Dr. Z's Introduction to Linear Algebra Notes for Understanding the Basic Concepts (and using Maple)

By Doron Zeilberger

NOT responsible for any errors. The first finder of any error will get a dollar.

All the references are to the textbook used in this class

"Elementary Linear Algebra, a Matrix Approach", 2e, by L.E. Spence, A. J. Insel, and A. H. Friedberg .

Note: These short notes are meant to solidify the most important concepts so that you will see the *Forest* (in addition to seeing the *trees*), and don't confuse related but different concepts. Also, since I am a Maple person, and not a MatLab person, I will describe how to do problems, whenever possible, with the simpler Maple package `linalg`. There is a more complicated package called `LinearAlegbra`, that sometimes is needed, but whenever possible, I prefer `linalg`, and all (the computational, of course, not the conceptual) problems can be done with Maple. Of course, in the tests you can **not** use Maple, but it is great for checking the homework problems (the book only gives the final answers, not the intermediate step), and for solidifying the concepts.

If you are a Rutgers students, you can freely download Maple to your laptop (or desktop). All the Rutgers computer Labs (I think) have Maple (at any rate ARC does).

WARNING: These notes are not instead of the book! Their purpose to *emphasize* the **basic** concepts, remove some of the confusing things in the book (for example, where they make you work harder than necessary by telling you to do 'reduced row-echelon form' where the easier task of 'row-echelon form' (sometimes) suffices), and to tell you how to use Maple (using Maple will also clarify the concepts!).

Note: The part about Maple commands is **optional**. People who do not like programming, or Maple, can safely ignore them. However, I believe that doing the problems in Maple (in addition to doing it the traditional way, by paper-and-pencil) will enhance your understanding.

Basic Concept 1: One Equation in One Unknown (Variable)

An equation is a **puzzle** but not every puzzle is an equation.

For example:

Puzzle (Spoken): What is black and white and red [read] all-over?

First Answer: *Newspaper.* Second Answer: *A sun-burned zebra.*

Here there are (at least!, you are welcome to come up with more) answers. But the answers are not numbers.

Another example:

Another Puzzle In a certain village the (only) barber shaves all those men who do not shave themselves. Can you find someone who shaves himself?

Ans. No solutions.

An *equation* (in one variable) is a puzzle whose answers are *numbers*.

For example, the equation

$$2x + 1 = 5 \quad ,$$

is a puzzle. 'I am a number, if you multiply me by 2 and add 1 you would get 5, who am I'.

In this case is is easy to solve, and get $x = 2$.

Sometimes it is not so easy, for example

$$x^5 + x = 2 \quad ,$$

is also such a puzzle. Once someone tells you to **check** whether a proposed solution is indeed a solution, it is very easy, just plug-it in. For example, to check whether $x = 2$ is a solution, you ask yourself whether

$$2^5 + 2 = 2 \quad ,$$

but $34 = 2$ is wrong, so $x = 2$ is **not** a solution. On the other hand, if I propose $x = 1$ you get

$$1^5 + 1 = 2 \quad ,$$

and this is a true statement, so $x = 1$ is a solution.

How to solve one equation in Maple?

Maple has a nice command called `solve`. The syntax is

```
solve(eq,var) ;
```

For example, to solve the equation $x^5 + 1 = 2$ you do

```
solve(x**5+1=2,x);
```

`Maple convention:` Instead of doing

```
solve(eq=0,x);
```

one can simply type:

```
solve(eq,x);
```

In other words, if Maple does not see an equal sign (it automatically thinks that the right side is 0).

Basic Concept 2: One Linear Equation in One Unknown

The equation $x^2 - 3x + 1 = 0$ in the unknown x is **not**, linear (it is *quadratic*). Neither is the equation $x^3 + x - 5 = 0$ (it is cubic), neither is the equation $\cos x + e^x = 1$ (it is transcendental). One **linear** equation in **one** unknown (usually called x, but you can use any letter), is extremely simple, it has the form

$$a\,x = b \quad ,$$

where a and b are numbers, and it has exactly **one** solution $x = b/a$, unless $a = 0$, and $b \neq 0$, in which case it has **no** solutions, or $a = 0$ and $b = 0$ in which case it has **infinitely** many solutions, all x. (This is a bit confusing since in the equation '$0 = 0$', the variable, x, does not show up, but the answer to the problem: 'Solve the equation $0 = 0$ for the unknown x' is:

$$\{x \,|\, -\infty < x < \infty\} \quad .$$

i.e. *every* real number is a solution.

Basic Concept 3: System of Equations in Several variables

Problem: Solve the system of equations

$$\{x^2 + xy + y^2 = 1, x^3 + xy + y^3 = 3\} \quad ,$$

in the **unknowns** (alias **variables**) x, y. Don't worry, in this class you will never have to do such problems since these equations are *non-linear*. Nevertheless, Maple can do it. The command is

```
solve( {x**2+x*y+y**2=3,x**3+x*y+y**3=3 },{ x ,y});
```

and Maple will give you $\{x = 1, y = 1\}$, as well as other (complex-numbers) solutions.

Basic Concept 4: System of LINEAR Equations in Several variables

One linear equation in the variables x, y has the format

$$ax + by = c \quad ,$$

where a, b, c are **numbers**. a and b are the *coefficients*, and c is the *right hand side* .

One linear equation in three variables x, y, z has the format

$$ax + by + cz = d \quad ,$$

where a, b, c, d are **numbers**. a and b and c are the *coefficients*, and d is the *right hand side.*

In general, **one** *linear* equation in k variables, $x_1, \ldots, x_k$ has the format

$$a_1\, x_1 \; + \; a_2\, x_2 \; + \; \ldots \; + \; a_k x_k = b \quad ,$$

where $a_1, \ldots, a_k$ are numbers called the **coefficients** and b is a number called the 'right hand side'.

A *system* of m linear equations in n unknowns $x_1, \ldots, x_k$ has the format

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1k}x_k = b_1$$

$$\ldots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mk}x_k = b_m \quad .$$

Maple can always solve such a system **without** linear-algebra, using the 'high-school algorithm' (elimination and back-substitution), essentially as I did here

`http://sites.math.rutgers.edu/~zeilberg/LinAlg18/Sol.txt` .

For example, let's use Maple to solve exercise 4 in section 1.4 (p. 52). It is a system of two linear equations in the three variables x_1, x_2, x_3.

$$x_1 - x_2 - 3x_3 = 3$$

$$2x_1 + x_2 - 3x_3 = 0$$

In Maple, you would do

```
solve( { x1-x2-3*x3=3,2*x1+x2-3*x3=0 },{x1,x2,x3});
```

getting

```
x1 = 1 + 2 x3, x2 = -2 - x3, x3 = x3
```

Whenever you have something in the form `x3=x3` it means that `x3` is a *free* variable.

