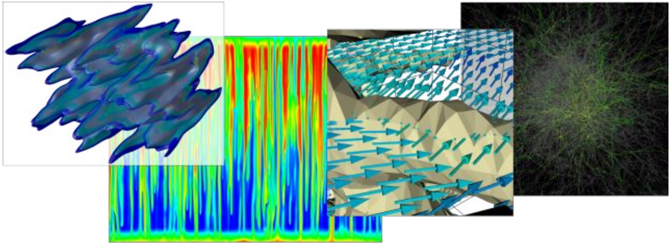
DUNE PDELab Tutorial 01

Conforming FEM for a Nonlinear Poisson Equation



Speaker:

Peter Bastian
IWR
Heidelberg University

Motivation

- Newton Method instead of StationaryLinearProblem Solver
- numerical Jacobian

- other FiniteElementMaps
- Numerical quadrature

This tutorial extends on tutorial 00 by

1) Solving a **nonlinear** stationary PDE
2) Using conforming finite element spaces of **arbitrary order**
3) Using **different types of (conforming) meshes** (simplicial and cubed)
4) Using **different types of boundary conditions**

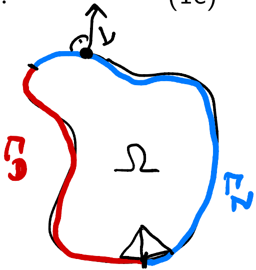
use different grid managers and FiniteElementMaps.

↪ lambda_boundary method in local operator

PDE Problem

We consider the problem

$$
\begin{aligned}
-\Delta u + q(u) &= f && \text{in } \Omega, && \text{(1a)}\\
u &= g && \text{on } \Gamma_D \subseteq \partial\Omega, && \text{(1b)}\\
-\nabla u \cdot \nu &= j && \text{on } \Gamma_N = \partial\Omega \setminus \Gamma_D. && \text{(1c)}
\end{aligned}
$$



- $q : \mathbb{R} \to \mathbb{R}$ a nonlinear function
- $f : \Omega \to \mathbb{R}$ the source term
- $g : \Omega \to \mathbb{R}$ a function for Dirichlet boundary conditions on Γ_D
- $j : \Gamma_N \to \mathbb{R}$ a function for Neumann (flux) boundary conditions
- ν: unit outer normal to the domain

Weak Formulation

now reads as follows:

$$\text{Find } u \in U \text{ s.t.:} \quad r^{\text{NLP}}(u, v) = 0 \quad \forall v \in V, \tag{2}$$

with the continuous residual form

$$r^{\text{NLP}}(u, v) = \int_\Omega \nabla u \cdot \nabla v + (q(u) - f) v \, dx + \int_{\Gamma_N} jv \, ds$$

and the function spaces

- $U = \{v \in H^1(\Omega) : \text{“}v = g\text{” on } \Gamma_D\}$ (affine space)
- $V = \{v \in H^1(\Omega) : \text{“}v = 0\text{” on } \Gamma_D\}$

We **assume** that this problem has a unique solution

Algebraic Problem

→ Solve weak formulation in finite-dimensional setting

$$U_h = \text{span}\{\phi_1, \ldots, \phi_n\} + u_{h,g}, \quad V_h = \text{span}\{\phi_1, \ldots, \phi_n\}$$

Expanding $u_h = u_{h,g} + \sum_{j=1}^{n} (z)_j \phi_j$ results in an algebraic equation for $z \in \mathbb{R}^n$:

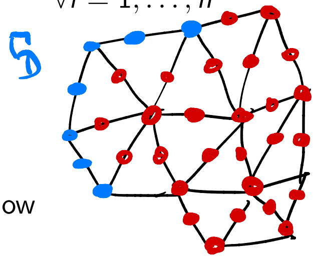
$$\text{Find } u_h \in U_h \text{ s.t.:} \qquad r(u_h, v) = 0 \qquad \forall v \in V_h$$

$$\Leftrightarrow \qquad r\left(u_{h,g} + \sum_{j=1}^{n} (z)_j \phi_j, \phi_i\right) = 0 \qquad \forall i = 1, \ldots, n$$

$$\Leftrightarrow \qquad R(z) = 0,$$

with $R : \mathbb{R}^n \to \mathbb{R}^n$ and $R_i(z) = R\left(u_{h,g} + \sum_{j=1}^{n} (z)_j \phi_j, \phi_i\right)$

