LMGC90v2_Pre : le pré-processeur de LMGC90

A. Martin, M. Bagnéris, F. Dubois, R. Mozul

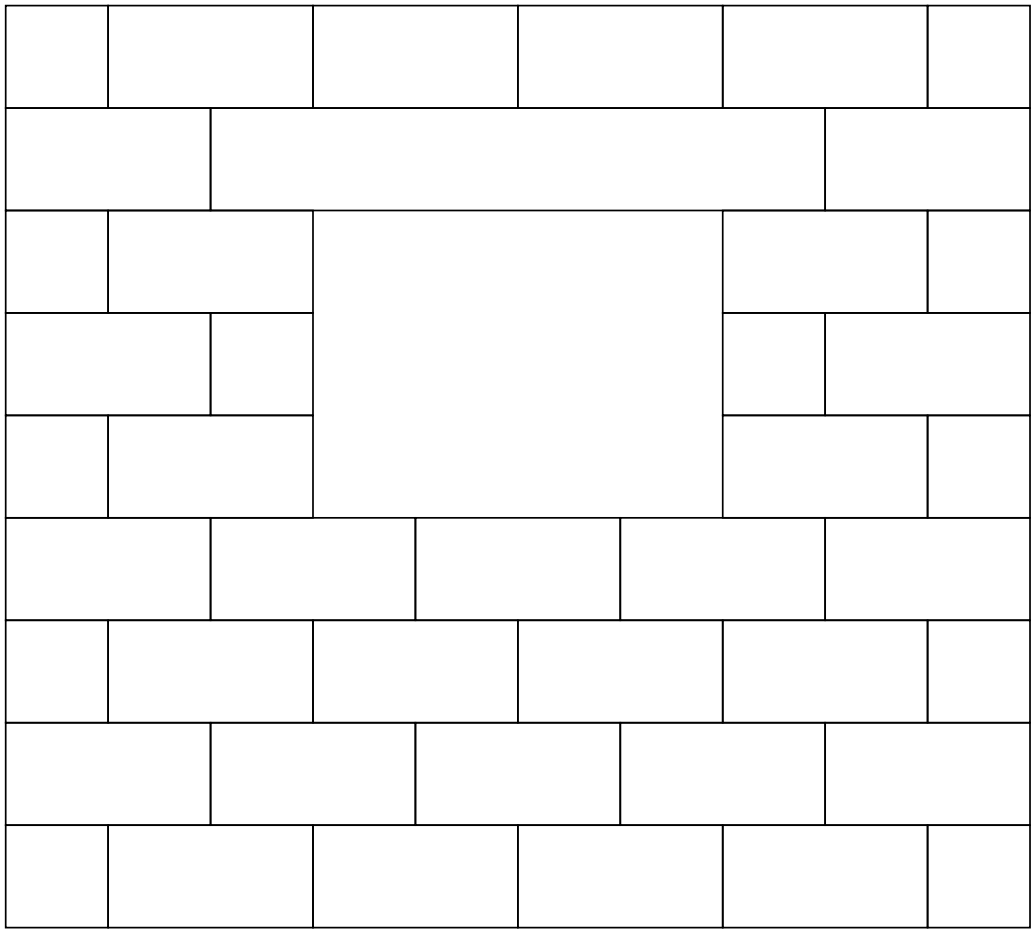
Laboratoire de Mécanique et Génie Civil

Formation LMGC90 - janvier 2013

Sommaire

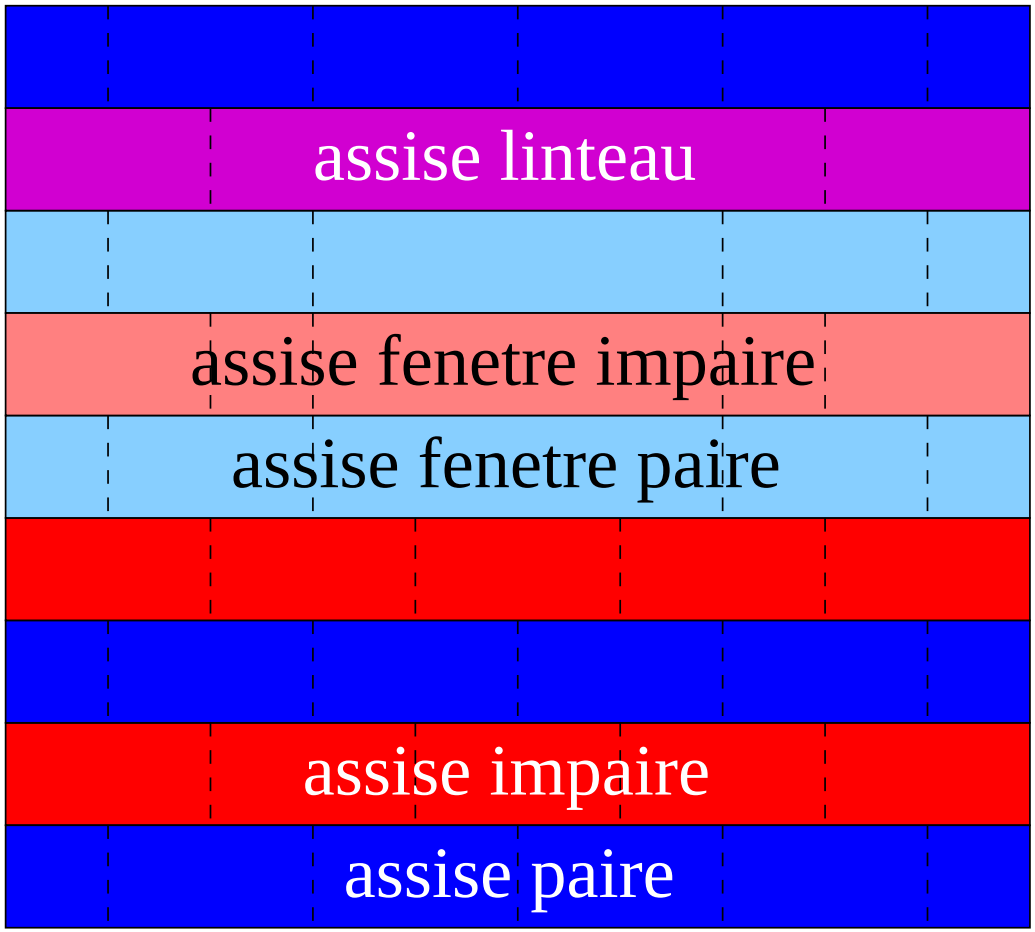
Algorithme de construction d'un mur

- un exemple de mur.



Algorithme de construction d'un mur

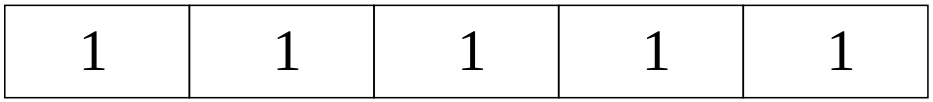
- décomposition du mur en assises.



Algorithme de construction d'un mur

