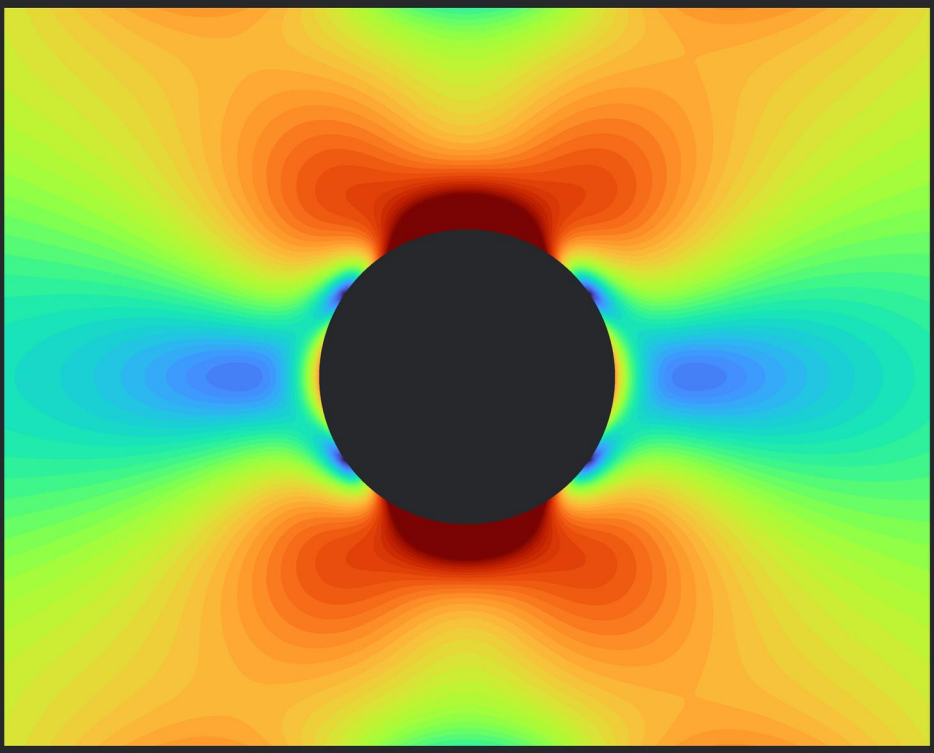

Von Mises stress concentrating at a hole in a loaded plate — linear elasticity, verified in DOLFINx.

Verified Finite Elements with DOLFINx

Solving PDEs with the FEniCSx Stack, Checked by Manufactured Solutions

Alexios Zervos

ODIN PRESS

Verified Finite Elements with DOLFINx

Paperback version, First edition: 2026.
ISBN: forthcoming (Bowker allocation pending)
Trim Size: 6×9 inch. Paper color: White.

Published by Odin Press.
`https://odinpress.org`
This book was typeset in LaTeX by Odin Press from a manuscript submitted by the author.

Printed and Bound in the United States of America.

9 8 7 6 5 4 3 2 1 0

Preface

A finite-element simulation you cannot verify is an opinion, not a result. I have believed this for forty years, and nothing in the migration of the finite-element community onto DOLFINx has given me cause to soften it. A new solver, a cleaner library, an assembly kernel far faster than the one before — these are welcome, and this book is built on them. But a new tool is exactly when a discipline's verification habits lapse: the tool is exciting and the checking is not, and a smooth-looking stress field asks no questions of the person who produced it. This book exists so the move to DOLFINx does not cost us the one habit that separates a computed result from a guess: forcing a solver to reproduce an answer we already know before we trust one we do not.

The gap this book fills is not in the methods: the finite-element method is old, its theory settled, DOLFINx implements it well. The gap is that the methods are almost always taught as if producing a field were the end of the work, when it is only the beginning. A textbook shows the weak form, assembles the matrix, and plots the solution; it rarely measures whether that solution converges at the order the theory promised, and almost never for every method it presents. The commercial packages are worse, not because their elements are wrong but because you cannot audit them — I have watched two disagree by fifteen percent on the same thermal-stress problem, neither willing to show me its quadrature. An unauditable result is untrustworthy, however professional the picture. What has been missing is a book that solves real problems with a modern open stack and verifies every one of them, in the open, by a method the reader can run himself.

So that is what this book does, the same way throughout: every method is verified before it is trusted. Where a problem has a closed-form solution — a Lamé cylinder, a Timoshenko bimetallic strip — the computed answer is matched against it. Where it does not, and most real geometries do not, the code is verified by the method of manufactured solutions: cook a

solution, derive the forcing it satisfies, and measure whether the error falls at the a-priori rate the theory promises. The convergence table goes on the page: I do not report "good agreement," I report the slope and let it make the argument. Where neither a closed form nor a manufactured solution settles the matter, the physics does — a reaction-force balance that must sum to zero, a peak stress that must stop moving as the mesh refines, the same answer on one processor and on eight. The book is candid, too, about where the method quietly fails: the volumetric locking that stiffens a low-order element to nonsense without warning, the checkerboard pressure of an inf-sup pair chosen by habit, the geometry error that hides in a functional rather than a solution slope. Every figure is produced by one self-contained file against a pinned library version; a figure you cannot regenerate is an anecdote, and there are none here.

