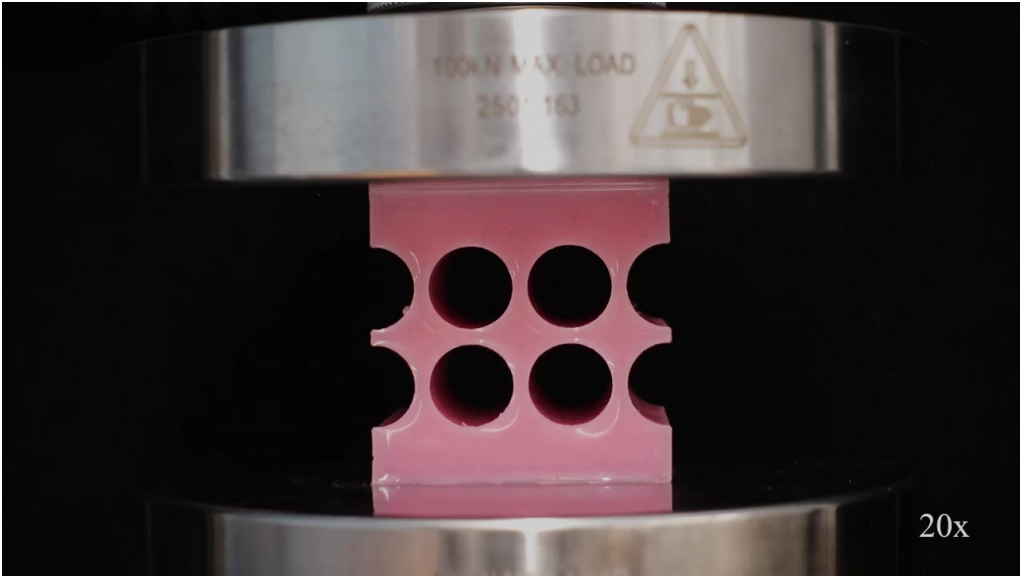
Controlling the energy jump of multistable structures using shape optimisation

Patrick E. Farrell[1,2] Arselane Hadj-Slimane[3] Àlex Ferrer[4,5] Alberto Paganini[6]

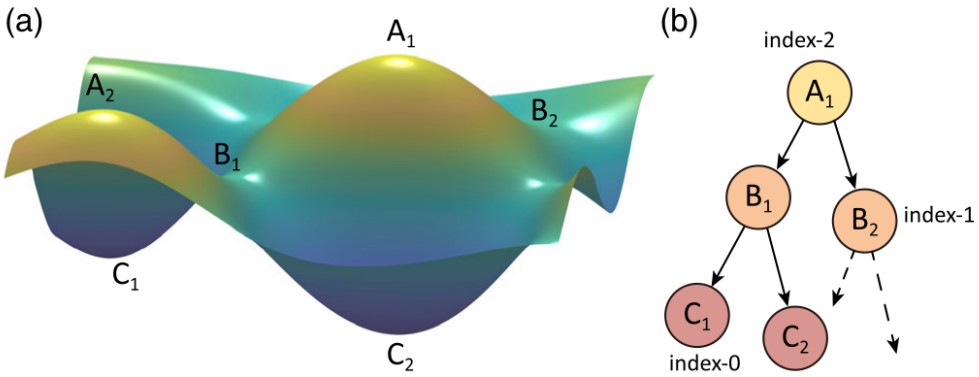



[1]University of Oxford [2]Charles University [3]ÉNS Paris-Saclay [4]Universitat Politècnica de Catatlunya
[5]CIMNE [6]University of Leicester



From Medina, Farrell, Bertoldi, Rycroft (2000).

Many PDEs have multiple solutions for the same data.



Multiple solutions for the stationary points of an energy. Credit: Yin et al. (2020)

Question

How can we control the energy jump required to switch between two solutions?

Question

How can we control the energy jump required to switch between two solutions?

In a hard drive 1s and 0s are represented by bistable/multistable physical systems (floating-gate MOSFETs in SSDs).

Question

How can we control the energy jump required to switch between two solutions?

In a hard drive 1s and 0s are represented by bistable/multistable physical systems (floating-gate MOSFETs in SSDs).

In general, there is a 'right' energy jump: too little and the system will switch when not desired, too much and it is too expensive (e.g. on battery).

Question

How can we control the energy jump required to switch between two solutions?

In a hard drive 1s and 0s are represented by bistable/multistable physical systems (floating-gate MOSFETs in SSDs).

In general, there is a 'right' energy jump: too little and the system will switch when not desired, too much and it is too expensive (e.g. on battery).

In the mechanical context, this is used in impact padding: energy is dissipated on impact to switch the structure from one state to another.

Main mathematical challenge

We want to control *the relationship between several different solutions of the same PDE*.

Main mathematical challenge

We want to control *the relationship between several different solutions of the same PDE*.

