Queens College, CUNY, Department of Computer Science
Numerical Methods
CSCI 361 / 761
Fall 2017
Instructor: Dr. Sateesh Mane

March 18, 2018

9 Lecture 9

9.1 Applied linear algebra

- We shall study how to solve a set of coupled linear equations in several variables.
- This topic is also called (applied) **linear algebra.**
- Applied linear algebra is a vast topic, and there are many clever algorithms available.
- Many of the algorithms involve the use of matrices.
- Hence some of the algorithms focus more on clever techniques to process matrices.
- We shall study only a few relatively simple algorithms (but useful in practice).
- Much good quality literature can be found online.
- **An obvious question to ask:** ***why linear equations?***
- The answer is simple: nonlinear equations are *much* more difficult to solve.
- Let us begin with the simpler case of linear equations.

9.2 Introduction

- Consider the following set of two equations in two variables x_1 and x_2:

$$x_1 + x_2 = 1\,, \tag{9.2.1a}$$
$$2x_1 - 3x_2 = 3\,. \tag{9.2.1b}$$

- We seek to find the values of x_1 and x_2 which satisfy *both* eqs. (9.2.1a) and (9.2.1b) is *simultaneously.* How shall we accomplish this?

- A widely employed technique is to perform **Gaussian elimination**.

  1. In fact, the technique is so obvious it is surprising it even has a modern name ("Gauss") as opposed to something from ancient times.
  2. Let us eliminate x_1 from the set of equations. Take two times eq. (9.2.1a) and subtract eq. (9.2.1b) to obtain

$$\begin{aligned} 2(x_1 + x_2) - (2x_1 - 3x_2) &= 2 - 3\,, \\ 5x_2 &= -1\,. \end{aligned} \tag{9.2.2}$$

  3. Now eq. (9.2.2) contains only one unknown variable, and the solution is $x_2 = -\frac{1}{5} = -0.2$.
  4. We now **backsubstitute** the solution for x_2 into either of eqs. (9.2.1a) or (9.2.1b) to obtain

$$x_1 - 0.2 = 1\,, \tag{9.2.3a}$$
$$2x_1 + 0.6 = 3\,. \tag{9.2.3b}$$

  5. Both eq. (9.2.3a) and eq. (9.2.3b) yield the same solution for x_1, which is $x_1 = 1.2$.
  6. The simultaneous solution of eqs. (9.2.1a) and (9.2.1b) is

$$(x_1, x_2) = (1.2, -0.2)\,. \tag{9.2.4}$$

- Let us formalize the steps in the procedure of Gaussian elimination.

- Equally importantly, let us examine the weak points of what can go wrong, **and why.**

- Some of the weak points can be cured, some not, hence the "why" is important.

9.3 Gaussian elimination

- Suppose we have n unknown variables $x_1, \dots, x_n$ and n linear equations

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \,, \tag{9.3.1a}$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \,, \tag{9.3.1b}$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n = b_3 \,, \tag{9.3.1c}$$
$$\vdots \qquad\qquad = \vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n \,. \tag{9.3.1d}$$

- The coefficients a_{ij} $(1 \le i, j \le n)$ and the numbers b_i on the right hand side $(1 \le i \le n)$ are constants, i.e. they do not depend on the variables $x_1, \dots, x_n$.

- We eliminate the variable x_1 from the second and third, etc. equations by subtracting a suitable multiple of the first equation eq. (9.3.1a). For example, using eq. (9.3.1b) and subtracting a_{21}/a_{11} times eq. (9.3.1a) yields

$$\begin{aligned} &a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n \\ &-\frac{a_{21}}{a_{11}}\big(a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n\big) = b_2 - \frac{a_{21}}{a_{11}}\,b_1 \,. \end{aligned} \tag{9.3.2}$$

- **It should be obvious that we do not need to eliminate x_1 first.** We could eliminate any one of the variables $x_2, \dots, x_n$ instead. This is an important fact and will be discussed in Sec. 9.7, after we see the problems which arise with the algorithm described in this section.

