scikit-fem Documentation

Release 8.0.0

scikit-fem developers

Jan 26, 2024

CONTENTS

scikit-fem is a pure Python 3.7+ library for performing finite element assembly. Its main purpose is the transformation of bilinear forms into sparse matrices and linear forms into vectors. The library supports triangular, quadrilateral, tetrahedral and hexahedral meshes as well as one-dimensional problems.

Note: Installing the library is as simple as running

```
pip install scikit-fem[all]
```

Remove `[all]` to not install the optional dependencies `meshio` and `matplotlib`.

CHAPTER

ONE

TABLE OF CONTENTS

1.1 examples package

1.1.1 Submodules

1.1.2 examples.ex01 module

examples.ex01.**visualize**()

1.1.3 examples.ex02 module

This example demonstrates the solution of a slightly more complicated problem with multiple boundary conditions and a fourth-order differential operator. We consider the Kirchhoff plate bending problem which finds its applications in solid mechanics. For a stationary plate of constant thickness d, the governing equation reads: find the deflection $u : \Omega \to \mathbb{R}$ that satisfies

$$\frac{Ed^3}{12(1-\nu^2)}\Delta^2 u = f \quad \text{in } \Omega,$$

where $\Omega = (0,1)^2$, f is a perpendicular force, E and ν are material parameters. In this example, we analyse a $1\,\mathrm{m}^2$ plate of steel with thickness $d = 0.1\,\mathrm{m}$. The Young's modulus of steel is $E = 200 \cdot 10^9$ Pa and Poisson ratio $\nu = 0.3$.

In reality, the operator

$$\frac{Ed^3}{12(1-\nu^2)}\Delta^2$$

is a combination of multiple first-order operators:

$$\boldsymbol{K}(u) = -\boldsymbol{\varepsilon}(\nabla u), \quad \boldsymbol{\varepsilon}(\boldsymbol{w}) = \frac{1}{2}(\nabla \boldsymbol{w} + \nabla \boldsymbol{w}^T),$$

$$\boldsymbol{M}(u) = \frac{d^3}{12}\mathbb{C}\boldsymbol{K}(u), \quad \mathbb{C}\boldsymbol{T} = \frac{E}{1+\nu}\left(\boldsymbol{T} + \frac{\nu}{1-\nu}(\operatorname{tr}\boldsymbol{T})\boldsymbol{I}\right),$$

where $\boldsymbol{I}$ is the identity matrix. In particular,

$$\frac{Ed^3}{12(1-\nu^2)}\Delta^2 u = -\operatorname{div}\mathbf{div}\,\boldsymbol{M}(u).$$

There are several boundary conditions that the problem can take. The *fully clamped* boundary condition reads

$$u = \frac{\partial u}{\partial \boldsymbol{n}} = 0,$$

where $\boldsymbol{n}$ is the outward normal. Moreover, the *simply supported* boundary condition reads

$$u = 0, \quad M_{nn}(u) = 0,$$

where $M_{nn} = \boldsymbol{n} \cdot (\boldsymbol{M}\boldsymbol{n})$. Finally, the *free* boundary condition reads

$$M_{nn}(u) = 0, \quad V_n(u) = 0,$$

where V_n is the Kirchhoff shear force. The exact definition is not needed here as this boundary condition is a natural one.

The correct weak formulation for the problem is: find $u \in V$ such that

$$\int_\Omega \boldsymbol{M}(u) : \boldsymbol{K}(v) \,\mathrm{d}x = \int_\Omega f v \,\mathrm{d}x \quad \forall v \in V,$$

where V is now a subspace of H^2 with the essential boundary conditions for u and $\frac{\partial u}{\partial \boldsymbol{n}}$.

Instead of constructing a subspace for H^2, we discretise the problem using the non-conforming Morley finite element which is a piecewise quadratic C^0-continuous element for biharmonic problems.

The full source code of the example reads as follows:

`examples.ex02.visualize()`

1.1.4 examples.ex03 module

Linear elastic eigenvalue problem.

`examples.ex03.visualize()`

1.1.5 examples.ex04 module

Contact problem.

Mortar methods allow setting interface conditions on non-matching meshes. They are useful also when solving variational inequalities such as elastic contact problems.

`examples.ex04.gap(`x`)`

`examples.ex04.visualize()`

1.1.6 examples.ex05 module

Integral condition.

This short example demonstrates the implementation of an integral boundary
condition

$$\int_\Gamma \nabla u \cdot \boldsymbol{n} \,\mathrm{d}s = 1$$

on a part of the boundary of the domain $\Gamma \subset \partial\Omega$
for the Laplace operator. In this example, Γ is the right boundary of the unit square and the solution satisfies $u = 0$ on the bottom boundary and $\nabla u \cdot \boldsymbol{n} = 0$ on the rest of the boundaries. The constraint is introduced via a Lagrange multiplier leading to a saddle point system.

1.1.7 examples.ex06 module

High-order plotting.

This simple example demonstrates the usage and visualisation of biquadratic finite element basis. Many plotting tools, including matplotlib, provide tools for visualising piecewise-linear triangular fields. Visualisation of higher-order basis functions cannot be done directly and the mesh should be further refined just for visualisation purposes.

`CellBasis` object includes the method `refinterp()` which refines and simultaneously interpolates any solution vector. The resulting mesh is non-conforming, i.e. the connectivity between neighboring elements is lost, and hence it can be used only for visualisation purposes.

Note: As of 0.4.0, this functionality is included in `plot()`, i.e. inputting `CellBasis` instead of `Mesh` uses `refinterp()` automatically. The steps in this example are still useful when, e.g., exporting to different formats for visualization purposes.

As an example, we solve the Poisson equation in a unit square with zero boundary conditions and biquadratic basis on quadrilateral elements. The quadrilateral elements are defined using an isoparametric local-to-global mapping.

1.1.8 examples.ex07 module

Discontinuous Galerkin method.

`examples.ex07.`**`visualize`**`()`

1.1.9 examples.ex08 module

Visualize Argyris basis.

`examples.ex08.`**`visualize`**`()`

1.1.10 examples.ex09 module

Preconditioned conjugate gradient for 3-D Poisson.

Note: This example will make use of the external packages PyAMG or pyamgcl, if installed.

Whereas most of the examples thus far have used direct linear solvers, this is not appropriate for larger problems, which includes most of those posed in three dimensions.

Direct methods fail to scale beyond a certain size, typically of the order of a few millions of unknowns, due to their intrinsic memory requirements and shear computational cost. This makes preconditioned iterative methods the only viable approach for solution of large scale problems. (Demidov 2019)

scikit-fem provides access to simple preconditioners (diagonal and incomplete-LU) from SciPy, but it's also easy to connect others from external packages, e.g. PyAMG or AMGCL for algebraic multigrid.

