Part IV. Linear Differential Equations

Section 4. Matrix Methods

In this section, you will learn enough about vectors and matrices in Maple to use them to obtain vector solutions to linear systems of the form

$$x' = a\,x + b\,y$$
$$y' = c\,x + d\,y$$

where *x,* and *y* are functions of *t* and *a, b, c, d* are constants. Three dimensional systems will also be considered. Vectors and matrices arise naturally by replacing scalar equations with one vector equation v' = A v, where v is a vector valued function having the unknown functions as its components and A is a square matrix.

```
Warning: It is assumed that the reader knows what eigenvectors
and eigenvalues are and understands the role they play in
formulating solutions to systems of differential equations.
(See Ledder, Chapter 6.) Our principal goal is to demonstrate
how to handle vectors, matrices, eigenvectors, etc. so that
Maple can be applied successfully to the analysis of such
systems.
```

The LinearAlgebra *package: Matrices and Vectors*

Maple's **LinearAlgebra** package has over 100 procedures that operate on vectors and matrices. We will use only a handful of them, and all of their names should be familiar to you.

```
> with(LinearAlgebra):
```

Unlike most names in Maple, **LinearAlgebra** procedure names are all capitalized, and none of them are abbreviated. This is to avoid confusion with the names of similar procedures in the deprecated Maple package named **linalg**. So, for example, a 2 x 2 matrix named A can be entered by applying the **Matrix** procedure to a list of lists like this,

```
> A := Matrix( [ [1,2], [3,4] ] );
```

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

The determinant of A is obtained with the entry

```
Determinant(A)
```

```
> Determinant(A);
```

```
-2
```

Similarly, the entry

```
CharacteristicPolynomial(A,t)
```

outputs the characteristic polynomial of A using *t* as the variable.

```
> CharacteristicPolynomial(A,t);
```

$$t^2 - 5\,t - 2$$

A vector is defined by applying the **Vector** procedure to a list.

```
> v := Vector([-2,3]);
```

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

As a default, Maple works with vectors as columns. A row vector can be entered like this.

```
> Vector[row]([-2,3]);
```

$$[-2, 3]$$

Matrix A and (column) vector *v* are multiplied by putting a period between them

```
A.v
```

Read this product as "A times v", and not as "A dot v". The product is a column vector, as it should be. We name it w.

```
> w := A.v;
```

$$w := \begin{bmatrix} 4 \\ 6 \end{bmatrix}$$

The dot product of vector *v* and vector *w* is also denoted with a period

```
v.w
```

However, this product should be read as "v dot w", and not as "v times w". The output is a scalar, as it should be.

```
> v.w;
```

$$10$$

We will use lower case letters to represent vectors and upper case letters for matrices, so there should be no confusion. Be careful, however, because (unlike what you may be accustomed to) in Maple a column vector is not the same thing as a column matrix.

> *Maple would like you to think of a matrix as a linear transformation and a vector as an object that the matrix transforms.*

Consider the following example. We define V to be the the column matrix having the same entries as the vector v defined above. This can be done as follows.

```
> V := Matrix( [ [-2] , [3] ] );
```

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

Judging from the appearance of the output, *v* and V are the same objects, but Maple knows the difference. Because V is a matrix, the entry

```
V.v
```

tells Maple to form a matrix product, but it cannot. This is because V and *v* are not compatible for matrix multiplication. V, as a transformation, cannot transform v. (Read the Error message).

```
> V.v;
```

```
Error, (in LinearAlgebra:-MatrixVectorMultiply) vector dimension (2) must be the same as the matrix column dimension (1)
```

However, the transpose of V, entered as

```
Transpose(V)
```

is the right shape for matrix multiplication times *v*.

```
> Transpose(V);
```

$$\begin{bmatrix} -2 & 3 \end{bmatrix}$$

The product is a 1 dimensional column vector.

```
> Transpose(V).v;
```

$$\begin{bmatrix} 13 \end{bmatrix}$$

Compare this to the dot product of *v* with itself. The output is a scalar. (Recall that the dot product of two vectors is also referred to as the scalar product.)

```
> v.v;
```

$$13$$

The dot product of *v* with itself is the square of its length. Thus the length of *v* can be calculated as

```
sqrt(v.v)
```