- Hence we obtain the following equation in $n-1$ unknowns $x_2, \dots, x_n$:

$$\Big(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\Big)x_2 + \cdots + \Big(a_{2n} - \frac{a_{21}}{a_{11}}a_{1n}\Big)x_n = b_2 - \frac{a_{21}}{a_{11}}\,b_1 \,. \tag{9.3.3}$$

- This looks messy, so we clean it up by defining some new coefficients

$$a'_{2,j} = a_{2j} - \frac{a_{21}}{a_{11}}a_{1j} \qquad\qquad (j = 2, \dots, n) \,. \tag{9.3.4}$$

  We also define a new number for the right hand side

$$b'_2 = b_2 - \frac{a_{21}}{a_{11}}\,b_1 \,. \tag{9.3.5}$$

- With an obvious notation, this yields the following set of $n-1$ linear equations in $n-1$ unknowns $x_2, \ldots, x_n$:

$$a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n = b'_2\,, \tag{9.3.6a}$$
$$a'_{32}x_2 + a'_{33}x_3 + \cdots + a'_{3n}x_n = b'_3\,, \tag{9.3.6b}$$
$$\vdots \qquad\qquad\qquad\qquad = \vdots$$
$$a'_{n2}x_2 + a'_{n3}x_3 + \cdots + a'_{nn}x_n = b'_n\,. \tag{9.3.6c}$$

- **Notice that the equations in eq. (9.3.6) have the same formal structure as those in eq. (9.3.1), but there are $n-1$ equations and $n-1$ unknowns.**

- Hence we eliminate x_2 from the equations in eq. (9.3.6) by subtracting a suitable multiple of eq. (9.3.6a) from eqs. (9.3.6b)–(9.3.6c).

- We define new coefficients and new right hand side variables

$$a''_{i,j} = a'_{ij} - \frac{a'_{i2}}{a'_{22}}a_{ij}\,, \qquad b''_i = b'_i - \frac{a'_{i2}}{a'_{22}}\,b'_2 \qquad (3 \le i, j \le n)\,. \tag{9.3.7}$$

- Obviously the structure of the $n-2$ equations in $n-2$ unknowns $x_3, \ldots, x_n$ is

$$a''_{33}x_3 + \cdots + a''_{3n}x_n = b''_3\,, \tag{9.3.8a}$$
$$\vdots \qquad\qquad\qquad = \vdots$$
$$a''_{n3}x_3 + \cdots + a''_{nn}x_n = b''_n\,. \tag{9.3.8b}$$

- The procedure should be obvious by now. We eliminate one unknown at a time. This is the process of **elimination**.

- Finally we end up with only one equation for the last variable x_n and we solve it for x_n.

- Then we **backsubstitute** the solution for x_n into one of the equations with two unknowns x_{n-1} and x_n. We solve that equation for x_{n-1}.

- The backsubstitution proceeds recursively, yielding the solutions for x_j, $j = n, n-1, \ldots, 1$ in reverse order.

- The procedure is simple to understand and to implement in a computer program.

- When it works, it does a good job.

- ***But what if it does not work?***

  1. How/why can it fail?
  2. Can we do anything to fix things if the procedure fails?
  3. Some causes of failure can be fixed, some not.
  4. These are all important things to know.
  5. As always, many clever people have studied the problem, and have formulated clever ideas to deal with these issues.

9.4 Sources of failure

9.4.1 Inconsistent equations

- Before we get lost in the details of "clever mathematics" and "programming" and the weaknesses of Gaussian elimination (as formulated above, anyway), let us first recognize that there may be no solution at all. In that case ***no*** algorithm will succeed, no matter how clever.
- Instead of the set of equations in eq. (9.2.1), let us consider the following set of equations, also in two variables x_1 and x_2:

$$x_1 + x_2 = 1\,, \tag{9.4.1.1a}$$
$$x_1 + x_2 = 2\,. \tag{9.4.1.1b}$$

- These two equations are **inconsistent**.
- There is no solution for (x_1, x_2) which will simultaneously satisfy eqs. (9.4.1.1a) and (9.4.1.1b).
- If the equations are inconsistent, no algorithm will work.
- What will happen in Gaussian elimination, when we subtract eq. (9.4.1.1a) from eq. (9.4.1.1b), is that *both* the variables x_1 and x_2 cancel out. We are left with an equation of the form

$$0 = (\text{nonzero value})\,. \tag{9.4.1.2}$$

- This is a contradiction and there is no solution.
- When we detect such a situation, using any algorithm, we stop and report that the equations are inconsistent.

