DonnishJournals 2041-1189

Donnish Journal of Educational Research and Reviews.
Vol 1(2) pp. 023-029 December, 2014.
http://www.donnishjournals.org/djerr
Copyright © 2014 Donnish Journals

Original Research Paper

Discrete Data Interpolation and Matrices with MAXIMA

Niyazi Ari and Savaş Tuylu*

Computer Science, Nigerian Turkish Nile University, Nigeria.

Accepted 24th November, 2014.

The methods of calculus lie at the heart of the physical sciences and engineering. Maxima can help you make faster progress, if you are just learning calculus. The examples in this research paper will offer an opportunity to see some Maxima tools in the context of simple examples, but you will likely be thinking about much harder problems you want to solve as you see these tools used here. This research paper includes Discrete Data Interpolation and Matrices with MAXIMA.

Keywords: Interpolation, Discrete Data Interpolation, Lagrange interpolation, Polynomial interpolation, Cubic-Spline interpolation, Rational interpolation, Matrices, Eigenvalue, Eigenvector, Maxima.

1. INTRODUCTION

This research paper includes Discrete Data Interpolation and Matrices with MAXIMA. Maxima is a system for the manipulation of symbolic and numerical expressions, including differentiation, integration, Taylor series, Laplace transforms, ordinary differential equations, systems of linear equations, polynomials, sets, lists, vectors, matrices, tensors, and more. Maxima yields high precision numeric results by using exact fractions, arbitrary precision integers, and variable precision floating point numbers. Maxima can plot functions and data in two and three dimensions. Maxima source code can be compiled on many computer operating systems, including Windows, Linux, and MacOS X. The source code for all systems and precompiled binaries for Windows and Linux are available at the SourceForge file manager.

Maxima is a descendant of Macsyma, the legendary computer algebra system developed in the late 1960s at the Massachusetts Institute of Technology. It is the only system based on that effort still publicly available and with an active user community, thanks to its open source nature. Macsyma was revolutionary in its day, and many later systems, such as Maple and Mathematica, were inspired by it. The Maxima branch of Macsyma was maintained by William Schelter from 1982 until he passed away in 2001. In 1998 he obtained permission to release the source code under the GNU General Public License (GPL). It was his efforts and skills that made the survival of Maxima possible.
MAXIMA has been constantly updated and used by researcher and engineers as well as by students.

2. DISCRETE DATA INTERPOLATION

Discrete data sets are generally used for engineering applications. So, the function that fits the data is required in order to estimate outcomes. If the estimated function passes through the given points, this approach is an interpolation method. If the estimated function is a good fit to the given points, this is the curve fitting approach.

There are many different interpolation methods, some of which are described below within MAXIMA. Some of the concerns to take into account when choosing an appropriate algorithm are: How accurate is the method? How expensive is it? How smooth is the interpolant? How many data points are needed?

Corresponding Author: savastyludf@hotmail.com

Table 1: Within MAXIMA, these functions are given in the package Interpol, which has to load first.

