2D and 3D Graphical Routines for Teaching Linear Algebra

M. B. Monagan
Department of Mathematics, Simon Fraser University,
Burnaby, British Columbia, V5A 1S6 Canada.
E-mail: `monagan@cecm.sfu.ca`

Abstract

We present a collection of Maple graphics routines written primarilly for teaching Linear Algebra. The package includes routines for visualizing vectors, planes, linear systems, linear transformations, subspaces, eigenvectors, singular vectors, projections and least squares approximations in $\mathbb{R}^2$ and $\mathbb{R}^3$.

1 Introduction

Many authors have written on the use of computers to teach Linear Algebra. An early Maple reference is *Linear Algebra with Maple V* by Johnson [2]. This laboratory manual is intended to serve as a supplement to a regular text. Ideally, the material in it would be taught in a weekly computer laboratory. One focus in the manual is on using Maple commands to do the steps required to perform calculations in Linear Algebra. For example, to reduce a matrix to row Echelon form, students would use the Maple routines `addrow`, `swaprow`, `mulrow` from the `linalg` package to do elementary row operations on a matrix instead of doing it by hand.

A similar approach is taken by Nicholson where he supplements his text *Elementary Linear Algebra* [6] with a numerical calculator written specifically for doing Linear Algebra but made accessible over the internet. The publisher charges students an additional $20 Canadian for access to the website. Students may use the website to practice calculations, to check answers, and to solve larger problems not doable by hand.

In [3], Kneppers reports on his use of Maple in teaching Linear Algebra to systems engineers where he has given a weekly lab using Maple. Most of the Maple exercises he cites in his paper are of an exploratory flavor. For example, students are given some 3 by 3 matrices and asked to us Maple to compute their powers and figure out what is going on. A second example asks students to use Maple to work through several examples to check that a theorem holds before proving it.

A different approach is taken by the authors of the text *Linear Algebra: Modules for Interactive Learning Using Maple 6* [4] developed by the Linear Algebra Modules Project (LAMP). The text is a printout of a collection of Maple worksheets which are Maple tutorials with exercises. An accompanying CD contains an electronic version of the Maple worksheets and a library of support routines. Students can work through the tutorials on their own. The authors have placed a stronger focus on visualization. For example, the `movie` subroutine permits the student to visualize a linear transformation by animating what happens when it is applied to a given object. This invites a more experimental approach to the subject. The student can be asked to construct a linear transformation to accomplish a certain task. Motivated by the desire to see the animation work, the student will keep trying until it works correctly visually. It also makes, in this example, the computer graphics application obvious to the student.

There are obviously many good uses of the computer in the references cited. However, in all four cases, the instructor does not "teach" with the computer; rather the student uses the computer as an aid to practice calculations, do assignment problems, learn material on their own and do exploratory computations. This is not "teaching" Linear Algebra with the computer, because the instructor may not be using the computer at all. Rather, it is "learning" Linear Algebra with the computer.

Inspired by the visualization routines in the LAMP modules and ideas exchanged with participants of the Maple course [5] that the author has given in the summers of 1998–2001, we have attempted in this paper to provide a systematic and more complete coverage of graphical aids aimed at teaching this foundational subject.

Before starting, let us pose a question to the reader. Is the subject of Linear Algebra primarilly algebraic or geometric? Let me restate the question to get to the heart of the matter: Did the authors of the theorems of Linear Algebra find these theorems by thinking algebraically or by thinking geometrically? we suspect that in many cases they "saw" and "understood" the theorems geometrically but proved them algebraically. Yet when we look at texts in Linear Algebra we will not find a systematic attempt to present the subject geometrically. The subject is presented by definitions, examples and theorems which are to be understood algebraically. The texts we use at SFU, Fraleigh and Beauregard [1] and Nicholson [6], are typical in this regard. The development of geometric intuition is largely absent. This is partly because of time pressure. There isn't enough time to discover and see the theorem; there is only time to give one example, state the theorem, and maybe give a proof of the theorem. This will not lead the students to a deep understanding of the material for it is possible to understand each step in an algebraic proof yet not "see" why the theorem is true nor what it is saying.

It is our goal in this paper to present a set of Maple subroutines that would enable geometric intuition to be developed rapidly with the computer. These tools should be helpful to instructors in the lecture as well as to the student in the lab.

2 Visuals for Linear Algebra

In this section we present the visuals we have developed. The order of presentation follows a traditional presentation of Linear Algebra as in [1] where one covers

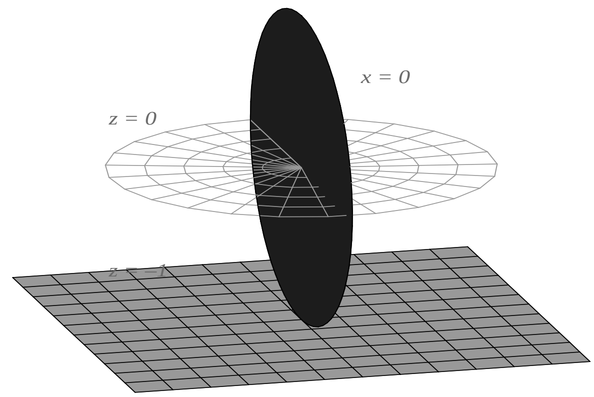
1. vectors, matrices, and linear systems,
2. bases, subspaces, and linear transformations,
3. determinants, eigenvalues and eigenvectors,
4. projection and orthogonality.

The visuals have been designed so that the relevant numerical information is also displayed in the graphic. For example, if the instructor wants to produce a handout or an overhead showing a visual of the eigenvectors of a matrix, we will want the matrix and the numerical values of the eigenvalues and eigenvectors there too.