It can also be calculated as the Euclidean norm of *v*, using the entry

```
Norm(v,2)
```

```
> Norm(v,2);
```

$$\sqrt{13}$$

The entry

```
Norm(v)
```

calculates what is called the "infinity" or "max" norm of *v* (the maximum of the absolute values of its entries).

```
> Norm(v);
```

$$3$$

Maple's "column entry" method for vectors and matrices

Maple developers have come up with a quick and easy way to enter vectors and matrices in terms of their columns. Many textbooks use the notation v = < a , b , c > for a column vector. Maple allows us to use the same notation at an input prompt to enter a Vector.

```
> v := <a,b,c>;
```

$$v := \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Many textbooks also represent a matrix C in terms of its column vectors c1, c2, c3 using the notation

$$C = < c1 \mid c2 \mid c3 > .$$

Maple permits the same notation in an input entry.

```
> C := < <1,2> | <3,4> | <5,6> >;
```

$$C := \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Note that matrix C and vector *v* can be multiplied, in the order C *v* .

```
> C.v;
```

$$\begin{bmatrix} a + 3\,b + 5\,c \\ 2\,a + 4\,b + 6\,c \end{bmatrix}$$

But they cannot be multiplied in the other order.

```
> v.C;
```

```
Error, (in LinearAlgebra:-VectorMatrixMultiply) invalid input:
`LinearAlgebra:-VectorMatrixMultiply` expects its 1st argument, v, to be of
type Vector[row] but received Vector[Column]( 1..3,[ a, b, c] , datatype =
anything, storage = rectangular, order = C_order )
```

The entry **C[i , j]** outputs the *i j* entry of matrix C. The entry **v[k]** is the kth component of the vector *v*.

```
> C[2,2], v[2];
```

$$4, b$$

The output for **v[2]** is *b* because of the last definition of v appearing above.

To get the jth column of matrix C, enter **Column(C , j)**. The entry **Row(C , i)** outputs its ith row. Both of these outputs are considered by Maple to be vectors.

```
> Column(C,3), Row(C,2);
```

$$\begin{bmatrix} 5 \\ 6 \end{bmatrix}, [2, 4, 6]$$

The difference between a row vector and a row matrix can be seen in the output display. Compare, for example, the second term in the last output sequence to the output when **Row(C , 2)** is converted into a Matrix using the **convert** procedure.

```
> convert(Row(C,2),Matrix);
```

$$[2 \quad 4 \quad 6]$$

Maple allows multiple assignments using sequences on both sides of the assignment operator := . For example, the following input makes three column vector assignments using only one entry.

```
> a1, a2, a3 := <3,0,2>, <2,-1,1>, <1,-1,1>;
```

$$a1, a2, a3 := \begin{bmatrix} 3 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

The matrix A = < a1 | a2 | a2 >, with these vectors in its columns, can then be entered like this.

```
> A := <a1|a2|a3>;
```

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

Augmenting and stacking (*not needed later, but kind of neat anyway*)

If you would like to augment the matrix A with the column vector < x, y, z >, do it like this.

```
> A_augmented := < A | <x,y,z> >;
```

$$A\_augmented := \begin{bmatrix} 3 & 2 & 1 & x \\ 0 & -1 & -1 & y \\ 2 & 1 & 1 & z \end{bmatrix}$$

If you would like to stack a row vector along the bottom of the matrix A_augmented, do it like this.

```
> A_augmented_stacked := < A_augmented, Vector[row]([p,q,r,s]) >;
```

$$A\_augmented\_stacked := \begin{bmatrix} 3 & 2 & 1 & x \\ 0 & -1 & -1 & y \\ 2 & 1 & 1 & z \\ p & q & r & s \end{bmatrix}$$

And then, if you need to calculate the determinant of the 3 x 3 submatrix of A_augmented_stacked that is obtained by deleting its first row and its first column, do it like this.

```
> A_augmented_stacked[2..4,2..4];
  Determinant(%);
```

$$\begin{bmatrix} -1 & -1 & y \\ 1 & 1 & z \\ q & r & s \end{bmatrix}$$

$$z\,r + r\,y - q\,z - q\,y$$

Eigenvectors and eigenvalues

The matrix A (above) is square. It has eigenvectors and eigenvalues. Let's find them.