Consequence

The parameter-to-solution map (here $\Omega \mapsto u$) is multi-valued.

Main mathematical challenge

We want to control *the relationship between several different solutions of the same PDE*.

Consequence

The parameter-to-solution map (here $\Omega \mapsto u$) is multi-valued.

How do we formulate such problems? How do we analyse them? How do we solve them?

Section 2

Formulation

Let $\mathcal{U}_{\text{ad}}$ denote a collection of admissible shapes. We want to solve

$$\text{find } \Omega^\star \in \underset{\Omega \in \mathcal{U}_{\text{ad}}}{\text{argmin}} \; J\left(\Omega, \{u_k(\Omega)\}\right)$$

Let $\mathcal{U}_{\text{ad}}$ denote a collection of admissible shapes. We want to solve

$$\text{find } \Omega^\star \in \underset{\Omega \in \mathcal{U}_{\text{ad}}}{\text{argmin}}\ J\left(\Omega, \{u_k(\Omega)\}\right)$$

where the PDE solutions satisfy the PDE

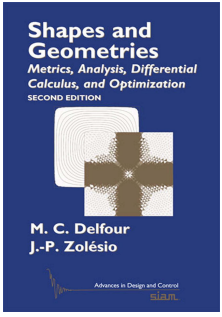
$$u_k(\Omega) \in V(\Omega) \text{ and } F(\Omega, u_k(\Omega)) = 0 \text{ for all } k = 1, \ldots, n,$$

and are all distinct on all domains:

$$u_k(\Omega) \neq u_l(\Omega) \text{ for all } \Omega \in \mathcal{U}_{\text{ad}}, k = 1, \ldots, n, l = k+1, \ldots, n.$$

For our admissible set, we make the standard choice of mapping from a reference domain

$$\mathcal{U}_{\mathsf{ad}} = \{T(\Omega_0) \mid T \in \mathcal{T}_{\mathsf{ad}}\},$$



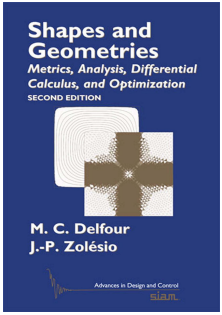
For our admissible set, we make the standard choice of mapping from a reference domain

$$\mathcal{U}_{\text{ad}} = \{T(\Omega_0) \mid T \in \mathcal{T}_{\text{ad}}\},$$

where the admissible transformations are

$$\mathcal{T}_{\text{ad}} = \left\{I + \theta \mid \theta \in \Theta, (I+\theta)^{-1} \text{ exists, and } (I+\theta)^{-1} - I \in \Theta\right\}$$

for $\Theta = C^{0,1}(\mathbb{R}^N, \mathbb{R}^N)$.



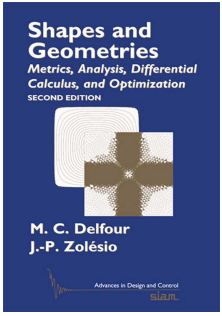
For our admissible set, we make the standard choice of mapping from a reference domain

$$\mathcal{U}_{\text{ad}} = \{T(\Omega_0) \mid T \in \mathcal{T}_{\text{ad}}\},$$

where the admissible transformations are

$$\mathcal{T}_{\text{ad}} = \left\{I + \theta \mid \theta \in \Theta, (I+\theta)^{-1} \text{ exists, and } (I+\theta)^{-1} - I \in \Theta\right\}$$

for $\Theta = C^{0,1}(\mathbb{R}^N, \mathbb{R}^N)$.



Good news

This preserves the property that the mapped domain is Lipschitz if the initial domain is.

For the PDE, we take a standard primal neo-Hookean hyperelasticity.

The function space is

$$V(\Omega) \subset H^1(\Omega; \mathbb{R}^N) \text{ with boundary conditions,}$$

and the PDE is the first-order optimality condition for an energy minimisation problem

$$\langle F(\Omega, u_k(\Omega)), v \rangle = \mathcal{E}'(\Omega, u_k; v).$$

For the PDE, we take a standard primal neo-Hookean hyperelasticity.

The function space is

$$V(\Omega) \subset H^1(\Omega; \mathbb{R}^N) \text{ with boundary conditions,}$$

and the PDE is the first-order optimality condition for an energy minimisation problem

$$\langle F(\Omega, u_k(\Omega)), v \rangle = \mathcal{E}'(\Omega, u_k; v).$$

The hyperelastic energy is

$$\mathcal{E}(\Omega, u) = \int_\Omega \psi(I + \nabla u) \, \mathrm{d}x - \int_\Omega b \cdot u \, \mathrm{d}x,$$

for a (neo-Hookean) strain energy density ψ

$$\psi(\mathbf{F}) = \mu \left(\mathrm{tr}(\mathbf{F}^\top \mathbf{F}) - N \right) - \mu \ln \det \mathbf{F} + \frac{\lambda}{2} (\ln \det \mathbf{F})^2.$$