2.1 Vectors and Planes

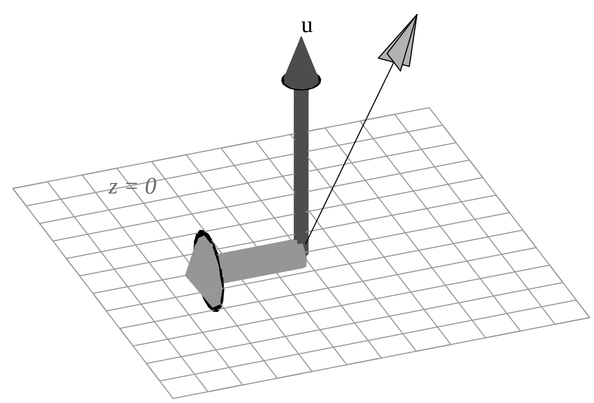
Most of our visuals contain one or more vectors and/or planes. Hence we describe these utility routines first. The `planeplot` routine will draw a plane in $\mathbb{R}^3$. The plane may be specified by giving (i) the equation of the plane, (ii) a basis for the plane and a point on the plane, or (iii) parametrically in the form $(x(s,t), y(s,t), z(s,t))$. By default, the plane is centered on the point on the lane which is closest to the origin and is labelled with its equation which is computed if not explicitly given. Several styles for drawing the plane are provided including a circular shape as well as the traditional rectangular shape.

```
> p1 := planeplot(z=-1):
> p2 := planeplot({<1,0,0>,<0,1,0>},
>    shape=circular,style=grid):
> p3 := planeplot(<0,s,t>,
>    shape=circular,color=blue):
> plots[display]({p1,p2,p3});
```



The `vectorplot3d` command will draw a solid vector in $\mathbb{R}^3$. Several styles of vectors are available - best described by looking at some pictures.

```
> p1 := planeplot(z=0,
>    planestyle=grid,length=2.5,width=3):
> v1 := vectorplot3d(<0,0,2>,label="u"):
> v2 := vectorplot3d(<1,0,0>,
>    thickness=thick,color=green):
> v3 := vectorplot3d(<-1,0,2>,
>    arrowstyle=arrow,color=cyan):
> plots[display]({p1,v1,v2,v3});
```



2.2 Linear Systems

The treatment of linear systems in $\mathbb{R}^2$ can be done quite easily by an intructor with sketches on a board or prepared in advance on an overhead transparency. For this reason we will only show examples from $\mathbb{R}^3$ where the computer is necessary; we need be able to move and rotate the graphic so that the orientation of the objects becomes clear.

Given a linear system in $\mathbb{R}^3$ input either as a set of equations or a matrix A and right-hand-side vector b the `linsysplot3d` command will, for each equation, draw the plane and label it with its equation. Thus the solutions will be seen as the point(s) where all planes intersect. Also displayed is a parametric description of the solution(s). We will illustrate the functionality by solving three systems each with three equations, one system with a single solution, one with a line of solutions, and one with no solution.

```
> linsysplot3d( {x+y+z=0, x-y+z=1, y-z=2} );
```

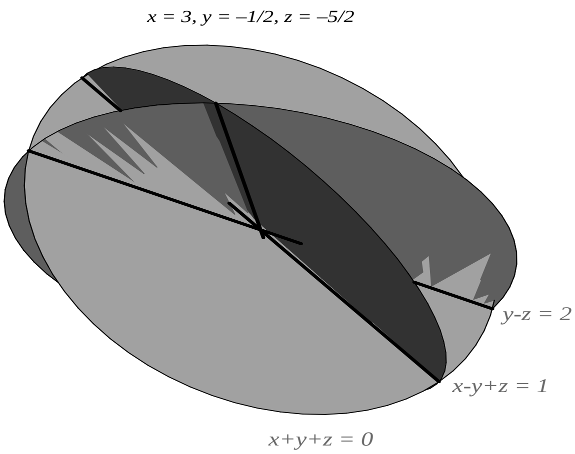


Figure 1: A linear system with one solution.

```
> linsysplot3d( {x+y+z=0, x-y+z=1, 2*x+2*z=1 },
>  planeshape=rectangular, planestyle=mesh,
>  planelabel=false, colors=[cyan,red,green] );
```

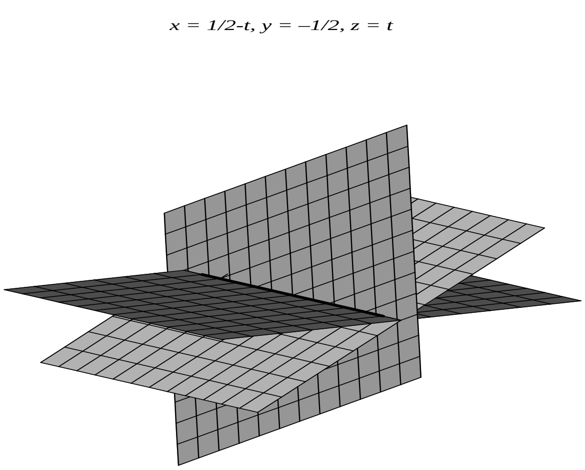


Figure 2: A linear system with a line of solutions.

```
> A := Matrix([[1,1,1],[1,-1,1],[1,0,1]]);
```

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

```
> b := <0,1,2>;
```

$$b := \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

```
> linsysplot3d(A,b);
```

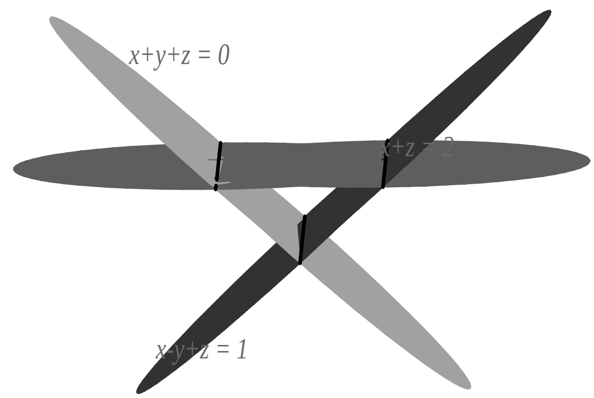


Figure 3: A linear system with no solutions.

