FORMERLY-Math

Constrained Form-Finding through Membrane Equilibrium Analysis in Mathematica

https://github.com/colivieri89/FORMERLY-Math

C. Olivieri

Department of Civil Engineering (DICIV), University of Salerno, Via Giovanni Paolo II, 132, 84084 Fisciano, SA, Italy

colivieri@unisa.it

What is FORMERLY-Math?

FORMERLY-Math: Constrained Form-Finding through Membrane Equilibrium Analysis in Mathematica

It is a package designed for the form-finding of compressed or tensile membranes in Mathematica.

Mission:

- static assessment method for masonry vaults within the framework of Limit Analysis;
- searching for efficient new forms in structural design.

FORMERLY-Math Modules

- **Pre-processing**
- **Reference Shape Definition**
- **Constrained Form Finding**
- **Stresses Evaluation and Outputs**

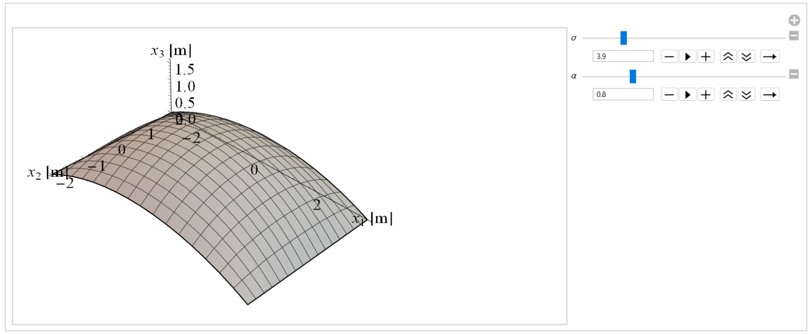
Pre-processing

Data Input

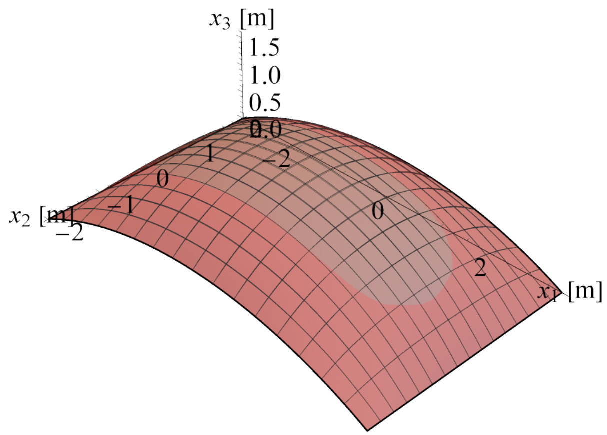
Airy Stress Function

Boundary conditions

Reference Shape Definition

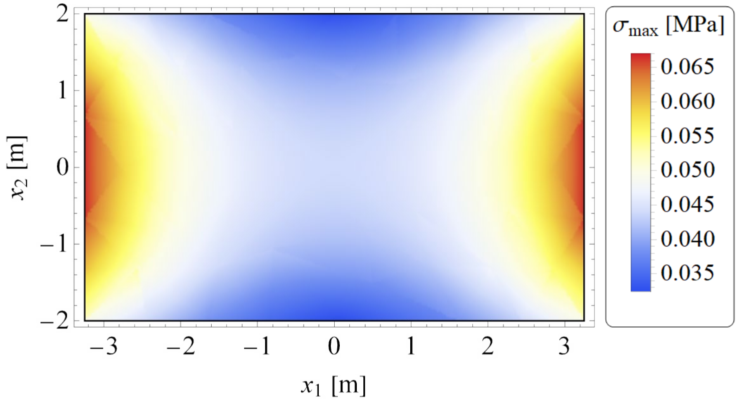


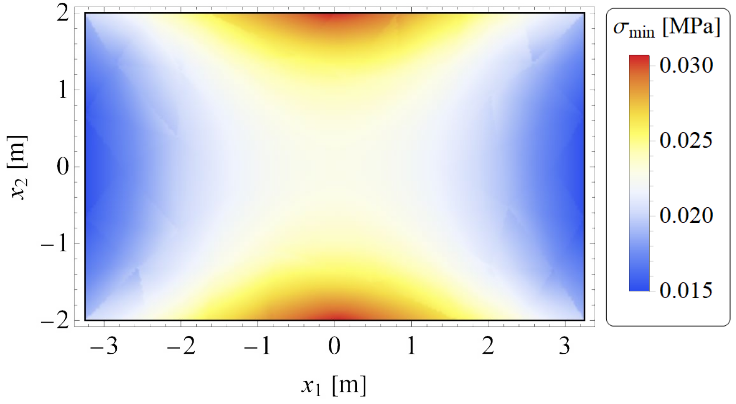
Constrained Form Finding



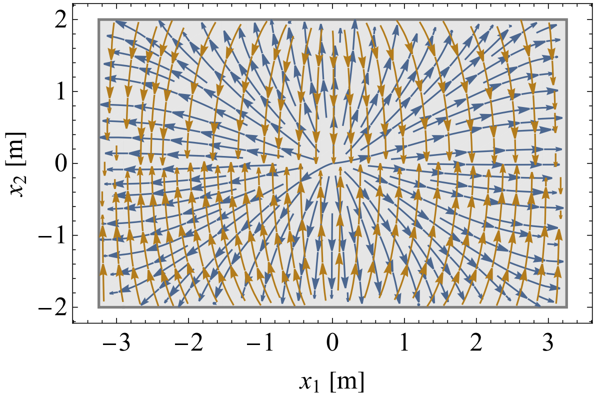
Stresses Evaluation and Outputs

Eigenvalues





Eigenvectors



Software

FORMERLY-Math is compatible with the proprietary software Wolfram Mathematica© (run successfully on versions 11.2, 12.1, and 13.0)

