

École des Ponts ParisTech

2014-2015

Pierre Cuvilliers

Cours

Titre

Table des matières

[1. Conception des coques et toiles tendues 5](#_Toc419219939)

[1.1. Équivalence tension-compression, intérêt de la méthode 5](#_Toc419219940)

[1.2. Surfaces minimales, présentation, propriétés 5](#_Toc419219941)

[1.3. Forces extérieures 5](#_Toc419219942)

[1.4. Surfaces connues 5](#_Toc419219943)

[2. Recherche de surfaces à courbure constante 6](#_Toc419219944)

[2.1. Notations 6](#_Toc419219945)

[2.2. Hypothèses de base, justifications 6](#_Toc419219946)

[2.2.i. État de contraintes 6](#_Toc419219947)

[2.2.ii. Discrétisation, maillage 6](#_Toc419219948)

[2.3. Forces sur un élément de surface 6](#_Toc419219949)

[2.4. Équilibre d’un nœud dans le repère local 6](#_Toc419219950)

[2.5. Équilibre d’un nœud dans le repère global 6](#_Toc419219951)

[3. Résolution numérique, algorithme 7](#_Toc419219952)

[3.1. Point fixe 7](#_Toc419219953)

[3.2. Gradient 7](#_Toc419219954)

[3.3. Convergence 7](#_Toc419219955)

[3.3.i. Justification 7](#_Toc419219956)

[3.3.ii. Critère d’arrêt 7](#_Toc419219957)

[3.4. Pseudo-inverse 7](#_Toc419219958)

[3.5. Remise à jour qs 7](#_Toc419219959)

[3.6. Implémentation 7](#_Toc419219960)

[3.6.i. Boucles 7](#_Toc419219961)

[3.6.ii. Erreur numérique 8](#_Toc419219962)

[4. Forces extérieures 9](#_Toc419219963)

[4.1. Câbles 9](#_Toc419219964)

[4.2. Pression : idem 9](#_Toc419219965)

[4.3. Gravité : idem 9](#_Toc419219966)

[5. Grasshopper 10](#_Toc419219967)

[1.1. Images, utilisation générale 10](#_Toc419219968)

[5.1. Entrées 10](#_Toc419219969)

[5.2. Sorties 10](#_Toc419219970)

[5.3. Visualisations 11](#_Toc419219971)

[5.4. Exemples 11](#_Toc419219972)

[5.5. Verifications 11](#_Toc419219973)

# Conception des coques et toiles tendues

## Équivalence tension-compression, intérêt de la méthode

## Surfaces minimales, présentation, propriétés

* État de contraintes uniforme
* Courbure moyenne
* Problème de bord type « bulle de savon »

## Forces extérieures

* Pression + Gravité
* Câbles

## Surfaces connues

* PH
* Caténoïde
* Schwartz
* Plat avec câbles
* Coussin gonflé
* Isler, Kapoor, ILEK, Mantra, Tanz-Brunnen

# Recherche de surfaces à courbure constante

## Notations

## Hypothèses de base, justifications

### État de contraintes

* Tension pure + isotrope
* Densité de contrainte

### Discrétisation, maillage

* FEM masquée
* Tension uniforme vs. Taille différente des mailles
* Différentes formes possibles via qs(i)

## Forces sur un élément de surface

## Équilibre d’un nœud dans le repère local

## Équilibre d’un nœud dans le repère global

# Résolution numérique, algorithme

## Point fixe

## Gradient

* Inversion de matrice
* Méthode de Newton ?
* Instable ? Vitesse d’approche

## Convergence

### Justification

* Lemme de Banach
* Dérivée partielle vs dérivée totale (article)

### Critère d’arrêt

* Résidu forces vs déplacement
* Adimensionnement

## Pseudo-inverse

## Remise à jour qs

* Équilibre des tensions

## Implémentation

### Boucles

* Nœuds
* Itérations positions
* Itérations qs

### Erreur numérique

# Forces extérieures

## Câbles

* Intérêt : poutres de rive, fixation toiles
* Force, équilibre repère local, équilibre repère global
* Algo point fixe

## Pression : idem

## Gravité : idem

# User manual

## Features

Kroto is a membrane form-finding tool for Rhino 5 / Grasshopper. It is based on the surface stress density method, open-source[[1]](#footnote-1), and written in Python. It is specifically aimed at finding the form of a membrane in equilibrium under a set of predefined loads, such as:

* isotropic membrane stresses, uniform (minimal surfaces) or not;
* uniform pressure;
* uniform vertical load;
* edges constraints (cables, fixed points).

However, it is not a structural analysis software and so will not output physically meaningful force and stress values nor will it find the behavior of the form-found membrane under a new set of loads (“live loads”) disrupting its equilibrium. Only the position of the points on the membrane in its equilibrium state have a meaningful value; while forces and stresses are directly proportional to the chosen stress-density coefficient, this coefficient does not have a direct physical interpretation.

Kroto does not take into account second-order effects such as:

* bending stiffness;
* material non-linearities;
* non-isotropic membrane stress states, i.e. no shear stress.

Kroto only accepts valid triangular Rhino meshes as input.

Kroto was written by Pierre Cuvilliers with Lionel du Peloux and Cyril Douthe at the Laboratoire Navier, École des Ponts ParisTech. It is part of the THINkSHELL project[[2]](#footnote-2).