9.4.2 Linear independence

- Next, instead of the set of equations in eq. (9.2.1), let us consider the following set of equations, also in two variables x_1 and x_2:

$$x_1 + x_2 = 1\,, \tag{9.4.2.1a}$$
$$2x_1 + 2x_2 = 2\,. \tag{9.4.2.1b}$$

- These equations are not inconsistent. They are obviously consistent, and in fact they are the same equation (in disguise).

- What will happen in Gaussian elimination, when we subtract 2 times eq. (9.4.2.1a) from eq. (9.4.2.1b), is that both the variables x_1 and x_2 cancel out, *and also the right hand side.* We obtain the equation

$$0 = 0\,. \tag{9.4.2.2}$$

- This is not a contradiction.

- The set of equations in eq. (9.4.2.1) are **not linearly independent**.

- In terms of matrices and rank, the rank is less than the number of equations, or the equations are not of full rank, which is simply a fancy way of saying the same thing.

  1. **The term 'rank' is a pure mathematical concept which is given in textbooks.**
  2. The rank tells us the maximum number of linearly independent equations we can obtain.
  3. It is a bit complicated to explain and we shall not go into details in this course.
  4. The rank of a matrix is the dimension of the vector space spanned by its columns.
  5. The rank is also equal to the dimension of the vector space spanned by its rows.
  6. Note that the matrix need not be square, but (column rank) = (row rank) anyway.
  7. The above concepts require pure mathematics which we shall not discuss in this course.

- In the case of eq. (9.4.2.1), there is only one independent equation in two variables:

$$x_1 + x_2 = 1\,. \tag{9.4.2.3}$$

- Hence there is an infinity of solutions, not one unique answer.

- A computer program can detect and report that the equations are not linearly independent.

- Can a set of equations be both inconsistent and not linearly independent? Of course:

$$x_1 + x_2 + x_3 = 1\,, \tag{9.4.2.4a}$$
$$2x_1 + 2x_2 + 2x_3 = 2\,, \tag{9.4.2.4b}$$
$$x_1 + x_2 + x_3 = 3\,. \tag{9.4.2.4c}$$

9.5 Undetermined and overdetermined sets of linear equations

- Let us digress briefly from "failure" to analyze the concept of linearly independent equations in more detail.
- It may be the case that we have more variables than (linear) equations to begin with. As opposed to discovering that the equations are not linearly independent, we *know* ahead of time that there are not enough equations to determine a unique solution.
- Such a set of equations is called **underdetermined.**
- An underdetermined set of equations is not hopeless. There are numerical techniques which can deal with them.
- Conversely, we may have more equations than unknowns.
- Such a set of equations is called **overdetermined.**
- If a set of equations is overdetermined, the situation is also not hopeless.
- A solution may exist, if the equations are consistent. Basically, if there are n unknowns, we must select a set of n linearly independent equations from the set, solve them, and check if the remaining equations are also satisfied.
- However, finding a simultaneous solution of all the equations is not usually the goal of a person with an overdetermined set of equations. Usually, with an overdetermined set of equations, we are searching for a "least squares fit" (or some other "best fit") to the equations.
- However, that is not the subject of this lecture. Finding a "best fit" for an overdetermined set of equations falls in the subject area of **fitting of data.**
- Fitting of data is a very important subject, but it is a different topic.

