RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

PROGRAMME DE RECHERCHE
NUMÉRIQUE POUR L'EXASCALE

Feel++

Finite Element Embedded Library in C++ Modern Architecture for Exascale Computing

Version: 4c61980 (2025-12-10 14:52:27 +0100)

Christophe Prud'homme

January 20, 2026

Cemosis / IRMA – Université de Strasbourg – NumPEx

What is Feel++?

A Full Computational Pipeline

Modern C++ framework (C++20/C++23) for mono & multi-physics: advanced methods + toolboxes + MOR

Methods

- Continuous Galerkin (CG)
- Discontinuous Galerkin (DG)
- Hybridized DG (HDG)
- Low to high-order (incl. geometry)
- Model order reduction

Capabilities

- 1D, 2D, 3D geometries
- MPI parallelization
- GPU portability (in progress)
- Multi-physics coupling
- Python bindings

Repository: `https://github.com/feelpp/feelpp` • **800+ tests** • **v0.111.0**

Feel++ in the NumPEx/Exa-MA Context

WP1 – Discretization & Geometry

- Low to high-order FE (incl. geom)
- Space/time adaptivity
- Multi-physics coupling

WP2 – Model Order Reduction

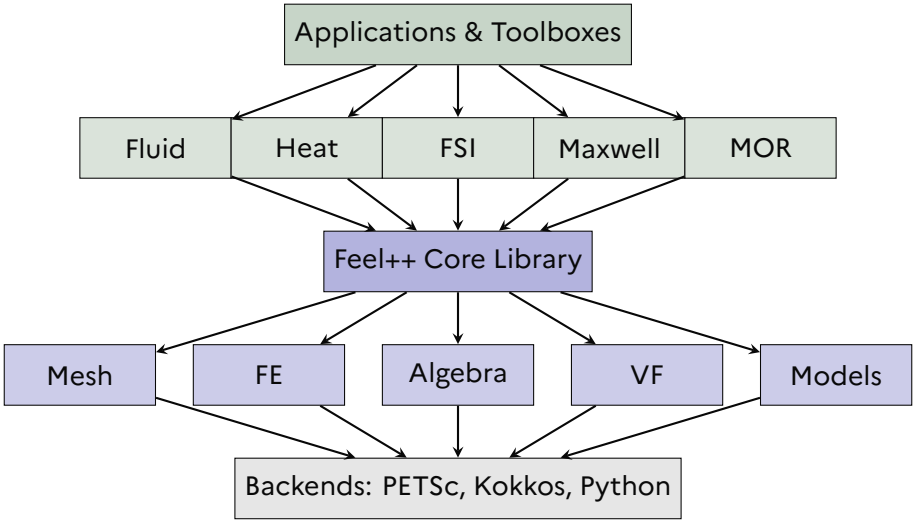
- Nonlinear Compressive RB (NLCRB)
- ML-accelerated coefficients
- C++20 MOR architecture

WP3 – Solvers & Preconditioners

- PETSc/SLEPc integration
- Block structure exposure
- Multi-physics coupling

WP4 (DA) & WP6 (UQ): GEIM/PBDW, EnKF, sensor placement, multi-query UQ

Architecture Overview



Recent Developments: C++20 Refactoring

Modern C++ Architecture (C++20 features, C++23 ready)

Major refactoring efforts leveraging C++20 features:

Runtime Order Selection:

- FE and geometry order at runtime
- C++20 concepts for type safety
- No code duplication
- Enables adaptive methods

Tensor Rank System:

- Compile-time classification
- Scalar/Vector/Matrix/Tensor3

MOR/ROM Refactoring:

- Explicit offline/online API
- Concepts-based contracts
- Pluggable hyper-reduction
- NLCRB (WP2), DA (WP4)

GPU Portability (2025–2026):

- Kokkos integration
- CUDA/HIP backends

C++20 Refactoring: Runtime Order Selection

Key Innovation

C++20 enables runtime-selectable FE and geometry order without code duplication or performance loss

Before (compile-time)

- Order fixed at compile time
- Code bloat for each order
- Hard to do *hp*-adaptivity

After (runtime)

- Order selected at runtime
- Single code path
- Enables *hp*-adaptivity, MOR

Applications: Adaptive methods • MOR basis construction • Multi-fidelity workflows

C++20 Concepts: Why It Matters

Core Insight

C++20 concepts replace implicit conventions with explicit, compiler-enforced contracts: **safer APIs, better errors, no virtual dispatch → GPU-ready design**

Before (C++14/17)

- Implicit template requirements
- Cryptic error messages
- SFINAE-based dispatch

After (C++20)

- Explicit concept definitions
- Clear, actionable errors
- Static polymorphism (GPU-ready)

Key feature: Runtime-selectable polynomial order without code duplication—critical for adaptive methods and MOR

MOR/ROM Architecture Refactoring

Motivation

Support NLCRB (WP2), GEIM/PBDW (WP4), and multi-query UQ (WP6) with clean architecture

