

Mathematica Module for ARAP Interpolation

MARAP User's Manual

Version 0.24

17 November 2016

Editor: Yoshihiro Mizoguchi, Kyushu University.

Ver. 0.1 13 September 2016

Table of Contents

1	Introduction	1
2	Basic Functions	2
2.1	MakePolygon	2
2.2	HorizontalSnake	2
2.3	VerticalSnake	3
2.4	SnakeTriangle	3
2.5	AnimationRange	3
2.6	Polar	4
2.7	Cartesian	4
2.8	LinearInterpolate	4
2.9	LinearInterpolation	4
2.10	PolarInterpolate	5
2.11	PolarInterpolation	5
2.12	LinearInterpolateSnake2D	6
2.13	PolarInterpolateSnake2D	6
2.14	PolarDecomposition	7
2.15	PolarDecompositionPlus	7
2.16	RotateAngle	7
2.17	CogTrans	7
2.18	PolygonToTriangles	7
2.19	TrianglesToTriangles	8
2.20	TrianglesToPolygon	8
2.21	ValNames	8
2.22	NormF	8
3	Matrix Functions	9
3.1	QuadraticFormVariableMatrix	9
3.2	Div2if	9
3.3	Div2Matrix	9
3.4	QuadraticFormMatrix	9
3.5	LinearFormVector	10
3.6	VtoTriangle	10
3.7	VtoTriangles	10
3.8	Cog	10
3.9	Trans	11
3.10	FindMatrix	11
3.11	FindMatrix1	11
3.12	FindMatrices	11
3.13	FindAffineMatrix	12
3.14	FindAffineMatrices	12
3.15	F1a	12

3.16	F2a	13
3.17	EmbedMatrix.....	13
3.18	EmbedMatrix.....	13
3.19	EmbedMatrix2.....	14
3.20	F1v	14
3.21	EmbedVector.....	15
4	Local Interpolations.....	16
4.1	RotateAngle.....	16
4.2	NewFindMatrices	16
4.3	LocalLinear.....	16
4.4	LocalPolar	16
4.5	LocalAlexa.....	17
4.6	LocalLogExp.....	17
4.7	LocalInterpolations.....	17
5	Grobal Interpolations.....	18
5.1	Constraint Function and Energy Function.....	18
5.1.1	ConstMatrix.....	18
5.1.2	ConstVector	18
5.1.3	ConstMatrixM.....	18
5.1.4	ConstVectorM.....	19
5.1.5	ConstMatrix2.....	19
5.1.6	ConstfixMatrix.....	19
5.1.7	ConstfixVector.....	19
5.1.8	Constfix2Vector	20
5.1.9	ConstPair	20
5.1.10	ConstPair	20
5.1.11	DoubleMatrix.....	20
5.2	Grobal Interpolations.....	20
5.2.1	QuadraticFormAlexa.....	21
5.2.2	QuadraticFormSim.....	21
5.2.3	ARAP	21
6	Draw Animation	22
6.1	ShowStatus	22
6.2	DrawAnimation.....	22
6.3	ListAnimation.....	23
	Index.....	25

1 Introduction

This is a Mathematica Module for investigating an interpolation in Computer Graphics, called the ARAP (as rigid as possible) interpolation. Our module contains about 100 functions of elementary matrix operations, matrix and polygon interpolations and drawing polygons.

To use this package "ARAPlibv024" and sample data "ARAPdata021" users should set a directory where the modules are stored.

[Example]

```
SetDirectory[FileNameJoin[$HomeDirectory, "--- Some Folder ---"]];
<< ARAPlibv024';
<< ARAPdata021';
```

This module was used and introduced in the followings:

- [1] S.Kaji, S.Hirose, S.Sakata, Y.Mizoguchi, Mathematical analysis on affine maps for 2D shape interpolation (<http://dl.acm.org/citation.cfm?id=2422368>), SCA '12 Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp.71-76.
- [2] Y. Mizoguchi, Mathematical Aspects of Interpolation Technique for Computer Graphics, Forum "Math-for-Industry" 2012, Information Recovery and Discovery, 22 October 2012. <http://fmi2012.imi.kyushu-u.ac.jp/>
- [3] T.Hirano, A.hirakawa, N.Miyaki, C.Matsufuji and Y.Mizoguchi. Mathematica Module for ARAP Interpolation
<https://github.com/KyushuUniversityMathematics/MathematicaARAP>

