A Brief Introduction to Applications of Linear Algebra

By James Derry

> …when the chips are down we close the office door and compute with matrices like fury.
> -- Irving Kaplansky, from *Paul Halmos: Celebrating 50 Years of Mathematics*

We've written this chapter as a set of applications for a matrix calculator for the user who has downloaded & installed MtrxCal onto a Palm for the first time, but may not know what to do with it. MtrxCal, especially designed for the palm platform by ADACS LLC, can be downloaded from http://www.adacs.com/

Maybe you don't know what good MtrxCal is because you're new to linear algebra or because putting your knowledge of linear algebra to practical use has so far escaped you. This chapter intends to briefly introduce applications of linear algebra that are practical to run on MtrxCal. If there's any theory in this chapter, it's accidental (and serves as a yet briefer introduction to linear algebra).

Let's start with the design philosophy behind the calculator. MtrxCal is to linear algebra what CplxCalPro is to algebra and complex numbers: it does operations (like addition, subtraction, multiplication, and division) on data objects (like matrices or complex numbers), relieving you of the grunt work. You enter the right data in the right order, you get the right result.[1]

MtrxCal was designed to be highly compatible with MatLab, math software that's used by scientists and engineers. MatLab uses matrices and arrays to do all its computations. This means that a solution you develop on MtrxCal should run on MatLab with few if any modifications.

Because MtrxCal was written to run on Palm handheld devices, it makes the perfect calculator on which to run modest-size problems that yield solutions more easily to a linear algebra approach than to an algebra one.[2] Look these applications over to see what we're talking about:

[1] CALCULATORS DO NOT RELIEVE YOU OF THINKING! Yes, you enter a formula you see in a textbook into a calculator in the right order, and you get the right answer; but so what? For those of you who are using our calculators in the classroom, a caveat: calculators like MtrxCal and CplxCalPro are aids intended to shorten the time you need to solve a problem, or to make a concept clear (as you might do by graphing the output), or to double-check your answers.

[2] NOT ALL LINEAR ALGEBRA PROBLEMS ARE DIRECTLY SOLVABLE ON MTRXCAL. This is an important warning that some who work with linear algebra might consider superfluous. MtrxCal is intended strictly for numerical analyses. Many lin alg textbooks consider parametrized solutions for systems of linear equations (that is, problems in which one or more variables are expressed in terms of other variables) in their first chapter. Solving variables in terms of variables is beyond the current version of MtrxCal. This means that the function **A\b**, which solves systems of linear equations, works only for square matrices; that is, in systems in which the number of equations equals the number of variables. When using this guide to see how to solve your own problems, please check the **REQUIRES:** entry on the problems you're considering in this guide to ensure can be solved on our calculator. If your problem doesn't meet the requirement to solve on MtrxCal, please check the appendix at the back of this guide for suggestions on how to proceed.

Geometry

Sometimes a picture is worth a thousand words. In linear algebra, coordinate systems like the (x,y) ones that we draw our geometry problems in and the (x,y,z) ones that 3D graphics are rendered in are called vector spaces. With MtrxCal we can visualize matrix operations by rendering them as graphics in a 2D vector space. In this section we shall investigate matrix operations upon graphics objects. We begin by learning how MtrxCal creates graphics. We shall do this by drawing a circle, then analyzing what we have done. We start with a circle because it is easy to draw on MtrxCal.

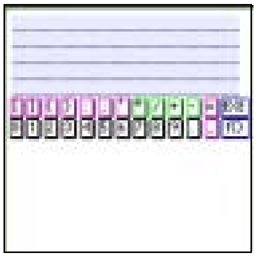


Figure 1

MtrxCal was designed to render angular measurements in radians, and to plot complex numbers as points in 2-space, where the real part is plotted as the x-coordinate and the imaginary part as the y-coordinate. Using $z = \cos\theta + j\sin\theta$, where θ = the angular measurement in radians and **j** denotes the imaginary part of the complex number, **plot(z)** on MtrxCal will plot a unit circle (radius=1). Let's try it:

1. Turn on MtrxCal and go to the matrix keyboard layout (if you don't already see the layout show in fig. 1, press **menu**, **keys**, **matrix**)

2. Enter these lines on MtrxCal:
   theta=0:.1:2*pi;
   z=(cos(theta)+j*sin(theta));
   plot(z)

3. Press **EXE**:

We end each line whose result we don't want to see with a semicolon.

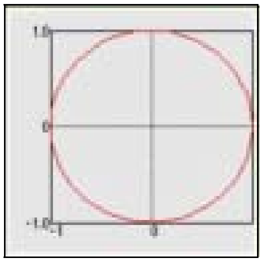


Figure 2

MtrxCal renders the circle and automatically scales the plot. Tap the screen to return to the matrix keyboard layout. The first line reads **theta=0:.1:2*pi;** If we read this line aloud, we might say, "For theta, generate values from 0 to 2pi, incrementing by .1." So the first line creates 63 values: 0, .1, .2, .3,…6.0, 6.1, 6.2. Each value is then plugged into our equation for **z**, and the output is a one-dimensional array holding the 63 results of the calculation **(cos(theta)+j*sin(theta))**. That array is **z**. We can see this for ourselves if we remove the semicolon from the end of the third line and delete the last line, then press **EXE**. Now MtrxCal displays the array of complex numbers from which it plotted the circle. If we wish to know the size of the array, we can replace the semicolon on line three and write **size(z)** on the last line. **size()** takes a matrix as its argument and returns a row array whose first element is the number of rows in the matrix and whose second element is the number of columns. **size(z)** in this instance returns **[1 63]**.

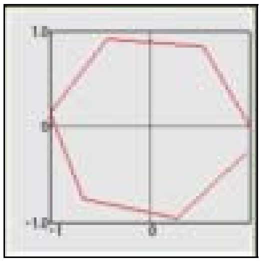


Figure 3

Before we manipulate our unit circle with a transformation matrix, let's change the increment value of theta to 1, like this: **theta=0:1:2*pi;** and see what happens when we press **EXE**:

The result is shown in figure 3.

From what we know of how MtrxCal plots, this shouldn't be so surprising. **size(z)** returns **[1 7]**, which as we know means that seven values are generated from **theta=0:1:2*pi**, which in turn generate seven (x,y) pairs from **(cos(theta)+j*sin(theta))**; and these points get plotted. MtrxCal then connects each sequential pair of points with a line segment during the plotting. This knowledge suggests how we might plot other graphics objects. We are going to pick up the thread of this idea very shortly.

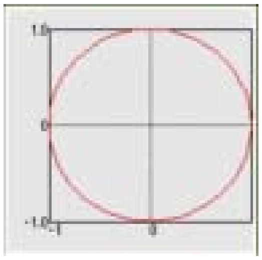


Figure 4

Let's manipulate our unit circle with matrices and plot the results so we can see how the matrices modify its values. We refer to each manipulation matrix as a transformation matrix **T**. In 2-space a transformation matrix **T** is a 2x2 matrix, and we transform our graphics object by multiplying its x-, y-coordinates with **T** in the form **M=T*b**, where **b** is the column array $\begin{bmatrix} x - coordinate \\ y - coordinate \end{bmatrix}$. On MtrxCal we shall add some lines and modify one so that we can manipulate our circle, thus:

```
theta=0:.1:2*pi;
z=(cos(theta)+j*sin(theta));
T=[1 0;0 1];b=[real(z);imag(z)];
A=T*b;
plot(A)
```

Note on line 3 our transformation matrix **T** and column array **b**. **T**= $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

A square matrix in which all entries on the diagonal = 1 is called an identity matrix. In **b**, we use the functions **real()** and **imag()** to put the x- and y-values from **z** into our column array. Finally, we **plot(A).**
Press **EXE**:

No change! Multiplying a value by the identity matrix yields that value, just as multiplying a value by identity over multiplication (otherwise known as 1) yields that value. But to work with other graphics objects, we want to understand what MtrxCal is doing, line by line. We already grasp what's going on on the first two lines. Is **b** really a column array? Let's replace the last line with **size(b)** and press **EXE**. **size(b)** returns **[2 63]**. Row one holds the x-coordinate values and row 2 holds the y-coordinate values. **b** is not a column vector though we can think of it as 63 column vectors; it's convenient if we do because that is how MtrxCal uses it. If we replace the last line with **size(A)** and press **EXE**. **size(A)** returns **[2 63]**. So MtrxCal takes each (x,y) pair in **b**, multiplies it with our matrix **T**, and outputs the results into another 2x63 matrix.

