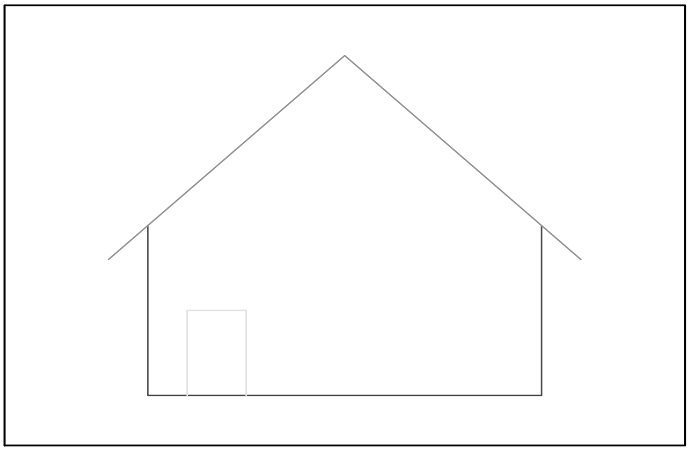
Ma 322 Linear Algebra Notes

Carl Eberhart
Department of Mathematics, University of Kentucky
carl@ms.uky.edu



July 16, 1997

Contents

Ma 322 Section 6 – Spring 1996

Information Sheet

Time and Place: 2-3:15 pm Tues and Thurs, Rm CB 341

Teacher: Carl Eberhart, 759 POT, (606) 257-1258, carl@ms.uky.edu

Office Hours: 4-5:30 MTWR, or by appointment. We can also consult by phone or email. Text: Introduction to Linear Algebra, by Gilbert Strang

Course Goals: The language of vectors and matrix algebra has proven to be one of the most useful tools in mathematics. The ideas in this course have application to all other areas of mathemtics, and most other sciences. Our primary goal is help you construct a working knowledge of this valuable subject, by all means possible. This includes the use of such programs as Maple and Matlab in our studies.

Topics: The algebra and geometry of vectors, dot product, matrix multiplication, Gaussian elimination, LU factorization, vector spaces, linear independence, dimension, basis, subspaces, linear transformations, QR factorization, determinants, eigenvalues and eigenvectors, diagonalization. We will cover most of chapters 1 thru 7 of the text.

Homework, quizzes, and class attendance: There will be weekly homework assignments and periodic quizzes. Plan on spending at least 4 to 6 hours per week outside of class reading the text and working on problems. Come to class.

Testing: There will be two midterms and a final exam. The midterms will be during class on Thursday, Feb 15 and Thursday, March 28. The final will be on Tuesday, April 30, 1 - 3 pm.

Grades: Grades will be based on 400 points: 100 points for each midterm, 120 points for the final, and an 80 point instructor grade, which will be based on your homework and classwork average. Your point total will be divided by 4 and a letter grade assigned: 90-100 A, 80-89 B, 70-79 C, 60-69 D, and below 60 an E.

Calendar of Class days: Jan 11, 16, 18, 23, 25, 30; Feb 1, 6, 8, 13, 15, 20, 22, 27, 29; March 5, 7, 19, 21, 26, 28; April 2, 4, 9 11, 16, 18, 23, 25.

- Intro to vectors

- Cover: Getting the pictures on the cover of Strang.

- Hilbert: invertible matrices

- rods : Using eigenstuff

- Markov : Another use of eigenstuff

- plu factorization

1 Using Vectors

In this worksheet, we start using the linear algebra package in Maple. It turns out to be very useful.

1.1 Some sample vector calculations

Here are some examples of how to do vector algebra with the Maple package with(linalg).

```
with(linalg);
```

[*BlockDiagonal*, *GramSchmidt*, *JordanBlock*, *LUdecomp*, *QRdecomp*, *Wronskian*, *addcol*, *addrow*, *adj*, *adjoint*, *angle*, *augment*, *backsub*, *band*, *basis*, *bezout*, *blockmatrix*, *charmat*, *charpoly*, *cholesky*, *col*, *coldim*, *colspace*, *colspan*, *companion*, *concat*, *cond*, *copyinto*, *crossprod*, *curl*, *definite*, *delcols*, *delrows*, *det*, *diag*, *diverge*, *dotprod*, *eigenvals*, *eigenvalues*, *eigenvectors*, *eigenvects*, *entermatrix*, *equal*, *exponential*, *extend*, *ffgausselim*, *fibonacci*, *forwardsub*, *frobenius*, *gausselim*, *gaussjord*, *geneqns*, *genmatrix*, *grad*, *hadamard*, *hermite*, *hessian*, *hilbert*, *htranspose*, *ihermite*, *indexfunc*, *innerprod*, *intbasis*, *inverse*, *ismith*, *issimilar*, *iszero*, *jacobian*, *jordan*, *kernel*, *laplacian*, *leastsqrs*, *linsolve*, *matadd*, *matrix*, *minor*, *minpoly*, *mulcol*, *mulrow*, *multiply*, *norm*, *normalize*, *nullspace*, *orthog*, *permanent*, *pivot*, *potential*, *randmatrix*, *randvector*, *rank*, *ratform*, *row*, *rowdim*, *rowspace*, *rowspan*, *rref*, *scalarmul*, *singularvals*, *smith*, *stack*, *submatrix*, *subvector*, *sumbasis*, *swapcol*, *swaprow*, *sylvester*, *toeplitz*, *trace*, *transpose*, *vandermonde*, *vecpotent*, *vectdim*, *vector*, *wronskian*]

First, a vector is defined as an array. So, the vectors ——u = <3,1>— and ——v = <3,4>— would be defined

```
u := array([3,1]);
```

$$u := [3,\ 1]$$

```
v := array([3,4]);
```

$$v := [3,\ 4]$$

To get the angle formed by u and v, use the Maple word angle, and to convert this angle to degrees, use the Maple words ——evalf— and ——convert—.

```
ang := angle(u,v);
```

$$ang := \arccos(\frac{13}{250}\sqrt{10}\sqrt{25})$$

```
evalf(convert(ang,degrees));
```

$$34.69515354\ degrees$$

Now to form a linear combination of u and v, use the word ——evalm—.

```
w := evalm(t*u + s*v);
```

$$w := [3\,t + 3\,s,\ t + 4\,s]$$

The first component of w is referenced by w[1], the second component by w[2].

```
w[1];
```

$$3\,t + 3\,s$$

So, now suppose I wanted to find the line in space which passes through —— A = (2,1,9)—, ——B = (-3,0,1)— and crosses the xy-plane. We could set up a vector equation for this line.

```
A := array([2,1,9]);
```

$$A := [2,\ 1,\ 9]$$

```
B := array([-3,0,1]);
```

$$B := [-3,\ 0,\ 1]$$

```
X := evalm(A+t*(B-A));
```

$$X := [2 - 5\,t,\ 1 - t,\ 9 - 8\,t]$$

Now the line crosses the xy-plane when the third component of X is 0, so we solve ——z = 0— for t and substitute that into x and y.

```
t:=solve(X[3]=0,t );
```

$$t := \frac{9}{8}$$

```
[x,y] = [X[1],X[2]];
```

$$[x,\ y] = [\frac{-29}{8},\ \frac{-1}{8}]$$

```
x :='x':  y:='y':  t:='t';
```

$$t := t$$

1.2 Using Vector Algebra to solve problems

As we have mentioned earlier, you enter the Maple package linalg before doing vector algebra on Maple.

The most useful words for our purposes are ——crossprod—, ——det—, ——dotprod—, ——equal—, ——grad—, ——linsolve—, ——matrix—, ——norm—, and ——vector—. As before, in order to learn how to use a word just type ?word. For example, ?vector gives the help screen for the word vector. So to define the vector v pointing from ——A = (1,2,4)— to —— B = (-3,1,-8)—, we could write

```
A := vector([1,2,4]);
```

$$A := [1,\ 2,\ 4]$$

```
B := vector([-3,1,-8]);
```

$$B := [-3,\ 1,\ -8]$$

```
v := evalm(B-A);
```

$$v := [-4, -1, -12]$$

Here the word evalm is a Maple word that you can use to compute linear combinations of vectors.
˜

Problem: Suppose we wanted to find the line passing through A, B and crosses the yz-plane. A point X on the line through A and B would be of the form ——A + tv— for some scalar t.

```
X := evalm(A+t*v);
```

$$X := [1 - 4\,t,\ 2 - t,\ 4 - 12\,t]$$

To reference a component of a vector, write the vector followed by the index enclosed in square brackets. The first component of X is X[1]. If we wanted to know where the line passing through A and B crossed the yz-plane, we could solve ——X[1] $=$ 0— for t and then substitute that value of t into X.

```
subs(t=solve(X[1]=0,t) ,evalm(X));
```

$$\left[0,\ \frac{7}{4},\ 1\right]$$

The line crosses the yz-plane at $y = \frac{7}{4}$, $z = 1$.
˜—Another problem:— What if we wanted to prove using vector methods that the midpoints of an arbitrary quadrilateral ABCD in the plane is a parallelogram?
˜

Well, set up the vectors first.

```
A := vector(2):

B := vector(2):

C := vector(2):

DD := vector(2):
```

Next, calculate the midpoints of each of the segments AB, BC, CD, and DA.

```
M1 := evalm(1/2*(A+B)):

M2 := evalm(1/2*(B+C)):

M3 := evalm(1/2*(C+D)):

M4 := evalm(1/2*(D+A)):
```

Next, we want to show that the quadrilateral M1M2M3M4 is a parallelogram. This can be done in several ways. For example, we could show the vectors M2 - M1 and M3 - M4 are equal, ie, their difference is the zero vector. (Draw a picture for yourself here.)

```
evalm((M2-M1)-(M3-M4));
```

$$[0, 0]$$

Well, that proves it.

Another problem: Lets get the distance from a point $X = (3,\ 2,\ 5)$ to the line through $A = (2,\ 0,\ 0)$ and $B = (0,\ 0,\ 4)$.

First set up the vectors. (Draw a picture for yourself to see what's going on.)

```
A := vector([3,2,5]):
```

```
B := vector([2,0,0]):
```

```
X := vector([0,0,4]):
```

Next, calculate the projection of the vector from A to X onto the vector from A to B.

```
proj :=
evalm(dotprod(X-A,B-A)/dotprod(B-A,B-A)*(B-A));
```

$$proj := \left[\frac{-2}{5},\ \frac{-4}{5},\ -2\right]$$

Now calculate the perpendicular projection ——perp = (X - A) - proj—.

```
perp := evalm(X-A-proj);
```

$$perp := \left[\frac{-13}{5},\ \frac{-6}{5},\ 1\right]$$

The distance we want is the magnitude of perp.

```
dist := sqrt(dotprod(perp,perp));
```

$$dist := \frac{1}{5}\sqrt{230}$$

We can create a Maple word, ie, procedure out of this.

```
distpttoline := proc(X,A,B)
local proj,perp;
proj := evalm(dotprod(X-A,B-A)/dotprod(B-A,B-A)*(B-A));
perp := evalm(X-A-proj);
sqrt(dotprod(perp,perp))
end:
```

For example, if we wanted to get the distance from the point (3,4,1) to the line through the points X and B,

```
distpttoline(vector([3,4,1]),X,B);
```

$$\frac{1}{5}\sqrt{445}$$

1.3 Some Exercises

(a) For the points $A = (3,\ 2)$, $B = (2,\ 5)$ and $C = (7,\ 9)$:

```
restart; with(linalg):
```

```
Warning, new definition for norm
```

```
Warning, new definition for trace
```

```
a := [3,2] ; b := [2,5] ; c := [7,9] ;
```

$$a := [3,\ 2]$$

$$b := [2,\ 5]$$

$$c := [7,\ 9]$$

(i) Find the magnitude of the vectors from A to B, from B to C, and from C to A. Verify that the sum of any two is greater than the third. ˜

```
ab := norm(b-a,2); bc := norm(c-b,2); ca :=
norm(a-c,2);
```

$$ab := \sqrt{10}$$

$$bc := \sqrt{41}$$

$$ca := \sqrt{65}$$

```
map(evalf,[ab+bc,ca,bc+ca,ab,ca+ab,bc]);
```

$$[9.565401897,\ 8.062257748,\ 14.46538199,\ 3.162277660,\ 11.22453541,\ 6.403124237]$$

(ii) Find the measures of the angle ABC, the angle BCA, and the angle BAC. Verify that the sum of these is π. ˜

```
anga :=
evalf(convert(angle(b-a,c-a),degrees));
angb := evalf(convert(angle(a-b,c-b),degrees));
angc := evalf(convert(angle(b-c,a-c),degrees));
anga+angb+angc;
```

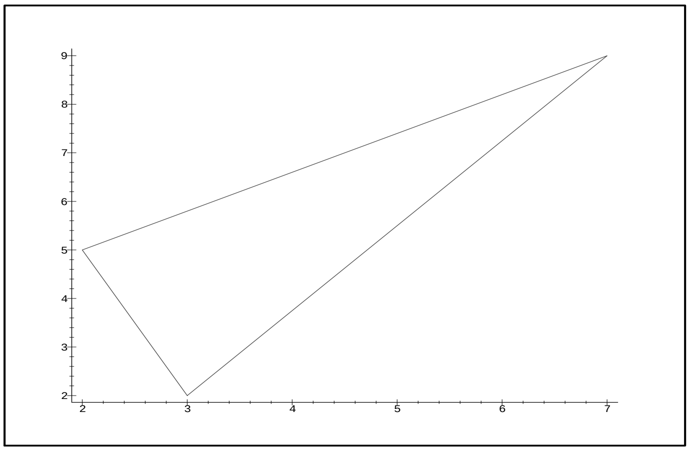
$$anga := 48.17983012\ degrees$$

$$angb := 110.2248594\ degrees$$

$$angc := 21.59531042\ degrees$$

$$179.9999999\ degrees$$

```
plot([a,b,c,a] );
```



(iii) Find the projection of the vector AB along the vector AC. ˜

```
prj :=
evalm(dotprod(b-a,c-a)/dotprod(c-a,c-a)*(c-a));
```

$$prj := \left[\frac{68}{65}, \frac{119}{65}\right]$$

(iv) Find the component of the vector AB along the vector AC. ˜

```
norm(prj,2);
```

$$\frac{17}{65}\sqrt{65}$$

(v) Find the perpendicular projection of the vector AB along the vector AC. ˜

```
perp := evalm((b-a)-prj);
```

$$perp := \left[\frac{-133}{65}, \frac{76}{65}\right]$$

(vi) Find the area of triangle ABC. One way: 1/2 the base times the height.

```
area := 1/2*norm(c-a,2)*norm(perp,2);
```

$$area := \frac{19}{2}$$

Another way: 1/2 * bc sin(angle b) ab

```
1/2*norm(c-b,2)*sin(angle(a-b,c-b))*norm(b-a,2); simplify(");
```

$$\frac{19}{820}\sqrt{41}\sqrt{410}\sqrt{10}$$

$$\frac{19}{2}$$

Yet another way: 1/2 norm(crossprod(b-a,c-a))

```
u := evalm(b-a);v := evalm(c-a);
1/2*norm(crossprod([u[1],u[2],0],[v[1],v[2],0]));
```

$$u := [-1,\ 3]$$

$$v := [4,\ 7]$$

$$\frac{19}{2}$$

(b) If the wind is blowing at 25 miles per hour to the Northeast, and our plane has an airspeed of 300 mph, what should our heading be if we are planning to go 100 miles due North? How long will it take us to make the trip?

```
restart; with(linalg):
```

```
Warning, new definition for norm

Warning, new definition for trace
```

```
wind := 25* [ cos(Pi/4), sin(Pi/4)] ;
```

$$wind := [\frac{25}{2}\sqrt{2},\ \frac{25}{2}\sqrt{2}]$$

```
air := 300*[cos(theta), sin(theta)] ;
```

$$air := [300\cos(\theta),\ 300\sin(\theta)]$$

```
eq := evalm(t*(wind+air)-[0,100]) ;
```

$$eq := \left[t\,(300\cos(\theta) + \frac{25}{2}\sqrt{2}),\ t\,(300\sin(\theta) + \frac{25}{2}\sqrt{2}) - 100\right]$$

```
th := solve(evalm(eq)[1],theta);
```

$$th := \pi - \arccos(\frac{1}{24}\sqrt{2})$$

```
tim := subs(theta=th,solve(eq[2],t));
ang := evalf(convert(th,degrees));
```

$$tim := \frac{8}{24\sin(\pi - \arccos(\frac{1}{24}\sqrt{2})) + \sqrt{2}}$$

$$ang := 93.37814305\ degrees$$

```
z := vector([0,0]);
```

$$z := [0,\ 0]$$

```
evalf([ang,tim]);
```

$$[93.37814305\ degrees,\ .3153018678]$$

```
op(air);
```

$$300\cos(\theta),\ 300\sin(\theta)$$

We can draw a picture of this.

```
vec := (a,b) -> plot([convert(a
,list),convert(b,list)] ,scaling=constrained);
```