2 Basic Functions

2.1 MakePolygon

`MakePolygon[V, tindex]`
 :: Make polygon

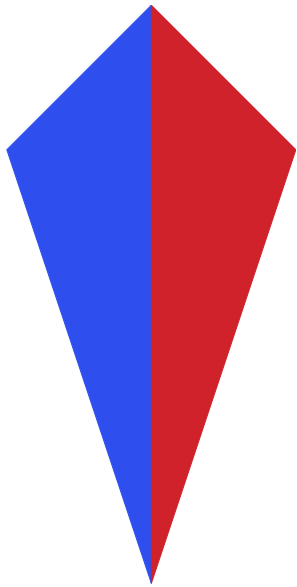
V vertex set

tindex constitution of triangle

return polygon

[Example]

`Graphics[MakePolygon[dataS2, dataT2]]`



2.2 HorizontalSnake

`HorizontalSnake[n]`
 :: Make vertexes of horizontally long rectangle

n rectangle size

return vertex set of rectangle

[Example]

`HorizontalSnake[5]`

`={{0, 0}, {0, 1}, {1, 0}, {1, 1}, {2, 0}, {2, 1}, {3, 0}, {3, 1}, {4, 0}, {4, 1}}`

2.3 VerticalSnake

```

VerticalSnake[n]
    :: Make vertexes of vertically long rectangle
n
    rectangle size
return
    vertex set
[Example]
VerticalSnake[5]
= {{1, 0}, {0, 0}, {1, 1}, {0, 1}, {1, 2}, {0, 2}, {1, 3}, {0, 3}, {1, 4}, {0, 4}}

```

2.4 SnakeTriangle

```

SnakeTriangle[n]
    :: Triangulate Horizontal/Vertex Snake
n
    rectangle size
return
    constitution of triangle
[Example]

Graphics[MakePolygon[HorizontalSnake[10], SnakeTriangle[10]]]

```



```

Graphics[MakePolygon[VerticalSnake[10], SnakeTriangle[10]]]

```



2.5 AnimationRange

```

AnimationRange[conf]
    :: size of Display conforming with polygon
conf
    { start vertex set , end vertex set }
return
    coordinates of display

```

[Example]

```
AnimationRange[Configuration2]
={{-2, 4}, {-3, 5}}
```

2.6 Polar

`Polar[a]` :: Convert cartesian coordinates to polar coordinates

a cartesian coordinates

return polar coordinates

[Example]

```
Polar[{0,1}]
={1,2 $\pi$ }
```

2.7 Cartesian

`Cartesian[a]`

:: Convert polar coordinates to cartesian coordinates

a polar coordinates

return cartesian coordinate

[Example]

```
Cartesian[{1,2 $\pi$ }]
={0,1}
```

2.8 LinearInterpolate

`LinearInterpolate[a,b,t]`

:: Perform linear interpolation of point

a a start point

b an end point

t time

return an interpolated point

[Example]

```
LinearInterpolate[{1, 1}, {2, 2}, 0.5]
={1.5, 1.5}
```

2.9 LinearInterpolation

`LinearInterpolation[p,q,t]`

:: Perform linear interpolation of vertex set

p start vertex set

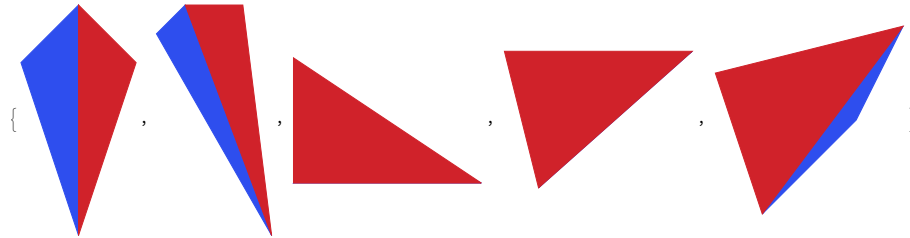
q end vertex set

t time

return interpolated vertex set

[Example]

```
Graphics[MakePolygon[LinearInterpolation[dataS2, dataE2, #], dataT2]] & /
{0, 0.25, 0.5, 0.75, 1}
```