For the functional, we define the energy jump

$$\Delta\mathcal{E}(\Omega, u_i, u_j) = |\mathcal{E}(\Omega, u_i) - \mathcal{E}(\Omega, u_j)|$$

For the functional, we define the energy jump

$$\Delta\mathcal{E}(\Omega, u_i, u_j) = |\mathcal{E}(\Omega, u_i) - \mathcal{E}(\Omega, u_j)|$$

and construct functionals like

$$J(\Omega) = \left(\Delta\mathcal{E}(\Omega, u_1, u_2) - \frac{1}{5}\Delta\mathcal{E}(\Omega_0, u_1, u_2)\right)^2 + \left(\Delta\mathcal{E}(\Omega, u_2, u_3) - \frac{1}{5}\Delta\mathcal{E}(\Omega_0, u_2, u_3)\right)^2$$

where solutions 1 and 3 are local minimisers and solution 2 is the saddle point between them.

For the functional, we define the energy jump

$$\Delta\mathcal{E}(\Omega, u_i, u_j) = |\mathcal{E}(\Omega, u_i) - \mathcal{E}(\Omega, u_j)|$$

and construct functionals like

$$J(\Omega) = \left(\Delta\mathcal{E}(\Omega, u_1, u_2) - \frac{1}{5}\Delta\mathcal{E}(\Omega_0, u_1, u_2)\right)^2 + \left(\Delta\mathcal{E}(\Omega, u_2, u_3) - \frac{1}{5}\Delta\mathcal{E}(\Omega_0, u_2, u_3)\right)^2$$

where solutions 1 and 3 are local minimisers and solution 2 is the saddle point between them.

Intuition

This functional seeks to reduce the energy jump by 80%.

For the functional, we define the energy jump

$$\Delta\mathcal{E}(\Omega, u_i, u_j) = |\mathcal{E}(\Omega, u_i) - \mathcal{E}(\Omega, u_j)|$$

and construct functionals like

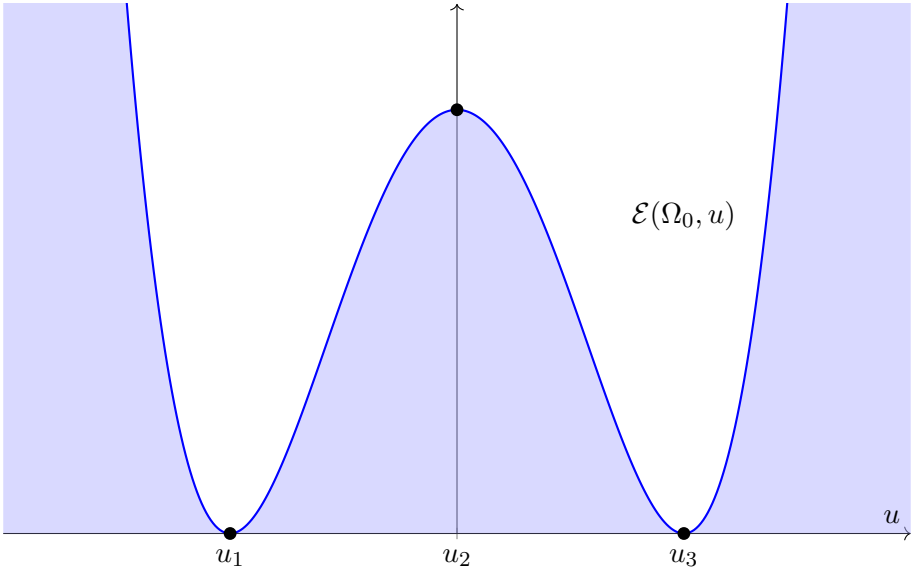
$$J(\Omega) = \left(\Delta\mathcal{E}(\Omega, u_1, u_2) - \frac{1}{5}\Delta\mathcal{E}(\Omega_0, u_1, u_2)\right)^2 + \left(\Delta\mathcal{E}(\Omega, u_2, u_3) - \frac{1}{5}\Delta\mathcal{E}(\Omega_0, u_2, u_3)\right)^2$$

where solutions 1 and 3 are local minimisers and solution 2 is the saddle point between them.

