Package 'FEA'

February 28, 2022

Type Package

Title Finite Element Modeling for R

Version 0.0.1

Author Henna D. Bhramdat

Maintainer Henna D. Bhramdat <bhramdath@ufl.edu>

Description Finite element modeling of 2D geometries using constant strain triangles. Applies material properties and boundary conditions (load and constraint) to generate a finite element model. The model produces stress, strain, and nodal displacements; a heat map is available to demonstrate regions where output variables are high or low. Also provides options for creating a triangular mesh of 2D geometries.

References Bathe, K. J. (1996). Finite Element Procedures._____
Seshu, P. (2012). Textbook of Finite Element Analysis._____
Mustapha, K. B. (2018). Finite Element Computations in Mechanics with R.

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.2

Imports geometry, geosphere, ptinpoly, sp, spatstat, MASS

Depends R (>= 3.5.0)

R topics documented:

ApplyBC . . . 2
AutoAdjust . . . 3
Dimensions . . . 4
ElementMat . . . 5
ExpandEM . . . 5
ExpandSFT . . . 6
FEMStrain . . . 7
FEMStress . . . 8
ForceVector . . . 10
GLForces . . . 10
GlobalMat . . . 11
LocalStress . . . 12
ManualAdjust . . . 13
NodeDis . . . 13
PlotSystem . . . 14

ReducedEM 16
ReducedSF 17
SinglePoly 17
SurfaceTraction 18
ThreshPts 19
triangulate0 19

Index **21**

`ApplyBC` *ApplyBC*

Description

Boundary constraint for element centroids based on coordinate points. For the x & y direction per centroid create matrix with boundary 1(unfixed) or 0(fixed).

Usage

```
ApplyBC(meshP, BoundConx, BoundCony)
```

Arguments

`meshP`	Matrix (2 x n) containing coordinate points of the mesh
`BoundConx`	Boundary constraint for nodes in the x-direction
`BoundCony`	Boundary constraint for nodes in the y-direction

Value

A data frame with constraint parameters applied to each node in the x and y directions. Formatted for use in reduced element matrix.

`NodeKnownL`	Constraint parameters

Examples

```
# meshP = MeshPts$p #mesh points
# BoundConx = numeric(NROW(meshP))
# BoundCony = numeric(NROW(meshP))
# BoundConx[1:NROW(meshP)] = BoundCony[1:NROW(meshP)] = 1
# ApplyBC(meshP, BoundConx, BoundCony)
```

AutoAdjust *AutoAdjust*

Description

Allows for automatic refinement of the triangular mesh generated based on given parameters. Will remove elements that are outside the margin of the geometry.

Usage

```
AutoAdjust(meshP, meshT, edge, centroid, AspectR, AR)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
meshT	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
edge	Coordinate points of the initial geometry.
centroid	Matrix (2 x n) of triangle elements.
AspectR	Aspect ratio of each triangle element.
AR	maximum desired aspect ratio, numeric value.

Value

Generates new mesh and centroid tables

Meshpts	Includes both new mesh coordinate points and triangulation of points.
Centroids	Centroid positions for each triangle element.

Examples

```
# meshP = MeshPts$p #mesh points
# meshT = MeshPts$T #triangulation list
# edge =  Line #original geometry
# centroid = centroids
# AspectR = AspectRatio #dimensional value
# AR = 10 #aspect ratio limit for elimination
# AutoAdjust(meshP, meshT, edge, centroid, AspectR, AR)
```

Dimensions *Dimensions*

Description

Calculates dimensional values for each triangular element, including truss length & angles, distance from nodal point to centroid, aspect ratio of each triangle element, and area of the triangle.

Usage

```
Dimensions(meshP, meshT, centroid)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
meshT	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
centroid	Matrix (2 x n) containing coordinate points of the centroid of each triangular element.

Value

Evaluation of triangle elements truss, angle, and area.

Truss	Nodal pairs that form each truss.
TrussLength	Distance between each paired nodes forming a truss, its length.
Dist2Cent	Shortest distance from truss to triangle centroid.
Truss angle	Angles of the triangle created from truss meeting.
AspectRatio	Aspect ratio of triangle elements.
Area	Area within triangle elements.