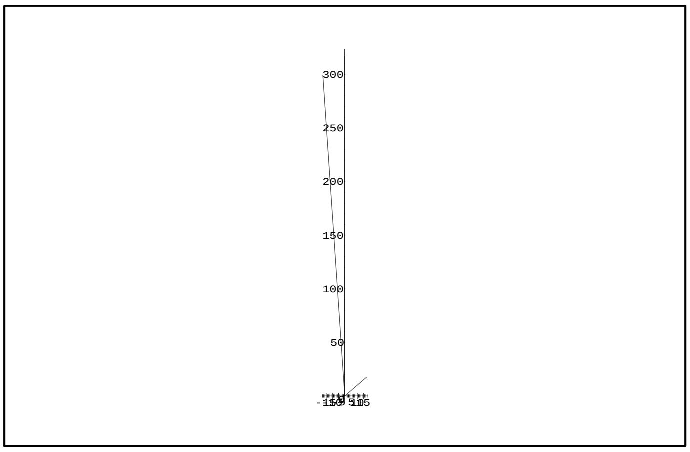
$$vec := (a,\, b) \to \mathrm{plot}([\mathrm{convert}(a,\, list),\, \mathrm{convert}(b,\, list)],\, scaling = constrained)$$

```
good := subs(theta=th ,[op(air)]) ;
```

$$good := [300\cos(\pi - \arccos(\frac{1}{24}\sqrt{2})),\ 300\sin(\pi - \arccos(\frac{1}{24}\sqrt{2}))]$$

```
pl1 := vec(z,good):
```

```
pl2 := vec(z,wind):
pl3 := vec(z,evalm(wind+good) ):
plots[display]([pl1,pl2,pl3 ]);
```



Back

2 Generating the pictures on the cover of Strang.

The six views of a house shown on the cover of our textbook, are images of the same figure under six different linear transformations. Let's see how to generate these views (or close to them).

```
with(linalg);
```

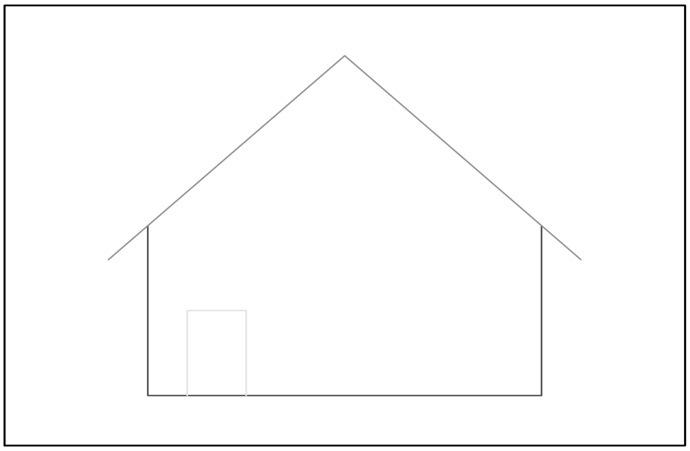
[*BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian, addcol, addrow, adj, adjoint, angle, augment, backsub, band, basis, bezout, blockmatrix, charmat, charpoly, cholesky, col, coldim, colspace, colspan, companion, concat, cond, copyinto, crossprod, curl, definite, delcols, delrows, det, diag, diverge, dotprod, eigenvals, eigenvalues, eigenvectors, eigenvects, entermatrix, equal, exponential, extend, ffgausselim, fibonacci, forwardsub, frobenius, gausselim, gaussjord, geneqns, genmatrix, grad, hadamard, hermite, hessian, hilbert, htranspose, ihermite, indexfunc, innerprod, intbasis, inverse, ismith, issimilar, iszero, jacobian, jordan, kernel, laplacian, leastsqrs, linsolve, matadd, matrix, minor, minpoly, mulcol, mulrow, multiply, norm, normalize, nullspace, orthog, permanent, pivot, potential, randmatrix, randvector, rank, ratform, row, rowdim, rowspace, rowspan, rref, scalarmul, singularvals, smith, stack, submatrix, subvector, sumbasis, swapcol, swaprow, sylvester, toeplitz, trace, transpose, vandermonde, vecpotent, vectdim, vector, wronskian*]

First, get the house together. It consists of walls (and floor), a roof, and a door slightly offset from the center.

```
walls := [[-1,1],[-1,0],[1,0],[1,1]] :
roof := [[-1.2,.8],[0,2],[1.2,.8]]:
door := [[-.8,0],[-.8,.5],[-.5,.5],[-.5,0]]:
house := {walls,roof,door}:
```

So, the house in the upper left hand corner is the picture we have below.

```
plot(house,scaling=constrained,axes=none,thic
kness=3);
```



Now the middle right picture is obtained by flipping about the y axis. The 2 by 2 matrix can be muliplied by each point in the picture to get the newhouse.

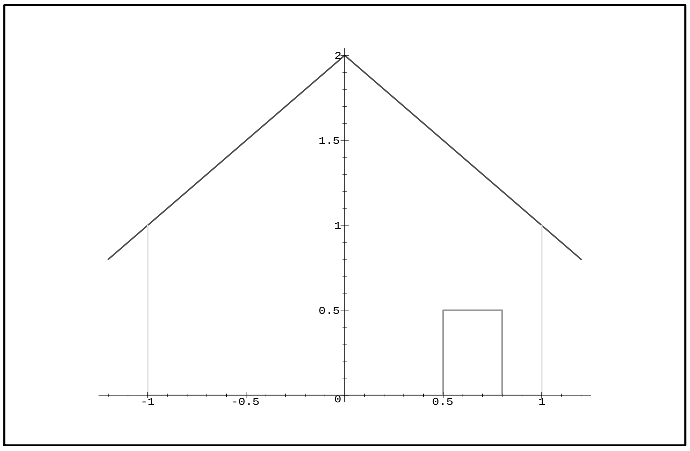
```
reflect := matrix([[-1,0],[0,1]]);
```

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

```
newhouse := NULL:
for part in house do
newpart := NULL:
for pt in part do
npt := evalm(reflect &* pt) ;
newpart := newpart,[npt[1],npt[2]];
od:
newhouse := newhouse,[newpart];
od:
newhouse := {newhouse};
```

$newhouse := \{$
$[[1.2,\ .8],\ [0,\ 2],\ [-1.2,\ .8]],\ [[.8,\ 0],\ [.8,\ .5],\ [.5,\ .5],\ [.5,\ 0]],\ [[1,\ 1],\ [1,\ 0],\ [-1,\ 0],\ [-1,\ 1]]\}$

```
plot(newhouse,scaling=constrained,thickness=3
);
```



The newhouse can be generated with less typing using the word seq .

```
{seq(map(convert,
   [seq(evalm(reflect &* house[j][i]),i = 1 ..  nops(house[j]))],
   list),j = 1 ..  nops(house))} ;
```

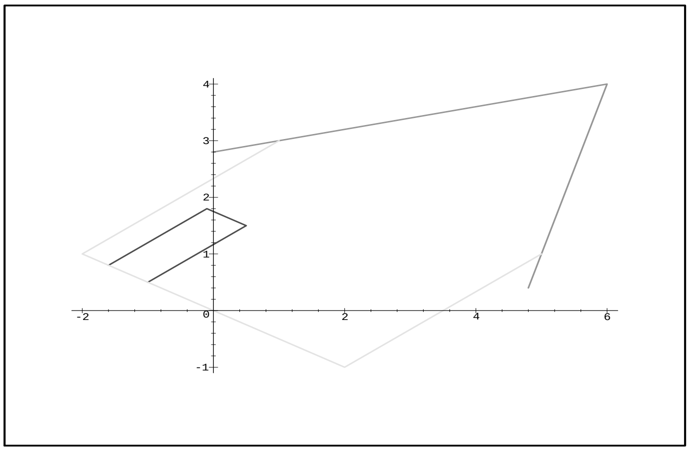
$\{[[1.2,\ .8],\ [0,\ 2],\ [-1.2,\ .8]],\ [[.8,\ 0],\ [.8,\ .5],\ [.5,\ .5],\ [.5,\ 0]],\ [[1,\ 1],\ [1,\ 0],\ [-1,\ 0],\ [-1,\ 1]]$
$\}$

We can define a new word to take any house and any matrix and draw the image of house under the matrix.

```
showit :=
proc(tr, house)
local j,i;
plot({seq(map(convert,
[seq(evalm(tr &* house[j][i]),i = 1 ..  nops(house[j]))],
```

```
list),j = 1 ..  nops(house))},scaling = constrained,thickness=3);
end:
```

```
showit( matrix([[2,3],[-1,2]]),house);
```



Problem: Construct matrices which transform the house to the other 4 newhouses shown on the cover of your text.

Hint: A matrix is completely determined by where it takes (1,0) and (0,1). So for example, the matrix reflect takes (1,0) to (-1,0) and (0,1) to (0,1). The

2.1 An animation problem.

We can imagine the view of the house shown in the upper left hand corner being distorted gradually into the view in the upper right hand corner. This "movie" can be simulated by making series of "frames" or intermediate images starting with the original house and ending with the flipped house. The intermediate images can be constructed by multiplying a matrix in between the identity matrix and the reflection matrix reflect. Then this list of plots is displayed in sequence to get the animation.

```
moveit := proc(tr,house)
local id,movie,t,inter;
id := matrix([[1,0],[0,1]]);
movie := NULL:
for t from 0 by 1/8 to 1 do
inter := evalm(t*tr + (1-t)*id);
movie := movie,showit(inter,house)
od;
plots[display]([movie],insequence = true )
end:
```

So the distortion of the upper left hand corner to the upper right hand corner looks like this –

```
moveit( reflect,house);
```

Problem. What happens in the middle of this distortion?

Problem. Find the distortions from the upper left hand corner to the other 4 newhouses. Find a natural classification of the newhouses into two classes.

3 Problems with Inverses.

3.1 Bidiagonal and tridiagonal matrices of ones.

```
with(linalg);
```

[*BlockDiagonal*, *GramSchmidt*, *JordanBlock*, *LUdecomp*, *QRdecomp*, *Wronskian*, *addcol*, *addrow*, *adj*, *adjoint*, *angle*, *augment*, *backsub*, *band*, *basis*, *bezout*, *blockmatrix*, *charmat*, *charpoly*, *cholesky*, *col*, *coldim*, *colspace*, *colspan*, *companion*, *concat*, *cond*, *copyinto*, *crossprod*, *curl*, *definite*, *delcols*, *delrows*, *det*, *diag*, *diverge*, *dotprod*, *eigenvals*, *eigenvalues*, *eigenvectors*, *eigenvects*, *entermatrix*, *equal*, *exponential*, *extend*, *ffgausselim*, *fibonacci*, *forwardsub*, *frobenius*, *gausselim*, *gaussjord*, *geneqns*, *genmatrix*, *grad*, *hadamard*, *hermite*, *hessian*, *hilbert*, *htranspose*, *ihermite*, *indexfunc*, *innerprod*, *intbasis*, *inverse*, *ismith*, *issimilar*, *iszero*, *jacobian*, *jordan*, *kernel*, *laplacian*, *leastsqrs*, *linsolve*, *matadd*, *matrix*, *minor*, *minpoly*, *mulcol*, *mulrow*, *multiply*, *norm*, *normalize*, *nullspace*, *orthog*, *permanent*, *pivot*, *potential*, *randmatrix*, *randvector*, *rank*, *ratform*, *row*, *rowdim*, *rowspace*, *rowspan*, *rref*, *scalarmul*, *singularvals*, *smith*, *stack*, *submatrix*, *subvector*, *sumbasis*, *swapcol*, *swaprow*, *sylvester*, *toeplitz*, *trace*, *transpose*, *vandermonde*, *vecpotent*, *vectdim*, *vector*, *wronskian*]

Inverses of bidiagonal matrices of ones. The bidiagonal matrix of ones is n by n and has ones down the diagonal and the superdiagonal. All other entries are 0. Here is a word to generate it.

```
bidiag := proc(n)
local B, i, j;
B := matrix(n,n);
for i from 1 to n do for j from 1 to n do
if j = 1+i or j = i then B[i,j] :=1 else B[i,j] := 0 fi od od;
evalm(B);
end:
```

```
bidiag (7);
```

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

```
inverse(bidiag(8));
```

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

Problem: Is every bidiagonal matrix of ones invertible? Prove or find a counterexample. Tridiagonal matrix of ones are defined in a similar manner.

```
tridiag := proc(n)
local B, i, j;
B := matrix(n,n);
for i from 1 to n do for j from 1 to n do
if j = i -1 or j = 1+i or j = i then B[i,j] :=1 else B[i,j] :=
0 fi od od;
evalm(B);
end:
```

```
tridiag(3);
```

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

```
inverse(tridiag(6));
```

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

Problem. Which tridiagonal matrices of ones are invertible?

3.2 Hilbert matrices.

These matrices can be used to show that one must exercise care when using finite precision arithmetic to perform matrix operations. The results may be way off.

```
h6 :=hilbert(6);
```

$$h6 := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} \\ \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} & \frac{1}{11} \end{bmatrix}$$

```
ih6:=inverse(h6);
```

$$ih6 := \begin{bmatrix} 36 & -630 & 3360 & -7560 & 7560 & -2772 \\ -630 & 14700 & -88200 & 211680 & -220500 & 83160 \\ 3360 & -88200 & 564480 & -1411200 & 1512000 & -582120 \\ -7560 & 211680 & -1411200 & 3628800 & -3969000 & 1552320 \\ 7560 & -220500 & 1512000 & -3969000 & 4410000 & -1746360 \\ -2772 & 83160 & -582120 & 1552320 & -1746360 & 698544 \end{bmatrix}$$

```
Digits := 6;
```

$$Digits := 6$$

```
eh6 := evalf(evalm(h6));
```

$$eh6 := \begin{bmatrix} 1. & .500000 & .333333 & .250000 & .200000 & .166667 \\ .500000 & .333333 & .250000 & .200000 & .166667 & .142857 \\ .333333 & .250000 & .200000 & .166667 & .142857 & .125000 \\ .250000 & .200000 & .166667 & .142857 & .125000 & .111111 \\ .200000 & .166667 & .142857 & .125000 & .111111 & .100000 \\ .166667 & .142857 & .125000 & .111111 & .100000 & .0909091 \end{bmatrix}$$

```
ieh6 :=inverse(eh6);
```

$$ieh6 := \begin{bmatrix} 11.0599 & 58.5632 & -1203.88 & 4144.57 & -5226.01 & 2226.06 \\ 58.5632 & -4338.50 & 38101.0 & -112425. & 133701. & -55342.0 \\ -1203.88 & 38101.0 & -273828. & 740738. & -840337. & 337873. \\ 4144.57 & -112425. & 740738. & -.189645\,10^7 & .207174\,10^7 & -810478. \\ -5226.01 & 133701. & -840337. & .207174\,10^7 & -.219506\,10^7 & 837382. \\ 2226.06 & -55342.0 & 337873. & -810478. & 837382. & -312216. \end{bmatrix}$$

```
map(evalm,[eh6&*ieh6,h6&*ih6]);
```

$$\left[\begin{bmatrix} .999 & -.19 & 1.3 & -1. & -1. & -1.1 \\ 0 & .91 & .9 & 1. & 0 & -.2 \\ 0 & -.05 & 1.5 & 0 & 0 & .1 \\ .001 & -.10 & .5 & 2.0 & .4 & .8 \\ -.001 & .02 & .1 & .2 & 1.2 & -.2 \\ -.001 & -.08 & -.1 & .2 & -.4 & 1.0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}\right]$$

```
Digits := 10;
```

$$Digits := 10$$

```
eh6 := evalf(evalm(h6));
```

eh6 :=
[1., .5000000000 , .3333333333 , .2500000000 , .2000000000 , .1666666667]
[.5000000000 , .3333333333 , .2500000000 , .2000000000 , .1666666667 , .1428571429]
[.3333333333 , .2500000000 , .2000000000 , .1666666667 , .1428571429 , .1250000000]
[.2500000000 , .2000000000 , .1666666667 , .1428571429 , .1250000000 , .1111111111]
[.2000000000 , .1666666667 , .1428571429 , .1250000000 , .1111111111 , .1000000000]
[.1666666667 , .1428571429 , .1250000000 , .1111111111 , .1000000000 ,
.09090909091]

```
ieh6 := inverse(eh6);
```

ieh6 :=
[35.99835642 , −629.9538302 , 3359.692089 , −7559.208458 , 7559.134650 ,
−2771.661717]
[−629.9538302 , 14698.70830 , −88191.40873 , 211657.9574 , −220475.9375 ,
83150.60435]
[3359.692089 , −88191.40873 , 564422.9609 , $-.1411053845\ 10^7$, $.1511840609\ 10^7$,
−582057.8112]
[−7559.208458 , 211657.9574 , $-.1411053845\ 10^7$, $.3628425849\ 10^7$, $-.3968592256\ 10^7$,
$.1552161002\ 10^7$]
[7559.134650 , −220475.9375 , $.1511840609\ 10^7$, $-.3968592256\ 10^7$, $.4409555888\ 10^7$,
$-.1746186894\ 10^7$]
[−2771.661717 , 83150.60434 , −582057.8111 , $.1552161002\ 10^7$, $-.1746186894\ 10^7$,
698476.5498]

```
evalm(eh6&*ieh6);
```

$$\begin{bmatrix} .9999994 & -.00001 & -.00010 & .0001 & -.0001 & .0001 \\ -.5\,10^{-6} & 1.00000 & 0 & 0 & .0002 & -.00001 \\ -.3\,10^{-6} & -.00002 & 1.00001 & .0001 & 0 & .00023 \\ 0 & -.3\,10^{-5} & -.00002 & 1.0000 & .0001 & .00003 \\ 0 & -.6\,10^{-5} & .00009 & -.0001 & 1.0000 & .00008 \\ 0 & -.1\,10^{-5} & .00005 & 0 & 0 & 1.00006 \end{bmatrix}$$

This is much better, but now check out hilbert(10).

```
evalm(evalf(hilbert(10))&*inverse(evalf(hilbe
rt(10))));
```

$$\begin{bmatrix} 1.0024 & .10 & -1.1 & 4. & 0 & 30. & 180. & -80. & -50. & -29. \\ .0005 & 1.04 & -.7 & -6. & 90. & 40. & 140. & -10. & -150. & 0 \\ .0005 & .05 & .1 & 3. & -10. & 20. & 90. & 30. & -40. & -4. \\ .0004 & .09 & -.8 & 2. & 50. & -10. & 70. & -110. & 50. & 2. \\ 0 & .02 & -.4 & 3. & 10. & 80. & 160. & 20. & -50. & -8. \\ .0002 & 0 & .3 & 13. & 40. & 20. & 180. & -50. & -30. & 11. \\ .0007 & .03 & .8 & -7. & 30. & 70. & -10. & 50. & -10. & 14. \\ .0002 & .03 & -.3 & 11. & -40. & 80. & 70. & -50. & 20. & 13. \\ 0 & .01 & -1.0 & -5. & -9. & 50. & 90. & 30. & -90. & -6. \\ .0004 & .02 & .2 & -5. & -8. & -40. & 60. & -30. & -140. & 19. \end{bmatrix}$$

This is not even close to the identity 10by10 matrix.
Problem: Show that the hilbert n by n matrix is invertible for each n.