Moral: You don't need this class to solve systems of linear equations! This class teaches you more efficient ways, and other things (the abstract theory of matrices and suspaces of R^n), but *just* for solving systems of linear equations, you don't need matrices, and fancy stuff. The Maple command `solve` (or my home-made code `S(eq,var)`), can handle it.

How to Solve a system of Linear Equations using Matrices and Gaussian Elimination?

Do not confuse the hammer with the nail! If you want to nail a nail into a piece of wood, using a hammer is only one way of doing it. You can also use a shoe, or a heavy book, or any flat heavy object. Using a hammer is just the most efficient way.

The usual way to solve a system of linear equations is via an algorithm called *Gaussian Elimination.*

Warning: Gaussian Elimination is good for other things too. For example to find the *rank* of a matrix (and hence the nullity: 'number of **columns** minus the rank'), and to find out whether or not its columns are linearly independent. Some people get confused between the tasks (e.g. to decide whether a system is consistent, and finding the rank). These are two **different** problems, and the book is confusing that it introduced Gaussian elimination in the context of whether a system is consistent or inconsistent, and then solving it. This is only **one** application.

Writing a system of linear equations in matrix notation

The **matrix** notation for the system

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1k}x_k = b_1 \quad ,$$

$$\ldots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mk}x_k = b_m \quad .$$

is

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1k} \\ a_{21} & a_{22} & \ldots & a_{2k} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & \ldots & a_{mk} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_k \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_k \end{bmatrix} \quad .$$

Or, for short

$$A\mathbf{x} = \mathbf{b} \quad ,$$

where A is the **matrix of coefficients**,

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1k} \\ a_{21} & a_{22} & \ldots & a_{2k} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & \ldots & a_{mk} \end{bmatrix} \quad ,$$

x is the (column) vector of **variables** (unknowns)

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_k \end{bmatrix} \quad ,$$

b is the (column) vector of the **right side**

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_k \end{bmatrix} \quad .$$

Note: This started out as *shorthand* but it can also be used to define the 'matrix times (column) vector' operation.

An even shorter shorthand is not to mention the variable names $x_1, \dots, x_k$ explicitly, and form the **augmented** matrix

$$A\mathbf{b} \quad ,$$

i.e.

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1k} & b_1 \\ a_{21} & a_{22} & \dots & a_{2k} & b_2 \\ \dots & \dots & \dots & \dots & \\ a_{m1} & a_{m2} & \dots & a_{mk} & b_m \end{bmatrix} \quad .$$

First application of Gaussian Elimination: Deciding whether a system is consistent

Bring the **augmented matrix** to **row-echelon form**

(Warning: to find whether a system is consistent or not you don't need to do **reduced** row-echelon form, that takes longer).

If you get a row of the form (after you applied row-echelon form)

$$0000 \dots NotZero \quad ,$$

then the system is *inconsistent.* If you don't have this scenario (it is OK to have all zeros rows!) then the system is *consistent.*

Second application of Gaussian Elimination: Deciding whether a system is consistent AND Solving it

First you do row-echelon form. If you are lucky and you find out (as above), that the system is inconsistent, do not waste time to bring it to reduced row-echelon form, the answer is *no solution* (alias *inconsistent*). If you are not lucky, and the system is consistent, you must go all the way to

reduced row-echelon form (see section 1.4). Once you get it, you find the general solution as in section 1.3 .

How to represent vectors and matrices in Maple

A (row) vector is represented as a **list**, a matrix is a **list of lists**. For example, the (row) vector $[1, 2, 3]$ is represented in Maple

`[1,2,3]` .

A matrix is represented as a list of lists. For example, the augmented matrix

$$x_1 - 2x_2 - 3x_3 = 3$$

$$2x_1 + x_2 - 3x_3 = 0$$

is represented in Maple as

`[[1,-2,-3,3],[2,1,-3,0]]`.

How to compute Reduced-Row-Echelon Form in Maple?

Maple does not have a command for just row echelon form, since computers do not mind working harder. Of course, it does not *hurt* to have reduced-row-echelon form (if the computer does it), the command is

```
rref(matrix);
```

but you **must** first type:

`with(linalg):`   .

For example

```
with(linalg):  rref( [[1,-1,-3,3],[2,1,-3,0]]);
```

outputs

$$\begin{bmatrix} 1 & 0 & -2 & 1 \\ 0 & 1 & 1 & -2 \end{bmatrix}$$

Basic Concept 5: Pivot

Already with row-echelon form you can find out the *pivot entries*, they are the **first** non-zero entries in the not-all-zeros rows of the **row-echelon form** (also of the **reduced row-echelon form**).

A **pivot row** is a row with a pivot, alias a **not-all-zero** row.

A **pivot column** is a column with a pivot.

Warning: The notion of 'pivot' only makes sense for matrices that are already in row-echelon-form. It does not make sense for other matrices.

Later (section 2.3) it would make sense, using the **correspondence principle**. A column in the original matrix is called a 'pivot column' if its *corresponding* column, in the reduced-row-echelon form, is a pivot column. But everything is determined by the reduced-row-echelon form (and in fact, already by the row-echelon form, since pivots stay pivots).

Note the notion of 'pivot column' makes sense for any matrix, (since it hinges on the notion of 'pivot column' for the reduced-row-echelon form. But the notion of 'pivot' itself *only* makes sense for matrices in reduced-row-echelon form (or row-echelon form).

Warning: Every row has **at most** one pivot. the number of pivots is the number of not-all zero rows.

Also every column has at most one pivot. In the application to solving a system of linear equation, a column with a pivot corresponds to a **basic** variable, and a column that does not have a pivot corresponds to a **free** variable.

Basic Concept 6: Rank

The notion of *rank* is an attribute of *any* matrix (not necessarily an augmented matrix). To find it you also use *Gaussian Elimination*, but it is another use of it.

Find the row-echelon form of the matrix, and count the pivots, in other words, the number of rows that are not all-zero.

Important Fact: The rank of an $m \times n$ matrix is at most m. If it is exactly m, it means that its n columns (that are vectors in R^m) form a *generating set* of R^m.

Maple command:

```
rank(matrix);        .
```

For example, to do exercise 39 of section 1.4 (p. 53), do

```
rank([[1,1,1,1],[1,2,4,2],[2,0,-4,1]]);
```

getting 3. The **nullity** is the number of columns, n, minus the rank, so the nullity, in this case is $4 - 3 = 1$.