The book is arranged as an ascent. The opening chapters build the discipline: the weak form and the assembly it drives, the method of manufactured solutions and the order it measures, the boundary conditions that must be imposed exactly, the unstructured meshes that come from a real CAD pipeline, and the elements whose interpolation error sets every rate that follows. The middle chapters put that discipline to work on the physics: linear elasticity and the locking that manufactured solutions expose, time-dependent heat and the two rates a transient carries, the Stokes saddle point and the inf-sup condition that governs it, and the nonlinear problems whose Newton iteration converges quadratically only when the Jacobian is exact. The computational chapters make the whole thing scale: the multigrid solver that cuts the cost without touching the answer, the parallelism whose rank-independence is itself a verification check, and the reproducible workflow that turns a claim into a test anyone can rerun. The book then spends all of it at once on a single real part — an irregular steel wrench, meshed at its fillets, heated until its steel softens and loaded until it yields, verified by closed form where one exists and by convergence and conservation where none does, and reported honestly down to the safety factor below one that says the red-hot part cannot take the load.

I came to DOLFINx late and with one instinct: never trust a solver you cannot force to reproduce an answer you already know. This book is that instinct worked out in full, on a tool worth trusting once you have made it prove the order first.

Αλέξιος Ζερβός

Contents

Chapter 1

The Weak Form and Your First Solve: Poisson from Space to Solution

A finite-element program that produces a smooth, colourful picture has told you nothing you can trust. This book is built on the opposite discipline: a solver earns belief only when it reproduces an answer you already knew, at the rate the theory promised, and it earns that belief once per method and never permanently. The engine that supplies the known answers is the method of manufactured solutions, and the yardstick is the measured order of convergence — the slope of error against mesh size, read off a table and reported as a number. Every chapter that follows solves a partial differential equation with DOLFINx and the FEniCSx stack [1], and every chapter verifies the solve before it believes it.

The Poisson problem is where the whole apparatus assembles for the first time. It is the simplest elliptic equation with a genuine boundary-value structure, it is symmetric and positive definite so its linear algebra is the friendliest we will meet, and it has manufactured solutions of every smoothness we could want. That makes it the right place to lay the six pieces the rest of the book reuses: the weak form the finite-element method actually solves, the finite-dimensional space that discretizes it, the Dirichlet condition that pins the solution to its data, the sparse system that assembly produces, the PETSc Krylov solver that inverts it [2], and the manufactured-solution check that certifies the result. This chapter builds

each piece, proves the two theorems that make the target well posed and its discretization quasi-optimal, and closes with the convergence study that turns the pipeline from an assertion into a measurement. When the L^2 error falls by four and the H^1 error by two each time the mesh is halved, the solver has passed its first verification — and only then is it a result rather than an opinion.

1.1 The Poisson Problem and Its Weak Form

Let $\Omega \subset \mathbb{R}^2$ be a bounded domain with Lipschitz boundary $\partial\Omega$, and let f be a given source and g given boundary data. The Poisson problem asks for a scalar field whose negative Laplacian matches f inside Ω while the field matches g on the boundary. This is the classical statement, and it is the statement the finite-element method does *not* solve; it solves a reformulation that trades one derivative of the unknown for one derivative of a test function, and that trade is the entire reason the method works on the rough functions a computer can represent.

Definition 1.1 (Strong form of the Poisson problem)**.** Given $f : \Omega \to \mathbb{R}$ and $g : \partial\Omega \to \mathbb{R}$, the strong form seeks $u : \overline{\Omega} \to \mathbb{R}$ with

$$-\nabla \cdot (\nabla u) = f \quad \text{in } \Omega, \qquad u = g \quad \text{on } \partial\Omega. \quad \text{on } \partial\Omega. \tag{1.1}$$

A classical solution must be twice continuously differentiable, so that the Laplacian $\nabla \cdot (\nabla u)$ is defined pointwise.

The regularity demand of the strong form is the problem. A piecewise-linear function on a triangular mesh — the cheapest thing a finite-element code can build — is not twice differentiable; its second derivative is a sum of Dirac masses on the element edges. If we insist on the strong form, the discrete solution is illegal before we begin. The weak form removes the demand by integrating against a test function and moving one derivative across the integral.

To state it we need the spaces the derivatives are measured in. The natural space is not the space of smooth functions but the Sobolev space of functions whose first derivatives are square-integrable [3].