One aspect of the Fraleigh and Beauregard text [1] that we particularly like is the True/False exercises which are mathematical statements which force the student to think. For example, consider the following statements:

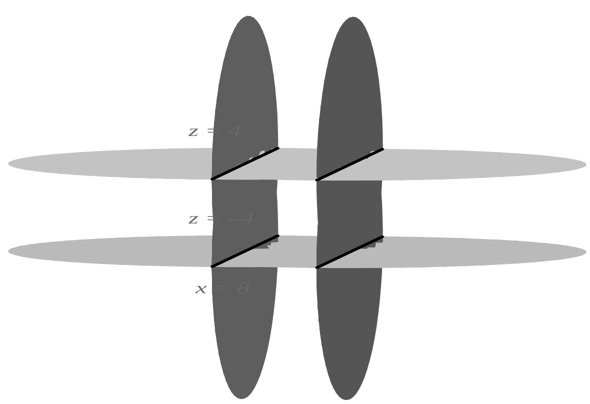
(i) A linear system over $\mathbb{R}$ with more equations than unknowns has no solution.

(ii) A linear system over $\mathbb{R}$ with fewer equations than unknowns has one or more solutions.

Generally, students find such questions more difficult than calculations. To answer these questions by thinking about what can happen when one row reduces an augmented matrix is not easy. It is our contention that being able to visualize linear systems in two and three dimensions will readily yield answers to these questions, moreover, with a picture in mind, stating counter examples follows easily.

The most difficult part of the `linsysplot3d` command is deciding what region of $\mathbb{R}^3$ to show if the user does not explicitly give us a region, a viewing box, in the form $x = a_1..b_1, y = a_2..b_2, z = a_3..b_3$. We compute the viewing box as follows. We include the origin in the viewing box. For each plane we compute the point on the plane closest to the origin and ensure this point is inside the viewing box. This ensures that each plane will be seen in the final graphic. For each pair of planes we compute their line of intersection, if any, and ensure that the point on this line closest to the origin is in the viewing box. We display all lines of intersection in heavy black so that the intersections of the planes are clearly visible. For each three planes we compute their common point of intersection, if any, and ensure that this point is in the viewing box. Thus all useful geometric information about the linear system will be inside the viewing box. For example, here is a plot of a system of four equations.

```
> linsysplot3d({z=-4,z=+4,x=2,x=8});
```



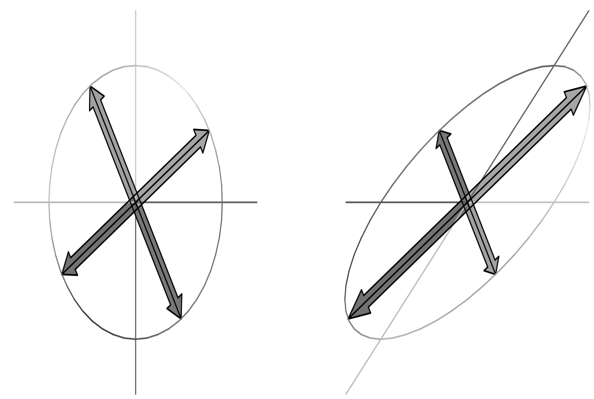
2.3 Linear Transformations

Linear transformations in texts are usually introduced algebraically rather than geometrically. The geometry if given is usually depicted by showing what happens to the unit square under the linear transformation given by the 2 by 2 matrix A. What is usually not explained is how this is actually showing what happens to all points in $\mathbb{R}^2$. Given $A \in \mathbb{R}^2$, the `lintransplot` command will show what happens to the unit circle under the linear transformation $T : \mathbb{R}^2 \to \mathbb{R}^2$ where $Tx = Ax$. We draw the unit circle on the left and the result of applying T to it on the right. If the rank of A is 2 then the result is an ellipse and if the rank of A is 1 then the result is a line. Similarly, given $A \in \mathbb{R}^3$ the `lintransplot3d` command will show what happens to the unit sphere under the linear transformation implied by A. If the rank of A is 3 then the result is an ellipsoid, if 2 an ellipse, and if 1 a line. In order that one can see rotations and reflections we encode points on the perimeter of the unit circle and the surface of the unit sphere in color. We draw the singular vectors as black lines and the real eigenvectors as arrows. We also compute and display the numerical values of the determinant, singular values, and the eigenvalues of A.

```
> A := Matrix([[1,1],[1,0]]);
```

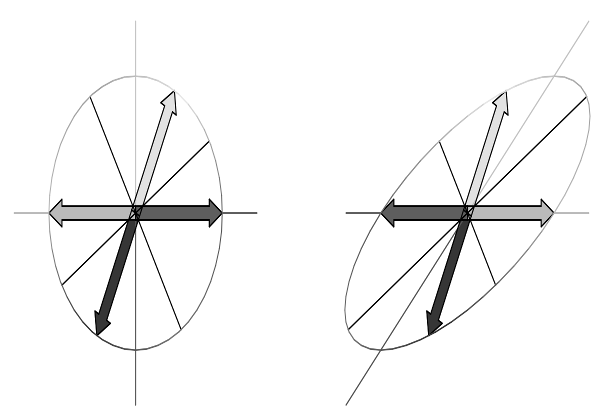
$$A := \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

```
> lintransplot(A);

[1 1]   determinant = -1
[1 0]   sigma = 1.62, .620   lambda = 1.62, -.618
```