4 Tridiagonal matrices come up.

4.1 Heating a Metal Rod

Problem. The temperature at the ends of a 4 foot metal rod is being maintained at 100 and 150 degrees respectively. Initially, the rod is uniformly 0 degrees everywhere. Three thermometers spaced at 1 foot intervals are consulted at 1 minute intervals and the temperatures recorded.

```
                       x1                 x2                  x3
  ==============================================
 100               0                   0                     0
                 150
```

The person in charge of recording the temperatures notes that the temp at x2 is 90 percent of the average of the temps at x1, x2, and x3 the previous minute, the temp at x1 is 90 percent of the average of 100 and the temps at x1 and x2 the previous minute, and the temp at x3 is 90 percent of the average of 150 and the temps at x2 and x3 the previous minute. What will the temperature readings be in 100 minutes? Will this thing settle down?

Solution:

We have a tridiagonal system of equations!

```
y1 =  .9 (100 + x1 + x2)/3                 ([100]      [1 1 0] [x1])
y2 =  .9 (x1 + x2 + x3)/3      =   .9/3 ([   0 ]  + [1 1 1] [x2])
y3 =  .9 (x2 + x3 + 150)/3                 ([150]      [0 1 1] [x3])
```

We want to track the sequence of temperature readings t0, t1, t2, t3, ... where t0 is any initial set of temperatures and the next set is obtained from the last set by the equation

```
t(n+1) =  .9/3 (etmps  + A tn),  where etmps = (   100, 0,  150) and
       A = [[1,1,0],[1,1,1],[0,1,1]].
```

Load in the linear algebra package.

```
restart;with(linalg);
```

```
Warning, new definition for norm

Warning, new definition for trace
```

[*BlockDiagonal*, *GramSchmidt*, *JordanBlock*, *LUdecomp*, *QRdecomp*, *Wronskian*, *addcol*, *addrow*, *adj*, *adjoint*, *angle*, *augment*, *backsub*, *band*, *basis*, *bezout*, *blockmatrix*, *charmat*, *charpoly*, *cholesky*, *col*, *coldim*, *colspace*, *colspan*, *companion*, *concat*, *cond*, *copyinto*, *crossprod*, *curl*, *definite*, *delcols*, *delrows*, *det*, *diag*, *diverge*, *dotprod*, *eigenvals*, *eigenvalues*, *eigenvectors*, *eigenvects*, *entermatrix*, *equal*, *exponential*, *extend*, *ffgausselim*, *fibonacci*, *forwardsub*, *frobenius*, *gausselim*, *gaussjord*, *geneqns*, *genmatrix*, *grad*, *hadamard*, *hermite*, *hessian*, *hilbert*, *htranspose*, *ihermite*, *indexfunc*, *innerprod*, *intbasis*, *inverse*, *ismith*, *issimilar*, *iszero*, *jacobian*, *jordan*, *kernel*, *laplacian*, *leastsqrs*, *linsolve*, *matadd*, *matrix*, *minor*, *minpoly*, *mulcol*, *mulrow*, *multiply*, *norm*, *normalize*, *nullspace*, *orthog*, *permanent*, *pivot*, *potential*, *randmatrix*, *randvector*, *rank*, *ratform*, *row*, *rowdim*, *rowspace*, *rowspan*, *rref*, *scalarmul*, *singularvals*, *smith*, *stack*, *submatrix*, *subvector*, *sumbasis*, *swapcol*, *swaprow*, *sylvester*, *toeplitz*, *trace*, *transpose*, *vandermonde*, *vecpotent*, *vectdim*, *vector*, *wronskian*]

We define a word to build tridiagonal matrices easily.

```
tridiag := proc(a,b,c,n)
local B, i, j;
B := matrix(n,n);
for i from 1 to n do for j from 1 to n do
if j = i -1 then B[i,j] := a
elif j = 1+i then B[i,j] := c
elif j = i then B[i,j] :=b else B[i,j] := 0 fi od od;
evalm(B);
end:
```

To set up the particular problem we started with, let

```
n := 3:
A :=tridiag(1.,1.,1.,n);
etmps := vector([ 100,seq( 0,i=1..n-2),150]);
```

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

$$etmps := [100,\ 0,\ 150]$$

We can start with an initially cold rod, where the thermometers read 0 degrees accross the board.

```
step := vector([0,0,0]);
```

$$step := [0,\ 0,\ 0]$$

```
step := evalm(.9/3*(etmps+A&*step));
```

$$step := [30.00000000,\ 0,\ 45.00000000]$$

```
for i from 1 to 15 do step :=
evalm(.9/3*(etmps+A&*step)) ;
print(evalm(step));
od :
```

$$[39.00000000,\ 22.50000000,\ 58.50000000]$$

$$[48.45000000,\ 36.00000000,\ 69.30000000]$$

$$[55.33500000,\ 46.12500000,\ 76.59000000]$$

$$[60.43800000,\ 53.41500000,\ 81.81450000]$$

$$[64.15590000,\ 58.70025000,\ 85.56885000]$$

$$[66.85684500,\ 62.52750000,\ 88.28073000]$$

$$[68.81530350,\ 65.29952250,\ 90.24246900]$$

$$[70.23444780,\ 67.30718850,\ 91.66259745]$$

$$[71.26249089,\ 68.76127014,\ 92.69093580]$$

[72.00712830, 69.81440904, 93.43566177]

[72.54646119, 70.57715973, 93.97502124]

[72.93708627, 71.12959263, 94.36565430]

[73.22000367, 71.52969996, 94.64857407]

[73.42491108, 71.81948331, 94.85348220]

[73.57331832, 72.02936298, 95.00188965]

So we can see that there is some apparent convergence of the temperature distribution to a stable configuration. We can try to represent this graphically some way. For example, a simple movie showing a plot of the temperature curve as it changes over time is easily done and can be useful.

All we have to do here is keep a list of plots and display them later. First a little word (unnecessary but it improves readibality).

```
plotstep := proc(step,n)
local i;
plot([[0,0],[0,100],
seq([10*i,step[i]],i = 1 ..  n),
[40,150]])
end:
```

```
n := 3:
A :=tridiag(1.,1.,1.,n);
etmps := vector([ 100,seq( 0,i=1..n-2),150]);
step := vector([0,0,0]);
movie := plotstep(step,n):
step := evalm(.9/3*(etmps+A&*step));
movie := movie,plotstep(step,n):
for i from 1 to 30 do step := evalm(.9/3*(etmps+A&*step)) ;
movie := movie, plotstep(step,n):
print(evalm(step));
od :
```

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

$$etmps := [100,\ 0,\ 150]$$

$$step := [0,\ 0,\ 0]$$

$$step := [30.00000000,\ 0,\ 45.00000000]$$

[39.00000000, 22.50000000, 58.50000000]

[48.45000000, 36.00000000, 69.30000000]

[55.33500000, 46.12500000, 76.59000000]

[60.43800000, 53.41500000, 81.81450000]

[64.15590000, 58.70025000, 85.56885000]

[66.85684500, 62.52750000, 88.28073000]

[68.81530350, 65.29952250, 90.24246900]

[70.23444780, 67.30718850, 91.66259745]

[71.26249089, 68.76127014, 92.69093580]

[72.00712830, 69.81440904, 93.43566177]

[72.54646119, 70.57715973, 93.97502124]

[72.93708627, 71.12959263, 94.36565430]

[73.22000367, 71.52969996, 94.64857407]

[73.42491108, 71.81948331, 94.85348220]

[73.57331832, 72.02936298, 95.00188965]

[73.68080439, 72.18137130, 95.10937578]

[73.75865271, 72.29146545, 95.18722413]

[73.81503546, 72.37120269, 95.24360688]

[73.85587146, 72.42895353, 95.28444288]

[73.88544750, 72.47078037, 95.31401892]

[73.90686837, 72.50107404, 95.33543979]

[73.92238272, 72.52301466, 95.35095414]

[73.93361922, 72.53890545, 95.36219064]

[73.94175741, 72.55041459, 95.37032883]

[73.94765160, 72.55875024, 95.37622302]

[73.95192054, 72.56478744, 95.38049199]

[73.95501240, 72.56916000, 95.38358382]

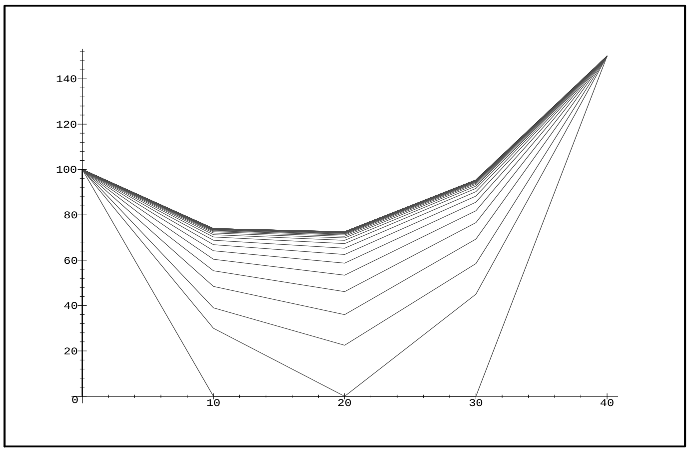
[73.95725172, 72.57232686, 95.38582314]

[73.95887358, 72.57462051, 95.38744500]

[73.96004823, 72.57628173, 95.38861965]

We can look at all of these plots together, and get a good sense of the convergence.

```
plots[display]([movie]);
```



Or we can look at them in sequence, to get another feeling the convergence of the approximation toward a limiting set of temp readings.

```
plots[display]([movie],insequence=true);
```

So, on the basis of our calculation and picture drawing, we could say that the three thermometers will read about 74, 72.5, and 95.4 degrees from left to right after 30 minutes or so.

Problem 1. Does this end temperature distribution depend on the initial distribution? Experiment, by changing the first step from [0,0,0] to some other things like [10,20,10] and running the cells again. Better still, copy the input cells down to here and make the changes here. Discuss the results.

Problem 2. Suppose there is no heat loss and the 90 percent becomes 100 percent. What happens? Think about it and try to predict what will happen. Then copy down the appropriate input cells and make appropriate changes, and experiment. Discuss. Was your guess supported?

Problem 3. Suppose the rod is 6 feet long, and there are 5 thermometers. With end temps of 100 and 150, what is the minimum stable temp of the rod?

4.2 Back to the original problem

Using Eigenvalues and eigenvectors here can help us analyse the original problem.

First compute them. We could do this directly, but for the moment we can do it the fast way, using eigenvects from the linalg package.

```
eigs := eigenvects(A);
```

$$eigs := [1.000000000,\ 1,\ \{[-.7071067808,\ -.14\,10^{-9},\ .7071067812]\}],\\ [2.414213564,\ 1,\ \{[.4999999998,\ .7071067814,\ .5000000002]\}],\\ [-.4142135626,\ 1,\ \{[.5000000000,\ -.7071067813,\ .5000000001]\}]$$

So there are three eigenvalues:

```
eigs[1][1], eigs[2][1],eigs[3][1];
```

$$1.000000000,\ 2.414213564,\ -.4142135626$$

The eigenvectors belonging to the first eigenvalue are all scalar multiples of the 1 eigenvector that is given.

```
eigs[1][3][1];
```

$$[-.7071067808,\ -.14\,10^{-9},\ .7071067812]$$

In general, each eigenvalue comes with an integer specifying the dimension of the space of eigenvectors belonging to that eigenvalue, and a set of basis vectors for that eigenspace.

Going back to the original analysis, we want to track the sequence of temperature readings t0, t1, t2, t3, ... where t0 is any initial set of temperatures and the next set is obtained from the last set by the equation

t(n+1) = .9/3 (etmps + A tn), where etmps = (100, 0, 150) and

A = [[1,1,0],[1,1,1],[0,1,1]].

For the moment, let r = .9/3, and simply compute the first few terms of the sequence t0, t1, t2, ...

```
t1 = r(etmps + A t0)
t2 = r(etmps + A t1) = r(etmps + A r(emtps + A t0))
      =r etmps + r\symbol{94}2 A etmps + r\symbol{94}2 A\symbol{94}2
t0.
t3 =  r etmps + r\symbol{94}2  A etmps + r\symbol{94}3 A\symbol{94}2
etmps + r\symbol{94}3 A\symbol{94}3   t0.
```

```
r :='r':
```

```
f := t0 -> r*( etmps + A*t0);
```

$$f := t0 \to r\,(etmps + A\,t0)$$

We can see that the nth set tn of temp readings is a partial sum of a geometric series of vectors plus a final term. For example, the set t6 looks like :

```
expand( (f@@6)(t0));
```

$$r\,etmps + A\,r^2\,etmps + A^2\,r^3\,etmps + A^3\,r^4\,etmps + A^4\,r^5\,etmps + A^5\,r^6\,etmps + A^6\,r^6\,t0$$

So, write etmps as a linear combination of the eigenvectors and sum the geometric series of scalar multiples that pops out for each of the eigenvectors. In each case,

(1 + r*lamda + (r*lamda)^2 + ...) = r *1/(1-r*lamda)

```
eqn :=
evalm(a*eigs[1][3][1]+b*eigs[2][3][1]+c*eigs[3][3][1]-etmps);
```

$$eqn := [-.7071067808\ a + .4999999998\ b + .5000000000\ c - 100,\\ -.14\,10^{-9}\ a + .7071067814\ b - .7071067813\ c,\\ .7071067812\ a + .5000000002\ b + .5000000001\ c - 150]$$

```
sol :=
solve({eqn[1],eqn[2],eqn[3]},{a,b,c});
```

$$sol := \{b = 125.0000000,\ c = 125.0000000,\ a = 35.35533902\}$$

Now by summing three geometric series, we can compute the limiting set of temperature readings.

```
assign (sol):  r := .3:
```

```
evalm(a*r*1/(1-r*eigs[1][1])*eigs[1][3][1]+b*r*1/(1-r*eigs[2][1])*eigs
[2][3][1]+c*r*1/(1-r*eigs[3][1])*eigs[3][3][1]);
```

$$[73.96313376,\ 72.58064538,\ 95.39170523]$$

This compares nicely with the readings after 30 minutes.

Problem 4: Use eigenvalue/eigenvector analysis to investigate Problems 1,2, and/or 3.

5 Flourbarrel Problems.

Problem: You have three barrels containing at first 25, 17, and 35 pounds of flour respectively. Every hour, the flour from barrel 1 is divided into the proportions 6:2:9, the flour from barrel 2 is divided into the proportions 11:7:3, and the flour from barrel 3 is divided into proportions 14:2:21. These are then redistributed into barrels 1,2, and 3. How many pounds in the first barrel after 100 hours?

Solution 1. Brute force. Set up a system of equations.

let xn, yn, and zn be the number of pounds of flour in barrel 1, 2, and 3 respectively after n hours.

$$\text{So } x_{n+1} = \frac{6\,x_n}{17} + \frac{11\,y_n}{21} + \frac{14\,z_n}{37}\text{pounds}$$

$$y_{n+1} = \frac{2\,x_n}{17} + \frac{7\,y_n}{21} + \frac{2\,z_n}{37}\text{ pounds}$$

$$z_{n+1} = \frac{9\,x_n}{17} + \frac{3\,y_n}{21} + \frac{21\,z_n}{37}\text{ pounds}$$

A matrix equation!