2.10 PolarInterpolate

`PolarInterpolate[a,b,t]`

:: Perform Polar interpolation of point

a a start point

b an end point

t time

return an interpolated point

[Example]

```
PolarInterpolate[{1, 1}, {2, 2}, 0.5]
={1.5, 1.5}
```

2.11 PolarInterpolation

`PolarInterpolation[p,q,t]`

:: Perform Polar interpolation of vertex set

p start vertex set

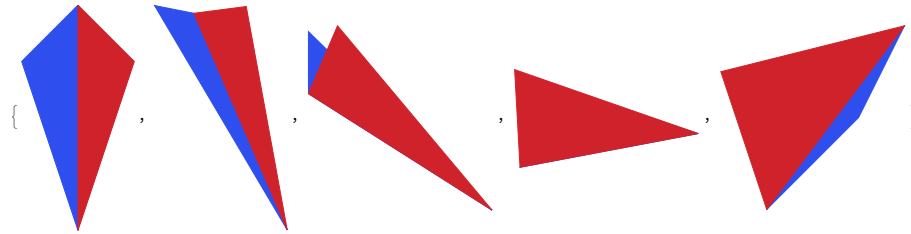
q end vertex set

t time

return interpolated vertex set

[Example]

```
Graphics[MakePolygon[PolarInterpolation[dataS2, dataE2, #], dataT2]] & /
{0, 0.25, 0.5, 0.75, 1}
```



2.12 LinearInterpolateSnake2D

```
LinearInterpolateSnake2D[n,t]
:: aiueo
```

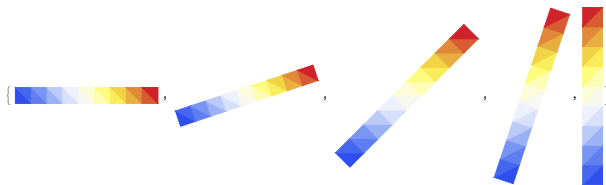
n Snake size

t time (0~1)

return interpolated snake rectangle

[Example]

```
Graphics[LinearInterpolateSnake2D[10, #]] & / {0, 0.25, 0.5, 0.75, 1}
```



2.13 PolarInterpolateSnake2D

```
PolarInterpolateSnake2D[n,t]
:: aiueo
```

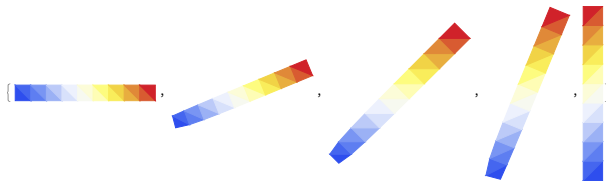
n Snake size

t time (0~1)

return interpolated snake rectangle

[Example]

```
Graphics[PolarInterpolateSnake2D[10, #]] & / {0, 0.25, 0.5, 0.75, 1}
```



2.14 PolarDecomposition

```
PolarDecomposition[m]
    :: Perform polar decomposition

m
    matrix

return
    orthogonal matrix and positive-semidefinite matrix

[Example]
PolarDecomposition[{{1, 1}, {-1, -1}}]
= {{0., 1.}, {-1., 0.}}, {{1., 1.}, {1., 1.}}
```

2.15 PolarDecompositionPlus

```
PolarDecompositionPlus[m]
    :: aiueo

m
    aiueo

return
    aiueo

[Example]
PolarDecompositionPlus[m_]
```

2.16 RotateAngle

```
RotateAngle[m]
    :: Compute angle of Rotation matrix that is performed polar decomposition

m
    matrix

return
    angle

[Example]
RotateAngle[{{1, 2}, {3, 4}}]
= ArcTan[3]
```

2.17 CogTrans

```
CogTrans[pl]
    :: aiueo

pl
    aiueo

return
    aiueo

[Example]
CogTrans[pl_]
```

2.18 PolygonToTriangles

```
PolygonToTriangles[l]
    :: aiueo

l
    aiueo
```

```

return      aiueo
  [Example]
  PolygonToTriangles[l_]