Examples

```
# meshP = MeshPts$p #mesh points
# meshT = MeshPts$T #triangulation list
# centroid = Centroids
# Dimensions(meshP, meshT, centroid)
```

ElementMat *ElementMat*

Description

Generates an element stiffness matrix

Usage

```
ElementMat(meshP, meshT, Nu, Y, Thick)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
meshT	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
Nu	Value of Poisson's ratio for each element
Y	Value of Young's (Elastic) modulus for each element
Thick	Value of the thickness of the mesh, a positive value must be given.

Value

Generates initial element matrix needed for the finite element model.

EMPStress	An element matrix of the geometry under stress.
EMPStrain	An element matrix of the geometry under strain.

Examples

```
# Y = 20 #Elastic modulus
# Nu = 0.45 #Poisson ratio
# Thick = 0.001
# DOF = 6
# meshP = MeshPts$p
# meshT = MeshPts$T
# test7 = ElementMat(meshP, meshT, Nu, Y, Thick)
```

ExpandEM *ExpandEM*

Description

Generates the expanded element matrix, which represents the contribution of individual finite elements towards the global structural matrix

Usage

```
ExpandEM(meshP, meshT, centroid, EMatrixlist)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
meshT	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
centroid	Matrix (2 x n) containing coordinate points of the centroid of each triangular element.
EMatrixlist	EMPStress or EMPStrain generated from ElementMat function. List of element matrices.

Value

Produces large (n x n) matrix.

ExpandedMat	The expanded element matrix

Examples

```
# EMatrixlist = EMPStress
# meshP = MeshPts$p #nodal points
# meshT = MeshPts$T #triangulation list
# centroid = Centroids
# ExpandEM(meshP, meshT, centroid, EMatrixlist)
```

ExpandSFT *ExpandSFT*

Description

Generates expanded surface force element matrix from SurfaceTraction function

Usage

```
ExpandSFT(meshP, meshT, SurfTrac)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
meshT	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
SurfTrac	List of surface forces.

Value

Produces a large (n x n) element matrix of surface forces.

ExpandedSurf	Expanded surface force element matrix.

Examples

```
# meshT = MeshPts$T
# meshP = MeshPts$p
# SurfTrac = SurfaceTraction #matrix
# ExpandSFT(meshP, meshT, SurfTrac)
```

FEMStrain *FEMStrain*

Description

Creates a complete finite element model using strain for a given 2D mesh under specified boundary conditions (constrain and load).

Usage

```
FEMStrain(meshP, meshT, centroid, BoundConx, BoundCony, SFShear,
SFTensile, Length, area, Fx, Fy, Y, Nu, Thick)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
meshT	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
centroid	Matrix (2 x n) containing coordinate points of the centroid of each triangular element.
BoundConx	Boundary constraint for nodes in the x-direction
BoundCony	Boundary constraint for nodes in the y-direction
SFShear	Magnitude of positive shear traction; if there is no surface traction then SFShear = 0
SFTensile	Magnitude of tensile surface traction; if there is no surface traction then SFTensile = 0
Length	Truss length
area	Triangle element area
Fx	Load vector for the x-direction
Fy	Load vector for the y-direction
Y	Value of Young's (Elastic) modulus
Nu	Value of Poisson's ratio
Thick	Value of the thickness of the mesh, a value must be given.

Value

Completes the FEM to generate values of stress and strain and nodal displacement.

NodeDisplacement	Node displacement on each axis
LocalStress	Stress as calucated from stress, strain, and stress from strain. Three (3) [3 x n] matrices where [x, y, tau]

Examples

```
Y = 17e9
Nu = 0.45
Thick = 0.001
DOF = 6

# meshP = MeshPts$p
# meshT = MeshPts$T
# centroid = Centroids
# BoundConx = numeric(NROW(meshP))
# BoundCony = numeric(NROW(meshP))
# BoundConx[1:NROW(meshP)] = BoundCony[1:NROW(meshP)] = 1
# SFShear = 0
# SFTensile = 0
# Length = TrussLength
# area = Area
# Fx = 20
# Fy = 10
# FEAtest = FEAStrain(meshP, meshT, centroid, BoundConx, BoundCony, SFShear, SFTensile, Length, area, Fx, Fy, N

# Following the FEAtest, plot for value map:
# PlotVal = FEAtest$Strain
# Kol = 3
# a= 1; b= 2; c=3; d=4; e=5; f=6; g=7; h=8; i=9
# PlotSystem(meshP, meshT, PlotVal, Kol, a, b, c, d, e, f, g, h, i)
```

FEMStress *FEMStress*

Description

Creates a complete finite element model using stress for a given 2D mesh under specified boundary conditions (constrain and load).