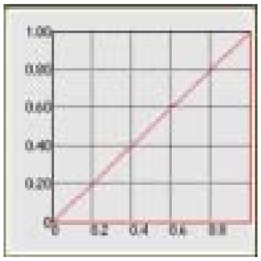


Figure 5

Back to plotting other graphics objects. If, for example, we want to plot a triangle, you might see how we can do so with a 2x4 matrix that holds our (x,y) pairs defining the vertices of the triangle (what? 4 vertices for a triangle? no. be patient. we'll explain ourselves in a moment.). This matrix will be **b** and should be thought of as 4 column arrays. Let's define the vertices of our triangle as (x,y) pairs (0,0),(1,1),(1,0). But remember: MtrxCal draws line segments between points while it plots those points. If we plotted from a 2x3 matrix using these pairs, we'd get a triangle that's missing its third leg. So we repeat the first pair at the end, and we get:
b=[0 1 1 0;0 1 0 0]; X-coordinates are in the first row, y-coordinates in the second one. Replace **b=[real(z);imag(z)];** with **b=[0 1 1 0;0 1 0 0];** and press **EXE**: (fig. 5)

Beautiful, but too symmetric for our purposes. Let's use a scalene triangle, and let's enter the (x,y) pairs in a manner that's easier for us to track (we want an easier way to enter (x,y) pairs in case we want to build other graphics objects later).

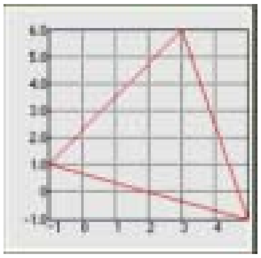


Figure 6

We'll work with (x,y) pairs (5,-1),(3 6),(-1 1). Let's enter each (x,y) pair on a row. Once we've built our matrix **b**, we'll transpose it so it'll be in the proper form: **b=[5 -1;3 6;-1 1;5 -1]; b=b';** The result is: (fig. 6)

Now we shall consider how changing values in the transformation matrix changes our circle and our triangle.

Contractions, Dilations

To dilate or contract the object, we use the transformation matrix $\begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix}$, where **r > 1** yields dilation and **0 < r < 1** yields contraction. Let's define **r** on the first line, and change our transformation matrix, thus:

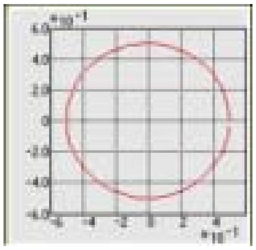


Figure 7

Circle

```
r=1;
theta=0:.1:2*pi;
z=(cos(theta)+j*sin(theta));
T=[r 0;0 r];
b=[real(z);imag(z)];
A=T*b;
plot(A)
```

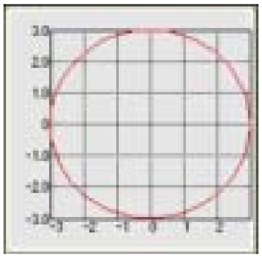


Figure 8

First we do a contraction, set **r=.5**, and press **EXE**: (Fig. 7)

Next we do a dilation, set **r=3**, and press **EXE**: (Fig. 8)

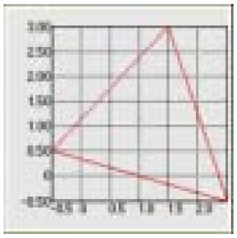


Figure 9

Triangle

```
r=1; T=[r 0;0 r];
b=[5 -1;3 6;-1 1;5 -1]; b=b';
A=T*b;
plot(A)
```

First we do a contraction, set **r=.5**, and press **EXE**: (Fig. 9)

Next we do a dilation, set **r=3**, and press **EXE**: (Fig. 10)

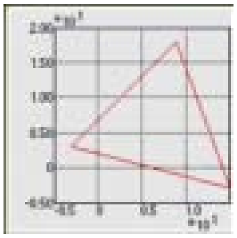


Figure 10

Translation, displacement by amount (u,v):

To demonstrate reflection in the x-axis and reflection in the y-axis, we'll look now at how to translate our circle. The operation is an array addition, $\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} u \\ v \end{bmatrix}$. We'll remove our transformation matrix, and add x-displacement **u** and y-displacement **v** and give them values 3,5, thus:

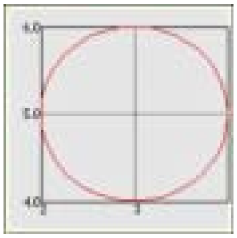


Figure 11

Circle

```
u=3; v=5;
theta=0:.1:2*pi;
z=(cos(theta)+j*sin(theta));
b=[real(z)+u;imag(z)+v];
plot(b)
```

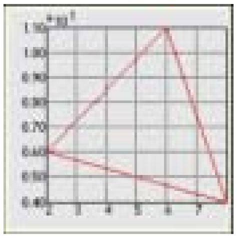


Figure 12

Triangle

```
u=3; v=5;
b=[5+u -1+v;3+u 6+v;-1+u 1+v;5+u -1+v]; b=b';
plot(b)
```

Reflection in x- & y-axis

To reflect our circle in the x-axis, we use the transformation matrix $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, while to reflect our circle in the y-axis, we use the transformation matrix $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$.

Now to MtrxCal. We add our transformation matrix **T** to reflect in the x-axis, multiply it by **b**, and plot the result:

Circle

```
theta=0:.1:2*pi; u=3;v=5;
z=(cos(theta)+j*sin(theta));
b=[real(z)+u;imag(z)+v];
T=[1 0;0 -1];
A=T*b;
plot(A)
```

Translated

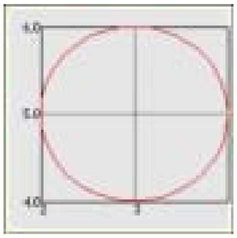


Figure 13

Reflected on x-axis

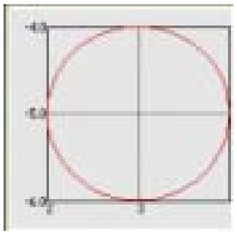


Figure 14

Reflected on y-axis

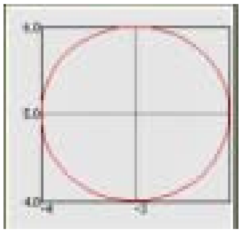


Figure 15

Triangle

```
u=3;v=5;
b=[5+u -1+v;3+u 6+v;-1+u 1+v;5+u -1+v]; b=b';
T=[1 0;0 -1];
A=T*b;
plot(A)
```

Translated

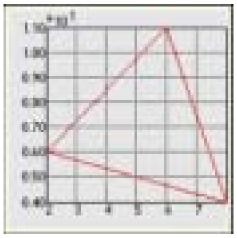


Figure 16

Reflected on x-axis

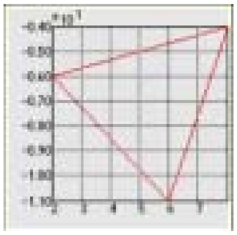


Figure 17

Reflected on y-axis

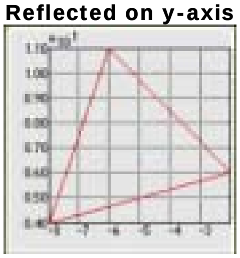


Figure 18

Transformation matrix T

So far our investigations concerning the transformation matrix have been confined to the diagonal. Now we ask what happens when we change the other values. Let's construct transformation matrix **T**:

$\begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}$ and see what it does to our circle and triangle:

Circle

```
theta=0:.1:2*pi;
z=(cos(theta)+j*sin(theta));
b=[real(z);imag(z)];
T=[1 .5;0 1];
A=T*b;
plot(A)
```

Triangle

```
T=[1 .5;0 1];
b=[5 -1;3 6;-1 1;5 -1]; b=b';
A=T*b;
plot(A)
```

Before

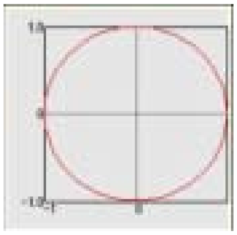


Figure 19

After

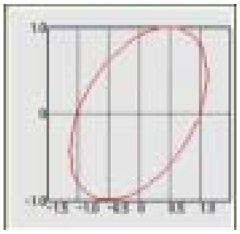


Figure 20

Before

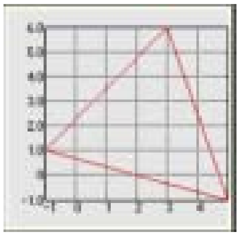


Figure 21

After

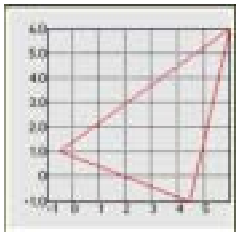


Figure 22

This transformation is called a shear along the x-axis of ½ with respect to the y-coordinate. That is, (x,y) after the transformation becomes
(x+½y, y).