```

2.19 TrianglesToTriangles

```

TrianglesToTriangles[offset,cog]
  :: aiueo

offset      aiueo
cog         aiueo
return      aiueo
  [Example]
  TrianglesToTriangles[offset_,cog_]

```

2.20 TrianglesToPolygon

```

TrianglesToPolygon[t1]
  :: aiueo

tl          aiueo
return      aiueo
  [Example]
  TrianglesToPolygon[t1_]

```

2.21 ValNames

```

ValNames[n]
  :: Name variable

n          number of variable
return     neme of variable
  [Example]
  ValNames[3]
  ={{v1x, v1y}, {v2x, v2y}, {v3x, v3y}}

```

2.22 NormF

```

NormF[m]   :: Calculate Frobenius norm

m          matrix
return     resultant value of Frobenius norm
  [Example]
  NormF[1,2,3,4]
  =30

```

3 Matrix Functions

3.1 QuadraticFormVariableMatrix

```
QuadraticFormVariableMatrix[v1]
    :: QuadraticFormVariableMatrix

v1          list of values

return      matrix

[Example]
QuadraticFormVariableMatrix[{1,2,3}]
{{1, 2, 3}, {2, 4, 6}, {3, 6, 9}}
```

3.2 Div2if

```
Div2if[n,l]
    :: divide all elements except n-th element of l by 2

n,l        index,list of values

return      vector divided all elements except n-th element of l by 2

[Example]
Div2if[3, {3, 6, 7, 5, 1}]
{3/2, 3, 7, 5/2, 1/2}
```

3.3 Div2Matrix

```
Div2Matrix[m]
    :: divide all elements except diagonal ones of m by 2

m          matrix

return      matrix divided all elements except diagonal ones of m by 2

[Example]
Div2Matrix[{{1, 3}, {7, 4}}]
{{1, 3/2}, {7/2, 4}}
```

3.4 QuadraticFormMatrix

```
QuadraticFormMatrix[poly,v1]
    :: QuadraticFormMatrix

poly,v1    polynomial,list of values

return      QuadraticFormMatrix

[Example]
QuadraticFormMatrix[(a*x^2 + b*x + c-y)^2, {a, b, c}]
{{x^4, x^3, x^2}, {x^3, x^2, x}, {x^2, x, 1}}
```

3.5 LinearFormVector

`LinearFormVector[poly,vl]`
 :: LinearFormVector

poly,vl polynomial,list of values

return LinearFormVector

[Example]

```
LinearFormVector[(a*x^2 + b*x + c - y)^2, {a, b, c}]
{-2 x^2 y, -2 x y, -2 y}
```

3.6 VtoTriangle

`VtoTriangle[V,t]`
 :: return coordinates of triangle

V,t list of vertices,list of triangulation

return list of coordinates of vertices of triangle

[Example]

```
VtoTriangle[{{-1, 1}, {0, -2}, {1, 1}, {0, 2}}, {1, 2, 4}]
{{-1, 1}, {0, -2}, {0, 2}}
```

3.7 VtoTriangles

`VtoTriangles[V,T]`
 :: return coordinates of triangles

V,T list of vertices,list of triangulation

return list of coordinates of vertices of triangles

[Example]

```
VtoTriangles[{{-1, 1}, {0, -2}, {1, 1}, {0, 2}}, {{1, 2, 3}, {1, 2, 4}, {2, 3, 4}}]
{{{ -1, 1}, {0, -2}, {1, 1}}, {{ -1, 1}, {0, -2}, {0, 2}}, {{0, -2}, {1, 1}, {0, 2}}}
```

3.8 Cog

`Cog[P]` :: return triangle center

P triangle

return triangle center of P

[Example]

```
Cog[{{-1, 1}, {0, -2}, {0, 2}}]
{-(1/3), 1/3}
```

3.9 Trans

```
Trans[P,l]
    :: parallel shift by l
P,l      triangle,vector
return    triangle
[Example]
Trans[{1, 2}, {5, -3}, {-4, 1}], {5, 6}]
{{-4, -4}, {0, -9}, {-9, -5}}
```

