Simulator for Active Origami: Functions

This documentation gives a summary of all functions used in this simulation package. We will focus on the input & output & structures of each functions. We will start with pre-processing functions.

Mesh_IdentifyCrease

- identify the creases of old (input) geometry for future process.
- Input: [node0], {panel0}
- Output: oldCreaseNum, [oldCreaseConnect], [oldCreaseType]
- Code Structure:

```
For (1 → oldPanelNum)
    check all creases of each panel, records how many
    time a crease "i-j" appears. If appearing only
    once, it is at boundary.
End For
```

Mesh_CompliantCreaseGeometry

- This is the major preprossessing code that generates the topology & geometry of the compliant crease origami after meshing.
- Input: [node0], {panel0}, oldCreaseNum,
  [oldCreaseConnect], [oldCreaseType]
  {modelGeometryConstant}
- Output: [newNode], {newPanel}, [barType],
  [barConnect], [sprIJKL], type1BarNum
  panelInnerBarStart, centerNodeStart,
  [newNode2OldNode], [newCrease2OldCrease],
  [newPanel2OldPanel], newPanelNum.

- Code Structure:

For (1 → oldPanelNum)

① Identify tempNodeType

i.e. 4 [1] 5 [1] 6

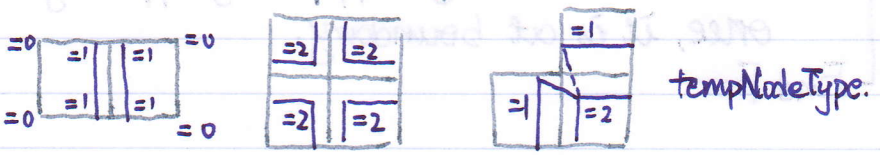
[1] [>1] [1] ⇝ oldCreaseType.

1 [1] 2 [1] 3

loop through nodes within each panel, check how many crease connects to a node has >1 type.

tempNodeType = { 0 BoundaryNodes (0 [>1] crease); 1 PorticalBoundary (1 [>1] crease); 2 Internal (2 [>1] crease) }

② Shrink the size of each panel.



tempNodeType.

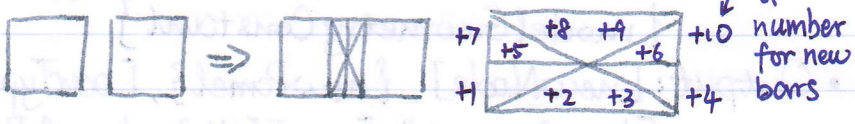
this step create the space for compliant crease.

③ Construct: [barType], [barConnect], & others
type1BarNum, [newNode]

EndFor

For (1 → oldCreaseNum)

① Generate the Compliant Creases



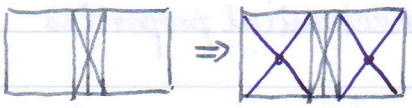
number for new bars

the numbering starts from type1BarNum.
Each old crease generate 10 more bars.

② Construct: [barType], [barConnect] & others
panelInnerBarStart & centerNodeStart

EndFor

For (1 → oldPanelNum)



numbering starts from centerNode Start (for Node) & panelInnerBarStart (for Bar)

add additional nodes & bars to the panels.
construct: [barType], [barConnect], & others

End For

For (1 → oldCreaseNum)

Construct [sprIJKL] for type 5 bars

End For

Mesh_NumberingForContact

- Generates the [creaseRef], which is used for mapping old creases back to new bars. (Not just used for contact).
- Input: [newCrease2OldCrease], [oldCreaseNum]
- Output: [creaseRef]

Mesh_BarLength

- Calculate the length of bars. Based on the new origami pattern after generating meshing.
- Input: [newNode], [newBarConnect]
- Output: [barLength]

Note: Because it usually needs manual adjustment after generating the compliant crease geometry. We create a separate function to calculate the [barLength].

Mesh_MechanicalProperty

- This function generate the mechanical properties for the solver for analysis.
- Input: {modelMechanicalConstant}, {modelGeometryConstant}
  [oldCreaseType], oldCreaseNum, [creaseRef]
  [barLength], {panelO}, [barConnect], {newNode}
- Output: [barArea], [sprK], [sprTargetZeroStrain]
  [sprFoldingSequence].
- CodeStruetuve:
  - For (1 → oldCreaseNum)
    using [creaseRef] to calculate stiffness parameters for bars & springs in the crease region.
  - EndFor
  - For (1 → oldPanelNum)
    calculate the stiffness parameters for bars & springs within each panels.
  - EndFor

Next, we give a brief introduction of functions that implement the bar and spring elements and those that perform the nonlinear solver algorithms.

Bar_Cons

- Gives the constitutive model of bar elements.
- Input: [barType], [Ex], panelE, creaseE
- Output: [Sx], [C].

Bar_Strain

- Calculate the strain of each bar elements
- Input: [U], [newNode], [barArea], [barConnect], [barLength]
- Output: [Ex].