The combination of a Krylov subspace method with algebraic multigrid (AMG) as a preconditioner is considered to be one of the most effective choices for solution of such systems. (Demidov 2019)

These four preconditioners are demonstrated with a conjugate gradient solver for a simple Poisson problem,

$$-\Delta u = 1, \quad \text{in } \Omega,$$
$$u = 0, \quad \text{on } \partial\Omega,$$

where $\Omega = (0, 1)^3$.

- Demidov, D. (2019). AMGCL: an efficient, flexible, and extensible algebraic multigrid implementation. arXiv:1811.05704

`examples.ex09.`**`build_pc_amgsa`**(*A: spmatrix*, ***kwargs*) → LinearOperator

AMG (smoothed aggregation) preconditioner

1.1.11 examples.ex10 module

Minimal surface problem.

This example solves the nonlinear minimal surface problem using Newton's method.

1.1.12 examples.ex11 module

Linear elasticity.

This example solves the linear elasticity problem using trilinear elements. The weak form of the linear elasticity problem is defined in `skfem.models.elasticity.linear_elasticity()`.

1.1.13 examples.ex12 module

Postprocessing Laplace equation.

A basic postprocessing step in finite element analysis is evaluating linear forms over the solution. For the Poisson equation, the integral of the solution (normalized by the area) is the 'Boussinesq k-factor'; for the square it's roughly 0.03514, for the circle 1/Pi/8 = 0.03979. Linear forms are easily evaluated in skfem using the 1-D arrays assembled using the @LinearForm decorator. In poisson, the linear form required for simple integration happens to be the same one used on the right-hand side of the differential equation, so it's already to hand.

Another is interpolation; i.e. evaluation of the solution at a specified point which isn't necessarily a node of the mesh. For this problem, the maximum of the solution (normalized by the area) is the 'Boussinesq k'-factor'; by symmetry, this occurs for squares (k' = 0.07363) and circles (k' = 1/Pi/4) at the centre and so can be evaluated by interpolation.

`examples.ex12.`**`visualize`**`()`

1.1.14 examples.ex13 module

Laplace with mixed boundary conditions.

This example is another extension of *ex01.py*, still solving the Laplace equation but now with mixed boundary conditions, two parts isopotential (charged and earthed) and the rest insulated.

The example is $\Delta u = 0$ in $\Omega = \{(x, y) : 1 < x^2 + y^2 < 4,\ 0 < \theta < \pi/2\}$, where $\tan\theta = y/x$, with $u = 0$ on $y = 0$ and $u = 1$ on $x = 0$. The mesh is first constructed as a rectangle in the $r\theta$-plane, where the isopotential parts are conveniently tagged using *skfem.Mesh.with_boundaries*; then the mesh is mapped to the xy-plane.

The exact solution is $u = 2\theta/\pi$. The field strength is $|\nabla u|^2 = 4/\pi^2(x^2 + y^2)$ so the conductance (for unit potential difference and conductivity) is $\|\nabla u\|^2 = 2\ln 2/\pi$.

examples.ex13.**visualize**()

1.1.15 examples.ex14 module

Laplace with inhomogeneous boundary

Another simple modification of *ex01.py*, this time showing how to impose coordinate-dependent Dirichlet conditions with `condense()` and `doflocs`. The forcing term is suppressed for simplicity. The boundary values are set as the real part $x^2 - y^2$ of an analytic complex function $(x + \mathrm{i}y)^2$ which is harmonic and so that is the exact solution through the domain.

This is checked quantitatively by computing the integral of the squared magnitude of the gradient, by evaluating the quadratic form associated with the laplacian at the solution; the exact value is 8/3.

This code would also work with `ElementTriP1()`, in which case, since the three nodes of the elements coincide with the points of the mesh, the coordinate-dependent Dirichlet conditions could be assigned more directly; however, the present method is recommended as more general.

examples.ex14.**dirichlet**(x)

return a harmonic function

examples.ex14.**visualize**()

1.1.16 examples.ex15 module

One-dimensional Poisson.

1.1.17 examples.ex16 module

Spatially varying coefficient.

This example demonstrates a spatially varying coefficient.

Legendre's equation in self-adjoint Sturm–Liouville form is

$$((1 - x^2)u')' + ku = 0, \quad (-1 < x < 1)$$

The eigenvalues are $k = n(n + 1)$ for $n = 0, 1, 2, \ldots$ The conventional normalization is $u(1) = 1$.

The x-coordinate for the spatially varying coefficient $1 - x^2$ is accessed inside the bilinear form as *w.x[0]*.

1.1.18 examples.ex17 module

Insulated wire.

This example solves the steady heat conduction with generation in an insulated wire. In radial coordinates, the governing equations read: find T satisfying

$$\nabla \cdot (k_0 \nabla T) + A = 0, \quad 0 < r < a,$$

and

$$\nabla \cdot (k_1 \nabla T) = 0, \quad a < r < b,$$

with the boundary condition

$$k_1 \frac{\partial T}{\partial r} + hT = 0, \quad \text{on } r = b.$$

The parameter values are $k_0 = 101$, $k_1 = 11$, $A = 5$, $h = 7$, and the geometry is defined as $a = 2$ and $b = 3$.

For comparison purposes, the exact solution at the origin is

$$T(r = 0) = \frac{Ab^2}{4k_0} \left(\frac{2k_0}{bh} + \frac{2k_0}{k_1} \log \frac{b}{a} + 1 \right).$$

1.1.19 examples.ex18 module

Stokes equations.

This solves for the creeping flow problem in the primitive variables, i.e. velocity and pressure instead of the stream-function. These are governed by the Stokes momentum

$$0 = -\rho^{-1} \nabla p + \boldsymbol{f} + \nu \Delta \boldsymbol{u}$$

and continuity equations

$$\nabla \cdot \boldsymbol{u} = 0.$$

This is an example of a mixed problem because it contains two different kinds of unknowns; pairs of elements for them have to be chosen carefully. One of the simplest workable choices is the Taylor–Hood element: P_2 for velocity and P_1 for pressure.

Once the velocity has been found, the stream-function ψ can be calculated by solving the Poisson problem

$$-\Delta \psi = \operatorname{rot} \boldsymbol{u},$$

where $\operatorname{rot} \boldsymbol{u} \equiv \partial u_y / \partial x - \partial u_x / \partial y$. The boundary conditions are that the stream-function is constant around the impermeable perimeter; this constant can be taken as zero without loss of generality. In the weak formulation

$$(\nabla \phi, \nabla \psi) = (\phi, \operatorname{rot} \boldsymbol{u}) \quad \forall \phi \in H_0^1(\Omega),$$

the right-hand side can be converted using Green's theorem and the no-slip condition to not involve the derivatives of the velocity:

$$(\phi, \operatorname{rot} \boldsymbol{u}) = (\boldsymbol{rot}\, \phi, \boldsymbol{u})$$

where $\boldsymbol{rot}$ is the adjoint of rot:

$$\boldsymbol{rot}\, \phi \equiv \frac{\partial \phi}{\partial y} \hat{i} - \frac{\partial \phi}{\partial x} \hat{j}.$$

1.1.20 examples.ex19 module

Heat equation.