9.6 Ill-conditioned equations

- **Ill-conditioned equations** are also not a "failure" although the situation ***is*** a problem of numerical analysis.
- This is as opposed to saying that a solution does not exist or is not unique.
- What ill conditioning means is that the computed solution is not accurate, or not reliable. Let us understand what this means, and why.
- As usual, we illustrate the situation with an example. Consider the following set of two equations in two variables x_1 and x_2:

$$x_1 + 0.999x_2 = 1\,, \tag{9.6.1a}$$
$$1.001x_1 + x_2 = 1\,. \tag{9.6.1b}$$

- Using Gaussian elimination, we obtain the following equation for x_2:

$$\begin{aligned}(1 - 1.001 \times 0.999)x_2 &= 1 - 1.001 \times 1\\ 10^{-6}x_2 &= -0.001\\ x_2 &= -1000\,.\end{aligned} \tag{9.6.2}$$

  Backsubstitution into eq. (9.6.1a) yields

$$x_1 - 999 = 1\,. \tag{9.6.3}$$

  The solution is

$$x_1 = 1000\,. \tag{9.6.4}$$

- So far, so good. Now suppose we modify the coefficient of x_1 in eq. (9.6.1a) to 1.001. The new set of equations is

$$1.001x_1 + 0.999x_2 = 1\,, \tag{9.6.5a}$$
$$1.001x_1 + x_2 = 1\,. \tag{9.6.5b}$$

- Using Gaussian elimination, we obtain the following equation for x_2:

$$\begin{aligned}(1 - 0.999)x_2 &= 1 - 1\\ 0.001x_2 &= 0\\ x_2 &= 0\,.\end{aligned} \tag{9.6.6}$$

  Backsubstitution into either eq. (9.6.5a) or eq. (9.6.5b) yields

$$1.001x_1 = 1\,. \tag{9.6.7}$$

  The solution is

$$x_1 = \frac{1}{1.001} \simeq 0.999000999000\ldots\,. \tag{9.6.8}$$

- The coefficient of x_1 changed from 1 in eq. (9.6.1a) to 1.001 in eq. (9.6.5a), which is only a small change (one part in a thousand). All the other coefficients in eqs. (9.6.1) and (9.6.5) were unchanged.
- Nevertheless, the solution changed from $(x_1, x_2) = (1000, -1000)$ to $(x_1, x_2) = (1/1.001, 0) \simeq (0.999, 0)$.
- An **ill-conditioning system** is one where a small change in the input parameters causes a large change in the solution.
- **Ill conditioning** applies to numerical algorithms in general, not just solutions of systems of linear equations.
- Ill conditioning means the solution is very sensitive to the values of the inputs. In practice, the values of the inputs in a numerical calculation may not be known with absolute precision. If the input values are themselves the outputs of a different numerical computation, they may only be known approximately and will contain roundoff error. Under such circumstances, the computed value of the solution may not be trustworthy.
- In the context of solving a set of linear equations, there are ways to determine if the equations are ill conditioned. The "solver function" can return a warning to the calling application that the equations are ill conditioned.
- A system that is not ill conditioned is **well conditioned.**
- For lack of time, we shall not discuss ill conditioning in these lectures. However, bear in mind that ill conditioning is a genuine problem.

9.7 Pivoting: basics

- Consider the following (perfectly reasonable) set of equations in two variables x_1 and x_2:

$$x_2 = 1\,, \tag{9.7.1a}$$
$$x_1 + 2x_2 = 3\,. \tag{9.7.1b}$$

  1. The coefficient of x_1 in eq. (9.7.1a) is **zero.**
  2. Nevertheless, the Gaussian elimination algorithm in Sec. 9.3 tells us to eliminate x_1 by multiplying eq. (9.7.1a) by $1/0$ and subtract it from eq. (9.7.1b).
  3. The procedure encounters a division by zero.

- Next consider the following admittedly ridiculous set of equations in three variables (x_1, x_2, x_2):

$$10^{-20}x_1 + 4x_2 + x_3 = 1\,, \tag{9.7.2a}$$
$$x_1 + 2x_2 = 3\,, \tag{9.7.2b}$$
$$2x_1 + x_3 = 4\,. \tag{9.7.2c}$$