Basic Concept 7: Linear Combination

A specific linear combination of a set of vectors $\{\mathbf{u_1} \ldots, \mathbf{u_k}\}$ is any contraption of the form

$$c_1\mathbf{u_1} + c_2\mathbf{u_2} + \ldots + c_k\mathbf{u_k} \quad ,$$

where $c_1 \ldots, c_k$ are any numbers. For example, if the set is

$$\mathcal{S} = \{ \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad , \quad \begin{bmatrix} -1 \\ 3 \end{bmatrix} \} \quad ,$$

then **one** specific linear combination is

$$3 \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 2 \begin{bmatrix} -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix} + \begin{bmatrix} -2 \\ 6 \end{bmatrix} = \begin{bmatrix} 1 \\ 12 \end{bmatrix} \quad .$$

How to find whether a certain vector is a linear combination of a given set of vectors?

The input is a set of vectors (in R^m) , $\mathcal{S} = \{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}\}$ and another vector in R^m , $\mathbf{v}$. In order to find out whether $\mathbf{v}$ is a linear combination of the vectors in $\mathcal{S}$ you ask whether you can come up with some numbers $c_1, \ldots, c_k$ such that

$$\mathbf{v} = c_1\mathbf{u_1} + \ldots + c_k\mathbf{u_k} \quad .$$

You can (in simple cases) spell-it out and get a system of linear equations in c_1 , ... , c_k and try to solve them (using the high-school way), or use Gaussian elimination (sections 1.4 and 1.3) to solve the system

$$[\mathbf{u_1} \ldots \mathbf{u_k}] \, [c_1, \ldots, c_k]^T = \mathbf{v} \quad ,$$

whose augmented matrix is

$$[\mathbf{u_1} \ldots \mathbf{u_k} \, \mathbf{v}] \quad .$$

Basic Concept 8: Span

The set of *all* possible linear combinations of a **finite** set of vectors $\mathcal{S} = \{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}\}$ is the **infinite** set, called the **span** of $\mathcal{S}$, and denoted by $Span(\mathcal{S})$,

$$\{c_1\mathbf{u_1} + \ldots + \ldots c_k\mathbf{u_k} \,|\, -\infty < c_1 < \infty, \ldots, -\infty < c_k < \infty\} \quad .$$

Important Fact: Every member of $\mathcal{S}$ also belongs to $Span(\mathcal{S})$, since, for example

$$\mathbf{u_1} = 1 \cdot \mathbf{u_1} + 0 \cdot \mathbf{u_2} + \ldots + 0 \cdot \mathbf{u_k} \quad ,$$

so $\mathbf{u_1}$ is a linear combination of the members of $\mathcal{S}$. Similarly

$$\mathbf{u_2} = 0 \cdot \mathbf{u_1} + 1 \cdot \mathbf{u_2} + 0 \cdot \mathbf{u_3} + \ldots + 0 \cdot \mathbf{u_k} \quad ,$$

etc. Since **every** member of $\mathcal{S}$ belongs to $Span(\mathcal{S})$, it follows that $\mathcal{S}$ is a (tiny!) *subset* of the **huge** (infinite!) set $Span(\mathcal{S})$.

How to kick out superfluous members of a set and still have the same Span?

• The all-zero vector $\mathbf{0}$ can always be kicked out!

• If one vectors is a (non-zero) multiple of another one, one of them (but not both!) can be kicked out of the set without shrinking the span.

• If one of the vector is sum of two (or more) or more generally, any linear combination of other vectors, it can be kicked out.

Basic Concept 8': Generating Set

If the span of a set of vectors in R^m is the whole R^m it is called a *generating set* of R^n. The simplest generating set of R^m is the set of **standard vectors**, $\{\mathbf{e_1}, \ldots, \mathbf{e_m}\}$.

Basic Concept 9: Linear Dependence and Independence

A set of vector $\mathcal{S} = \{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}\}$, in R^m is **linearly dependent** if you can come up with numbers $c_1, \ldots, c_k$ **not all zero**, such that

$$c_1\mathbf{u_1} + \ldots + c_k\mathbf{u_k} = \mathbf{0} \quad .$$

If you can't, i.e. the only possibility is the obvious solution $c_1 = 0, c_2 = 0, \ldots, c_k = 0$ then the set is **linearly independent**.

How to decide whether a given set is linearly dependent by Inspection?

Same as how to decide whether you can kick out some vectors from a generating set (see above).

Warning: If you have a set of **two** vectors, then a quick way to find out whether the set is linearly dependent or linearly independent is to find out whether one is a multiple of the other. But if the set has more than two members, then if you can find two vectors that are multiples of each other, then you know for sure that the set is linearly dependent, but if you can't find such two vectors, do not jump to conclusions. It may be linearly dependent for other reasons. for example if one of the vectors is a sum of two (or three or whatever) other ones, or, more generally a linear combination of other ones. In that case you can't use inspection, and do it the 'official way'.

The Official Way to Decide whether a set of vectors in R^m is linearly independent

Given a set $\mathcal{S} = \{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_n}\}$, in R^m form the matrix whose columns are the members of $\mathcal{S}$

$$[\mathbf{u_1}\,\mathbf{u_2}\,\ldots\,\mathbf{u_n}] \quad ,$$

and find its rank! If the rank is n (the number of vectors, alias number of columns) then they are linearly **independent**. If it is less (it is never more!) then they are linearly **dependent**.

To summarize:

If the rank of the matrix $[\mathbf{u_1}\,\mathbf{u_2}\,\ldots\,\mathbf{u_n}]$ is m then it is a **generating set** (of R^m), if it is n then the set is **linearly independent**.

(Note: often it is neither. If $m = n$ and the rank is n then it is both a generating set and linearly independent. Such a set is called a **basis**).

Important special case

If $m = n$, and the set $\{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_n}\}$ is both a generating set and linearly independent (i.e. they form a basis) , then the **reduced-row-echelon-form** is the identity matrix I_n.

Important Fact: The rank of an $m \times n$ matrix is at most n (it is also at most m (see above)). If it is exactly n, it means that its n columns (that are vectors in R^m) are *linearly independent.*

Important Operation: Matrix Multiplication

Warning 1: If A and B are matrices, their **product** is not always defined. It is only defined if the number of columns of A is exactly the same as the number of rows of B.

Warning 2: When the product is defined, order (usually) matters. Usually AB and BA are very different. If A is an $m \times n$ matrix and B is an $n \times m$ matrix (when $m \neq n$), then AB and BA even have different sizes! Namely $m \times m$ and $n \times n$ respectively. When they are both square matrices of the same size, $n \times n$, then AB and BA are both $n \times n$, but usually completely different matrices. The fancy name for this is that matrix multiplication is **non-commutative**.

Some exceptions: One of them is the all-zero matrix. Another one is a when one is a multiple of the other. Yet another case where A and B commute is when one of them is the identity matrix. Yet another one exception is when A is a power of B (or vice vesra). But these are exceptions.

Thank God Matrix multiplication is not commutative

Quantum mechanics, that makes sure that matter is stable, uses this property, where A and B are matrices that correspond to **observables**. For example, position and momentum (and time and energy).