Now let's construct transformation matrix **T**: $\begin{bmatrix}1 & 0\\ 0.5 & 1\end{bmatrix}$ and see what it does to our circle and triangle:

Circle

```
theta=0:.1:2*pi;
z=(cos(theta)+j*sin(theta));
b=[real(z);imag(z)];
T=[1 0;0.5 1];
A=T*b;
plot(A)
```

Triangle

```
T=[1 0;0.5 1];
b=[5 -1;3 6;-1 1;5 -1]; b=b';
A=T*b;
plot(A)
```

Before

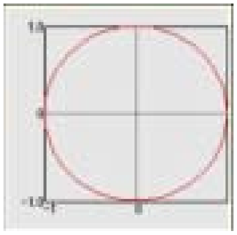


Figure 23

After

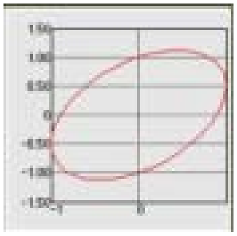


Figure 24

Before

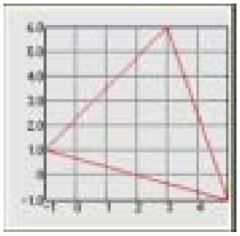


Figure 25

After

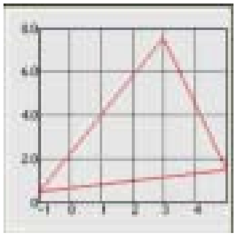


Figure 26

This transformation is called a shear along the y-axis of ½ with respect to the x-coordinate. That is, (x,y) after the transformation becomes (x,½x+y).

Rotation:

To rotate our objects, we use transformation matrix **T**:

$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$, where θ = rotation angle. To rotate our triangle by some radian measure **a**, we set up our matrix **T**:

```
Triangle
a=1; T=[cos(a) –sin(a);sin(a) cos(a)];
b=[5 -1;3 6;-1 1;5 -1]; b=b';
A=T*b;
plot(A)
```

Before

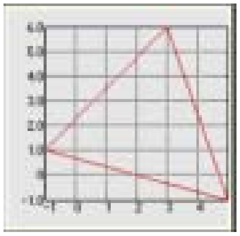


Figure 27

After

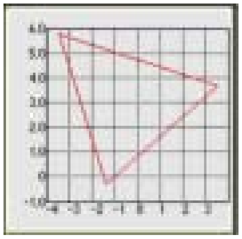


Figure 28

Physics

Systems of Linear Equations: Dimensional Analysis

The three basic dimensions of measurement, mass M, length L, and time T, are sufficient to describe the mechanical attributes of an object or system of objects.[3] In dimensional analysis, we reduce physical properties to their basic dimensions to find the interrelationships between them. We shall use MtrxCal to quickly solve problems of dimensional analysis.

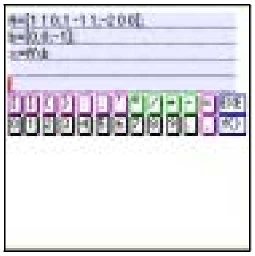
Example 1.[◊] Mersenne's Law relates frequency of vibration to length, tension, and density of string. The relationship can be expressed as:
frequency = $(\text{tension})^a(\text{line density})^b(\text{length})^g$ x constant
where a,b, g are exponents we need to find (that is, their values express the interrelationships between the physical properties in this formula).

Solution. (Requires: □ matrix) We need to set the problem up in a form that we can use on MtrxCal. First, in terms of MLT, the equation
frequency = $(\text{tension})^a(\text{line density})^b(\text{length})^g$ becomes: $T^{-1} = (MLT^{-2})^a(ML^{-1})^b(L^1)^g$, where
time = T
frequency = T^{-1}
tension= MLT^{-2}
line density= ML^{-1}
length= L^1

[*] Not all physical attributes can be rendered in terms of MLT alone, however. While charge (as in the Coulomb) can be expressed in MLT dimensions (for charge, $M^{1/2}L^{3/2}T^{-1}$), others, like temperature and luminosity, cannot. When such attributes are necessary to a dimensional analysis, they become dimensions in that analysis.

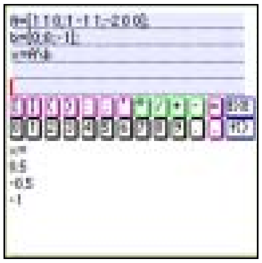
[◊] From Fletcher, T.J., *Linear Algebra: Through Its Applications*, Van Nostrand Reinhold, 1972. p.23-24.

Figure 29



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

Figure 30



```
A=[1 1 0;1 -1 1;-2 0 0];
b=[0;0;-1];
x=A\b
x=
0.5
-0.5
-1
```

Figure 31

The constant is dimensionless and so need not be considered. To be perfectly explicit, we'll rewrite this equation so that each property is expressed in terms of M, L, and T (where one is not a dimension of a property, we shall use the identity $x^0=1$ for all x). Thus,

$M^0L^0T^{-1}=(M^1L^1T^{-2})^a(M^1L^{-1}T^0)^b(M^0L^1T^0)^g$

Now we consider only the exponents:
$(0\ 0\ -1)=(1\ 1\ -2)^a\ (1\ -1\ 0)^b\ (0\ 1\ 0)^g$

And we set this up as a system of linear equations in the form **x**=**Ab**:

$$\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -2 \\ 1 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$$

Now we solve the problem on MtrxCal:

1. Turn on MtrxCal and go to the matrix keyboard layout (if you don't already see the layout show in fig. 29, press **menu**, **keys**, **matrix**)

2. Enter the problem. MtrxCal uses square brackets to enclose matrices and arrays, and semicolons within matrices to separate rows. On the top line, write **A=[1 1 0;1 -1 1;-2 0 0];** On the second row, write **b=[0;0;-1];** On the third row, write **x=A\b**

3. Press **EXE**: (fig. 31)

We end each line whose result we don't want to see with a semicolon. Because the third line doesn't end with a semicolon, its result, which is what we want, will appear.

x is a column vector that we read from top-to-bottom, so that a=0.5, b=-0.5, and g=-1. Substituting into the first equation, we have frequency = $(\text{tension})^{1/2}(\text{line density})^{-1/2}(\text{length})^{-1}$ x constant, which solves the problem.

Vector Arithmetic: System of Forces

Mechanics concerns itself with the effect of forces upon a body or system of bodies. When the forces acting upon a body or bodies are balanced, the body or bodies either do not move, or move with constant velocity (Newton's First Law of Motion). When unbalanced, the forces accelerate the body or bodies in rotation, in translation, or both.

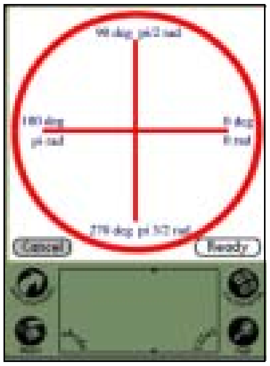


Figure 32

An analysis of the system of forces that act upon the body or bodies often begins with a diagram of the system. To simplify the analysis, the body or bodies are considered a point, the origin of a coordinate system whose axes are dimensions in space and whose units of measurement are units of force. Vectors that represent the forces acting upon the body or bodies are drawn; the direction of a vector is the direction of the force, and the length is the magnitude. Once the analysis has taken into account all the forces under consideration (many simple problems, for example, neglect the rotational forces of our non-inertial Earth-frame), one simply adds up all the forces to get a resultant force.

In linear algebra, we render vectors as arrays; the computational part of the analysis becomes vector addition. We can use MtrxCal to quickly do our calculations.

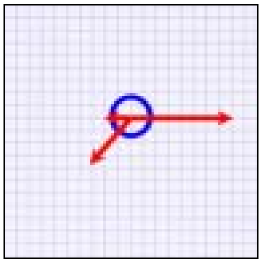


Figure 33

Example: A puck used in an ice hockey game has a mass of 170 grams. In an instant, three players strike the puck. The forces with which they strike have magnitude (in Newtons) and direction (in radians) of (10, pi), (25, 4), and (40, 2*pi) through the puck's center-of-mass. Consider only translation of the puck. If the forces are unbalanced, give the puck's acceleration.

Setup (pencil and paper work): Diagram the system.* To simplify understanding the problem, we place the puck's center-of-mass at the origin of a 2-dimensional coordinate system in which a unit equals 5 Newtons. We shall graph the problem in MtrxCal's angular coordinate system, illustrated in figure 5, where we place the tail of each vector at the origin.

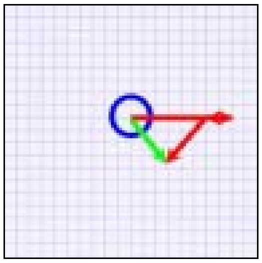


Figure 34

The problem with such a diagram is that it doesn't give us a clear sense of what the resultant vector is (though it does show us the forces are unbalanced). A clear graphical sense of the direction and magnitude of the resultant vector gives us confidence in whether the numerical values for direction and magnitude MtrxCal gives us is correct. (If incorrect, we can go over how we set the problem up to see if we've made some error.)

A more instructive diagram results from drawing the vectors tip-to-tail, then drawing the resultant vector from origin to tip of last vector as shown in figure 6.

Eyeballing the diagram, we expect a magnitude of ~25 N and direction of ~5.5 radians.

