Stabilization of POD-ROMs

David Wells

Virginia Tech/Rensselaer Polytechnic Institute

Wednesday, August 5, 2015

Overview

`deal.II`: A Powerful System for Model Reduction

Overview

`deal.II`: A Powerful System for Model Reduction

1. POD
2. POD with `deal.II`
3. Filtering and ROM
4. REG-ROMs
5. Conclusions & Future Work

Collaborators

Some of my collaborators:

Volker John, (WIAS), Traian Iliescu (VT), Swetlana Giere (WIAS), Zhu Wang (SC), Xuping Xie (VT)

A special thanks to Abner Salgado (UT) for writing `step-35`.

VT to RPI

Geometric Modeling Using Octree Encoding

DONALD MEAGHER*

Rensselaer Polytechnic Institute,
Troy, New York 12181

Received June 19, 1981

A geometric modeling technique called Octree Encoding is presented. Arbitrary 3-D objects can be represented to any specified resolution in a hierarchical 8-ary tree structure or "octree." Objects may be concave or convex, have holes (including interior holes), consist of disjoint parts, and possess sculptured (i.e., "free-form") surfaces. The memory required for representation and manipulation is on the order of the surface area of the object. A complexity metric is

Why `deal.II`?

1. ROMs are *usually* expressed as finite element methods

Why `deal.II`?

1. ROMs are *usually* expressed as finite element methods
2. Community is nice

Why `deal.II`?

1. ROMs are *usually* expressed as finite element methods
2. Community is nice
3. Great documentation

Why `deal.II`?

1. ROMs are *usually* expressed as finite element methods
2. Community is nice
3. Great documentation

```
[drwells@archway dealii-dev]$ cloc ./include
    378 text files.
    378 unique files.
    2 files ignored.

http://cloc.sourceforge.net v 1.64  T=1.42 s (264.3 files/s, 180140.8 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C/C++ Header                   375          34261         113105         108829
CMake                            1              4             23             18
-------------------------------------------------------------------------------
SUM:                           376          34265         113128         108847
-------------------------------------------------------------------------------
```

Why `deal.II`?

1. LAPACK support (`geev`, `getrf`, `getrs`)
2. HDF5 and XDMF support
3. C++11 support

The Navier-Stokes Equations

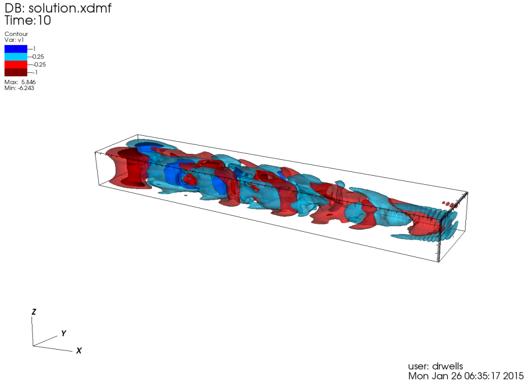
$$\begin{aligned} \vec{u}_t + \vec{u} \cdot \nabla \vec{u} - \frac{1}{Re} \Delta \vec{u} + \nabla p = 0, \\ \nabla \cdot \vec{u} = 0 \end{aligned} \tag{1}$$

1. Specified (parabolic) inflow
2. $\vec{u} \times \vec{n} = 0$ outflow
3. `deal.II` step 35 [1, 2]
4. Fractional step method
5. About $600,000$ DoFs, $Re = 100$

The Navier-Stokes Equations

Goal: Preserve large structures and phase portraits.

The Navier-Stokes Equations



y-velocity contours at $t = 10$. There is a circular cylinder near the inflow on the left.

Proper Orthogonal Decomposition (POD)

Given a set of data with high dimensionality, what is the best (under some norm) approximation to the data for a given rank r?

What are POD-derived basis functions?

Deriving POD basis functions is a linear procedure. Let Y denote the "snapshot" matrix [5] and $M = LL^T$ denote the mass matrix.

What are POD-derived basis functions?

Deriving POD basis functions is a linear procedure. Let Υ denote the "snapshot" matrix [5] and $M = LL^T$ denote the mass matrix.

$$ESV^T = SVD(L^T \Upsilon) \rightarrow \Phi = (L^T)^{-1} E \qquad (2)$$

What are POD-derived basis functions?