What this equation says is that if A is the matrix of coefficients then after n hours there will be respectively

$A^{n_{25,\,17,\,35}}$ pounds of flour in the barrels.

```
with(linalg): with(plots):
```

```
Warning, new definition for norm


Warning, new definition for trace
```

```
barrels := vector([25,17,35]);
```

$$barrels := [25,\ 17,\ 35]$$

```
distribute := evalf(matrix(3,3,[6/17,11/21,14/37,2/17,7/21,2/37,9/17,3/21,21/37]));
```

$$distribute := \begin{bmatrix} .3529411765 & .5238095238 & .3783783784 \\ .1176470588 & .3333333333 & .05405405405 \\ .5294117647 & .1428571429 & .5675675676 \end{bmatrix}$$

Checking to see what happens in the first 10 hours.

```
seq( evalf(evalm(distribute^i
&*barrels)),i=1..10);
```

[30.97153456, 10.49973503, 35.52873042], [29.87429446, 9.064093537, 38.06161201],
[29.69341819, 8.593371822, 38.71320999], [29.62956158, 8.450406532, 38.92003190],
[29.61039431, 8.406418458, 38.98318726], [29.60448463, 8.392914595, 39.00260078],
[29.60267107, 8.388767428, 39.00856151], [29.60211408, 8.387493885, 39.01039205],
[29.60194304, 8.387102790, 39.01095419], [29.60189050, 8.386982688, 39.01112681]

Seems to be converging. Check what happens at the hundredth hour.

```
evalm(distribute^100&*barrels);
```

[29.60186723, 8.386929459, 39.01120335]

We can see that the convergence is quite rapid.

So after 100 hours, there will be 29.6 pounds of flour in the first barrel.

Solution 2. Find three independent eigenvectors x1, x2, and x3 with eigenvalues l1, l2, and l3 so that (25,17,35) can be written as a linear combination of them (25,17,35) is $a\,x_1 + b\,x_2 + c\,x_3$

Then

$$A\,(25,\,17,\,35) = A\,(a\,x_1 + b\,x_2 + c\,x_3)$$

$$A\,(a\,x_1 + b\,x_2 + c\,x_3) = a\,A\,x_1 + b\,A\,x_2 + c\,A\,x_3$$

$$a\,A\,x_1 + b\,A\,x_2 + c\,A\,x_3 = a\,l_1\,x_1 + b\,l_2\,x_2 + c\,l_3\,x_3,$$

and in general

$A^n\,(25,\,17,\,35) = a\,{l_1}^n\,x_1 + b\,{l_2}^n\,x_2 + c\,{l_3}^n\,x_3.$

We can do this the fast way, using eigenvects from the linalg package.

```
vecs := eigenvects( distribute );
```

$$\begin{aligned} vecs := &[1.000000002,\ 1,\ \{[-.6044692561,\ -.1712608526,\ -.7966076239]\}],\\ &[-.05324953854,\ 1,\ \{[.7778754590,\ -.1487613558,\ -.6291141042]\}],\\ &[.3070916160,\ 1,\ \{[.3299800257,\ .7545004714,\ -1.084480496]\}]\end{aligned}$$

So there are three eigenvalues and three eigenvectors. Find a,b, and c first.

```
eqs :=
evalm(a*vecs[1][3][1]+b*vecs[2][3][1]+c*vecs[3][3][1]-barrels);
```

$$\begin{aligned} eqs := [&-.6044692561\,a + .7778754590\,b + .3299800257\,c - 25,\\ &-.1712608526\,a - .1487613558\,b + .7545004714\,c - 17,\\ &-.7966076239\,a - .6291141042\,b - 1.084480496\,c - 35]\end{aligned}$$

```
sol :=
solve({eqs[1],eqs[2],eqs[3]},{a,b,c});
```

$$sol := \{b = -9.928136969,\ a = -48.97166708,\ c = 9.458108639\}$$

The limiting value of the distribution is $a\,x_1$ since the other eigenvalues are less than 1 in absolute value.

```
evalm(rhs(sol[3])*vecs[1][3][1]);
```

$$[-5.717135893,\ -1.619803749,\ -7.534401450]$$

We can use the eigenvalues and eigenvectors to compute the distribution after 100 hours.

$A^{100}\,(25,\,17,\,35) = a\,{l_1}^{100}\,x_1 + b\,{l_2}^{100}\,x_2 + c\,{l_3}^{100}\,x_3$

```
evalm(
rhs(sol[3])*vecs[1][3][1]+rhs(sol[1])*vecs[2][1]^100*vecs[2][3][1]+
rhs(sol[2])*vecs[3][1]^100*vecs[3][3][1] );
```

$$[-5.717135893,\ -1.619803749,\ -7.534401450]$$

We can check whether we have gained or lost any flour by summing the amounts in the barrels after 100 hours.

```
sum("[k],k=1..3),sum(barrels[l],l=1..3);
```

$$-14.87134109,\ 77$$

Problem 1: Calculate the eigenvalues and eigenvectors of A without using the word eigenvects.

Problem 2: It turns out there is a leak in barrel 1, in which 3 percent of the flour in the barrel is lost each hour. How much in each barrel after 100 hours now? Solve by brute force and by eigenvalues/vectors.

Problem 3: It turns out there is a leak in barrel 1, in which 1 ounce of the flour in the barrel is lost each hour. How much in each barrel after 100 hours now? Can you use the eigen/eigen method here?

Problem 4: Define a word flourbarrels(A,beg,n) which takes a matrix A, a beginning point distribution, and a positive integer n and returns the distribution after n hours. Test it out.

5.1 Visualising the change in distribution over time.

Here is one way of tracking the change in distribution over time. Just plot the track: that is, you start with a certain distribution u and a 'movie' showing the distribution in 'barrelspace'.

```
trackit := proc(A,u,n)
local track,x,film,i;
track := u;
x := u;
if nops(u)=3 then
film := pointplot3d({ u });
for i to n do
x := evalm(A &* x);
track := track,[x[1],x[2],x[3]];
film := film,pointplot3d({track})
od;
print(track);
display([film],insequence = true,axes = box)
else
film := plot({u},style=point,symbol=cross,color=black);
for i to n do
x := evalm(A &* x);
track := track,[x[1],x[2] ];
film := film,
plot({track},style=point,symbol=cross,color=black)
od;
print(track);
display([film],insequence = true,axes = box)
fi;
end:
```

```
trackit(
distribute,convert(barrels,list),10);
```

[25, 17, 35], [30.97153456, 10.49973503, 35.52873042],
[29.87429446, 9.064093538, 38.06161202], [29.69341819, 8.593371822, 38.71321001],
[29.62956158, 8.450406536, 38.92003191], [29.61039430, 8.406418459, 38.98318725],
[29.60448463, 8.392914596, 39.00260078], [29.60267107, 8.388767433, 39.00856150],
[29.60211406, 8.387493887, 39.01039205], [29.60194302, 8.387102788, 39.01095418],
[29.60189050, 8.386982686, 39.01112680]

An alternative would be too draw three barrels showing how full they are of flour at the end of each hour. For a drawing challenge, here is

Problem 5. Make a 'movie' of the barrels, showing the distribution of flour in the barrels changing over time.

```
?pointplot3d ;
```

$$printlevel := 1$$

6 Gaussian Elimination with good record keeping.

A elimination matrix is a square matrix with 1's down the diagonal and 0's elsewhere except at one entry below the diagonal. Here is a word to generate an elimination matrix.

```
with(linalg);
```

[*BlockDiagonal*, *GramSchmidt*, *JordanBlock*, *LUdecomp*, *QRdecomp*, *Wronskian*, *addcol*, *addrow*, *adj*, *adjoint*, *angle*, *augment*, *backsub*, *band*, *basis*, *bezout*, *blockmatrix*, *charmat*, *charpoly*, *cholesky*, *col*, *coldim*, *colspace*, *colspan*, *companion*, *concat*, *cond*, *copyinto*, *crossprod*, *curl*, *definite*, *delcols*, *delrows*, *det*, *diag*, *diverge*, *dotprod*, *eigenvals*, *eigenvalues*, *eigenvectors*, *eigenvects*, *entermatrix*, *equal*, *exponential*, *extend*, *ffgausselim*, *fibonacci*, *forwardsub*, *frobenius*, *gausselim*, *gaussjord*, *geneqns*, *genmatrix*, *grad*, *hadamard*, *hermite*, *hessian*, *hilbert*, *htranspose*, *ihermite*, *indexfunc*, *innerprod*, *intbasis*, *inverse*, *ismith*, *issimilar*, *iszero*, *jacobian*, *jordan*, *kernel*, *laplacian*, *leastsqrs*, *linsolve*, *matadd*, *matrix*, *minor*, *minpoly*, *mulcol*, *mulrow*, *multiply*, *norm*, *normalize*, *nullspace*, *orthog*, *permanent*, *pivot*, *potential*, *randmatrix*, *randvector*, *rank*, *ratform*, *row*, *rowdim*, *rowspace*, *rowspan*, *rref*, *scalarmul*, *singularvals*, *smith*, *stack*, *submatrix*, *subvector*, *sumbasis*, *swapcol*, *swaprow*, *sylvester*, *toeplitz*, *trace*, *transpose*, *vandermonde*, *vecpotent*, *vectdim*, *vector*, *wronskian*]

```
elim := proc(size,row,col,entry)
local A,i;
A := diag(seq(1,i = 1 ..  size));
A[row,col] := entry;
evalm(A)
end:
```

```
elim(3,3,2,9);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 9 & 1 \end{bmatrix}$$

The LU factorization theorem: If A is a square matrix which can be reduced to upper triangular form without row interchanges so that no pivot is zero, then A can be factored as the product of a lower unit triangular matrix with an upper triangular matrix with nonzero diagonal entries.

Look at an example. Generate a random 3 by 3 matrix.

```
A := randmatrix(3,3);
```

$$A := \begin{bmatrix} -85 & -55 & -37 \\ -35 & 97 & 50 \\ 79 & 56 & 49 \end{bmatrix}$$

Now the first elimination matrix looks like this –

```
e21 := elim(3,2,1,-A[2,1]/A[1,1]);
```

$$e21 := \begin{bmatrix} 1 & 0 & 0 \\ \frac{-7}{17} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiply E21 by A to make the 21 entry 0.

```
U := evalm(e21 &* A);
```

$$U := \begin{bmatrix} -85 & -55 & -37 \\ 0 & \frac{2034}{17} & \frac{1109}{17} \\ 79 & 56 & 49 \end{bmatrix}$$

```
e31 := elim(3,3,1,-U[3,1]/U[1,1]);
```

$$e31 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{79}{85} & 0 & 1 \end{bmatrix}$$

```
U := evalm(e31 &* U);
```

$$U := \begin{bmatrix} -85 & -55 & -37 \\ 0 & \frac{2034}{17} & \frac{1109}{17} \\ 0 & \frac{83}{17} & \frac{1242}{85} \end{bmatrix}$$

```
e32 := elim(3,3,2,-U[3,2]/U[2,2]);
```

$$e32 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{-83}{2034} & 1 \end{bmatrix}$$

```
U := evalm(e32 &* U);
```

$$U := \begin{bmatrix} -85 & -55 & -37 \\ 0 & \frac{2034}{17} & \frac{1109}{17} \\ 0 & 0 & \frac{121529}{10170} \end{bmatrix}$$

So at this point, we have that e32 e31 e21 A = U. To undo the eliminations, we only need to multiply U on the left successively by the inverse of e32, then e31, then e21 to get A back again. Note that the inverse of elimin(size,row,col,entry) is just the elimination matrix elimin(size,row,col,-entry).

```
evalm(elim(5,3,2,entry)&*elim(5,3,2,-entry));
```

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

```
L :=
evalm(inverse(e21)&*inverse(e31)&*inverse(e32));
```

$$L := \begin{bmatrix} 1 & 0 & 0 \\ \frac{7}{17} & 1 & 0 \\ \frac{-79}{85} & \frac{83}{2034} & 1 \end{bmatrix}$$

Note that the product of the elimination matrices in reverse order is just the lower triangular matrix whose entries below the diagonal are the negatives of the multiplier entries of the elimination matrices below the diagonal.

Here is a word to factor a matrix into the product of a lower triangular and upper triangular matrix.

```
lufact := proc(A,‘L‘,‘U‘)
local k,i,j,el;
k := rowdim(A);
U := evalm(A);
L := diag(seq(1,i = 1 .. k));
for i to k-1 do
if U[i,i] = 0 then
ERROR(cat( ‘zero pivot encountered at ‘,i,‘ step ‘))
else
for j from i+1 to k do
el := elim( k,j,i,-U[j,i]/U[i,i] );
L[j,i] := U[j,i]/U[i,i];
U := evalm(el &* U)
od
fi
od;
if U[k,k] = 0 then
ERROR(cat( ‘zero pivot encountered at ‘,i,‘ step ‘)) fi;
NULL:
end:
```

We can check this out on some random 4 by 4 matrices.

```
A := randmatrix(4,4): lufact(A ,L,U):
map(evalm,[A,L,U,L&*U]);
```

$$\left[\begin{bmatrix} 63 & 57 & -59 & 45 \\ -8 & -93 & 92 & 43 \\ -62 & 77 & 66 & 54 \\ -5 & 99 & -61 & -50 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{-8}{63} & 1 & 0 & 0 \\ \frac{-62}{63} & \frac{-2795}{1801} & 1 & 0 \\ \frac{-5}{63} & \frac{-2174}{1801} & \frac{98139}{375740} & 1 \end{bmatrix}, \begin{bmatrix} 63 & 57 & -59 & 45 \\ 0 & \frac{-1801}{21} & \frac{5324}{63} & \frac{341}{7} \\ 0 & 0 & \frac{751480}{5403} & \frac{313169}{1801} \\ 0 & 0 & 0 & \frac{-12415311}{375740} \end{bmatrix}, \begin{bmatrix} 63 & 57 & -59 & 45 \\ -8 & -93 & 92 & 43 \\ -62 & 77 & 66 & 54 \\ -5 & 99 & -61 & -50 \end{bmatrix} \right]$$

After doing a few of these, you begin to wonder what the probability is that a random 4 by 4 matrix cannot be reduced to upper triangular form with a full set of (nonzero) pivots.

Sometimes they do, and in this case we need to interchange a row lower down with what would have been the pivot row if the pivot entry were nonzero. Permutation matrices perform this operation.

Permutation Matrices are square matrices with exactly one 1 in each row and column and 0's elsewhere. They obtained from the identity matrix by permuting its rows. The simplest type is the transposition matrix, which is obtained by interchanging two rows of the identity matrix.

```
interchange := proc(size,row1,row2)
local A,i;
A := diag(seq(1,i = 1 ..  size));
A := swaprow(A,row1,row2)
end:
```

The pa = lu factorization theorem: If a square matrix A is invertible, then a permutation matrix p can be found so that pa can be factored into l u.

Problem. Modify the definition of lufact to get a
plufact('P',A,'L','U').

The key observation here is that an interchange matrix Pkl commutes with an elimination matrix Eij provided l exceeds j and k exceeds i. This means we can move all the interchange matrices to the right as they occur.

Here is a word you can use to check this out.

```
checkit := proc(k,l,i,j)
local eij,pkl;
eij := elim(5,i,j,m);
pkl := interchange(5,k,l);
equal(evalm(eij &* pkl),
evalm(pkl &* eij))
end:
```

```
checkit(2,1,4,3);
```

$$true$$

Now how do we implement the back substitution ?

There are words in linalg to do gaussian elimination with row interchanges on an augmented matrix.

```
A := randmatrix(3,4);
```

$$A := \begin{bmatrix} -12 & -18 & 31 & -26 \\ -62 & 1 & -47 & -91 \\ -47 & -61 & 41 & -58 \end{bmatrix}$$

```
gausselim(A);
```

$$\begin{bmatrix} -12 & -18 & 31 & -26 \\ 0 & 94 & \frac{-1243}{6} & \frac{130}{3} \\ 0 & 0 & \frac{-67093}{1128} & \frac{5563}{141} \end{bmatrix}$$

```
backsub(");
```

$$\left[\frac{131129}{67093}, \frac{-67153}{67093}, \frac{-44504}{67093}\right]$$

This gives the solution to the original system.

7 Gaussian Elimination with row interchanges.

Row operations are left multiplication by elimination matrices and interchange matrices.
A elimination matrix is a square matrix with 1's down the diagonal
and 0's elsewhere except at one entry below the diagonal.
Here is a word to generate an elimination matrix.