Note: We remark on the realization of Dirichlet conditions below



Solution of Algebraic Problem

Use *iterative* methods to solve $R(z) = 0$. Fixed point iteration reads:

$$z^{(k+1)} = G(z^{(k)}) = z^{(k)} - \lambda^k W(z^{(k)})R(z^{(k)}). \tag{3}$$

$z = G(z)$

- $\lambda^k \in \mathbb{R}$ is a damping factor
- $W(z^{(k)})$ is a preconditioner matrix, e.g. in Newton's method one uses

$$W(z^{(k)}) = (J(z^{(k)}))^{-1} \quad \text{where } (J(z^{(k)}))_{i,j} = \frac{\partial R_i}{\partial z_j}(z^{(k)})$$

← „linearization point"

  i.e. one needs to solve $J\left(z^{(k)}\right) w = R(z^{(k)})$ per iteration

The following algorithmic building blocks are required:

i) residual evaluation $R(z)$,

ii) Jacobian evaluation $J(z)$ (or an approximation of it), ← inexact Newton.

iii) alternatively: matrix-free Jacobian application $J(z)w$ (or an approximation).

Note on Matrix-free Evaluation

Nonlinear case:

$$(J(z)w)_i = \sum_{j=1}^{n} (J(z))_{i,j}(w)_j = \sum_{j=1}^{n} \frac{\partial}{\partial z_j} \overbrace{r\left(u_{h,g} + \sum_{l=1}^{n} (z)_l \phi_l, \psi_i\right)}^{R_i(z)} (w)_j.$$

Linear case: $r(u, v) = a(u, v) - l(v)$, a BLF, l LF

$$(J(z)w)_i = \sum_{j=1}^{n} \frac{\partial}{\partial z_j} r\left(u_{h,g} + \sum_{l=1}^{n} (z)_l \phi_l, \psi_i\right)(w)_j$$

$$= \sum_{j=1}^{n} \frac{\partial}{\partial z_j} \left(a\left(u_{h,g} + \sum_{l=1}^{n} (z)_l \phi_l, \psi_i\right) - l(\psi_i)\right)(w)_j$$

$$= \sum_{j=1}^{n} \frac{\partial}{\partial z_j} \left(\sum_{l=1}^{n} (z)_l \underbrace{a(\phi_l, \psi_i)}_{(A)_{i,j}}\right)(w)_j = \sum_{j=1}^{n} a(\phi_j, \psi_i)(w)_j = a\left(\underbrace{\sum_{j=1}^{n} (w)_j \phi_j}_{w_h \in U_h}, \psi_i\right)$$

$$= (Aw)_i$$

Recall Finite Element Mesh Notation

i) Ordered sets of vertices and elements:

$$\mathcal{X}_h = \{x_1, \dots, x_N\}, \quad \mathcal{T}_h = \{T_1, \dots, T_M\}$$

ii) Partitioning of vertex index set $\mathcal{I}_h = \{1, \dots, N\}$ into $\mathcal{I}_h = \mathcal{I}_h^{int} \cup \mathcal{I}_h^{\partial\Omega}$:

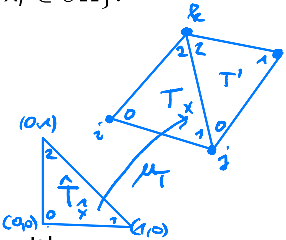
$$\mathcal{I}_h^{int} = \{i \in \mathcal{I}_h \,:\, x_i \in \Omega\}, \quad \mathcal{I}_h^{\partial\Omega} = \{i \in \mathcal{I}_h \,:\, x_i \in \partial\Omega\}.$$

iii) For every element $T \in \mathcal{T}_h$ a local-to-global map

$$g_T : \{0, \dots, n_T - 1\} \to \mathcal{I}_h$$

iv) For every element $T \in \mathcal{T}_h$ an element transformation map

$$\mu_T : \hat{T} \to T$$



μ_T is differentiable with invertible Jacobian and consistent with g_T:

$$\forall i \in \{0, \dots, n_T - 1\} : \mu_T(\hat{x}_i) = x_{g_T(i)}$$

Conforming Finite Element Space

with **polynomial degree** k in **dimension** d on **mesh** $\mathcal{T}_h$:

$$V_h^{k,d}(\mathcal{T}_h) = \left\{ v \in C^0(\overline{\Omega}) \,:\, \forall T \in \mathcal{T}_h : v|_T = \hat{p}_T \circ \mu_T^{-1} \wedge \hat{p}_{\hat{T}} \in \mathbb{P}_{\hat{T}}^{k,d} \right\}$$

where the multivariate polynomials $\mathbb{P}_T^{k,d}$ depend on element type:

$$\mathbb{P}_{\hat{T}}^{k,d} = \begin{cases} \left\{ p \,:\, p(x_1,\dots,x_d) = \sum\limits_{0 \le \|\alpha\|_1 \le k} c_\alpha x_1^{\alpha_1} \cdot \ldots \cdot x_d^{\alpha_d} \right\} & \hat{T} = \hat{S} \text{ (simplex)}, \\ \left\{ p \,:\, p(x_1,\dots,x_d) = \sum\limits_{0 \le \|\alpha\|_\infty \le k} c_\alpha x_1^{\alpha_1} \cdot \ldots \cdot x_d^{\alpha_d} \right\} & \hat{T} = \hat{C} \text{ (cube)} \end{cases}$$

The dimension of $\mathbb{P}_T^{k,d}$ is:

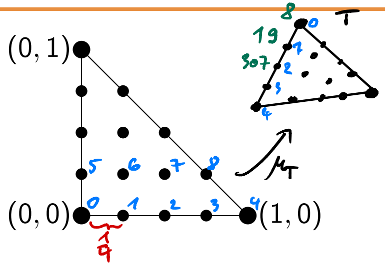
$$n_{\hat{C}}^{k,d} = (k+1)^d \text{ (cube)}, \quad n_{\hat{S}}^{k,d} = \begin{cases} 1 & k = 0 \vee d = 0 \\ \sum\limits_{i=0}^{k} n_{\hat{S}}^{i,d-1} & \text{else} \end{cases} \text{ (simplex)}$$

Local Lagrange Basis

$g_T(0) = 8$

$g_T(1) = 19$

$g_T(2) = 307$



polynomial degree 4, $n_{\hat{T}}^{4,2} = 15$

Lagrange points and polynomials (shape functions) on $\hat{T}$:

$$L_{\hat{T}} = \left\{ \hat{x}_0^{\hat{T}}, \ldots, \hat{x}_{n_{\hat{T}}^{k,d}-1}^{\hat{T}} \right\}, \quad P_{\hat{T}} = \left\{ \hat{p}_0^{\hat{T}}, \ldots, \hat{p}_{n_{\hat{T}}^{k,d}-1}^{\hat{T}} \right\}$$

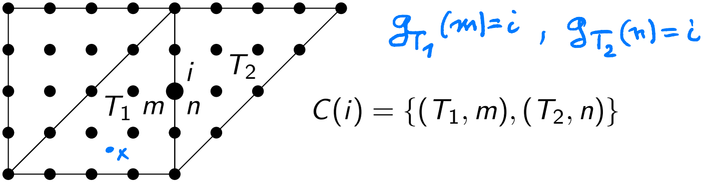
such that

$$\hat{p}_i^{\hat{T}}(\hat{x}_j^{\hat{T}}) = \delta_{i,j} = \begin{cases} 1 & i = j \\ 0 & \text{else} \end{cases}$$

Extend local to global map:

$$g_T : \{0, \ldots, n_{\hat{T}}^{k,d} - 1\} \to \mathcal{I}_h^{k,d} = \left\{ 0, \ldots, \dim V_h^{k,d}(\mathcal{T}_h) - 1 \right\}$$

Global Lagrange Basis



$$C(i) = \{(T_1, m), (T_2, n)\}$$

Define inversion of the local-to-global map:

$$C(i) = \{(T, m) \in \mathcal{T}_h \times \mathbb{N} \, : \, g_T(m) = i\}$$

then the global Lagrange basis functions are

$$\phi_i(x) = \begin{cases} \hat{p}_m^{\hat{T}}(\mu_T^{-1}(x)) & x \in T \wedge g_T(m) = i \\ 0 & \text{else} \end{cases} , \quad i \in \mathcal{I}_h^{k,d}.$$

corresponding to the global Lagrange points

$$\mathcal{X}_h^{k,d} = \left\{ x_i \in \overline{\Omega} \, : \, x_i = \mu_T(\hat{x}_m^{\hat{T}}) \wedge g_T(m) = i \right\}$$

Dirichlet Boundary Conditions I

Indices of Lagrange points on the Dirichlet boundary are:

$$\mathcal{I}_h^{D,k,d} = \left\{ i \in \mathcal{I}_h^{k,d} \; : \; x_i \in \mathcal{X}_h^{k,d} \cap \Gamma_D \right\}.$$

Then the test space with zero Dirichlet condition is:

$$V_{h,0}^{k,d}(\mathcal{T}_h) = \left\{ v \in V_h^{k,d}(\mathcal{T}_h) \; : \; v(x_i) = 0 \quad \forall i \in \mathcal{I}_h^{D,k,d} \right\}$$

