NIMROD abstraction update

Jacob King (Tech-X),
Brian Cornille (U. Wisconsin-Madison)

NIMROD Team meeting
8/1/18

Overview of topics

- 4:30-4:50 Tour of the GitLab interface
- 4:50-5:00 Update on progress of porting the existing code
- 5:00-5:20 Interface considerations for H(curl) elements
- 5:20-5:30 Plans

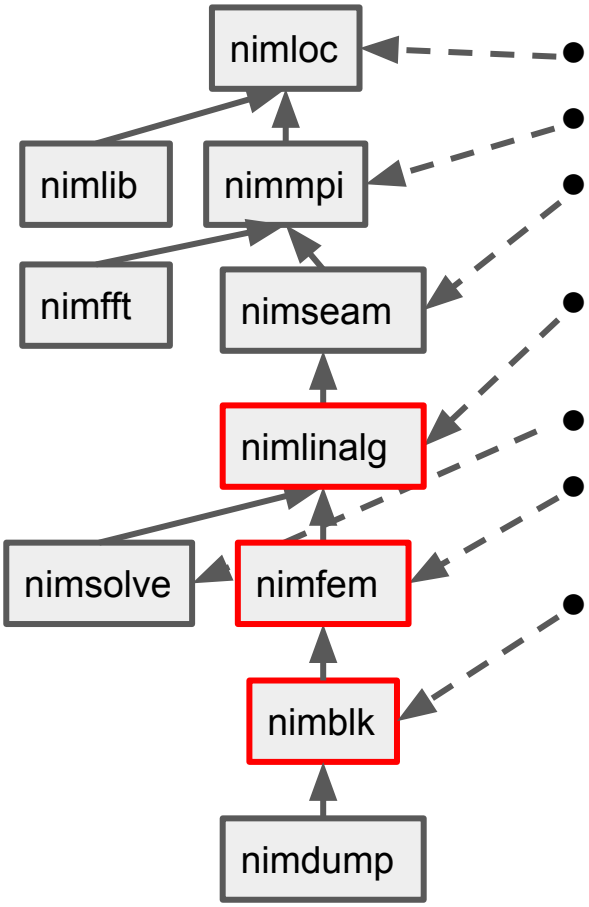
Abstract branch has been moved to GitLab

- https://gitlab.com/NIMRODteam/nimrod-abstract
- git vs svn: git is more powerful and verbose but has more pitfalls
- Highlights of interface:
  - View repository
  - Issue tracker
  - Merge requests (code review)
  - Continuous Integration (testing: unit, format, docs)
  - Wiki (integrated with FORD docs)
  - Snippets (useful code bits not in the repository)
- Hey! I don't have an access!
  - Create an account on GitLab
  - Email me (jking at txcorp.com) your account name
- Notes of interest (not GitLab related):
  - FORD docs are available on the nimrodteam.org internal website
    - Updated: when I feel like it
    - Source is available too with an extremely painful html checkout
  - Slack is used to coordinate the abstract brach development (join us!)
    - Benefit of being informal, open
    - Good medium for code development, not so good for science questions

Merge requests are opportunity for code review

- Tests (formatting, unit, docs) are run automatically
  - Does the new code pass the prior tests?
  - Is the new code tested?
- Provides interface to review/comment on changes
- Provides interface to require approval to merge
  - Ideal system: Invested, independent , senior developer signs off on changes
  - In practice: We're trying to be agile and move fast (but we still do some review)
- Discuss: Does the development (feature/enhancement/bug fix) belong in the trunk?