In this example, $\det(A) = -1$ tells us that the area of the ellipse on the right is equal to that of the unit circle on the left. Recall that the singular vectors are the principal axes of the resulting ellipse and the singular values 1.62 and 0.62 tell us the relative sizes of the axes compared with those in the unit circle. In this example the singular vectors are the same as the eigenvectors. The next example shows that this need not be the case.

```
> A := Matrix([[-1,1],[0,1]]):
> lintransplot(A);

[-1 1]   determinant = -1
[ 0 1]   sigma = 1.62, .620   lambda = -1, 1
```

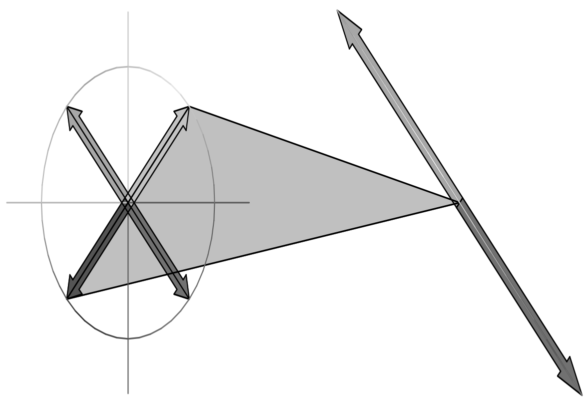


Looking at this picture for a moment, suppose we change the linear transformation to be $Sx = RAx$ for some rotation matrix $R \in \mathbb{R}^{2\times 2}$. Obviously this will result in rotating the ellipse on the right. Obviously (i) the singular vectors will rotate but their lengths will remain the same hence the singular values will not change and (ii) the area of the ellipse on the right will not change hence the magnitude of the determinant will not change.

This third example is an example where the matrix is singular. The kernel of A is displayed visually and a basis for the kernel is given numerically.

```
> A := Matrix([[1,-1],[-1,1]]):
> lintransplot(A);
```

```
[ 1 -1]  kernel = <1,1>
[-1  1]  sigma = 0, 2  lambda = 0, 2
```



The `lintransplot3d` command does the same but in three dimensions. We draw the surface of the unit sphere and the result of the linear transformation as wireframe objects so that we can the singular vectors and real eigenvectors inside them. Note the plots do not have a 1:1 scaling (so that the plot is not too small) and consequently the unit sphere does not look like a unit sphere.

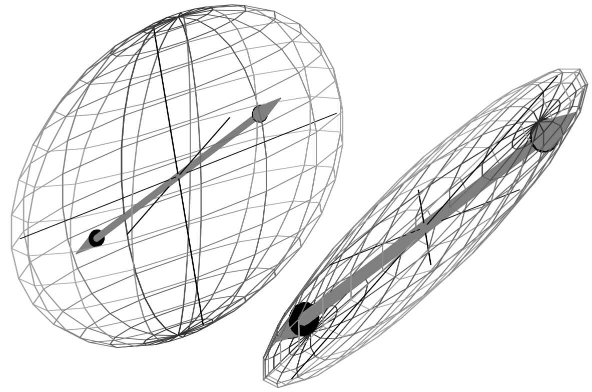
```
> F := Matrix([[0,1.2,2],
>   [.60,0,0],[0,0.81,0.81]]);
```

$$F := \begin{bmatrix} 0 & 1.2 & 2 \\ .60 & 0 & 0 \\ 0 & .81 & .81 \end{bmatrix}$$

```
> lintransplot3d(F,digits=2);
```

```
[  0 1.2   2]  determinant = .39
[.60   0   0]  lambda = 1.5, -.33+.39*I, -.33-.39*I
[  0 .81 .81]  sigma = 2.6, .60, .25
```

This next example shows the kernel visually as a plane inside the unit sphere as a subspace of $\mathbb{R}^3$. Also displayed



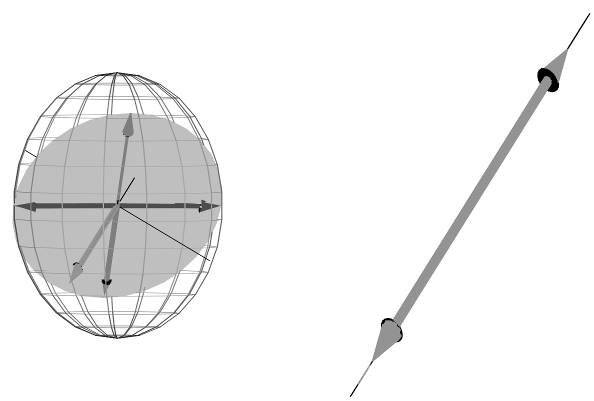
is a basis for the kernel. The two zero singular values also confirm numerically that the range of the linear transformation really is a line, and not just a very thin ellipsoid.

```
> A := Matrix([[0,1,1]$3]);
```

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

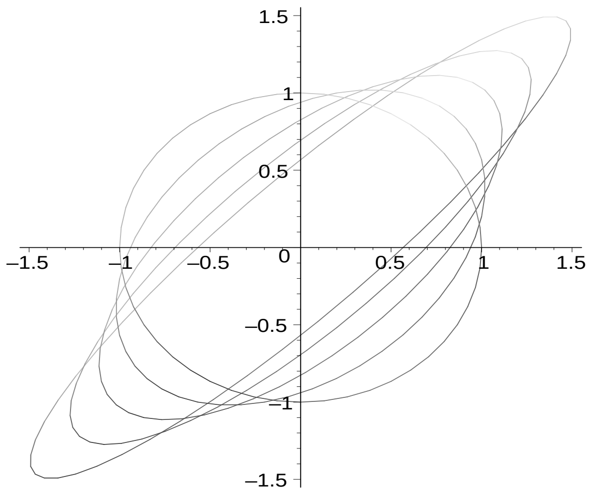
```
> lintransplot3d(A);
```

```
[0 1 1]  kernel = {<0.,.707,-.707>,<-1.,0.,0.>}
[0 1 1]  lambda = 0., 2., 0.
[0 1 1]  sigma = 2.45, 0., 0.
```