Definition 1.2 (Sobolev spaces H^1 and H^1_0)**.** The space $H^1(\Omega)$ is the set of functions $v \in L^2(\Omega)$ whose weak gradient ∇v also lies in $L^2(\Omega)$, normed by

$$\|v\|^2_{H^1(\Omega)} = \int_\Omega \left(v^2 + |\nabla v|^2\right) dx. \tag{1.2}$$

The subspace $H^1_0(\Omega)$ is the closure in this norm of the smooth functions with compact support in Ω; its members vanish on $\partial\Omega$ in the trace sense. The trial space for the Poisson problem is H^1, and the test space is H^1_0.

The reason H^1 is exactly right, and not H^2 or C^2, is that it is the largest space on which the weak form's bilinear form is finite and coercive. That is a claim we will prove, not assert. First the weak form itself.

Multiply the strong equation by a test function $v \in H^1_0(\Omega)$ and integrate over Ω. The left side carries a second derivative, which the divergence theorem moves onto v:

$$-\int_\Omega \nabla \cdot (\nabla u)\, v\, dx = \int_\Omega \nabla u \cdot \nabla v\, dx - \int_{\partial\Omega} (\nabla u \cdot n)\, v\, ds. \tag{1.3}$$

Because v vanishes on $\partial\Omega$, the boundary integral is zero, and one derivative has crossed from u to v. What remains is the weak form.

Theorem 1.3 (Weak form of the Poisson problem)**.** A sufficiently smooth solution of the strong form satisfies, and is characterized by,

$$a(u,v) := \int_\Omega \nabla u \cdot \nabla v\, dx = \int_\Omega f v\, dx =: L(v) \quad \text{for all } v \in H^1_0(\Omega), \tag{1.4}$$

where a is a bilinear form on $H^1 \times H^1_0$ and L is a linear form on H^1_0. This bilinear-and-linear pair is the object a finite-element library assembles [4].

Proof. Let u solve the strong form and let $v \in H^1_0(\Omega)$. Multiplying $-\nabla \cdot (\nabla u) = f$ by v and integrating gives $-\int_\Omega \nabla\cdot(\nabla u)\, v\, dx = \int_\Omega f v\, dx$. Applying the integration-by-parts identity above and using $v|_{\partial\Omega} = 0$ discards the boundary term, leaving $a(u,v) = L(v)$. Conversely, suppose $u \in H^2(\Omega)$ satisfies $a(u,v) = L(v)$ for every $v \in H^1_0$. Integrating the gradient term back by parts returns $\int_\Omega (-\nabla \cdot (\nabla u) - f) v\, dx = 0$ for all $v \in H^1_0$. Since the smooth compactly supported functions are dense in $L^2(\Omega)$ and the bracket lies in L^2, the fundamental lemma of the calculus of variations forces $-\nabla \cdot (\nabla u) = f$ almost everywhere, recovering the strong equation. The two formulations therefore coincide on H^2. □

Bilinearity of a and linearity of L are immediate from the linearity of the integral and the gradient. What is not immediate — and what guarantees

Chapter 3

Boundary Conditions, Done Right: Dirichlet, Neumann, and Robin, Each Verified

Chapter 2 verified a solver on a problem whose boundary was pinned everywhere to a known value. Most problems are not so obliging: a heat sink holds a temperature on one face, a flux crosses another, and a convective law couples value and flux on a third. Each of those is a boundary condition of a different kind, each enters the finite-element method by a different mechanism, and each is a separate opportunity to be wrong in a way no picture reveals. This chapter treats the three canonical conditions — Dirichlet, Neumann, Robin — derives where each one enters the weak form, implements each in DOLFINx, and, following the discipline of the previous chapter, verifies each by a manufactured solution before trusting it.

The organizing idea is a single act of integration by parts. Moving one derivative off the trial function and onto the test function produces a boundary integral, and every boundary condition is a statement about that integral. A Dirichlet condition constrains the value and is imposed on the function space, so the boundary integral never appears; a Neumann condition prescribes the flux and *is* the boundary integral, added to the linear form; a Robin condition mixes the two and contributes to both forms. That

dichotomy — essential conditions on the space, natural conditions in the form — is the whole chapter in one sentence, and the sections that follow make it precise, prove that each implementation is well posed, and measure the order each one delivers. The deliverable is every condition implemented and individually verified, a mixed Dirichlet–Neumann–Robin problem confirmed to the a-priori order, and a missing boundary term caught as a collapsed order and a corrupted error field.

3.1 Essential and Natural: The Two Kinds of Boundary Condition

A boundary condition is a constraint the solution must satisfy on $\partial\Omega$, but the finite-element method sees two structurally different kinds, and the difference is not a matter of taste — it is dictated by where the condition can be enforced. One kind constrains the solution's value and can only be imposed on the space of admissible functions; the other constrains the solution's flux and can only enter through the weak form. Which is which falls out of integration by parts.