Deriving POD basis functions is a linear procedure. Let Y denote the "snapshot" matrix [5] and $M = LL^T$ denote the mass matrix.

$$ESV^T = SVD(L^T Y) \rightarrow \Phi = (L^T)^{-1} E \tag{2}$$

$$Y^T M Y v_i = \lambda_i v_i \rightarrow \varphi_i = \sum_{n=0}^{N-1} y_n v_i(n) \tag{3}$$

The Method of Snapshots

1. Does either the method of snapshots or the reduced order matrices suffer a loss of accuracy from inaccurate inner product calculations?

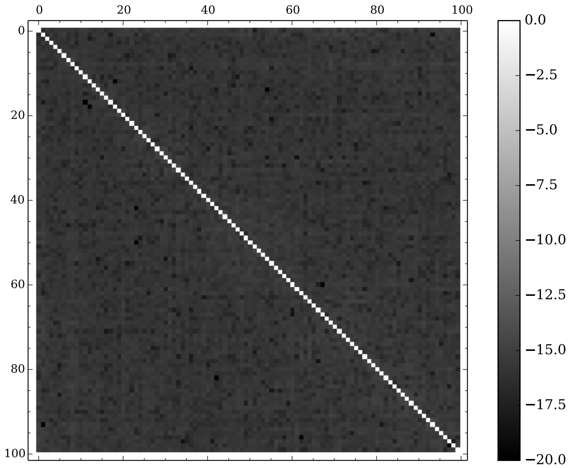
The Method of Snapshots

1. Does either the method of snapshots or the reduced order matrices suffer a loss of accuracy from inaccurate inner product calculations?
2. Do the POD vectors calculated by the method of snapshots recover the POD interpolation error equations?

From `vector.templates.h`:

```
// this is the main working loop for all vector sums using the templated
// operation above. it accumulates the sums using a block-wise summation
// algorithm with post-update. this blocked algorithm has been proposed in
// a similar form by Castaldo, Whaley and Chronopoulos (SIAM
// J. Sci. Comput. 31, 1156-1174, 2008) and we use the smallest possible
// block size, 2. Sometimes it is referred to as pairwise summation. The
// worst case error made by this algorithm is on the order O(eps *
// log2(vec_size)), whereas a naive summation is O(eps * vec_size). Even
```

The Method of Snapshots



Magnitudes of entries in the POD mass matrix.

Interpolation Errors

r_1	$\sum_{i=r_1}^{R-1} \sigma_i^2$	$\sum_{n=0}^{R-1} \left\| \vec{u}_n - \sum_{i=0}^{r_1-1} \langle \vec{u}_n, \vec{\varphi}_i \rangle \vec{\varphi}_i \right\|^2$
2	182753.567915	182753.570693
4	164311.705302	164311.712343
6	156296.758264	156296.757146
8	148780.806184	148780.808336
10	141653.502162	141653.507387
20	114794.313701	114794.326822
40	83565.3337824	83565.3313631
60	65667.1960201	65667.1963493
80	53841.1631402	53841.1635371
100	45045.8004678	45045.8035251

Introduction

Regularized models imply the use of a filter.

The POD Projection Filter

For a fixed $r_1 < r$ and a given $\vec{u}_r \in X^r$, the POD projection seeks $\mathcal{F}(\vec{u}_r) \in X^{r_1}$ such that

$$(\mathcal{F}(\vec{u}_r), \vec{\varphi}_j) = (\vec{u}_r, \vec{\varphi}_j), \forall j = 0, \cdots, r_1 - 1. \tag{4}$$

The POD Projection Filter

For a fixed $r_1 < r$ and a given $\vec{u}_r \in X^r$, the POD projection seeks $\mathcal{F}(\vec{u}_r) \in X^{r_1}$ such that

$$(\mathcal{F}(\vec{u}_r), \vec{\varphi}_j) = (\vec{u}_r, \vec{\varphi}_j), \forall j = 0, \cdots, r_1 - 1. \tag{4}$$

Doesn't work so well, see [6].