Usage

```
FEMStress(meshP, meshT, centroid, BoundConx, BoundCony, SFShear,
SFTensile, Length, area, Fx, Fy,  Y, Nu, Thick)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
meshT	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
centroid	Matrix (2 x n) containing coordinate points of the centroid of each triangular element.
BoundConx	Boundary constraint for nodes in the x-direction
BoundCony	Boundary constraint for nodes in the y-direction
SFShear	Magnitude of positive shear traction; if there is no surface traction then SFShear = 0

SFTensile	Magnitude of tensile surface traction; if there is no surface traction then SFTensile = 0
Length	Truss length
area	Triangle element area
Fx	Load vector for the x-direction
Fy	Load vector for the y-direction
Y	Value of Young's (Elastic) modulus
Nu	Value of Poisson's ratio
Thick	Value of the thickness of the mesh, a value must be given.

Value

Completes the FEM to generate values of stress and strain and nodal displacement.

NodeDisplacement	Node displacement on each axis
LocalStress	Stress as calucated from stress, strain, and stress from strain. Three (3) [3 x n] matrices where [x, y, tau]

Examples

```
Y = 20e5
Nu = 0.45
Thick = 0.001
DOF = 6

# meshP = MeshPts$p
# meshT = MeshPts$T
# centroid = Centroids
# BoundConx = numeric(NROW(meshP))
# BoundCony = numeric(NROW(meshP))
# BoundConx[1:NROW(meshP)] = BoundCony[1:NROW(meshP)] = 1
# SFShear = 0 #can leave as 0
# SFTensile = 0 #can leave as 0
# Length = TrussLength
# area = Area
# Fx = 20
# Fy = 10
# FEAtest = FEAStress(meshT, meshP, centroid, BoundConx, BoundCony, SFShear, SFTensile, Length, area, Fx, Fy, N

# Following the FEAtest, plot for value map:
# PlotVal = FEAtest$Stress
# Kol = 3
# a= 1; b= 2; c=3; d=4; e=5; f=6; g=7; h=8; i=9
# PlotSystem(meshP, meshT, PlotVal, Kol, a, b, c, d, e, f, g, h, i)
```

ForceVector *ForceVector*

Description

Creates a matrix of loads in the x & y direction for each load unconstrained node.

Usage

```
ForceVector(Fx, Fy, RSF, meshP, NodeKnownL)
```

Arguments

Fx	Load vector for the x-direction
Fy	Load vector for the y-direction
RSF	If surface traction is present assign value as the ReducedSF matrix; if there is no surface traction set RSF = 0
meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
NodeKnownL	data frame with constraint parameters applied to each node in the x and y directions. Formatted for use in reduced element matrix. Generated from ApplyBC function.

Value

Produces a matrix with loading parameters for each node.

ReducedFV	Reduced force vector matrix containing the model load parameters.

Examples

```
# meshP = MeshPts$p #mesh points
# RSF = RSF #reduced surface traction. If none is present, RSF = 0
# Fx = 10
# Fy = 10
# NodeKnownL = Nodelist #boundary conditions at nodes
# ForceVector(Fx, Fy, RSF, meshP, NodeKnownL)
```

GLForces *GLForces*

Description

Uses nodal displacements to determine global and local forces at each node

Usage

```
GLForces(meshP, meshT, GMat, GlobalND, EMatrixlist)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
meshT	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
GMat	Global matrix
GlobalND	Global nodal displacement
EMatrixlist	Element matrix list

Value

Matrices of global and local forces

GForce	Large global force matrix.
Lforce	Large local force matrix.

Examples

```
# meshP = MeshPts$p
# meshT = MeshPts$T
# GMat = Gm #global matrix
# GlobalND = GlobalND #global nodal displacement
# EMatrixlist = EMPStress
# GLForces(GMat, GlobalND, EMatrixlist, meshT)
```

GlobalMat *GlobalMat*

Description

Generates global stiffness matrix - once established, the expanded element matrix must be combined to create the global structural stiffness matrix by adding the expanded matrices.

Usage

```
GlobalMat(meshP, meshT, ExEM)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
meshT	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
ExEM	Expanded element matrix

Value

Produces large (n x n) global matrix

GlobalMat	Global matrix

Examples

```
# meshP = MeshPts$p
# meshT = MeshPts$T
# ExEM = ExEM #expanded element matrix
# GlobalMat(ExEM, meshT)
```

`LocalStress` *LocalStress*

Description

Calculates local stress and strain for triangular elements of the mesh

Usage