3.10 FindMatrix

```
FindMatrix[P1,P2]
    :: find matrix converts P1 to P2
P1,P2    triangles whose center is origin
return    matrix convert P1 to P2
[Example]
P11 = {{0, 2}, {-3, -1}, {3, -1}};
P21 = {{-4, 3}, {1, -2}, {3, -1}};
FindMatrix[P11, P21]
{{1/3, -2}, {1/6, 3/2}}
```

3.11 FindMatrix1

```
FindMatrix1[P1,P2]
    :: find matrix converts P1 to P2
P1,P2    triangles whose center is origin
return    matrix convert P1 to P2
[Example]
P11 = {{0, 2}, {-3, -1}, {3, -1}};
P21 = {{-4, 3}, {1, -2}, {3, -1}};
FindMatrix1[P11, P21]
{{1/3, -2}, {1/6, 3/2}}
```

3.12 FindMatrices

```
FindMatrices[V1,V2,T]
    :: find matrices converts each triangles represented by V1 and T to ones represented by V2 and T
V1,V2,T   V1,V2:list of vertices T:list of triangulation
return     matrices
[Example]
V1 = {{-1, 1}, {0, 2}, {1, -3}, {4, -5}};
```

```

V2 = {{-3, 3}, {-2, 5}, {2, 1}, {3, 1}};
T = {{1, 2, 3}, {1, 2, 4}, {2, 3, 4}};
FindMatrices[V1, V2, T]
{{{3/2, -(1/2)}, {1, 1}},
 {{12/11, -(1/11)}, {10/11, 12/11}},
 {{-(3/13), -(11/13)}, {8/13, 12/13}}}

```

3.13 FindAffineMatrix

```

FindAffineMatrix[P1,P2]
    :: find affine matrix converts P1 to P2

P1,P2    triangles

return    matrix converts P1 to P2

[Example]
FindAffineMatrix[{{-1, 1}, {0, -2}, {0, 2}}, {{-4, 3}, {1, -2}, {3, 0}}]

```

3.14 FindAffineMatrices

```

FindAffineMatrices[V1,V2,T]
    :: find affine matrix convert each triangles

V1,V2,T    V1,V2:list of vertices T:list of triangulation

return    matrices

[Example]
V1 = {{-1, 1}, {0, 2}, {1, -3}, {4, -5}};
V2 = {{-3, 3}, {-2, 5}, {2, 1}, {3, 1}};
T = {{1, 2, 3}, {1, 2, 4}, {2, 3, 4}};
FindAffineMatrices[V1, V2, T]
{{{3/2, -(1/2), -1}, {1, 1, 3}, {0, 0, 1}},
 {{12/11, -(1/11), -(20/11)}, {10/11, 12/11, 31/11}, {0, 0, 1}},
 {{-(3/13), -(11/13), -(4/13)}, {8/13, 12/13, 41/13}, {0, 0, 1}}}

```

3.15 F1a

```

F1a[{{a1x,a1y},{b1x,b1y},{c1x,c1y}},{{m11,m12},{m21,m22}}]
    :: compute quadratic form matrix

{{a1x,a1y},{b1x,b1y},{c1x,c1y}},{{m11,m12},{m21,m22}}
    triangle,matrix

return    quadratic form matrix

[Example]
F1a[{{-1, 0}, {1, 1}, {2, -3}}, {{1, 3}, {4, 2}}]
{{17/81, -(5/27), -(2/81)},
 {-(5/27), 2/9, -(1/27)},
 {-(2/81), -(1/27), 5/81}}

```


3.16 F2a

```
F1a[{a1x,a1y},{b1x,b1y},{c1x,c1y}},{m11,m12},{m21,m22}]
:: compute quadratic form matrix

{{a1x,a1y},{b1x,b1y},{c1x,c1y}},{m11,m12},{m21,m22}}
triangle,matrix

return quadratic form matrix
```

[Example]

```
F2a[{-1, 0}, {1, 1}, {2, -3}], {{1, 3}, {4, 2}}]
{{137/2430, 0, -(7/405), -(1/27), -(19/486), 1/27},
{0, 137/2430, 1/27, -(7/405), -(1/27), -(19/486)},
{-(7/405), 1/27, 4/135, 0, -(1/81), -(1/27)},
{-(1/27), -(7/405), 0, 4/135, 1/27, -(1/81)},
{-(19/486), -(1/27), -(1/81), 1/27, 25/486, 0},
{1/27, -(19/486), -(1/27), -(1/81), 0, 25/486}}
```