- décomposition des assises en briques.
  - "assise paire" :

  - "assise impaire" :

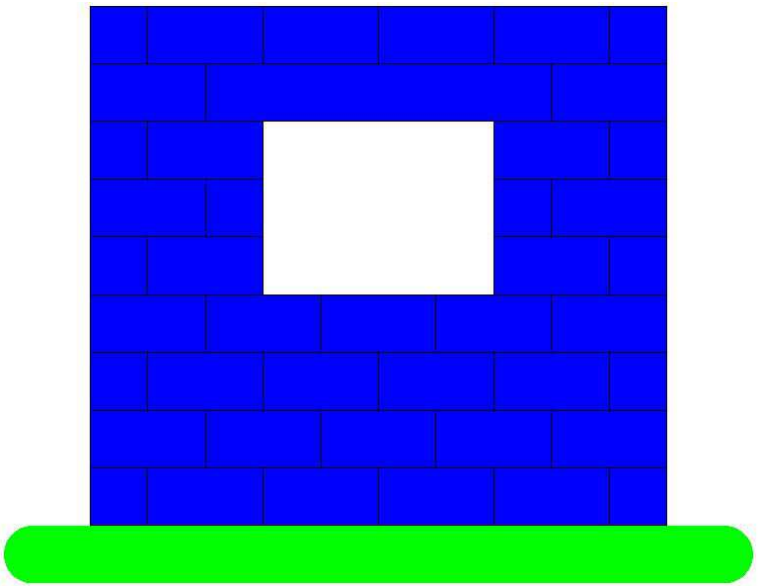


  - "assise fenetre paire" :

  - "assise fenetre impaire" :

  - "assise fenetre linteau" :

Mur avec fenêtre



- brique de référence : 10 cm x 5 cm
- mur :
  - appareillage en paneresse,
  - longueur : 5 briques,
  - hauteur : 7 assises,
- fenêtre :
  - largeur : 2 briques,
  - hauteur : 3 assises,
  - centrée horizontalement,
  - hauteur sous allège : 4 briques.