For the trial space choose any extension

$$V_h^{k,d} \ni u_{h,g} = \sum_{i \in \mathcal{I}_h^{k,d}} g(x_i)\phi_i, \qquad \text{so } u_{h,g}(x_i) = g(x_i) \; \forall i \in \mathcal{I}_h^{D,k,d}$$

Then the trial ~~space~~ is

$$U_h^{k,d}(\mathcal{T}_h) = \left\{ u \in V_h^{k,d}(\mathcal{T}_h) \; : \; u = u_{h,g} + w \wedge w \in V_{h,0}^{k,d}(\mathcal{T}_h) \right\} = u_{h,g} + V_{h,0}^{k,d}(\mathcal{T}_h)$$

Dirichlet Boundary Conditions II

There are different options to realize Dirichlet conditions in practice:

1. Elimination of all Dirichlet conditions from the algebraic systems, i.e.

$$R : \mathbb{R}^{n_0} \to \mathbb{R}^{n_0}, \qquad n_0 = \dim\left(V_{h,0}^{k,d}(\mathcal{T}_h)\right), \qquad R_i(z) = R\left(u_{h,g} + \sum_{j=1}^{n} (z)_j \phi_j, \psi_i\right)$$

2. Keep degrees of freedom at Dirichlet boundary in the algebraic system, i.e.

$$R : \mathbb{R}^{n} \to \mathbb{R}^{n}, \qquad n = \dim\left(V_{h}^{k,d}(\mathcal{T}_h)\right) \qquad n > n_0$$

   with *additional* equations

$$z_i = g(x_i), \qquad \forall i \in \mathcal{I}_h^{D,k,d}$$

   This approach is used in PDELab

3. Nitsche's method: Essential boundary conditions are *not* built into the function space, instead certain terms are added to the weak formulation

General Constraints

Dirichlet boundary conditions are a special case of the following

Task: Given $V_h = \text{span}\,\{\phi_j : j \in J_h = \{1, \ldots, n\}\}$ construct a subspace $\tilde{V}_h \subseteq V_h$

This is how it is done in PDELab:

1) Assume $V_h = \text{span}\{\phi_i : i \in J_h\}$
2) Select a subset of indices $\tilde{J}_h \subset J_h$, $\dim(\tilde{J}_h)$ is the dimension of the subspace $\tilde{V}_h$
3) Set $\tilde{V}_h = \text{span}\left\{\tilde{\phi}_j : j \in \tilde{J}_h\right\}$, where the new basis functions have the form

$$\tilde{\phi}_j = \phi_j + \sum_{l \in J_h \setminus \tilde{J}_h} (B)_{j,l}\phi_l, \qquad \forall j \in \tilde{J}_h.$$

Any such subspace is thus characterized by $C = (\tilde{J}_h, B)$
PDELab implements general constraints in this way, e.g. for so-called hanging nodes

Element-wise Computations

Now return to the evaluation of the residual form, which is element-wise

$$r^{\mathrm{NLP}}(u,v) = \sum_{T\in\mathcal{T}_h} \alpha_T^V(u,v) + \sum_{T\in\mathcal{T}_h} \lambda_T^V(v) + \sum_{F\in\mathcal{F}_h^{\partial\Omega}} \lambda_F^B(v)$$

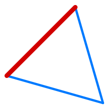
with

α-Volume λ-volume λ-boundary

$$\alpha_T^V(u,v) = \int_T \nabla u \cdot \nabla v + q(u)v\,dx, \quad \lambda_T^V(v) = -\int_T fv\,dx, \quad \lambda_F^B(v) = \int_{F\cap\Gamma_N} jv\,ds$$

$\mathcal{F}_h^{\partial\Omega}$: intersections of elements with the domain boundary, i.e.

$$F = T_F^- \cap \partial\Omega$$



with $T_F^- \in \mathcal{T}_h$ the “minus” element associated with F (see tutorial 2)

λ Volume Term

For any $(T, m) \in C(i)$ we obtain

$$\lambda_T^V(\phi_i) = -\int_T f\phi_i \, dx = -\int_{\hat{T}} f(\mu_T(\hat{x}))\hat{p}_m^{\hat{T}}(\hat{x})|\det J_{\mu_T}(\hat{x})| \, d\hat{x}.$$

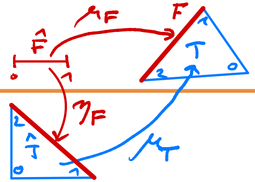
J_{μ_T} is the Jacobian of the element map μ_T

