster University

Updated June 9, 2019

Contents

Complementary Material

Textbook
-   Gilat, V. Subramaniam: *Numerical Methods for Engineers and Scientists*. Wiley. Third Edition (2014)
  - Chapter 2: 2.1, 2.2, 2.3, 2.4
  - Chapter 4: 4.1

Online Material
-   [I lecture online Linear Algebra Videos](#)

Learning Outcomes and Applications of Module

Goal of Numerical Methods for Solving Linear Systems of Equations (LSoEs)
Use the *structures* of equations to obtain results that would otherwise be a very long and tedious process to solve by hand. Some systems in Chemical Engineering can have thousands or even millions of equations, which are practically unsolvable without computers doing the calculations!

The Science of Numerical Methods
A set of mathematical techniques made to exploit *patterns* and *routines* to solve large problems systematically.

Learning Outcomes of Chapter 01
- Discuss and apply the terminology and properties of linear systems.
- Convert an engineering problem into a system of equations obeying a standard *structure*.
- Extend problem formulation to include large-scale systems, including *discretized* systems.
- Apply several *elimination* methods for solving LSoEs.
- Adapt and modify *elimination* methods for certain applications.
- Apply several *iterative* methods for solving LSoEs.
- Adapt, modify, and compare *iterative* methods to *elimination* methods for various engineering problems.

Applications of Numerical Methods/Linear Systems to Chemical Engineering

Whenever you have an LSoE, chances are that you will use some form of numerical method to solve it. Linear systems are apparent in almost every field of engineering, but also in other professions such as finance, logistics, aerospace and more. Chemical engineering examples include resource allocation, mixing problems, mass/energy balances, thermodynamic calculations, and discretized systems.

For example, examine the block diagram below of a process to make Butanol. Corn and water are mashed in the corm milling section, resulting in some by-products. Nutrients and bacteria are added and fermented, converting corn sugars to alcohols (and more by-products such as CO_2). Then, several separation sections (an emphasis of ChE 3M04 and 4M03) are used to purify the butanol. It is our responsibility as chemical engineers to know how much butanol is produced, what the by-products are, how much energy is required in each step, *etc*. Trying to compute the mass and energy flows of each stream by hand would be a PAIN. Instead, we can exploit the *structure* of the problem to make this routine easier.
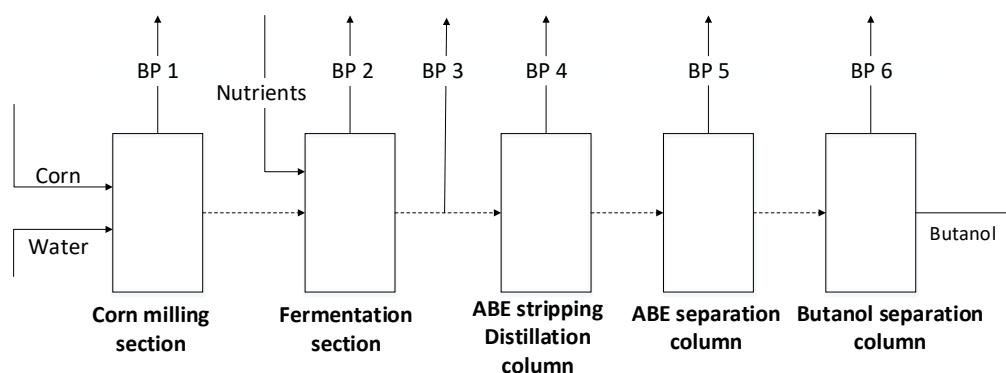
Figure 1: Butanol Production process, where BP is the by-products from the system

Types of Numerical Methods for Solving LSoEs

There are two main types of numerical methods to solve LSoEs: *direct* methods, and *iterative* methods.