Mur avec fenêtre : définition des briques et des joints

```
1  # definition des briques utilisees
2  #   * dimensions de la brique de reference
3  L = 1.e-1; h = 5.e-2
4  #   * la brique entiere :
5  moellon=brick2D('brique', L, h)
6  #   * la demi-brique :
7  demi_moellon=brick2D('demi-brique', 0.5*L, h)
8  #   * brique speciale pour le linteau :
9  linteau=brick2D('linteau', 3.*L, h)
10 #   * brique fantome, pour positionner la fenetre :
11 fantome=brick2D('ghost', 2.*L, h)
12 # definition de l'epaisseur des joints
13 # horizontaux et verticaux
14 epaisseur_joint_horizontal=0.
15 epaisseur_joint_vertical=0.
```

Mur avec fenêtre : définition des assises et du mur

```
1  # definition des assises, comme des listes de briques
2  assise_paire=[demi_moellon, moellon, moellon, moellon,
3      moellon, demi_moellon]
4  assise_impaire=[moellon, moellon, moellon, moellon,
5      moellon]
6  assise_fenetre_paire=[demi_moellon, moellon, fantome,
7      moellon, demi_moellon]
8  assise_fenetre_impaire=[moellon, demi_moellon, fantome,
9      demi_moellon, moellon]
10 assise_linteau=[moellon, linteau, moellon]
11 # definition du mur, comme une liste d'assises
12 mur=[assise_paire, assise_impaire, assise_paire,
13     assise_impaire, assise_fenetre_paire,
14     assise_fenetre_impaire, assise_fenetre_paire,
15     assise_linteau, assise_paire]
```

Mur avec fenêtre : boucle de construction du mur

```
1  # initialisation de la position du centre d'inertie
2  # de la brique courante
3  x=0.; y=0.
4  # pour chaque assise :
5  for j in range(0, len(mur)):
6      # recuperation de l'assise courante
7      assise=mur[j]
8      # definition de la couleur des briques selon la
9      # parite de l'indice de l'assise
10     if j % 2 == 0:
11         color='BLEUx'
12     else:
13         color='REDxx'
14     # reinitialisation de l'abscisse du centre d'inertie
15     # de la prochaine brique a 0
16     x=0.
```

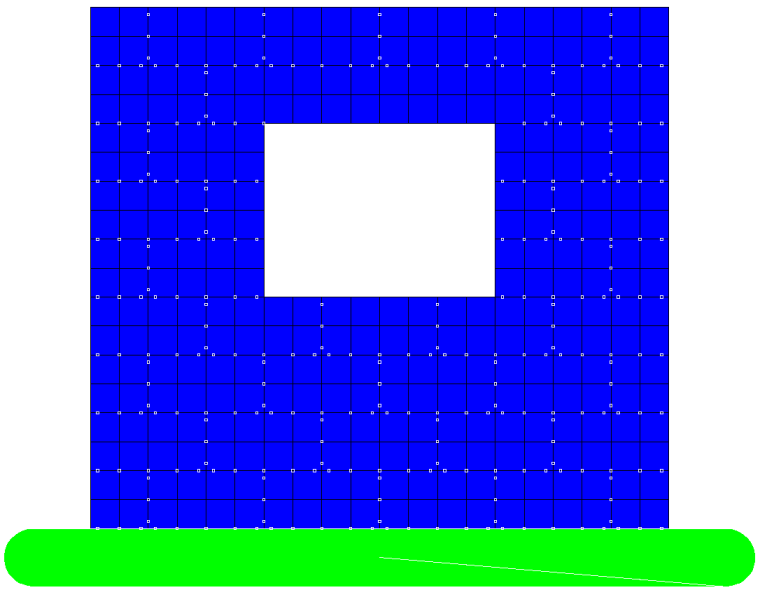
Mur avec fenêtre : boucle de construction du mur (suite)

```
1     # pour chaque brique de l'assise courante
2     for i in range(0, len(assise)):
3         # recuperation la brique courante
4         brique=assise[i]
5
6         # si c'est la premiere brique de l'assise
7         if i == 0:
8             # incrementation de l'ordonnee du centre
9             # d'inertie de la prochaine brique
10            y += 0.5*brique.ly
11
12        # incrementation de l'absisse du centre d'inertie
13        # de la prochaine brique
14        x += 0.5*brique.lx
```