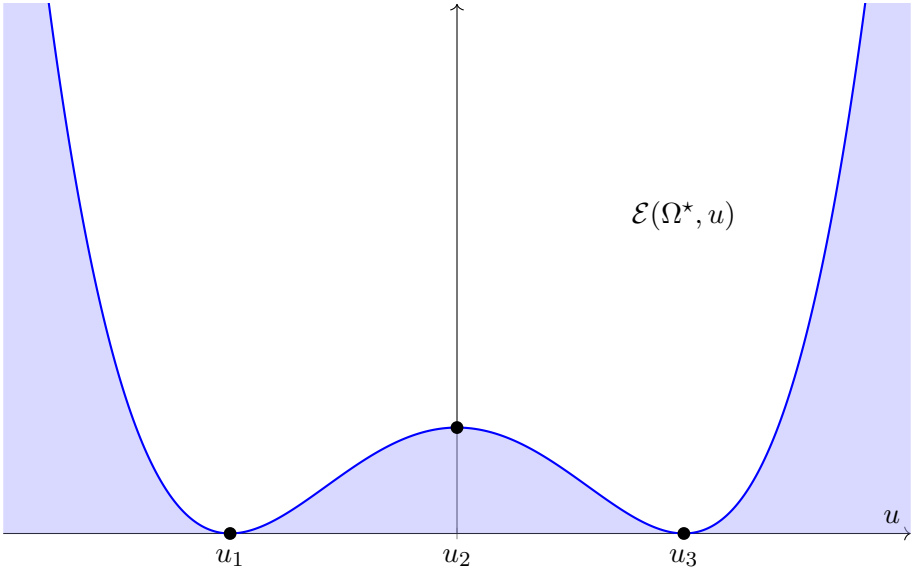
Intuition

This functional seeks to reduce the energy jump by 80%.

The number of solutions $n(\Omega)$ might vary, but we need $n(\Omega) \geq 3$.



Before optimisation.



After optimisation.

Section 3

Preliminary analysis

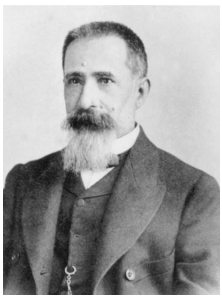
Suppose we start with an initial domain Ω_0 that supports n solutions.

We would like the n solutions we have to persist as distinct, well-defined branches $u_k(\Omega)$.

Suppose we start with an initial domain Ω_0 that supports n solutions.

We would like the n solutions we have to persist as distinct, well-defined branches $u_k(\Omega)$.

If our parameter were a vector of real numbers, the implicit function theorem would give sufficient conditions for this.



Ulisse Dini

Suppose we start with an initial domain Ω_0 that supports n solutions.

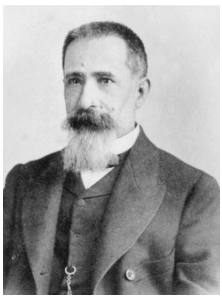
We would like the n solutions we have to persist as distinct, well-defined branches $u_k(\Omega)$.

If our parameter were a vector of real numbers, the implicit function theorem would give sufficient conditions for this.

We can apply the implicit function theorem by rewriting the PDE via pullback onto Ω_0, and by parameterising the transformation $T_t \in \mathcal{T}_{\text{ad}}$ as

$$T_t = I + t\theta$$

for $t \in \mathbb{R}$.



Ulisse Dini

Suppose we start with an initial domain Ω_0 that supports n solutions.

We would like the n solutions we have to persist as distinct, well-defined branches $u_k(\Omega)$.

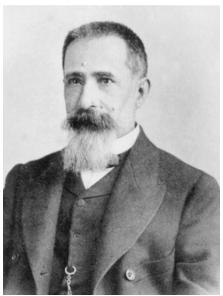
If our parameter were a vector of real numbers, the implicit function theorem would give sufficient conditions for this.

We can apply the implicit function theorem by rewriting the PDE via pullback onto Ω_0, and by parameterising the transformation $T_t \in \mathcal{T}_{\text{ad}}$ as

$$T_t = I + t\theta$$

for $t \in \mathbb{R}$.

This gives sufficient conditions that guarantee that the solutions in the statement of the problem continue to exist.



Ulisse Dini

Section 4

Computational results

We solve the problem with Netgen, Firedrake, Fireshape, and ROL.

We solve the problem with Netgen, Firedrake, Fireshape, and ROL.

The shape derivatives are calculated automatically by UFL.

We solve the problem with Netgen, Firedrake, Fireshape, and ROL.

The shape derivatives are calculated automatically by UFL.

We use a dogleg trust-region BFGS algorithm with $H^1(\Omega; \mathbb{R}^N)$ inner product.

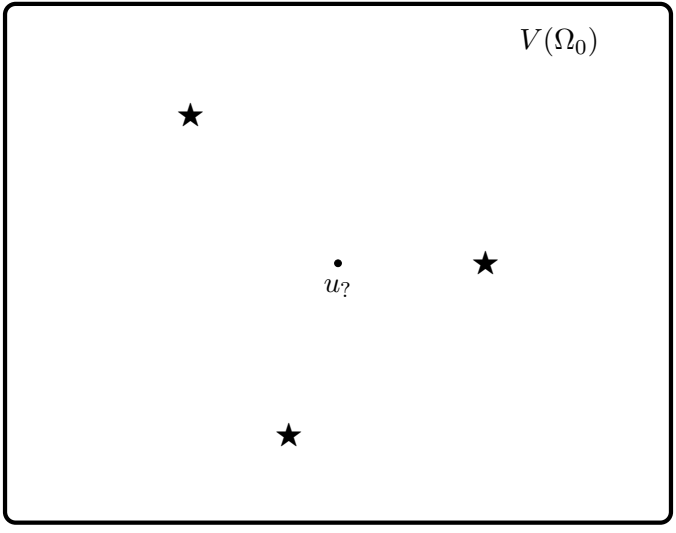
We solve the problem with Netgen, Firedrake, Fireshape, and ROL.