  1. *Who in their right mind would multiply eq.* (9.7.2a) *by* 10^{20} *and subtract from eq.* (9.7.2b) *and eq.* (9.7.2c)*, to eliminate* x_1*?*
  2. However, this is exactly what the Gaussian elimination algorithm in Sec. 9.3 says to do.
  3. Let us work through the algorithm in Sec. 9.3 and see what happens.
  4. First we eliminate x_1 to obtain the equations

$$(2 - 4\times 10^{20})x_2 - 10^{20}x_3 = 3 - 10^{20}\,, \tag{9.7.3a}$$
$$-8\times 10^{20}x_2 + (1 - 2\times 10^{20})x_3 = 4 - 2\times 10^{20}\,. \tag{9.7.3b}$$

  5. Next we eliminate x_2 by adding $8\times 10^{20}/(2-4\times 10^{20})$ times eq. (9.7.3a) to eq. (9.7.3b). This yields an equation in x_3 only:

$$\left[1 - 2\times 10^{20} - \frac{8\times 10^{40}}{2 - 4\times 10^{20}}\right] x_3 = 4 - 2\times 10^{20} + \frac{8\times 10^{20}}{2 - 4\times 10^{20}}(3 - 10^{20})\,. \tag{9.7.4}$$

  6. The coefficients in eq. (9.7.4) are plagued with overflow problems and cancellations of large numbers.
  7. For example the value of the right hand side of eq. (9.7.4) is approximately

$$4 - 2\times 10^{20} + \frac{8\times 10^{20}}{2 - 4\times 10^{20}}(3 - 10^{20}) \simeq 4 - 2\times 10^{20} - 2(3 - 10^{20}) = -2\,. \tag{9.7.5}$$

  All the terms of $O(10^{20})$ cancel out and the final result is only -2. This is a classic example of large numbers cancelling to yield a small result.

8. Let us approximate the coefficients in eqs. (9.7.3a) and (9.7.3b) themselves. We obtain the approximate equations

$$-4 \times 10^{20} x_2 - 10^{20} x_3 \simeq -10^{20} \,, \tag{9.7.6a}$$

$$-8 \times 10^{20} x_2 - 2 \times 10^{20} x_3 \simeq -2 \times 10^{20} \,. \tag{9.7.6b}$$

9. Observe that eqs. (9.7.6a) and (9.7.6b) are not linearly independent. We cannot obtain a unique solution from them.

- The above problems count as failures, but *they are not failures in the original equations.*
- They are ***weak points of the algorithm in Sec. 9.3.***
- In the algorithm in Sec. 9.3, notice that we divide by a_{11}, then a'_{22}, then a''_{33}, etc.
- The parameters a_{11}, a'_{22}, a''_{33}, etc. are called **pivots**.
- The algorithm in Sec. 9.3 has the weakness that it may employ division by pivots with small or zero magnitudes. This will obviously lead to numerical inaccuracies in the computation, even though the orginal equations may be linearly independent, consistent and well conditioned.
- Since it is we who choose the pivots, we can make a better choice. We can avoid the problem of small or zero pivots.

9.8 Full and partial pivoting

- It should be obvious that we do not need to eliminate x_1 from the equations first. We can eliminate any one of the variables $x_2, \ldots, x_n$ instead. After that we can select any other variable, etc. Basically, the order of elimination can be any permutation of $(1, 2, \ldots, n)$.
- It is equally obvious that even if we decide to eliminate x_1 first, we do not need to subtract multiples of the first row from the rest. We could select any other row, say i_1, and use that to eliminate x_1 from the other rows. After that, we would select a row indexed by i_2, where clearly $i_2 \neq i_1$, and use the row i_2 to eliminate x_2. Proceeding in this way, the set $(i_1, i_2, \ldots, i_n)$ is also any permutation of $(1, 2, \ldots, n)$.
- **Full pivoting** is the algorithm where we eliminate any one of the variables first, then another one, then another, etc. till the end, in a permutation of $(1, 2, \ldots, n)$.
- **Partial pivoting** is the algorithm where we eliminate x_1 first, then x_2, then x_3, etc., where the row indexes $(i_1, i_2, \ldots, i_n)$ are a permutation of $(1, 2, \ldots, n)$.
- We shall employ partial pivoting in these lectures.