Function	Description
lagrange *(points)* **lagrange** *(points, option)*	Computes the polynomial by the Lagrangian method. Argument *points* must be either: • a two column matrix, p:matrix([1.2], [3.4], [5,6]), • a list of pairs, p:[[2.4], [6.8], [10,12]], • a list of numbers, p:[5,7,3], in which case the abscissas will be assigned automatically to 1,2,3, etc. In the first two cases the pairs are ordered with respect to the first coordinate before making computations. With the *option* argument, it is possible to select the name for the independent variable, which is 'x by default; to define another one, write something like varname='z.
linearinterpol *(points)* **linearinterpol** *(points, option)*	Computes the polynomial by the linear method. Argument *points* must be either: • a two column matrix, p:matrix([1.2], [3.4], [5,6]), • a list of pairs, p:[[2.4], [6.8], [10,12]], • a list of numbers, p:[5,7,3], in which case the abscissas will be assigned automatically to 1,2,3, etc. In the first two cases the pairs are ordered with respect to the first coordinate before making computations. With the *option* argument, it is possible to select the name for the independent variable, which is 'x by default; to define another one, write something like varname='z.
espline *(points)* **espline** *(points, option1, option2)*	Computes the polynomial interpolation by the cubic spline method. Argument *points* must be either: • a two column matrix, p:matrix([1.2], [3.4], [5,6]), • a list of pairs, p:[[2.4], [6.8], [10,12]], • a list of numbers, p:[5,7,3], in which case the abscissas will be assigned automatically to 1,2,3, etc. In the first two cases the pairs are ordered with respect to the first coordinate before making computations. There are three options to fit specific needs: • 'd1 default 'unknown, is the first derivative at x_1; if it is made equal to 0 (natural cubic spline); if it is equal to a number, the second derivative is calculated based on this number. • 'dn default 'unknown, is the first derivative at x_n; if it is 'unknown, the second derivative at x_n is made equal to 0 (natural cubic spline); if it is equal to a number, the second derivative is calculated based on this number. • 'varname, default 'x, is the name of the independent variable.
charfun2 *(x, a, b)*	Returns true if number x belongs to the interval *[a, b),* and false otherwise. • a two column matrix, p:matrix([1.2], [3.4], [5,6]), • a list of pairs, p:[[2.4], [6.8], [10,12]], • a list of numbers, p:[5,7,3], in which case the abscissas will be assigned automatically to 1,2,3, etc.
ratinterpol *(points, numdeg)* **ratinterpol** *(points, numdeg, option1, option2)*	Generates a rational interpolator for data given by *points* and the degree of the numerator being equal to *numdeg*; the degree of the denominator is calculated automatically. Argument points must be either: In the first two cases the pairs are ordered with respect to the first coordinate before making computations. There are three options to fit specific needs: • 'denterm, default 1, is the independent term of the polynomial in the denominator. • 'varname, default 'x, is the name of the independent variable.
map *(f, expr_1, …, expr_n)*	Returns an expression whose leading operator is the same as that of the expressions *expr_1, …, expr_n* but whose subparts are the results of applying *f* to the corresponding subparts of the expressions. *f* is either the name of a function of n arguments or is a lambda form of n arguments.

2.1. Examples

2.1.1. Lagrange interpolation

```
(%i1) load(interpol)$
      p:matrix([1,1],[3,5],[6,14])$
      lagrange(p);
```

(%o3) $\frac{14(x-3)(x-1)}{15}-\frac{5(x-6)(x-1)}{6}+\frac{(x-6)(x-3)}{10}$

```
(%i4) f(x):=''%;
```

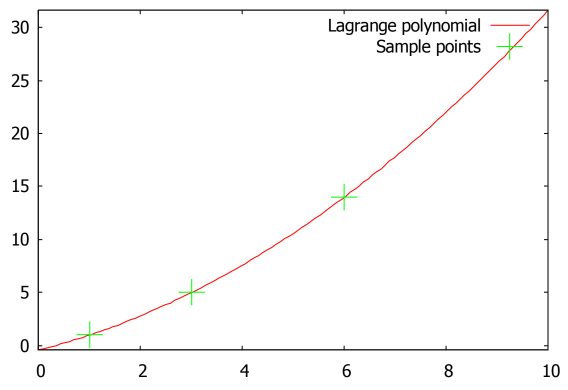
(%o4) $f(x):=\frac{14(x-3)(x-1)}{15}-\frac{5(x-6)(x-1)}{6}+\frac{(x-6)(x-3)}{10}$

Evaluate the polynomial at some points.

```
(%i5) expand(map(f,[1.5,4.5,5.5]));
(%o5) [1.85,9.050000000000001,12.25]
```

Load draw package for plot the polynomial together with points

```
(%i6) load(draw)$
      draw2d(
      color=red,
      key="Lagrange polynomial",
      explicit(f(x),x,0,10),
      point_size=3,
      color=green,
      key="Sample points",
      points(p)
      )$
```