How to use Maple to multiply matrices

First go into Maple, and type

```
with(linalg);          .
```

Suppose that you want to do exercise #11 in section 2.1 of the book. Type:

```
A:=matrix([[1,-2],[3,4]]);
```

```
B:=matrix([[7,4],[1,2]]);
```

then to do AB, you type

```
C:=multiply(A,B);   .
```

Later on, to see the product, C, you type

```
evalm(C);   .
```

To do powers, for example A^5, you type

```
evalm(A**5)   .
```

To get the (i, j) entry of a matrix A, you type:

```
evalm(A)[i,j]; .
```

For example to get the $(1, 2)$ entry of A type

```
evalm(A)[1,2];
```

Important concept: Invertible matrix

A **square** matrix A is invertible, if you can come up with a square matrix of the same size, let's call it B, such that AB and BA are both the identity matrix of that size. In fact, it is enough to come up with either of them, and the other one will automatically also be true.

Important Operation: Inverse of a matrix

When A is invertible, i.e. A is square and there is a matrix B such that AB is the identity matrix, then B is called the **inverse** of A.

Important Notation:

The inverse of A (when it exists) is denoted by A^{-1}.

Important fact: If A is an invertible $n \times n$ matrix, then

$$A\,A^{-1} = I_n \quad , \quad A^{-1}\,A = I_n \quad .$$

How to find the inverse of a matrix in Maple

The command is

```
inverse(A);                .
```

For example, to do #9 in 2.4 type

```
with(linalg):  A:=matrix([[1,1,2],[2,-1,1],[2,3,4]]); inverse(A);
```

Important Object: Elementary Matrix

An elementary matrix is one that is obtained from the identity matrix by performing **one** elementary operation. The justification for the algorithm for computing the inverse of the matrix (described in section 2.4 of the book), uses Gaussian elimination. It gets the inverse as a product of the elementary matrices corresponding to the elementary row operations used in bringing A to reduced-row-echelon form. Of course, when you (or the computer) perform the algorithm, you don't need to know that, but this is the justification for the validity of the algorithm.

important Operation and Concept: Partitioning a matrix into blocks

You can break up a matrix into **blocks** and give them each a name. If you do it for two matrices A and B and the partitioning is **compatible** (see section 2.5), then you can do AB pretending the blocks are numbers (but be careful not to change the order).

If many blocks are zero or identity matrices, this could be a major time-saving device.

Note:

• If you partition an $m \times n$ matrix into mn 1×1 blocks, then it is essentially doing nothing.

• The other extreme, partitioning an $m \times n$ matrix into **one** block of size $m \times n$ is also essentially doing nothing.

The Maple package `LinearAlgebra`

For many things in Linear Algebra, the older package `linalg` suffices, but for the more advanced topics, for example, that of Lecture 11 (section 2.6 in the book), the **LU Decomposition**, the newer package `LinearAlgebra` is needed.

The first thing you need to do, once you have a Maple window is type

```
with(LinearAlgebra);
```

To define a matrix you use the command **Matrix** (with CAPITAL M) . For example, if

$$A = \begin{bmatrix} 1 & 3 \\ 4 & 5 \end{bmatrix} \quad ,$$

you type

```
A:=Matrix([[1,3],[4,5]]);
```

Suppose that

$$B = \begin{bmatrix} 11 & 3 \\ 14 & 5 \end{bmatrix} \quad ,$$

you tell Maple this by typing:

```
B:=Matrix([[11,3],[14,5]]);
```

To get their product you type

```
evalm(A&*B);
```

(Note that `evalm` stands for 'evaluate matrix').

You can also do matrix multiplication without naming them. For example, to compute the matrix product

$$\begin{bmatrix} 1 & 3 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} 11 & 3 \\ 14 & 5 \end{bmatrix} \quad ,$$

you type:

```
evalm(Matrix([[1,3],[4,5]])&*Matrix([[11,3],[14,5]]));          .
```

To get the k-th power of a matrix A you type `evalm(A&^k)`. For example to compute

$$\begin{bmatrix} 1 & 3 \\ 4 & 5 \end{bmatrix}^4 \quad ,$$

you type

```
evalm(Matrix([[1,3],[4,5]])&^4);          .
```

This even works for inverses. To get the inverse of the matrix A you type `evalm(A&^(-1))`. For example

```
evalm(Matrix([[1,3],[4,5]])&^(-1));          .
```

If the matrix does not have an inverse (i.e. is not invertible), e.g.

```
evalm(Matrix([[1,3],[1,3]])&^(-1));
```

you get an error message.

To get the reduced row echelon form of a matrix, you type

```
ReducedRowEchelonForm(A);
```

For example,

ReducedRowEchelonForm(Matrix([[1,3],[1,3]]));

Important Algorithm: The LU Decomposition

The input is any kind of matrix (does not have to be a square matrix), say an $m \times n$ matrix and the output (if lucky) are two matrices L and U such that

$$LU = A \quad .$$

L is an $m \times m$ **unit lower-triangular matrix** and U is an $m \times n$ **upper triangular matrix** (usually not unit, i.e. the diagonal does not have to be all 1).

To get it, you transform it to **row-echelon from** (NOT **reduced echelon form!**) and hope that you never have to resort to swapping (recall that in the first phase of Gaussian elimination, scaling is optional. Right now (for the LU problems), it is forbidden.). You are only allowed to use the elementary row operations of the type called $E(i, j; c)$, i.e. the operation '$cr_j + r_i \to r_i$').

The U is just the row-echelon form that you got. The L is obtained by first writing down the $m \times m$ identity matrix, and looking at the elementary row operations that showed up and for each such $E(i, j; c)$ you put $L_{i,j} = -c$. Everything else below and above the diagonal stays the same).

If there is a need to do one or more swapping operations, then there is a so-called **permutation matrix** (see the book), P, such that

$$LU = PA \quad .$$

(Note: in this class we will not deal with this more general case.)

Warning about LU

When you do things by hand, you are **not** allowed to use scaling in the reduction to row-echelon form. Recall that, in general, scaling is optional, but never crucial, but in the context of LU, it is forbidden.

Also **row-interchange** that *is* sometimes needed when you bring a matrix to row-echelon form, is **forbidden**. If you need to resort to row-interchange, it means that there is no LU decomposition.

Sanity Check: Check that in your answers, L is indeed **lower-triangular** (in fact it is **unit lower-triangular**, i.e. the diagonal entries are all 1). Also check that U is indeed **upper-triangular** (but usually not unit, and it may even some zero entries on the diagonal, sometimes).