3.17 EmbedMatrix

```
EmbedMatrix[n,i,j,M]
:: embed 2-degree matrix M in n-degree 0 matrix

n,i,j,M degree,index,index,matrix

return matrix
```

[Example]

```
EmbedMatrix[6, 2, 4, {{1, 2}, {3, 4}}]
{{0, 0, 0, 0, 0, 0},
{0, 1, 0, 2, 0, 0},
{0, 0, 0, 0, 0, 0},
{0, 3, 0, 4, 0, 0},
{0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0}}
```

3.18 EmbedMatrix

```
EmbedMatrix[n,i,j,k,M]
:: embed 2-degree matrix M in n-degree 0 matrix

n,i,j,k,M degree,index,index,index,matrix

return matrix
```

[Example]

```
EmbedMatrix[8, 2, 4, 7, {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}]
{{0, 0, 0, 0, 0, 0, 0, 0},
{0, 1, 0, 2, 0, 0, 3, 0},
{0, 0, 0, 0, 0, 0, 0, 0},
{0, 4, 0, 5, 0, 0, 6, 0},
{0, 0, 0, 0, 0, 0, 0, 0}}
```

```
{0, 0, 0, 0, 0, 0, 0, 0},
{0, 7, 0, 8, 0, 0, 9, 0},
{0, 0, 0, 0, 0, 0, 0, 0}}
```

3.19 EmbedMatrix2

```
EmbedMatrix2[n,i,j,k,M]
  :: embed 6-degree matrix M in 2n-degree 0 matrix
```

```
n,i,j,k,M    n:size, i,j,k:index M:matrix
```

```
return       matrix
```

[Example]

```
A = EmbedMatrix[6, 1, 3, 5, {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}] +
  EmbedMatrix[6, 2, 4, 6, {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}]
```

```
{{1, 0, 2, 0, 3, 0},
 {0, 1, 0, 2, 0, 3},
 {4, 0, 5, 0, 6, 0},
 {0, 4, 0, 5, 0, 6},
 {7, 0, 8, 0, 9, 0},
 {0, 7, 0, 8, 0, 9}}
```

```
EmbedMatrix2[4, 1, 3, 4, A]
```

```
{{1, 0, 0, 0, 2, 0, 3, 0},
 {0, 1, 0, 0, 0, 2, 0, 3},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0},
 {4, 0, 0, 0, 5, 0, 6, 0},
 {0, 4, 0, 0, 0, 5, 0, 6},
 {7, 0, 0, 0, 8, 0, 9, 0},
 {0, 7, 0, 0, 0, 8, 0, 9}}
```

3.20 F1v

```
F1v[{{a1x,a1y},{b1x,b1y},{c1x,c1y}},{{m11,m12},{m21,m22}}]
  :: compute linear form vector
```

```
{{a1x,a1y},{b1x,b1y},{c1x,c1y}},{{m11,m12},{m21,m22}}
  triangle,matrix
```

```
return       linear form vector
```

[Example]

```
F1v[{{-1, 0}, {1, 1}, {2, -3}}, {{1, 3}, {4, 2}}]
{14/9, 4, -(8/3), -4, 10/9, 0}
```

3.21 EmbedVector

`EmbedVector[n,i,j,k,V]`

:: embed vector V in 2n-degree 0 vector

`n,i,j,k,V` `n`:size `i,j,k`:index `V`:vector

return vector

[Example]

`EmbedVector[5, 1, 3, 4, {-6, 7, 28, 19, -4, 31}]`
`{-6, 7, 0, 0, 28, 19, -4, 31, 0, 0}`

4 Local Interpolations

4.1 RotateAngle

`RotateAngle[m_,flag_]`
 ::controlled rotate angle of m by flag

m

flag 1 or 0 or -1, control the rotate angle

return angle

[Example]

`RotateAngle[m,1]`

4.2 NewFindMatrices

`NewFindMatrices[conf_]`
 ::find matrices converted each triangles

conf configuration

return matrices

[Example]