Mur avec fenêtre : boucle de construction du mur (suite et fin)

```
1          # si la brique n'est pas une brique fantome
2          if brique.name != 'ghost':
3              # creation d'un nouvel avatar pour la brique
4              body=brique.rigidBrick(center=[x, y],
5                  model=mR2D, material=stone, color=color)
6              # ajout de la brique dans le conteneur de
7              # corps
8              bodies.addAvatar(body)
9          # incrementation de l'abscisse du centre d'inertie
10         # de la prochaine brique, pour obtenir la position
11         # du bord gauche de la prochaine brique
12         x += 0.5*brique.lx + epaisseur_joint_vertical
13     # incrementation de l'ordonnee du centre d'inertie de
14     # la prochaine brique, pour obtenir la position du
15     # bord superieur de la prochaine brique
16     y += 0.5*brique.ly + epaisseur_joint_horizontal
```

Mur avec fenêtre



- brique de référence : 10 cm x 5 cm
- mur :
  - appareillage en paneresse,
  - longueur : 5 briques,
  - hauteur : 7 assises,
- fenêtre :
  - largeur : 2 briques,
  - hauteur : 3 assises,
  - centrée horizontalement,
  - hauteur sous allège : 4 briques.

Mur avec fenêtre : définition des briques et des joints

```
1  # definition des briques utilisees
2  #   * dimensions de la brique de reference
3  L = 1.e-1; h = 5.e-2
4  #   * la brique entiere :
5  moellon=brick2D('brique', L, h)
6  #   * la demi-brique :
7  demi_moellon=brick2D('demi-brique', 0.5*L, h)
8  #   * brique speciale pour le linteau :
9  linteau=brick2D('linteau', 3.*L, h)
10 #   * brique fantome, pour positionner la fenetre :
11 fantome=brick2D('ghost', 2.*L, h)
12 # definition de l'epaisseur des joints
13 # horizontaux et verticaux
14 epaisseur_joint_horizontal=0.
15 epaisseur_joint_vertical=0.
```

Mur avec fenêtre : définition des assises et du mur

```
1  # definition des assises, comme des listes de briques
2  assise_paire=[demi_moellon, moellon, moellon, moellon,
3      moellon, demi_moellon]
4  assise_impaire=[moellon, moellon, moellon, moellon,
5      moellon]
6  assise_fenetre_paire=[demi_moellon, moellon, fantome,
7      moellon, demi_moellon]
8  assise_fenetre_impaire=[moellon, demi_moellon, fantome,
9      demi_moellon, moellon]
10 assise_linteau=[moellon, linteau, moellon]
11 # definition du mur, comme une liste d'assises
12 mur=[assise_paire, assise_impaire, assise_paire,
13     assise_impaire, assise_fenetre_paire,
14     assise_fenetre_impaire, assise_fenetre_paire,
15     assise_linteau, assise_paire]
```

Mur avec fenêtre : boucle de construction du mur

```
1  # initialisation de la position du centre d'inertie
2  # de la brique courante
3  x=0.; y=0.
4  # pour chaque assise :
5  for j in range(0, len(mur)):
6      # recuperation de l'assise courante
7      assise=mur[j]
8      # definition de la couleur des briques selon la
9      # parite de l'indice de l'assise
10     if j % 2 == 0:
11         color='BLEUx'
12     else:
13         color='REDxx'
14     # reinitialisation de l'abscisse du centre d'inertie
15     # de la prochaine brique a 0
16     x=0.
```