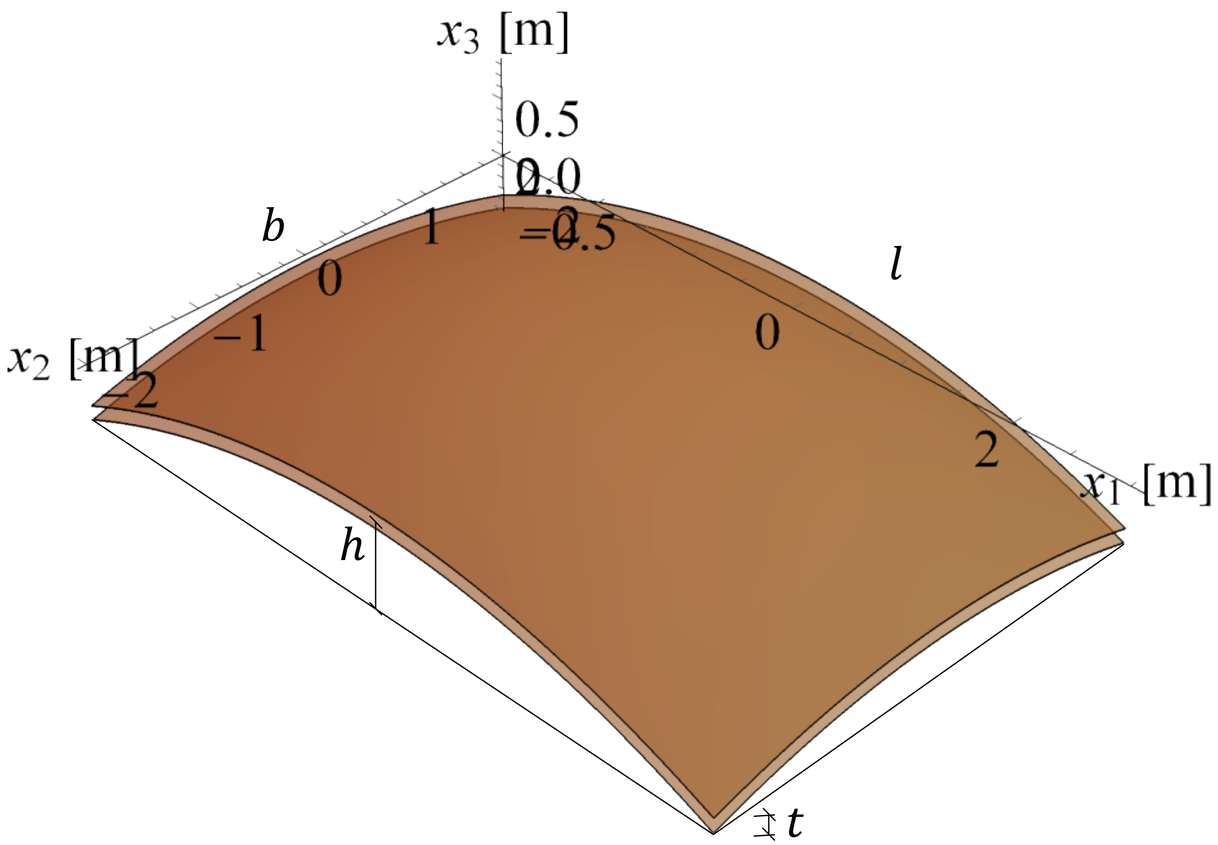
Pre-Processing

Initializing

Initializing »

Load the required packages (`NDSolve 'FEM'` and `Notation`) by running the sub-module.

Data



In this sub-module is possible to assign the geometrical parameters b, l, h, and t, as well as the shell material density ρ_a, and the mesh size `size`.

These parameters will define the initial sell weight w, the shell domain Ω, and the boundary conditions Γ_{S^0}, Γ_{S^1}, and Γ_{S^2}.