Linear transformations can also be visualized by animating them. The `applintrans` command applies a linear transformation in $\mathbb{R}^2$ to an ojbect. The ojbect may be unit square or unit circle, a user defined object given by a list of points specifying the boundary of the object, or a Maple two-dimensional plot object. For example, to show what happens to the unit circle we may do

```
> A := Matrix([[1,0.2],[0.2,1]]):
> applintrans(A,unitcircle,animated=false);
```



2.4 Eigenvalues and Eigenvectors

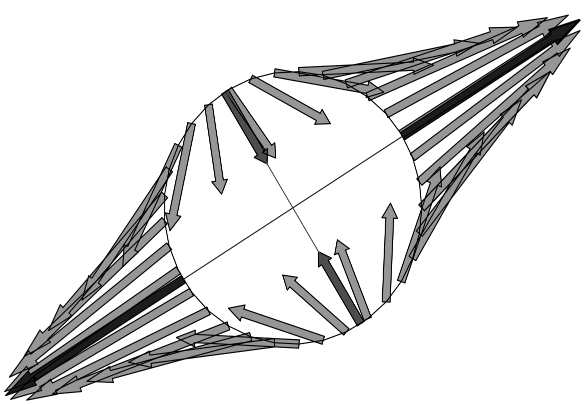
On input of $A \in \mathbb{R}^2$, the `eigenplot` command computes and displays the eigenvectors and, implicitly, the eigenvalues of A as arrows in red and blue and also their numerical values as text. This example shows a matrix with two real eigenvectors.

```
> A := Matrix([[1,1],[1,0]]):
```

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

```
> eigenplot(A);
```

```
[1 1]   lambda[1] = 1.62, v[1] = <1.62,1>
[1 0]   lambda[2] = -.620, v[2] = <-.620,1>
```



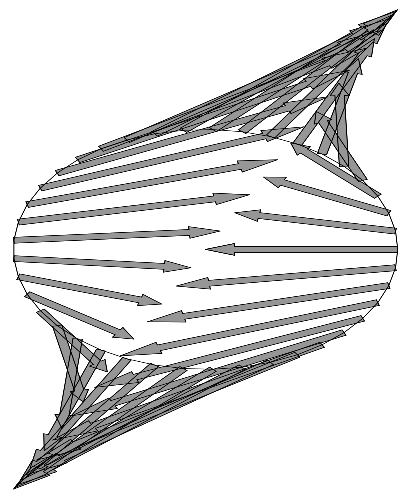
For m unit vectors $u_i, i = 1, \ldots, m$ equally spaced around the unit circle, the `eigenplot` command computes $v_i = Au_i$ and displays each v_i as a yellow arrow with its tail at u_i. This gives a visualization of the linear transformation implied by A. Visually, if m is sufficiently large, we will be able to see where the eigenvectors are. This gives a visual interpretation of the definition: A nonzero vector v is an *eigenvector* of A if $Av = \lambda v$ for some *eigenvalue* $\lambda \in \mathbb{R}$. In this next example the eigenvectors are not explicitly drawn so that the instructor may ask students to estimate their numerical values.

```
> A := Matrix([[-1,1],[0,1]]):
```

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

```
> eigenplot(A,eigenvectors=false);
```

```
[-1 1]   lambda[1] = -1, v[1] = <1,0>
[ 0 1]   lambda[2] = 1, v[2] = <1,2>
```



The most difficult cases to understand are the degenerate cases, namely, (i) a matrix with one zero eigenvalue, (ii) a matrix with two equal eigenvalues but with one eigenvector, and (iii) a matrix with two equal eigenvalues (where the eigenvectors change from real to complex). For example

```
> A := Matrix([[1,-1],[-1,1]]);
```

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

```
> eigenplot(A);
```

```
[ 1 -1]   lambda[1] = 2, v[1] = <-1,1>
[-1  1]   lambda[2] = 0, v[2] = <1,1>
```

```
> A := Matrix([[1,1],[0,1]]):
```

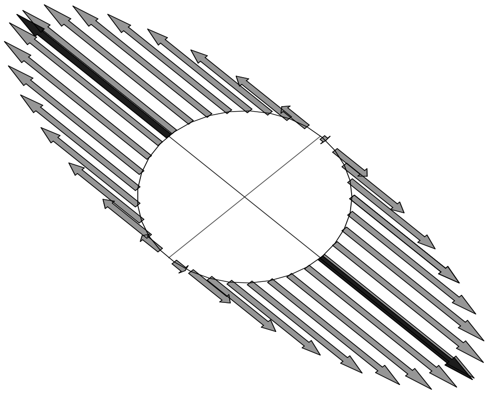
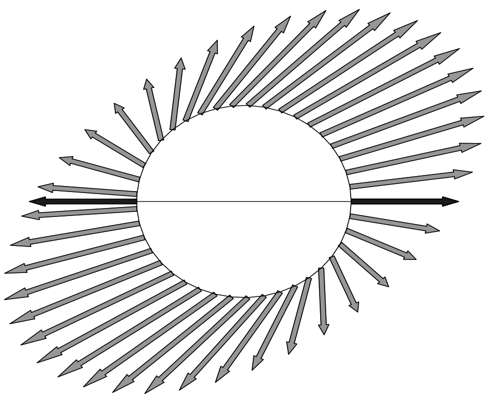


Figure 4: A matrix with a zero eigenvector.