Textbook method, using Maple

First find A's characteristic polynomial, and then the roots of that polynomial (the eigenvalues). Assign the name lambda to the sequence of roots.

```
> CharacteristicPolynomial(A,t);
  lambda := solve(%);
```

$$t^3 - 3\,t^2 - 2\,t + 2$$

$$\lambda := -1,\ 2 + \sqrt{2},\ 2 - \sqrt{2}$$

Use the **NullSpace** procedure to obtain bases for the three eigenspaces. Recall that the kth eigenspace is the null space of the matrix

```
A - lambda[k]*Identity
```

where "Identity" is the 3 x 3 identity matrix.

```
> 'NullSpace( A - lambda[k]*IdentityMatrix(3) )' $ k=1..3;
```

$$\left\{\begin{bmatrix} -\frac{1}{2} \\ 1 \\ 0 \end{bmatrix}\right\},\ \left\{\begin{bmatrix} \frac{1+\sqrt{2}}{(3+\sqrt{2})(-1+\sqrt{2})} \\ -\frac{1}{3+\sqrt{2}} \\ 1 \end{bmatrix}\right\},\ \left\{\begin{bmatrix} -\frac{-1+\sqrt{2}}{(-3+\sqrt{2})(1+\sqrt{2})} \\ \frac{1}{-3+\sqrt{2}} \\ 1 \end{bmatrix}\right\}$$

The output for **NullSpace** is a set containing a basis for the null space, in column vector form. If further computations are required it would probably be wise to convert the eigenvalues and eigenvectors to floating point form.

```
> eigenvalues, evalf[3](lambda);
  eigenvectors, evalf[3](%%);
```

$$eigenvalues,\ -1.,\ 3.41,\ 0.59$$

$$eigenvectors,\ \left\{\begin{bmatrix} -0.500 \\ 1. \\ 0. \end{bmatrix}\right\},\ \left\{\begin{bmatrix} 1.33 \\ -0.227 \\ 1. \end{bmatrix}\right\},\ \left\{\begin{bmatrix} 0.107 \\ -0.629 \\ 1. \end{bmatrix}\right\}$$

Maple's direct method

Use the **LinearAlgebra** procedures named **Eigenvalues** and/or **Eigenvectors**.

To obtain A's eigenvalues apply the **Eigenvalues** procedure to A. The output is a column vector containing the eigenvalues, repeated if necessary.

```
> Eigenvalues(A);
```

$$\begin{bmatrix} -1 \\ 2+\sqrt{2} \\ 2-\sqrt{2} \end{bmatrix}$$

The **Eigenvectors** procedure applied to A outputs a sequence. The first term is a column vector containing the eigenvalues, the second term is a matrix containing the corresponding eigenvectors in its columns. We have assigned the names lambda and EV to the two terms in the output.

```
> lambda, EV := Eigenvectors(A);
```

$$\lambda, EV := \begin{bmatrix} 2+\sqrt{2} \\ 2-\sqrt{2} \\ -1 \end{bmatrix}, \begin{bmatrix} \frac{1+\sqrt{2}}{(3+\sqrt{2})(-1+\sqrt{2})} & \frac{1-\sqrt{2}}{(3-\sqrt{2})(-1-\sqrt{2})} & \frac{-1}{2} \\ -\frac{1}{3+\sqrt{2}} & -\frac{1}{3-\sqrt{2}} & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