```
LocalStress(meshP, meshT, Y, Nu, GlobalND)
```

Arguments

`meshP`	Matrix (2 x n) containing coordinate points of the mesh nodes.
`meshT`	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
`Y`	Value of Young's (Elastic) modulus
`Nu`	Value of Poisson's ratio
`GlobalND`	Global nodal displacement, return from function NodeDis

Value

Completes FEM by calculating values of stress and strain, produces three (3) [3 x n] matrix.

`Strain`	Calculated strain. [x, y, tau]
`Stress`	Calculated stress in pascals. [x, y, tau]
`StressStrain`	Stress as calucated from strain. [x, y, tau]

Examples

```
# Y = 17e9
# Nu = 0.45
# meshP = MeshPts$p
# meshT = MeshPts$T
# GlobalND = GlobalND
# LocalStress(meshP, meshT, Y, Nu, GlobalND)
```

ManualAdjust *ManualAdjust*

Description

Allows for manual refinement of the triangular mesh generated based on given parameters. Will remove triangle elements that are identified in the input (loc).

Usage

```
ManualAdjust(meshP, meshT, edge, centroid, loc)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
meshT	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
edge	Coordinate points of the initial geometry.
centroid	Matrix (2 x n) of triangle elements.
loc	String containing the number of the meshT matrix row of the triangle chosen to be removed.

Value

Generates new mesh and centroid tables

Meshpts	Includes both new mesh coordinate points and triangulation of points.
Centroids	Centroid positions for each triangle element.

Examples

```
# meshP = MeshPts$p
# meshT = MeshPts$T
# edge = Line
# centroid = tCentroids
# loc = c(11, 12, 13) #number of the centroid point of the triangle that is chosen for removal
# ManualAdjust(meshP, meshT, edge, centroid, loc)
```

NodeDis *NodeDis*

Description

Calculates global nodal displacements

Usage

```
NodeDis(meshP, REM, ForceV, NodeKnownL)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
REM	Reduced element matrix, returned from function ReducedEM.
ForceV	Reduced force vector matrix containing the model load parameters. Returned from function ForceVector.
NodeKnownL	data frame with constraint parameters applied to each node in the x and y directions. Formatted for use in reduced element matrix. Generated from ApplyBC function.

Value

Produces tables with new node coordinates that are produced by the geometry under an applied load.

NodeDis	Nodal displacement
GlobalND	Nodal displacement in the global environment

Examples

```
# meshP = MeshPts$p #mesh points
# REM = REM #reduced element martix
# ForceV = ForceVector #reduced force vector
# NodeKnownL = Nodelist #boundary conditions at nodes
# NodeDis(meshP, REM, ForceV, NodeKnownL)
```

PlotSystem *PlotSystem*

Description

Generates heat map for given stress or strain on the geometry. Threshold values for the color must be assigned.

Usage

```
PlotSystem(
  meshP,
  meshT,
  PlotVal,
  a,
  b,
  c,
  d,
  e,
  f,
  g,
  h,
  i,
  j,
  Oc,
```

```
  ac,
  bc,
  cc,
  dc,
  ec,
  fc,
  gc,
  hc,
  ic,
  jc
)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
meshT	Matrix (3 x n) containing the number of the coordinate point that forms a given triangle within the mesh.
PlotVal	Value to be plotted, either stress or strain, return from function LocalStress function.
a	Threshold 1
b	Threshold 2
c	Threshold 3
d	Threshold 4
e	Threshold 5
f	Threshold 6
g	Threshold 7
h	Threshold 8
i	Threshold 9
j	Threshold 10
0c	Color for all zero values
ac	Color 1
bc	Color 2
cc	Color 3
dc	Color 4
ec	Color 5
fc	Color 6
gc	Color 7
hc	Color 8
ic	Color 9
jc	Color 10

Value

Plot of colored polygon with mesh colored based on the plot value

Examples

```
# meshP = MeshPts$p #mesh points
# meshT = MeshPts$T #triangulation list
# PlotVal = FEAtest$Stress  #output value wanted in plot
# Kol =  3 #plotting shear stress/strain
# a= 1; b= 2; c=3; d=4; e=5; f=6; g=7; h=8; i=9; j=10 #color threshold values from min to max
# Ox = "mediumpurple"; ax = "dodgerblue"; bx = "darkturquoise"; cx = "seagreen3"; dx = "oliverab3"; ex = "gold";
# PlotSystem(meshP, meshT, PlotVal, a, b, c, d, e, f, g, h, i, j, Oc, ac, bc, cc, dc, ec, fc, gc, hc, ic, jc)
```

ReducedEM *ReducedEM*

Description

Reduced stiffness matrix - use boundary condition to reduce matrix to smaller form by removing systems that are bound.