The shape derivatives are calculated automatically by UFL.

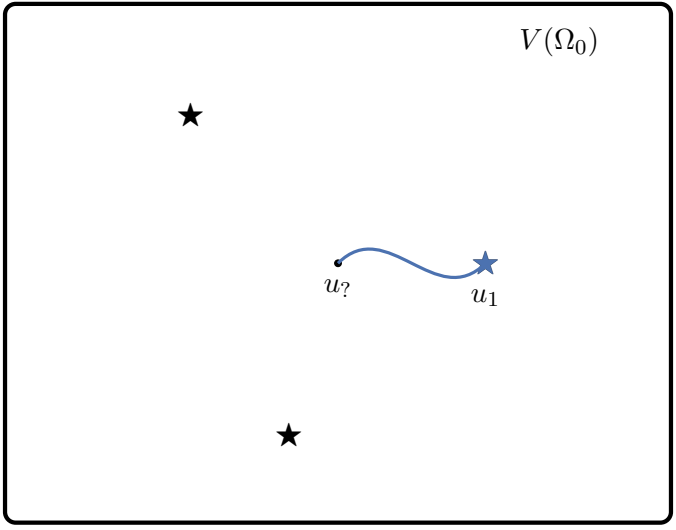
We use a dogleg trust-region BFGS algorithm with $H^1(\Omega; \mathbb{R}^N)$ inner product.

This gives mesh-independent convergence of the outer optimisation iteration.

We use deflation + continuation to compute the multiple solutions on the initial domain.

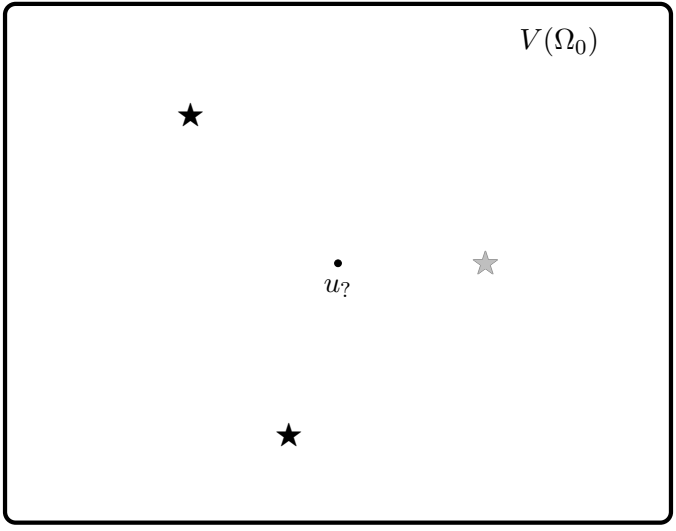


We use deflation + continuation to compute the multiple solutions on the initial domain.



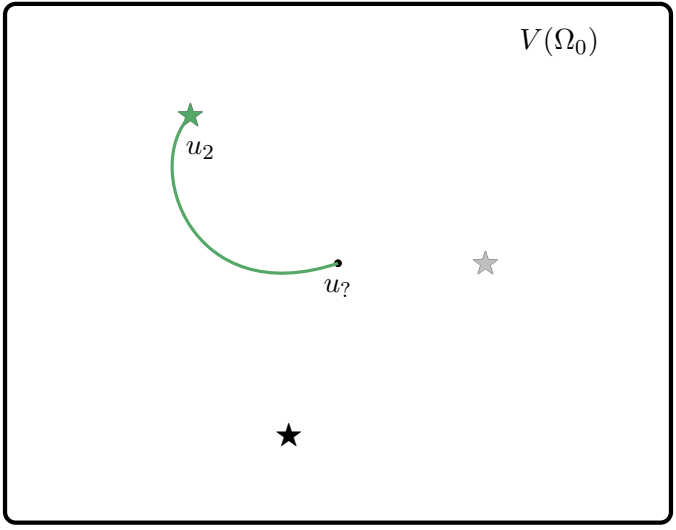
Newton from initial guess.

We use deflation + continuation to compute the multiple solutions on the initial domain.