* Problem-solving is often a matter of making choices. Given the knowns and what is to be solved, as we have here, the challenge is to find the steps that take us from the one to the other; ideally, in the fewest number of steps possible. One could argue that there are many ways to diagram the system described in the problem; a few of those would be useful in understanding the problem, thereby serving as a step towards what is to be solved. Ideally, we want the one that will aid us in setting up the solution in the fewest number of steps.

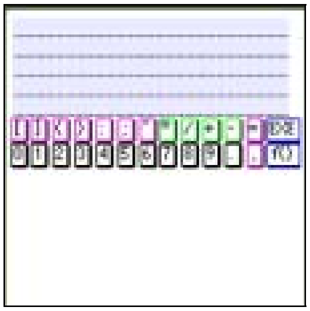


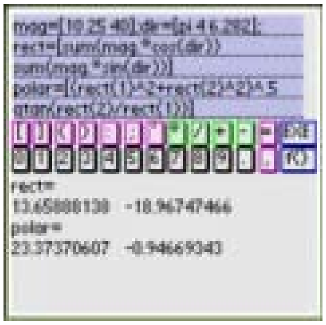
Figure 35

Work on MtrxCal:

1. Turn on MtrxCal and go to the matrix keyboard layout (if you don’t already see the layout below, press **menu**, **keys**, **matrix**)

2. Enter the problem. MtrxCal uses square brackets to enclose matrices and arrays, and semicolons within matrices to separate rows. On the top line, write **mag=[10 25 40]; dir=[pi 4 2*pi];** Magnitude and direction are polar coordinates in this system. To add the vectors, we need to convert them to rectangular coordinates, then sum all the x-values and sum all the y-values. On the second row, write **rect=[sum(mag.*cos(dir)) sum(mag.*sin(dir))]** We won’t end the line with a semicolon so that the result, which is the resultant vector in x- and y-values, will be displayed. To get the puck’s acceleration, we need the resultant vector’s magnitude, which means converting our answer to polar coordinates. On the third row, write **polar=[(rect(1)^2+rect(2)^2)^.5 atan(rect(2)/rect(1))]**

3. Press **EXE**: (fig. 36)



```
mag=[10 25 40];dir=[pi 4 6.282];
rect=[sum(mag.*cos(dir))
sum(mag.*sin(dir))]
polar=[(rect(1)^2+rect(2)^2)^.5
atan(rect(2)/rect(1))]
rect=
13.65888138   -18.96747466
polar=
23.37370607   -0.94669343
```

Figure 36

So the magnitude is 23 N. Adding the negative radian value to 6.28 (2 pi) gives 5.3. Both values are close to our eyeball estimates, giving us confidence in our answer. But we aren’t finished. The problem statement asks for the puck’s acceleration. We rearrange **F**=m**a** to **F**/m=**a**.
23.4 N/.170 Kg =137 m/s^2, which solves the problem.

Chemistry

Vector Arithmetic: Balancing Chemical Equations

The section discusses considering a chemical equation as a system of vectors.[*] Our intent is to show you how to use MtrxCal to double-check work done by hand. This section and the next, in which we consider balancing equations through systems of linear equations, form a unit.

Conceptually, the constituents on either side of a chemical equation can be thought of as vectors. Rendered as such, these constituents (molecules, most of the time) can be manipulated with vector arithmetic. To do this, we must analyze the constituents further, into their component parts, which are called species.
Let's look at a simple equation to see what I'm talking about: $NH_3 + O_2 = NO + H_2O$. This equation has left-hand side, $NH_3 + O_2$, and a right-hand side, $NO + H_2O$. The left-hand side (LHS) has two constituents, NH_3 and O_2; the right-hand side (RHS) also has two constituents, NO and H_2O. Each constituent has a scalar, a number that it is multiplied by in the balanced equation; in fact, a chemical equation is considered balanced when the scalars are found. Scalars need not be cardinal (counting numbers, 1, 2, 3, etc) although by convention balanced equations are represented with the smallest possible cardinals as scalars. To be explicit, we rewrite the equation as $\alpha NH_3 + \beta O_2 = \gamma NO + \delta H_2O$, with Greek letters as scalar variables. Finally, we note that the equation has three species: N, H, and O.
The last paragraph illustrates by example all the vocabulary we need when talking about how to transform a chemical equation into a system of vectors. Now let's work an example.

Example: Balance $NH_3 + O_2 \longrightarrow NO + H_2O$.

Use MtrxCal to double-check work done by hand.

Setup (pencil and paper work): Rewrite $NH_3 + O_2 \rightarrow NO + H_2O$ as $\alpha NH_3 + \beta O_2 = \gamma NO + \delta H_2O$, making the implied scalar variables explicit so that you can track them. Make up a table listing species at the head of columns and constituents in the leftmost entry of rows. In each cell for a constituent write how many of the species in the column head are present.

The entry for NH_3 will look like this:

	N	H	O
NH_3	1	3	0

The table will look like this:

	N	H	O
NH_3	1	3	0
O_2	0	0	2
NO	1	0	1
H_2O	0	2	1

[*] Based on a suggestion in Fletcher, T.J., *Linear Algebra: Through Its Applications*, Van Nostrand Reinhold, 1972. p.26

The numerical entries for each constituent make up its vector. The vector representing NH_3 is [1 3 0], the vector representing O_2 is [0 0 2], the vector representing NO is [1 0 1], and H_2O is [0 2 1]. We prefix each vector with its scalar, rewriting the chemical equation as $\alpha[1\ 3\ 0] + \beta[0\ 0\ 2] = \gamma\,[1\ 0\ 1]\ + \delta\,[0\ 2\ 1]$.. This equation is considered balanced when the sum of constituents on the left-hand side equals the sum of constituents on the right-hand side.

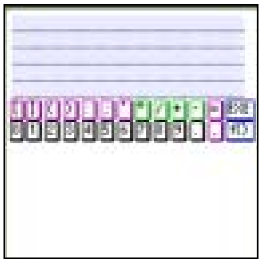
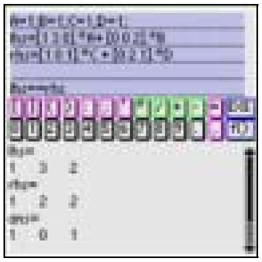


Figure 37

Work on MtrxCal:

1. Turn on MtrxCal and go to the matrix keyboard layout (if you don't already see the layout shown in fig. 37, press **menu**, **keys**, **matrix**)

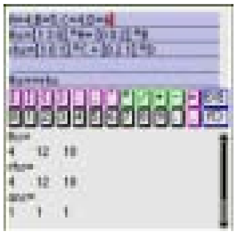


```
A=1;B=1;C=1;D=1;
lhs=[1 3 0].*A+[0 0 2].*B
rhs=[1 0 1].*C+[0 2 1].*D

lhs==rhs
lhs=
1   3   2
rhs=
1   2   2
ans=
1   0   1
```

Figure 38

2. Enter the problem. MtrxCal uses square brackets to enclose matrices and arrays, and semicolons within matrices to separate rows Because Greek letters aren't available, we'll use capital letters A, B, C, and D as our scalar variables, which we enter on the first line and set to 1: **A=1;B=1;C=1;D=1;**
   On the second line we write out the left-hand side of the equation: **lhs=[1 3 0].*A + [0 0 2].*B**
   On the third line we write out the right-hand side of the equation: **rhs=[1 0 1].*C+[0 2 1].*D**
   On the last line we write a test for equivalence: **lhs==rhs**



```
A=4;B=5;C=4;D=6
lhs=[1 3 0].*A+[0 0 2].*B
rhs=[1 0 1].*C+[0 2 1].*D

lhs==rhs
lhs=
4   12   10
rhs=
4   12   10
ans=
1   1   1
```

Figure 39

3. Press **EXE**: (fig. 39)

Our test for equivalence shows that the sum of species 1 and 3 are the same on the left- and right-hand sides. Solving by hand,* we arrive at **4**NH_3 + **5**O_2 = **4**NO + **6**H_2O. Entering these scalar values on the first line, we get **ans=1 1 1**, meaning that we've arrived as scalar values that do balance the equation:

Example: Balance $KMnO_4$ + HCl ⟶ KCl + $MnCl_2$ + Cl_2 + H_2O.

Use MtrxCal to double-check work done by hand.

Setup (pencil and paper work):
Rewrite $KMnO_4 + HCl \rightarrow KCl + MnCl_2 + Cl_2 + H_2O$ as $\alpha\ KMnO_4 + \beta\ HCl = \gamma\ KCl + \delta\ MnCl_2 + \varepsilon Cl_2 + \zeta H_2O$, making the implied scalar variables explicit so that you can track them. Make up a table listing species at the head of columns and constituents in the leftmost entry of rows. In each cell for a constituent write how many of the species in the column head are present.

The entry for NH_3 will look like this:

	K	Mn	O	H	Cl
$KMnO_4$	1	1	4	0	0

The table will look like this:

	K	Mn	O	H	Cl
$KMnO_4$	1	1	4	0	0
HCl	0	0	0	1	1
KCl	1	0	0	0	1
$MnCl_2$	0	1	0	0	2
Cl_2	0	0	0	0	2
H_2O	0	0	1	2	0

[*] The problems in the first two examples can be set up as systems of linear equations, which means MtrxCal can solve them. Please check the first example of Chemistry: Systems of Linear Equations: Chemical Stoichiometry to see how to do this.