The POD Differential Filter

The POD differential filter is defined as follows: let δ be the radius of the POD differential filter. For a given $\vec{u}_r \in X^r$, find $\mathcal{F}(u_r) \in X^r$ such that

$$\Big((I - \delta^2 \Delta)\mathcal{F}(u^r), \vec{\varphi}_j\Big) = (\vec{u}_r, \vec{\varphi}_j), \forall j = 0, \cdots, r-1. \qquad (5)$$

What does the differential filter do?

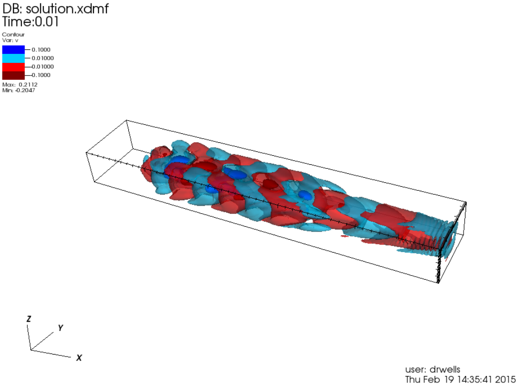


Figure: The first POD vector for the NSE experiment.

What does the differential filter do?

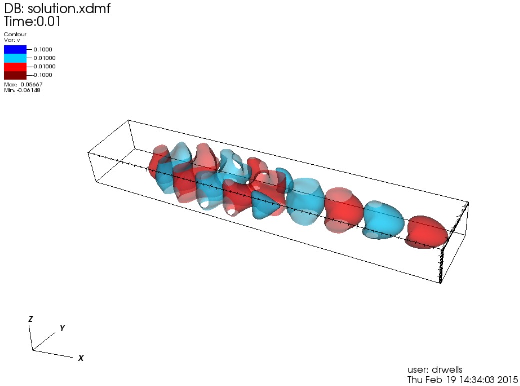


Figure: The filtered first POD vector for the NSE experiment, $\delta = 0.5$.

What does the differential filter do?

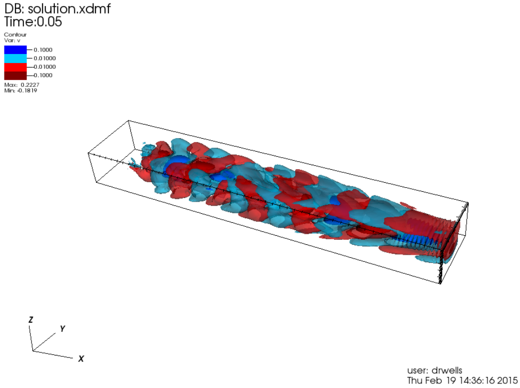


Figure: The fifth POD vector for the NSE experiment.

What does the differential filter do?

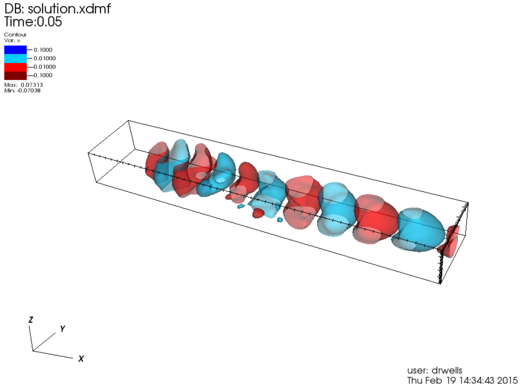


Figure: The filtered fifth POD vector for the NSE experiment, $\delta = 0.5$.

Overview

Considered REG-ROMs:

1. Leray regularization [4]
2. Evolve-then-filter [3]

Overview

Considered REG-ROMs:

1. Leray regularization [4]
2. Evolve-then-filter [3]

REG-ROMs are not:

1. consistent with the original PDE

The Navier-Stokes ROM

For $\vec{u} \approx \vec{u}_s + \vec{u}_r$:

$$(\vec{\varphi}_i, \vec{u}_{r,t}) = -\frac{1}{Re}(\nabla \vec{\varphi}_i, \nabla(\vec{u}_s + \vec{u}_r)) + \frac{1}{Re}\int_{\Gamma_2} \vec{\varphi}_i[0]u_x[0]dl \\ - (\vec{\varphi}_i, (\vec{u}_s + \vec{u}_r) \cdot \nabla(\vec{u}_s + \vec{u}_r)) \tag{6}$$

Combination of linear $(r \times r)$, quadratic $(r \times r \times r)$, and constant $(r \times 1)$ terms.