This integral is computed using numerical quadrature

Collect all contributions of T in a small vector:

$$(\mathcal{L}_T^V)_m = -\int_{\hat{T}} f(\mu_T(\hat{x}))\hat{p}_m^{\hat{T}}(\hat{x})|\det J_{\mu_T}(\hat{x})| \, d\hat{x}.$$

λ Boundary Term



For $F \in \mathcal{F}_h^{\partial\Omega}$ with $F \subseteq \Gamma_N$ and $(T_F^-, m) \in C(i)$ we obtain

$$\lambda_T^B(\phi_i) = \int_F j\,\phi_i\,ds = \int_{\hat{F}} j(\mu_F(\hat{x}))\hat{p}_m^{\hat{T}}(\eta_F(\hat{x}))\sqrt{|\det(J_{\mu_F}^T(\hat{x})J_{\mu_F}(\hat{x}))|}\,ds$$

The integration is more involved here because it is over a face:

- $\mu_F : \hat{F} \to F$ maps the reference element of the face to the face
- $\eta_F : \hat{F} \to \hat{T}_F^-$ maps reference element of the face to the reference element of T_F^-

Collect all contributions of F in a small vector:

$$(\mathcal{L}_T^B)_m = \int_{\hat{F}} j(\mu_F(\hat{x}))\hat{p}_m^{\hat{T}}(\eta_F(\hat{x}))\sqrt{|\det J_{\mu_T}^T(\hat{x})J_{\mu_T}(\hat{x})|}\,ds. \approx \sum_q I(q)\,w_q$$

Numerical quadrature is applied to compute the integral

α Volume Term

For any $(T, m) \in C(i)$ we get

$$
\begin{aligned}
\alpha_T^V(u_h, \phi_i) &= \int_T \nabla u \cdot \nabla \phi_i + q(u)\phi_i \, dx, \\
&= \int_T \sum_j (z)_j \left(\nabla \phi_j \cdot \nabla \phi_i\right) + q\left(\sum_j (z)_j \phi_j\right) \phi_i \, dx, \\
&= \int_{\hat{T}} \sum_n (z)_{g_T(n)} (J_{\mu_T}^{-T}(\hat{x}) \hat{\nabla} \hat{p}_n^{\hat{T}}(\hat{x})) \cdot (J_{\mu_T}^{-T}(\hat{x}) \hat{\nabla} \hat{p}_m^{\hat{T}}(\hat{x})) \\
&\quad + q\left(\sum_n (z)_{g_T(n)} \hat{p}_n^{\hat{T}}(\hat{x})\right) \hat{p}_m^{\hat{T}}(\hat{x}) |\det J_{\mu_T}(\hat{x})| \, dx
\end{aligned}
$$

and again collect all contributions from T in a small vector $\mathcal{R}_T^V(R_T z)$

Putting It All Together

With these local contributions the evaluation of the algebraic residual is

$$R(z) = \sum_{T\in\mathcal{T}_h} R_T^T \mathcal{R}_T^V(R_T z) + \sum_{T\in\mathcal{T}_h} R_T^T \mathcal{L}_T^V + \sum_{F\in\mathcal{F}_h^{\partial\Omega}\cap\Gamma_N} R_T^T \mathcal{L}_F^B$$

α-volume λ-volume λ-boundary

where R_T is the "picking out" matrix of element T

The Jacobian of R is

$$(J(z))_{i,j} = \frac{\partial R_i}{\partial z_j}(z) = \sum_{(T,m,n):(T,m)\in C(i)\wedge(T,n)\in C(j)} \frac{\partial(\mathcal{R}_T^V)_m}{\partial z_n}(R_T z)$$

jacobian_volume

Note that:

i) Entries of the Jacobian can be computed element by element.

ii) The derivative is independent of the λ-terms

iii) Jacobian entries may be computed by numerical differentiation

Implementation Overview

1) `tutorial01.ini` holds parameters controlling the execution
2) `tutorial01.cc` includes the necessary C++, DUNE and PDELab header files; contains `main` function calling `driver`
3) Function `driver` in `driver.hh` instantiates the necessary PDELab classes for solving a nonlinear stationary problem and finally solves the problem.
4) File `nonlinearpoissonfem.hh` contains class `NonlinearPoissonFEM` realizing a PDELab local operator
5) File `problem.hh` contains a "parameter class" which encapsulates the user-definable part of the PDE problem
6) Finally, the tutorial provides some mesh files.