```
with(linalg);
```

[*BlockDiagonal*, *GramSchmidt*, *JordanBlock*, *LUdecomp*, *QRdecomp*, *Wronskian*, *addcol*, *addrow*, *adj*, *adjoint*, *angle*, *augment*, *backsub*, *band*, *basis*, *bezout*, *blockmatrix*, *charmat*, *charpoly*, *cholesky*, *col*, *coldim*, *colspace*, *colspan*, *companion*, *concat*, *cond*, *copyinto*, *crossprod*, *curl*, *definite*, *delcols*, *delrows*, *det*, *diag*, *diverge*, *dotprod*, *eigenvals*, *eigenvalues*, *eigenvectors*, *eigenvects*, *entermatrix*, *equal*, *exponential*, *extend*, *ffgausselim*, *fibonacci*, *forwardsub*, *frobenius*, *gausselim*, *gaussjord*, *geneqns*, *genmatrix*, *grad*, *hadamard*, *hermite*, *hessian*, *hilbert*, *htranspose*, *ihermite*, *indexfunc*, *innerprod*, *intbasis*, *inverse*, *ismith*, *issimilar*, *iszero*, *jacobian*, *jordan*, *kernel*, *laplacian*, *leastsqrs*, *linsolve*, *matadd*, *matrix*, *minor*, *minpoly*, *mulcol*, *mulrow*, *multiply*, *norm*, *normalize*, *nullspace*, *orthog*, *permanent*, *pivot*, *potential*, *randmatrix*, *randvector*, *rank*, *ratform*, *row*, *rowdim*, *rowspace*, *rowspan*, *rref*, *scalarmul*, *singularvals*, *smith*, *stack*, *submatrix*, *subvector*, *sumbasis*, *swapcol*, *swaprow*, *sylvester*, *toeplitz*, *trace*, *transpose*, *vandermonde*, *vecpotent*, *vectdim*, *vector*, *wronskian*]

```
elim := proc(size,row,col,entry)
local A,i;
A := diag(seq(1,i = 1 ..  size));
A[row,col] := entry;
evalm(A)
end:
```

```
elim(3,3,2,9);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 9 & 1 \end{bmatrix}$$

<u>The LU factorization theorem:</u> If A is a square matrix which can be reduced to upper triangular form without row interchanges so that no pivot is zero, then A can be factored as the product of a lower unit triangular matrix with an upper triangular matrix with nonzero diagonal entries.
Look at an example. Generate a random 3 by 3 matrix.

```
A := randmatrix(3,3);
```

$$A := \begin{bmatrix} -90 & 53 & -1 \\ 94 & 83 & -86 \\ 23 & -84 & 19 \end{bmatrix}$$

Now the first elimination matrix looks like this –

```
e21 := elim(3,2,1,-A[2,1]/A[1,1]);
```

$$e21 := \begin{bmatrix} 1 & 0 & 0 \\ \frac{47}{45} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiply E21 by A to make the 21 entry 0.

```
U := evalm(e21 &* A);
```

$$U := \begin{bmatrix} -90 & 53 & -1 \\ 0 & \frac{6226}{45} & \frac{-3917}{45} \\ 23 & -84 & 19 \end{bmatrix}$$

```
e31 := elim(3,3,1,-U[3,1]/U[1,1]);
```

$$e31 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{23}{90} & 0 & 1 \end{bmatrix}$$

```
U := evalm(e31 &* U);
```

$$U := \begin{bmatrix} -90 & 53 & -1 \\ 0 & \frac{6226}{45} & \frac{-3917}{45} \\ 0 & \frac{-6341}{90} & \frac{1687}{90} \end{bmatrix}$$

```
e32 := elim(3,3,2,-U[3,2]/U[2,2]);
```

$$e32 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{6341}{12452} & 1 \end{bmatrix}$$

```
U := evalm(e32 &* U);
```

$$U := \begin{bmatrix} -90 & 53 & -1 \\ 0 & \frac{6226}{45} & \frac{-3917}{45} \\ 0 & 0 & \frac{-318543}{12452} \end{bmatrix}$$

So at this point, we have that —— e32 e31 e21 A = U—. To
undo the eliminations, we only need to multiply U on the left successively by the inverse of e32, then e31, then e21 to get A
back again. Note that the inverse of elimin(size,row,col,entry)
is just the elimination matrix elimin(size,row,col,-entry).

```
evalm(elim(5,3,2,entry)&*elim(5,3,2,-entry));
```

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

```
L :=
evalm(inverse(e21)&*inverse(e31)&*inverse(e32));
```

$$L := \begin{bmatrix} 1 & 0 & 0 \\ \frac{-47}{45} & 1 & 0 \\ \frac{-23}{90} & \frac{-6341}{12452} & 1 \end{bmatrix}$$

Note that the product of the elimination matrices in reverse order is
just the lower triangular matrix whose entries below the diagonal
are the negatives of the multiplier entries of the elimination matrices below the diagonal.
Here is a word to factor a matrix into the product of a lower triangular and upper triangular matrix.

```
lufact := proc(A,‘L‘,‘U‘)
local k,i,j,el;
k := rowdim(A);
U := evalm(A);
L := diag(seq(1,i = 1 ..  k));
for i to k-1 do
if U[i,i] = 0 then
ERROR(cat( ‘zero pivot encountered at ‘,i,‘ step ‘))
else
for j from i+1 to k do
el := elim( k,j,i,-U[j,i]/U[i,i] );
L[j,i] := U[j,i]/U[i,i];
U := evalm(el &* U)
od
fi
od;
if U[k,k] = 0 then
ERROR(cat( ‘zero pivot encountered at ‘,i,‘ step ‘)) fi;
NULL:
end:
```

We can check this out on some random 4 by 4 matrices.

```
A := randmatrix(4,4):  lufact(A ,s,t):
B := stack(row(A,1),row(A,2),evalm(row(A,1)-3*row(A,2)));
```

$$B := \begin{bmatrix} -50 & 88 & -53 & 85 \\ 49 & 78 & 17 & 72 \\ -197 & -146 & -104 & -131 \end{bmatrix}$$

```
lufact(B,s,t);
```

```
Error, (in lufact) zero pivot encountered at 3 step
```

```
map(evalm,[B,s,t,L&*U]);
```

$$\left[\begin{bmatrix} -50 & 88 & -53 & 85 \\ 49 & 78 & 17 & 72 \\ -197 & -146 & -104 & -131 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ \frac{-49}{50} & 1 & 0 \\ \frac{197}{50} & -3 & 1 \end{bmatrix}, \begin{bmatrix} -50 & 88 & -53 & 85 \\ 0 & \frac{4106}{25} & \frac{-1747}{50} & \frac{1553}{10} \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} -90 & 53 & -1 \\ 94 & 83 & -86 \\ 23 & -84 & 19 \end{bmatrix}\right]$$

After doing a few of these, you begin to wonder what the probability is that a random 4 by 4 matrix cannot be reduced to upper triangular form with a full set of (nonzero) pivots.
Sometimes they do, and in this case we need to interchange a row lower down with what would have been the pivot row if the pivot entry were nonzero. Permutation matrices perform this operation.

Permutation Matrices are square matrices with exactly one 1 in each row and column and 0's elsewhere. They obtained from the identity
matrix by permuting its rows. The simplest type is the transposition matrix, which is obtained by interchanging two rows of the identity matrix.

```
interchange := proc(size,row1,row2)
local A,i;
A := diag(seq(1,i = 1 ..  size));
A := swaprow(A,row1,row2)
end:

evalm(e1[3]);
```

$$e1_3$$

```
p23 := interchange(5,3,2);
```

$$p23 := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

```
evalm( e1[3]&*e1[2]- e1[2] );
```

$$(e1_3 \mathbin{\&*} e1_2) - e1_2$$

If a zero pivot occurs, then we want to search down the pivot column until a nonzero entry (to our liking) is found and do a row interchange p on U to bring that entry up to the pivot row. We can, making use of a clever trick, arrange things so that the production of all the row interchanges needed can be done at the beginning. That's the statement of
<u>The pa = lu factorization theorem:</u> If a square matrix A is invertible, then a permutation matrix P can be found so that PA can be factored into l u.

Indication of proof: In the reduction of A to upper triangular form, we multiply on the left by elimination matrices and interchange matrices. So suppose we are at the ith stage and a row interchange pij is needed. A has been reduced to U = E Pi A where E is a product of elimination matrices with nonzero entries below the diagonal and to the left of the ith column, and Pi is product of the row interchanges needed (if any) needed up to the ith step. The trick is to notice that pij E = (pij E pij) pij because pij pij = Identity matrix, and pij E pij is a lower triangular matrix that looks just like E except that the multipliers in the ith and jth rows have been interchanged. So

write pijU = pij E Pi A = (pij E pij) (pij Pi) A = E' P'i+1 A.

Continue this for n-1 steps where A is an n by n matrix. At this point we have U = E P A.

Problem. Modify the definition of lufact to get a

plufact('P',A,'L','U').

There are words in linalg to do gaussian elimination with

row interchanges on an augmented matrix.

```
A := randmatrix(3,4);
```

$$A := \begin{bmatrix} -91 & -53 & -19 & -47 \\ 68 & -72 & -87 & 79 \\ 43 & -66 & -53 & -61 \end{bmatrix}$$

```
gausselim(A);
```

$$\begin{bmatrix} -91 & -53 & -19 & -47 \\ 0 & \frac{-8285}{91} & \frac{-5640}{91} & \frac{-7572}{91} \\ 0 & 0 & \frac{-41795}{1657} & \frac{1208607}{8285} \end{bmatrix}$$

```
backsub(");
```

$$\left[\frac{-230146}{208975}, \frac{1013748}{208975}, \frac{-1208607}{208975}\right]$$

This gives the solution to the original system.

```
plufact := proc(`P`,A,`L`,`U` )
local k,i,j,t,el,p,l ;
k := rowdim(A);
U := evalm(A);
L := diag(seq(1,i = 1 .. k));
P := diag(seq(1,i = 1 .. k));
for i to k-1 do
if U[i,i] = 0 then
for t from i+1 to k while U[t,i]=0 do od ;
if U[t,i] <> 0 then p := interchange(k,t,i);
P := evalm( p &*P);
U := evalm(p &*U);
L := evalm(p&*L&*p)
else
RETURN(ERROR(cat( `zero pivot encountered at `,i,` step `)))fi fi;
for j from i+1 to k do
if not U[j,i] = 0 then
L[j,i] := U[j,i]/U[i,i];
el := elim( k,j,i,-L[j,i] );
```

```
U := evalm(el &* U); fi;
od;
od;
if U[k,k] = 0 then
RETURN( ERROR(cat( `zero pivot encountered at `,i,` step `))) fi;
NULL:
end:
```

Lets test this on some matrices where row interchanges have to take place.

```
n := 7 :  A := randmatrix(n,n);
```

$$A := \begin{bmatrix} -23 & -37 & 31 & -34 & -42 & 88 & -76 \\ -65 & 25 & 28 & -61 & -60 & 9 & 29 \\ -66 & -32 & 78 & 39 & 94 & 68 & -17 \\ -98 & -36 & 40 & 22 & 5 & -88 & -43 \\ -73 & 25 & 4 & -59 & 62 & -55 & 25 \\ 9 & 40 & 61 & 40 & -78 & 62 & 11 \\ 88 & 1 & 30 & 81 & -5 & -28 & 4 \end{bmatrix}$$

```
for i from 1 to n-2 do for j from 1 to n-i-1
do A[i,j] := 0 od od:
```

```
plufact(p,A,l,u);
```

```
map(evalm,[A,l,u,p]);
```

$$\left[\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 88 & -76 \\ 0 & 0 & 0 & 0 & -60 & 9 & 29 \\ 0 & 0 & 0 & 39 & 94 & 68 & -17 \\ 0 & 0 & 40 & 22 & 5 & -88 & -43 \\ 0 & 25 & 4 & -59 & 62 & -55 & 25 \\ 9 & 40 & 61 & 40 & -78 & 62 & 11 \\ 88 & 1 & 30 & 81 & -5 & -28 & 4 \end{bmatrix},\right.$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{88}{9} & \frac{-3511}{225} & \frac{-18901}{1500} & \frac{-55013}{2250} & \frac{-18389099}{270000} & \frac{-29316623}{7920000} & 1 \end{bmatrix},$$

$$\left.\begin{bmatrix} 9 & 40 & 61 & 40 & -78 & 62 & 11 \\ 0 & 25 & 4 & -59 & 62 & -55 & 25 \\ 0 & 0 & 40 & 22 & 5 & -88 & -43 \\ 0 & 0 & 0 & 39 & 94 & 68 & -17 \\ 0 & 0 & 0 & 0 & -60 & 9 & 29 \\ 0 & 0 & 0 & 0 & 0 & 88 & -76 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{6075891931}{5940000} \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}\right]$$

```
evalm(p&*A-l&*u);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Problem: Devise some more random matrices to test plufact on.

Problem: Modify plufact so that it computes and returns the determinant of A as it is computing P, L, and U.

Problem: Modify plufact so that it works on any matrix. That is, get a genplufact(p,a,l,u) that takes an n by m matrix a and returns an nbyn permutation matrix p, a lower unit triangular matrix l, and an echelon matrix u. Devise and use a way of constructing a family of test matrices to submit to genplufact

The idea is to copy down plufact and try to modify it to genplufact.

Solution:

```
with(linalg):

elim := proc(size,row,col,entry)
local A,i;
A := diag(seq(1,i = 1 ..  size));
A[row,col] := entry;
evalm(A)
end:

interchange := proc(size,row1,row2)
local A,i;
A := diag(seq(1,i = 1 ..  size));
A := swaprow(A,row1,row2)
end:

plufact := proc(`P`,A,`L`,`U` )
local k,i,j,t,el,p,l,hold,dt;
k := rowdim(A);
U := evalm(A);
L := diag(seq(1,i = 1 ..  k));
P := diag(seq(1,i = 1 ..  k));
```

```
dt := 1:

for i to k-1 do

if U[i,i] = 0 then

for t from i+1 to k while U[t,i]=0
do od ;

if U[t,i] <> 0 then p :=
interchange(k,t,i);

dt := -dt:

P := evalm( p &*P);

U := evalm(p &*U);

L := evalm(p&*L&*p)

else

RETURN(ERROR(cat( `zero pivot encountered
at `,i,` step `)))fi fi;

dt := dt*U[i,i]:

for j from i+1 to k do

if not U[j,i] = 0 then
L[j,i] := U[j,i]/U[i,i];

el := elim( k,j,i,-L[i,j]
);

U := evalm(el &* U); fi;

od;

od;

if U[k,k] = 0 then

RETURN( ERROR(cat( `zero pivot encountered
at step `, i))) fi;

dt := dt*U[k,k]:

dt;

end:

n := 5:

A := randmatrix(n,n):

for i from 1 by 2 to n do for j from 1 by 3
to n do A[i,j] := 0:  od od:

evalm(A);
```

$$\begin{bmatrix} 0 & 10 & 57 & 0 & -48 \\ -11 & 38 & -7 & 58 & -94 \\ 0 & 14 & -35 & 0 & -9 \\ -51 & -73 & -73 & -91 & 1 \\ 0 & -86 & 43 & 0 & -50 \end{bmatrix}$$

```
Digits := 10:  plufact(p,evalf(evalm(A)),l,u);
Digits := 10:
```

$$-.17517500\,10^{8}$$

```
map(evalm,[transpose(p)&*l&*u]);
```

$$\left[\begin{bmatrix} 0 & 10. & 57. & 0 & -48. \\ -11. & 38. & -7. & 58. & -94. \\ 0 & 28.00000000 & 44.80000000 & 0 & -76.20000000 \\ -102.0000000 & 59.3818182 & -594.5545455 & 177.9090909 & -103.1896104 \\ 0 & -189.2000000 & -404.2000000 & 0 & 373.8571429 \end{bmatrix}\right]$$

```
evalm(u);
```

$$\begin{bmatrix} -11. & 38. & -7. & 58. & -94. \\ 0 & 10. & 57. & 0 & -48. \\ 0 & 14. & -35. & 0 & -9. \\ -51. & -73. & -73. & -91. & 1. \\ 0 & -86. & 43. & 0 & -50. \end{bmatrix}$$

```
printlevel:=1;
```

$$printlevel := 1$$

```
gplufact := proc(`P`,A,`L`,`U`)
local sol,p,j,el,c,k,i,fv;
#global c,k,i,fv;
k := rowdim(A);
c := coldim(A);
U := evalm(A);
L := diag(seq(1,i = 1 ..  k));
P := diag(seq(1,i = 1 ..  k));
fv := 0;
for i to min(k,c) while i+fv <= c do
sol := getpc(U,c,k,i,fv );
if not sol = `done` then
p := interchange(k,sol[1],i);
fv := sol[2]-i;
P := evalm(p &* P);
U := evalm(p &* U);
L := evalm((p &* L) &* p);
for j from i+1 to k do
if not U[j,i+fv] = 0 then
```

```
L[j,i ] := U[j,i+fv]/U[i,i+fv];
el := elim(k,j,i,-L[j,i ]);
U := evalm(el &* U)
fi
od
else RETURN(‘done‘)
fi
od
end:
```

```
getpc := proc(U,c,k,i,fv)
local pc,t;
for pc from i+fv to c do
for t from i to k do
if not U[t,pc] = 0 then
RETURN([t,pc])
fi
od
od;
RETURN(‘done‘)
end:
```