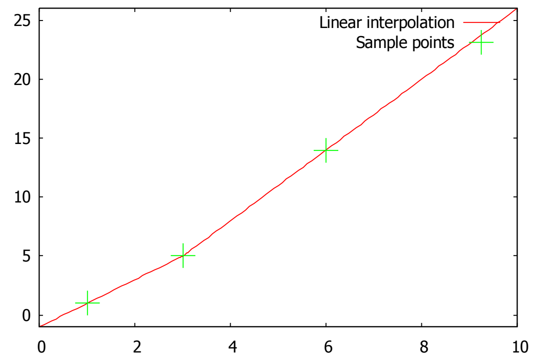


2.1.2. Polynomial interpolation by the linear method

```
(%i8) load(interpol)$
      p:matrix([1,1],[3,5],[6,14])$
      linearinterpol(p);
(%o10) (2 x-1)charfun2(x,-∞,3)+(3 x-4)charfun2(x,3,∞)
(%i11) f(x):=''%;
(%o11) f(x):=(2 x-1)charfun2(x,-∞,3)+(3 x-4)charfun2(x,3,∞)
(%i12) f(x):=''%;
       map(f,[1.5,4.5,5.5]),numer;
(%o12) f(x):=f(x):=(2 x-1)charfun2(x,-∞,3)+(3 x-4)
charfun2(x,3,∞)
 (%o13) [f(x):=(2 x-1)charfun2(x,-∞,3)+(3 x-4)charfun2(x,3,∞),9.
,12.5]
```

Load draw package for plot the polynomial together with points

```
(%i14) load(draw)$
       draw2d(
       color=red,
       key="Linear interpolation",
       explicit(f(x),x,0,10),
       point_size=3,
       color=green,
       key="Sample points",
       points(args(p))
       )$
```



2.1.3. Cubic-Spline interpolation method

```
(%i16) load(interpol)$
       p:matrix([1,1],[3,5],[6,14])$
       cspline(p);
```

(%o18) $\left(\frac{x^3}{20}-\frac{3x^2}{20}+\frac{39x}{20}-\frac{17}{20}\right)\mathrm{charfun2}(x,-\infty,3)+\left(-\frac{x^3}{30}+\frac{3x^2}{5}-\frac{3x}{10}+\frac{7}{5}\right)$
$\mathrm{charfun2}(x,3,\infty)$

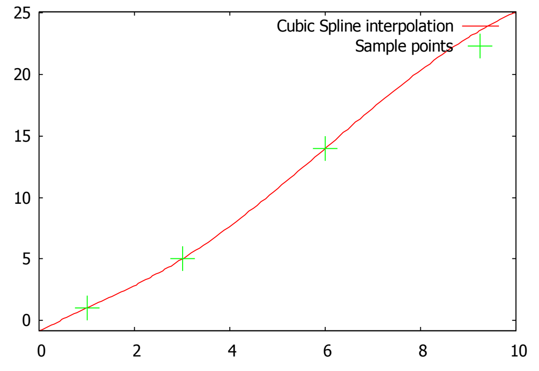
```
(%i19) f(x):=''%;
       map(f,[1.5,4.5,5.5]),numer;
```

(%o19) $f(x):=\left(\frac{x^3}{20}-\frac{3x^2}{20}+\frac{39x}{20}-\frac{17}{20}\right)\mathrm{charfun2}(x,-\infty,3)+\left(-\frac{x^3}{30}+\frac{3x^2}{5}-\frac{3x}{10}+\frac{7}{5}\right)$
$\mathrm{charfun2}(x,3,\infty)$

```
 (%o20) [1.90625,9.162500000000001,12.35416666666666]
```