Definition 3.1 (Essential boundary condition)**.** An essential boundary condition constrains the value of the solution on the boundary, $u = g$ on Γ_D. It is imposed directly on the trial and test spaces — the trial functions are required to take the value g and the test functions to vanish — and never appears as a term in the weak form.

Definition 3.2 (Natural boundary condition)**.** A natural boundary condition constrains the flux of the solution — its outward normal derivative — on the boundary. It enters through a boundary integral in the weak form rather than through the space, so the trial and test spaces are unchanged; UFL represents it as an integral over the boundary measure [8].

Why the two kinds are handled so differently is not a convention but a consequence of the weak form's structure. The boundary integral that carries the natural data appears only after integration by parts, and it is worth deriving explicitly because every later section substitutes into it.

Theorem 3.3 (The boundary term)**.** Integrating the Poisson operator by parts against a test function v produces

$$\int_\Omega \left(-\nabla \cdot \nabla u \right) v \, dx = \int_\Omega \nabla u \cdot \nabla v \, dx - \int_{\partial\Omega} (\nabla u \cdot n) \, v \, ds, \tag{3.1}$$

so the weak form carries a boundary integral of the normal derivative $\partial_n u := \nabla u \cdot n$ times the test function. This boundary term is the exact location where natural conditions are substituted.

Proof. Apply the divergence theorem to the vector field $v\,\nabla u$: $\int_\Omega \nabla \cdot (v\,\nabla u)\,dx = \int_{\partial\Omega} v\,\nabla u \cdot n\,ds$. Expanding the divergence, $\nabla \cdot (v\,\nabla u) = \nabla v \cdot \nabla u + v\,\nabla \cdot \nabla u$. Substituting and rearranging, $\int_\Omega v\,\nabla \cdot \nabla u\,dx = \int_{\partial\Omega} v\,\partial_n u\,ds - \int_\Omega \nabla u \cdot \nabla v\,dx$. Negating both sides gives the claim. The manipulation requires $u \in H^2$ for a classical reading, but the identity extends to H^1 with the normal derivative interpreted in the weak trace sense [7]. □

The boundary term is the fork in the road. If we constrain v to vanish on part of the boundary, the term drops there and the value of u must be supplied some other way — on the space. If we leave v free and substitute a known value for $\partial_n u$, the term survives and carries the flux data into the form. That is the dichotomy, stated as a proposition because it governs every implementation choice in the chapter.

Proposition 3.4 (The essential/natural dichotomy)**.** Every boundary condition for the Poisson problem is handled in exactly one of two ways: an essential condition shrinks the trial space — the test space loses the corresponding boundary degrees of freedom, so the boundary term vanishes there — while a natural condition modifies the bilinear or linear form by substituting known flux data into the boundary term. The two mechanisms are exclusive on any given piece of boundary.

Proof. On a piece of boundary where $u = g$ is imposed essentially, the test space is H_0^1 there, so $v = 0$ and the boundary integral $\int (\partial_n u) v\,ds$ vanishes identically; the value g is carried by the trial space, not the form. On a piece where the flux $\partial_n u$ is prescribed, v is unconstrained, the boundary integral does not vanish, and substituting the known $\partial_n u$ moves the data into the form. A single boundary piece cannot be both: prescribing $v = 0$ there kills the integral that a natural condition needs, and leaving v free there leaves the value g unenforced. Hence the two mechanisms partition the boundary [13]. □

Both mechanisms require the boundary to be regular enough that a value and a normal flux are defined at all — a mild condition met by every polygonal mesh.

A finite-element simulation you cannot verify is an opinion, not a result. This book solves real PDEs with DOLFINx and the FEniCSx stack and verifies every method it presents — in the open, by a test the reader can run — because verification comes before validation, and a slope you can measure beats a picture you can only admire.

The code is the argument, and all of it is here. Every figure comes from one self-contained Python program — the weak form in UFL, the mesh in gmsh, the system through PETSc, run in parallel over MPI — each printed in the book, runnable against a pinned stack. Newton converges quadratically on an automatic Jacobian; a Stokes flow stays stable on an inf-sup pair; a steel wrench is solved from CAD as a coupled thermo-mechanical system. Nothing is a black box, nothing left as an exercise.

Where a problem has a closed form, the answer is matched against it; where it does not, manufactured solutions measure the error against the order the theory promises. The book is candid about where finite elements quietly fail — volumetric locking, checkerboard pressure, geometry error hiding in a functional — and names the test that catches each. It builds to one verified analysis: a steel wrench, heated until its steel softens and loaded until it yields, reported down to the safety factor below one.