Direct Methods	Iterative Methods
Calculate the solution using direct mathematical operations. This involves using and manipulating matrices and matrix coefficients into a form that makes the solution easier to see (i.e. into an upper triangular, lower triangular, or diagonal matrix).	An initial guess of the solution is made, and then used in a series of iterative steps to obtain a solution accurate within a user defined tolerance.
Advantages • Directly exploits the structure of a LSoE • Will always find a solution, barring numerical issues Disadvantages • Can be *computationally demanding* • Susceptible to *numerical precision* error • Susceptible to *structural* flaws	Advantages • May be more computationally *efficient* • No matrix manipulation required Disadvantages • Not *guaranteed* to converge to the solution • Susceptible to *numerical precision* error • Susceptible to *convergence* error
Methods covered in 2E: • Gauss Elimination • Gauss-Jordan Elimination • LU Decomposition	Methods covered in 2E: • Jacobi • Gauss-Seidel

Formulating Linear Systems of Equations

Defining Linear Equations

Let's warm up with a few definitions. First and foremost, this section deals with *Linear Equations* only.

Linear Equation

Any equation in which the dependent variable (y) is computed as a *linear combination* of independent variables (x). Linear terms are *constants* or variables *multiplied by constants*. The general structure is:

$$y = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n$$

Where $a_1 \dots a_n$ are constants and $x_1 \dots x_n$ are variables.

NB – A Non-Linear Equation is simply defined as any equation that is not linear! ☺

Some examples of one-dimensional linear and non-linear equations are shown below:

Linear	Linear	Non-Linear	Non-Linear	Non-Linear	Non-Linear	Non-Linear
$3 + x_1$	$3x_1$	$3 + x_1^2$	x_1 / x_2	$x_1^{x_2}$	$\sin(x_1)$	e^{x_1}

It is also important to note that we often write linear equations as a combination of independent variables equated to *a constant* (b) on the right-hand-side (RHS):

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = b$$

This is of importance when we write out *systems* of linear equations.

Defining Systems of Linear Equations

An LSoE is nothing more than a list of linear equations with constants on the RHS. For example, consider the following three equations that all depend on $x_1 \dots x_3$:

$$
\begin{array}{rrrcr}
4x_1 & +5x_2 & -3x_3 & = & 6 \\
10x_1 & -3.2x_2 & +2x_3 & = & 32 \\
& 8x_2 & +7x_3 & = & 56
\end{array}
$$

You will notice that each equation has *its own value for* a_i, the coefficient multiplied by variable x_i. Also note that there is only one term for each variable in each equation. This implies that all *like-terms* must be grouped.

There is thus a very convenient method to represent these equations using *matrices*. In this section, nearly all math will involve the use of matrices, so prepare yourself! Matrices format everything neatly, making math operations easier to visualize (especially when working with larger systems).

Workshop: Writing out Systems of Equations

Scenario: You are handed a scrap of paper from your colleague with the following equations scribbled down:

$$3x_1 - x_2 + 5x_4 - 2 = 0$$
$$3x_2 + 2x_3 - x_4 + 4x_4 - 0.5x_2 = 3$$
$$9(x_1 + x_2) - 6x_2 + 2 = 8$$

Goal: Write these equations out as a LSoEs as it is defined above.

Matrix Representation of Linear Equation Systems

All LSoEs can be written in the form $Ax = b$. For a system with m equations and n variables:
- Each equation gets its own row.
- Each variable gets its own column.
- The RHS cannot contain any variables. This may require rearranging the given equations.
- Coefficients of zero must still be shown in A and b as appropriate (cannot be left blank).

$$\underbrace{\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & a_{i,j} & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_m \end{bmatrix}}_{b}$$

Matrix elements are named as $a_{i,j}$ and b_i, where $i \triangleq$ row (equation) number and $j \triangleq$ column (variable) number.

Visual Example: For a system of $m = 2$ equations and $n = 3$ variables:

$$a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 = b_1$$
$$a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}x_3 = b_2$$

$$\Rightarrow \qquad \underbrace{\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}}_{b}$$

Converting the set of equations in the previous example into matrix form gives:

$$\begin{matrix} 4x_1 & +5x_2 & -3x_3 & = & 6 \\ 10x_1 & -3.2x_2 & +2x_3 & = & 32 \\ & 8x_2 & +7x_3 & = & 56 \end{matrix} \quad \Rightarrow \quad \begin{bmatrix} 4 & 5 & -3 \\ 10 & -3.2 & 2 \\ 0 & 8 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 32 \\ 56 \end{bmatrix}$$

Remember - even though there is no x_1 term in the third equation, it must still be represented in the matrix with a zero! This is because the coefficient on x_1 is zero.