Domain and Airy Stress Function Definition

Airy Stress Function

```
𝒜[2][σ_, α_] := σ (1 / 8 (1 (b^2 - 4 x2^2) + α (l^2 - 4 x1^2))) (*Quadratic Airy Stress Function*)
𝒜[4][σ_, α_] := σ / 2 ((b^2 - x2^2) ^2 + α (l^2 - x1^2) ^2) (*Quartic Airy Stress Function*)
```

To select an Airy stress function for the equation, please use one of the following notations: $i = 2$ for the quadratic function, or $i = 4$ for the quartic function. Note that different Airy stress functions can be used to analyze different solution fields, but it's important to ensure that the function you choose is either completely concave or convex.

```
i = 2;
```

```
A[σ_, α_] := {{x1, x2, 𝒜[i][σ, α]}, {x1, x2} ∈ Ω}
```

Conditions $\Gamma_{\mathcal{S}i}$

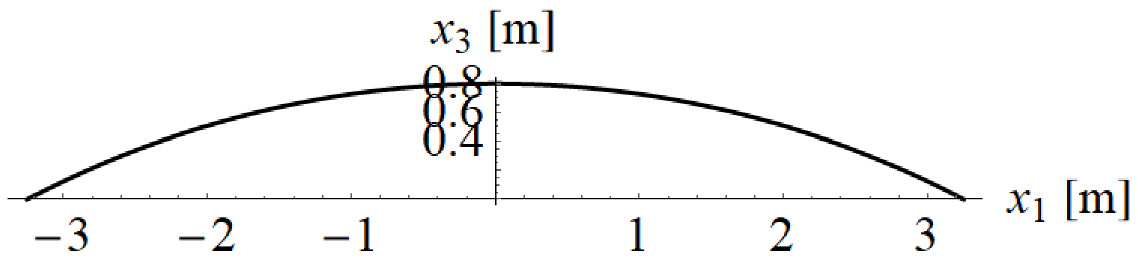
In this sub-module three Dirichlet boundary conditions are defined:

Domain boundary ($\Gamma_{\mathcal{S}0}$)

```
Γ𝒮0 = 0;
```

Arches in x_1 direction ($\Gamma_{\mathcal{S}1}$)

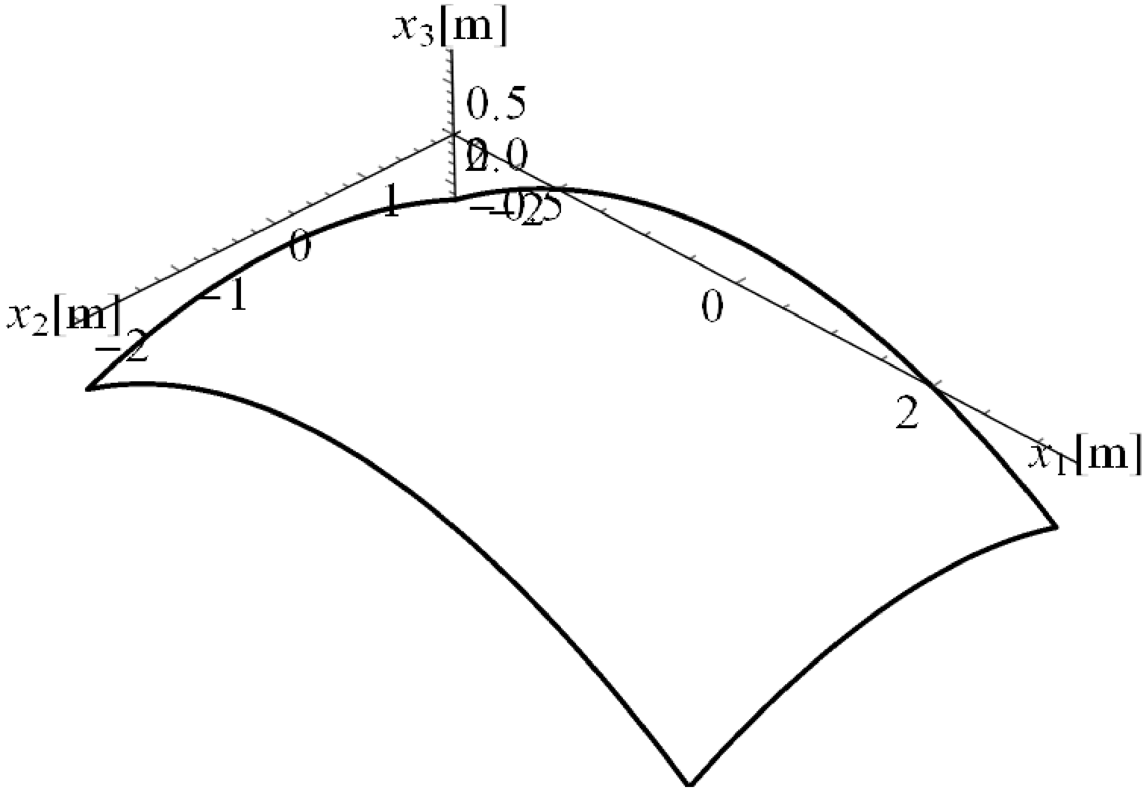
```
Γ𝒮1 = x3 /. solI[[2]];
```



Conditions Γ_{S^i}

Arches in x_1 and x_2 direction (Γ_{S^2})

```
Γ_S2 = h + (-(2 h / b^2) (x1^2 + x2^2));
```



Reference Shape Definition

Transversal Equilibrium Equation Solution

The second-order partial differential equation $F_{,22}\, f_{,11} + F_{,11}\, f_{,22} - 2\, F_{,12} f_{,12} - w = 0$ is set up in this module and solved parametrically with the chose Dirichlet boundary conditions `bcs` through a FEM routine.