How to use LU to solve a system

If you have a system $A\mathbf{x} = \mathbf{b}$, and $A = LU$, then you write it as

$$LU\mathbf{x} = \mathbf{b} \quad .$$

Then make a **change of variable**

$$\mathbf{y} = U\mathbf{x} \quad .$$

Then

$$L\mathbf{y} = \mathbf{b} \quad .$$

Then you transcribe it into every-day notation, and use high-school methods to find $\mathbf{y}$, by **forward substitution**. Having found $\mathbf{y}$, you solve

$$U\mathbf{x} = \mathbf{y} \quad ,$$

also by transcribing it, and use the high-school way.

LU Decomposition in Maple

First type:

```
with(LinearAlgebra);
```

Then type:

```
LUDecomposition(A);
```

For example, to do #8 in section 2.6 (p. 164) you type:

```
LUDecomposition(Matrix([[-1,2,1,-1,3],[1,-4,0,5,-5],[-2,6,-1,-5,7], [-1,-4,4,11,-2]]));
```

• The first output is the permutation matrix P (that in all of **your** homework problems (i.e. section 2.6 , problems 1-8) should be the identity matrix)

• The second output is L

• The third output is U

It is always good to check. If you type

```
evalm(%[2]&*%[3]);
```

you would get A back.

For the more general case, where it is not possible to find an LU decomposition, the first output is a permutation matrix (see the book), such that $PA = LU$.

For example, to do #18 of section 2.6 (NOT required for this class),

```
LUDecomposition(Matrix([[0,2,-1],[2,6,0],[1,3,-1]]));
```

gives you

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad ,$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{bmatrix} \quad ,$$

$$U = \begin{bmatrix} 2 & 6 & 0 \\ 0 & 2 & -1 \\ 0 & 0 & -1 \end{bmatrix} \quad .$$

You are welcome to check that

$$PA = LU \quad .$$

Important Operation: Determinant

If A is a **square** matrix, then there is an important number called the **determinant** of A written,

$$\det(A) \quad ,$$

and sometimes, in the stupid notation $|A|$ (that has **nothing** to do with 'absolute value').

The determinant **determines** whether the matrix is invertible. If $\det(A) \neq 0$ then it is invertible. If $\det(A) = 0$ then it is **not invertible**.

How to compute a determinant

A 2×2 matrix is really easy

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc \quad .$$

For a 3×3 matrix, you do **cofactor expansion** (see the book), and you can use *any* row, and *any column.*

Remember:

• for an odd-numbered row (i.e. the first and third row if we are talking about 3×3 matrices) the signs are plus, minus, plus

• for an even-numbered row (i.e. the second row if we are talking about 3×3 matrices) the signs are minus, plus, minus

So computing a 3×3 determinants requires computing three 2×2 determinants, and doing some more calculations.

Similarly for column expansions .

You can also do a 4×4 determinant this way, reducing it to computing four 3×3 determinants plus more tedious calculations, but it is **not** recommended. It takes for ever.

The **efficient** way is to use the **simplification** rules gotten by the elementary row operations, except that **now** you can also operate with columns, in other words, you are welcome to also use *elementary column operations.*

• $kr_j + r_i \to r_i$ (or $kc_j + c_i \to c_i$) does **not** change the determinant

• $kr_i \to r_i$ (or $kc_i \to c_i$) You 'factor out' k. The answer is k times the determinant of the matrix obtained by dividing the given row (or column) by k.

• $r_i \leftrightarrow r_j$ (or $c_i \leftrightarrow c_j$) multiplies the determinant by -1.

Important Property of determinant

$$\det(AB) = \det(A)\,\det(B) \quad .$$

If A is invertible:

$$\det(A^{-1}) = \frac{1}{\det(A)} \quad .$$

When are these useful?

• Suppose that you are given two matrices A and B for which it is *easy* to find their determinants. For example, either upper-triangular or lower-triangular, for which the determinant is simply the product of the diagonal entries. Then you are asked to find $\det(AB)$. You would be stupid to first compute AB and then take the determinant. It would be much more efficient to compute $\det(A)$ and $\det(B)$ and then use the above fact that $\det(AB) = \det(A).\det(B)$.

• Suppose that you are a given a matrix A and you have to find $\det(A^{-1})$. You could, of course, first find A^{-1}, and *then* compute $\det(A^{-1})$, but it would be much more efficient to compute $\det(A)$, and then take the reciprocal.

How to compute determinants in Maple

In the simpler Maple package, `linalg`, the command is `det(A)`.

For example, to do

$$\det \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 7 \\ 13 & 14 & 17 \end{bmatrix} \quad ,$$

in `linalg` you type

```
with(linalg):  det(matrix([[1,2,3],[3,4,7],[13,14,17]]));
```

With the more advanced package, `LinearAlgebra`, you type

```
with(LinearAlgebra):  Determinant(Matrix([[1,2,3],[3,4,7],[13,14,17]]));
```

Very Important Concept: Subspace

A subset V, of R^n is a **subspace** if it satisfies the following **three** 'axioms'

(i) $\mathbf{0} \in V$ [In plain English: the zero vector belongs to V]

(ii) $u, v \in V \Rightarrow u + v \in V$ [In plain English: the sum of any two members of V is yet another member of V]

(iii) $u \in V$ and $k \in R \Rightarrow k\,u \in V$ [if you multiply *any* member of V by *any* real number, you would get yet another member of V]

How to prove that a candidate subset of R^n is a subspace?

You have to show, logically, that the three axioms (properties) are all satisfied.

Problem: Prove (logically) that the following set

$$V = \{ \begin{bmatrix} c \\ 2c \\ -3c \\ 5c \end{bmatrix} \; ; -\infty < c < \infty\} \quad ,$$

is a subspace of R^4 .

Solution: First note that the members of V are vectors with four components, hence they all belong to R^4. Hence V is a *subset* of R^4. To prove that it is also a **subspace** of R^4 we have to prove that the three axioms are true.

(i) The choice $c = 0$ yields the zero vector. Since 0 is a real number, $\mathbf{0}$, alias $[0, 0, 0, 0]^T$ belongs to V.

(ii) Take any two members of V, let's call them u and v. This means that

$$u = \begin{bmatrix} c_1 \\ 2c_1 \\ -3c_1 \\ 5c_1 \end{bmatrix} \quad ,$$

for some *specific* number c_1. Similarly

$$v = \begin{bmatrix} c_2 \\ 2c_2 \\ -3c_2 \\ 5c_2 \end{bmatrix} \quad ,$$

for another specific number c_2. Adding them up, we have

$$u + v = \begin{bmatrix} c_1 \\ 2c_1 \\ -3c_1 \\ 5c_1 \end{bmatrix} + \begin{bmatrix} c_2 \\ 2c_2 \\ -3c_2 \\ 5c_2 \end{bmatrix} = \begin{bmatrix} (c_1 + c_2) \\ 2(c_1 + c_2) \\ -3(c_1 + c_2) \\ 5(c_1 + c_2) \end{bmatrix} \quad .$$

Calling $c_1 + c_2 = c_3$, c_3 is, of course, also a number, so $u+v$ can be expressed in the *format* required to grant membership of V with the choice $c = c_1 + c_2$.