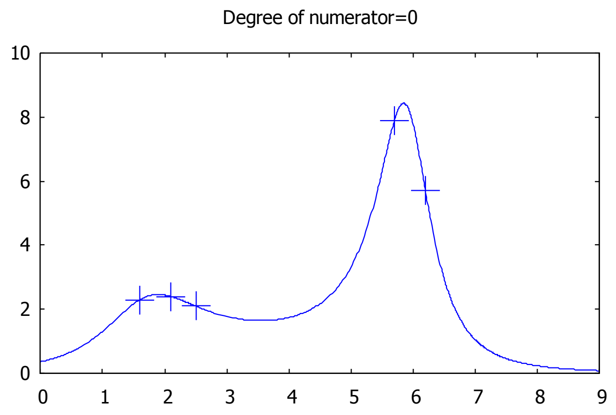
Load draw package for plot the polynomial together with points

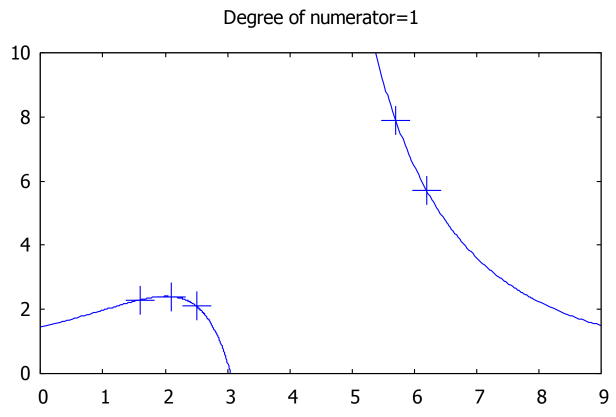
```
(%i21) load(draw)$
       draw2d(
       color=red,
       key="Cubic Spline interpolation",
       explicit(f(x),x,0,10),
       point_size=3,
       color=green,
       key="Sample points",
       points(p)
       )$
```