Equilibrium solution

```
h𝒮1 = f𝒮1 /. ParametricNDSolve[
    {EqDiff[σ, α] == 0, bcs[i]}, f𝒮1, {x1, x2} ∈ Ω, {σ, α},
    Method → {"FiniteElement",
      "MeshOptions" → {"MeshOrder" → 2, MaxCellMeasure → size},
      "IntegrationOrder" → 4,
      "InterpolationOrder" → {f𝒮1 → 2}}];
```

Transversal Equilibrium Equation Solution

The value of `i` should be selected based on the desired boundary condition for solving the transverse equilibrium equation. For example, setting `i`=1 implies the use of boundary condition bcs[1], which corresponds to the boundary condition $\Gamma_{\mathcal{S}^1}$.

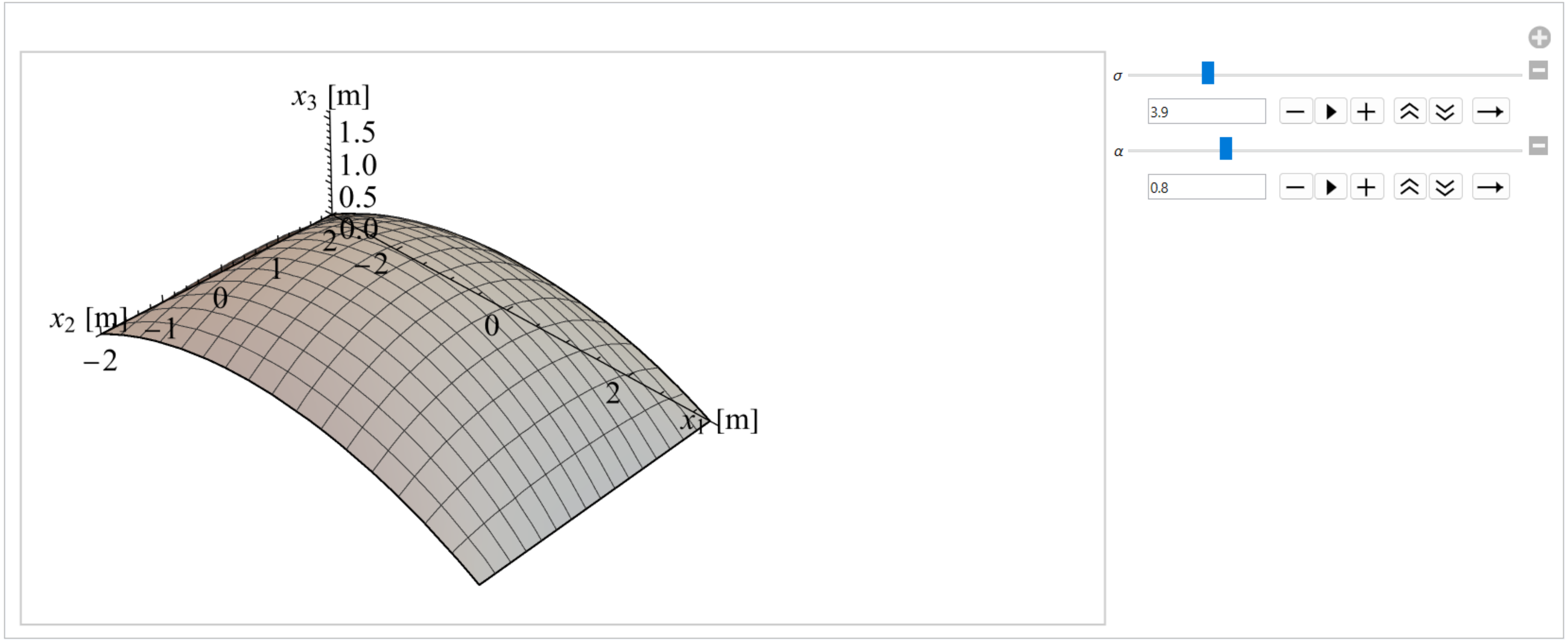
```
i = 1;
```

Equilibrium solution

```
hS1 = fS1 /. ParametricNDSolve[
    {EqDiff[σ, α] == 0, bcs[i]}, fS1, {x1, x2} ∈ Ω, {σ, α},
    Method → {"FiniteElement",
      "MeshOptions" → {"MeshOrder" → 2, MaxCellMeasure → size},
      "IntegrationOrder" → 4,
      "InterpolationOrder" → {fS1 → 2}}];
```