```
printlevel:=1 ;
```

$$printlevel := 1$$

```
gplufact(p, C2,l ,v);
```

```
Error, (in rowdim) expecting a matrix
```

```
evalm(l&*v);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1.400000000 & 1 & 0 & 0 \\ 4.636363636 & -7.300000000 & 2.085714286 & 1 & 0 \\ 0 & -8.600000000 & -1.228571429 & 0 & 1 \end{bmatrix} \&* \, v$$

```
evalm(p&*C2-l&*v);
```

$$\left(\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \&* \, C2\right) - \left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1.400000000 & 1 & 0 & 0 \\ 4.636363636 & -7.300000000 & 2.085714286 & 1 & 0 \\ 0 & -8.600000000 & -1.228571429 & 0 & 1 \end{bmatrix} \&* \, v\right)$$

```
evalm(u);
```

$$\begin{bmatrix} -11. & 38. & -7. & 58. & -94. \\ 0 & 10. & 57. & 0 & -48. \\ 0 & 14. & -35. & 0 & -9. \\ -51. & -73. & -73. & -91. & 1. \\ 0 & -86. & 43. & 0 & -50. \end{bmatrix}$$

```
A := randmatrix(3,6);
```

$$A := \begin{bmatrix} 50 & 67 & -39 & 8 & -49 & 11 \\ 93 & -14 & -99 & -67 & 68 & 45 \\ 76 & 6 & 72 & -28 & -61 & -59 \end{bmatrix}$$

```
nullspace(A);
```

$$\left\{\left[1, 0, \frac{119758}{46853}, \frac{20301}{46853}, 0, \frac{196864}{46853}\right], \left[0, 1, \frac{353557}{140559}, \frac{-63799}{46853}, 0, \frac{178862}{46853}\right], \left[0, 0, \frac{-103734}{46853}, \frac{63145}{46853}, 1, \frac{-204999}{46853}\right]\right\}$$

```
gausselim(A);
```

$$\begin{bmatrix} 50 & 67 & -39 & 8 & -49 & 11 \\ 0 & \frac{-6931}{50} & \frac{-1323}{50} & \frac{-2047}{25} & \frac{7957}{50} & \frac{1227}{50} \\ 0 & 0 & \frac{1036698}{6931} & \frac{114020}{6931} & \frac{-669169}{6931} & \frac{-642411}{6931} \end{bmatrix}$$

```
B := stack(evalm(3*row(
A,1)-2*row(A,3)),A);
```

$$B := \begin{bmatrix} -2 & 189 & -261 & 80 & -25 & 151 \\ 50 & 67 & -39 & 8 & -49 & 11 \\ 93 & -14 & -99 & -67 & 68 & 45 \\ 76 & 6 & 72 & -28 & -61 & -59 \end{bmatrix}$$

```
C := stack(row(B,1),B);
```

$$C := \begin{bmatrix} -2 & 189 & -261 & 80 & -25 & 151 \\ -2 & 189 & -261 & 80 & -25 & 151 \\ 50 & 67 & -39 & 8 & -49 & 11 \\ 93 & -14 & -99 & -67 & 68 & 45 \\ 76 & 6 & 72 & -28 & -61 & -59 \end{bmatrix}$$

```
C2 := stack(C,C);
```

$$C2 := \begin{bmatrix} -2 & 189 & -261 & 80 & -25 & 151 \\ -2 & 189 & -261 & 80 & -25 & 151 \\ 50 & 67 & -39 & 8 & -49 & 11 \\ 93 & -14 & -99 & -67 & 68 & 45 \\ 76 & 6 & 72 & -28 & -61 & -59 \\ -2 & 189 & -261 & 80 & -25 & 151 \\ -2 & 189 & -261 & 80 & -25 & 151 \\ 50 & 67 & -39 & 8 & -49 & 11 \\ 93 & -14 & -99 & -67 & 68 & 45 \\ 76 & 6 & 72 & -28 & -61 & -59 \end{bmatrix}$$

```
B[1,2]:=0:B[2,2]:=0:  B[3,2]:=0:  B[4,2]:=0:
nullspace(B);
```

$$\left\{\left[1,\ 0,\ \frac{119758}{46853},\ \frac{20301}{46853},\ 0,\ \frac{196864}{46853}\right],\ \left[0,\ 0,\ \frac{-103734}{46853},\ \frac{63145}{46853},\ 1,\ \frac{-204999}{46853}\right],\ [0,\ 1,\ 0,\ 0,\ 0,\ 0]\right\}$$

```
gausselim(B);
```

$$\begin{bmatrix} -2 & 0 & -261 & 80 & -25 & 151 \\ 0 & 0 & -6564 & 2008 & -674 & 3786 \\ 0 & 0 & 0 & \frac{-49216}{547} & \frac{354143}{2188} & \frac{20301}{2188} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
gplufact(p,B,l,u);
```

$$done$$

```
map(evalm,[B,p,l,u]);
```

$$\left[\begin{bmatrix} -2 & 0 & -261 & 80 & -25 & 151 \\ 50 & 0 & -39 & 8 & -49 & 11 \\ 93 & 0 & -99 & -67 & 68 & 45 \\ 76 & 0 & 72 & -28 & -61 & -59 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ -25 & 1 & 0 & 0 \\ \frac{-93}{2} & \frac{8157}{4376} & 1 & 0 \\ -38 & \frac{3}{2} & 0 & 1 \end{bmatrix},\right.$$
$$\left.\begin{bmatrix} -2 & 0 & -261 & 80 & -25 & 151 \\ 0 & 0 & -6564 & 2008 & -674 & 3786 \\ 0 & 0 & 0 & \frac{-49216}{547} & \frac{354143}{2188} & \frac{20301}{2188} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right]$$

8 Projections and Reflections

Problem. Given a line through the origin in Rn, find the orthogonal projection of Rn onto the line. Soln This is a problem we solved in Ma 213, although we didn't find a matrix which provided the projection.

```
with(linalg):
n := 3:
a := matrix(3,1,[3,2,9]);
```

$$a := \begin{bmatrix} 3 \\ 2 \\ 9 \end{bmatrix}$$

```
P =
cat(evalm(a),inverse(transpose(a)&*a),transpose(a));
```

$$P = . \begin{bmatrix} 3 \\ 2 \\ 9 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{94} \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 & 9 \end{bmatrix}$$

```
P:=
evalm(a&*(inverse(transpose(a)&*a)&*transpose(a)));
```

$$P := \begin{bmatrix} \frac{9}{94} & \frac{3}{47} & \frac{27}{94} \\ \frac{3}{47} & \frac{2}{47} & \frac{9}{47} \\ \frac{27}{94} & \frac{9}{47} & \frac{81}{94} \end{bmatrix}$$

```
evalm(P&*[6,4,2]);
```

$$\left[\frac{66}{47}, \frac{44}{47}, \frac{198}{47}\right]$$

Problem: Given linearly independent vectors a1 through an in R^m, find the matrix that projects R^m onto the subspace spanned by the ai's. Soln: Form a matrix a with the ai's as columns. Then the matrix A*(A^T*A)^(-1)*A^T is the desired projection matrix. For example, to find the projection P of R^3 onto the plane spanned by $a_1=(1,2,3)$ and $a_2=(3,0,1)$ –

```
a := matrix(3,2,[1,3,2,0,3,1]);
```

$$a := \begin{bmatrix} 1 & 3 \\ 2 & 0 \\ 3 & 1 \end{bmatrix}$$

```
p:=evalm(evalm(a&*inverse(transpose(a)&*a))&*
transpose(a));
```

$$p := \begin{bmatrix} \frac{25}{26} & \frac{-2}{13} & \frac{3}{26} \\ \frac{-2}{13} & \frac{5}{13} & \frac{6}{13} \\ \frac{3}{26} & \frac{6}{13} & \frac{17}{26} \end{bmatrix}$$

Problem: Given a house in R^3, project it onto the plane spanned by a1 and a2 given above. Soln: Whats a house? A set of faces: front, back, two sides and a roof. Maybe a door. Each of these faces is determined by a set of vertices: So one version of house is

```
front :=
[[-1,0,0],[1,0,0],[1,0,4],[0,0,6],[-1,0,4]];
```

$$front := [[-1, 0, 0], [1, 0, 0], [1, 0, 4], [0, 0, 6], [-1, 0, 4]]$$

```
back :=
[[-1,3,0],[1,3,0],[1,3,4],[0,3,6],[-1,3,4]];
```

$$back := [[-1, 3, 0], [1, 3, 0], [1, 3, 4], [0, 3, 6], [-1, 3, 4]]$$

```
roof1:=[[0,0,6],[0,3,6],[1,3,4],[1,0,4]];
```

$$roof1 := [[0, 0, 6], [0, 3, 6], [1, 3, 4], [1, 0, 4]]$$

```
roof2:=[[0,0,6],[0,3,6],[-1,3,4],[-1,0,4]];
```

$$roof2 := [[0, 0, 6], [0, 3, 6], [-1, 3, 4], [-1, 0, 4]]$$

```
side1 :=[[1,0,0],[1,3,0],[1,3,4],[1,0,4]];
```

$$side1 := [[1, 0, 0], [1, 3, 0], [1, 3, 4], [1, 0, 4]]$$

```
side2
:=[[-1,0,0],[-1,3,0],[-1,3,4],[-1,0,4]];
```

$$side2 := [[-1, 0, 0], [-1, 3, 0], [-1, 3, 4], [-1, 0, 4]]$$

```
door :=
[[1,1,0],[1,1.5,0],[1,1.5,3],[1,1,3]];
```

$$door := [[1, 1, 0], [1, 1.5, 0], [1, 1.5, 3], [1, 1, 3]]$$

```
house :=
[front,back,roof1,roof2,side1,side2,door];
```

$$\begin{aligned} house := [&[[-1, 0, 0], [1, 0, 0], [1, 0, 4], [0, 0, 6], [-1, 0, 4]], \\ &[[-1, 3, 0], [1, 3, 0], [1, 3, 4], [0, 3, 6], [-1, 3, 4]], [[0, 0, 6], [0, 3, 6], [1, 3, 4], [1, 0, 4]], \\ &[[0, 0, 6], [0, 3, 6], [-1, 3, 4], [-1, 0, 4]], [[1, 0, 0], [1, 3, 0], [1, 3, 4], [1, 0, 4]], \\ &[[-1, 0, 0], [-1, 3, 0], [-1, 3, 4], [-1, 0, 4]], [[1, 1, 0], [1, 1.5, 0], [1, 1.5, 3], [1, 1, 3]]] \end{aligned}$$

```
polygonplot3d(house,scaling=constrained,style=patch);
```

polygonplot3d([[[−1, 0, 0], [1, 0, 0], [1, 0, 4], [0, 0, 6], [−1, 0, 4]],
[[−1, 3, 0], [1, 3, 0], [1, 3, 4], [0, 3, 6], [−1, 3, 4]], [[0, 0, 6], [0, 3, 6], [1, 3, 4], [1, 0, 4]],
[[0, 0, 6], [0, 3, 6], [−1, 3, 4], [−1, 0, 4]], [[1, 0, 0], [1, 3, 0], [1, 3, 4], [1, 0, 4]],
[[−1, 0, 0], [−1, 3, 0], [−1, 3, 4], [−1, 0, 4]], [[1, 1, 0], [1, 1.5, 0], [1, 1.5, 3], [1, 1, 3]]],
scaling = *constrained*, *style* = *patch*)

We can save this plot for later display by giving it a name.

```
pl1 :=
polygonplot3d(house,scaling=constrained );
```

pl1 := polygonplot3d([[[−1, 0, 0], [1, 0, 0], [1, 0, 4], [0, 0, 6], [−1, 0, 4]],
[[−1, 3, 0], [1, 3, 0], [1, 3, 4], [0, 3, 6], [−1, 3, 4]], [[0, 0, 6], [0, 3, 6], [1, 3, 4], [1, 0, 4]],
[[0, 0, 6], [0, 3, 6], [−1, 3, 4], [−1, 0, 4]], [[1, 0, 0], [1, 3, 0], [1, 3, 4], [1, 0, 4]],
[[−1, 0, 0], [−1, 3, 0], [−1, 3, 4], [−1, 0, 4]], [[1, 1, 0], [1, 1.5, 0], [1, 1.5, 3], [1, 1, 3]]],
scaling = *constrained*)

So to translate this house around we need a couple of words, trans and transhouse

```
trans := proc(pt,part)

local i;

[seq(convert(evalm(pt+part[i]),list),i=1..nop
s(part))] end:

transhouse := proc(pt,house)

local thouse,part;

thouse := NULL:

for part in house do

thouse := thouse,trans(pt,part) od;

[thouse] end:

with(plots):
```

Lets move house away from the coordinate planes.

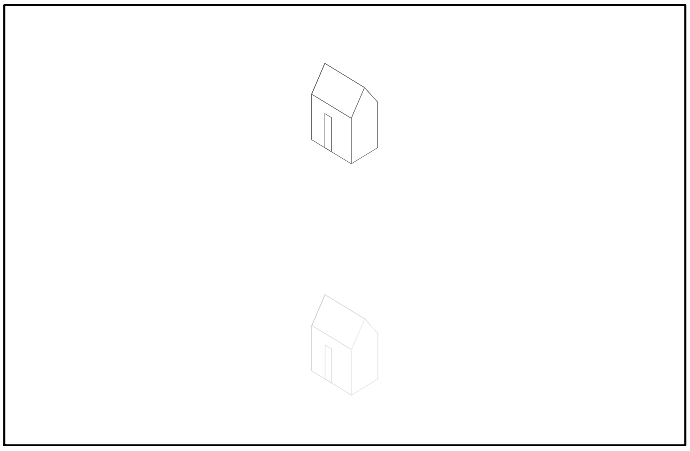
```
thouse := transhouse([-8,-8,9],house);
```

thouse := [[[−9, −8, 9], [−7, −8, 9], [−7, −8, 13], [−8, −8, 15], [−9, −8, 13]],
[[−9, −5, 9], [−7, −5, 9], [−7, −5, 13], [−8, −5, 15], [−9, −5, 13]],
[[−8, −8, 15], [−8, −5, 15], [−7, −5, 13], [−7, −8, 13]],
[[−8, −8, 15], [−8, −5, 15], [−9, −5, 13], [−9, −8, 13]],
[[−7, −8, 9], [−7, −5, 9], [−7, −5, 13], [−7, −8, 13]],
[[−9, −8, 9], [−9, −5, 9], [−9, −5, 13], [−9, −8, 13]],
[[−7, −7, 9], [−7, −6.5, 9], [−7, −6.5, 12], [−7, −7, 12]]]

```
pl2 :=
polygonplot3d(thouse,scaling=constrained);
```

$pl2$:= **PLOT3D(POLYGONS(**
[[−9., −8., 9.], [−7., −8., 9.], [−7., −8., 13.], [−8., −8., 15.], [−9., −8., 13.]],
[[−9., −5., 9.], [−7., −5., 9.], [−7., −5., 13.], [−8., −5., 15.], [−9., −5., 13.]],
[[−8., −8., 15.], [−8., −5., 15.], [−7., −5., 13.], [−7., −8., 13.]],
[[−8., −8., 15.], [−8., −5., 15.], [−9., −5., 13.], [−9., −8., 13.]],
[[−7., −8., 9.], [−7., −5., 9.], [−7., −5., 13.], [−7., −8., 13.]],
[[−9., −8., 9.], [−9., −5., 9.], [−9., −5., 13.], [−9., −8., 13.]],
[[−7., −7., 9.], [−7., −6.5, 9.], [−7., −6.5, 12.], [−7., −7., 12.]]), **SCALING(*CONSTRAINED*))**

```
plots[display]([pl1,pl2]);
```



To project house onto the space spanned by a1 and a2, we need a couple of words.