The Navier-Stokes ROM

For $\vec{u} \approx \vec{u}_s + \vec{u}_r$:

$$\begin{aligned}(\vec{\varphi}_i, \vec{u}_{r,t}) = -\frac{1}{Re}(\nabla \vec{\varphi}_i, \nabla(\vec{u}_s + \vec{u}_r)) + \frac{1}{Re}\int_{\Gamma_2} \vec{\varphi}_i[0]u_x[0]dl \\ - (\vec{\varphi}_i, (\vec{u}_s + \vec{u}_r) \cdot \nabla(\vec{u}_s + \vec{u}_r))\end{aligned} \tag{6}$$

Combination of linear $(r \times r)$, quadratic $(r \times r \times r)$, and constant $(r \times 1)$ terms. *Goal:* only change computational complexity by at most $\mathcal{O}(r^2)$.

The Leray Regularization

Jean Leray, 1934 [4]: existence of solutions (modulo a subsequence) to the *regularized* Navier-Stokes equations

$$\vec{u}_t - \frac{1}{Re}\Delta\vec{u} + \mathcal{F}(\vec{u}) \cdot \nabla\vec{u} + \nabla p = 0 \tag{7}$$

We rewrite the nonlinearity in the ROM as

$$\int_\Omega \vec{\varphi}_i \cdot (\vec{\varphi}_j \cdot \nabla\vec{\varphi}_k)\, dx \to \int_\Omega \vec{\varphi}_i \cdot (\mathcal{F}(\vec{\varphi}_j) \cdot \nabla\vec{\varphi}_k)\, dx \tag{8}$$

```cpp
std::pair<std::string, std::unique_ptr<ODE::RungeKuttaBase>> rk_factory
  (const FullMatrix<double>                 &boundary_matrix,
   const FullMatrix<double>                 &joint_convection,
   const FullMatrix<double>                 &laplace_matrix,
   const FullMatrix<double>                 &linear_operator,
   const Vector<double>                     &mean_contribution_vector,
   const FullMatrix<double>                 &mass_matrix,
   const std::vector<FullMatrix<double>>    &nonlinear_operator,
   const POD::NavierStokes::Parameters      &parameters);
```

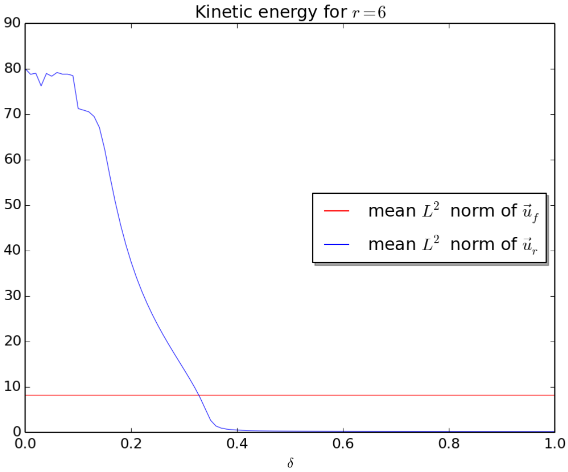
```
std::pair<std::string, std::unique_ptr<ODE::RungeKuttaBase>> rk_factory
  (const FullMatrix<double>              &boundary_matrix,
   const FullMatrix<double>              &joint_convection,
   const FullMatrix<double>              &laplace_matrix,
   const FullMatrix<double>              &linear_operator,
   const Vector<double>                  &mean_contribution_vector,
   const FullMatrix<double>              &mass_matrix,
   const std::vector<FullMatrix<double>> &nonlinear_operator,
   const POD::NavierStokes::Parameters   &parameters);
```