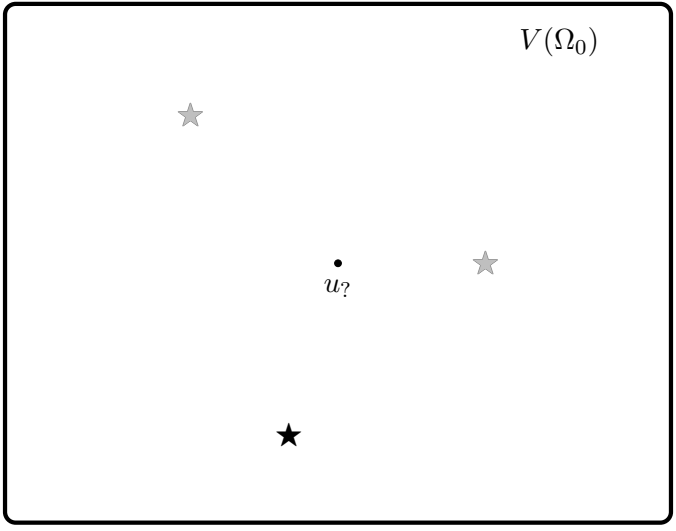
Deflate solution found.

We use deflation + continuation to compute the multiple solutions on the initial domain.



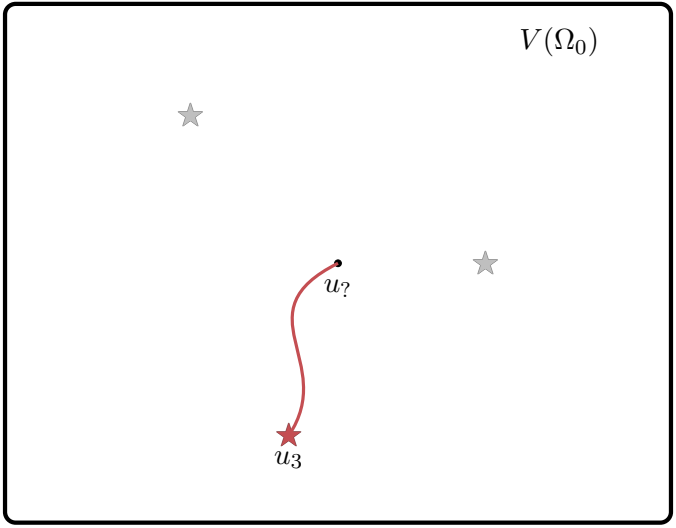
Newton from initial guess.

We use deflation + continuation to compute the multiple solutions on the initial domain.



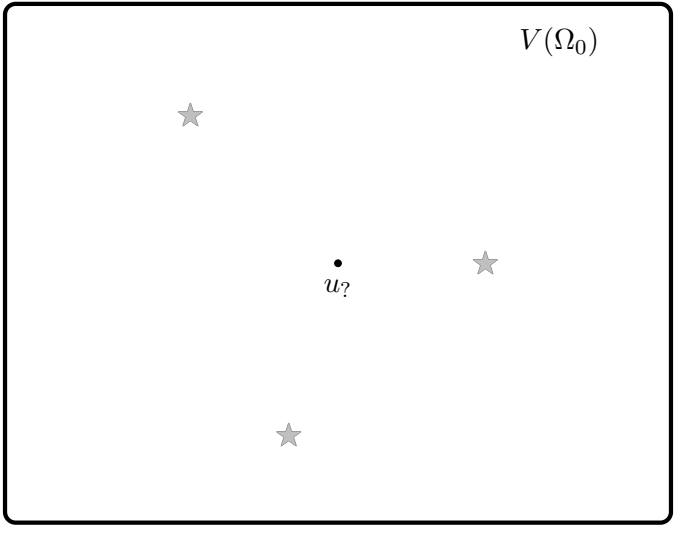
Deflate solution found.

We use deflation + continuation to compute the multiple solutions on the initial domain.



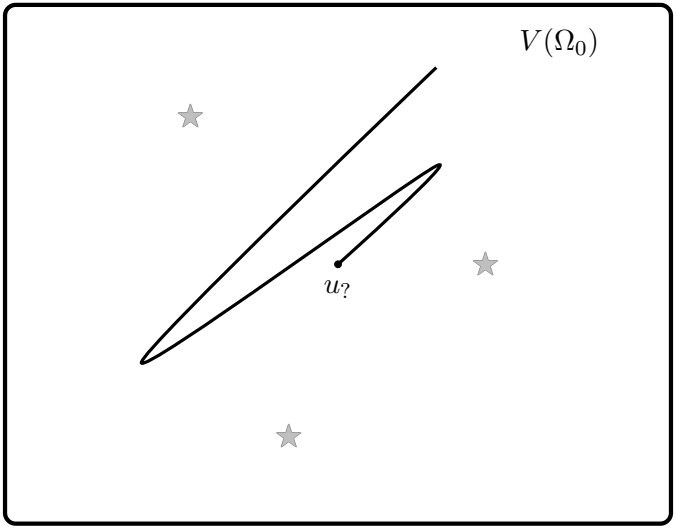
Newton from initial guess.

We use deflation + continuation to compute the multiple solutions on the initial domain.