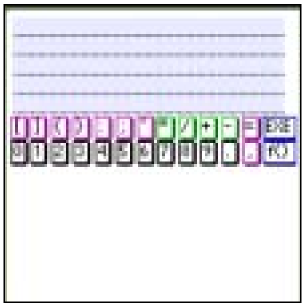


Figure 40

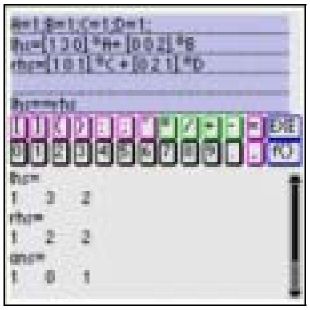
The numerical entries for each constituent make up its vector. We prefix each vector with its scalar, rewriting the chemical equation as α [1 1 4 0 0] + β [0 0 0 1 1] = γ [1 0 0 0 1] + δ [0 1 0 0 2] + ε [0 0 0 0 2] + ζ [0 0 1 2 0]. This equation is considered balanced when the sum of constituents on the left-hand side equals the sum of constituents on the right-hand side.

Work on MtrxCal:

1. Turn on MtrxCal and go to the matrix keyboard layout (if you don't already see the layout shown in fig.40, press **menu**, **keys**, **matrix**)

2. Enter the problem. MtrxCal uses square brackets to enclose matrices and arrays, and semicolons within matrices to separate rows Because Greek letters aren't available, we'll use capital letters A, B, C, and D as our scalar variables, which we enter on the first line and set to 1: **A=1;B=1;C=1;D=1;E=1;F=1**
   On the second line we write out the left-hand side of the equation: **lhs=[1 3 0].*A + [0 0 2].*B**
   On the third line we write out the right-hand side of the equation: **rhs=[1 0 1].*C+[0 2 1].*D**
   On the last line we write a test for equivalence: **lhs==rhs**

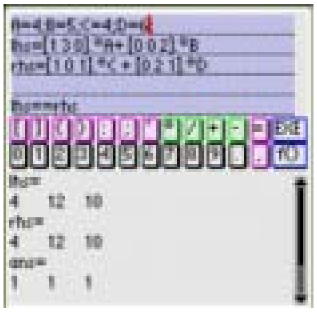
3. Press **EXE**: (fig. 42)



```
A=1;B=1;C=1;D=1;
lhs=[1 3 0].*A+[0 0 2].*B
rhs=[1 0 1].*C+[0 2 1].*D
lhs==rhs
```

```
lhs=
1   3   2
rhs=
1   2   2
ans=
1   0   1
```

Figure 41



```
A=4;B=5;C=4;D=6
lhs=[1 3 0].*A+[0 0 2].*B
rhs=[1 0 1].*C+[0 2 1].*D
lhs==rhs
```

```
lhs=
4   12   10
rhs=
4   12   10
ans=
1   1   1
```

Figure 42

Our test for equivalence shows that the sum of species 1 and 3 are the same on the left- and right-hand sides.
Solving by hand,[*] we arrive at **4**NH_3 + **5**O_2 = **4**NO + **6**H_2O. Entering these scalar values on the first line, we get **ans=1 1 1**, meaning that we've arrived as scalar values that do balance the equation:

Systems of Linear Equations: Chemical Stoichiometry[∇]

Stoichiometry (from Greek, measure of elements) deals with the balancing of elements or molecules (called species) and charges in reactions in a closed system (being in a closed system, the net mass and charge are preserved in the reaction: nothing gained, nothing lost). MtrxCal can be used to balance complex chemical equations though only for a subset of such equations. Specifically, the requirements are:

[*] This particular problem can be set up as a system of linear equations, which means MtrxCal can solve it. Please check the first example of Chemistry: Systems of Linear Equations: Chemical Stoichiometry to see how to do this.

[∇] The problems and methods presented in this section come from a series of articles in the *The Journal of Chemical Education*:
Kolb,Doris: "Balancing Complex Redox Equations by Inspection", J. Chem. Educ. **1981**, 58, 642.
Kennedy, John H.: "Balancing Chemical Equations with a Calculator", J. Chem. Educ. **1982**, 59, 523.
Alberty, Robert A.: "Balancing Complex Chemical Equations Using a Hand-held Calculator", J. Chem. Educ. **1983**,60,102.

The Journal of Chemical Education is online at http://jchemed.chem.wisc.edu. Unfortunately, the articles cited are available in print only.
The interested reader might also want to navigate a browser to http://www.chemical-stoichiometry.net.

1. We need a square matrix, which is required for the inversion. This means the number of species equals the number of linear equations in the set up.

2. Too many or too few species. Sometimes it is possible to make the number of species equal the number of linear equations if we consider a combination of atoms that stay a combination on either side of the reaction as one species, instead of considering the atoms as separate species. For example, if the hydroxyl radical OH^- remains throughout the reaction, the radical can be considered a single species.

3. Two elements in constant ratio complicates the problem. If, for example, N and O occur in five species only as NO_3, the five equations would not be linearly independent, and we could calculate the numbers of moles of only four species. The system effectively contains four "elements."

4. Ions. If ions are involved, electrical charge has to be conserved through an additional linear equation.

OK, now that we know what the requirements are, let's look at how to use MtrxCal to do stoichiometry.

Example: Balance $NH_3 + O_2 \longrightarrow NO + H_2O$.

Setup (pencil and paper work): Rewrite $NH_3 + O_2 \rightarrow NO + H_2O$ as $\alpha NH_3 + \beta O_2 = \gamma NO + \delta H_2O$, making the implied scalar variables explicit so that you can track them. Make up a table listing constituents at the head of columns and species in the leftmost entry of rows. In each cell for a constituent write how many of the species in the leftmost entry of a row are present.

The entry for N will look like this:

	NH_3	O_2	NO	H_2O
N	1	0	1	0

The table will look like this:

	NH_3	O_2	NO	H_2O
N	1	0	1	0
H	3	0	0	2
O	0	2	1	1

The idea here is to set up $\alpha NH_3 + \beta O_2 = \gamma NO + \delta H_2O$ as a system of linear equations that we can solve in the form of **b**=**Mx**. To create our square matrix, we rearrange the terms, thus: $\alpha NH_3 + \beta O_2 - \gamma NO = \delta H_2O$, then substitute column vectors for the constituents they represent,

$$\alpha * \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix} + \beta * \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} - \gamma * \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \delta * \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

Next, we do 3 steps at one time: first, we redistribute the negative sign in the γ entry so that the γ is positive while the entries in the column matrix are negative, we build our square matrix, set δ=1 and we put the other scalar variables in their own column matrix, multiplied by the square matrix. Given that δ=1, the following equation is equivalent to the preceding one:

$$\begin{bmatrix} 1 & 0 & -1 \\ 3 & 0 & 0 \\ 0 & 2 & -1 \end{bmatrix} * \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

The problem now can be solved using MtrxCal's **x**=**M****b** function

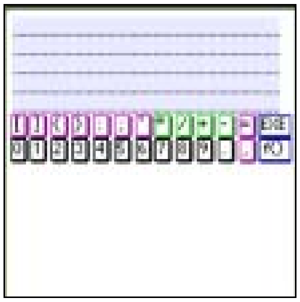


Figure 43

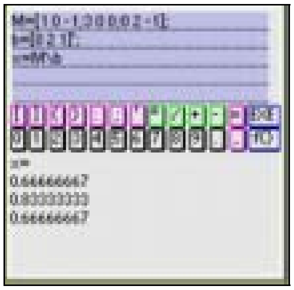
Work on MtrxCal:

1. Turn on MtrxCal and go to the matrix keyboard layout (if you don't already see the layout shown in fig.43, press **menu**, **keys**, **matrix**)

2. Enter the problem. MtrxCal uses square brackets to enclose matrices and arrays, and semicolons within matrices to separate rows. One line at a time, we enter:
   M=[1 0 -1;3 0 0;0 2 -1];
   b=[0 2 1]';
   x=M\b

3. Press **EXE**: (fig. 44)

We read off the column vector as α=0.67, β=0.83, and γ=0.67 (remember, δ=1). If our problem asked for us to render the scalar values in moles (M), the fractional values would suffice; and we'd be finished. However, by convention balanced equations are represented with the smallest possible cardinals as scalars. Multiplying each by 6 yields α=4, β=5, γ=4, and δ=6. The balanced equation is: 4 NH_3 + 5 O_2 = 4 NO + 6 H_2O.



```
M=[1 0 -1;3 0 0;0 2 -1];
b=[0 2 1]';
x=M\b
x=
0.66666667
0.83333333
0.66666667
```

Figure 44

Example: Balance $KMnO_4$ + HCl ⟶ KCl + $MnCl_2$ + Cl_2 + H_2O.