To see simplified "exact" versions of the eigenvectors, **map** a procedure called **rationalize** into the matrix EV.

```
> map(rationalize,EV);
```

$$\begin{bmatrix} \frac{1}{7}(1+\sqrt{2})(1+2\sqrt{2}) & \frac{1}{7}(-1+\sqrt{2})(-1+2\sqrt{2}) & \frac{-1}{2} \\ -\frac{3}{7}+\frac{1}{7}\sqrt{2} & -\frac{3}{7}-\frac{1}{7}\sqrt{2} & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Not simple enough? Use **map** to apply the **expand** procedure to the entries in the last output.

```
> map(expand,%);
```

$$\begin{bmatrix} \frac{5}{7}+\frac{3}{7}\sqrt{2} & \frac{5}{7}-\frac{3}{7}\sqrt{2} & \frac{-1}{2} \\ -\frac{3}{7}+\frac{1}{7}\sqrt{2} & -\frac{3}{7}-\frac{1}{7}\sqrt{2} & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Eigenvector solutions to v' = A v, *the* 2 x 2 *case*

Let's begin by solving a simple 2 x 2 system

$$x' = x + 2y$$
$$y' = x - y$$

using eigenvectors. This is equivalent to v' = A v where v = < x , y > and

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

Start with the eigenvectors and eigenvalues of A.

```
> A := <<1,1>|<2,-1>>;
  lambda,EV := Eigenvectors(A);
```

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

$$\lambda, EV := \begin{bmatrix} \sqrt{3} \\ -\sqrt{3} \end{bmatrix}, \begin{bmatrix} \frac{2}{\sqrt{3} - 1} & \frac{2}{-\sqrt{3} - 1} \\ 1 & 1 \end{bmatrix}$$

Let E[k] be the kth eigenvector. We can define both E[1] and E[2] with the following input.

```
> E[1],E[2] := Column(EV,1),Column(EV,2);
```

$$E_1, E_2 := \begin{bmatrix} \frac{2}{\sqrt{3} - 1} \\ 1 \end{bmatrix}, \begin{bmatrix} \frac{2}{-\sqrt{3} - 1} \\ 1 \end{bmatrix}$$

Now let v1 and v2 be the two vector solutions (straight line trajectories in this case). Recall that vk is defined using the formula

$$\mathrm{vk} = \mathbf{e}^{(\lambda_k t)} E_k$$

```
> v1,v2 := 'exp(lambda[k]*t)*E[k]' $ k=1..2;
```

$$v1, v2 := \begin{bmatrix} \frac{2\,\mathbf{e}^{(\sqrt{3}\ t)}}{\sqrt{3} - 1} \\ \mathbf{e}^{(\sqrt{3}\ t)} \end{bmatrix}, \begin{bmatrix} \frac{2\,\mathbf{e}^{(-\sqrt{3}\ t)}}{-\sqrt{3} - 1} \\ \mathbf{e}^{(-\sqrt{3}\ t)} \end{bmatrix}$$

• The single quotes around exp(lambda[k]*t)*E[k] are needed to prevent premature evaluation.

Now that we have v1 and v2, the easiest way to obtain solutions to the IVP

$$\mathrm{v}' = \mathrm{A}\ \mathrm{v},\ \mathrm{v}(0) = \mathrm{v0}$$

is to use the formula

$$\mathrm{v} = X(t)\ X(0)^{(-1)}\ v0$$

where X(t) is the fundamental matrix having v1 and v2 as its columns. See Ledder, Section A.5.

```
> <v1|v2>;
  X := unapply(%,t):
```

$$\begin{bmatrix} \frac{2\,e^{(\sqrt{3}\,t)}}{\sqrt{3}-1} & \frac{2\,e^{(-\sqrt{3}\,t)}}{-\sqrt{3}-1} \\ e^{(\sqrt{3}\,t)} & e^{(-\sqrt{3}\,t)} \end{bmatrix}$$

