Summary of Linear Algebra in **Derive** (DOS Version)

NUMERIC and SYMBOLIC Matrix and Vector Data Entry

Options Precision Approximate Enter	Future work done in numeric approximation mode.
Options Precision Exact Enter	Future work done in exact, symbolic mode.
Transfer Load Utility **vector** Enter	Load the extra package needed to do linear algebra.
Author **a := [[2,1,3], [-5,2,-1], [6,-1,4]]** Enter	A matrix is a vector of row vectors.
Author **B := [[4,3], [1,2]]** Enter	Derive will use only lower case names.
Declare Matrix **3** Tab **4** Enter	Alternate way to enter a matrix, especially if large.
2 -1 4 6 0 1 5 7 2 -9 0 3	Press the Enter key after each number typed.
Author **c :=** F3 Enter	Name the declared matrix. F3 key copies item down.

Highlight **a:=**···. Author F3 Edit using Home and arrow keys to change **a** to **e** and **6** to **7** Enter.

Author **w := [[9], [1], [6]]** Enter	Column matrix. Each row has one entry.
Author **[[5, -2, 4]]** Enter	Row matrix uses double [[]]. No commas in display.
Author **v := [5, -2, 4]** Enter	Single [] for vector. Display of a vector has commas.
Declare vectoR Dimension **3** Enter **2** Enter **-7** Enter **9** Enter	Useful for entering large vectors.
Author **u :=** F3	Name the declared vector.
Author **c** Ctrl-Enter	Simplify and display matrix *c* without symbolizing.
Author **c sub 3 sub 2** Enter	Symbolize matrix entry c_{32}. Use Alt-**v** instead of **sub**.
Simplify Enter	Calculate value of symbol(s). For decimals: approX.
Author **c sub [3,2] + element(c,3,2)** Ctrl-Enter	Alternate ways to access matrix entry.
Author **c sub 2** Enter Simplify Enter	Second row of matrix as a vector. Could use Alt-**v**.
Author **c' sub 3** Enter Simplify Enter	Third column of matrix as a vector, uses transpose.

NUMERIC and SYMBOLIC Matrix and Vector Operations (Try exact & approx modes.)

Use Author ··· Enter Simplify Enter or Author ··· Ctrl-Enter with each individual expression.

a + e, a - e, 2*w, 3a + 2e	Linear combinations. Use of * optional.
a*c, a c, v*a, a*w, c*a	Matrix products. May use space instead of *.
matprod (a, c, 2, 3)	Compute single entry in product *ac*
b^-1, b^3, b^-2, e^(-1)	Matrix inverse and powers. *e* is not invertible.
a^-1 * w, a^-1 * v, a^-1 * c	Last result is solution to a*x = c.
det(b), trace(a), rank(c), rank(e), c'	Back accent for transpose.
u . v, v . u, v . w, w . v, abs(v), abs(w), abs(b)	Dot product & norm.
dimension(c), dimension(c'), dimension(v)	# of rows of matrix. Size of vector.

Special Matrices and Functions

Author **ident := identity_matrix(3)** Enter Simplify Enter	Produces 3×3 identity matrix.
Author **zero := vector (vector (0, j, 4), i, 3)** Ctrl-Enter	Produces 3×4 zero matrix.

Author **minor (c, 2, 3)** Ctrl-Enter	Matrix with 2nd row and 3rd column of *c* deleted.
Author **cofactor (a, 2, 3)** Ctrl-Enter	Computer cofactor of entry a_{ij}.
Author **adjoint (a)** Ctrl-Enter	Adjoint matrix, transpose of matrix of cofactors.

NUMERIC and SYMBOLIC Linear Algebra Algorithms (Use Simplify or Ctrl-Enter as desired.)

Author **[2x + 3y = 6, 4x + 5y = 7]** Enter soLve Enter	Solve system of linear equations.
Author **a * [x, y, z] = v** Enter Simplify Enter soLve Enter	Alternate way to enter system.
Author **[2x + 3y = 6, 4x + 6y = 12]** Enter soLve Enter	@1 is an arbitrary parameter.
Author **[2x+3y+z=6, 4x+5y+6z=7]** Enter soLve Enter **x** Enter **y** Enter	# unknowns > # eqn's

scale_element (c, 1, 1/2)	Compute matrix resulting from row multiplication.
Author **c :=** F3 Enter	Copy resulting matrix to *c*.
subtract_elements (c, 3, 1, 2)	Result of subtracting twice row 1 from row 3.
swap_elements (c, 2, 3)	Compute matrix resulting from row swap.

Author **row_reduce (c)** Enter Simplify Enter	Compute reduced row-echelon form.
Author **row_reduce (a, w)** Enter Simplify Enter	Reduce the augmented matrix.
row_reduce (e), (a, v), (a, c), (e, w), (a, ident)	Try other examples separately.

charpoly (b, t)	Compute characteristic polynomial of *b* using variable *t*.
eigenvalues (b, t)	Compute eigenvalues of matrix *b* using variable *t*.
exact_eigenvector (b, 5)	Compute eigenvector corresponding to exact eigenvalue 5.
approx_eigenvector (b, 5.001) Enter approX	Use close-to-eigenvalue, but NOT EXACT.