Reference surface selection

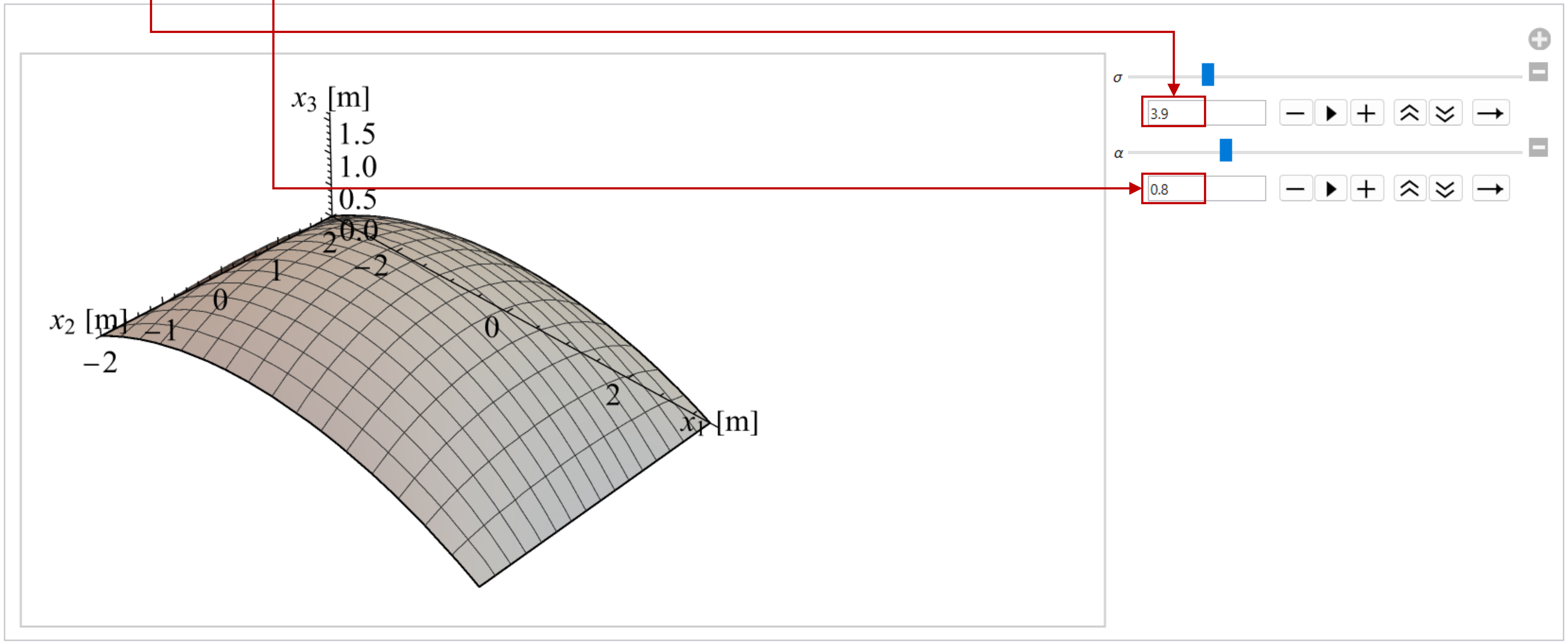
Through the `Manipulate` function, it is then possible to see how as the value of the parameters changes, the shape of the membrane changes, thus choosing the one that best reflects the user's architectural needs. The value of the parameters chosen will then go on to define what is the reference surface for constrained form-finding



Reference surface selection

Assign the values of σ and α that visualize on the right panel of the previous output once the shape that meets the user's architectural requirements are chosen.

```
RuleSol = {σ→3.9,α→0.8};
```

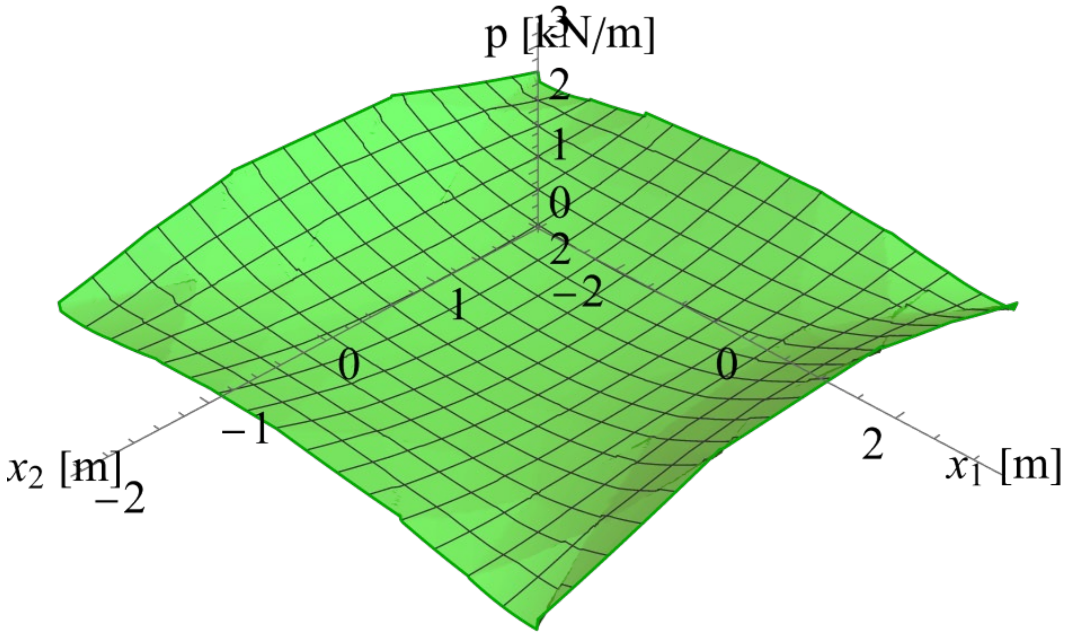


Constrained Form-Finding

The module calculates the load associated with the reference surface and incorporates it into the parametric solution of the second-order partial differential equation using the same FEM routine as demonstrated previously. The primary objective of this module is to minimize the mean distance between the shell and the reference surface by selecting appropriate parameters through the minimization of an objective function.