(iii) Take any member of V, let's call it u, and *any* number k. u has the form

$$u = \begin{bmatrix} c_1 \\ 2c_1 \\ -3c_1 \\ 5c_1 \end{bmatrix} \quad ,$$

for *some* number c_1. Multiply by k, and get:

$$ku = \begin{bmatrix} (kc_1) \\ 2(kc_1) \\ -3(kc_1) \\ 5(kc_1) \end{bmatrix} \quad ,$$

so it has the right format for membership in V with the choice $c = kc_1$.

How to prove that a candidate subset of V is NOT a subspace

This is usually easier and faster. All you need is to come up with one of the three axioms that is violated.

Example: Prove that the set

$$V = \{ \begin{bmatrix} 1+c \\ 2+c \\ 3+c \\ 5+c \end{bmatrix} \; ; -\infty < c < \infty \} \quad ,$$

is **not** a subspace of V.

Solution: We claim that $\mathbf{0} \notin V$. Suppose that it is. Then there must be a number c such that

$$\begin{bmatrix} 1+c \\ 2+c \\ 3+c \\ 5+c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad .$$

Spelling it out, we must have

$$1 + c = 0 \quad , \quad 2 + c = 0 \quad ,$$

and two more such equations ($c + 3 = 0$ and $c + 5 = 0$, but they are not needed right now). So we get that $c = -1$ and $c = -2$. This is a **contradiction**, there can't be such a c, hence the assumption that $\mathbf{0}$ belonged to V lead to nonsense, hence axiom (i) is violated, hence V is **not** a subspace of R^4.

(Note that it is not necessary to look at the other two axioms, once something is bad for one reason, you don't have to find other reasons why it is bad).

Another Example: Prove that the set

$$V = \{ \begin{bmatrix} c \\ 2c \\ 3c \\ 5c \end{bmatrix} ; 0 \leq c < \infty \} \quad ,$$

is **not** a subspace of V.

Solution: Note that axioms (i) and (ii) are satisfied. But axiom (iii) is not. Take any member of V, say when $c = 1$

$$u = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 5 \end{bmatrix} \quad .$$

Now multiply it by -1 (or any negative number for that matter)

$$(-1)u = \begin{bmatrix} -1 \\ -2 \\ -3 \\ -5 \end{bmatrix} \quad .$$

But this is **not** a member of V, since the c is negative.

Important Concept: Generating set of a subspace V.

Recall that a (usually finite) subset of R^n, let's call it $\mathcal{S}$, is a *generating set* of R^n if its **span** $Span(\mathcal{S})$ is the whole of R^n. This definition extends to any subspace of R^n .

Important definition; $\mathcal{S}$ is a generating set of V if $Span(\mathcal{S}) = V$.

Recall that the Span of a set $\mathcal{S}$ is the set of *all* linear combinations of the members of $\mathcal{S}$.

Easy but important fact:

A generating set of a subspace of the form

$$Span(\mathcal{S}) \quad ,$$

is $\mathcal{S}$.

Recall that a set of vectors in R^n is **linearly independent** if none of its members is a linear combination of other members (or equivalently, that $\mathbf{0}$ can never be expressed as a non-trivial linear combination of its members). In other words, each of its members is **crucial** for its span not to change. No vector can't be kicked out.

Important Concept: Basis

A (finite) set of vectors $\mathcal{S}$ is a **basis** of a subspace V if:

• $\mathcal{S}$ is a generating set for V, i.e. $Span\mathcal{S} = V$

• $\mathcal{S}$ is **linearly independent**

important Fact: Every subspace (except the trivial zero subspace) has **many** different bases, in fact, infinitely many of them. but they all have the **same** number of elements. That common number is called the **Dimension**.

Since the standard basis of R^n $\{\mathbf{e_1}, \mathbf{e_2}, \quad \dots \quad, \mathbf{e_n}\}$, is indeed a basis (it is both linearly independent and a generating set) the dimension of R^n is indeed n (as it should be by common sense). Hence *every* basis of R^n **must** have n members.

Quick way to show that a proposed set $\mathcal{S}$ is NOT a basis of R^n

If its number of elements is **not** n, then forget it! There is no way that it can be a basis.

Example; Explain why

$$\{ \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix} , \begin{bmatrix} 4 \\ 1 \\ 7 \end{bmatrix} \} \quad ,$$

is **not** a basis of R^3

Solution: Any basis of R^3 **must** have three members. Since $\mathcal{S}$ only has two members, there is no way that it can be a basis.

Another Example; Explain why

$$\{ \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix} , \begin{bmatrix} 4 \\ 1 \\ 7 \end{bmatrix} , \begin{bmatrix} 14 \\ 11 \\ 27 \end{bmatrix} , \begin{bmatrix} 44 \\ 11 \\ 47 \end{bmatrix} \} \quad ,$$

is **not** a basis of R^3

Solution: Any basis of R^3 **must** have three members. Since $\mathcal{S}$ has four members, there is no way that it can be a basis.

A not so Quick way to show that a proposed set $\mathcal{S}$ is NOT a basis of R^n

If the set $\mathcal{S}$ has exactly n members, then the algorithmic way to prove that it is a basis for R^n is to form the $n \times n$ matrix whose columns are the members of $\mathcal{S}$, and find its rank. If the rank is n then it is a basis, otherwise (i.e. if the rank is less than n, it can never be more) it is not. But sometimes, by inspection, you can spot that the set is not linearly independent, and in that case you can immediately disqualify it from being a basis. This also applies to subspaces of R^n

Example Explain why $\mathcal{S}$

$$\mathcal{S} = \{ \begin{bmatrix} 2 \\ 3 \\ 5 \\ 7 \end{bmatrix} , \begin{bmatrix} 4 \\ 11 \\ -5 \\ -7 \end{bmatrix} , \begin{bmatrix} 6 \\ 14 \\ 0 \\ 0 \end{bmatrix}$$

can **not** be a basis of the subspace $Span(\mathcal{S})$.

Solution: Obviously $\mathcal{S}$ is a generating set of $Span(\mathcal{S})$ (this is always true), but by inspection the last vector is the sum of the first two, hence the set $\mathcal{S}$ is **not** linearly independent. Hence it is **not** a basis.

Algorithmic way to find a basis of $Span(\mathcal{S})$

If $\mathcal{S}$ is a set of n vectors in R^m, form the $m \times n$ matrix consisting of the vectors of $\mathcal{S}$, let's call it A, then perform the 'Forward pass' (first-phase) of Gaussian elimination to find out the pivots, and hence the pivot columns of the row-echelon form. The **corresponding** numbered columns in A, a certain subset of $\mathcal{S}$, is **a** basis of $Span(\mathcal{S})$.