Use `unique_ptr` to implement factories and assemble the correct regularized model.

```cpp
std::pair<std::string, std::unique_ptr<ODE::RungeKuttaBase>> rk_factory
  (const FullMatrix<double>                &boundary_matrix,
   const FullMatrix<double>                &joint_convection,
   const FullMatrix<double>                &laplace_matrix,
   const FullMatrix<double>                &linear_operator,
   const Vector<double>                    &mean_contribution_vector,
   const FullMatrix<double>                &mass_matrix,
   const std::vector<FullMatrix<double>> &nonlinear_operator,
   const POD::NavierStokes::Parameters   &parameters);
```

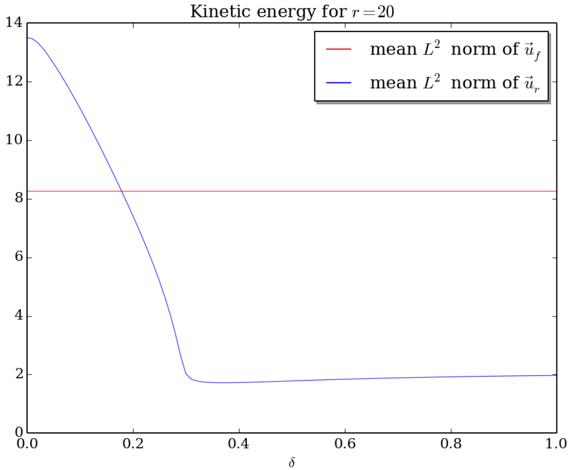
Use `unique_ptr` to implement factories and assemble the correct regularized model. No need for `delete`.

What happens as we vary δ?



The effect of δ (x-axis) on the mean kinetic energy (y-axis), with the correct mean in red and the Leray-DF-ROM in blue.

What happens as we vary δ?



The effect of δ (x-axis) on the mean kinetic energy (y-axis), with the correct mean in red and the Leray-DF-ROM in blue.

What is the optimal value for δ?

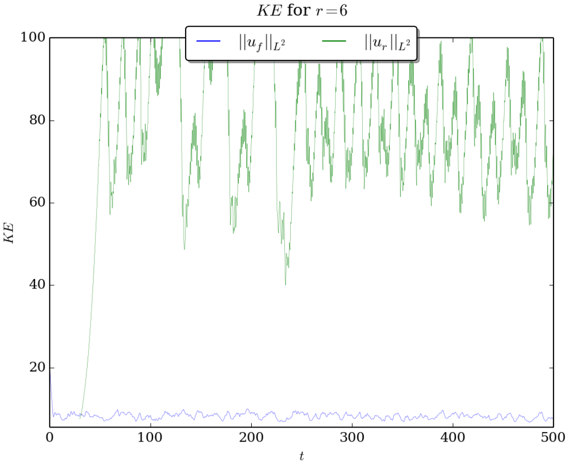


Figure: Galerkin ROM ($r = 6$), in green, with the DNS in blue.

What is the optimal value for δ?

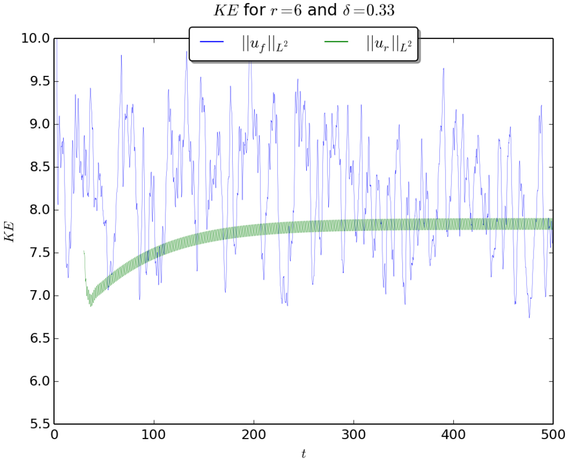


Figure: POD-ROM with $\delta = 0.33$, green.

What is the optimal value for δ?