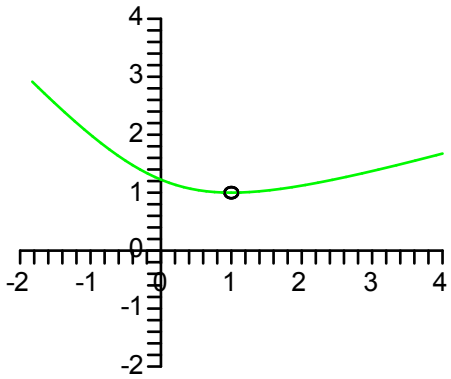
Note that X is defined as a matrix-valued function. This makes it easy to use the solution formula displayed above.

For example, the following entry defines the vector function v as the solution to

$$v' = A\,v\,,\ \ v(0) = \langle 1\,,\,1 \rangle\,,$$

then plots the trajectory and the initial point. The output for v is suppressed because it's a mess.

```
> v := X(t).X(0)^(-1).<1,1>:
  plot( [[v[1],v[2],t=-1..1],[[1,1]]], style=[line,point],
        color=[green,black], symbol=circle, symbolsize=16,
        view=[-2..4,-2..4]); Trajectory := %:
```



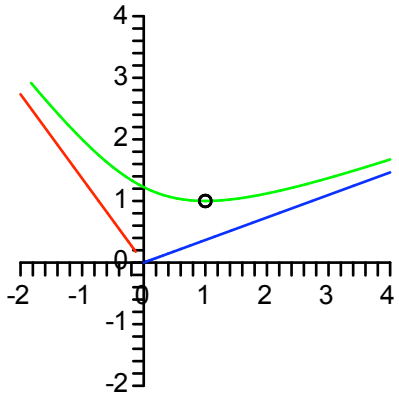
Here is the solution vector v, simplified.

```
> v := simplify(v);
```

$$v := \begin{bmatrix} \frac{1}{6}\left(e^{(\sqrt{3}\,t)}\sqrt{3} + 3\,e^{(\sqrt{3}\,t)} + e^{(-\sqrt{3}\,t)}\sqrt{3} - 3\,e^{(-\sqrt{3}\,t)}\right)\sqrt{3} \\ \frac{1}{2}\,e^{(\sqrt{3}\,t)} + \frac{1}{2}\,e^{(-\sqrt{3}\,t)} \end{bmatrix}$$

And here is a picture of the solution trajectory and two eigenvector trajectories, one blue, one red.

```
> EV1 := plot( [v1[1],v1[2],t=-5..1], color=blue):
  EV2 := plot( [v2[1],v2[2],t=-5..1], color=red):
  plots[display]( Trajectory, EV1, EV2 , scaling=constrained);
```



The matrix exponential

The **LinearAlgebra** procedure named **MatrixExponential**, when applied to A as

```
MatrixExponential(A,t)
```

outputs the matrix $X(t)\,X(0)^{(-1)}$ that was found above using the eigenvector solutions. We enter it below, applied the the 2 x 2 matrix A defined above, and define Exp_At to be a simplified version.

```
> MatrixExponential(A,t):
  Exp_At := map(collect,simplify(%),exp);
```

$$Exp\_At := \left[\left[\left(\frac{1}{2} + \frac{1}{6}\sqrt{3} \right) e^{(\sqrt{3}\,t)} + \left(\frac{1}{2} - \frac{1}{6}\sqrt{3} \right) e^{(-\sqrt{3}\,t)}, -\frac{1}{3} e^{(-\sqrt{3}\,t)} \sqrt{3} + \frac{1}{3} e^{(\sqrt{3}\,t)} \sqrt{3} \right], \left[-\frac{1}{6} e^{(-\sqrt{3}\,t)} \sqrt{3} + \frac{1}{6} e^{(\sqrt{3}\,t)} \sqrt{3}, \left(\frac{1}{2} - \frac{1}{6}\sqrt{3} \right) e^{(\sqrt{3}\,t)} + \left(\frac{1}{2} + \frac{1}{6}\sqrt{3} \right) e^{(-\sqrt{3}\,t)} \right] \right]$$

How did you do that?