Workshop: Problem Formulation

Scenario: You manage 3 plants (Houston, Nanticoke, and Sarnia) for Imperial Oil. Each plant uses a different process to refine crude oil. As a result, each plant has different yields for their three primary products (lubricating oil, diesel, and gasoline) per tonne of unrefined crude oil.

Chart values have units of barrels of output per tonne of input.

Plant	Lubricating Oil	Diesel	Gasoline
Houston	1	1	3
Nanticoke	1	3	1
Sarnia	2	2	2

Your employer wants you to schedule production (how much crude to refine) at each plant:

- You must meet the consumer demands of: 19 barrels of lubricating oil, 25 barrels of diesel, and 31 barrels of gasoline.
- There are no storage tanks, so you cannot over or under produce.

Goal: Formulate this scenario as an LSoE first, then present them in matrix form.

Linear Dependence and Independence

A *linear combination* occurs when two or more vectors are combined through vector addition/subtraction, and/or a vector is added to itself through vector multiplication. Each row in a matrix is called a *row vector* and can be thought of as a line in n-dimensional space, where n is the same dimension as the row vector.

Linear Dependence and Independence

A set of vectors, $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$, is said to be linearly independent if the expression:

$$k_1 v_1 + k_2 v_2 + \cdots + k_n v_n = 0$$

is satisfied *if and only if* $k_1 = k_2 = \cdots = k_n = 0$. In other words, no vector within the set can be achieved through a linear combination of the others (unless they are each multiplied by 0 first).

Otherwise, the set is linearly *dependent*, meaning that at least one of the vectors within the set can be constructed through a linear combination of the others (it literally depends on them).

When operating in a 2-dimensional space, you deal with 2-dimensional vectors (lines). Some comments about these vectors and how they behave geometrically:

- A set of two *distinct, non-parallel* line vectors, $\mathcal{V} = \{v_1, v_2\}$, are known to be *linearly independent*.

- If you were to extend these lines forever, they would intersect exactly once in $\mathbb{R}^2$ (2-space).

- Using these vectors, you will be able to achieve any vector in $\mathbb{R}^2$ through linear combinations of v_1 and v_2. A linear combination is $v_3 = k_1 v_1 + k_2 v_2$, where k_1 and k_2 are any scalars.

- Note that v_3 is by definition *linearly dependent* because it could instead be represented as its function of v_1 and v_2. Geometrically speaking, *it offers no new coordinate information*.

- The two vectors in $\mathcal{V}$ thus form a *basis* of $\mathbb{R}^2$, since together they can act as a 'base' or as 'building blocks' of any other real vector in that vector-space.

Critical Connection: Independence and Solutions

When we search for a solution to an LSoE, we desire the values of the variables $x_1 \dots x_n$ that satisfy *all equations simultaneously*. Geometrically, this is represented by the *intersection* of all vectors in $\mathbb{R}^n$.

In two dimensions, it is simply where the lines v_1 and v_2 intersect. This is also true for all dimensions, even those that cannot be visualized. If the lines intersect once, the solution is unique (only ONE value for each of $x_1 \dots x_n$ satisfy all equations at the same time).
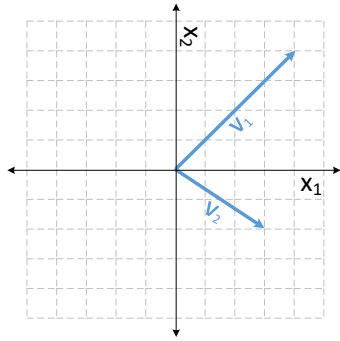