Setup (pencil and paper work): Rewrite $KMnO_4 + HCl \rightarrow KCl + MnCl_2 + Cl_2 + H_2O$ as $\alpha KMnO_4 + \beta HCl = \gamma KCl + \delta MnCl_2 + \varepsilon Cl_2 + \zeta H_2O$, making the implied scalar variables explicit so that you can track them. Make up a table listing constituents at the head of columns and species in the leftmost entry of rows. In each cell for a constituent write how many of the species in the leftmost entry of a row are present.

The entry for N will look like this:

	$KMnO_4$	HCl	KCl	$MnCl_2$	Cl_2	H_2O
K	1	0	1	0	0	0

The table will look like this:

	$KMnO_4$	HCl	KCl	$MnCl_2$	Cl_2	H_2O
K	1	0	1	0	0	0
Mn	1	0	0	1	0	0
O	4	0	0	0	0	1
H	0	1	0	0	0	2
Cl	0	1	1	2	2	0

The idea here is to set up $\alpha KMnO_4 + \beta HCl = \gamma KCl + \delta MnCl_2 + \varepsilon Cl_2 + \zeta H_2O$ as a system of linear equations that we can solve in the form of **b**=**Mx**. To create our square matrix, we rearrange the terms, thus: $\alpha KMnO_4 + \beta HCl - \gamma KCl - \delta MnCl_2 - \varepsilon Cl_2 = \zeta H_2O$, then substitute column vectors for the constituents they represent,

$$\alpha * \begin{bmatrix} 1 \\ 1 \\ 4 \\ 0 \\ 0 \end{bmatrix} + \beta * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} - \gamma * \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} - \delta * \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 2 \end{bmatrix} - \varepsilon * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 2 \end{bmatrix} = \zeta \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ 0 \end{bmatrix}$$

Next, we do 3 steps at one time: first, we redistribute the negative sign in the γKCl, $\delta MnCl_2$, and εCl_2 entries so that the γ, δ, and ε are positive while the entries in their column matrices are negative, we build our square matrix, set ζ=1 and we put the other scalar variables in their own column matrix, multiplied by the square matrix. Given that ζ=1, the following equation is equivalent to the preceding one:

$$\begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 4 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & -2 & -2 \end{bmatrix} * \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \varepsilon \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ 0 \end{bmatrix}$$

The problem now can be solved using MtrxCal's **x**=**M****b** function

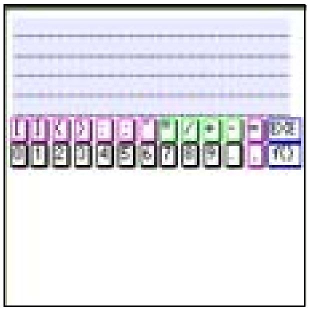


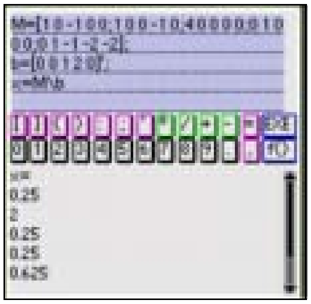
Figure 45

Work on MtrxCal:

1. Turn on MtrxCal and go to the matrix keyboard layout (if you don't already see the layout shown in fig.45, press **menu**, **keys**, **matrix**)

2. Enter the problem. MtrxCal uses square brackets to enclose matrices and arrays, and semicolons within matrices to separate rows. One line at a time, we enter:
   M=[1 0 -1 0 0;1 0 0 -1 0;4 0 0 0 0;0 1 0 0 0;0 1 -1 -2 -2];
   b=[0 0 1 2 0]';
   x=M\b

3. Press **EXE**: (fig. 46)



```
M=[1 0 -1 0 0;1 0 0 -1 0;4 0 0 0 0;0 1 0
0 0;0 1 -1 -2 -2];
b=[0 0 1 2 0]';
x=M\b
x=
0.25
2
0.25
0.25
0.625
```

Figure 46

We read off the column vector as α=0.25, β=2, and γ=0.25, δ=0.25, and ε=0.625 (remember, ζ=1). If our problem asked for us to render the scalar values in moles (M), the fractional values would suffice; and we'd be finished. However, by convention balanced equations are represented with the smallest possible cardinals as scalars. Multiplying each by 8 yields α=2, β=16, γ=2, δ=2, ε=5, and ζ=8. The balanced equation is:
$2\ KMnO_4 + 16\ HCl = 2\ KCl + 2\ MnCl_2 + 5\ Cl_2 + 8\ H_2O$.

An alternate method to balancing chemical equations

Our last example introduces an alternate method to balancing chemical equations through systems of linear equations.[◊]

In this method our setup includes establishing an algebraic relationship between the species and their scalars in the balanced equation. With this method we shall go one step further than in the earlier examples by balancing an ionic equation. We use the following rules as our guide:

<table>
<tr><th colspan="2">Ling's Rules for Balancing Redox Equations by Inspection</th></tr>
<tr><td><em>Step 1</em></td><td>Locate any elements that must have the same scalar in the balanced equation, those appearing only <em>once</em> on each side of the equation and in equal numbers on both sides. Mark these terms with arrows.</td></tr>
<tr><td><em>Step 2</em></td><td>Locate any elements that appear only <em>once</em> on each side of the equation but have <em>unequal</em> numbers of atoms. Balance these elements first.</td></tr>
<tr><td><em>Step 3</em></td><td>When steps 1 and 2 do not work, look for elements that must be present in a <em>constant ratio</em> on each side of the equation, elements appearing only <em>once</em> and in the <em>same compound</em> on one side or the other. Balance these elements so as to maintain this ratio on both sides of the equation.</td></tr>
<tr><td colspan="2">After applying the three steps above, if the equation still cannot be balanced, use the algebraic method to complete the balancing process. Assign letters to whichever scalars remain unknown and determine their values by simple algebraic equations.[*]</td></tr>
</table>

Example: Balance $Zn + NO_3^- + H^+ \longrightarrow Zn^{2+} + NH_4^+ + H_2O$.

Setup (pencil and paper work): Rewrite $Zn + NO_3^- + H^+ = Zn^{2+} + NH_4^+ + H_2O$ as $\alpha Zn + \beta\ NO_3^- + \gamma\ H^+ = \delta\ Zn^{2+} + \varepsilon\ NH_4^+ + \zeta H_2O$, making the implied scalar variables explicit so that you can track them. Make up a table listing scalars at the head of columns and species in the leftmost entry of rows. The last row will be charge. Next to the species we write the algebraic expression that relates the scalars in the constituents in which the species appears. The algebraic expression shows the proportions required in the balanced equation. In the rightmost column we set up our column vector **b** in the form: first entry equals 1; all others equal 0.

[◊] This method comes from Kennedy, John H.: "Balancing Chemical Equations with a Calculator", J. Chem. Educ. **1982**, 59, 523.

[*] Kolb,Doris. "Balancing Complex Redox Equations by Inspection", *J. Chem. Educ.* **1981**, 58, 643.

Now we work out the algebraic relationships between the scalars:
Zn:$\alpha = \delta$, N:$\beta = \varepsilon$, O:$3\beta = \zeta$; H: $\gamma = 4\,\varepsilon + 2\,\zeta$; and charge: $-\beta + \gamma = 2\,\delta + \varepsilon$.

In setting up our square matrix, we rearrange the terms, thus: $\alpha Zn + \beta\, NO_3^- + \gamma\, H^+ - \delta\, Zn^{2+} - \varepsilon\, NH_4^+ - \zeta H_2O$. We'll mark our negative species in our table, to remind us that the species entries in their columns take negative values, and we change signs for charge entries in these columns:

We assume $\alpha = 1$.
The first two entries will look like this:

	eqn	α	β	γ	$-\delta$	$-\varepsilon$	$-\zeta$	**b**
	$\alpha = 1$	1	0	0	0	0	0	1
Zn	$\alpha = \delta$	1	0	0	1	0	0	0

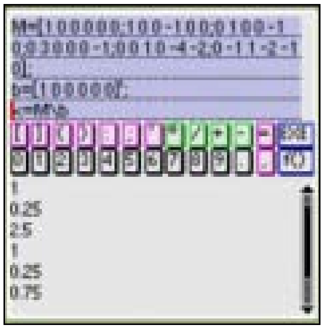
Remember, our rightmost column is the column vector **b**; and following our rules, its first entry is 1. The final table will look like this:

	eqn	α	β	γ	$-\delta$	$-\varepsilon$	$-\zeta$	**b**
	$\alpha = 1$	1	0	0	0	0	0	1
Zn	$\alpha = \delta$	1	0	0	-1	0	0	0
N	$\beta = \varepsilon$	0	1	0	0	-1	0	0
O	$3\beta = \zeta$	0	3	0	0	0	-1	0
H	$\gamma = 4\varepsilon + 2\zeta$	0	0	1	0	4	-2	0
charge	$\gamma - \beta = 2\delta + \varepsilon$	0	-1	1	-2	-1	0	0