Answer: The **simplify** procedure was applied first, then we factored out the exponential terms by mapping **collect/exp** into the simplified version.

Compare Exp_At to X(t)*X(0)^(-1), also simplified, then collected on exp.

```
> simplify(X(t).X(0)^(-1)):
  map(collect,%,exp);
```

$$\left[\left[\frac{1}{6} e^{(\sqrt{3}\,t)} \sqrt{3}\,(\sqrt{3}+1) + \frac{1}{6}(\sqrt{3}-1)\sqrt{3}\, e^{(-\sqrt{3}\,t)}, -\frac{1}{3} e^{(-\sqrt{3}\,t)} \sqrt{3} + \frac{1}{3} e^{(\sqrt{3}\,t)} \sqrt{3} \right], \left[-\frac{1}{6} e^{(-\sqrt{3}\,t)} \sqrt{3} + \frac{1}{6} e^{(\sqrt{3}\,t)} \sqrt{3}, \left(\frac{1}{2} - \frac{1}{6}\sqrt{3} \right) e^{(\sqrt{3}\,t)} + \left(\frac{1}{2} + \frac{1}{6}\sqrt{3} \right) e^{(-\sqrt{3}\,t)} \right] \right]$$

A 3 dimensional example, approximate solutions

We will now compute three linearly independent eigenvector solutions to v' = A v for the 3 x 3 matrix A defined several pages ago and redefined below.

```
> A := <a1|a2|a3>;
```

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

Because this is a small system, exact formulas can be obtained. However we will make our calculations in floating point form. We do this for three reasons.

1. Floating point calculations are easier to make, and are more relevant for real world applications.

2. It will give us some experience with error analysis.

3. If exact solution formulas are required, it is easier to get them directly from the matrix exponential or by simply applying **dsolve**.

We will work to the default 10 digit accuracy.

Here, once more, are A's eigenvalues and eigenvectors.

```
> lambda,EV := Eigenvectors(A);
```

$$\lambda, EV := \begin{bmatrix} 2+\sqrt{2} \\ 2-\sqrt{2} \\ -1 \end{bmatrix}, \begin{bmatrix} \frac{1+\sqrt{2}}{(3+\sqrt{2})(-1+\sqrt{2})} & \frac{1-\sqrt{2}}{(3-\sqrt{2})(-1-\sqrt{2})} & \frac{-1}{2} \\ -\frac{1}{3+\sqrt{2}} & -\frac{1}{3-\sqrt{2}} & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Replace lambda with floating point approximations (10 digit).

```
> lambda := evalf(lambda);
```

$$\lambda := \begin{bmatrix} 3.414213562 \\ 0.585786438 \\ -1. \end{bmatrix}$$

Use a **for..do** loop to define E[k] as the kth eigenvector, in floating point form.

```
> for k from 1 to 3
    do
      E[k] := evalf(Column(EV,k)):
    end do:
  unassign('k');
```

Define the three eigenvector solutions.

```
> v1, v2, v3 := 'exp(lambda[k]*t)*E[k]'$k=1..3;
```

$$v1, v2, v3 := \begin{bmatrix} 1.320377242\, e^{(3.414213562\, t)} \\ -0.2265409197\, e^{(3.414213562\, t)} \\ 1.\, e^{(3.414213562\, t)} \end{bmatrix}, \begin{bmatrix} 0.1081941875\, e^{(0.585786438\, t)} \\ -0.6306019373\, e^{(0.585786438\, t)} \\ 1.\, e^{(0.585786438\, t)} \end{bmatrix}, \begin{bmatrix} -0.5000000000\, e^{(-1.\, t)} \\ 1.\, e^{(-1.\, t)} \\ 0. \end{bmatrix}$$