Bar_GlobalForce

- Calculate the global inner force coming from the contribution of bar elements.
- Input: [U], [Sx], [C], [barArea], [barLength]
  [barConnect], [newNode].
- Output: [Tbar]

Bar_GlobalStiffAssemble

- Calculate the global stiffness matrix from the contribution of bar elements.
- Input: [U], [Sx], [C], [barArea], [barLength]
  [barConnect], [newNode]
- Output: [Kbar]

Spr_Cons

- Gives the constitutive relationship of spring elements:
- Input: [sprTargetZeroStrain], [theta], [sprK] [creaseRef], oldCreaseNum, panelInnerBarStart [sprIJKL], [newNode], [U], compliantCreaseOpen
- Output: [M], [sprKadj]. *

Spr_Theta

- Calculates the rotation angle of eeeeh spring elements.
- Input: [U], [sprIJKL], [newNode]
- Output: [theta]

Spr_GlobalForce

- Calculates the global inner force vector from the contribution of spring elements.
- Input: [U], [M], [sprIJKL], [sprKadj], [newNode] *
- Output: [Tspr]

Spr_GlobalStiffAssemble

- Calculate the global stiffness matrix from spring.
- Input: [U], [M], [sprIJKL], [sprKadj], [newNode] *
- Output: [Kspr].

Note: we use [sprKadj] because the [sprK] is adjusted to prevent the spring from folding an entire 180°.

Next are several contact model related functions:

Contact_P2TDistance:

- This function determines the zones of the point & calculate the point to triangle distance.
- Input: [Point], [T1], [T2], [T3]
- Output: d, zone, N1, N2.

Contact_DerivativeZone0:

- This function calculates the derivatives for Zone0.
- Input: [Point], [T1], [T2], [T3]
- Output: [Dd2x2], [Ddx]

Contact_DerivativeZone1:

- This function calculates the derivatives for Zone1
- Input: [Point], [T1], d
- Output: [Dd2x2], [Ddx]

Contact_DerivativeZon2:

- This function calculate the derivatives for Zone 2
- Input: [Point], [T1], [T2]
- Output: [Dd2x2], [Ddx]

Contact_AssembleForceStiffness:

- This function calculates the global stiffness matrix and the global force vector.
- Input: {panel0}, [newNode2OldNode], [newNode] {U}, ke, dOedge, dOcenter, centerNodeStart, compliantCreaseOpen.
- Output: [Tcontact], [Kcontact]
- Code Structure:

```
For (i=1:oldPanelNum).
    ① pick a point in the panel as thePoint;
    For (i=1:oldPanelNum).
        ⓐ Check if contact needs to be calculated & pick 3 nodes;
        ⓑ Calculate Point to Triangle distance;
        ⓒ Calculate Derivatives & assemble force
          & stiffness matrix based on zones;
    End For
End For
```

Note: As can be seen from the Code Structure, checking contact requires looping through all elements twice, so contact detection is quadratically more expensive than assembling the stiffness matrix.

Solver_ModKforSupp:

- This function gives a modified inner force & stiffness matrix after considering the support. The updated inner force vector & stiffness matrix can be send to "Ax = b" solver directly.
- Input: [K], [Supp], [Tinput], elasticSupportOpen, [suppElastic], [U]
- Output: [Kwsupp], [T].

Note: this is the more accurate method & can consider the elastic deformable support.

Solver_ModKforSuppBigNum:

- This is a numerical method that modifies the stiffness matrix & inner force vectors by filling in "Big" numbers.
- Input: [K], [Supp]
- Output: [Kwsupp]

Note: this method is not considering the deformable support. It is a numerical method & is usually not called in the solver. I created it because some time I use it to check the rank of [K].

Next, we will give a short documentation on the solvers created for this package.

Solver_Assemble:

- This solver simulates the self folding proces of an origami; Basically, the stress free angle of rotational springs are updated and the new equilibrium position is tracked with a Newton-Raphson iteration.
- Input: [panel0], [newNode2OldNode], [newNodeBeforeSelfFold]*
  [barConnect], [barType], [barLength],
  [barArea], [sprIJKL], [sprK], [sprTargetZeroStrain]
  [sprFoldingSequence], [creaseRef], oldCreaseNum
  [assembleConstant], [modelGeometryConstant]
  [modelMechanicalConstant],
  [supportInfo], [load]*
- Output [U], [UhisAssemble], [strainEnergyAssemble]

Note: the [load] is here because the self-fold could happen after applying certain load (i.e. Gravity).

Note: there is no input [U] for this function so the [newNodeBeforeSelfFold] is the current configuration: that is to say, the configuration before self fold.

[newNodeBeforeSelfFold] = [newNode] + [U]

current deformation field before assemble

Solver_LoadingNR:

- This solver implement the loading proces of the origami. The function uses an incremental-iterative Newton-Raphson method.
- Input: {panel0}, [newNode2OldNode], oldCreaseNum,
  * [newNode], [barConnect], [barType], [barArea] [barLength], [sprIJKL], [sprK], [sprTargetZeroStrain], [creaseRef], [load]* [supportInfo], [U]*, [loadConstant], [modelGeometryConstant], [modelMechanicalConstant]
- Output: [U], [UhisLoading], [loadHis] [strainEnergy], [nodeForce], [loadForce] [contactForce].