We use our table to set up our linear algebra expression:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 3 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -4 & -2 \\ 0 & -1 & 1 & -2 & -1 & 0 \end{bmatrix} * \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \varepsilon \\ \zeta \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The problem now can be solved using MtrxCal's **x=M\b** function.



```
M=[1 0 0 0 0 0;1 0 0 -1 0 0;0 1 0 0 -1
0;0 3 0 0 0 -1;0 0 1 0 -4 -2;0 -1 1 -2 -1
0];
b=[1 0 0 0 0 0]';
x=M\b
1
0.25
2.5
1
0.25
0.75
```

Figure 47

Enter the problem. MtrxCal uses square brackets to enclose matrices and arrays, and semicolons within matrices to separate rows. One line at a time, we enter:

```
M=[1 0 0 0 0 0;1 0 0 -1 0 0;0 1 0 0 -1 0 0;0 3 0 0 0 -1;0 0 1 0 -4 -2;0 -1 1 -2 -1 0];
b=[1 0 0 0 0 0]';
x=M\b
```

then press **EXE**:

We read off the column vector as α=1, β=0.25, and γ=2.5, δ=-1, ε=0.25, and ζ=0.75. If our problem asked for us to render the scalar values in moles (M), the fractional values would suffice; and we'd be finished. However, by convention balanced equations are represented with the smallest possible cardinals as scalars. Multiplying each by 4 yields α=4, β=1, γ=10, δ=4, ε=1, and ζ=3. The balanced equation is:
$4\ Zn + NO_3^- + 10\ H^+ = 4\ Zn^{2+} + NH_4^+ + 3\ H_2O$.

Experimental Sciences:

Orthogonal Projections in Inner Product Spaces

The accuracy of data obtained by experiment can be affected by measurement error. When experimental data should show a polynomial relationship y=f(x) between two variables, the experimenter must then find the curve that best fits the data. A common application of linear algebra to experimental sciences is using least squares to fit polynomials to bivariate data. Because our purpose here is more application, less theory, we start by talking briefly about the general case for least squares fit of a polynomial, then apply it to specific cases of a line and a quadratic. This we hope will lead the reader to understand how to extend the method to higher-order polynomials.

The General Case. Given experimental data points (x_1, y_1), (x_2, y_2), (x_3, y_3),… (x_n, y_n), we wish to fit a polynomial of degree m
$y= a_0+ a_1 x+… a_m x^m$
to these data points. We do so in matrix form with **y**=M**v**, where

$$y=\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}, M=\begin{bmatrix} x_1^0 & x_1^1 & \cdots & x_1^m \\ x_2^0 & x_2^1 & \cdots & x_2^m \\ \cdots & \cdots & \cdots & \cdots \\ x_n^0 & x_n^1 & \cdots & x_n^m \end{bmatrix}, v=\begin{bmatrix} a_0 \\ a_1 \\ \cdots \\ a_m \end{bmatrix}$$

and a_0, a_1, a_2,… a_m are the values we seek.

For practical purposes, we shall replace the values in column 1 of matrix M with ones, since for all *x*, x^0=1. Also, the number of columns we need in matrix M depends on the degree of the polynomial. For a line, which is of degree 1, we need two columns – the first column of *x*'s that are all of degree 0, and the second of *x*'s that are all of degree 1. For a quadratic, which is of degree 2, we need three columns – the first column of *x*'s that are all of degree 0, the second of *x*'s that are all of degree 1; and the third of *x*'s that are all of degree 2.

For curves in general, we simply extend our observation by analogy: the number of columns we need in matrix M equals the degree of the polynomial plus 1.

The equation we shall use on MtrxCal to get those values is **((M'*M)^-1)*M'*y**. It will return a unique solution provided that **M'*M** is invertible; that is, if and only if its determinant is not equal to zero.

Least Squares Fitting of Polynomials to Data

Example 1. Least Squares Fit of a Line: y= mx+b

Given sample points (1,1), (3,0), (4,2), (7,2), (9,4), and (13,3), find the line of best fit.

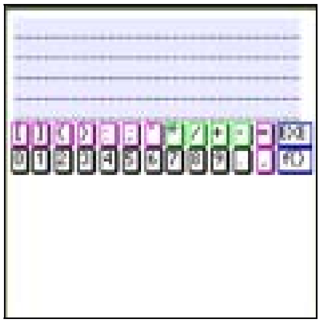
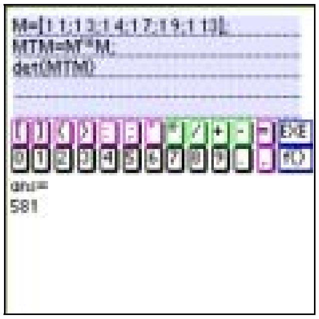
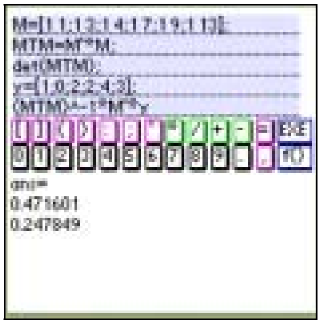


Figure 48



```
M=[1 1;1 3;1 4;1 7;1 9;1 13];
MTM=M'*M;
det(MTM)
ans=
581
```

Figure 49



```
M=[1 1;1 3;1 4;1 7;1 9;1 13];
MTM=M'*M;
det(MTM);
y=[1,0,2,2,4,3];
(MTM)^-1*M'*y
ans=
0.471601
0.247849
```

Figure 50

Set up (pencil and paper work): None.

Work on MtrxCal:

1. Turn on MtrxCal and go to the matrix keyboard layout (if you don't already see the layout shown in fig.48, press **menu**, **keys**, **matrix**)

2. Input matrix M, create the variable MTM and instantiate it with **M'*M**; get the determinant of MTM. We end each line whose result we don't want to see with a semicolon. Because the third line doesn't end with a semicolon, its result, which is what we want, will appear.
   Press **EXE**

3. Because the determinant of M^TM is not zero, a unique solution exists. Now enter **y; (MTM)^-1*M'*y**, and press **EXE**

The answer is a column vector that we read from top to bottom. The best fit line is y=.471601+.247849x. To graph this line we add three more lines x=[0:15]; y1= a(1)+a(2)*x and plot(x,y) As expected the first value is the y-value where the line intercepts the y-axis and the second value is the slope. Now we read our output as the line intercepting the y-axis at ~.5, or 1/2, and having a rise/run of ~.25, or ¼.

Example 2. Least Squares Fit of a Quadratic: $y=a+bx+cx^2$

Given sample points (.5,1), (2.5,3), (2,8), (4.5,13), and (5,23), find the line of best fit.

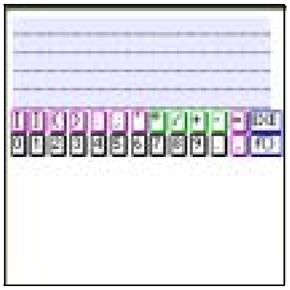
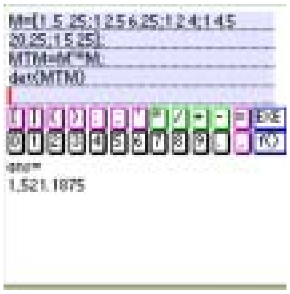


Figure 51

Set up (pencil and paper work): on paper, make 5 rows, one for each sample point, and divide the rows into 3 columns. Label column one "y", column two "x", and column three "x^2". In columns one and two, write your sample points' y and x values. In column three write the square for each x value. Your table should look like this:

Y	X	X^2
1	.5	.25
3	2.5	6.25
8	2	4
13	4.5	20.25
23	5	25



```
M=[1 .5 .25;1 2.5 6.25;1 2 4;1 4.5
20.25;1 5 25];
MTM=M'*M;
det(MTM)
ans=
1,521.1875
```

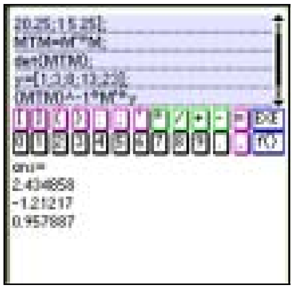
Figure 52

Work on MtrxCal:

1. Turn on MtrxCal and go to the matrix keyboard layout (if you don't already see the layout as shown in fig.51, press **menu**, **keys**, **matrix**)

2. Input matrix M, create the variable MTM and instantiate it with **M'*M**; get the determinant of MTM. We end each line whose result we don't want to see with a semicolon. Because the third line doesn't end with a semicolon, its result, which is what we want, will appear.