The solutions of the heat equation

$$\frac{\partial T}{\partial t} = \kappa \Delta T$$

in various tensor-product domains consist of the sum of modes which are tensor-products of the modes on the factor-domains (Carslaw & Jaegar 1959, §5.6).

- Carslaw, H. S., & J. C. Jaeger (1959). Conduction of Heat in Solids (2nd ed.). Oxford University Press

For example, in the rectangle $|x| < w_0, |y| < w_1$, with homogeneous Dirichlet boundary conditions, the modes are products of a cosine in each direction,

$$\exp\left[-\frac{\kappa\pi^2 t}{4}\left\{\left(\frac{2n_0+1}{w_0}\right)^2+\left(\frac{2n_1+1}{w_1}\right)^2\right\}\right]\cos\frac{(2n_0+1)\pi x}{2w_0}\cos\frac{(2n_1+1)\pi y}{2w_1}$$

for $n_0, n_1 = 0, 1, 2, \ldots$.

Here we simulate the decay of the fundamental, $n_0 = n_1 = 0$, discretizing time using the generalized ('theta method') trapezoidal rule.

For a constant time-step, this leads to a linear algebraic problem at each time with the same matrix but changing right-hand side. This motivates factoring the matrix; e.g. with *scipy.sparse.linalg.splu*.

`examples.ex19.`**`backsolve`**`()`

solve(rhs[, trans])

Solves linear system of equations with one or several right-hand sides.

Parameters

- **rhs** (*ndarray, shape (n,) or (n, k)*) – Right hand side(s) of equation
- **trans** (*{'N', 'T', 'H'}, optional*) – Type of system to solve:

```
'N':   A   @ x == rhs  (default)
'T':   A^T @ x == rhs
'H':   A^H @ x == rhs
```

i.e., normal, transposed, and hermitian conjugate.

Returns

x – Solution vector(s)

Return type

ndarray, shape `rhs.shape`

`examples.ex19.`**`evolve`**`(`*t: float*, *u: ndarray*`)` → Iterator[Tuple[float, ndarray]]

`examples.ex19.`**`exact`**`(`*t: float*`)` → ndarray

1.1.21 examples.ex20 module

Creeping flow.

The stream-function ψ for two-dimensional creeping flow is governed by the biharmonic equation

$$\nu\Delta^2\psi = \operatorname{rot}\boldsymbol{f}$$

where ν is the kinematic viscosity (assumed constant), $\boldsymbol{f}$ the volumetric body-force, and $\operatorname{rot}\boldsymbol{f} \equiv \partial f_y/\partial x - \partial f_x/\partial y$. The boundary conditions at a wall are that ψ is constant (the wall is impermeable) and that the normal component of its gradient vanishes (no slip). Thus, the boundary value problem is analogous to that of bending a clamped plate, and may be treated with Morley elements as in the Kirchhoff plate tutorial.

Here we consider a buoyancy force $\boldsymbol{f} = x\hat{j}$, which arises in the Boussinesq approximation of natural convection with a horizontal temperature gradient (Batchelor 1954).

For a circular cavity of radius a, the problem admits a polynomial solution with circular stream-lines:

$$\psi = \left(1-(x^2+y^2)/a^2\right)^2/64.$$

1.1.22 examples.ex21 module

Structural vibration.

This example demonstrates the solution of a three-dimensional vector-valued problem. For this purpose, we consider an elastic eigenvalue problem.

The governing equation for the displacement of the elastic structure Ω reads: find $\boldsymbol{u} : \Omega \to \mathbb{R}^3$ satisfying

$$\rho\ddot{\boldsymbol{u}} = \operatorname{div}\boldsymbol{\sigma}(\boldsymbol{u}) + \rho\boldsymbol{g},$$

where $\rho = 8050\,\frac{\mathrm{kg}}{\mathrm{m}^3}$ is the density, $\boldsymbol{g}$ is the gravitational acceleration and $\boldsymbol{\sigma}$ is the linear elastic stress tensor defined via

$$\boldsymbol{\sigma}(\boldsymbol{w}) = 2\mu\boldsymbol{\epsilon}(\boldsymbol{w}) + \lambda\operatorname{tr}\boldsymbol{\epsilon}(\boldsymbol{w})\boldsymbol{I},$$
$$\boldsymbol{\epsilon}(\boldsymbol{w}) = \frac{1}{2}(\nabla\boldsymbol{w} + \nabla\boldsymbol{w}^T).$$

Moreover, the Lamé parameters are given by

$$\lambda = \frac{E}{2(1+\nu)}, \quad \mu = \frac{E\nu}{(1+\nu)(1-2\nu)},$$

where the Young's modulus $E = 200 \cdot 10^9$ Pa and the Poisson ratio $\nu = 0.3$.

We consider two kinds of boundary conditions. On a *fixed part* of the boundary, $\Gamma_D \subset \partial\Omega$, the displacement field $\boldsymbol{u}$ satisfies

$$\boldsymbol{u}|_{\Gamma_D} = \boldsymbol{0}.$$

Moreover, on a *free part* of the boundary, $\Gamma_N = \partial\Omega \setminus \Gamma_D$, the *traction vector* $\boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n}$ satisfies

$$\boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n} \cdot \boldsymbol{n}|_{\Gamma_N} = 0,$$

where $\boldsymbol{n}$ denotes the outward normal.

Neglecting the gravitational acceleration $\boldsymbol{g}$ and assuming a periodic solution of the form

$$\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{w}(\boldsymbol{x})\sin\omega t,$$

leads to the following eigenvalue problem with $\boldsymbol{w}$ and ω as unknowns:

$$\operatorname{div}\boldsymbol{\sigma}(\boldsymbol{w}) = \rho\omega^2\boldsymbol{w}.$$

The weak formulation of the problem reads: find $(\boldsymbol{w}, \omega) \in V \times \mathbb{R}$ satisfying

$$(\boldsymbol{\sigma}(\boldsymbol{w}), \boldsymbol{\epsilon}(\boldsymbol{v})) = \rho\omega^2(\boldsymbol{w}, \boldsymbol{v}) \quad \forall\boldsymbol{v} \in V,$$

where the variational space V is defined as

$$V = \{\boldsymbol{w} \in [H^1(\Omega)]^3 : \boldsymbol{w}|_{\Gamma_D} = \boldsymbol{0}\}.$$

The bilinear form for the problem can be found from `skfem.models.elasticity.linear_elasticity()`. Moreover, the mesh for the problem is loaded from an external file *beams.msh*, which is included in the source code distribution.

1.1.23 examples.ex22 module

Adaptive Poisson equation.