Mur avec fenêtre : boucle de construction du mur (suite)

```
1     # pour chaque brique de l'assise courante
2     for i in range(0, len(assise)):
3         # recuperation la brique courante
4         brique=assise[i]
5
6         # si c'est la premiere brique de l'assise
7         if i == 0:
8             # incrementation de l'ordonnee du centre
9             # d'inertie de la prochaine brique
10            y += 0.5*brique.ly
11
12        # incrementation de l'absisse du centre d'inertie
13        # de la prochaine brique
14        x += 0.5*brique.lx
```

Mur avec fenêtre : boucle de construction du mur (suite)

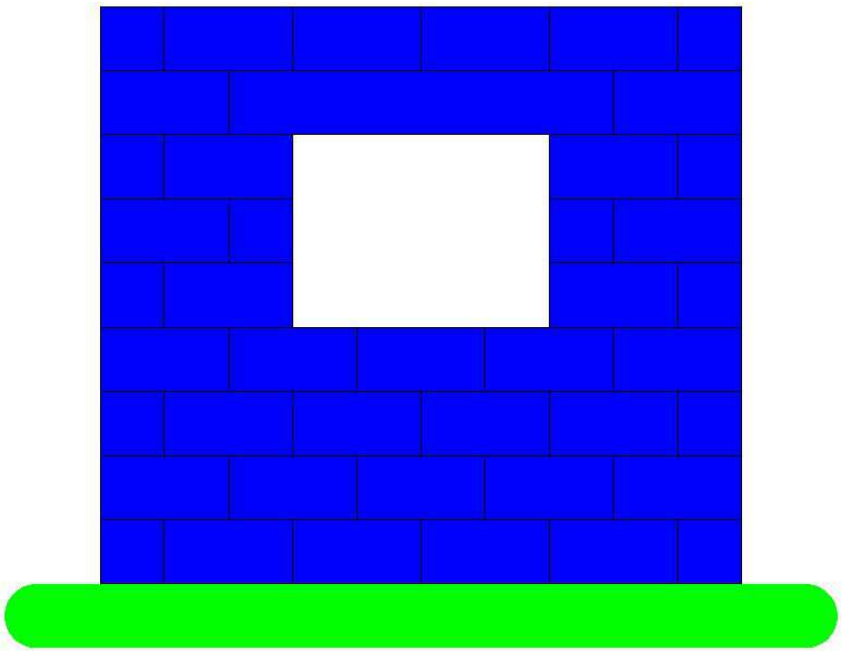
```
1         # si la brique n'est pas une brique fantome
2         if brique.name != 'ghost':
3             # calcul du nombre d'elements suivant chaque
4             # direction, en fonction des dimensions de la
5             # brique
6             nb_elem_x=int(math.floor(brique.lx/(0.25*L)))
7             nb_elem_y=int(math.floor(brique.ly/(0.5*h)))
8
9             # creation d'un nouvel avatar pour la brique
10            body=brique.deformableBrick(center=[x, y],
11                material=stone, model=m2Dl, type='Q4',
12                nb_elem_x=nb_elem_x, nb_elem_y=nb_elem_y)
13
14            # ajout de la brique dans le conteneur de
15            # corps
16            bodies.addAvatar(body)
```