Usage

```
ReducedEM(GMat, NodeKnownL)
```

Arguments

GMat	Global stiffness matrix
NodeKnownL	data frame with constraint parameters applied to each node in the x and y directions. Formatted for use in reduced element matrix. Generated from ApplyBC function.

Value

Produces a large matrix.

ReducedEM	Reduced element matrix.

Examples

```
# GMat = GMat #stiffness matrix
# NodeKnownL = Nodelist #boundary conditions at nodes
# ReducedEM(GMat, NodeKnownL)
```

ReducedSF *ReducedSF*

Description

Reduced matrix of surface forces

Usage

```
ReducedSF(meshP, ExSurf)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
ExSurf	Expanded surface matrix, output from ExpandSFT

Value

Produces a large matrix.

RSF	Produces a large, reduced surface force matrix

Examples

```
# meshP = MeshPts$p
# ExSurf = ExSurf #expanded surface matrix
# ReducedSF(meshP, ExSurf)
```

SinglePoly *SinglePoly*

Description

Generates a mesh for polygon with a single continuous geometry

Usage

```
SinglePoly(x, y, ptDS, ptDL)
```

Arguments

x	X-coordinates for geometry.
y	Y-coordinates for geometry.
ptDS	Density of points desired within the geometry.
ptDL	Density of points desired at the perimeter of the geometry.

Value

Coordinate points of nodes distributed within and on the line of a given geometry.

AllCoords	all coordinate points distributed across the geometry.
Within	all coordinate points within the geometry ONLY.
Line	all coordinate points that lay on the perimeter of the geometry ONLY.

Examples

```
x= c(0.23, 0.61, 0.75, 0.61, 0.23, -0.23, -0.61, -0.75, -0.61, -0.23, -0.23)
y= c(0.62, 0.38, 0.51, -0.38, -0.62, -0.62, -0.38, -0.51, 0.38, 0.62, 0.65)
ptDS = 30
ptDL = 20
SinglePoly(x, y, ptDS, ptDL)
```

SurfaceTraction *SurfaceTraction.*

Description

Element Surface Traction - generates the column matrix for uniformly distributed surface traction. If surface traction is not present, assign SFTensile and SFShear a value of 0.

Usage

```
SurfaceTraction(meshP, SFTensile, SFShear, Length, Thick, area)
```

Arguments

meshP	Matrix (2 x n) containing coordinate points of the mesh nodes.
SFTensile	Magnitude of tensile surface traction
SFShear	Magnitude of positive shear traction
Length	Truss length
Thick	Triangle element thickness
area	Triangle element area

Value

List of element matrices containing surface forces.

SurfT	List of surface forces for each element.

Examples

```
# SFShear = 0
# SFTensile = 0
# Length = TrussLength
# area = Area
# SurfaceTraction(meshP, SFTensile, SFShear, Length, Thick, area)
```

ThreshPts *ThreshPts*

Description

Clean node distribution within or outside of geometry. Optional function for complex geometries.

Usage

```
ThreshPts(coords, thresh, edge)
```

Arguments

coords	Nodal coordinates
thresh	Threshold for point removal. Ranges include: 500000-50000000
edge	Coordinate points of the initial geometry.

Value

Coordinate points of valid nodes.

CleanedNodes	Matrix of new nodes that abide by given threshold rules.
NodeReport	Report identifying with nodes were kept and which were removed.

Examples

```
# coords = Within
# thresh = 500000
# edge = Line
# ThreshPts(coords, thresh, edge)
```

triangulate0 *triangulate0*

Description

Triangulation by Delaunayn algorithm. Automatically generates a triangular mesh for a geometry containing nodal points.

Usage

```
triangulate0(u0, edge)
```

Arguments

u0	Matrix (2 x n) of node coordinates within the geometry.
edge	Matrix (2 x n) of coordinate points on the perimeter of the geometry.

Value

Produces data for generated mesh.

`Meshpts`	Includes both new mesh coordinate points and triangulation of points.
`Centroids`	Centroid positions for each triangle element.

Examples

```
# u0 = CleanedNodes
# edge = Line
# triangulate0(u0, edge)
```

Index

ApplyBC, 2
AutoAdjust, 3

Dimensions, 4

ElementMat, 5
ExpandEM, 5
ExpandSFT, 6

FEMStrain, 7
FEMStress, 8
ForceVector, 10

GLForces, 10
GlobalMat, 11

LocalStress, 12

ManualAdjust, 13

NodeDis, 13

PlotSystem, 14

ReducedEM, 16
ReducedSF, 17

SinglePoly, 17
SurfaceTraction, 18

ThreshPts, 19
triangulate0, 19