Make the fundamental matrix function X(t).

```
> <v1|v2|v3>;
  X := unapply(%,t):
```

$$\begin{bmatrix} 1.320377242\, e^{(3.414213562\, t)} & 0.1081941875\, e^{(0.585786438\, t)} & -0.5000000000\, e^{(-1.\, t)} \\ -0.2265409197\, e^{(3.414213562\, t)} & -0.6306019373\, e^{(0.585786438\, t)} & 1.\, e^{(-1.\, t)} \\ 1.\, e^{(3.414213562\, t)} & 1.\, e^{(0.585786438\, t)} & 0. \end{bmatrix}$$

Obtain the solution to v' = A v satisfying v(0) = < 2, 4, 1 >.

```
> v := X(t).X(0)^(-1).<2,4,1>;
```

$$v := \begin{bmatrix} 3.927955574\, e^{(3.414213562\, t)} - 0.2136698588\, e^{(0.585786438\, t)} - 1.714285714\, e^{(-1.\, t)} \\ -0.6739306313\, e^{(3.414213562\, t)} + 1.245359201\, e^{(0.585786438\, t)} + 3.428571430\, e^{(-1.\, t)} \\ 2.974873732\, e^{(3.414213562\, t)} - 1.974873732\, e^{(0.585786438\, t)} \end{bmatrix}$$

The next entry checks the initial value.

```
> eval(v,t=0);
```

$$\begin{bmatrix} 2.000000001 \\ 4.000000000 \\ 1.000000000 \end{bmatrix}$$

Error analysis over the interval t = 0 to t = 4

The vector function v defined above does not satisfy the differential equation exactly. Let's see how large the error is over the time interval from 0 to 4. The error can be analyzed as follows.

First calculate the "error vector" v' - A v. (The differentiation procedure must be mapped into v.)

```
> ErrorVector := map(diff,v,t) - A.v;
```

$$ErrorVector := \begin{bmatrix} 5\,10^{-10}\,e^{(0.585786438\,t)} - 4\,10^{-9}\,e^{(-1.\,t)} \\ -6\,10^{-10}\,e^{(0.585786438\,t)} \\ -1\,10^{-8}\,e^{(3.414213562\,t)} - 2\,10^{-9}\,e^{(-1.\,t)} \end{bmatrix}$$

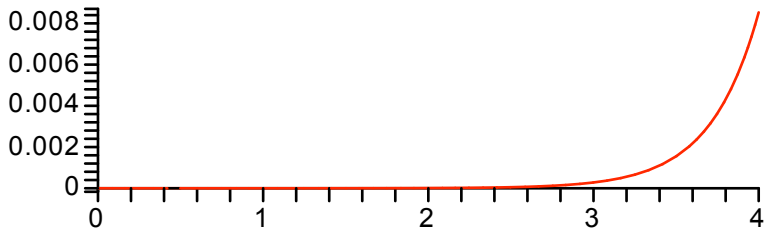
Then define Error as a function whose value at t is the largest of the absolute errors in the three components of ErrorVector. This requires the "infinity" norm.

```
> Error := Norm(ErrorVector);
```

$$Error := \max\left(\left| 5\,10^{-10}\,e^{(0.585786438\,t)} - 4\,10^{-9}\,e^{(-1.\,t)} \right|, 6\,10^{-10}\,e^{(0.585786438\,\Re(t))}, \left| 1\,10^{-8}\,e^{(3.414213562\,t)} + 2\,10^{-9}\,e^{(-1.\,t)} \right| \right)$$

Now graph Error over the interval from 0 to 4.

```
> plot(Error,t=0..4);
```



This looks very good. Although the error clearly grows exponentially, it is only 0.008 at $t = 4$ (years, maybe?). We say years because at $t = 4$ the approximate solution vector is over 4 million units long. (The length calculation uses the Euclidean norm.)

```
> length_of_approx_soln = Norm(eval(v,t=4),2);
```

$$length\_of\_approx\_soln = 4.25652130383485835\,10^{6}$$

Thus the maximum relative error could be as small as 2 parts in 10^{9}.