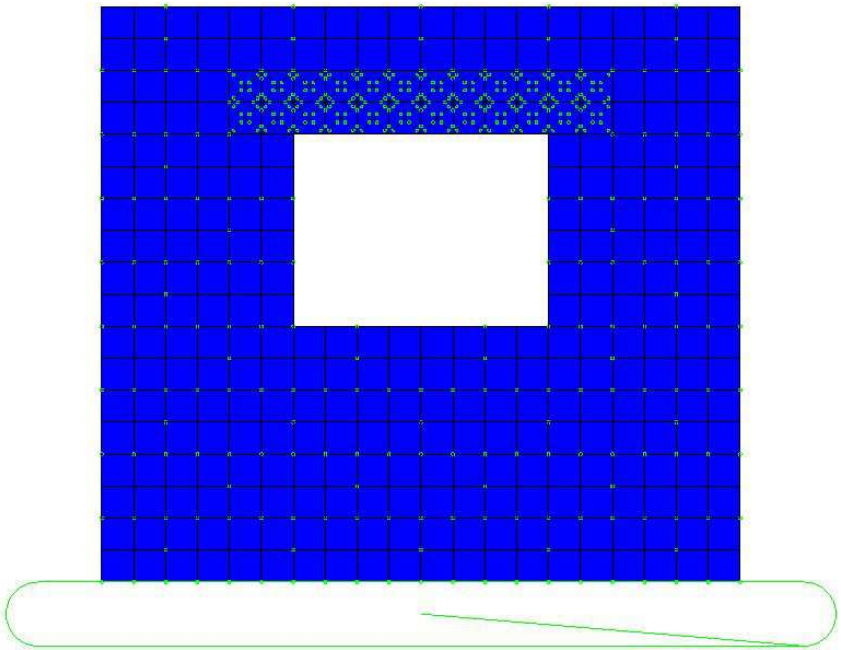
Mur avec fenêtre : boucle de construction du mur (suite et fin)

```
1          # incrementation de l'abscisse du centre d'inertie
2          # de la prochaine brique, pour obtenir la position
3          # du bord gauche de la prochaine brique
4          x += 0.5*brique.lx + epaisseur_joint_vertical
5
6      # incrementation de l'ordonnee du centre d'inertie de
7      # la prochaine brique, pour obtenir la position du
8      # bord superieur de la prochaine brique
9      y += 0.5*brique.ly + epaisseur_joint_horizontal
```

Maçonneries régulières

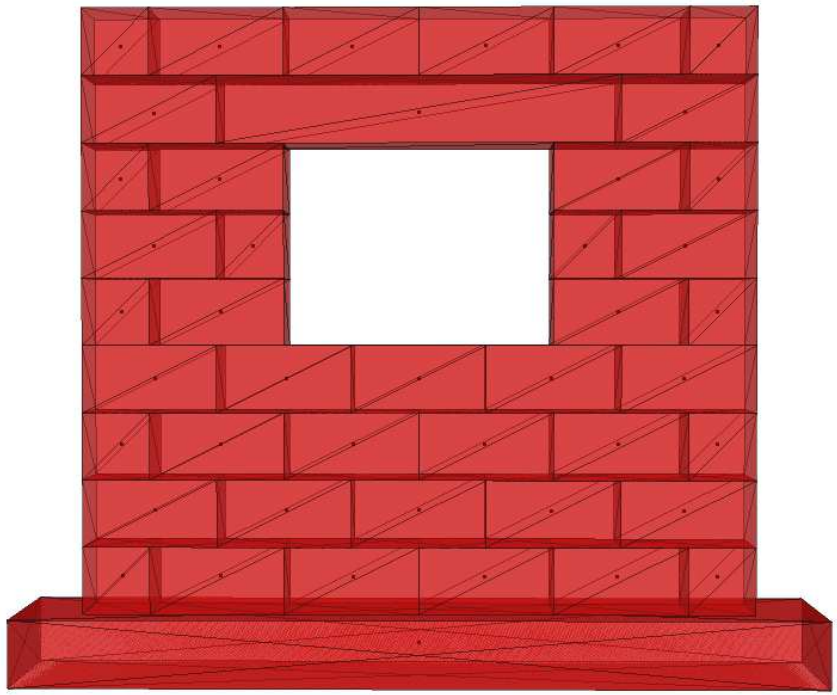
- classe "brique 2D" permettant l'écriture d'algorithmes de constuction de murs (différents appareillages) ▸ catalogue
  - écriture d'une brique sous la forme d'un corps rigide ou déformable, voire fissurable,





Maçonneries régulières

- classe "brique 2D" permettant l'écriture d'algorithmes de constuction de murs (différents appareillages) ▸ catalogue
  - écriture d'une brique sous la forme d'un corps rigide ou déformable, voire fissurable,
  - extension au 3D *via* l'extrusion,