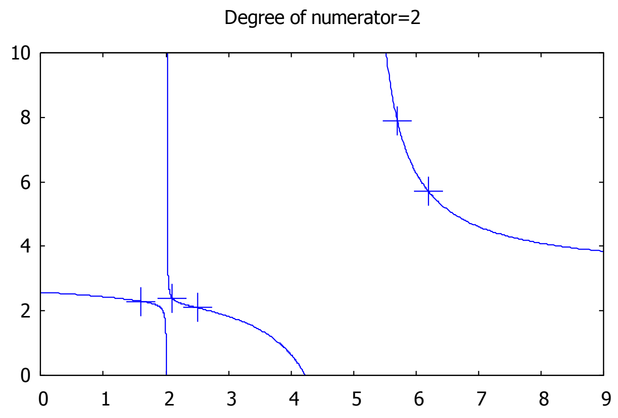


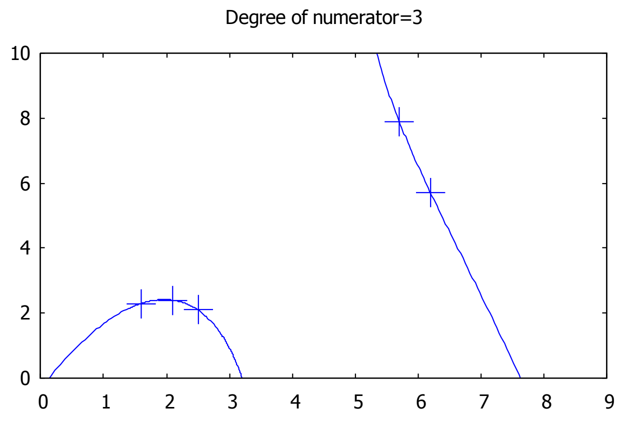
2.1.4. Rational interpolation method

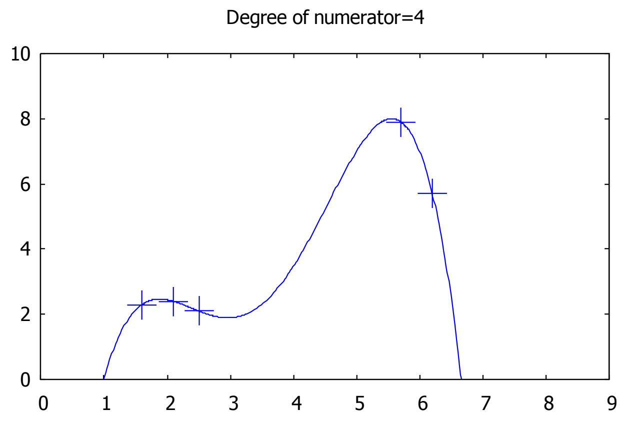
```
(%i35) load(interpol)$
       load(draw)$
       p:[[2.2,2.5],[3.5,2.1],[4.6,5.7],[3.6,2.3],[6.7,8.9]]$
       for k:0 thru length(p)-1 do
       draw2d(
       explicit(ratinterpol(p,k),x,0,9),
       point_size=3,
       points(p),
       title=concat("Degree of numerator=",k),
       yrange=[0,10])$
```











3. MATRICES

Table 1: Some selected functions from MAXIMA about matrices are given in the following table

Function	**Description**
entermatrix *(m,n)*	Returns an *m* by *n* matrix, reading the elements interactively.
genmatrix *(a, i_2, j_2, i_1, j_1)* **genmatrix** *(a, i_2, j_2, i_1)* **genmatrix** *(a, i_2, j_2)*	Returns a matrix generated from *a*, taking element *a*[*i_1*, *j_1*] as the upper-left element and *a*[*i_2*, *j_2*] as the lower-right element of the matrix. Here *a* is a declared array (created by array)
matrix *(row_1,…, row_n)*	Returns a rectangular matrix which has the rows *row_1*, …,*row_n*. Each row is a list of expressions. All rows must be the same length.
invert *(M)*	Returns the inverse of the matrix *M*. The inverse is computed by the adjoint method.
transpose (M)	Returns the transpose of *M*. If *M* is a matrix, the return value is another matrix *N* such that N[i,j] = M[j,i].
triangularize *(M)*	Returns the upper triangular form of the matrix M, as produced by Gaussian elimination. The return value is the same as echelon, except that the leading nonzero coefficient in each row is not normalized to 1.
zeromatrix *(m, n)*	Returns an m by n matrix, all elements of which are zero.

3.1. Examples

3.1.1. Construction of matrices from lists

```
(%i1) a:matrix([a11,a12,a13],[a21,a22,a23],[a31,a32,a33]);
```

$$(\%o1)\quad \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix}$$

```
(%i2) b:matrix([b11,b12,b13],[b21,b22,b23],[b31,b32,b33]);
```

$$(\%o2)\quad \begin{bmatrix} b11 & b12 & b13 \\ b21 & b22 & b23 \\ b31 & b32 & b33 \end{bmatrix}$$

Addition, element by element:

```
(%i3) a+b;
```

$$(\%o3)\quad \begin{bmatrix} b11+a11 & b12+a12 & b13+a13 \\ b21+a21 & b22+a22 & b23+a23 \\ b31+a31 & b32+a32 & b33+a33 \end{bmatrix}$$

Subtraction, element by element:

```
(%i4) a-b;
```

$$(\%o4)\quad \begin{bmatrix} a11-b11 & a12-b12 & a13-b13 \\ a21-b21 & a22-b22 & a23-b23 \\ a31-b31 & a32-b32 & a33-b33 \end{bmatrix}$$

Multiplication, element by element:

```
(%i5) a*b;
```

$$(\%o5)\quad \begin{bmatrix} a11\,b11 & a12\,b12 & a13\,b13 \\ a21\,b21 & a22\,b22 & a23\,b23 \\ a31\,b31 & a32\,b32 & a33\,b33 \end{bmatrix}$$

Division, element by element:

```
(%i6) a/b;
```

$$(\%o6)\quad \begin{bmatrix} \frac{a11}{b11} & \frac{a12}{b12} & \frac{a13}{b13} \\ \frac{a21}{b21} & \frac{a22}{b22} & \frac{a23}{b23} \\ \frac{a31}{b31} & \frac{a32}{b32} & \frac{a33}{b33} \end{bmatrix}$$