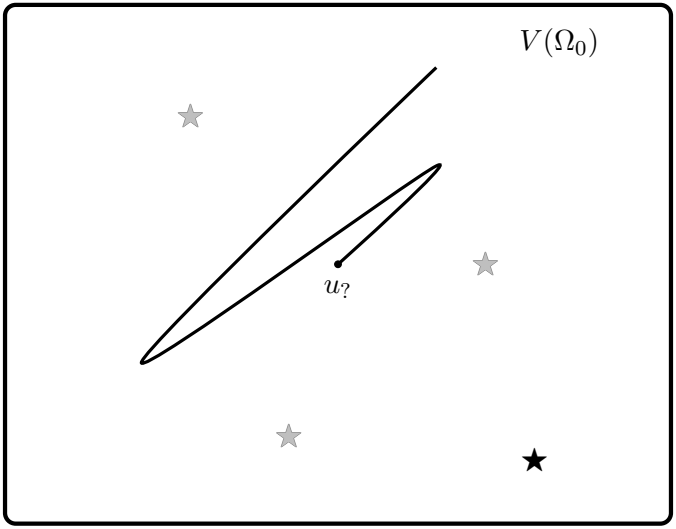
Deflate solution found.

We use deflation + continuation to compute the multiple solutions on the initial domain.



Terminate on nonconvergence.

We use deflation + continuation to compute the multiple solutions on the initial domain.



Terminate on nonconvergence.

The next major subtlety is tracking the branches separately during the optimisation. Newton's method can jump from branch to branch.

The next major subtlety is tracking the branches separately during the optimisation. Newton's method can jump from branch to branch.

Pragmatic remedies

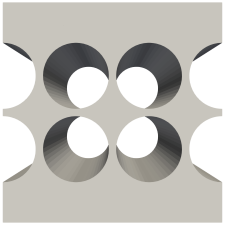
1. Limit the number of Newton iterations so that it does not go cross-country.
2. If we fail to track all branches, take a smaller optimisation step.
3. Add a regularisation to J that penalises if solutions get too close together.

The next major subtlety is tracking the branches separately during the optimisation. Newton's method can jump from branch to branch.

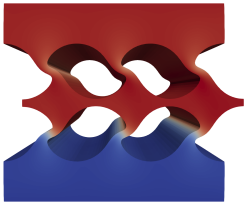
Pragmatic remedies

1. Limit the number of Newton iterations so that it does not go cross-country.
2. If we fail to track all branches, take a smaller optimisation step.
3. Add a regularisation to J that penalises if solutions get too close together.

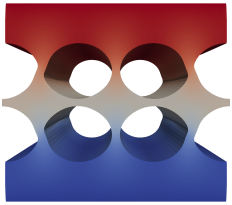
With these the optimisation algorithm works robustly in practice.



