Maple for Applications of Linear Algebra

Dr. Dale Siegel
Dr. Fred Peskoff

Paper

Computer algebra systems are capable of solving many problems that arise in linear algebra. The authors have developed a complete set of computer laboratory projects using Maple in linear algebra. These "hands on" exercises demonstrate Maple's ability and versatility in solving a wide variety of applications. The projects progress from precalculus mathematics to traditional linear algebra topics and then applications. The sequential titles of these projects are:
(1) Getting Started with Maple, (2) Functions, (3) Matrix Operations, (4) Matrix Form for Linear Systems of Equations, (5) Row-Reduced Echelon Form for Matrices, (6) Kernel and Range of Linear Systems, (7) Determinants, (8) Eigenvalues and Eigenvectors, (9) Mathod of Least Squares Fitting to Data, (10) Introduction to Linear Programaming, (11) Markov Chains, (12) Other Applications.

The Maple software has several packages, one of which is "linalg", the linear algebra package. Several of the functions (commands) in "linalg" will be displayed in solving select applications of linear algebra. Maple input commands are shown in bold with "prompt" symbols preceding them. Note: In Maple, colons at the end of a command line suppresses output, whereas semi-colons are for displaying output.

From Projects 4,5: Solving Systems of Equations

Load Maple's linear algebra package, which is necessary to use linear algebra functions.

> with(linalg);

Solve three equations in three unknowns using Gaussian elimination.

> equations:=[x-2*y=5 , 3*x+y-3*z=-11 , 6*y+3*z=0]:

> Ab:= genmatrix(equations , [x,y,z] , flag);

> G:= gausselim(Ab);

> solution:= backsub(G);

For the linear system of equations above, enter the coefficient matrix and right-hand side vector, separately. Then solve using: (1) linsolve function, (2) inverse of A , (3) row-reduced echelon form of [A | b].

> A:= array([[1,-2,0] , [3,1,-3] , [0,6,3]]):

> b:= vector([5,-11,0]):

> x:= linsolve(A,b);

> x:= evalm(inverse(A) &* b);

> rref(augment(A,b));

From project 9: Method of Least Squares

x	0	1	2	3	4
y	5	4	7	8	8

Find the least squares (regression) line $y = b_0 + b_1x$ to the data above using: (1) the least squares formula, (2) linalg's **leastsqrs** function

> with(linalg) :

> X:=matrix([[1,0] , [1,1] , [1,2] , [1,3] , [1,4]]) :

> Y:= vector(5 , [5,4,7,8,8]) :

> B:= inverse(multiply(transpose(X) , X)) &* transpose(X) &* Y :

> evalm(%) ;

> leastsqrs(X,Y) ;

Graph the data points and estimated regression line on the same axes. The "plots" package is needed for the display function and the "statplots" package is needed for scatterplot.

> with(plots) :

> with(statplots) :

```
> xdata:= [0,1,2,3,4] :
> ydata:= [5,4,7,8,8] :
> Points:= scatterplot(xdata , ydata) :
> Line:= plot(4.4 + x , x=-1..5 , y=--1..10 , color=red) :
> display({Points , Line}) ;
```

From project 10: Linear Programming

Maximize $Z = 3x + 5y$

subject to: $x \leq 4$

$2y \leq 12$

$3x + 2y \leq 18$

and $x \geq 0 , y \geq 0$

The "simplex" package has many functions for use in solving linear programming problems. Use its **feasible** command to check if a solution to the above problem exists.

```
> with(simplex) ;
> constraints:= [x<=4 , 2*y<=12 , 3*x+2*y<=18] :
> feasible(constraints , nonnegative) ;
```

Place this problem into matrix form via a simplex table(including slack variables *_SL1*, *_SL2*, *_SL3*) and manually apply the simplex algorithm.

```
> with(linalg) :
> A:= matrix(4,5, [1,0,1,0,0, 0,2,0,1,0, 1,2,0,0,1, -3,-5,0,0,0]) ;
> b:= vector(4, [4, 12, 18, 0]) ;
> T:= augment(A,b) ;
```

first iteration: y is the entering non-basic variable and *_SL2* is the leaving basic variable. Multiply row 2 by ½ so that pivot element becomes 1. Convert all other elements in the pivot column into 0 using linalg's **pivot** command ("simplex" package has **pivot** also).

```
> temp:= mulrow(T , 2 , 1/2) ;
> T1:= linalg[pivot](temp, 2 , 2) ;
```

second iteration: x is entering variable and $\_SL3$ is leaving variable.

> **T2:= pivot(mulrow(T1 , 3 , 1/3) , 3 , 1) ;**

Final solution from resulting table: Maximum $Z = 36$ occurring at $(x , y) = (2 , 6)$.

The "simplex" package has the functions **maximize** and **minimize** for solving linear programming problems directly.

> Z:= 3*x+5*y :

> xy:= maximize(Z , constraints) ;

> subs(xy , Z) ;

From project 11: Markov Chains

A square matrix can represent a transition matrix for a Markov chain (process) if its columns each sum to 1.

> with(linalg) :

> T:= matrix(3,3, [[0.85, 0.10, 0.05] , [.05, .80, .05] , [.1, .1, .9]]) ;

Verify that 1 is an eigenvalue of T (three ways).

> Id:= array(1..3, 1..3, identity) ;

> roots(det(matadd(scalarmul(Id , lambda) , T))) ;

> solve(charpoly(T , lambda)) ;

> eigenvals(T) ;

To find the steady-state vector (x) of the Markov process, one must solve $Tx = x$, or determine the eigenvector associated with the eigenvalue $\lambda = 1$. To be a state vector, the sum of its elements must equal 1.

> eig1:= nullspace(T - Id) ;

> eig1vector:= augment(op(eig1)) :

> x:= mulcol(eig1vector , 1 , 1/(sum(eig1vector[i,1] , i=1..3))) ;