Seams, vectors and matrices are ported



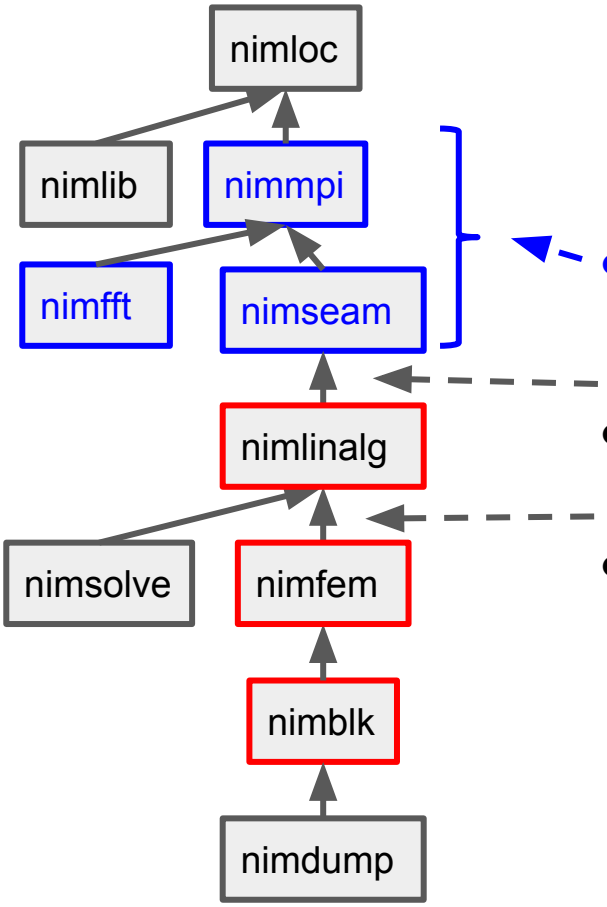
Layering by files (links to abstract base types):

- local.f
- mpi_serial/parallel.f, pardata.f (partial)
- edge.f, edge_type_mod.f, seam_storage.f
  - parallel.f (seam relevant part)
- vector.F90, matrix.F90
  - + instantiations
- iter_*
- nodal.F90, quadrature.F90, polynomials.F90
  - + instantiations
- gblock.F90, block_storage.F90, finite_element.f
  - + instantiations
  - parallel.f (block relevant part)

(it wouldn't be an abstraction talk without a flow chart like this!)

Abstract type

Layering rationale



- Basic parallel communication types
  - Blocks (seams) and Fourier components layers (FFTs)
- Seam load/unload routines are moved in the vector/matrix types in nimlinalg
- FEM types create appropriate type for their linear algebra space

Abstract type

Upcoming plans (ordering tentative)

- Integration test: Poisson test case with Natural and Dirichlet BCs
  - Can be parallel
  - Initialized from dump file which specifies concrete types
  - Good integration test and proof of abstract principle
- Assess performance
  - Implement performance timer
  - Test vectorization before porting mhd advance?
    - Pro: if reordering is needed, good to do it right away
    - Con: slows things down
- Enable H(curl) infrastructure (good to ensure general interfaces)
- Clean interfaces and finalize formatting and naming
- Port MHD advance/integration + run time diagnostics
- Port neutral fluid advance/integration
- Enable 1D infrastructure (good for quick tests: tearing, neutrals, etc.)
- Port regression tests
- Enable CK infrastructure, port CK advance/integration

We're not dead yet! Join us!

- Ultimately success will be judged by usage
- We're making good progress on a large effort, still have plenty to do
- Abstraction enables a faster path to development for future capabilities (time invested now can be recovered later)
  - Different function spaces (e.g. for H(curl), least-squares or Petrov-Galerkin)
  - Static condensation and flexibility for continuum kinetics
  - Vectorization and performance optimization through compartmentalization
  - Other capabilities?
- Another benefit: unit tests
  - make the code more robust during development

Help us appease The Knights Who Say Ni!

- The abstraction effort is large
- We're making good progress but still have plenty to do
- Abstraction enables a faster path to development for future capabilities (time invested now can be recovered later)
  - Different function spaces (e.g. for H(curl), least-squares or Petrov-Galerkin)
  - Static condensation and flexibility for continuum kinetics
  - Vectorization and performance optimization through compartmentalization
  - Other capabilities?
- Another benefit:unit tests
  - make the code more robust during development