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

```
> eigenplot(A);

[1 1]   lambda[1] = 1, v[1] = <1,0>
[0 1]   lambda[2] = 1  with geometric multiplicity 1
```



These static pictures do not reveal everything. In fact, we cannot illustrate case (iii) with a static plot easily. The `animate` command in Maple 8 will permit us to animate a plot on one parameter. Hence by animating the following matrices as ϵ passes through zero we can get a better picture of these degenerate cases. Notice the difference between cases (ii) and (iii).

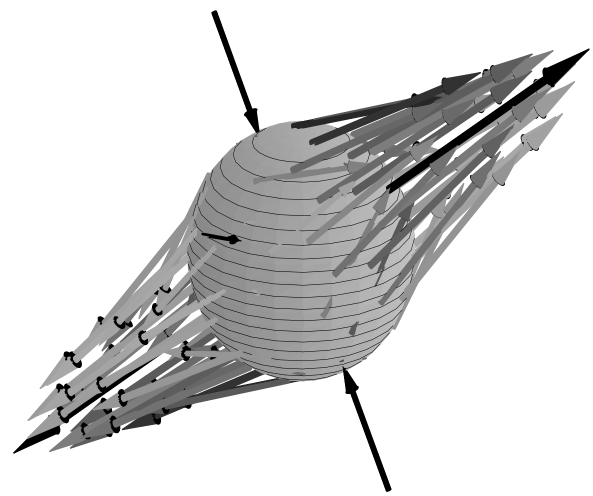
```
> A := Matrix([[1,-1],[-1,1+epsilon]]);
> animate(eigenplot,[A],epsilon=-1..1);
> A := Matrix([[1,1],[0,1+epsilon]]);
> animate(eigenplot,[A],epsilon=-1..1);
> A := Matrix([[1,1],[epsilon,1]]);
> animate(eigenplot,[A],epsilon=-1..1);
```

On input of $A \in \mathbb{R}^3$, the `eigenplot3d` command computes and displays the eigenvectors and eigenvalues of A as arrows in black and also their numerical values as text. This example shows a matrix with three real eigenvectors. Eigenvectors with negative eigenvalues are indicated by arrows pointing towards the surface of the unit sphere instead of away from it.

```
> A :=  Matrix([[1,1.5,2],[1.5,1,1.5],[2,1.5,1]]);
```

$$A := \begin{bmatrix} 1 & 1.5 & 2 \\ 1.5 & 1 & 1.5 \\ 2 & 1.5 & 1 \end{bmatrix}$$

```
> eigenplot3d(A,digits=2);

[  1 1.5   2]  lambda[1]=4.4   v[1]=<.60,.54,.60>
[1.5   1 1.5]  lambda[2]=-.35  v[2]=<-.38,.85,-.38>
[  2 1.5   1]  lambda[3]=-1.   v[3]=<.71,.26e-15,-.71>
```



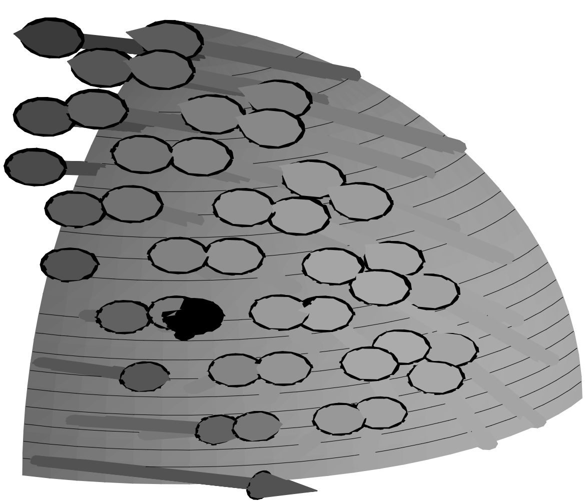
For $m \in \{20, 80, 320, ...\}$ unit vectors $u_i, i = 0, 1, ..., m$ approximately equally spaced around the unit sphere, the `eigenplot3d` command computes $v_i = Au_i$ and displays each v_i as an arrow with its tail at position u_i with RGB color values given by the absolute value of the xyz coordinates of the vector. This color assignment has the nice property that vectors pointing in opposite directions will have the same color.

In this second example, which is an example of a Leslie matrix, the dominant eigenvector is in the positive quadrant. In order to see more clearly what is happening in the positive quadrant we use the region option. We have also rotated the plot so that the dominant eigenvector is pointing at us; it is the black blob in the figure.

```
> F := Matrix([[0,1.2,2],[.60,0,0],[0,.81,.82]]);
```

$$F := \begin{bmatrix} 0 & 1.2 & 2 \\ .60 & 0 & 0 \\ 0 & .81 & .82 \end{bmatrix}$$

```
> eigenplot3d(F,region=posoct);

[  0 1.2   2]  lambda[1]=1.48          v[1]=...
[.60   0   0]  lambda[2]=-.330+.386I   v[2]=...
[  0 .81 .82]  lambda[3]=-.330-.386I   v[3]=...
```



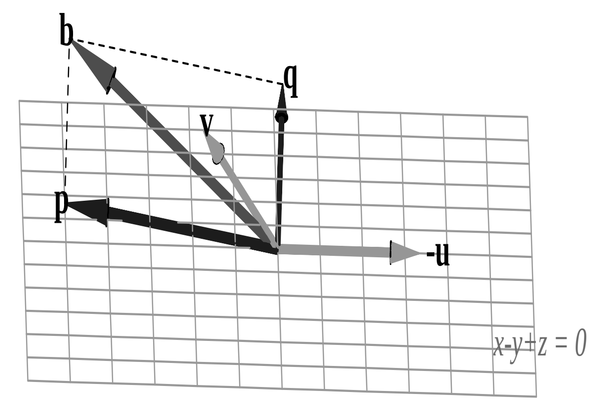
The matrix F has one real and two complex eigenvectors. The effect of the complex eigenvectors on unit vectors near the real eigenvector is to cause a rotation or twist around the real eigenvector. It can be shown that for a Leslie matrix L, if $u \in \mathbb{R}^n$ satisfies $u_i > 0$, then the limit of the sequence $Lu, L^2u, L^3u, ...$ is the dominant eigenvector of L. Visually, this is seen by observing that all vectors close to the boundary of the positive quadrant point into the positive quadrant.