Matrix to a scalar exponent, element by element:

```
(%i7) a^3;
```

$$(\%o7)\quad \begin{bmatrix} a11^3 & a12^3 & a13^3 \\ a21^3 & a22^3 & a23^3 \\ a31^3 & a32^3 & a33^3 \end{bmatrix}$$

Scalar base to matrix exponent, element by element:

```
(%i8) exp(a);
```

$$(\%o8)\quad \begin{bmatrix} \%e^{a11} & \%e^{a12} & \%e^{a13} \\ \%e^{a21} & \%e^{a22} & \%e^{a23} \\ \%e^{a31} & \%e^{a32} & \%e^{a33} \end{bmatrix}$$

Matrix base to matrix exponent. This is not carried out element by element:

```
(%i9) a^b;
```

$$(\%o9)\quad \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix}^{\begin{bmatrix} b11 & b12 & b13 \\ b21 & b22 & b23 \\ b31 & b32 & b33 \end{bmatrix}}$$

Noncommutative matrix multiplication:

```
(%i10) a.b;
(%o10)
```

$$\begin{bmatrix} a13\,b31+a12\,b21+a11\,b11 & a13\,b32+a12\,b22+a11\,b12 & a13\,b33+a12\,b23+a11\,b1 \\ a23\,b31+a22\,b21+a21\,b11 & a23\,b32+a22\,b22+a21\,b12 & a23\,b33+a22\,b23+a21\,b1 \\ a33\,b31+a32\,b21+a31\,b11 & a33\,b32+a32\,b22+a31\,b12 & a33\,b33+a32\,b23+a31\,b1 \end{bmatrix}$$

```
(%i11) b.a;
(%o11)
```

$$\begin{bmatrix} a31\,b13+a21\,b12+a11\,b11 & a32\,b13+a22\,b12+a12\,b11 & a33\,b13+a23\,b12+a13\,b1 \\ a31\,b23+a21\,b22+a11\,b21 & a32\,b23+a22\,b22+a12\,b21 & a33\,b23+a23\,b22+a13\,b2 \\ a31\,b33+a21\,b32+a11\,b31 & a32\,b33+a22\,b32+a12\,b31 & a33\,b33+a23\,b32+a13\,b3 \end{bmatrix}$$

Reading the matrix elements interactively:

```
(%i28) n:3$
       a:entermatrix(n,n)$
Is the matrix  1. Diagonal  2. Symmetric  3. Antisymmetric  4. General
 Answer 1, 2, 3 or 4 :   1$
 Row 1 Column 1: a11$
 Row 2 Column 2: a22$
 Row 3 Column 3: a33$
 Matrix entered.
(%i30) a;
```

$$(\%o30)\quad \begin{bmatrix} a11 & 0 & 0 \\ 0 & a22 & 0 \\ 0 & 0 & a33 \end{bmatrix}$$

Transpose of a matrix:

```
(%i32) n:3;
       a:entermatrix(n,n);
(%o32) 3
 Is the matrix  1. Diagonal  2. Symmetric  3. Antisymmetric  4. Genera
 Answer 1, 2, 3 or 4 : 4$
 Row 1 Column 1: a11$
 Row 1 Column 2: a12$
 Row 1 Column 3: a13$
 Row 2 Column 1: a21$
 Row 2 Column 2: a22$
 Row 2 Column 3: a23$
 Row 3 Column 1: a31$
 Row 3 Column 2: a32$
 Row 3 Column 3: a33$
 Matrix entered.
```

$$(\%o33)\quad \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix}$$

```
(%i34) b:transpose(a);
        [a11 a21 a31]
(%o34)  [a12 a22 a32]
        [a13 a23 a33]
```

Triangularize form of a matrix as produced by Gaussian elimination:

```
(%i35) b:triangularize(a);
(%o35)
[a11      a12                                a13
  0   a11 a22-a12 a21                  a11 a23-a13 a21
  0        0         (a11 a22-a12 a21) a33+(a13 a21-a11 a23) a32+(a12 a23-a13 a22) a3
```