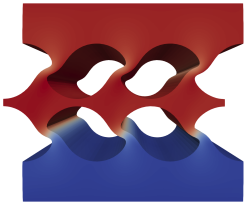
Ω_0



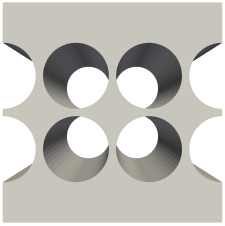
$u_1(\Omega_0)$



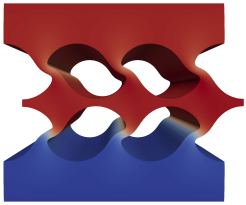
$u_2(\Omega_0)$



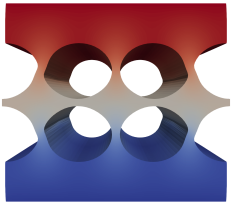
$u_3(\Omega_0)$



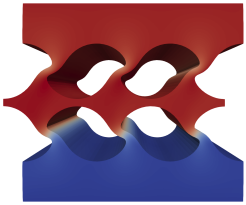
Ω_0



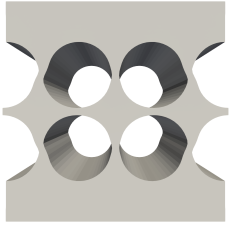
$u_1(\Omega_0)$



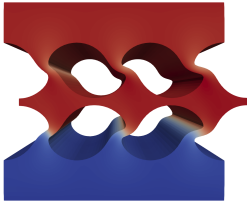
$u_2(\Omega_0)$



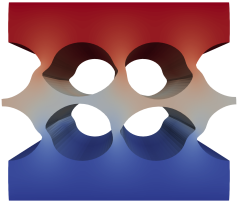
$u_3(\Omega_0)$



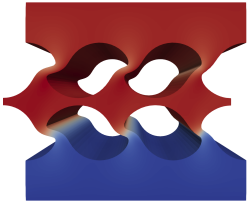
$\Omega^\star$



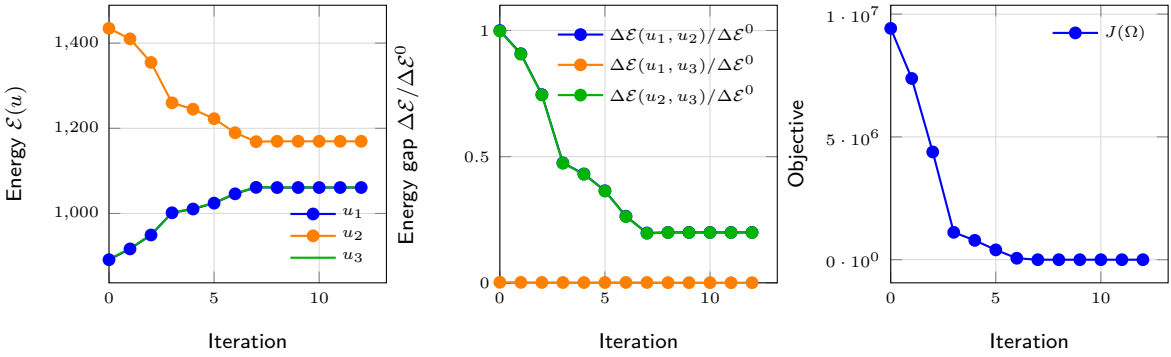
$u_1(\Omega^\star)$



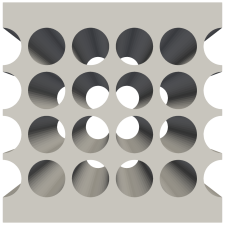
$u_2(\Omega^\star)$



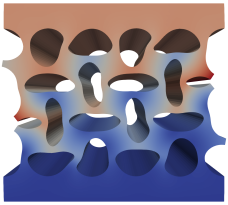
$u_3(\Omega^\star)$



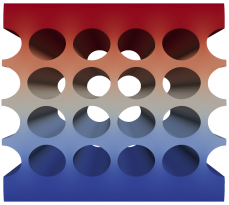
The energy gap converges to the target value in 7 iterations.



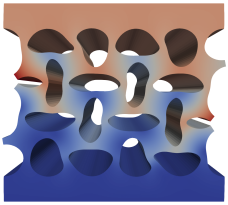
Ω_0



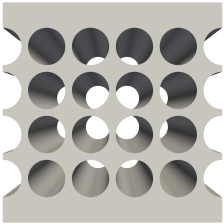
$u_1(\Omega_0)$



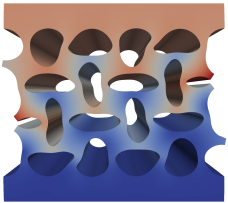
$u_2(\Omega_0)$



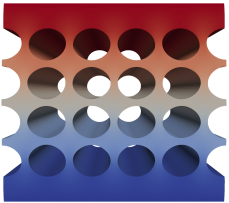
$u_3(\Omega_0)$



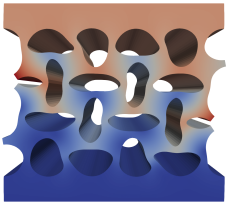
Ω_0



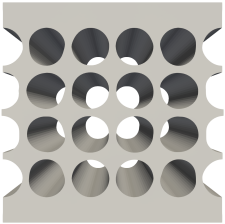
$u_1(\Omega_0)$



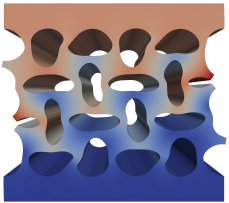
$u_2(\Omega_0)$



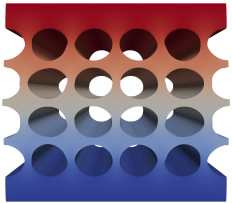
$u_3(\Omega_0)$



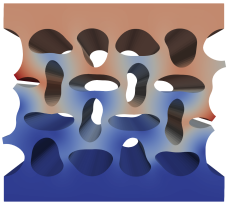
$\Omega^\star$



$u_1(\Omega^\star)$



$u_2(\Omega^\star)$



$u_3(\Omega^\star)$

Section 5

Conclusions

Good news

We can formulate and solve PDE-constrained optimisation problems with multivalued parameter-to-solution maps.

Good news

We can formulate and solve PDE-constrained optimisation problems with multivalued parameter-to-solution maps.

Many open questions remain:

1. Analysis of the problem and the algorithm.

Good news

We can formulate and solve PDE-constrained optimisation problems with multivalued parameter-to-solution maps.

Many open questions remain:

1. Analysis of the problem and the algorithm.
2. What should we do if the number of solutions varies dynamically?

Good news

We can formulate and solve PDE-constrained optimisation problems with multivalued parameter-to-solution maps.

Many open questions remain:

1. Analysis of the problem and the algorithm.
2. What should we do if the number of solutions varies dynamically?
3. Self-contact can be important here. How would this work for a QVI?