This example solves *ex01.py* adaptively in an L-shaped domain. Using linear elements, the error indicators read

$$\eta_K^2 = h_K^2 \|f\|_{0,K}^2$$

for each element K, and

$$\eta_E^2 = h_E \|[[\nabla u_h \cdot n]]\|_{0,E}^2$$

for each edge E.

examples.ex22.**eval_estimator**(*m*, *u*)

examples.ex22.**load_func**(*x*, *y*)

examples.ex22.**visualize**()

1.1.24 examples.ex24 module

Backward-facing step.

Following the example stokesex, this is another example of the Stokes flow. The difference here is that the domain has an inlet (with an imposed velocity) and an outlet (through which fluid issues against a uniform pressure).

The geometry is taken from Barkley et al (2002); i.e. an expansion ratio of 2, one step-length upstream and 35 downstream.

- Barkley, D., M. G. M. Gomes, & R. D. Henderson (2002). Three-dimensional instability in flow over a backward-facing step. Journal of Fluid Mechanics 473. pp. 167–190. doi:10.1017/s002211200200232x

examples.ex24.**parabolic**(*x*)

return the plane Poiseuille parabolic inlet profile

1.1.25 examples.ex25 module

Forced convection.

We begin the study of forced convection with the plane Graetz problem; viz. the steady distribution of temperature in a plane channel with zero inlet temperature and unit temperature on the walls and a steady laminar unidirectional parabolic plane-Poiseuille flow.

The governing advection–diffusion equation is

$$\mathrm{Pe}\, u \frac{\partial T}{\partial x} = \nabla^2 T$$

where the velocity profile is

$$u(y) = 6y(1-y), \qquad (0 < y < 1)$$

The equations here have been nondimensionalized by the width of the channel and the volumetric flow-rate. The governing parameter is the Péclet number, being the mean velocity times the width divided by the thermal diffusivity.

Because the problem is symmetric about $y = \frac{1}{2}$, only half is solved here, with natural boundary conditions along the centreline.

1.1.26 examples.ex26 module

Restricting a problem to a subdomain.

The *ex17.py* example solved the steady-state heat equation with uniform volumetric heating in a central core surrounded by an annular insulating layer of lower thermal conductivity. Here, the problem is completely restricted to the core, taking the temperature as zero throughout the annulus.

Thus the problem reduces to the same Poisson equation with uniform forcing and homogeneous Dirichlet conditions:

$$\nabla \cdot (k \nabla T) + A = 0, \qquad 0 < r < a$$

with

$$T = 0, \qquad \text{on} \quad r = a.$$

The exact solution is

$$T = \frac{s}{4k}(a^2 - r^2).$$

The novelty here is that the temperature is defined as a finite element function throughout the mesh ($r < b$) but only solved on a subdomain.

1.1.27 examples.ex28 module

Conjugate heat transfer.

The forced convection example can be extended to conjugate heat transfer by giving a finite thickness and thermal conductivity to one of the walls.

The example is modified to a configuration for which there is a fully developed solution which can be found in closed form: given a uniform (but possibly different) heat flux over each of the walls, the temperature field asymptotically is the superposition of a uniform longitudinal gradient and a transverse profile; here the analysis of a pipe of circular section (Bird, Stewart, & Lightfoot 1960, §3–13) is modified for a planar duct (Mallinson, McBain, & Brown 2019). The two-dimensional equation in the fluid $-1 < y < 1$

$$\mathrm{Pe}\left(1 - y^2\right) \frac{\partial T}{\partial x} = \nabla^2 T, \qquad (-1 < y < 1)$$

has the exact solution

$$T(x, y) = \frac{3x}{4\mathrm{Pe}} - \frac{(5 - y^2)(1 - y^2)}{16} - \frac{y}{2}.$$

Here Pe is the dimensionless Péclet number based on the centreline-velocity and channel half-height.

The governing equation in the solid is $\nabla \cdot k \nabla T = 0$ where k the ratio of the thermal conductivity to that of the fluid. This has the exact solution matching that in the fluid along the interface $y = -1$ is:

$$T(x, y) = \frac{3x}{4\mathrm{Pe}} + \frac{1}{2} - \frac{1 + y}{k}, \qquad (y < -1)$$

This combined fully developed solution applies throughout the entire domain if it is specified along the fluid and solid inlet $x = 0$, if the ceiling is insulated $\partial T / \partial y = 0$ on $y = 1$, if the floor is uniformly heated $k \partial t / \partial y = 1$ on $y = -2$, and if the longitudinal gradient is applied as a uniform Neumann condition on the outlet, $k \partial T / \partial x = 3k/4\mathrm{Pe}$ on $x = \ell$.

In conjugate heat transfer, part of the domain is advection–conduction while the rest is pure conduction; the main difficulty is specifying different governing equations on different subdomains. One way to do this has already been demonstrated: subdomain by only assembling the operator over a basis restricted to the elements belonging to a particular subdomain.

- Bird, R. B., W. E. Stewart, & E. N. Lightfoot (1960). Transport Phenomena. New York: Wiley
- Mallinson, S. G., McBain, G. D. & Brown, B. R. (2019). Conjugate heat transfer in thermal inkjet printheads. 14th International Conference on Heat Transfer, Fluid Mechanics and Thermodynamics, Wicklow, Ireland.

`examples.ex28.exact`(*x: ndarray*, *y: ndarray*) → ndarray

return the exact fully developed solution at specified points

1.1.28 examples.ex29 module

Linear hydrodynamic stability.

The linear stability of one-dimensional solutions of the Navier–Stokes equations is governed by the Orr–Sommerfeld equation (Drazin & Reid 2004, p. 156). This is expressed in terms of the stream-function ϕ of the perturbation, giving a two-point boundary value problem

$$\alpha\phi(\pm1) = \phi'(\pm1) = 0$$

for a complex fourth-order ordinary differential equation,

$$\left(\alpha^2 - \frac{\mathrm{d}^2}{\mathrm{d}z^2}\right)^2 \phi = (\mathrm{j}\alpha R)\left\{(c-U)\left(\alpha^2 - \frac{\mathrm{d}^2}{\mathrm{d}z^2}\right)\phi - U''\phi,\right\}$$

where $U(z)$ is the base velocity profile, c and α are the wavespeed and wavenumber of the disturbance, and R is the Reynolds number. In the 'temporal' stability problem, R and α are specified as positive and c found as the complex eigenvalue.

The fourth-order derivatives would require C^1 finite elements, e.g. of the Hermite family (Mamou & Khalid 2004); however, this can be avoided by reverting to the system from which the Orr–Sommerfeld stream-function equation is derived (Drazin & Reid 2004, eq. 25.9, p. 155), which is expressed in terms of pressure p and longitudinal u and transverse w components of the disturbance to the velocity:

$$\begin{aligned}\left(\mathrm{j}\alpha RU(z) + \alpha^2 - \frac{\mathrm{d}^2}{\mathrm{d}z^2}\right)u + RU'(z)w + \mathrm{j}\alpha Rp &= \mathrm{j}\alpha Rcu\\ \left(\mathrm{j}\alpha RU(z) + \alpha^2 - \frac{\mathrm{d}^2}{\mathrm{d}z^2}\right)w + R\frac{\mathrm{d}p}{\mathrm{d}z} &= \mathrm{j}\alpha Rcw\\ \mathrm{j}\alpha Ru + R\,\frac{\mathrm{d}w}{\mathrm{d}z} &= 0\end{aligned}$$

This primitive system is second-order and can be discretized using one-dimensional Taylor–Hood or Mini elements; here the latter are used.