2.5 Projection

Given a vector $b \in \mathbb{R}^3$ and a basis $\{u, v\}$ for a plane $W \subset \mathbb{R}^3$, the `projectionplot3d` command displays the vectors u, v, b, the plane W, the projection vector p of b onto W, and the vector $q = p - b$ which is the normal vector to W. Note that in the following example, the vector $-u$, not u is what is displayed. When p is too close to u or v we display $-u$, respectively, $-v$ instead. Displayed also are the numerical values of b, u, v, p, q and $||q||$ the distance from b to the closest point on W.

```
> b,W := <2,1,1>,{<1,1,0>,<0,1,1>}:
```

```
> projectionplot3d(b,W);

b = <2,1,1>,  u = <1,1,0>,  v = <0,1,1>
p = <4/3,5/3,1/3>,  q = <2/3,-2/3,2/3>,  ||q|| = 1.15
```



We can use this picture to explain the steps of the Gram Schmidt orthogonalization procedure for computing an orthogonal basis for $\{b, u, v\}$. We will do this in Maple for illustration. We first project v onto u to obtain an orthogonal basis for W. Next we compute p the projection of b onto W and $q = b - p$ and then we plot b, p, q and W.

```
> with(LinearAlgebra):
> Project := proc(b,u) (u.b)/(u.u)*u end:
> u,v := <1,1,0>,<0,1,1>:
> v := v-Project(v,u);
```

$$v := \begin{bmatrix} \frac{-1}{2} \\ \frac{1}{2} \\ 1 \end{bmatrix}$$

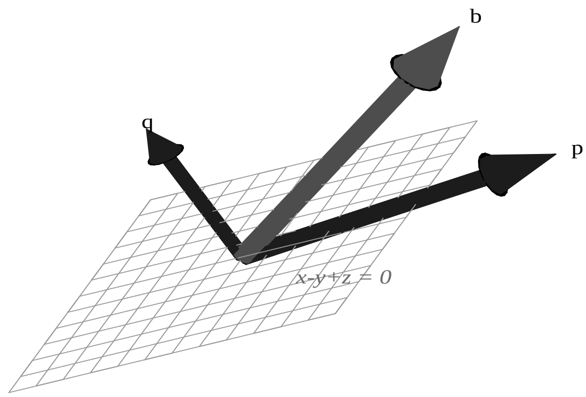
```
> b := <2,1,1>:
> p := Project(b,u)+Project(b,v);
```

$$p := \begin{bmatrix} \frac{4}{3} \\ \frac{5}{3} \\ \frac{1}{3} \end{bmatrix}$$

```
> q := b-p;
```

$$q := \begin{bmatrix} \frac{2}{3} \\ \frac{-2}{3} \\ \frac{2}{3} \end{bmatrix}$$

```
> B := vectorplot3d(b,label="b",color=red):
> P := vectorplot3d(p,label="p",color=blue):
> Q := vectorplot3d(q,label="q",color=blue):
> W := planeplot({u,v},shape=rectangular,
>        style=grid,radius=2):
> plots[display]({B,P,Q,W});
```

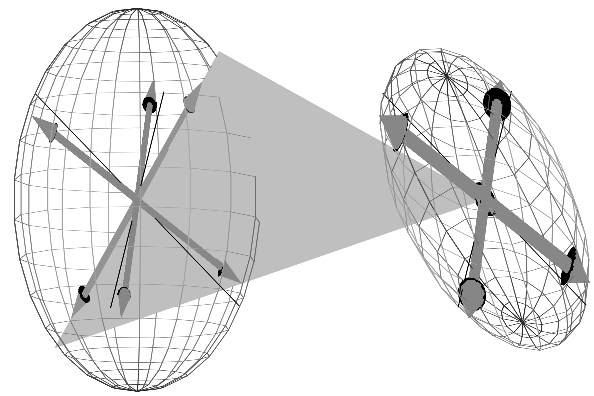


We construct the projection matrix P by projecting the elementary vectors onto W. We display the linear transformation $Tx = Px$. The kernel of T, the set of points mapped by T to the origin, a line in this example, is depicted by the triangle in the plot below. Now since the kernel is a line we know that the dimension of the nullspace of P is 1 hence the dimension of the range of T is 2 hence, although in the plot the range looks like an ellipsoid, it really has no volume. This is easily confirmed by rotating the plot. Notice also that one of the eigenvectors appears to be equal to the kernel. Indeed any non-zero vector in the nullspace of a matrix is an eigenvector of the matrix with eigenvalue 0, a fact which is confirmed in this plot.

```
> i,j,k := <1,0,0>,<0,1,0>,<0,0,1>:
> P := Matrix( [
>   Project(i,u)+Project(i,v),
>   Project(j,u)+Project(j,v),
>   Project(k,u)+Project(k,v) ] );
```

$$P := \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & \frac{-1}{3} \\ \frac{1}{3} & \frac{2}{3} & \frac{1}{3} \\ \frac{-1}{3} & \frac{1}{3} & \frac{2}{3} \end{bmatrix}$$