3.2. Eigenvalue and Eigenvector

The matrix eigenvalue problem for the given nxn $\mathbf{A}$ matrix with the scalar λ as

$$\mathbf{A}x = \lambda x \qquad (2.3.1),$$

and vector x is obtained by considering

$$(\mathbf{A} - \lambda\mathbf{I})x = 0 \qquad (2.3.2),$$

where I is the identity matrix. A trivial solution is x=0 and a nontrivial solution exists only if the determinant of the coefficient matrix is zero as

$$|\mathbf{A} - \lambda\mathbf{I}| = 0 \qquad (2.3.3).$$

The characteristic equation is

$$a_1\lambda^n + a_2\lambda^{n-1} + \ldots + a_n\lambda + a_{n+1} = 0 \quad (2.3.4),$$

which has n roots $\lambda_i, i = 1, 2, \ldots, \mathrm{n},$ (the eigenvalues of A).

The solution x_i of $(\mathbf{A} - \lambda_i\mathbf{I})x = 0$ forms eigenvectors.

Eigenvalues and eigenvectors have many applications in both pure and applied mathematics. They are used in matrix factorization, in quantum mechanics, and in many other areas.

Table 2: MAXIMA has the following functions for eigenvalue and eigenvector calculations

Function	Description
eigenvalues *(M)* **eivals** *(M)*	Returns a list of two lists containing the eigenvalues of the matrix *M*. The first sublist of the return value is the list of eigenvalues of the matrix, and the second sublist is the list of the multiplicities of the eigenvalues in the corresponding order.
eigenvectors *(M)* **eivects** *(M)*	Computes eigenvectors of the matrix *M*. The return value is a list of two elements. The first is a list of the eigenvalues of *M* and a list of the multiplicities of the eigenvalues. The second is a list of lists of eigenvectors. There is one list of eigenvectors for each eigenvalue. There may be one or more eigenvectors in each list.

3.2.1. Examples

```
(%i1) load(eigen)$
      m:matrix([0,2,0],[2,-2,-2],[0,-2,0]);
      eigenvalues(m);
      [0  2  0]
(%o2) [2 -2 -2]
      [0 -2  0]
 (%o3) [[-4,2,0],[1,1,1]]
```

```
(%i4) [vals,vecs]:eigenvectors(m);
(%o4) [[[-4,2,0],[1,1,1]],[[[1,-2,-1]],[[1,1,-1]],[[1,0
,1]]]]
(%i5) for i thru length (vals[1]) do disp (val [i]=vals
 [1] [i], multi [i]=vals[2] [i], vec [i]=vecs[i];)
val₁ = -4
multi₁ = 1
vec₁ = [[1,-2,-1]]
val₂ = 2
multi₂ = 1
vec₂ = [[1,1,-1]]
val₃ = 0
multi₃ = 1
vec₃ = [[1,0,1]]
(%o5) done
```

4. CONCLUSION

The research paper can apply each and every part of Discrete Data Interpolation and Matrices, help application of the physical sciences and engineering, make faster progress, and help to understand Discrete Data Interpolation and Matrices faster. The paper particularity helps to understand parts of Calculus and is going to extend to other parts of the Calculus.

5. ACKNOWLEDGEMENTS

I would like to thank the Maxima developers and Nigerian Turkish Nile University for their friendly help.

REFERENCES

[1] Niyazi ARI, Lecture notes, University of Technology, Zurich, Switzerland.
[2] Niyazi ARI, Symbolic computation of electromagnetics with Maxima (2013).
[3] http://maxima.sourceforge.net/
[4] R. H. Rand, Introduction to Maxima,
[5] R. Dodier, Minimal Maxima, 2005
[6] https://www.ma.utexas.edu/maxima/maxima_19.html.
[7] N. Ari, G. Apaydin, Symbolic Computation Techniques for Electromagnetics with MAXIMA and MAPLE,Lambert Academic Publishing, 2009.