The classical test-case for this problem is plane Poiseuille flow $U(z) = 1 - z^2$ on $-1 < z < 1$ at $\alpha = 1$ and $R = 10^4$ (Drazin & Reid 2004, figure 4.19; Mamou & Khalid 2004), typically seeking only the 'even' modes for which $u(0) = u(1) = w'(0) = w(1) = 0$. Good agreement with reference results for the complex wavespeed spectrum (Criminale, Jackson, & Joslin 2003, table 3.1) is obtained on a uniform mesh of 64 segments.

- Criminale, W. O., Jackson, T. L.,, Joslin, R. D. (2003). *Theory and Computation in Hydrodynamic Stability.* Cambridge: Cambridge University Press. doi:10.1017/CBO9780511550317
- Drazin, P. G., Reid, W. H. (2004). *Hydrodynamic Stability.* Cambridge University Press. doi:10.1017/CBO9780511616938
- Mamou, M. & Khalid, M. (2004). Finite element solution of the Orr–Sommerfeld equation using high precision Hermite elements: plane Poiseuille flow. *International Journal for Numerical Methods in Fluids* 44. pp. 721–735. doi:10.1002/fld.661

1.1.29 examples.ex30 module

Krylov–Uzawa method for the Stokes equation.

This example solves the Stokes equation iteratively in a square domain.

Direct solvers for the Stokes equation do not scale; iterative solvers are required for all but the smallest two-dimensional problems. A square domain is used here instead of the disk since it easier to generate the finer mesh required to demonstrate the time taken for a slightly larger problem.

A classical iterative procedure for the Stokes equation is the Uzawa conjugate gradient method in which an auxiliary operator is formed which maps a guessed pressure field p to the divergence

$$\Theta(p) = \nabla \cdot \{\boldsymbol{u}(p)\}$$

of the velocity field which satisfies the momentum equation with that pressure [GLOWINSKI-PIRONNEAU] [BRAESS].

$$\nu \Delta \boldsymbol{u} = -\rho^{-1} \nabla p + \boldsymbol{f}$$

This operator $\Theta(p)$ is affine but subtracting the divergence given by zero pressure gives a linear operator

$$K(p) = \Theta(p) - \Theta(0)$$

for which the linear operator equation

$$K(p) = -\Theta(0)$$

can be solved by a Krylov method, classically the method of conjugate gradients.
The solution is the pressure p that gives the velocity

$\boldsymbol{u}$ that has zero divergence.

At each iteration, the above vector Poisson equation is solved for a velocity field. As the problem gets larger, this too should be solved iteratively. The method of conjugate gradients could be used for this too, but a catch is that `scipy.sparse.linalg.cg()` is not re-entrant and so cannot be used at both levels. This is simply avoided here by using `scipy.sparse.linalg.minres()` for the pressure.

The results may be assessed using the value of the stream-function at the centre. As in biharmonic, the stream-function satisfies the same boundary value problem as the deflection of a clamped plate, for which the central deflection is known to be approximately 0.162/128 [LOVE].

`examples.ex30.`**`dilatation`**(*pressure: ndarray*) → ndarray

compute the dilatation corresponding to a guessed pressure

`examples.ex30.`**`flow`**(*pressure: ndarray*) → ndarray

compute the velocity corresponding to a guessed pressure

1.1.30 examples.ex31 module

Curved elements.

This example solves the eigenvalue problem

$$-\Delta u = \lambda u \quad \text{in } \Omega,$$

with the boundary condition $u|_{\partial\Omega} = 0$ using isoparametric mapping via biquadratic basis and finite element approximation using fifth-order quadrilaterals.

1.1.31 examples.ex32 module

Block diagonally preconditioned Stokes solver.

Note: This examples requires an implementation of AMG (either pyamgcl or pyamg).

This example again solves the Stokes problem,

$$0 = -\rho^{-1}\nabla p + x\boldsymbol{j} + \nu\Delta\boldsymbol{u}$$

$$\nabla \cdot \boldsymbol{u} = 0.$$

but this time in three dimensions, with an algorithm that scales to reasonably fine meshes (a million tetrahedra in a few minutes).

The exact solution for this equation in an ellipsoid is known [MCBAIN].

With Taylor-Hood elements, the discrete form of the problem is

$$\begin{matrix} A & B.T \\ B & 0 \end{matrix} \begin{Bmatrix} u \\ package \end{Bmatrix} = \begin{bmatrix} x\boldsymbol{j} \\ 0 \end{bmatrix}$$

A simple but effective preconditioning strategy [ELMAN] is a block-diagonal approach with algebraic multigrid for the momentum block and a diagonal approximation to its mass matrix for the pressure. Algebraic multigrid is easily accessible from scikit-fem via the external packages AMGCL or PyAMG; either will do here, and in either case the viscous matrix A is condensed to eliminate the essential boundary conditions (here zero velocity on the walls) and then passed to the external AMG library.

Because the two parts of the preconditioner differ in form, it is easier to define their action by a function, wrapped up as a LinearOperator which can then be passed to the MINRES sparse iterative solver from SciPy.

class examples.ex32.**Sphere**

Bases: NamedTuple

mesh() → MeshTet1

pressure(*x*, *y*, *z*) → ndarray

Exact pressure at zero Grashof number.

- McBain, G. D. (2016). `Creeping convection in a horizontally heated ellipsoid <http://people.eng.unimelb.edu.au/imarusic/proceedings/20/548/%20Paper.pdf>`_. *Proceedings of the Twentieth Australasian Fluid Mechanics Conference*, eq. 8

pressure_error()

examples.ex32.**build_pc_amg**(*A: spmatrix*, ***kwargs*) → LinearOperator

examples.ex32.**precondition**(*uvp*)

1.1.32 examples.ex33 module

Nédélec elements.

This example solves the vector-valued problem

$$\nabla\nabla E + E = f \quad \text{in } \Omega,$$

in domain $\Omega = [-1, 1]^3$ with the boundary condition $E \times n|_{\partial\Omega} = 0$ using the lowest order Nédélec edge element.

Note: The loading is from https://www.dealii.org/reports/nedelec/nedelec.pdf.

1.1.33 examples.ex34 module

Euler-Bernoulli beam.

This example solves the Euler-Bernoulli beam equation

$$(EIu'')'' = 1 \quad \text{in } [0, 1],$$

with the boundary conditions $u(0) = u'(0) = 0$ and using cubic Hermite elements. The analytical solution gives $u(1) = 1/8$.

1.1.34 examples.ex35 module

Computation of the characteristic impedance and velocity factor of RG316 coaxial cable.