```
proj := proc(p,part)
local i;
[seq(convert(evalm(p &*
part[i]),list),i=1..nops(part))] end:
proj(p,roof1);
```

$$[[\frac{9}{13}, \frac{36}{13}, \frac{51}{13}], [\frac{3}{13}, \frac{51}{13}, \frac{69}{13}], [\frac{25}{26}, \frac{37}{13}, \frac{107}{26}], [\frac{37}{26}, \frac{22}{13}, \frac{71}{26}]]$$

```
projhouse := proc(p,house)
local phouse,part;
phouse := NULL:
for part in house do
phouse := phouse,proj(p,part) od:
[phouse] end;
```

$$
\begin{array}{l}
\mathit{projhouse} := \mathrm{proc}(p,\ \mathit{house}) \\
\quad \mathrm{local}\,\mathit{phouse},\ \mathit{part}; \\
\qquad \mathit{phouse} := \mathit{NULL}\,;\ \mathrm{for}\ \mathit{part}\ \mathrm{in}\ \mathit{house}\ \mathrm{do}\ \mathit{phouse} := \mathit{phouse},\ \mathrm{proj}(p,\ \mathit{part})\ \mathrm{od}\ ;\ [\mathit{phouse}] \\
\quad \mathrm{end}
\end{array}
$$

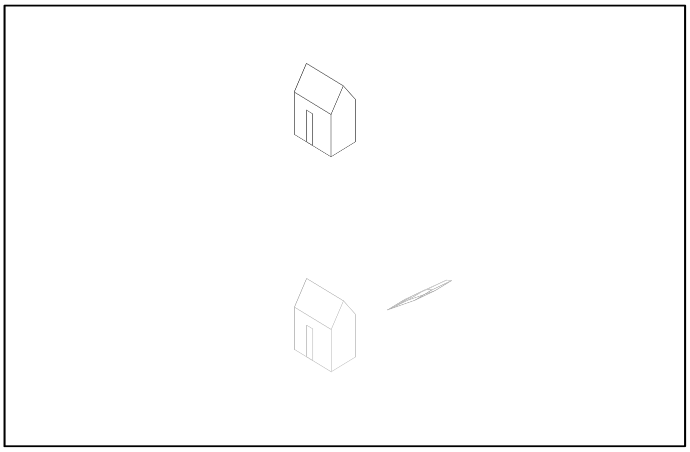
```
phouse := projhouse(p,thouse);
```

$$
\begin{array}{l}
\mathit{phouse} := [[[\frac{-83}{13},\ \frac{32}{13},\ \frac{15}{13}],\ [\frac{-58}{13},\ \frac{28}{13},\ \frac{18}{13}],\ [-4,\ 4,\ 4],\ [\frac{-123}{26},\ \frac{66}{13},\ \frac{135}{26}],\ [\frac{-77}{13},\ \frac{56}{13},\ \frac{49}{13}]], \\
\quad [[\frac{-89}{13},\ \frac{47}{13},\ \frac{33}{13}],\ [\frac{-64}{13},\ \frac{43}{13},\ \frac{36}{13}],\ [\frac{-58}{13},\ \frac{67}{13},\ \frac{70}{13}],\ [\frac{-135}{26},\ \frac{81}{13},\ \frac{171}{26}],\ [\frac{-83}{13},\ \frac{71}{13},\ \frac{67}{13}]], \\
\quad [[\frac{-123}{26},\ \frac{66}{13},\ \frac{135}{26}],\ [\frac{-135}{26},\ \frac{81}{13},\ \frac{171}{26}],\ [\frac{-58}{13},\ \frac{67}{13},\ \frac{70}{13}],\ [-4,\ 4,\ 4]], \\
\quad [[\frac{-123}{26},\ \frac{66}{13},\ \frac{135}{26}],\ [\frac{-135}{26},\ \frac{81}{13},\ \frac{171}{26}],\ [\frac{-83}{13},\ \frac{71}{13},\ \frac{67}{13}],\ [\frac{-77}{13},\ \frac{56}{13},\ \frac{49}{13}]], \\
\quad [[\frac{-58}{13},\ \frac{28}{13},\ \frac{18}{13}],\ [\frac{-64}{13},\ \frac{43}{13},\ \frac{36}{13}],\ [\frac{-58}{13},\ \frac{67}{13},\ \frac{70}{13}],\ [-4,\ 4,\ 4]], \\
\quad [[\frac{-83}{13},\ \frac{32}{13},\ \frac{15}{13}],\ [\frac{-89}{13},\ \frac{47}{13},\ \frac{33}{13}],\ [\frac{-83}{13},\ \frac{71}{13},\ \frac{67}{13}],\ [\frac{-77}{13},\ \frac{56}{13},\ \frac{49}{13}]],\ [[\frac{-60}{13},\ \frac{33}{13},\ \frac{24}{13}], \\
\quad [-4.692307693,\ 2.730769231,\ 2.076923077],\ [-4.346153846,\ 4.115384615,\ 4.038461538], \\
\quad [\frac{-111}{26},\ \frac{51}{13},\ \frac{99}{26}]]]
\end{array}
$$

```
pl3 :=
polygonplot3d(phouse,scaling=constrained);
```

$$
\begin{array}{l}
\mathit{pl3} := \mathrm{PLOT3D}(\mathrm{POLYGONS}([[-6.384615385,\ 2.461538462,\ 1.153846154], \\
\quad [-4.461538462,\ 2.153846154,\ 1.384615385],\ [-4.,\ 4.,\ 4.], \\
\quad [-4.730769231,\ 5.076923077,\ 5.192307692],\ [-5.923076923,\ 4.307692308,\ 3.769230769] \\
\quad],[[-6.846153846,\ 3.615384615,\ 2.538461538], \\
\quad [-4.923076923,\ 3.307692308,\ 2.769230769],\ [-4.461538462,\ 5.153846154,\ 5.384615385], \\
\quad [-5.192307692,\ 6.230769231,\ 6.576923077],\ [-6.384615385,\ 5.461538462,\ 5.153846154] \\
\quad],[[-4.730769231,\ 5.076923077,\ 5.192307692], \\
\quad [-5.192307692,\ 6.230769231,\ 6.576923077],\ [-4.461538462,\ 5.153846154,\ 5.384615385], \\
\quad [-4.,\ 4.,\ 4.]],[[-4.730769231,\ 5.076923077,\ 5.192307692], \\
\quad [-5.192307692,\ 6.230769231,\ 6.576923077],\ [-6.384615385,\ 5.461538462,\ 5.153846154], \\
\quad [-5.923076923,\ 4.307692308,\ 3.769230769]],[\\
\quad [-4.461538462,\ 2.153846154,\ 1.384615385],\ [-4.923076923,\ 3.307692308,\ 2.769230769], \\
\quad [-4.461538462,\ 5.153846154,\ 5.384615385],\ [-4.,\ 4.,\ 4.]],[\\
\quad [-6.384615385,\ 2.461538462,\ 1.153846154],\ [-6.846153846,\ 3.615384615,\ 2.538461538], \\
\quad [-6.384615385,\ 5.461538462,\ 5.153846154],\ [-5.923076923,\ 4.307692308,\ 3.769230769] \\
\quad],[[-4.615384615,\ 2.538461538,\ 1.846153846], \\
\quad [-4.692307693,\ 2.730769231,\ 2.076923077],\ [-4.346153846,\ 4.115384615,\ 4.038461538], \\
\quad [-4.269230769,\ 3.923076923,\ 3.807692308]]),\ \mathrm{SCALING}(\mathit{CONSTRAINED}))
\end{array}
$$

```
display([pl1,pl2,pl3]);
```



What about reflecting a house through the plane spanned by a1 and a2? Let R be the reflection matrix. Then for any b in R3, Rb is Pb + (Pb - b) = Pb + Pb - Ib = (2P - I)b, so R is the matrix twice the projection minus the identity matrix. A word to construct R from P is

```
reflect := proc(p)

local i;

evalm(2*p-diag(seq(1,i=1..rowdim(p))))
end;
```

$$\mathit{reflect} := \mathrm{proc}(p)\ \mathrm{local}\, i;\ \mathrm{evalm}(2 \times p - \mathrm{diag}(\mathrm{seq}(1,\ i = 1..\mathrm{rowdim}(p))))\ \mathrm{end}$$

```
r := reflect(p);
```

$$r := \begin{bmatrix} \frac{12}{13} & \frac{-4}{13} & \frac{3}{13} \\ \frac{-4}{13} & \frac{-3}{13} & \frac{12}{13} \\ \frac{3}{13} & \frac{12}{13} & \frac{4}{13} \end{bmatrix}$$

Let's reflect thouse through the plane spanned by a1 and a2.

```
rhouse := projhouse(r,thouse);
```

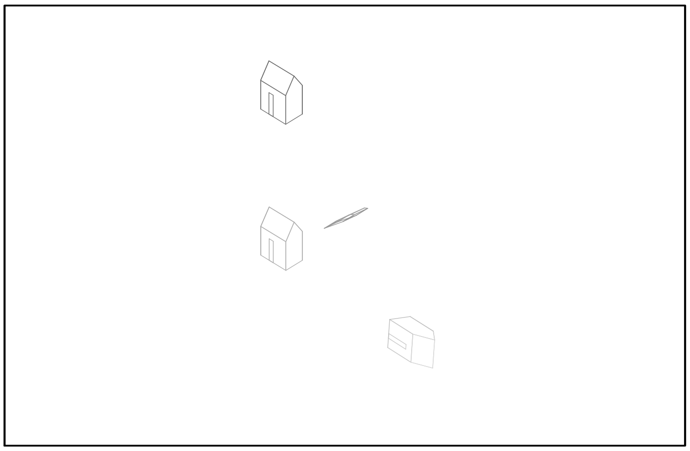
$$\begin{aligned}
\mathit{rhouse} := {} & [[[\tfrac{-49}{13}, \tfrac{168}{13}, \tfrac{-87}{13}], [\tfrac{-25}{13}, \tfrac{160}{13}, \tfrac{-81}{13}], [-1, 16, -5], [\tfrac{-19}{13}, \tfrac{236}{13}, \tfrac{-60}{13}], [\tfrac{-37}{13}, \tfrac{216}{13}, \tfrac{-71}{13}]], \\
& [[\tfrac{-61}{13}, \tfrac{159}{13}, \tfrac{-51}{13}], [\tfrac{-37}{13}, \tfrac{151}{13}, \tfrac{-45}{13}], [\tfrac{-25}{13}, \tfrac{199}{13}, \tfrac{-29}{13}], [\tfrac{-31}{13}, \tfrac{227}{13}, \tfrac{-24}{13}], [\tfrac{-49}{13}, \tfrac{207}{13}, \tfrac{-35}{13}]], \\
& [[\tfrac{-19}{13}, \tfrac{236}{13}, \tfrac{-60}{13}], [\tfrac{-31}{13}, \tfrac{227}{13}, \tfrac{-24}{13}], [\tfrac{-25}{13}, \tfrac{199}{13}, \tfrac{-29}{13}], [-1, 16, -5]], \\
& [[\tfrac{-19}{13}, \tfrac{236}{13}, \tfrac{-60}{13}], [\tfrac{-31}{13}, \tfrac{227}{13}, \tfrac{-24}{13}], [\tfrac{-49}{13}, \tfrac{207}{13}, \tfrac{-35}{13}], [\tfrac{-37}{13}, \tfrac{216}{13}, \tfrac{-71}{13}]], \\
& [[\tfrac{-25}{13}, \tfrac{160}{13}, \tfrac{-81}{13}], [\tfrac{-37}{13}, \tfrac{151}{13}, \tfrac{-45}{13}], [\tfrac{-25}{13}, \tfrac{199}{13}, \tfrac{-29}{13}], [-1, 16, -5]], \\
& [[\tfrac{-49}{13}, \tfrac{168}{13}, \tfrac{-87}{13}], [\tfrac{-61}{13}, \tfrac{159}{13}, \tfrac{-51}{13}], [\tfrac{-49}{13}, \tfrac{207}{13}, \tfrac{-35}{13}], [\tfrac{-37}{13}, \tfrac{216}{13}, \tfrac{-71}{13}]], [[\tfrac{-29}{13}, \tfrac{157}{13}, \tfrac{-69}{13}], \\
& [-2.384615385, 11.96153846, -4.846153846], \\
& [-1.692307693, 14.73076923, -3.923076923], [\tfrac{-20}{13}, \tfrac{193}{13}, \tfrac{-57}{13}]]]
\end{aligned}$$

Plot all of these togther.

```
pl4 := polygonplot3d(rhouse);
```

$pl4$:= **PLOT3D**(**POLYGONS**([[−3.769230769, 12.92307692, −6.692307692],
[−1.923076923, 12.30769231, −6.230769231], [−1., 16., −5.],
[−1.461538462, 18.15384615, −4.615384615],
[−2.846153846, 16.61538462, −5.461538462]], [
[−4.692307692, 12.23076923, −3.923076923],
[−2.846153846, 11.61538462, −3.461538462],
[−1.923076923, 15.30769231, −2.230769231],
[−2.384615385, 17.46153846, −1.846153846],
[−3.769230769, 15.92307692, −2.692307692]], [
[−1.461538462, 18.15384615, −4.615384615],
[−2.384615385, 17.46153846, −1.846153846],
[−1.923076923, 15.30769231, −2.230769231], [−1., 16., −5.]], [
[−1.461538462, 18.15384615, −4.615384615],
[−2.384615385, 17.46153846, −1.846153846],
[−3.769230769, 15.92307692, −2.692307692],
[−2.846153846, 16.61538462, −5.461538462]], [
[−1.923076923, 12.30769231, −6.230769231],
[−2.846153846, 11.61538462, −3.461538462],
[−1.923076923, 15.30769231, −2.230769231], [−1., 16., −5.]], [
[−3.769230769, 12.92307692, −6.692307692],
[−4.692307692, 12.23076923, −3.923076923],
[−3.769230769, 15.92307692, −2.692307692],
[−2.846153846, 16.61538462, −5.461538462]], [
[−2.230769231, 12.07692308, −5.307692308],
[−2.384615385, 11.96153846, −4.846153846],
[−1.692307693, 14.73076923, −3.923076923],
[−1.538461538, 14.84615385, −4.384615385]]))

```
display([pl1,pl2,pl3,pl4]);
```



Problems:

1. In the worksheet cover.ms, a movie showing a two dimensional house being reflected through a line was constructed. Using that as a model, construct a movie showing a three dimensional house being reflected through a plane.

2. To obtain a reflection about the line l through a and b, we could construct the reflection matrix p though the line t*(b-a). Then the reflection of a point x would be obtained by subtracting a from x, then reflecting x-a about the line t*(b-a), then adding a onto p*(x-a), getting r(a,p,x)= a+p*(x-a). For example, the reflection about the line through a=[2,3] and b=[-6,1] would look like:

```
with(linalg):

a := matrix(2,1,[2,3]); b :=
matrix(2,1,[-6,1]);
```

$$a := \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$b := \begin{bmatrix} -6 \\ 1 \end{bmatrix}$$

```
p :=
evalm((b-a)&*inverse(transpose(b-a)&*(b-a)) &*transpose(b-a));
```

$$p := \begin{bmatrix} \frac{16}{17} & \frac{4}{17} \\ \frac{4}{17} & \frac{1}{17} \end{bmatrix}$$

```
ref := proc(x)

evalm(p&*(x-a) + a) end;
```

$$ref := \text{proc}(x)\ \text{evalm}((p\ `\&*`\ x - a) + a)\ \text{end}$$

```
ref([Pi,8]),evalf(");
```

$$\begin{bmatrix} \frac{16}{17}\pi + \frac{22}{17} \\ \frac{4}{17}\pi + \frac{48}{17} \end{bmatrix},\ \text{proc}(x)\ \text{evalm}((p\ `\&*`\ x - a) + a)\ \text{end}$$

a) Verify visually that this works by plotting the segments [a,b] and [x, ref(x)] for various x's. Also verify algebraically that if a = [1,0], b = [1,3], then

ref([x,y]) = [1 - (x-1),y] for all x,y.

b) A theorem from plane geometry says that every rigid motion on the plane can be obtained by performing no more than three reflections (about lines not necessarily through the origin). We saw in class how to realize each rotation and each translation as the product of two reflections. What does the product of three reflections do in general? Make some calculations and form a conjecture.

3. To reflect through the plane containing three noncollinear points a, b, and c, what can we do? Well, we can translate the plane to the origin by subtracting a, then reflecting about the translated plane, then translating back by adding a. Carry this out in as general a manner as you can, and illustrate your method by reflecting rhouse above through the plane through a = [0,2,1], b= [-3,2,9], and c=[2,0,6]. Draw a picture.

9 Least Squares Approximation.

Problem: Given some data [xi,yi] i = 1..m , that a function f(x) = a1 u1(x) + a2 u2(x) + ... + an un(x) is supposed to give the values yi = f(xi) for each i, find the values of a1 through an which come the nearest to satisfying the equations in the senses that the sum of the squares of the errors (f(xi)-yi) is as small as possible.

Calculus solution for a sample set of data that we want to do a linear regression on.

Here the idea is to set up the linear function f with unknown coefficients a1 and a2. So here u1(x) = x and u2(x) = 1. Then set up the error function e(a1,a2) and find the critical point by locating where both partial derivatives vanish. The minimum error is attained at that critical point.

```
data := [[.3,1],[2,4],[4,2],[7,9],[10,3]];
```

$$data := [[.3,\ 1],\ [2,\ 4],\ [4,\ 2],\ [7,\ 9],\ [10,\ 3]]$$

```
f := x -> a1*x + a2;
```

$$f := x \rightarrow a1\ x + a2$$

```
i := 'i' :  e := (a1,a2) ->
sum((f(data[i][1])-data[i][2])^2, i =1..nops(data));
```

$$e := (a1,\ a2) \rightarrow \sum_{i=1}^{\mathrm{nops}(data)} (\mathrm{f}(data_{i\,1}) - data_{i\,2})^2$$

```
e(a,b);
```

$$(.3\ a1 + a2 - 1)^2 + (2\ a1 + a2 - 4)^2 + (4\ a1 + a2 - 2)^2 + (7\ a1 + a2 - 9)^2 + (10\ a1 + a2 - 3)^2$$

```
eqn1 := diff(e(a1,a2),a1);
```

$$eqn1 := 338.18\ a1 + 46.6\ a2 - 218.6$$

```
eqn2 := diff(e(a1,a2),a2);
```

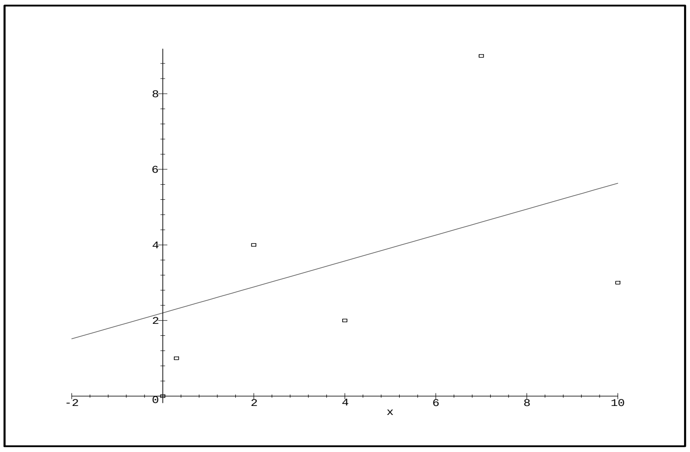
$$eqn2 := 46.6\ a1 + 10\ a2 - 38$$

```
sol1 := solve({eqn1,eqn2},{a1,a2});
```

$$sol1 := \{a2 = 2.201282390,\ a1 = .3430724484\}$$

```
pl1 :=plot({[0,0],seq(d,d=data )}
,style=point,symbol=box,color=black):
```

```
plots[display]([plot({
subs(sol1,f(x))},x=-2..10),pl1],color=black);
```



The average error in the data would be obtained by taking the square root of the average square error.