9.9 Partial pivoting Part 1

- We loop through the coefficients of x_1 in the equations, i.e. the values of a_{i1} in column 1, for $i = 1, \ldots, n$. We find the coefficient with the largest magnitude. (It does not matter if the value is negative.) Let the index of that largest pivot be r_1 ("r" for row). If there are two or more values of equal largest magnitude, pick either one.

- Hence
$$|a_{r_1 1}| \;\geq\; |a_{i1}| \qquad\qquad (i = 1, \ldots, n)\,. \tag{9.9.1}$$

- **This procedure can be fooled. We require an additional precaution.**

- Let us multiply the first equation by 10^{21} to obtain the following set of equations.
$$10x_1 + 4 \times 10^{21} x_2 + 10^{21} x_3 = 10^{21}\,, \tag{9.9.2a}$$
$$x_1 + 2x_2 = 3\,, \tag{9.9.2b}$$
$$2x_1 + x_3 = 4\,. \tag{9.9.2c}$$

  1. The above equations are equivalent to those in eq. (9.7.2).
  2. However, to eliminate x_1 using these equations, the pivot with the largest amplitude is $a_{11} = 10$, hence the pivot is $r_1 = 1$.
  3. We are back to the bad situation of eliminating x_1 by subtracting multiples of the first equation eq. (9.9.2a).

- We avoid this problem by scaling the amplitudes as follows.

  1. In each equation we find the coefficient with the largest amplitude
  $$\hat{a}_i = \max\{|a_{ij}|,\, j = 1, \ldots, n\}\,. \tag{9.9.3}$$
  2. Then we loop through the equations and compare the values of $|a_{i1}|/\hat{a}_i$.
  3. Note that by construction $|a_{i1}|/\hat{a}_i \leq 1$.
  4. We choose the pivot r_1 to be the one with the largest scaled amplitude
  $$\frac{|a_{r_1 1}|}{\hat{a}_{r_1}} \geq \frac{|a_{i1}|}{\hat{a}_i} \qquad\qquad (i = 1, \ldots, n)\,. \tag{9.9.4}$$
  5. We employ the equation in row r_1 to eliminate x_1 from the other equations. This yields a set of $n - 1$ equations in the unknowns $x_2, \ldots, x_n$.
  6. We repeat this procedure to eliminate x_2, etc. To choose r_2, we compute
  $$\hat{a}'_i = \max\{|a'_{ij}|,\, j = 1, \ldots, n\} \qquad\qquad (i \neq r_1)\,. \tag{9.9.5}$$
  Then we choose r_2 such that
  $$\frac{|a'_{r_2 2}|}{\hat{a}'_{r_2}} \geq \frac{|a'_{i2}|}{\hat{a}'_i} \qquad\qquad (i \neq r_1)\,. \tag{9.9.6}$$
  7. The procedure repeats in an obvious way for the remaining pivots.

- Using scaling, we avoid being fooled by false values for the pivot with the largest amplitude.

- The indices after swapping rows, say $(s_1, s_2, \ldots, s_n)$, are a permutation of $(1, 2, \ldots, n)$.

9.10 Partial pivoting Part 2

- There is a piece of messiness with the procedure in Sec. 9.9.
- Recall that the row indexes $(r_1, r_2, \ldots, r_n)$ are a permutation of $(1, 2, \ldots, n)$.
- In general, as we loop through the equations, after we eliminate more and more variables, the bookkeeping becomes messy, to keep track of the pivoted rows, and the sequence of the pivots.
- It would be nicer (and probably simpler to debug in a program), if the pivoted rows were always in the order $(1, 2, \ldots, n)$.
- Hence at each step in the pivoting process, ***we swap the rows,*** as will be explained below.
- For the equations in eq. (9.7.2), at the first step we set $r_1 = 3$ and swap rows 1 and 3, to obtain the set

$$2x_1 + x_3 = 4\,. \tag{9.10.1a}$$
$$x_1 + 2x_2 = 3\,, \tag{9.10.1b}$$
$$10^{-20}x_1 + 4x_2 + x_3 = 1\,, \tag{9.10.1c}$$

- Then at the next step, to eliminate x_2, we compare rows 2 and 3 (and swap rows if necessary).
- It is obvious how to implement the swapping procedure in general, for n unknowns.
- Clearly, we must keep track of the swaps of the rows.