Maçonneries régulières

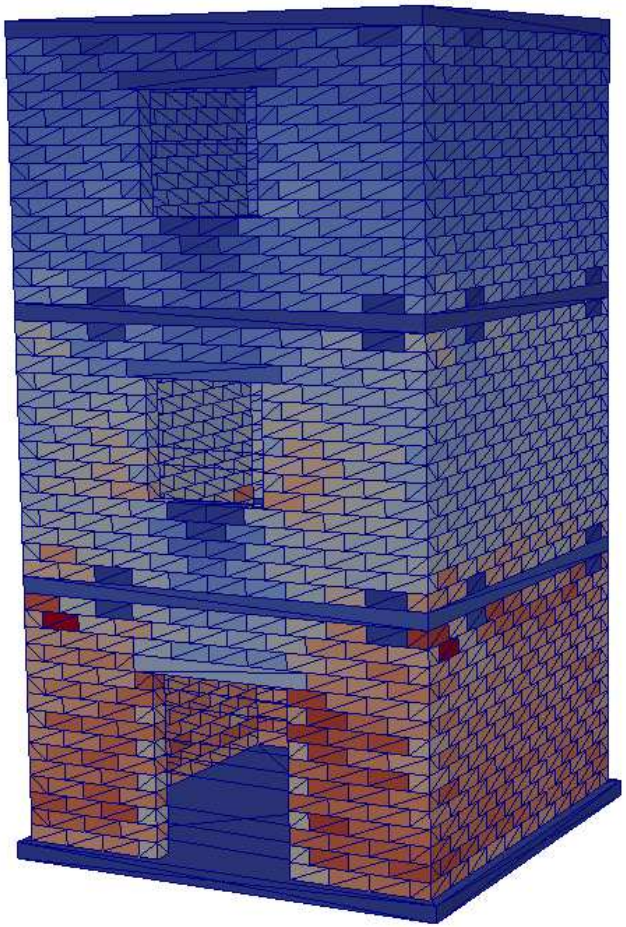
- classe "brique 2D" permettant l'écriture d'algorithmes de constuction de murs (différents appareillages) ▸ catalogue
  - écriture d'une brique sous la forme d'un corps rigide ou déformable, voire fissurable,
  - extension au 3D *via* l'extrusion,
- classe "brique 3D" permettant l'écriture d'algorithmes de constuction de murs (différents appareillages) ▸ catalogue
  - écriture d'une brique sous la forme d'un corps rigide,

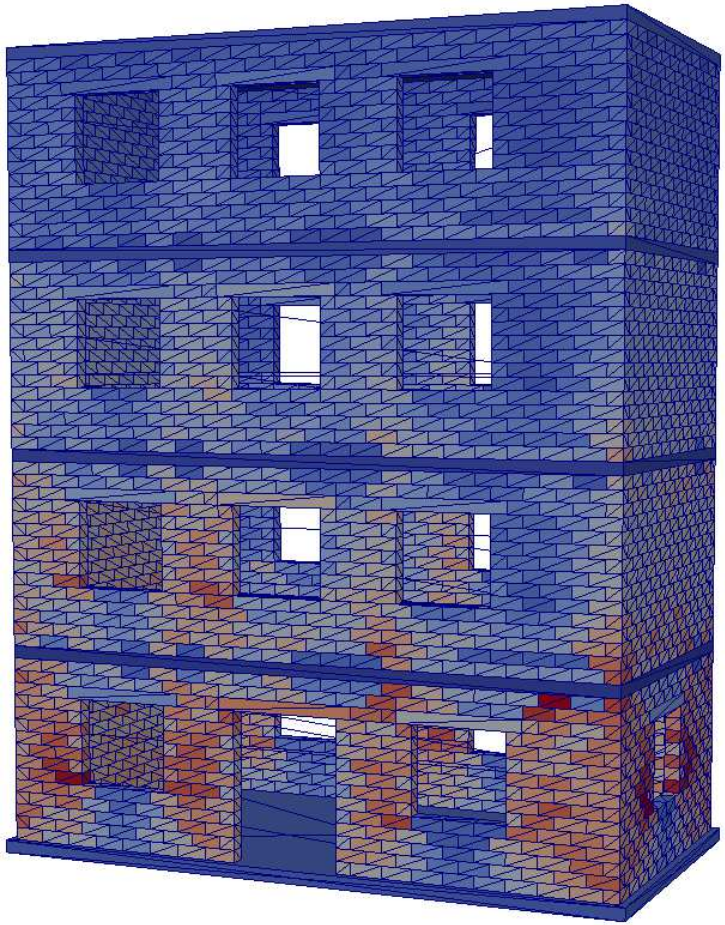
Maçonneries régulières

- classe "brique 2D" permettant l'écriture d'algorithmes de constuction de murs (différents appareillages) ▸ catalogue
  - écriture d'une brique sous la forme d'un corps rigide ou déformable, voire fissurable,
  - extension au 3D *via* l'extrusion,
- classe "brique 3D" permettant l'écriture d'algorithmes de constuction de murs (différents appareillages) ▸ catalogue
  - écriture d'une brique sous la forme d'un corps rigide,
- exemples disponibles dans les répertoires : examples/Pre/wall2D et examples/Pre/wall3D.