In particular the **rank** of A is the dimension of $Span(\mathcal{S})$ (alias column-space of A).

Algorithmic way to find a basis for the Null Space of A

This is more time-consuming. If you want to find a basis for the null-space of A, you have to solve the system

$$A\mathbf{x} = \mathbf{0} \quad ,$$

as was done in chapter 1. In other words, find the **reduced-row-echelon form** of the augmented matrix $[A\,\mathbf{0}]$. Then decide who are the **basic** variables (corresponding to the pivot columns), and who are the **free** variables (corresponding to the other columns). Express everything in terms of the free variables (and for any free variable, write $x_i = x_i$), then write it in **vector notation**, then extract the free variables, and the set of vectors multiplied by those free variables is **a basis**.

For an example, see Attendance quiz 15

`http://sites.math.rutgers.edu/~zeilberg/LinAlg18/p15S.pdf` .

WARNING: A basis is always a **set**. Don't forget the {} or you would get no credit!

The Four Subspaces associated with an $m \times n$ matrix A

• $Col(A)$, the **column space** of A, alias the span of its set of columns, a certain subspace of R^m. Its **dimension** equals the **rank** of A .

• $Null(A)$, the **nullspace** of A, $Null(A)$, a certain subspace of R^n, defined as the set of vectors $\in R^n$ such that $A\mathbf{x} = \mathbf{0}$. Its **dimension** is the **nullity** of A, alias $n - rank(A)$.

• $Row(A)$, the **row space** of A, alias the span of its set of rows, a certain subspace of R^n. Its

dimension also equals the **rank** of A .

• $Null(A^T)$, the **nullspace** of A^T, $Null(A^T)$, a certain subspace of R^m, defined as the set of (row) vectors $\mathbf{x} \in R^m$ such that $\mathbf{x}A = \mathbf{0}^T$. Its **dimension** is $m - rank(A)$.

Important Objects: Eigenvalue and Eigenvector

eigenvector:

Given a square $n \times n$, matrix A, a **non-zero** vector $\mathbf{x}$ is an **eigenvector** of A if , for some (possibly zero) **number**, t,

$$A\mathbf{x} = t\mathbf{x} \quad .$$

t is **the eigenvalue** corresponding to the eigenvector $\mathbf{x}$.

(Note that for each eigenvector there is a **unique** eigenvalue).

eigenvalue:

Given a square $n \times n$, matrix A, a number t is an **eigenvalue** if there exists a **non-zero** vector $\mathbf{x}$ such that

$$A\mathbf{x} = t\mathbf{x} \quad .$$

$\mathbf{x}$ is **an eigenvector** corresponding to t.

(Note that for each eigenvalue, there are infinitely many eigenvectors corresponding to it, since once you have an eigenvector, so is any non-zero multiple of it, and if you have two eigenvectors corresponding to the same eigenvalue so is their sum. In fact, for any given eigenvalue the set of eigenvectors corresponding to it is a **subspace** of R^n, called the **eigenspace**. It is the nullspace of $A - tI_n$).

How to find eigenvalues of a matrix A?

First you find the determinant of $A - t\, I_n$,

$$\det(A - t\, I_n) \quad .$$

This is called the **characteristic equation**. Then you solve it, getting possibly multiple roots, and possibly complex roots. These are the eigenvalues.

How to find a basis for the eigenspace of an eigenvalue t of matrix A

You solve the system

$$(A - tI_n)\mathbf{x} = \mathbf{0},$$

either by the 'high-school way' (if it is a 2×2 matrix, and even the 3×3 case) or by the Gaussian elimination way.

WARNING: An eigenvector is **NEVER** the zero-vector. If the only answer that you get is $\mathbf{0}$ it means that the proposed eigenvalue is not actually an eigenvalue, and you messed up, when you found the eigenvalues. If anyone on the Final Exam will return, for example, $\{\begin{bmatrix} 0 \\ 0 \end{bmatrix}\}$, as a basis to the eigensapce, or $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ as an eigenvector, he or she will FAIL this class, even if everything else is perfect.

How to find eigenvalues in the Maple package linalg ?

For example to find the eigenvalues of the matrix

$$\begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}$$

You type

```
with(linalg):eigenvalues([[3,1,1],[1,3,1],[1,1,3]]);
```

getting

```
5, 2, 2    .
```

How to find eigenvalues and eigenvectors in the Maple package linalg?

For the above matrix, you type:

```
with(linalg):eigenvectors([[3,1,1],[1,3,1],[1,1,3]]);      ,
```

getting

```
[2, 2, {[-1, 0, 1], [-1, 1, 0] }], [5, 1, {[1, 1, 1]}]      ,
```

which means that 2 is an eigenvalue with **multiplicity** 2, whose **eigenspace** has basis $\{[-1, 0, 1], [-1, 1, 0]\}$, (that is two-dimensional) and an eigenvalue 5 with **multiplicity** 1, whose eigenspace is the one-dimensional subspspace with basis $\{[1, 1, 1]\}$.

Important Concept: 'Diagonalizable'

A square matrix A, of size $n \times n$, is **diagonalizable** if there exists an **invertible** matrix P (of the same size, $n \times n$, of course), and a **diagonal** ($n \times n$) matrix, D such that

$$A = PDP^{-1} \quad .$$

Equivalently,

A square matrix A, of size $n \times n$, is **diagonalizable** if there exists an **invertible** matrix P such that

$$P^{-1}AP \quad ,$$

is a diagonal matrix (called D).

The diagonal matrix, D is called the *diagonalization* of A.

Notes: 1.Every diagonal matrix is considered diagonalizable, just take P to be the identity matrix.

2. Most, but **not all** square matrices are diagonalizable. The simplest example that is not diagonlizable is

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad .$$

Important Process: Diagonalization

Input: A squre-matrix A, of size $n \times n$, say.

Output: a diagonal matrix D, and an invertible matrix P, such that $A = PDP^{-1}$, if they exist, or 'does not exist.'

Description:

Step 1: Find the **eigenvalues** of A, by first finding $\det(A - tI_n)$, setting it equal to zero, and solving, getting, if lucky, n distinct roots. (if real) Order them in increasing order (or any order, but you have to stick to the order). You are not supposed to get complex roots in this class, but in general, you would have to include them.

Step 2: For each eigenvalue t_i, find a corresponding eigenvector, i.e. a **non-zero** vector $\mathbf{p_i}$ such that

$$A\mathbf{p_i} = t_i A \quad .$$

If all the eigenvalues are distinct (i.e. multiplicity 1) then D is the diagonal matrix whose diagonal entries are (in that order) t_1, t_2, etc. (of course, all other entries are 0), and P is the matrix whose columns are (in that order) $\mathbf{p_1}, \mathbf{p_2}$ etc.