9.11 Partial pivoting Part 3

- Let us apply partial pivoting to eq. (9.7.2), with partial pivoting and swapping.
- For ease of reference, the equations in eq. (9.7.2) are reproduced below

$$10^{-20}x_1 + 4x_2 + x_3 = 1\,, \tag{9.11.1a}$$
$$x_1 + 2x_2 = 3\,, \tag{9.11.1b}$$
$$2x_1 + x_3 = 4\,. \tag{9.11.1c}$$

- Our initial array of swap indices is $(1, 2, 3)$.
- For the first pivoting step, observe that $\hat{a}_1 = 4$, $\hat{a}_2 = 2$, $\hat{a}_3 = 2$ and

$$\frac{a_{11}}{\hat{a}_1} = \frac{10^{-20}}{4}\,, \qquad \frac{a_{21}}{\hat{a}_2} = \frac{1}{2}\,, \qquad \frac{a_{31}}{\hat{a}_2} = 1\,. \tag{9.11.2}$$

- Hence the first pivot is $r_1 = 3$. We swap rows 1 and 3 to obtain

$$2x_1 + x_3 = 4\,, \tag{9.11.3a}$$
$$x_1 + 2x_2 = 3\,, \tag{9.11.3b}$$
$$10^{-20}x_1 + 4x_2 + x_3 = 1\,. \tag{9.11.3c}$$

- Our array of swap indices is updated to $(3, 2, 1)$.
- We eliminate x_1 from eqs. (9.11.3b) and (9.11.3c) to obtain

$$2x_2 - 0.5x_3 = 1\,, \tag{9.11.4a}$$
$$4x_2 + (1 - 0.5 \times 10^{-20})x_3 = 1 - 2 \times 10^{-20}\,. \tag{9.11.4b}$$

- Next we compute $\hat{a}'_2 = 2$ and $\hat{a}'_3 = 4$, so

$$\frac{|a'_{22}|}{\hat{a}'_2} = 1\,, \qquad \frac{|a'_{32}|}{\hat{a}'_3} = 1\,. \tag{9.11.5}$$

- The values are equal so we can choose either row.
  1. If we select $r_2 = 2$, a swap is not required. Our array of swap indices remains at $(3, 2, 1)$.
  2. If we select $r_2 = 3$, a swap is required. Our array of swap indices is updated to $(3, 1, 2)$.
  3. **Whichever pivot row we choose, the array of swap indices always keeps track of the original equations.**
- For simplicity, we choose $r_2 = 2$. A swap is not required. The array of swap indices is $(3, 2, 1)$.
- Hence we eliminate x_2 from eqs. (9.11.4a) and (9.11.4b) to obtain an equation in x_3 alone:

$$\begin{aligned}(1 - 0.5 \times 10^{-20} + 1)x_3 &= 1 - 2 \times 10^{-20} - 2\\ (2 - 0.5 \times 10^{-20})x_3 &= -1 - 2 \times 10^{-20}\,.\end{aligned} \tag{9.11.6}$$

- Our final array of swap indices is $(3, 2, 1)$.
- We solve for x_3 to obtain
$$x_3 = -\frac{1 + 2 \times 10^{-20}}{2 - 0.5 \times 10^{-20}} \simeq -\frac{1}{2}\,. \tag{9.11.7}$$
- We backsubstitute into eq. (9.11.4a) to solve for x_2 (use approximate solution):
$$\begin{aligned} 2x_2 - 0.5x_3 \simeq 2x_2 + \frac{1}{4} &\simeq 1 \\ x_2 &\simeq \frac{3}{8}\,. \end{aligned} \tag{9.11.8}$$
- We backsubstitute into eq. (9.11.3a) to solve for x_1 (use approximate solution):
$$\begin{aligned} 2x_1 + x_3 \simeq 2x_1 - \frac{1}{2} &\simeq 4 \\ x_1 &\simeq \frac{9}{4}\,. \end{aligned} \tag{9.11.9}$$
- Hence our overall (approximate) solution is
$$(x_1, x_2, x_3) \simeq \left(\frac{9}{4}, \frac{3}{8}, -\frac{1}{2}\right). \tag{9.11.10}$$
- We validate our solution by substitution into the original equations:
$$10^{-20}x_1 + 4x_2 + x_3 \simeq 0 + \frac{3}{2} - \frac{1}{2} \simeq 1\,, \tag{9.11.11a}$$
$$x_1 + 2x_2 \simeq \frac{9}{4} + \frac{3}{4} \simeq 3\,, \tag{9.11.11b}$$
$$2x_1 + x_3 \simeq \frac{9}{2} - \frac{1}{2} \simeq 4\,. \tag{9.11.11c}$$
- The validation check is successful.