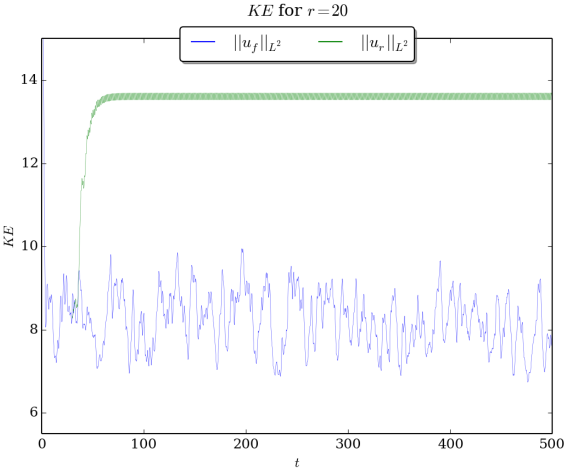


Figure: Galerkin ROM ($r = 20$), in green, with the DNS in blue.

What is the optimal value for δ?

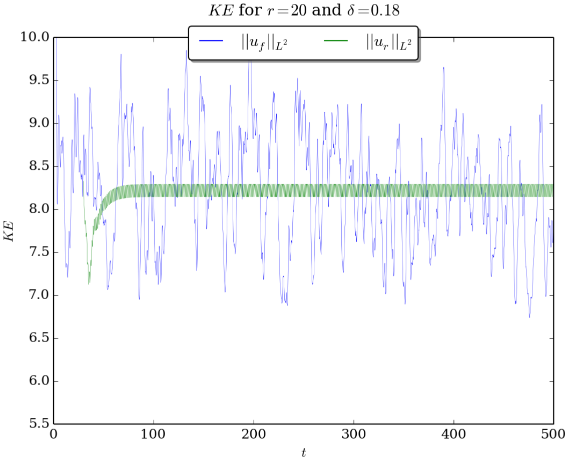
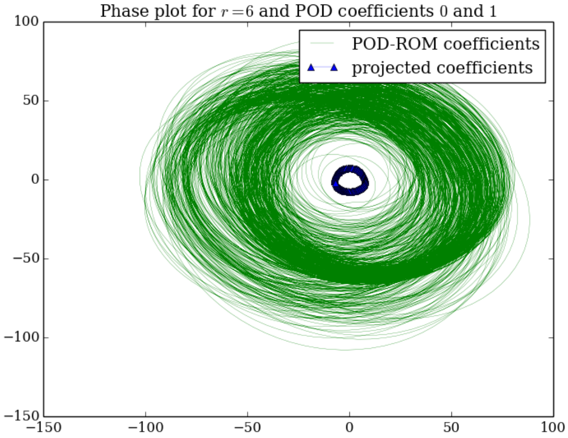


Figure: POD-ROM with $\delta = 0.18$, green.

Does changing δ “fix” the phase plots?

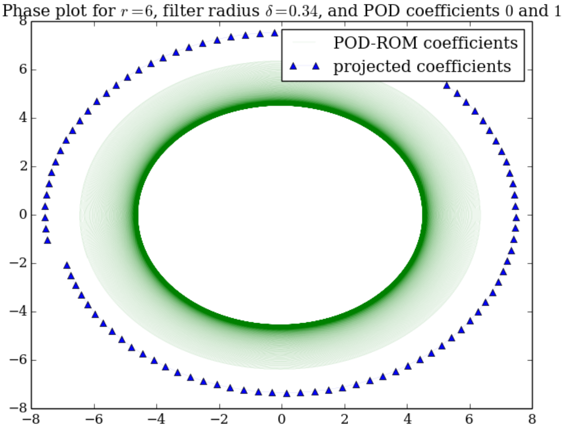
Yes!



Phase plot for the first and second POD vectors with $\delta = 0$.

Does changing δ "fix" the phase plots?

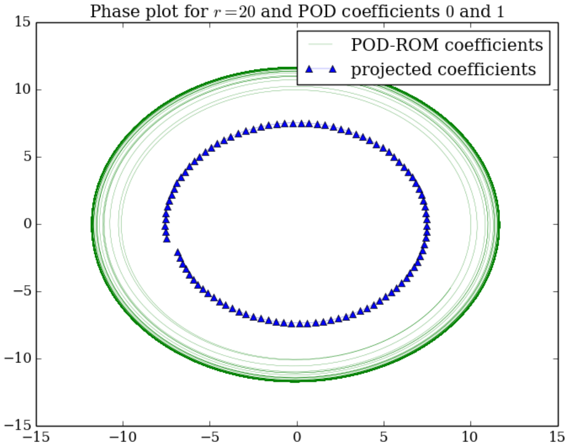
Yes!