Figure 2: Given v_1 and v_2

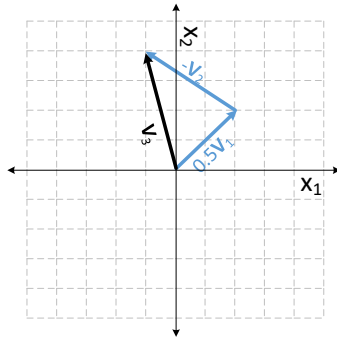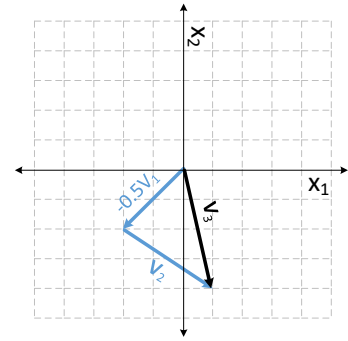Figure 3: $v_3 = 0.5v_1 - v_2$

Figure 4: $v_3 = -0.5v_1 + v_2$

As demonstrated above, any set with greater than 2 linearly independent vectors will be a linearly dependent set! The same is true with any n-dimensional space! This leads nicely into a discussion about Degrees of Freedom (DOF).

Reminder: Multiplication Rules for Matrices

1. If $AB = C$, the column number of **A** must equal the row number of **B** (inner dimensions must be the same). The resulting matrix will be the row number of **A** by the column number of **B**:

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$

2. When manipulating equations involving matrices, you must either multiply from the left or from the right:

Multiplying from the left:	Multiplying from the right:
$ABC = D$	$ABC = D$
$A^{-1}ABC = A^{-1}D$	$ABCC^{-1} = DC^{-1}$
$IBC = A^{-1}D$	$ABI = DC^{-1}$
$BC = A^{-1}D$	$AB = DC^{-1}$

Degrees of Freedom and Unique Solutions

Degrees of Freedom (DOF)

A system of equations should be considered as a set of constraints that must be satisfied simultaneously. The number of "free" variables that may be chosen while still satisfying the equation set are known as *Degrees of Freedom*.

Degrees of Freedom

The DOF of a system of equations (linear or otherwise) is the difference between the number of *independent* equations that must be satisfied (m) and the number of variables (n):

$$DOF = m - n$$

$DOF = 0$	The system is *square*. A *unique* solution exists that satisfies all equations.
$DOF > 0$	The system is *under-specified*. *Infinite* solutions exist because there are more variables to choose than equations to satisfy.
$DOF < 0$	The system is *over-specified*. No combination of variables can satisfy all equations simultaneously.

A system of linear equations with zero DOF is often referred to as a *square* system of linear equations.

Square Systems of Linear Equations

In general, a square LSoE in matrix form has the same number of rows as columns ($n = m$):

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

It is critical to recall that squareness and DOF analysis are only for systems of *independent* equations. When a system with $n = m$ rows and columns contains dependent rows, *removing the dependent row results in an underspecified system*. For example, any system where one or more rows can be reduced to zeros such as the one below will result in a system in which one or more variables are *unrestricted*.

$$\begin{bmatrix} 5 & -3 & 32 & -100 \\ 12 & 0 & 65 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 10 \\ 21 \\ 0 \\ 0 \end{bmatrix}$$

When you solve this system, you will find solutions for x_1 and x_2. However, they will *depend* on the selections of x_3 and x_4 because the 3rd and 4th rows tell you:

$$0x_1 + 0x_2 + 0x_3 + 0x_4 = 0$$

Geometric Interpretation of Unique Solutions

In two dimensions, the solutions to a system of equations is precisely where the vectors describing those equations intersect, as in the first panel of the figure below. If you have negative DOF (over-specified), you encounter the situation in the second panel (no solution can possibly exist). If you have positive DOF (under-specified), it is the equivalent of solving *less* equations in the same variable space (and thus many solutions exist, as in the final panel).
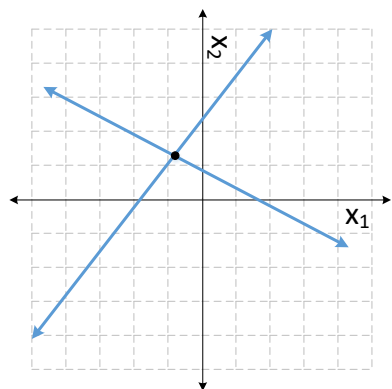
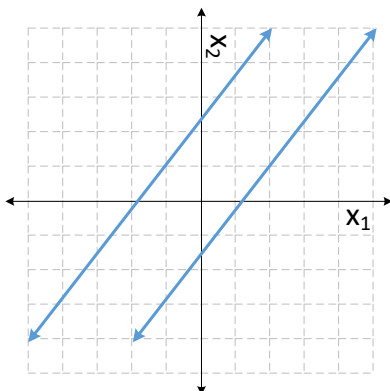*Figure 5: Non-Singular: One unique solution (One solution for x)*

Figure 6: Singular: no solutions (Something like $1 = 0$ in your matrix)
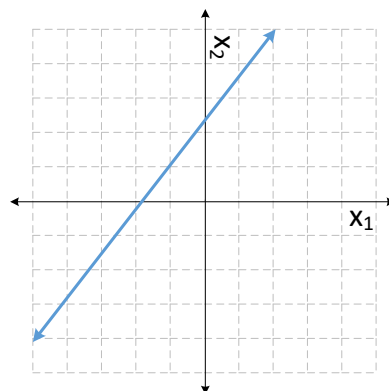
Figure 7: Singular: Both lines are the same (Infinite solutions due to free parameters)

In 3-D, it is the intersection of planes that give our solution. Trying to physically picture 4-D is possible but difficult (can you think of a method?). All vectors/systems beyond 4-D, although they are very common in engineering applications, are practically impossible to visualize.

Square Systems of Equations

The concepts of linear independence and square matrices (DOF = 0) come in handy when we try to solve our equations for the values of the variables. In fact, solutions can be directly related to *matrix invertibility* according to the following theorem.

Equivalent Statements: Square System of Equations

For the system $A\boldsymbol{x} = \boldsymbol{b}$, there is one unique solution $\boldsymbol{x}$ if:

- A is an $n \times n$ matrix of linearly independent equations
- The system has zero degrees of freedom
- A is square
- A is invertible
- The determinant of A, $\det(A) \neq 0$
- A has full rank

The bullets above are all equivalent – if one of them is true, they all must be true.

As a reminder, below you can see why A^{-1} must exist to find a unique solution $\boldsymbol{x}$:

$$A\boldsymbol{x} = \boldsymbol{b} \Rightarrow \underbrace{A^{-1}A}_{I}\boldsymbol{x} = A^{-1}\boldsymbol{b} \Rightarrow \boxed{\boldsymbol{x} = A^{-1}\boldsymbol{b}}$$

Reality Check: What is a Solution?

It is critical to recall that the *solution* to a system of linear equations is in fact the values of the variables (contained in the vector $\boldsymbol{x}$). We are technically not solving for the RHS or "outcome" of each equation, but rather the values that satisfy those equations.

Later, we will discuss the solutions to nonlinear equations, ordinary differential equations, and more. It is important to keep those definitions in context!

Formulating Large-Scale Systems of Linear Equations

One of the most useful applications of LSoEs is that typically complex nonlinear systems can be approximated as behaving linearly if we make some simplifying assumptions. For example, in *discretized chemical engineering systems* we may perform simple linear mass and energy balances for small control volumes (more of this in other courses) in order to obtain detailed information using simplified equations. A key learning outcome for this course is formulating large-scale systems of equations with zero DOF, so let's get some practice right now!