Real Load Evaluation

With this sub-module, you can easily calculate and visualize the load p on the designated reference surface. The sub-module achieves this by computing the prime derivatives of the shape, which are then used to establish the natural bases ($a_1 = \hat{e}_1\, f_{,1}$ and $a_1 = \hat{e}_2\, f_{,2}$). Subsequently, the Jacobian is defined as $J = |a_1 \times a_2|$. By leveraging the Jacobian, the load p can be estimated as the product of the uniformely distributed load w and the Jacobian J.



Solution and parameter minimization

The second-order partial differential equation $F_{,22}\ f_{,11} + F_{,11}\ f_{,22} - 2\ F_{,12} f_{,12} - p = 0$ is set up in this module and solved parametrically with the chose Dirichlet boundary conditions `dbs` through a FEM routine as previously shown. The same `bcs` assigned above is used to solve the pde.

Equilibrium solution

```
h𝒮2 = f𝒮2 /. ParametricNDSolve[
    {EqDiff2[σ, α] == 0, bcs2[i]}, f𝒮2, {x1, x2} ∈ Ω, {σ, α},
    Method → {"FiniteElement",
       "MeshOptions" → {"MeshOrder" → 2, MaxCellMeasure → size},
       "IntegrationOrder" → 4,
       "InterpolationOrder" → {f𝒮2 → 2}}];
```

Solution and parameter minimization

To define the mean square deviation between the reference shape h and the optimized shape f_m, we use the objective function, `ObjFun`, which depends on two numerical variables: σ and α.

```
ObjFun[σ_?NumericQ, α_?NumericQ] := Module[{h, ω},
    h = h_𝒮2[σ, α][x1, x2];
    ω = Ω;
    NIntegrate[((h / fr) - 1)^2, {x1, x2} ∈ ω]
   ] // Simplify // Chop // N
```

Solution and parameter minimization

Finally, the function `ObjFun` is minimized over the specified range of values for σ and α, with a working precision of two decimal places. It is suggested to assign solution range values of σ and α close to those assigned in `RuleSol`. The result of tThe robustness and efficiency of the FEM approach can be

```
sol = Assuming[α > 0, NMinimize[
      ObjFun[σ, α],
      {{σ, 5.2, 5.3}, {α, 0.5, 0.6}}, WorkingPrecision → 2]] // Quiet;
```

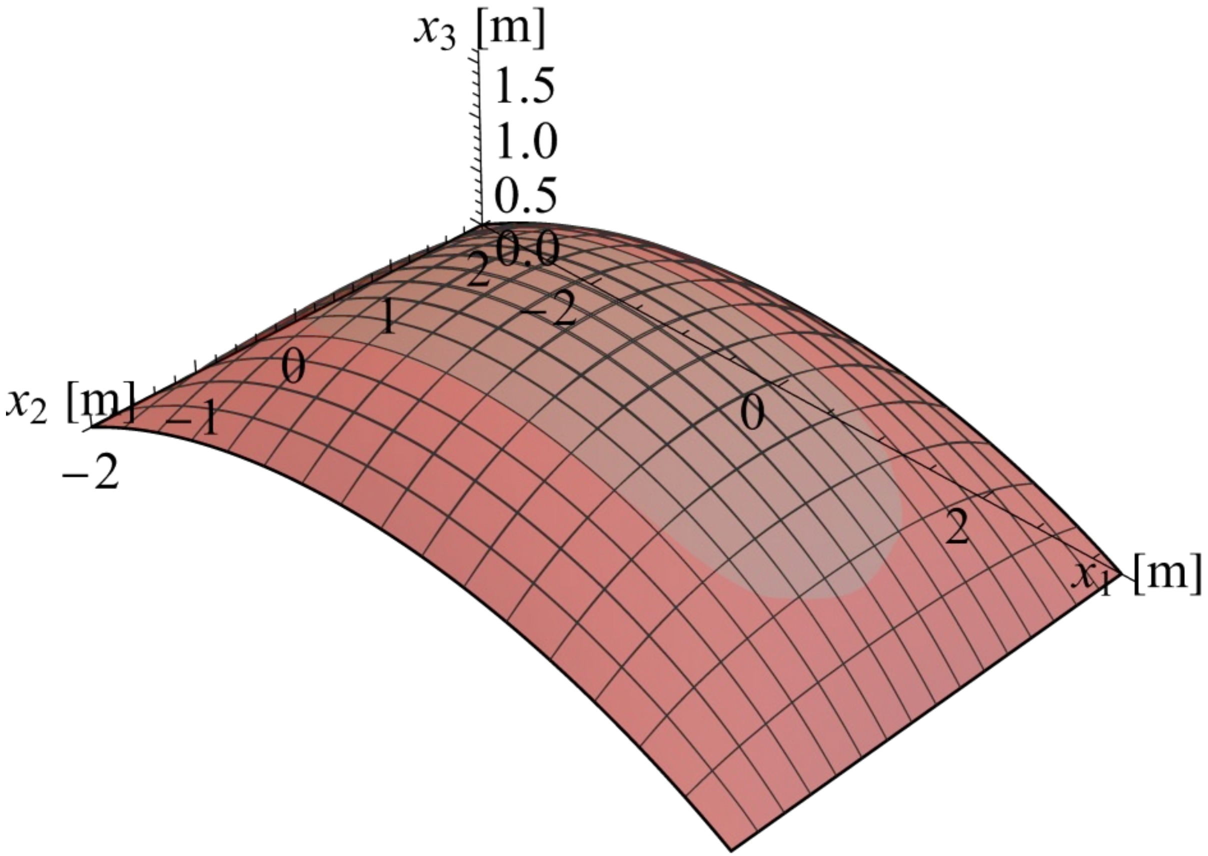
The robustness and efficiency of the FEM approach can be measured by the error norm, which is the average distance between the approximate and exact solutions. If the error norm is below a predetermined tolerance, such as 1%, it indicates a highly accurate, robust, and efficient solution. This value can be read from the output of the function `DistPerc`.

```
DistPerc = sol[[1]] 100 "%"
```

```
0.17 %
```