3. Because the determinant of M^TM is not zero, a unique solution exists. Now enter **y; (MTM)^-1*M'*y**, and press **EXE**



```
20.25;1 5 25];
MTM=M'*M;
det(MTM);
y=[1;3;8;13;23];
(MTM)^-1*M'*y
ans=
2.434858
-1.21217
0.957887
```

Figure 53

The answer is a column vector that we read from top to bottom. The best fit quadratic is $y=2.43-1.21x+.96x^2$. The phenomenon measured decides the meaning of the coefficients. Had our example measured distance displaced (y) against time (x) of a falling body, "a" would be the initial displacement at time 0, "b" would be the initial velocity at time 0; "c" would be ½ of the acceleration of gravity.

Graph Theory

Matrix Multiplication: Dominance Directed Graph

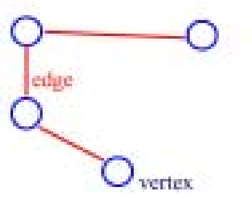


Figure 54

A graph, as used in graph theory, is a set of objects (called vertices) and the relationship among them (called edges). A standard representation of graphs shows vertices as circles and edges as lines connecting the circles.

In a directed graph, the direction by which one can move from vertex to vertex is defined; a picture representation uses arrows on the edges to show in which direction one can move along edges.

Put another way, the arrows show the direction of flow from one object to the next. It is possible for one edge to have "flow" in either direction. In a dominance directed graph, however, each pair of vertices is connected by an edge, and all edges are one-way (and a second edge connecting two vertices and having "flow" in the opposite direction is not permitted!).

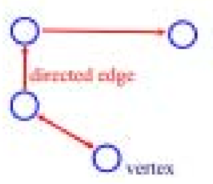


Figure 55

Graph theory, in which the properties of graphs are studied and defined by proofs and theorems, has proved to have applications in many fields, from network flow in vehicular traffic and telecommunications, to the study of finite automata, in which each vertex represents the state of a machine and each edge represents a transition from state to state. In our example we shall use graph theory to rank little league football teams after a season of play.*

Example: We shall consider the season of play as a tournament, with two requirements: each team has played each other team exactly once, and no game was allowed to end in a tie. The tournament can then be considered as a dominance directed graph, in which each team is a vertex, each game is an edge, and the arrow shows winner-loser with "flow" going from winner to loser. In a 10-team league, this means each vertex is connected to the other nine vertices, each connection a single edge, with an arrow on the edge going from winner to loser.

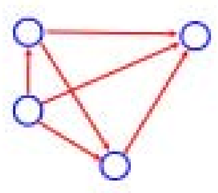


Figure 56

Set up (pencil and paper work): Instead of drawing this graph, let's set it up as a matrix. The teams in the league are:

Lions Ravens Bengals Eagles Jaguars Bears Vikings Rams Falcons Panthers

We shall create a table in which each team's win-loss record will fill a row; and keeping the order of the teams the same, we shall assign each column to the team played. The result should look like this:

* A much better treatment of graph theory in linear algebra can be found in Rorres, Chris & Anton, Howard, *Applications of Linear Algebra, 3rd Edition*, Wiley & Sons, 1984. Though out-of-print (and that's a shame: we strongly recommend it), used copies are available through amazon marketplace sellers at http://www.amazon.com.

	Lions	Ravens	Bengals	Eagles	Jaguars	Bears	Vikings	Rams	Falcons	Panthers
Lions	0									
Ravens		0								
Bengals			0							
Eagles				0						
Jaguars					0					
Bears						0				
Vikings							0			
Rams								0		
Falcons									0	
Panthers										0

We fill the diagonal with zeros. (We are making a vertex matrix. Vertex matrices have 2 properties: the diagonal entries are zero; all other entries are either 1 or 0.) In our example, this makes sense: a team does not play against itself (at least, not in tournament). Now, with the game results in front of us, we fill out each row. We enter 1 for a win, and 0 for a loss. The first row looks like this:

	Lions	Ravens	Bengals	Eagles	Jaguars	Bears	Vikings	Rams	Falcons	Panthers
Lions	0	1	0	0	1	0	1	0	0	1

This row reads as: the Lions won against the Ravens, Jaguars, Vikings, and Panthers; and lost against the Bengals, Eagles, Bears, Rams, and Falcons. The entire vertex matrix looks like this:

	Lions	Ravens	Bengals	Eagles	Jaguars	Bears	Vikings	Rams	Falcons	Panthers
Lions	0	1	0	0	1	0	1	0	0	1
Ravens	0	0	1	1	0	0	1	1	1	0
Bengals	1	0	0	1	1	0	1	0	1	1
Eagles	1	0	0	0	1	0	1	0	1	0
Jaguars	0	1	0	0	0	1	0	1	0	1
Bears	1	1	1	1	0	0	0	1	0	0
Vikings	0	0	0	0	1	1	0	1	1	1
Rams	1	0	1	1	0	0	0	0	0	1
Falcons	1	0	0	0	1	1	0	1	0	1
Panthers	0	1	0	1	0	1	0	0	0	0

Let's mark up this matrix to make a couple of observations. First, we run a red line down the diagonal. Second, we run blue lines through the entries perpendicular to the diagonal. (Only enough entries are blue-lined to make our observations). The result looks like the vertex matrix below:

	Lions	Ravens	Bengals	Eagles	Jaguars	Bears	Vikings	Rams	Falcons	Panther
Lions	0	1	0	0	1	0	1	0	0	1
Ravens	0	0	1	1	0	0	1	1	1	0
Bengals	1	0	0	1	1	0	1	0	1	1
Eagles	1	0	0	0	1	0	1	0	1	0
Jaguars	0	1	0	0	0	1	0	1	0	1
Bears	1	1	1	1	0	0	0	1	0	0
Vikings	0	0	0	0	1	1	0	1	1	1
Rams	1	0	1	1	0	0	0	0	0	1
Falcons	1	0	0	0	1	1	0	1	0	1
Panther	0	1	0	1	0	1	0	0	0	0

Our first observation is a reiteration: the diagonal entries are 0’s. Our second observation allows us to double-check our matrix entries: we call an entry in this matrix E_{ij}, and $i \neq j$, then if $E_{ij}=1$, $E_{ji}=0$, and if $E_{ij}=0$, $E_{ji}=1$. Explained in terms of our little league matrix, if the Ravens-Lions entry is 0, the Lions-Ravens entry is 1; if the Bears-Bengals entry is 1, the Bengals-Bears entry is 0; and so on.

Now that we’ve set up our matrix vertex, we’ll apply the definition of a power vertex to get league rankings. The definition says that if we do row sums on $A = M + M^2$, where M is our vertex graph, we’ll get the rankings: the greatest row sum will belong to our 1st place team, the 2nd greatest row sum to our 2nd place team, etc.

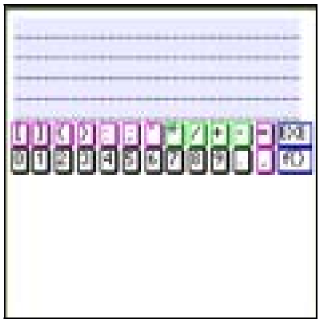


Figure 57

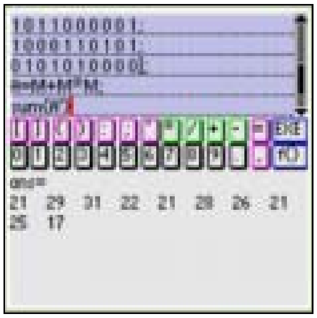
Work on MtrxCal:

1. Turn on MtrxCal and go to the matrix keyboard layout (if you don't already see the layout shown in fig.57, press **menu**, **keys**, **matrix**)

2. Enter the vertex matrix as **M=[0 1 0 0 1 0 1 0 0 1;0 0 1 1 0 0 1 1 1 0;1 0 0 1 1 0 1 0 1 1;1 0 0 0 1 0 1 0 1 0;0 1 0 0 0 1 0 1 0 1;1 1 1 1 0 0 0 1 0 0;0 0 0 0 1 1 0 1 1 1;1 0 1 1 0 0 0 0 0 1;1 0 0 0 1 1 0 1 0 1;0 1 0 1 0 1 0 0 0 0];**

   On the following lines enter:
   A=M+M*M;
   sum(A')
   We do a sum on the transpose of matrix **A** since sum() does column sums and we need row sums.

3. Press **EXE**:



```
1011000001;
1000110101;
0101010000];
a=M+M*M;
sum(A')
ans=
21  29  31  22  21  28  26  21
25  17
```

Figure 58

According to our results, the Bengals (6-4) place 1st, the Ravens (5-5) 2nd, the Bears (5-5) 3rd, the Vikings (5-5) 4th, the Falcons (5-5) 5th, the Eagles (4-6) 6th, while the Lions, Jaguars, and Rams (4-6) tie; and the Panthers come in last. So how are so many teams ending the season with 5-5 so clearly ranked? Remember, our math depends on a dominance directed graph: it all depends on who beats whom to set up the hierarchy.

The point here is that eyeballing the win-loss matrix, even doing quick-and-dirty sums of wins and losses, does not give reliable results

Table of contents