Summary of Linear Algebra in Derive for Windows

NUMERIC and SYMBOLIC Matrix and Vector Data Entry

Declare AlgebraState Precision	Future work done in numeric approximation mode
Approximate or Exact Enter	or in exact, symbolic mode.
File Load Utility **vector.mth** Enter	Load the extra package needed to do linear algebra.
Ctrl-A **a := [[2,1,3], [-5,2,-1], [6,-1,4]]** OK	A matrix is a vector of row vectors.
Author Expression **B := [[4,3], [1,2]]** Enter	Derive will use only lower case names.
Author Matrix **3** Tab **4** Enter	Alternate way to enter a matrix, especially if large.
2 -1 4 6 0 1 5 7 2 -9 0 3 OK	Press the Tab key after each number typed.
Ctrl-A **c :=** F3 Enter	Name the declared matrix. F3 key copies item down.

Highlight **a:=**···. Ctrl-A F3 Edit using Home and arrow keys to change **a** to **e** and **6** to **7** Enter.

Ctrl-A **w := [[9], [1], [6]]** Enter	Column matrix. Each row has one entry.
Ctrl-A **[[5, -2, 4]]** Enter	Row matrix. No commas in matrix display.
Ctrl-A **v := [5, -2, 4]** Enter	A vector display has commas.
Author Vector Elements: **3** Enter **2** Tab **-7** Tab **9** Tab OK	Useful for entering large dimension vectors.
Ctrl-A **u :=** F3 Enter	Name the declared vector.
Ctrl-A **c** Simplify	Simplify and display matrix *c* without symbolizing.
Ctrl-A **c sub 3 sub 2** Enter	Symbolize matrix entry c_{32}. Use Ctrl-B instead of **sub**.
Simplify Basic Enter	Calculate value of symbol(s). For decimals: Approximate.
Ctrl-A **c sub [3,2] + element(c,3,2)** Alt-S	Alternate ways to access matrix entry.
Ctrl-A **c sub 2** Enter Simplify Basic Enter	Second row of matrix as a vector. Could use Ctrl-B.
Ctrl-A **c' sub 3** Enter Ctrl-B Enter	3rd column as a vector; uses back accent for transpose.

NUMERIC and SYMBOLIC Matrix and Vector Operations (Try exact & approx modes.)

Use Ctrl-A ··· Enter Ctrl-B Enter or Ctrl-A ··· Alt-S with each individual expression.

a + e, a - e, 2*w, 3a + 2e	Linear combinations. Use of * optional.
a*c, a c, v*a, a*w, c*a	Matrix products. Last product incompatible.
matprod (a, c, 2, 3)	Compute single entry in product *ac*
b^-1, b^3, b^-2, e^(-1)	Matrix inverse and powers. *e* is not invertible.
a^-1 * w, a^-1 * v, a^-1 * c	Last result is solution to a*x = c.
det(b), trace(a), rank(c), rank(e), c'	Back accent for transpose.
u . v, v . u, v . w, w . v, abs(v), abs(w), abs(b)	Dot product and norm.
dimension(c), dimension(c'), dimension(v)	# of rows of matrix. Size of vector.

Special Matrices and Functions

Ctrl-A **ident := identity_matrix(3)** Enter Ctrl-B Enter	Produces 3×3 identity matrix.
Ctrl-A **zero := vector (vector (0, j, 4), i, 3)** Alt-S	Produces 3×4 zero matrix.

Ctrl-A **minor (c, 2, 3)** Alt-S	Matrix with 2nd row and 3rd column of *c* deleted.
Ctrl-A **cofactor (a, 2, 3)** Alt-S	Computer cofactor of entry a_{ij}.
Ctrl-A **adjoint (a)** Alt-S	Adjoint matrix, transpose of matrix of cofactors.

NUMERIC and SYMBOLIC Linear Algebra Algorithms

Author each **Expression** (Ctrl-A), and **Simplify** using Simplify, Alt-S, or Ctrl-B as appropriate.

Compute a numeric approximation of an expression using Simplify Approximate, ≈, or Ctrl-G.

Solve by using Solve Algebraic Variable: [x, y] Alt-S		Variables are [x, y, z] in second case.
[2x + 3y = 6, 4x + 5y = 7]	**Solve**	Solve system of linear equations.
a * [x, y, z] = v and **Simplify**	**Solve**	Alternate way to enter system.
[2x + 3y = 6, 4x + 6y = 12]	**Solve**	@1 is an arbitrary parameter.
[2x+3y+z=6, 4x+5y+6z=7]	**Solve**	# unknowns > # eqn's

scale_element (c, 1, 1/2)	Compute matrix resulting from row multiplication.
c := F3	Copy resulting matrix to *c*.
subtract_elements (c, 3, 1, 2)	Result of subtracting twice row 1 from row 3.
swap_elements (c, 2, 3)	Compute matrix resulting from row swap.

row_reduce (c)	Compute reduced row-echelon form.
row_reduce (a, w)	Reduce the augmented matrix.
row_reduce (e), (a, v), (a, c), (e, w), (a, ident)	Try other examples separately.

charpoly (b, t)	Compute characteristic polynomial of *b* using variable *t*.
eigenvalues (b, t)	Compute eigenvalues of matrix *b* using variable *t*.
exact_eigenvector (b, 5)	Compute eigenvector corresponding to exact eigenvalue 5.
approx_eigenvector (b, 5.001) ≈ (Approximate)	Use close-to-eigenvalue, but NOT EXACT.

1-11-02 def