Phase plot for the first and third POD vectors with $\delta = 0.34$.

Does changing δ "fix" the phase plots?

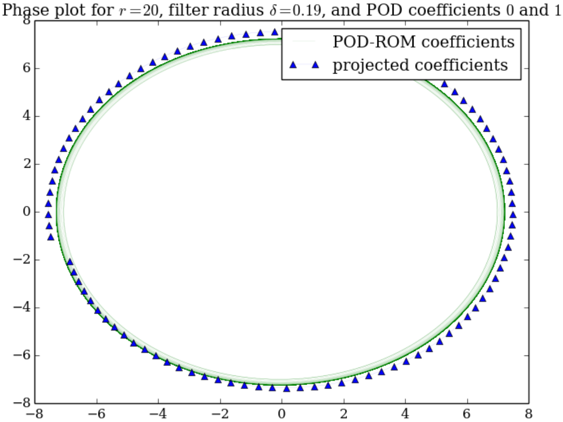
Yes!



Phase plot for the first and second POD vectors with $\delta = 0$.

Does changing δ “fix” the phase plots?

Yes!



Phase plot for the first and third POD vectors with $\delta = 0.34$.

An Evolve-And-Filter Model

Evolve:

$$\left(\frac{\vec{w}_r^{n+1} - \vec{u}_r^{n+1}}{\Delta t}\right) + \frac{1}{Re}\left(\nabla \vec{u}_s + \vec{u}_r^n, \nabla \vec{\varphi}_k\right) + \left(\left(\left(\vec{u}_s + \vec{u}_r^n\right) \cdot \nabla\right)\left(\vec{u}_s + \vec{u}_r^n\right), \vec{\varphi}_k\right) = 0, \tag{9}$$

then filter:

$$\vec{u}_r^{n+1} = \mathcal{F}(\vec{w}_r^{n+1}). \tag{10}$$

An Evolve-And-Filter Model

Evolve:

$$\left(\frac{\vec{w}_r^{n+1} - \vec{u}_r^{n+1}}{\Delta t}\right) + \frac{1}{Re}\left(\nabla \vec{u}_s + \vec{u}_r^n, \nabla \vec{\varphi}_k\right) + \left(\left(\left(\vec{u}_s + \vec{u}_r^n\right) \cdot \nabla\right)\left(\vec{u}_s + \vec{u}_r^n\right), \vec{\varphi}_k\right) = 0, \tag{9}$$

then filter:

$$\vec{u}_r^{n+1} = \mathcal{F}(\vec{w}_r^{n+1}). \tag{10}$$

In practice, I use RK4 instead of forward Euler.

An Evolve-And-Filter Model

For the differential filter:

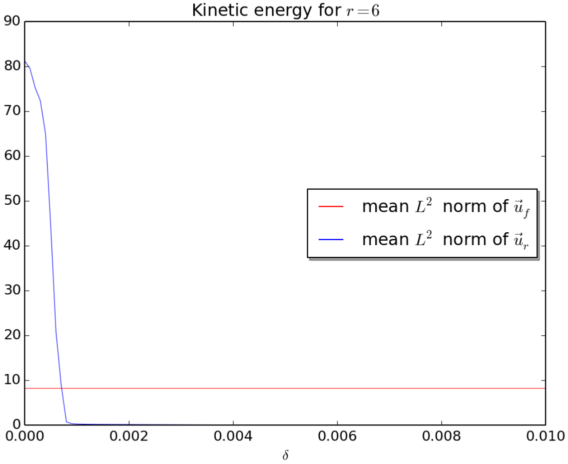


Figure: Mean kinetic energy based on filter radius.