(If $n = 2$ then delete the 'etc.')

If you have an eigenvalue with multiplicity larger than one, find a basic for the eigenspace. If the dimension of the eigenspace equals the multiplicity, then line them up as above, if it is less, the answer is 'does not exist'.

Optional (but Recommended) Important Final Step: Check!

After you found D and P, make sure that, indeed

$$A = PDP^{-1}$$

Warning: If you get the wrong answer, due to a careless computational error, and do not check you will get **no** partial credit. If you check and realize that you messed up, and say so, you may get up to one half partial credit.

Important Shortcut for the inverse of a 2×2 **matrix**

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \quad .$$

Reminder: For any scalar k, kA is the matrix obtained by multiplying *every* entry of A by k.

Important Operation: Dot Product

Given two vectors with the same number of components, i.e. both in R^n for some n,

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{bmatrix} \quad , \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix}$$

The **dot product** is

$$\mathbf{u}.\mathbf{v} = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n \quad .$$

Note: $\mathbf{u}.\mathbf{v} = u^T v$.

How To Compute Dot Product in Maple

First type

`with(LinearAlgebra)`, then, after you defined the vectors **u** and **v**, you type

`DotProduct(u,v) ;`

Important Concept: Orthogonality

Two vectors, **u** and **v**, are **orthogonal** if their dot product is 0, i.e. if

$$\mathbf{u}.\mathbf{v} \; = \; 0 \quad .$$

Important Operation: Norm

The **norm** of a vector $\mathbf{u}$,denoted by $||\mathbf{u}||$, is $\sqrt{\mathbf{u}.\mathbf{u}}$, in other words:

$$|| \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_n \end{bmatrix} || = \sqrt{u_1^2 + u_2^2 + \dots + u_n^2} \quad .$$

Important Operation: Normalization: The normalization of a vector $\mathbf{u}$ is obtained by dividing it by its norm. i.e. $\mathbf{u}/||\mathbf{u}||$. Note that it is a unit vector, i.e. has norm 1.

Important Concept: An **orthogonal set** of vectors in R^n is a set where every pair in the set is orthogonal.

Important Concept: An **orthonormal set** of vectors in R^n is an orthogonal set (see above), with the additional property that every member has norm 1.

Important Property of an Orthogonal (and hence also Orthonormal) Set:

It is always **linearly indpenendent**, hence forms a *basis* for the subspace spanned by it.

Given an orthonormal basis, let's call it $\{\mathbf{u_1}, \dots \mathbf{u_k}\}$, for some subspace of R^n, it is very easy to express *any* vector, $\mathbf{v}$, in that subspace, as a linear combination of the mebers of the basis (Recall that,in general, it is a big pain. It requires solving a system of equations $[\mathbf{u_1}, \dots \mathbf{u_k}][c_1, \dots, c_k]^T = \mathbf{v}$, that is very tedious thing to do).

$$\mathbf{v} = (\mathbf{v}.\mathbf{u_1})\mathbf{u_1} + \dots + (\mathbf{v}.\mathbf{u_k})\mathbf{u_k} \quad ,$$

In other words, to find the respective coefficients, all you need is take dot products, no need to solve a tedious system of equations.

Important Algorithm: Gram-Schmidt (w/o Normalization)

Input: A *linearly independent* set of vetors in R^n

Output: An orthogonal set with the same span.

How to do it: See Theorem 6.6 in the book (p. 378).

Important Algorithm: Gram-Schmidt (with Normalization)

Once you have an orthogonal set (done as above), to get an orthonormal set, you divide each and every member by its respective norm.

Gram-Schmidt in Maple

The command

(once you typed `with(LinearAlgebra):`), is:

```
GramSchmidt([w1,w2,w3, ...])  ;
```

where `w1, w2, w3,` are the vectors.

For example, to do problem 14 in section 6.2, type

```
GramSchmidt([<1,-1,0,2>,<1,1,1,3>,<3,1,1,5>]);
```

To get an orthonormal set type

```
GramSchmidt([w1,w2,w3, ...], normalized=true);
```

For example, to do the second part of problem 14 in section 6.2, type

```
GramSchmidt([<1,-1,0,2>,<1,1,1,3>,<3,1,1,5>],normalized=true);
```

Important Algorithm: The QR Decomposition

Given an $m \times n$ matrix A with linearly independent columns it can always be written as

$$A = QR \quad ,$$

where

• Q is an $m \times n$ matrix whose set of columns form an orthonormal set

• R is an $n \times n$ **upper triangular** matrix.

How to find it?: See the book, pp. 381-383.

The command

(after you typed `with(LinearAlgebra):`) , is:

```
QRDecomposition(A);
```

where A is the matrix. For example, to do exercise 28 in section 6.2 (p. 385) in the book, type

```
QRDecomposition(<<-1,3,4>|<-7,11,3>>);   .
```

Very Important Algorithm: The Least Square Line

If you have n data points

$$(x_1, y_1), \ldots, (x_n, y_n)$$

the **least-square** line $y = a_0 + a_1 x$ is computed as follows.

The formula (that you should memorize, it is SO IMPORTANT) is:

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = (C^T C)^{-1} C^T \mathbf{y} \quad ,$$

where

$$C = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ . & . \\ . & . \\ . & . \\ 1 & x_n \end{bmatrix} \quad , \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix} \quad .$$

Here is how to actually do it.

Step 1: Set-up the $n \times 2$ matrix, whose first column is all 1 and its second column is $[x_1, \dots, x_n]^T$, in other words

$$C = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ . & . \\ . & . \\ . & . \\ 1 & x_n \end{bmatrix}$$

Step 2: Set up the **column-vector**, $\mathbf{y}$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix} \quad .$$

Step 3: Compute the 2×2 matrix $C^T C$

Step 4: Compute the inverse of $C^T C$, $(C^T C)^{-1}$. Since $C^T C$ is 2×2 you can use the shortcut

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \quad .$$

Step 5: Compute the 2×1 column vector

$$C^T \mathbf{y} \quad .$$

Step 6: Compute

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = (C^T C)^{-1} C^T \mathbf{y} \quad ,$$

and write down $y = a_0 + a_1 x$.

How to do LeastSquares in Maple

It is a bit complicated, since you need to do some preprocessing. Luckily, you can use the following Maple code:

`http://sites.math.rutgers.edu/~zeilberg/LinAlg18/LS.txt`

Copy-and-paste (from the web-browser), this file, onto a Maple window.

Given a list of points, `L`, just type

```
LS(L);.
```

For example, to solve ex. 2 of section 6.4 of the book (p.409), type:

```
LS([[1,30],[2,27],[4,21],[7,14]],x);
```

To see a scatter-plot, and the line, type:

```
PLS([[1,30],[2,27],[4,21],[7,14]]);
```