9.12 Gaussian elimination with partial pivoting

- Let us review the procedure of Gaussian elimination with partial pivoting.
- We are given a set of linear equations in n unknowns, indexed $(1, \ldots, n)$.
- We initialize the array of swap indices to $(1, \ldots, n)$.
- To eliminate x_1, we compare the values of a set of scaled amplitudes of the coefficients and select the first pivot row r_1.
- If $r_1 \neq 1$, we swap rows and update the array of the swap indices.
- We eliminate x_1 and obtain a set of $n-1$ equations in $n-1$ unknowns $x_2, \ldots, x_n$.
- We repeat the above procedure, looping through the rows $(2, \ldots, n)$. We select the second pivot r_2. If $r_2 \neq 2$, we swap rows and update the array of the swap indices.
- We repeat the procedure until we obtain a single equation in the variable x_n.
- This is the **elimination process.**
- **It is possible to encounter a zero pivot during the elimination process.**
- **After all of our precautions, if we still encounter a zero pivot, we exit and report that the equations are either inconsistent or not linearly independent.**
- **To avoid overflow/underflow problems, we may impose a tolerance on the minimum amplitude of the pivots.**
- If the elimination process is successful, we proceed to solve the equations.
- The **backsubstitution process** is employed to solve for $x_n, \ldots, x_1$ in reverse order.
- The backsubstitution process will not fail. The equations are consistent, there are no zero pivots, etc. If the equations are well conditioned, the computed answer will be reliable.
- ***Warning:*** The equations may be **ill conditioned**. Then the computed answer will not be reliable. There are ways to detect if the original set of equations is ill conditioned, but we have not discussed them in these lectures.

9.13 Memory and storage

- Notice that during the backsubstitution process, we do not need the original equations.
- We can solve for $x_n, x_{n-1}, \ldots, x_1$ in reverse order using the swapped and subtracted equations.
- Suppose the input coefficients a_{ij} are given to us as an $n \times n$ array, and the input values on the right hand side b_i are given to us as an array of length n.
- **Using Gaussian elimination (with pivoting, etc.), we can overwrite the input data.**
- We do not need to allocate extra memory to perform Gaussian elimination.

9.14 Gaussian elimination: multiple right hand sides

- Gaussian elimination has a weak point that it processes not only the coefficients a_{ij} but also the right hand side entries b_i.
- What would happen if we wished to solve a set of equations with the *same coefficients* a_{ij} as in eq. (9.3.1a) ***but with a different set of inputs on the right hand side?***
    1. We have to repeat the entire Gaussian elimination process.
    2. However, the pivots and swapping for the elimination depend only on the coefficients a_{ij} and **not on the right hand side values b_i.**
- If we wish to solve using the same coefficients a_{ij} but with a different right hand side, Gaussian elimination performs a lot of unnecessary repeated calculations.
- It would be better to perform the elimination process for the coefficients only once, and to **save the processed values** (after swapping and subtraction).
- As we noted above, the processed values can be saved in the same array as the original coefficients, and returned to the calling application. The calling application can use those processed coefficients to solve equations multiple times, with different right hand sides.
- Because of the swapping of rows during pivoting, we must also return the array of swap indices to the calling application.
- This is the motivating idea underlying the matrix algorithms to solve sets of linear equations.
- We shall study matrix algorithms next.