`NewFindMatrices[Configuration2]`

4.3 LocalLinear

`LocalLinear[m_]`
 ::compute a local linear interpolations depend on time

m matrices computed by `NewFindMatrices[conf_]`

return matrices of local linear interpolation

[Example]

`LocalLiner[NewFindMatrices[Configuration2]]`

4.4 LocalPolar

`LocalPolar[m_]`
 ::compute a local polar interpolations depend on time

m matrices computed by `NewFindMatrices[conf_]`

return matrices of local polar interpolation

[Example]

`LocalPolar[NewFindMatrices[Configuration2]]`

4.5 LocalAlexa

```
LocalAlexa[m_]
    ::compute a local ARAP interpolations depend on time
m
    matrices computed by NewFindMatrices[conf_]
return
    matrices of local alexa interpolation
[Example]
LocalAlexa[NewFindMatrices[Configuration2]]
```

4.6 LocalLogExp

```
LocalLogExp[m_]
    ::compute a local log-exp interpolations depend on time
m
    matrices computed by NewFindMatrices[conf_]
return
    matrices of local log-exp interpolation
[Example]
LocalLogExp[NewFindMatrices[Configuration2]]
```

4.7 LocalInterpolations

```
LocalInterpolations[local_, conf_]
    ::compute local interpolations that you choice
local
    LocalLinear/LocalPolar/LocalAlexa/LocalLogExp
conf
    configuration
return
    [Example]
    LocalInterpolations[LocalPolar, Configuration2]
```

5 Grobal Interpolations

5.1 Constraint Function and Energy Function

5.1.1 ConstMatrix

```
ConstMatrix[m_,st_]
::

m          choice of vertex
st         list of start coordinates
return     matrix

[Example]
ConstMatrix[1,dataS2]
```

5.1.2 ConstVector

```
ConstVector[m_,st_,en_,t_]
::

m          choice of vertex
st         list of start coordinates
en         list of end coordinates
t          parameter of time(0...1)
return     vector

[Example]
ConstVector[1,dataS2,dataE2,0.5]
```

5.1.3 ConstMatrixM

```
ConstMatrixM[st_,en_,tri_,t_]
::

st         list of start coordinates
en         list of end coordinates
tri        list of triangulation
t          parameter of time(0...1)
return     matrix

[Example]
ConstMatrixM[dataS2,dataE2,dataT2,0.5]
```

5.1.4 ConstVectorM

```
ConstVectorM[st_,en_,tri_,t_]
::
st          list of start coordinates
en          list of end coordinates
tri         list of triangulation
t           parameter of time(0...1)
return      vector
[Example]
ConstVectorM[dataS2,dataE2,dataT2,0.5]
```

5.1.5 ConstMatrix2

```
ConstMatrix2[n_,k_,l_]
::
n           weight of constraint function
k,l         Choice two numbers you want to fix.
return      matrix
[Example]
ConstMatrix2[2,1,2]
```

5.1.6 ConstfixMatrix

```
ConstfixMatrix[n_,k_,l_,st_]
::
n           weight of constraint function
k,l         Choice two numbers you want to fix.
st          list of start coordinates
return      matrix
[Example]
ConstfixMatrix[2,1,2,dataS2]
```

5.1.7 ConstfixVector

```
ConstfixVector[n_,k_,l_,st_]
::
n           weight of constraint function
k,l         Choice two numbers you want to fix.
st          list of start coordinates
return      vector
[Example]
ConstfixVector[2,1,2,dataS2]
```

5.1.8 Constfix2Vector

```
Constfix2Vector[n_,k_,l_,st_,t_]
::

n          weight of constraint function
k,l        Choice two numbers you want to fix.
st         list of start coordinates
t          parameter of time(0...1)
return     vector

[Example]
Constfix2Vector[1,1,2,dataS2,0.5]
```

5.1.9 ConstPair

```
ConstPair[m_]
::

m          choice of vertex
return     {matrix,vector}

[Example]
ConstPair[1]
```

5.1.10 ConstPair

```
ConstPair[m_,n_]
::

m,n        choice of vertex
return     {matrix,vector}

[Example]
ConstPair[1,2]
```