Note: [newNode], [U] are in the input so the configuration before the start of loading is [newNode] + [U].

Note: [load] gives the loading per increment. With N-R solver the final loading obtained is:
[load] * increStep;

Solver_LoadingMGDCM

- This solver implement the modified generalized displacement controlled method to load the origami.
- Input: same as **Solver_LoadingNR**
- Output: same as **Solver_LoadingNR**

Note on selection of Solver:

NR is usually better for contact simulations, while the MGDCM can capture the snapthrough for bistable structures & multistable structures.

Solver_LoadingDC

- This function implement the standard displacement controlled method to solve the loading of origami.
- Input: same as **Solver_LoadingNR**.
  plus an extra [dispController]
- Output: same as **Solver_LoadingNR**.

Note: the dispController basically gives the selected degree of freedom for the DC method. Usually, we will pick the dof that has the largest deformation.

Next, we give a brief introduction of the functions that is related to the analysis of the heat transfer problems.

Thermal_Assemble Conduct Mat:

- This function generate the thermal conductivity matrix for the heat transfer analysis.
- Input: [newNode], {newPanel}, [barConnect] [barLength], [barType], [U], {modelThermalConstant} {modelMechanicalConstant}, [newCrease2OldCrease] [newPanel2OldPanel]
- Output: [thermalMat], [thermalNodeNum]
- Code Structure:

```
For (i=1: newPanelNum)
    Find out all thermal BC panels;
End For
For (i=1: newPanelNum \ BCpanelNum)
    For (j=1: BCPanelNum)
        Check closest distance for t_max
    End For
End For
For (i=1: newPanelNum)
    Assemble conductivity matrix for within structure heat transfer
End For
For (i=1: envLayer)
    Assemble matrix for structural air heat loss
End For
```

Thermal_SolveTemperature

- This function solves the temperature profile.
- Input: [qin], [thermalMat], thermalNodeNum
  {modelThermalConstant}.
- Output: [T], [indexArray].

Note: this is just a linear solver so the structure is very simple. Just take care of the "support" then solve $Ax = b$.

Thermal_Timoshenko

- This function solves the rotation comming out of the elavating temperature.
- Input: Tcrease, {modelTimoshenkoConstant}
  {modelThermalConstant}, creaseWidth
- Output: rot.

In the following part, I will introduce the functions used for plotting figures for post processing.

Plot_DeformedHis

- This function plots the deformation history of an active origami structure (include an assemble and a loading process). It also generate a GIF.
- Input: [viewControl], [newNode], {newPanel}
  [UhisLoading], [UhisAssemble].

Note: this is the major function used to check if the loading & assemble process make sense.

Plot_DeformedHisAndCalcTheta

- This function plots the deformation history and at the same time calculate the rotation angle of one compliant crease (or not compliant crease).
- Input: [viewControl], [newNode], {newPanel}
  [UhisLoading], [UhisAssemble],
  [sprIJKL], [angleNum].
- Output: [angleHis]

Note: patch function is used for plotting the figure of origami panels. Two loops "for (i=1:Incre1)" and "for (i=1:Incre2)" is used for the self-assemble & loading processes.

Plot_LoadHis

- The function plots the force-displacement curves. By default it use norm of disp vector for displacement value, but we can set up it with others.
- Input: [loadHis], [UhisLoading]

Plot_Energy:

- The function plots the strain energy curves for both the self-assemble and loading processes.
- Input: [UhisLoading], [strainEnergyLoading]
  [UhisAssemble], [strainEnergyAssemble].

Plot_OriginalMeshing

- The function plots the original input shape of the origami system..
- Input: [node0], {panel0}, oldCreaseNum
  [oldCreaseConnect], [viewControl].

Plot_ImprovedMeshing

- The function plots the meshed origami system.
- Input: [viewControl], [newNode],
  {newPanel}, [barConnect].

Plot_DeformedShape:

- The function plots the deformed shape of an origami and the reference configuration.
- Input: [viewControl], [newNode], [deformNode], {newPanel}.

Plot_DeformedShapeOnly:

- The function plots the deformed shape of origami without the referenee configuration.
- Input: [viewControl], [deformNode], {newPanel}.

Plot_LoadAndReaction

- This function plots the origami and overlay the support reaction and loading force on top.
- Input: [viewControl], [newNode], [deformNode] {newPanel}, [load], [supp], [nodeForce] [loadForce].

Plot_ContactForce:

- This function plots the deformed origami and overlay the contact force on top.
- Input: [viewControl], [newNode], [deformNode] {newPanel}, [contactforce].

Plot_DeformedHisTemp:

- This function generate an animation of the origami deformation history and overlay the temperature history of nodes on top.
- Input: [viewControl], [newNode], [VhisThermal] {newPanel}, [temperatureHistory]

Plot_DeformedShapeTemp:

- This function plots the deformed origami and overlay the temperature on top.
- Input: [viewControl], [newNode], [deformNode] {newPanel}, [T], [termalBoundaryPanels] [newPanel2OldPanel].