Design Principles

- Explicit offline/online separation
- Concepts over conventions
- Pluggable hyper-reduction

First-Class Objects

- `ParameterSpace`
- `Model`, `BasisBuilder`
- `OnlineSolver`

Benefit: Predictable data-flow, cacheable operators, compile-time validation

Spack Integration for NumPEx

Modern Package Management

Feel++ is integrated into the NumPEx Spack ecosystem for reproducible HPC deployments

Features

- URL-based repositories (GitHub)
- OCI build caches (ghcr.io)
- CI/CD integration
- Reproducible HPC stacks

Dependencies:
Core: Boost, Eigen, PETSc, SLEPc
Mesh: Gmsh, CGAL
GPU: Kokkos; Python: pybind11

Physics Toolboxes

Pre-configured Multi-Physics Applications:

- **Fluid**: Navier-Stokes CFD
- **Heat**: Thermal analysis
- **Solid**: Linear/nonlinear mechanics
- **FSI**: Fluid-Structure Interaction
- **Maxwell**: Electromagnetics
- **Heat-Fluid**: Coupled thermal-flow
- **HDG**: Hybridized DG methods
- **Level-Set**: Interface tracking
- **Advection**: Transport problems
- **Coefficient-Form PDEs**

Design Pattern: Standard BCs • Pre-configured discretizations • Solver integration • ParaView export • JSON config

PyFeelpp: Full Library Access

Complete Python bindings via pybind11 for rapid prototyping and integration.

Components:

- `pyfeelpp`: Core library bindings
- `pyfeelpp-mor`: Model order reduction
- `pyfeelpp-toolboxes`: Physics applications

Use Cases:

- Rapid prototyping
- ML frameworks (PyTorch, TF)
- Jupyter notebooks
- Workflow automation

Current Work: Namespace package reorganization for better structure

Performance and Testing

Parallelization (Production)

- MPI via PETSc
- Domain decomposition
- Distributed mesh partitioning

GPU Portability (2025–2026)

- Kokkos backend integration
- CUDA/HIP targets
- Matrix-free operators

Testing Framework

- 800+ automated tests
- Unit + integration tests
- Regression testing

CI/CD: Ubuntu, Debian, Fedora • Multiple compilers • Spack builds

Funded Projects (2025-2030)

SAGE-HPC (NumPEx, 2026)

AI-driven multi-fidelity orchestration for Bayesian optimization. **Feel++**: ROM/MOR as surrogate.

ANR JNL-G (2026)

Digital twin for LNCMI high-field magnets. **Feel++**: ROM+UQ+ML+DA integration.

Impact on Feel++ Development:

- Clean, extensible MOR architecture
- Runtime order for complex geom
- ML methods with ROM
- Real-time surrogates
- Data assimilation

Roadmap and Priorities

2025 (Short-term)

- MOR refactoring (NLCRB, GEIM, PBDW)
- Stabilize Kokkos GPU backend
- Python bindings reorganization

2026–2027 (Medium-term)

- SAGE-HPC: AI-driven ROM
- ANR JNL-G: LNCMI digital twin
- ML-accelerated coefficients

2028–2030 (Long-term):

- Exascale-ready applications
- Production digital twins

What We Need

- Benchmark problems/datasets
- Interface specifications
- Co-development partners

Community and Resources

Documentation:

- Website & Docs: `https://docs.feelpp.org`
- API Reference: Doxygen

Development:

- GitHub: `https://github.com/feelpp/feelpp`
- Issues and discussions
- Slack: feelpp.slack.com

Distribution:

- Docker / Apptainer images
- Debian/Ubuntu packages
- Spack packages
- Source builds

Summary

Feel++: Full Computational Pipeline for Exascale

Mono & multi-physics • Advanced methods • C++20/C++23 • Exa-MA (WP1–WP6)

What We Offer

- Library + toolboxes + MOR + Python
- Concepts, type safety, GPU-ready
- MPI + Kokkos + matrix-free

Key Differentiators

- Runtime FE/geometry order
- C++20 concepts architecture
- Refactor MOR/ROM(in progress)

Active: SAGE-HPC (2026) • ANR JNL-G (2026) • 800+ tests, CI/CD, Spack

Breakouts: Collaboration Outcomes (Today)

Goal

Convert needs into **2–3 co-owned integration items per breakout**

Breakout Tracks

- **WP1–WP2**: Geom/FE/Multiphysics ↔ ROM/SciML
- **WP3**: Solver hooks + benchmarks
- **WP4–WP6**: DA/UQ pipelines

Output per Item

1. Minimal deliverable
2. Acceptance criteria
3. Owner + co-owners
4. GitHub issue created live

Exa-MA App Ladder: mini-app → extended → demonstrator → proxy-app

Contribution: benchmark → spec → implementation → tests → co-maint

Questions?