Maçonneries régulières

- aboutissement du concept : génération automatique de bâtiments réguliers (P. Taforel).





Bricks generation

1. brick2D
   1.1 constructor
   1.2 rigidBrick
   1.3 deformableBrick
   1.4 explodedDeformableBrick
2. brick3D
   2.1 constructor
   2.2 rigidBrick

the *brick2D* object

constructor

Create a brick2D object

$$my\_brick = brick2D(name, lx, ly)$$

where :

- name (input string), name of the brick
- lx (input double), length of the brick
- ly (input double), height of the brick
- my_brick (returned brick2D object), the brick2D object

the *brick2D* object

constructor

Create a rigid avatar from a brick2D object

$$body = my\_brick.rigidBrick(center, model, material, color = 'BLEUx', number = None)$$

where :

- center (input double array), coordinates of the center of inertia of the avatar
- model (input model object), model of the avatar
- material (input material object), material of avatar
- color (optional input string), color of the POLYG contactor
- number (optional input integer), index of the avatar
- body (returned avatar object), the avatar object

the *brick2D* object

deformableBrick

Create a deformable avatar from a brick2D object

$$\begin{aligned} body = & my\_brick.deformableBrick(\\ & center, material, model, type = '4T3', nb\_elem\_x = 1, nb\_elem\_y = 1, \\ & apabh = [], apabv = [], apabhc = 0.25, apabvc = 0.25, \\ & colors = ['HORIx', 'VERTx', 'HORIx', 'VERTx', number = None) \end{aligned}$$

where :

- center (input double array) coordinates of the center of inertia of the avatar
- material (input material object), material of avatar
- model (input model object), model of the avatar,
- type (optional input '4T3', '2T3', 'Q4' or 'Q8'), type of element to mesh the brick
- nb_elem_x (optional input integer), number of elements following x-axis
- nb_elem_y (optional input integer), number of elements following y-axis
- apabh (optional input double array), curvilign abscisses used to put candidate points on horizontal lines, size of the array is nb_elem_x
- apabv (optional input double array), curvilign abscisses used to put

the *brick2D* object

explodedDeformableBrick

Create a list of deformable avatars from a brick2D object

$$
\begin{aligned}
body = & my\_brick.deformableBrick(\\
& center, material, model, type = '4T3', nb\_elem\_x = 1, nb\_elem\_y = 1, \\
& apabh = [], apabv = [], apabhc = 0.25, apabvc = 0.25, \\
& colors = ['HORIx', 'VERTx', 'HORIx', 'VERTx', shift = 0)
\end{aligned}
$$

where :

- center (input double array) coordinates of the center of inertia of the brick
- material (input material object), material of the avatars
- model (input model object), model of the avatars
- type (optional input '4T3', '2T3', 'Q4' or 'Q8'), type of element to mesh the brick
- nb_elem_x (optional input integer), number of elements following x-axis
- nb_elem_y (optional input integer), number of elements following y-axis
- apabh (optional input double), curvilign abscisses used to put candidate points on horizontal lines
- apabv (optional input double), curvilign abscisses used to put candidate points on vertical lines

the *brick3D*

constructor

Create a brick3D object

$$my\_brick = brick3D(name, lx, ly, lz)$$

where :

- name (input string), name of the brick
- lx (input double), length of the brick
- ly (input double), depth of the brick
- lz (input double), height of the brick
- my_brick (returned brick3D object), the brick3D object

the *brick3D*

rigidBrick

Create a rigid avatar from a brick3D object

$$body = my\_brick.rigidBrick(center, model, material, color = 'BLEUx')$$

where :

- center (input double array), coordinates of the center of inertia of the avatar
- model (input model object), model of the avatar
- material (input material object), material of avatar
- color (optional input string), color of the POLYR contactor
- body (returned avatar object), the avatar object