Example: Discretized Rectangular Cooling Fin

A rectangular cooling fin works by taking heat at the base of the fin and conducting it down the length of the fin towards the tip. As we move toward the tip, some heat is conducted throughout the material as Q, or is leaving from the surface towards the air via convection as R (shown in Figure A).
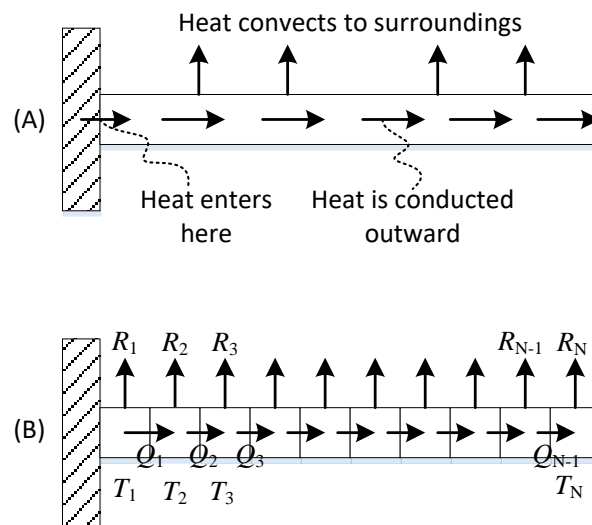
This system can be *discretized* by considering a fin of length L and dividing it into N individual sections (shown in Figure B) where T_i is the temperature in $°C$ of section i (where $i = 1, 2, \dots N$). The temperature at the base of the fin (T_1) is known and *constant* at $T_1 = T_{base}$. In ChE 2F04 and 3A04, you will learn methods to perform energy balances on each section of the fin. In this case, we can calculate the conduction Q_i and convection R_i for each section as:

Q_i is heat conducted from section i to $i + 1$ in Watts:	R_i is energy leaving the surface of section i by convection in Watts:
$$Q_i = kA_c(T_i - T_{i+1})$$	$$R_i = UA_s(T_i - T_\infty)$$
• k is a known conduction coefficient • A_c is the known cross-sectional area of the fin	• U is a known heat transfer coefficient • A_s is the known surface area of the section

The steady-state energy balance for an arbitrary discretized section i is (IN – OUT = 0):

$$R_i + Q_i - Q_{i-1} = 0$$

What we desire is to know the temperature along the fin. In other words, we want to know the temperature of each section (T_i). What better way to do this than using a workshop!

Advanced Workshop: Discretized Linear Systems (Midterm 1, 2018)

Goal: Form equations for ALL steady-state energy balances into the form $Ax = b$ to determine the temperature of each discretized section of the cooling fin. You may assume $T_1 = T_{base}$ and that there is no conduction in section N ($Q_N = 0$).

You can start by filling in this table:

Section 1	
Section i	
Section N	

And finish by writing the system $Ax = b$. You might have noticed that you were not given numerical values for N, k etc. A very valuable lesson that can be learned here is that defining your equations algebraically before using values is excellent for generalizing solution approaches.

Pattern Recognition! MIND BLOWN!

We've just dipped our toes into the HUGE field of *pattern recognition*. We noticed that each inner fin segment had the same energy balance equation structure! In other words, it was a pattern, and we could make an algorithm to solve a fin of any dimensions. We could only do so by recognizing the pattern!

- Patterns are natural and exist EVERYWHERE in the world and nature!
    - As a result, pattern recognition is literally bustling in the computing world.
- Facial/speech/colour/object recognition, DNA testing, barcode scanners; the list goes on.

Patterns are recognized either visually (by people), or by mathematical algorithms (how the computer recognizes things). And guess what? Computers use vectors and matrices everywhere in pattern recognition! To recognize patterns, a computer must first be 'trained' using training data – this is the machine learning stage. Afterwards, we use testing data to test the accuracy of the recognition.

We'll be getting cozy with `MATLAB` this term – we'll be doing some pattern recognition ourselves soon!

Conclusion and Summary

In this module we have:

- Defined linear equations and systems of linear equations
- Formulate small- and large-scale LSoEs
- Defined DOF and its geometric relationship with solution uniqueness
- Found that LSoEs can be solved by using A^{-1} of a linearly independent square matrix, and that the existence of A^{-1} is necessary for a solution to exist
- Algorithms exist that avoid the requirement for A^{-1} to solve LSoEs

Figure 8: Little dude having computer explodes after calculation A^{-1} using adjoint matrices!![1]

---

[1] http://masterwoodychan.com/interior-feng-shui/