This example solves the series inductance (per meter) and parallel capacitance (per meter) of RG316 coaxial cable. These values are then used to compute the characteristic impedance and velocity factor of the cable.

From transmission line theory it is known, that the characteristic impedance of a lossless transmission line is

$$Z = \sqrt{\frac{L}{C}},$$

while the phase velocity is

$$v = \frac{1}{\sqrt{LC}},$$

where L is the series inductance per unit length of the transmission line and C is the parallel capacitance per unit length of the transmission line.

Further, the phase velocity relative to the speed of light is called the velocity factor of the transmission line.

RG316

A piece of coaxial cable is composed of an inner conductor, which is surrounded by a dielectric insulator. The dielectric insulator in turn is surrounded by an outer conductor. Finally, the outer conductor is surrounded by an outer insulator.

For an RG316 cable, the dimensions and materials of the components are

- Inner conductor: OD 0.5mm, silver plated copper
- Inner insulator: OD 1.52mm, PTFE

- Outer conductor: OD 1.98mm, silver plated copper
- Outer insulator: OD 2.48mm, PEF

RG316 has a nominal characteristic impedance of 50 ohms and a velocity factor of 0.69.

Inductance

Inductance of the cable is computed using the magnetostatic equations

$$\nabla \cdot \boldsymbol{B} = 0$$
$$\nabla \times \boldsymbol{H} = \boldsymbol{J}$$

and the constitutive relation

$$\boldsymbol{B} = \mu \boldsymbol{H},$$

where μ is the permability of the medium.

Since $\boldsymbol{B}$ is divergence free, it can be written in terms of a vector potential $\boldsymbol{A}$ as

$$\boldsymbol{B} = \nabla \times \boldsymbol{A}.$$

Thus we have the strong form for the vector potential $\boldsymbol{A}$ as

$$\nabla \times (\mu^{-1} \nabla \times \boldsymbol{A}) = \boldsymbol{J}.$$

The corresponding weak form is: find $\boldsymbol{A} \in V$ such that

$$\int_\Omega (\mu^{-1} \nabla \times \boldsymbol{A}) \cdot (\nabla \times \boldsymbol{v}) \, \mathrm{d}x - \int_{\partial\Omega} (\mu^{-1} \nabla \times \boldsymbol{A}) \times \boldsymbol{n} \cdot \boldsymbol{v} \, \mathrm{d}x = \int_\Omega \boldsymbol{J} \cdot \boldsymbol{v} \, \mathrm{d}x \quad \forall \boldsymbol{v} \in V.$$

We take the boundary condition $\boldsymbol{B} \cdot \boldsymbol{n} = 0$ on $\partial\Omega$, which is equivalent to $\boldsymbol{A} = 0$ on $\partial\Omega$. This is an essential boundary condition, which is enforced by the choice of V. Thus we have the final weak form: find $\boldsymbol{A} \in V$ such that

$$\int_\Omega (\mu^{-1} \nabla \times \boldsymbol{A}) \cdot (\nabla \times \boldsymbol{v}) \, \mathrm{d}x = \int_\Omega \boldsymbol{J} \cdot \boldsymbol{v} \, \mathrm{d}x \quad \forall \boldsymbol{v} \in V.$$

For this application Ω is taken to be the cross section of the coaxial cable, and it is assumed that the cable has infinite length. It is assumed that the functions $\boldsymbol{J}$, $\boldsymbol{A}$ as well as and any $\boldsymbol{v} \in V$ depend only on coordinates x_1 and x_2 in the cross-section plane, and have a nonzero component only in the perpendicular direction to the cross-section plane. In other words, they are assumed to have the following form

$$\begin{aligned} \boldsymbol{J} &= J(x_1, x_2)\boldsymbol{e_3} \\ \boldsymbol{A} &= A(x_1, x_2)\boldsymbol{e_3} \\ \boldsymbol{v} &= v(x_1, x_2)\boldsymbol{e_3} \end{aligned}$$

This reduces the problem to two dimensions. Taking the curl of a function of the assumed form and substituting the result in the weak form gives a more familiar weak form in the cross-section plane: find $A \in V$ such that

$$\int_\Omega \mu^{-1} (\nabla A \cdot \nabla v) \, \mathrm{d}x = \int_\Omega J v \, \mathrm{d}x \quad \forall v \in V.$$

In order to actually compute the inductance (per unit length) of the cable, a current is passed through the inner conductor while an equal but opposite current is passed through the outer conductor. The energy (per unit length) stored in the produced magnetic field is computed as

$$E = \frac{1}{2} \int_\Omega \mu^{-1} |\nabla A|^2 \, \mathrm{d}x$$

However, the energy (per unit length) stored in the magnetic field of the inductor can also be stated in terms of its inductance (per unit length) as

$$E = \frac{1}{2} L I^2,$$

where L is the inductance (per unit length) and I is the current passed through the inductor. Thus

$$L = \frac{2E}{I^2}$$

Capacitance

Capacitance of the cable is computed using the electrostatic equations

$$\nabla \times \boldsymbol{E} = \boldsymbol{0}$$
$$\nabla \cdot \boldsymbol{D} = 0$$

and the constitutive relation

$$\boldsymbol{D} = \epsilon \boldsymbol{E},$$

where ϵ is the permittivity of the medium.

Since $\boldsymbol{E}$ is curl-free, it can be written in terms of a scalar potential U as

$$\boldsymbol{E} = -\nabla U$$

Thus we have the strong form for the scalar potential U as

$$-\nabla \cdot (\epsilon \nabla U) = 0.$$

However, this equation is only meaningful in a dielectric medium. In a conductor, the electric field is zero, and thus the potential is constant (and conceptually $\epsilon \to \infty$). The conductors need to be excluded from the computation domain.

In any case, the equation has the familiar weak form: find U such that

$$\int_\Omega \epsilon \nabla U \cdot \nabla v \, \mathrm{d}x = 0 \quad \forall v \in V.$$

Consider again the cross-section plane of the coaxial cable, with the cable itself extending out-of-plane into infinity. Take U to depend only on the coordinates x_1 and x_2 in the cross-section plane. This again reduces the problem to two dimensions.

Due to conducting media in the cross-section, the problem needs to split into two domains: the first domain consisting of the inner insulator and the second domain consisting of the space outside the outer conductor. In both domains, we have a non-homogeneous Dirichlet boundary condition for U on the conductor surfaces, while in the second domain the potential has a homogeneous Neumann condition on the free space boundary.