Solution and parameter minimization

Is then possible to plot the form-finded final surface f_m (red) compared with the reference one (grey).



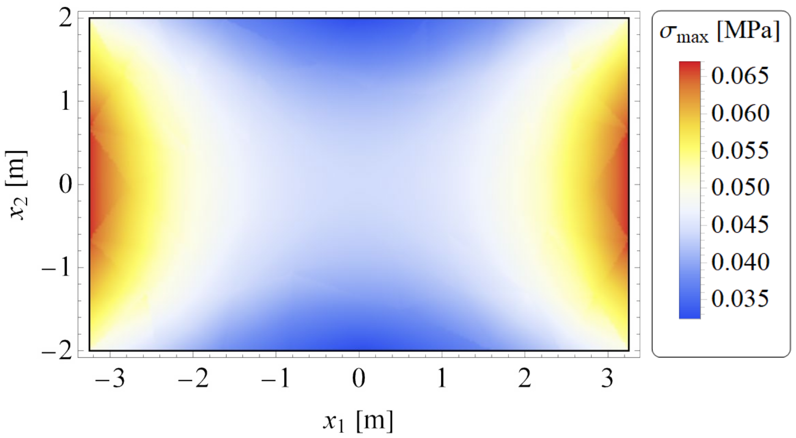
Stresses evaluation and outputs

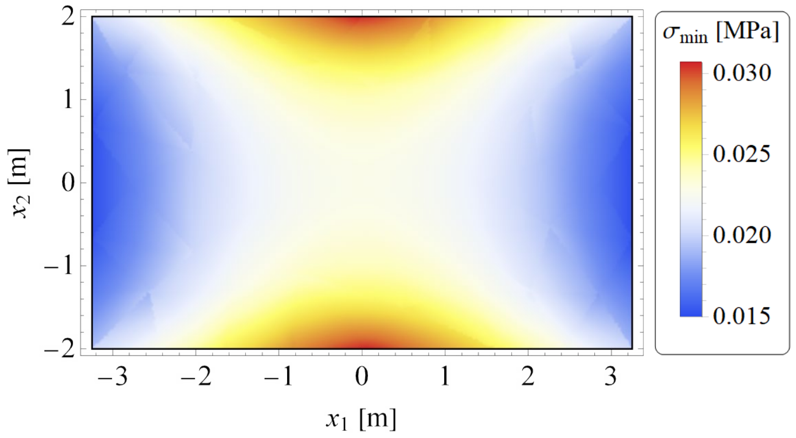
The last module of this code allows us to calculate the tensor of the stresses acting on the membrane and to evaluate and plot its eigenvalues and eigenvectors. This part provides fundamental data for the final design of the structure.

Stress tensor, eigenvalues, and eigenvectors

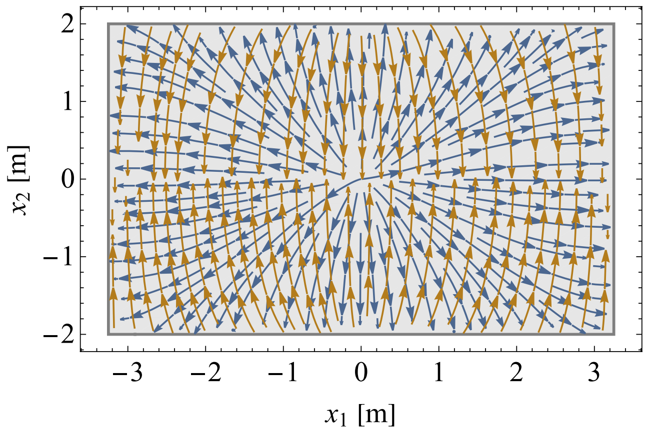
After obtaining the final surface, it is possible to evaluate its stress tensor and generate `DensityPlot` and `StreamPlot` graphs of the surface's eigenvalues and eigenvectors, respectively.