## Installation

To use Kroto, you will need:

* Rhino 5: <http://www.rhino3d.com/download>. At the time of writing, Rhino 5 for Mac does not support Grasshopper, so Windows only (or run it from Rhino, at your own risk[[3]](#footnote-3))! Rhino 5 SR9 recommended.
* Grashopper3: <http://www.grasshopper3d.com/page/download-1>. GH 0.9.0075 recommended.
* GH Python: <http://www.food4rhino.com/project/ghpython?ufh>. GH Python 0.6.0.3 recommended.
* Kroto:

To ease the workflow, it is recommended you also install the following Grasshopper plugins:

* TT Toolbox: <http://www.food4rhino.com/project/tttoolbox?ufh>. Used for the legend display.
* Weaverbird: <http://www.giuliopiacentino.com/weaverbird/>. Helps creating meshes in Grasshopper.

## First run

Kroto consists of five Grasshopper user objects (see Figure 5.1) and three supporting library files. The Grasshopper user objects are:

* Solver: solves the problem defined earlier, using the provided options. Calls the meshminimize module.
* Problem: defines a Kroto problem to solve, based on a triangular mesh, external forces, and edge conditions.
* Edges: defines the edge conditions for a mesh, using cables and fixed points.
* Options: Provides the default options, and allows to change them.
* Mesh closest points: small helper components that finds the vertices of a mesh that are closest to a set of points.

And the Python module files:

* Meshminimize.py: the solver itself.
* Meshminimizehelper.py: defines some helper functions, mainly for initialization.
* Vectorworks.py: defines vector and matrix calculus functions, in 3D and arbitrary dimension.
* In the sources, you will also find main.py, which can be used to run Kroto directly from Rhino, although this is undocumented.

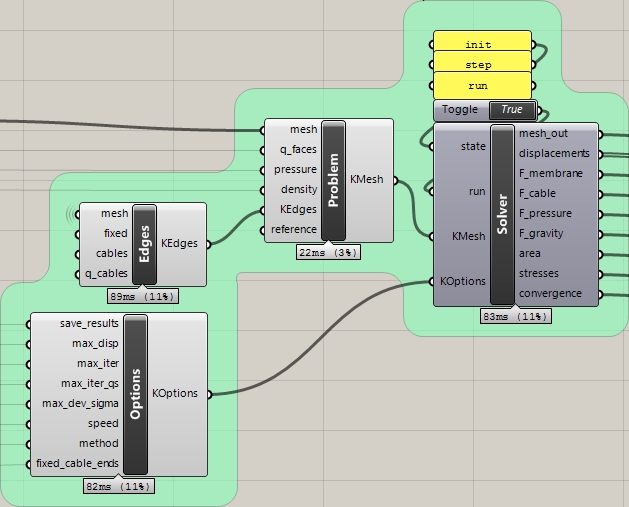


Figure 5.1: The four main Kroto components.

One of the simplest working definitions you can use to run Kroto is an inflated flat square mesh. Using only default options and no referenced Rhino geometry; it is defined by Figure 5.2. First, we create a flat meshed square with the “mesh plane” component. Then we triangularized this mesh as Kroto only accepts

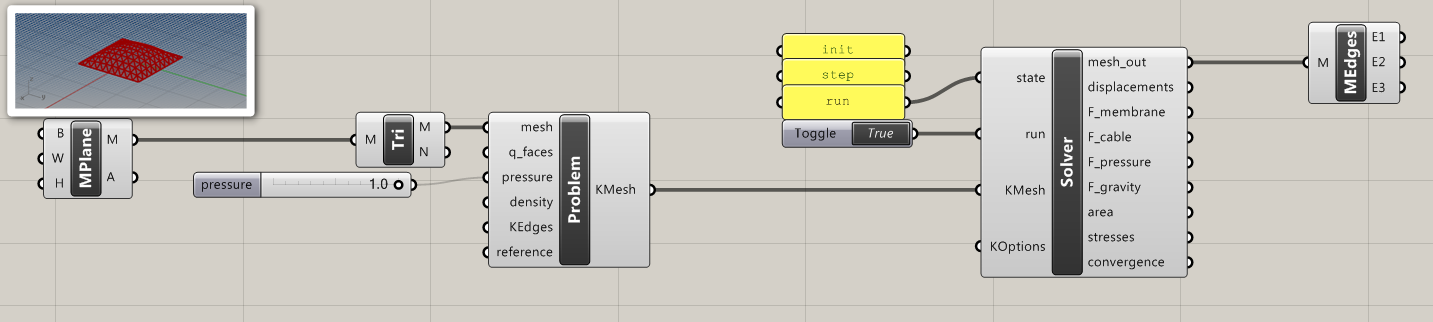


Figure 5.2: Grasshopper definition for a simple inflated square.

* PH
* Itérations
* Vitesse
* Type de maillage admissible

## Entrées

* Mesh
* Points Fixes
* Câbles
* Q\_faces
* Q\_cables
* Fixed\_cables\_ends
* Save\_results
* Max\_disp
* Max\_iter
* Max\_iter\_qs
* Max\_dev\_sigma
* Speed
* Method
* Pressure
* Density
* Reference
* State
* Run

## Sorties

* Mesh\_out
* Displacements
* F\_membrane
* F\_cable
* F\_pressure
* F\_gravity
* Stresses
* Area
* Convergence

## Visualisations

* Compare
* Colorize mesh
* Displacements

## Exemples

* Caténoide
* Schwartz
* Scherk
* Coussin
* ILEK
* Chapeau chinois
* Arcora

## Verifications

* Schwartz à la main (4 points)
* Exemples à solution analytique (Caténoïde, Schwartz, Scherk)
* Convergence en densité de maillage

1. Sources available on Github: https://github.com/THINkSHELL/Kroto [↑](#footnote-ref-1)
2. http://thinkshell.fr/ [↑](#footnote-ref-2)
3. You can run Kroto directly from Rhino, although it is not well documented and supported. See /src/main.py for pointers. [↑](#footnote-ref-3)