An Evolve-And-Filter Model

For the differential filter:

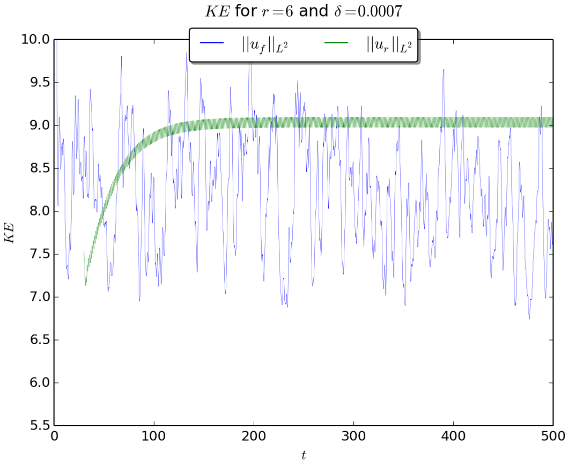


Figure: A slight overshoot: $\delta = 0.0007$.

An Evolve-And-Filter Model

For the differential filter:

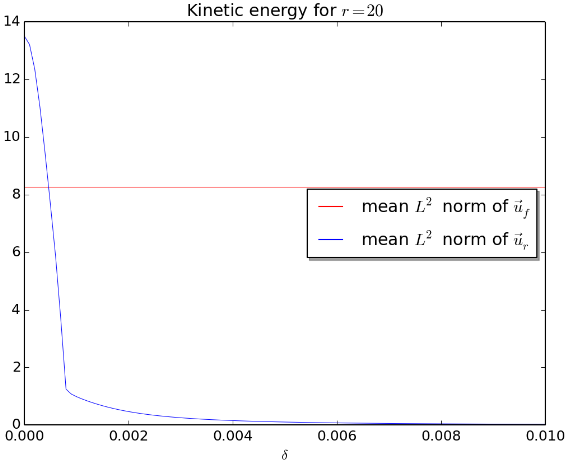


Figure: Mean kinetic energy based on filter radius.

An Evolve-And-Filter Model

For the differential filter:

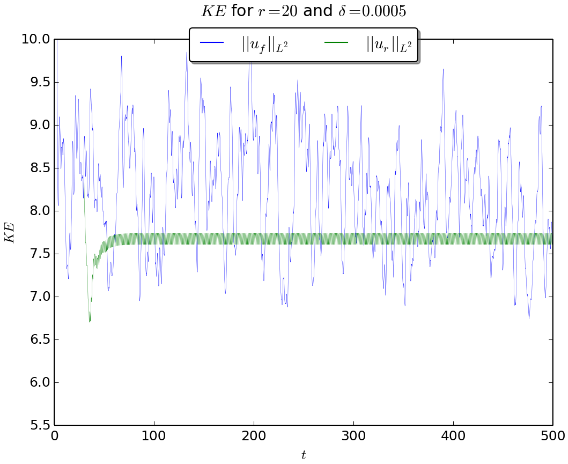


Figure: Kinetic energy over time for $\delta = 0.0005$.

Conclusions

1. Stabilization and regularization techniques can work for ROM.
2. Can ROM predict large structures? Sometimes.

Future Work: Big ROM Questions

1. Can ROMs predict large structures correctly?
2. Where do ROMs currently fail?
3. Does the energy cascade apply to POD-ROMs?
4. Can we justify this rigorously?

Future Work: Regularization

1. Are there better filtering models?

Future Work: Regularization

1. Are there better filtering models?
2. Deconvolution is an easy improvement to the Leray model.

W. Bangerth, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and T. D. Young.
The deal.ii library, version 8.1.
arXiv preprint, http://arxiv.org/abs/1312.2266v4, 2013.

J.-L. Guermond, P. Minev, and J. Shen.
Error analysis of pressure-correction schemes for the Navier-Stokes equations with open boundary conditions.
SIAM J. Numer. Anal., 43:239–258, 2005.

W. J. Layton and L. G. Rebholz.
Approximate Deconvolution Models of Turbulence, volume 2042 of *Lecture Notes in Mathematics*.
Springer, 2012.

Jean Leray.
Sur le mouvement d'un liquide visqueux emplissant l'espace.
Acta Mathematica, 63(1):193–248, 1934.

L. Sirovich.
Turbulence and the dynamics of coherent structures. Parts I–III.
Quart. Appl. Math., 45(3):561–590, 1987.

D. Wells, Z. Wang, X. Xie, and T. Iliescu.
Regularized reduced order models.
arXiv preprint arXiv:1506:07555, 2015.