In order to actually compute the capacitance (per unit length) of the cable, a potential is set on the inner conductor while an equal but opposite potential is set on the outer conductor. The energy (per unit length) of the produced electic field is computed as

$$E = \frac{1}{2} \int_\Omega \epsilon |\nabla U|^2 \, \mathrm{d}x$$

However, the energy (per unit length) stored in the electic field of a capacitor can also be stated in terms of its capacitance (per unit length) as

$$E = \frac{1}{2} C V^2,$$

where C is the capacitance (per unit length) and V is the potential difference across the capacitor. Thus

$$C = \frac{2E}{V^2}.$$

1.1.35 examples.ex36 module

Incompressible Hyperelasticity

This example solves the governing equations describing the mechanical response of a nearly incompressible elastomer using a mixed formulation. The elastomer, assumed to be made up of a Neo-Hookean solid, occupies the domain Ω in the undeformed configuration, with the internal stored energy given by

$$\int_\Omega \Psi(\mathbf{F}) d\mathbf{X}; \qquad \Psi(\mathbf{F}) = \mu/2(I_1 - 3) - \mu \ln(J) + \lambda/2(J-1)^2$$

where $I_1 = \mathbf{F} : \mathbf{F} = tr(\mathbf{F}^T\mathbf{F})$ and $J = \det(\mathbf{F})$ and $\mathbf{F} = \mathbf{I} + \nabla \mathbf{u}$ is the deformation gradient tensor. The standard variational formulation in the absence of body force and surface traction can be written as

$$\min_{u \in \mathcal{K}} \int_\Omega \Psi(\mathbf{F}) d\mathbf{X}$$

where $\mathcal{K}$ is a set of kinematically admissible fields that satisfy the Dirichlet boundary condition(s). However, this becomes ill-posed when $\lambda/\mu \to +\infty$. In order to circumvent this issue, we consider a mixed variational formulation, namely

$$\min_{\mathbf{u} \in \mathcal{K}} \max_p \int_\Omega \Psi * (\mathbf{F}, p) d\mathbf{X}$$

where

$$\Psi * (\mathbf{F}, p) = p(J - J*) + mu/2(I_1 - 3) - \mu \ln(J*) + \lambda/2(J* - 1)^2$$

and $J* = (\lambda + p + \sqrt{(\lambda+p)^2 + 4\lambda\mu})/(2\lambda)$. The numerical solution to the above problem requires choosing stable finite element spaces for the displacement ($\mathbf{u}$) and pressure (p). The corresponding weak form is given by

find $(\mathbf{u}, p) \in (V_1 x V_2)$ such that

$$\mathcal{F}_1 = \int_\Omega \left(\mu\mathbf{F} + p\mathbf{F}^{-T}\right) : \nabla \mathbf{v} d\mathbf{X} = 0$$

and

$$\mathcal{F}_2 = \int_\Omega \frac{\partial \Psi *}{\partial p} q \, d\mathbf{X} = 0$$

for all $(\mathbf{v}, q) \in (V_1 x V_2)$ and

$$V_1 = \left\{\mathbf{u} \ni \mathbf{u} \in (H^1(\Omega))^3 \cap u \in \mathcal{K}\right\}$$

and

$$V_2 = \left\{p \ni p \in L^2(\Omega)\right\}$$

Here, inspired by it's counterpart in fluid mechanics, we choose the lowest order Taylor-Hood element ($\mathbb{P}_2 - P_1$) which satifies the Babuska-Brezzi condition. Fore more details on the derivation, see http://pamies.cee.illinois.edu/Publications_files/IJNME_2015.pdf#page=4&zoom=100,312,414 The weak forms above result in a system of nonlinear algebraic equations for the degrees of freedom , and therefore needs to be solved using a nonlinear solver. In the example below, we linearize $\mathcal{F}_1$ and $\mathcal{F}_2$ and setup solve for incremental displacement and pressure dofs.

The following demonstrates uniaxial tension in one direction, and the lateral edges allowed to remain free. The geometry is a homogeneous unit cube made up of a Neo-Hookean solid with $\lambda/\mu = 10000$. For this loading and geometry, in the limit of $\lambda/\mu \to +\infty$, the deformation gradient would be given by $\mathbf{F} = \text{diag}(\lambda, 1/\sqrt{\lambda})$ and the pressure field admits a closed form solution $p = -\mu/\ell$ where ℓ is the applied stretch.

As another check, we can also compute the final volume of the deformed solid which, for a nearly incompressible solid, should be close to the initial undeformed volume.

examples.ex36.**A11**(*w*)

examples.ex36.**A12**(*w*)

examples.ex36.**A22**(*w*)

examples.ex36.**F1**(*w*)

examples.ex36.**F2**(*w*)

examples.ex36.**volume**(*w*)

1.1.36 examples.ex37 module

Mixed Poisson equation and Raviart-Thomas basis

1.1.37 examples.ex37b module

Mixed Poisson equation and Raviart-Thomas basis

1.1.38 examples.ex38 module

Point source.

Sources concentrated at points cannot be evaluated in the usual way, which involves discrete quadrature; instead, it requires direct use of the basis functions, as implemented in *CellBasis.point_source*.

Here this is demonstrated for a disk with homogeneous Dirichlet conditions. The exact solution is the well-known Green's function (e.g. Sadybekov, Turmetov, & Torebek 2015).

- Sadybekov, M. A., Turmetov, B. K. & Torebek, B. T. (2015). On an explicit form of the Green function of the Robin problem for the Laplace operator in a circle. *Advances in Pure and Applied Mathematics,* 6, 163-172. [doi: 10.1515/apam-2015-0003](https://doi.org/10.1515%2fapam-2015-0003)

examples.ex38.**greens**(*a: float*, *s: ndarray*, *x: ndarray*) → ndarray

Return the Green's function for a disk of radius *a*

with source at point *s*, evaluated at points *x*.

examples.ex38.**visualize**()

1.1.39 examples.ex39 module

Heat equation in one-dimension.

The solution of the heat equation is reduced from two dimensions in ex19 to one here, to demonstrate the different post-processing required.

The two-dimensional modes from ex19 reduce in the limit $w_1 \to \infty$ of the strip $|x| < w$ to

$$\exp\left[-\kappa\left\{\left(\frac{(2n+1)\pi}{2w}\right)^2\right\}t\right]\cos\frac{(2n+1)\pi x}{2w}$$

for $n = 0, 1, 2, \ldots$.

Here we simulate the decay of the fundamental, $n = 0$, again discretizing time using the generalized ('theta method') trapezoidal rule.

examples.ex39.**backsolve**()

solve(rhs[, trans])

Solves linear system of equations with one or several right-hand sides.

Parameters

- **rhs** (*ndarray, shape (n,) or (n, k)*) – Right hand side(s) of equation
- **trans** (*{'N', 'T', 'H'}, optional*) – Type of system to solve:

```
'N':   A   @ x == rhs  (default)
'T':   A^T @ x == rhs
'H':   A^H @ x == rhs
```

i.e., normal, transposed, and hermitian conjugate.

Returns

x – Solution vector(s)

Return type

ndarray, shape `rhs.shape`

examples.ex39.**evolve**(*t: float*, *u: ndarray*) → Iterator[Tuple[float, ndarray]]

examples.ex39.**exact**(*t: float*) → ndarray

1.1.40 examples.ex40 module

Hybridizable discontinuous Galerkin method.