5.1.11 DoubleMatrix

```
DoubleMatrix[m_]
::

m          matrix
return     matrix

[Example]
DoubleMatrix[]
```

5.2 Grobal Interpolations

5.2.1 QuadraticFormAlexa

```

QuadraticFormAlexa[local_,conf_]
::
local      choice of local interpolation
conf       configuration
return     {matrix,vector}
[Example]
QuadraticFormAlexa[LocalPolar,Configuration2]

```

5.2.2 QuadraticFormSim

```

QuadraticFormSim[local_,conf_]
::
local      choice of local interpolation
conf       configuration
return     {matrix,vector}
[Example]
QuadraticFormSim[LocalLogExp,Configuration2]

```

5.2.3 ARAP

```

ARAP[local_,energy_,const_,conf_]
::
local      choice of local interpolation
energy     choice of energy function
const      choice of constraint function
conf       configuration
return     {matrix,vector}
[Example]
ARAP[LocalPolar,QuadraticFormAlexa,Const[1],Configuration2]

```

6 Draw Animation

6.1 ShowStatus

`ShowStatus[st,en,tri,plotrange]`
 :: Draw start and end of Graphic

st start state of vertex set

en end state of vertex set

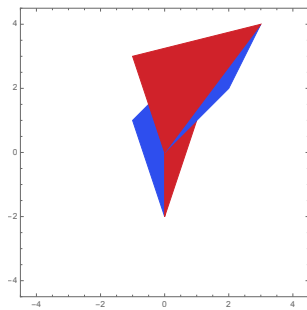
tri constitution of triangle

plotrange Display range

return graph of start and end status

[Example]

`ShowStatus[dataS2, dataE2, dataT2, 4.5]`



6.2 DrawAnimation

`DrawAnimation[local,energy,const,conf]`
 :: aiueo

local Choice of local interpolation

energy Choice of energy interpolation

const Choice of constraint interpolation

conf {start vertex set,end vertex set,constitution of triangle}

return animation

[Example]

`DrawAnimation[LocalAlexa, FrobeniusEnergy, Const2, Configuration2]`

6.3 ListAnimation

ListAnimation[*k*,*local*,*energy*,*const*,*conf*]
 :: aiueo

k number of frame division

local Choice of local interpolation

energy Choice of energy interpolation

const Choice of constraint interpolation

conf {start vertex set,end vertex set,constitution of triangle}

return list of graph

[Example]

ListAnimation[*k*_,*n*_,*c*_,*e*_,*st*_,*en*_,*tri*_,plotrange]



Index

A

AnimationRange	3
ARAP	21

C

Cartesian	4
Cog	10
CogTrans	7
Constfix2Vector	20
ConstfixMatrix	19
ConstfixVector	19
ConstMatrix	18
ConstMatrix2	19
ConstMatrixM	18
ConstPair	20
ConstVector	18
ConstVectorM	19

D

Div2if	9
Div2Matrix	9
DoubleMatrix	20
DrawAnimation	22

E

EmbedMatrix	13
EmbedMatrix2	14
EmbedVector	15

F

F1a	12
F1v	14
F2a	13
FindAffineMatrices	12
FindAffineMatrix	12
FindMatrices	11
FindMatrix	11
FindMatrix1	11

H

HorizontalSnake	2
-----------------------	---

L

LinearFormVector	10
LinearInterpolate	4
LinearInterpolation	4
ListAnimation	23
LocalAlexa	17
LocalInterpolations	17
LocalLinear	16
LocalLogExp	17
LocalPolar	16

M

MakePolygon	2
-------------------	---

N

NewFindMatrices	16
NormF	8

P

Polar	4
PolarDecomposition	7
PolarDecompositionPlus	7
PolarInterpolate	5
PolarInterpolateSnake2D	6
PolarInterpolation	5
PolygonToTriangles	7

Q

QuadraticFormAlexa	21
QuadraticFormMatrix	9
QuadraticFormSim	21
QuadraticFormVariableMatrix	9

R

RotateAngle	7, 16
-------------------	-------

S

ShowStatus	22
SnakeTriangle	3

T

Trans	11
TrianglesToPolygon	8
TrianglesToTriangles	8

V

ValNames	8
VerticalSnale	3
VtoTriangle	10
VtoTriangles	10