Eigenvalues





Eigenvectors

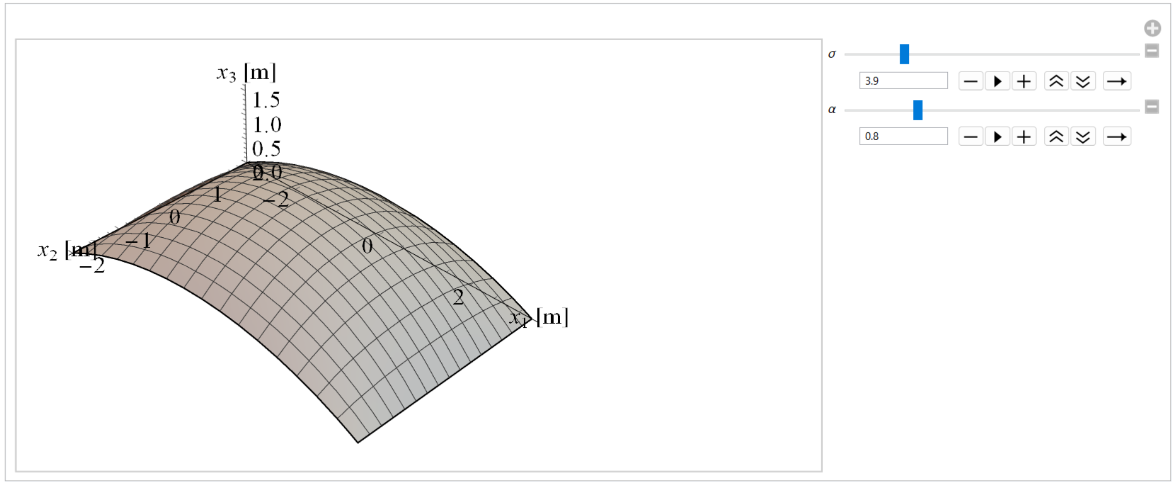


Examples

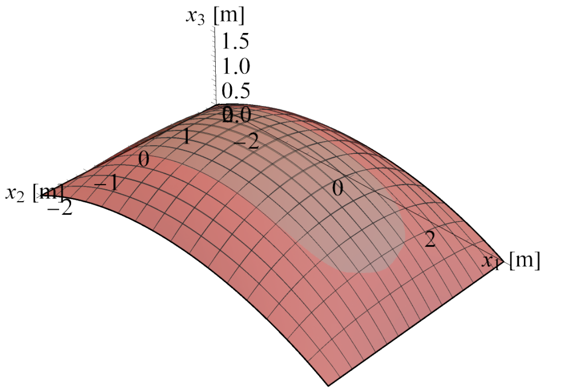
Example 1

Data: $b = 4$ m, $l = 6.5$ m, $h = 0.8$ m, $t = 0.12$ m, $\rho_a = 18\ \mathrm{kN/m^3}$, `size`$= 0.5$, $\mathcal{A}[2]$, Γ_{S^1}, $\sigma = 3.9$, $\alpha = 0.8$

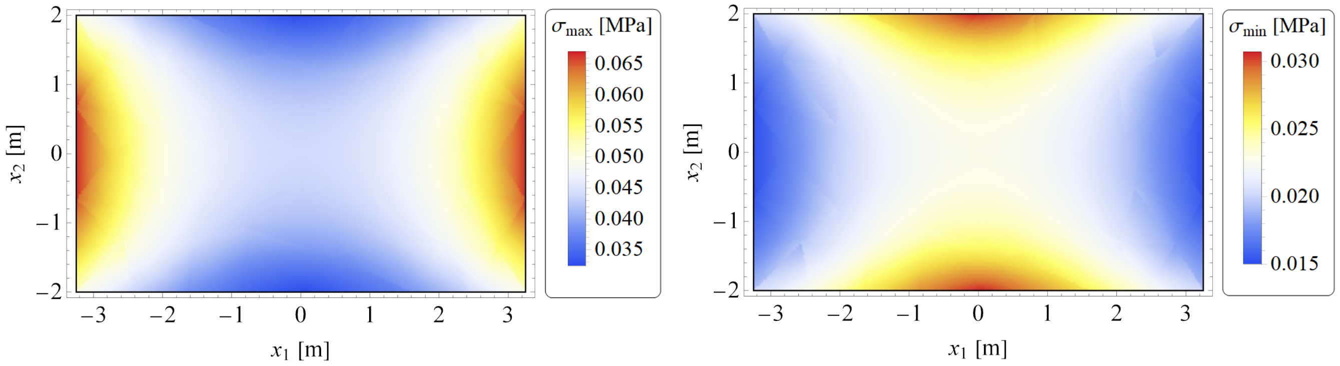
Reference surface



Final surface vs reference surface



Eigenvalues



Eigenvectors