```
sqrt(subs(sol1,e(a1,a2))/nops(data));
```

$$2.517054468$$

Linear algebra solution using projections.

Here we use the same linear function f(x) = a1 x + a2, but now we set up the over determined system Axbar = b, where A is a 4 by 2 matrix, xbar is the unknown vector [a1,a2] and b is the ydata. As we saw in class, xbar is the solution to the nonsingular matrix equation transpose (A) A xbar = transpose(A) b. That's because A xbar is the projection of b onto the column space of A.

```
with(linalg):
```

```
A := transpose(
matrix(2,nops(data),[seq(data[i][1],i=1..nops(data)),
seq(1,i=1..nops(data))]));
```

$$A := \begin{bmatrix} .3 & 1 \\ 2 & 1 \\ 4 & 1 \\ 7 & 1 \\ 10 & 1 \end{bmatrix}$$

```
A := evalf(evalm(A));
```

$$A := \begin{bmatrix} .3 & 1. \\ 2. & 1. \\ 4. & 1. \\ 7. & 1. \\ 10. & 1. \end{bmatrix}$$

```
b := [seq(data[i][2],i=1..nops(data))]:
xbar := evalm(inverse(
transpose(A)&*A)&*transpose(A)&*b );
```

$$xbar := [.3430724493,\ 2.201282391]$$

Note: linalg has a word to do leastsquare approximation already.

```
leastsqrs(A,b);
```

$$[.3430724479,\ 2.201282393]$$

Problems:

1. In the data above, replace the data point [2,4] with [2,t]. Then find the smallest positive t so that the average error in the data is 5 or greater.

2. Using the original data above, find the quadratic function f(x) = a1 x^2 + a2 x + a3 and the function f(x) = a1 exp(x) + a2 x + a3 which come nearest fittting the data in the sense of least squares. Plot both of these functions together with the data on the same graph and decide which fits the data better. Does either of these fit the data better than the linear function?

QR factorization. The sticky place in the calculation of either a projection matrix or a least squares approximation is in finding the inverse of transpose(A)&* A. This can be improved by replacing the colums of A with an orthonormal set. This is called the QR factorization (or Gramm Schmidt)

1 First make A into Q', a matrix with orthogonal columns.

2. Then make Q' into an orthogonal matrix Q by normalizing its columns.

3. Then get R = transpose(Q)&*A. done.

The word proj gives the projection of a vector onto a line through the origin.

```
proj := proc(b,a)
evalm(dotprod(b,a)/dotprod(a,a)*a)
end;
```

$$proj := \mathrm{proc}(b,\ a)\ \mathrm{evalm}(\mathrm{dotprod}(b,\ a) \times a/\mathrm{dotprod}(a,\ a))\ \mathrm{end}$$

```
proj([3,4,5],[3,4,2]);
```

$$\left[\frac{105}{29},\ \frac{140}{29},\ \frac{70}{29}\right]$$

Now to get the QR factorization of a randomly chosen 4 by 3 matrix.

```
A := randmatrix(4,3);
```

$$A := \begin{bmatrix} -55 & -37 & -35 \\ 97 & 50 & 79 \\ 56 & 49 & 63 \\ 57 & -59 & 45 \end{bmatrix}$$

```
seq(col(A,i),i=1..3);
```

$$[-55,\ 97,\ 56,\ 57],\ [-37,\ 50,\ 49,\ -59],\ [-35,\ 79,\ 63,\ 45]$$

```
q1 := col(A,1);
```

$$q1 := [-55, 97, 56, 57]$$

```
q2 := evalm(col(A,2)-proj(col(A,2), q1 ));
```

$$q2 := \left[\frac{-351673}{18819}, \frac{333148}{18819}, \frac{571235}{18819}, \frac{-489161}{6273}\right]$$

```
q3 := evalm(col(A,3)-proj(col(A,3),
q1)-proj(col(A,3),q2));
```

$$q3 := \left[\frac{1722285969}{144241313}, \frac{-415186976}{144241313}, \frac{2096197642}{144241313}, \frac{308978895}{144241313}\right]$$

```
for i from 1 to 3 do q.i :=
evalm(1/sqrt(dotprod(q.i,q.i))*q.i) od:
Q := transpose(stack (q1,q2,q3));
```

$$Q := \begin{bmatrix} -\frac{55}{6273}\sqrt{2091}, & -\frac{351673}{904825756449}\sqrt{301608585483}, & \frac{1722285969}{7628161695936074726}\,\%1 \\ \frac{97}{6273}\sqrt{2091}, & \frac{333148}{904825756449}\sqrt{301608585483}, & -\frac{207593488}{3814080847968037363}\,\%1 \\ \frac{56}{6273}\sqrt{2091}, & \frac{571235}{904825756449}\sqrt{301608585483}, & \frac{1048098821}{3814080847968037363}\,\%1 \\ \frac{19}{2091}\sqrt{2091}, & -\frac{489161}{301608585483}\sqrt{301608585483}, & \frac{308978895}{7628161695936074726}\,\%1 \end{bmatrix}$$

$$\%1 := \sqrt{7628161695936074726}$$

```
R :=evalm(transpose(Q)&*A);
```

$$R := \begin{bmatrix} 3\sqrt{2091} & \frac{6266}{6273}\sqrt{2091} & \frac{5227}{2091}\sqrt{2091} \\ 0 & \frac{1}{6273}\sqrt{301608585483} & \frac{2859439}{301608585483}\sqrt{301608585483} \\ 0 & 0 & \frac{1}{144241313}\sqrt{7628161695936074726} \end{bmatrix}$$

Problem. Define a word getqr(A,'Q','R') which takes an m by n matrix A of rank n <= m and returns an orthogonal matrix Q and an upper triangular matrix R such that A = Q R. Test your word out on several matrices.

- Back

10 Systems of linear differential equations, and diagonalization.

```
restartwith(linalg):
```

```
Warning, new definition for norm
```

```
Warning, new definition for trace
```

Suppose X = [x,y] is a moving point with velocity dX/dt = [3x -w y, -w x + 2y]. If [x,y]=[1,3] at t=0 find the path. Where does the point go as t goes to infinity
(as a function of w)?
Solution 1. Set up the system of first order equations and use dsolve.

```
x := 'x':  y := 'y':
```

```
deqns := diff(x(t),t)= 3*x(t)-
w*y(t),diff(y(t),t)=- w*x(t)+2*y(t) ;
```

$$deqns := \frac{\partial}{\partial t}\,\mathrm{x}(t) = 3\,\mathrm{x}(t) - w\,\mathrm{y}(t),\ \frac{\partial}{\partial t}\,\mathrm{y}(t) = -w\,\mathrm{x}(t) + 2\,\mathrm{y}(t)$$

```
inits := x(0)=1,y(0)=3 ;
```

$$inits := \mathrm{x}(0) = 1,\ \mathrm{y}(0) = 3$$

```
sol :=
dsolve({deqns,inits},{x(t),y(t)});
```

$$sol := \left\{\mathrm{x}(t) = \frac{\frac{1}{2}\,\%2\,\sqrt{1+4\,w^2} - \frac{1}{2}\,\%1 + \frac{1}{2}\,\%2 + \frac{1}{2}\,\%1\,\sqrt{1+4\,w^2} + 3\,w\,\%1 - 3\,w\,\%2}{\sqrt{1+4\,w^2}},\right.$$
$$\left.\mathrm{y}(t) = \frac{w\,\%1 - w\,\%2 + \frac{3}{2}\,\%2\,\sqrt{1+4\,w^2} + \frac{3}{2}\,\%1 - \frac{3}{2}\,\%2 + \frac{3}{2}\,\%1\,\sqrt{1+4\,w^2}}{\sqrt{1+4\,w^2}}\right\}$$
$$\%1 := e^{(-1/2\,(-5+\sqrt{1+4\,w^2})\,t)}$$
$$\%2 := e^{(1/2\,(5+\sqrt{1+4\,w^2})\,t)}$$

```
assign(sol);
```

```
f := unapply(x(t),t,w ):
```

```
g := unapply(y(t),t,w):
```

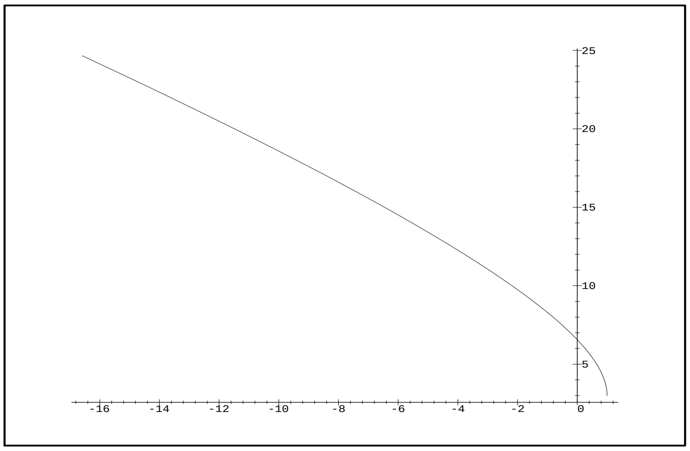
```
f(t,w) ;
```

$$\Big(\frac{1}{2}\,e^{(1/2\,(5+\sqrt{1+4\,w^2})\,t)}\,\sqrt{1+4\,w^2} - \frac{1}{2}\,e^{(-1/2\,(-5+\sqrt{1+4\,w^2})\,t)} + \frac{1}{2}\,e^{(1/2\,(5+\sqrt{1+4\,w^2})\,t)}$$
$$+ \frac{1}{2}\,e^{(-1/2\,(-5+\sqrt{1+4\,w^2})\,t)}\,\sqrt{1+4\,w^2} + 3\,w\,e^{(-1/2\,(-5+\sqrt{1+4\,w^2})\,t)}$$
$$- 3\,w\,e^{(1/2\,(5+\sqrt{1+4\,w^2})\,t)}\Big)\Big/\sqrt{1+4\,w^2}$$

```
f(1,1);
```

$$\frac{1}{5}\left(\frac{1}{2}e^{(5/2+1/2\sqrt{5})}\sqrt{5}+\frac{5}{2}e^{(5/2-1/2\sqrt{5})}-\frac{5}{2}e^{(5/2+1/2\sqrt{5})}+\frac{1}{2}e^{(5/2-1/2\sqrt{5})}\sqrt{5}\right)\sqrt{5}$$

```
plot([f(t,1 ),g(t,1 ),t=0..1 ]);
```



So we can see that [x,y] goes to [-infinity,infinity] as t goes to infinity when w is 1. By inspecting the form of the solutions, we could maybe figure out what happens to the solution as w changes.
The x coordinate looks like f(t,w)

```
f(t,w) ;
```

$$\Big(\frac{1}{2}e^{(1/2(5+\sqrt{1+4w^2})t)}\sqrt{1+4w^2}-\frac{1}{2}e^{(-1/2(-5+\sqrt{1+4w^2})t)}+\frac{1}{2}e^{(1/2(5+\sqrt{1+4w^2})t)}$$
$$+\frac{1}{2}e^{(-1/2(-5+\sqrt{1+4w^2})t)}\sqrt{1+4w^2}+3we^{(-1/2(-5+\sqrt{1+4w^2})t)}$$
$$-3we^{(1/2(5+\sqrt{1+4w^2})t)}\Big)\Big/\sqrt{1+4w^2}$$

```
limit(f(t,.4 ),t=infinity);
```

$$-\infty$$

Lets make a movie to study what happens as w decreases.

```
movie := NULL: for w from .5 by -.05 to .3
do movie := movie,

plot([f(t,w),g(t,w),t=0..4]) od :  w :=
'w':

plots[display]([movie],
insequence=true,axes=boxed);
```

So we can say after watching the movie that as w decreases from .5 to .1 [x(t),y(t)] 'seems' to change from moving towards [-infinity,infinity] to moving towards [infinity,-infinity]. We can check this by calculating some limits.

```
map(limit,[f(t,.5),g(t,.5)],t=infinity);
```

$$[-\infty, \infty]$$

```
map(limit,[f(t,.3),g(t,.3)],t=infinity);
```

$$[\infty, -\infty]$$

Problem.
This raises the question: at what value of w does the point make the 'flip flop'?
Solution 2:
We can do this analysis without dsolve, which is something of a black box. The idea is simple. Make a linear change of variable x = au + bv, v = cu + dv so that the system is transformed into a system that can be solved by inspection.

```
restart; with(linalg):
r := matrix(2,1,[x(t),y(t)]);
dr := map(diff,evalm(r),t);

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

So we want a change of coordinates so that the transformed system is diagonal. That is

```
chgcord := matrix(2,2,[a,b,c,d]);
newr := matrix(2,1,[u(t),v(t)]);
dnewr := map(diff,evalm(newr),t);
```

Since a,b,c and d are constant, the time derivatives of u,v and x,y are related by the same matrix.

```
evalm( r) =evalm(chgcord) *
evalm(newr);
```

```
evalm(dr) = evalm(chgcord) *
evalm(dnewr);
```

But we also are given that

```
evalm(dr) = evalm(A)*evalm(r);
```

Substituting the first and second equations into the third and solving for dnewr, we get

```
evalm(dnewr) =
```

```
evalm(chgcord)^(`-1`)*evalm(A)*evalm(chgcord)
*evalm(newr);
```

In order to be able to solve this by inspection, we would need the matrix product to be a diagonal matrix. That is, we want to diagonalize A. So get the eigenvalues for A and find an eigenvector for each of these. then normalize these and augment them to form Q = matrix(2,2,[a,b,c,d]); We could do this with eigenvects, but what the hey, since we're doing everything from scratch lets find the eigenvalues and eigenvectors without using eigenvects.

Get the characteristic equation and solve it.

```
eqn := det( A-lambda*diag( 1,1 ) );
```

```
eiv := solve(eqn,lambda);
```

Find an eigenvector for the first eigenvalue.

```
eqns := (
evalm((A-eiv[1]*diag(1,1))&*[x,y]));
```

```
assume(w>0);
```

```
soly := subs(x=1, op(solve(
eqns[1],{y})));
```

```
eig1 := [1,rhs(soly)];
```

```
u1 := map(simplify,
evalm(eig1/sqrt(dotprod(eig1,eig1))));
```

Now find a eigenvector for the second eigenvalue.

```
eqns := (
evalm((A-eiv[2]*diag(1,1))&*[x,y]));
```

```
soly := subs(x=1, op(solve(
eqns[1],{y})));
```

```
eig2 := [1,rhs(soly)];
```

```
u2 :=
map(simplify,(evalm(eig2/sqrt(dotprod(eig2,eig2)))));
```

```
Lets check that we have an eigenvector
```

```
belonging to eiv[1];
```

```
evalm(A&*eig1), map(expand,evalm(
(eiv[1])*eig1)) ;
```

```
Q := augment(u1,u2);
```

Now by inspection, the solution to the new system is

```
u(t) := c1* exp(eiv[1]*t);
```

```
v(t) := c2* exp(eiv[2]*t);
```

```
Soln := evalm(r) = evalm(Q)&*
matrix(2,1,[u(t),v(t)]) ;
```

or x

```
soln := evalm(rhs(Soln));
```

How can we check that these gives the same answer? First turn soln into a function we can evaluate easily.

```
r := map(unapply,evalm(soln),t,w);
```

```
r[1,1](1.,1.);
```

Next, use the intial values to get the values for c1 and c2.

```
c1 := 'c1':  c2 := 'c2':  sl := 'sl':
```

```
eqns := {r[1,1](0,w) = 1,r[2,1](0,w) =
3};
```

```
sl := solve (eqns,{c1,c2});
```

```
r1 := subs(sl,[r[1,1](t,w),r[();
```

```
plot([r1[1,1](t,1),r1[2,1](t,1)
,t=.1..1]);
```

Looks the same to me, I'm happy.

<u>Problem.</u> Solve the following continuous version of the flour barrel problem.
Three barrels contain x=10, y =20 and z =30 pounds of flour. at time t=0.
Each instant, 30 percent of the flour from barrel 1 is put into barrel 2 and
20 percent into barrel 3. Likewise, 40 percent of the flour in barrel 2 is distributed
30 percent to barrel 1 and 10 percent to barrel 3. Finally, 30 percent of the
flour in the last barrel is distributed 20 per cent to barrel 1 and 10 percent to barrel
2. What happens to the flour over time? Set up and solve a system of linear
differential equations. Try both methods outlined above.

• Back