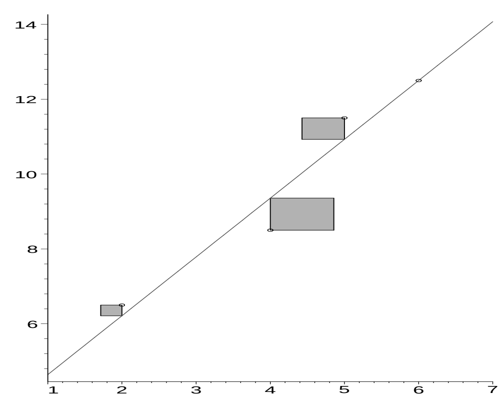
```
> lintransplot3d(P);

[ 2/3 1/3 -1/3]  kernel = {<.577,-.577,.577>}
[ 1/3 2/3  1/3]  lambda = .100e-9, 1., 1.00
[-1/3 1/3  2/3]  sigma = 1.00, 1., .100e-9
```



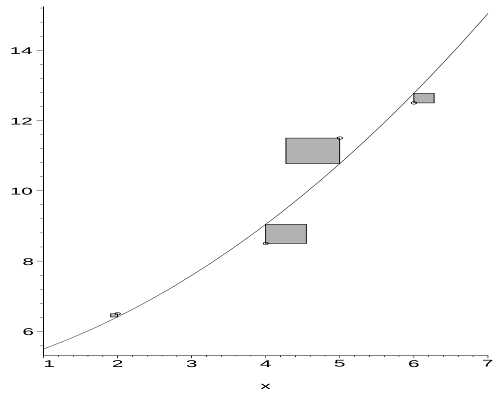
2.6 Least Squares

On input of n data points in $\mathbb{R}^2$, by default, the `leastsqrsplot` command by fits the data with the best linear function $f(x) = ax + b$ in the least squares sense. It displays the data points, line of best fit, and indicates the error of the fit visually by drawing squares. It also displays in text the best fit and the numerical value of the residual, that is, the area of the squares in the plot. In the example below we fit a line and a quadratic to the given data.

```
> X := [2,4,5,6]:
> Y := [6.5,8.5,11.5,12.5]:
> leastsqrsplot(X,Y,x=1..7);

fit = 3.07+1.57*x
error^2 = 1.142857
```

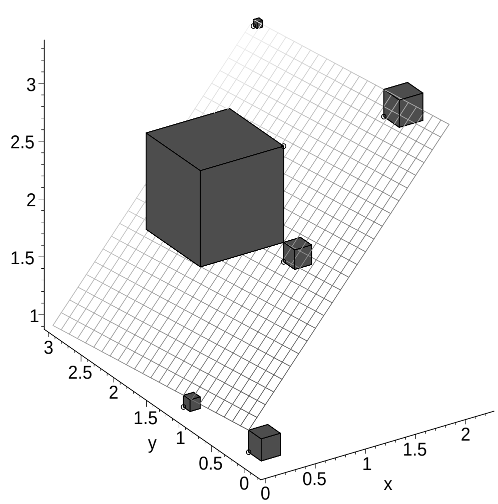


```
> leastsqrsplot(X,Y,basis=[1,x,x^2]);

fit = 4.86+.500*x+.136*x^2
error^2 = .909091
```



Similarly, on input of n data points in $\mathbb{R}^3$, the `leastsqrsplot3d` command fits the data with a plane $f(x,y) = ax + by + c$ using least squares. It displays the data points, plane of best fit, indicates the error by drawing cubes, and displays the fit information in text. The user may specify a different basis for the fit.

```
> X := [0,0,1,1,1,2,2]:
> Y := [0,1,1,2,1,1,3]:
> Z := [1,1,2,2,3,3,3]:
leastsqrsplot3d(X,Y,Z,x=0..2,y=0..3,
   scaling=constrained);

fit = 1.19+1.07*x-.899e-1*y
error^2 = .831461
```



3 Conclusion

All the examples of using the computer in a first Linear Algebra course that we cited in the introduction are targeted at the students using the computer and not the instructor. None suggest that the instructor teach the subject on the computer in the lecture. This is surely because a competent instructor can teach more effectively on the board. In this paper we have suggested one place where the computer will be appropriate in the lecture, as well as in the lab, namely, visualizing basic concepts of Linear Algebra in $\mathbb{R}^2$ and $\mathbb{R}^3$. Linear Algebra in $\mathbb{R}^n$ is a geometric subject. Even if textbooks develop the subject primarilly by stating and proving theorems algebraically, we contend that many concepts are more easily understood, and will be more easily recreated later, by thinking geometrically, and thus we think time should be spent in developing geometric intuition and understanding. In this paper we have presented a package of visualization routines being developed by the author for Maple for this purpose and have mentioned some theorems and conclusions that can be made from the visuals.

Acknowlegement

We are grateful to Allan Gold of the University of Windsor, Ontario, for sharing the idea for visualizing eigenvectors in $\mathbb{R}^2$ with us. We also acknowledge Larry Weldon of Simon Fraser University, British Columbia, for showing us a Java applet for least squares approximation in one variable.

References

[1] J.B. Fraleigh and R.A. Beauregard, *Linear Algebra*, 3rd edition, Addison-Wesley, 1995.

[2] E.W. Johnson, it Linear Algebra with Maple V, Brooks/Cole Pub. Com., 1993.

[3] H.A.W.M. Kneppers, The Influence of Maple on a Linear Algebra Course at the Delft University of Technology, *Proceedings of the 1994 Maple Summer Workshop*, R.J. Lopez editor, Birkhäuser, pp. 57–62, 1994.

[4] E.A. Herman, M.D. Pepe, R.T. Moore, J.R. King, *Linear Algebra: Modules for Interactive Learning Using Maple 6* 1st edition, Addison-Wesley, 2001.

[5] M.B. Monagan, *Teaching and Doing Mathematics with Maple*, Maple Course brochure (2001) at: `http://www.cecm.sfu/MapleBrochure.pdf`

[6] W.K. Nicholson, *Elementary Linear Algebra*, 1st edition, McGraw-Hill Ryerson, 2001.