This examples solves the Poisson equation with unit load using a technique where the finite element basis is first discontinous across element edges and then the continuity is recovered with the help of Lagrange multipliers defined on the mesh skeleton (i.e. a "skeleton mesh" consisting only of the edges of the original mesh).

As usual for these so-called "hybridizable" methods, the resulting system can be condensed to the skeleton mesh only using Schur's complement. However, this is not done here as the example is meant to simply demonstrate the use of finite elements defined on the mesh skeleton.

examples.ex40.**visualize**()

1.1.41 examples.ex41 module

Mixed meshes.

There exists a preliminary support for mixed meshes, i.e. meshes combining multiple element types such as triangles and quadrilaterals. This is mostly because the implementation of `asm()` supports lists of `AbstractBasis` objects and the resulting matrices of the multiple assembly operations is combined in the end.

The working principle is that a list of meshes is created by the user, one for each element type. Then the finite element basis is created separately for each mesh. Finally, the list of basis objects is passed on to `asm()`.

This example demonstrates how a mesh generated by Gmsh, consisting of triangles and quadrilaterals, can be used to solve the Poisson problem with unit load. Currently the user must verify that the finite element bases are compatible with one another. Moreover, the mixed meshes work properly only with finite elements that have nodal degrees-of-freedom. Support for higher-order elements on mixed meshes is work-in-progress.

examples.ex41.**visualize**()

1.1.42 examples.ex42 module

Periodic meshes.

When working with periodic boundary conditions, it may be more convenient to use a periodic mesh rather than to explicitly force the periodicity on the linear algebra level. This example defines a "discontinuous mesh topology" using `MeshTri1DG`. In practice this means that the local-to-global reference mapping is not done through an affine mapping but through the isoparametric mapping of a discontinuous P1 finite element. In the mesh data structure the vertices of the mesh are repeated even if shared across multiple elements through the degrees-of-freedom.

In this example we solve the advection equation with a Gaussian source in the middle of a unit square. The mesh is periodic from right-to-left so that the resulting solution is also periodic.

1.1.43 examples.ex43 module

Hyperelasticity.

The strain energy density function per unit undeformed volume of the isotropic hyperelastic Neo-Hookean material formulation is given as

$$\psi(\boldsymbol{F}) = \frac{\mu}{2}(I_1 - 3) - \mu \ln(J) + \frac{\lambda}{2}\ln^2(J)$$

by the first invariant of the right Cauchy-Green deformation tensor and the determinant of the deformation gradient (volume change).

$$I_1 = \mathrm{tr}(\boldsymbol{F}^T\boldsymbol{F}) = \boldsymbol{F} : \boldsymbol{F}$$
$$J = \det(\boldsymbol{F})$$

The variation of the strain energy function leads to

$$\delta\psi(\boldsymbol{F}) = \frac{\mu}{2}\delta I_1 - \mu\delta\ln(J) + \lambda\ln(J)\delta\ln(J)$$

with

$$\delta I_1 = 2\boldsymbol{F} : \delta\boldsymbol{F}$$
$$\delta\ln(J) = \boldsymbol{F}^{-T} : \delta\boldsymbol{F}$$

which finally results in

$$\delta\psi(\boldsymbol{F}) = \mu\boldsymbol{F} : \delta\boldsymbol{F} - (\lambda\ln(J) - \mu)\boldsymbol{F}^{-T} : \delta\boldsymbol{F}$$

The linearization of the variation leads to

$$\Delta\delta\psi(\boldsymbol{F}) = \mu\Delta\boldsymbol{F} : \delta\boldsymbol{F} + (\lambda\ln(J) - \mu)\Delta\boldsymbol{F}^{-T} : \delta\boldsymbol{F} + \lambda\left(\boldsymbol{F}^{-T} : \delta\boldsymbol{F}\right)\left(\boldsymbol{F}^{-T} : \Delta\boldsymbol{F}\right)$$

with

$$\Delta\boldsymbol{F}^{-T} = -\boldsymbol{F}^{-T}\Delta\boldsymbol{F}^T\boldsymbol{F}^{-T}$$

which may be alternatively formulated by expressions of the trace:

$$\Delta\delta\psi(\boldsymbol{F}) = \mu\Delta\boldsymbol{F} : \delta\boldsymbol{F} + (\lambda\ln(J) - \mu)\mathrm{tr}(\Delta\boldsymbol{F}\boldsymbol{F}^{-1}\delta\boldsymbol{F}\boldsymbol{F}^{-1}) + \lambda\mathrm{tr}(\delta\boldsymbol{F}\boldsymbol{F}^{-1})\mathrm{tr}(\Delta\boldsymbol{F}\boldsymbol{F}^{-1})$$

`examples.ex43.`**`deformation_gradient`**`(`*`w`*`)`

1.1.44 examples.ex44 module

Wave equation.

examples.ex44.**backsolve**()

solve(rhs[, trans])

Solves linear system of equations with one or several right-hand sides.

Parameters

- **rhs** (*ndarray, shape (n,) or (n, k)*) – Right hand side(s) of equation
- **trans** (*{'N', 'T', 'H'}, optional*) – Type of system to solve:

```
'N':   A   @ x == rhs  (default)
'T':   A^T @ x == rhs
'H':   A^H @ x == rhs
```

i.e., normal, transposed, and hermitian conjugate.

Returns

x – Solution vector(s)

Return type

ndarray, shape `rhs.shape`

examples.ex44.**bump**(*x*)

examples.ex44.**evolve**(*t*, *u*)

1.1.45 examples.ex45 module

1.1.46 examples.ex46 module

Waveguide cutoff analysis.

examples.ex46.**epsilon**(*x*)

1.1.47 examples.ex47 module

Projection between two meshes using supermesh.

1.1.48 Module contents

This is test message for examples module.

BIBLIOGRAPHY

[BRAESS] Braess, D. (2001). *Finite Elements.* Cambridge University Press, §IV.5.1

[GLOWINSKI-PIRONNEAU] Glowinski, R. & Pironneau, O. (1979). On numerical methods for the Stokes problem. In R. Glowinski, E. Y. Rodin & O. C. Zienkiewicz (eds.), *Energy Methods in Finite Element Analysis* (pp. 243–264), Wiley

[LOVE] Love, A. E. H. (1944). *A Treatise on the Mathematical Theory of Elasticity.* Dover

[ELMAN] Elman, H. C., Silvester, D. J.,, Wathen, A. J. (2014). *Finite Elements and Fast Iterative Solvers : with Applications in Incompressible Fluid Dynamics*, ch. 4 'Solution of discrete Stokes problems'. Oxford University Press. doi:10.1093/acprof:oso/9780199678792.001.0001

[McBAIN] McBain, G. D. (2016). Creeping convection in a horizontally heated ellipsoid. *Proceedings of the Twentieth Australasian Fluid Mechanics Conference*.

PYTHON MODULE INDEX

e