Fluid Simulation in Kernel Space

Yufeng Zhu*
Department of Computer Science
University of British Columbia
Vancouver, BC Canada
mike323@cs.ubc.ca

Abstract

Fluid simulation has been one of the popular research topics in several areas, like computational physics, mechanical engineering and computer graphics, etc. However, computational complexity of most current techniques heavily depend on the simulation resolution. Even adaptive methods based on well-designed spatial partition technique will be overwhelmed in some cases due to the complicated reconstruction of the data structure. Such non-uniform spatial discretization may also lead to a non-symmetric linear system in the divergence free projection step, which could be a severe problem for most efficient iterative sparse linear solvers like preconditioned Conjugate Gradient, MINRES, etc. In this course project, we present a kernel space method by projecting the original high dimension space $\sim \mathbb{R}^{100^3}$ to a low dimension space $\sim \mathbb{R}^{10}$. By making use of learning based Principal Component Analysis method (PCA), the Galerkin Projection operator, which is a matrix in our case, can be found approximately. The learning data comes from original high dimension simulation. Once we extract the projection matrix from the training data, such costly simulation is no longer required. This project only focused on efficient simulation and its analysis part, while the report gave most of the mathematical proof details which is not described in the original paper [8]. In addition to be regarded as a project report, it can also be taken as a supporting document if you were confused by the magic math in the original paper. For readers who are interested in how to add in user interaction and other practical applications, we refer you to the original one.

1 Introduction

For the purposes of computer graphics, machine learning can be viewed as a set of techniques for leveraging data. Given some data, we can model the process that generated the data. Then, we can make more data that is consistent with this process, possibly with new, user-defined constraints. In learning, we combine our prior knowledge of the problem with the information in the training data. By employing existing ideas and techniques, we get the benefit of the collective experience of the researchers who studied these problems in the past. Otherwise, we will waste substantial effort reinventing the wheel. The idea of driving graphics from data is hardly new, and lots of researchers in this field have already conducted very interesting work so far. Rubine [6] proposed a statistically based recognition system of single stroke stylus gestures using a "linear machine" classifier, which determines the Bayes decision boundaries under the assumption of Gaussian distributions. Veach et al. [9] used Markov Chain Monte Carlo sampling (MCMC) to approximate global illumination integrals. Numerical simultion of physical based motion was demonstrated to be replaced with neural network by Grzeszczuk et al. [3]. Recent computer graphics paper making use of machine learning techniques includes data driven reflectance model [5], style based inverse kinematics [2],

*http://www.cs.ubc.ca/ mike323/

kernel space deformable object simulation [1], etc. Last year, Jakob et al. [4] shew impressive work in solving difficult rough surface specular transport problem using MCMC. In general, the machine learning technique and community have much to offer computer graphics.

This work present a way using learning based model reduction approach to speedup fluid simulation. Offline physical based animation can produce stunning motion of fluid. However, realtime applications generally simulate fluids only over small domains as a secondary effect. Fluid effects distant from the user are usually simulated at very low resolution or not at all. In the future, however, we expect interactive virtual environments with millions of simultaneous users where complex dynamic effects will play a primary role in the interaction: moving vehicles will splash through puddles as buildings leave turbulent eddies in the wind. Enabling such applications requires that such phenomena be computed everywhere with high resolution, and moreover that the dynamical state be consistent for all users. The difficulty is computing high resolution dynamics over such large environments. Existing fluid simulation approaches cannot scale to these large, realtime scenarios. Our method attempts to fill this gap by running the simulation in an extremely low dimension space, which we call *kernel space*, without sacrificing too much appealing detail, like vorticity and kinetic engery preservation.

We only applied this method to 2D cases, but it is straightforward to convert it to 3D cases and more speedup benefits should be gained as the computational cost grows cubically when increasing the simulation resolution. Moreover, we merely explore the initial value problem (IVP) in the kernel space, which means we assume the boundary condition for both original space during training step and kernel space during test step should be exactly the same. To add in user interaction or dynamic scene simulation, boundary value problem (BVP) in the kernel space should also be considered. Even though the idea of external force inverse projection is easy to understand, we are not going to explain it here as it is beyond the scope of our focusing material. For readers who are interested in dealing with those problems, we refer you to Treuille et al.'s paper [8] for a thorough description.

2 Projection Based Model Reduction

2.1 Background Overview

Model reduction, sometimes also called dimensional model reduction or model order reduction, is a technique to simplify the simulation of dynamical systems described by differential equations. The idea is to project the original, high dimension, state space onto a properly chosen low dimension subspace to arrive at a much smaller system having properties similar to the original system. Complex system can thus be be approximated by simpler system involving few equations and unknown variables, which can be solved much more quickly than the original one. Such projection based model reduction appears in literature under the names of *Principal Orthogonal Directions Method*, or *Subspace Integration Method*, and it has a long history in the engineering and applied mathematics literature.

Model reduction has been used extensively in the field of control theory, eletrical circuit simulation, computational electromagnectis and microelectromechanical system. Most model reduction techniques in these fields, however, aim at linear systems, and linear time invariant system in particular, e.g. small perturbations of voltages in some complex nonlinear circuit. Another common characteristics of these applications is that both input and output are low dimension, i.e., one may want to study how the voltage level at some circuit location depends on the input voltage at another location, in a complex nonlinear circuit. In computer graphics, however, one is often interested in nonlinear system that exhibit interesting, visible, dynamics. The output in computer graphics is usually high dimension, e.g., the deformation of an entire 3D solid object, or fluid velocities sampled on a high resolution grid. For these reasons, many conventional reduction techniques do not immediately apply to computer graphics problem.

2.2 Mathematical Intepretation

Here we provide a brief mathematical intepretation of model reduction mechanism. Suppose we have a state vector $\mathbf{x} \in \mathbb{R}^n$ describing the characteristic properties of our simulating subject. Instead of representing the subject in a high dimension space $\mathbb{R}^n$, we can describe it in a lower dimension subspace $\mathbb{R}^m, m \ll n$ if features or dimensions in the origianl space are correlated. We can simply

apply orthogonal projection $\mathbf{P}$ to the state vector $\mathbf{x}$ and get the reduced state vector $\mathbf{r} \in \mathbb{R}^m$, if the error metric is based on l_2 norm. Such projection operation allows us to move between the two space, namely $\mathbf{P} : \mathbf{x} \mapsto \mathbf{r}$ and $\mathbf{P}^{-1} : \mathbf{r} \mapsto \mathbf{x}$. What we have described so far is also regarded as dimension reduction, which is very useful in statistics research. The terminology *kernel space*, which corresponds to the low dimension subspace $\mathbb{R}^m$, also comes from statistics.

However, we are interested not only in the state $\mathbf{x}$, but also in its time evolution $\dot{\mathbf{x}}$, typically described by an ordinary differential equation (ODE) like the following,

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{F}(\mathbf{x}(t)) \\ \mathbf{x}(0) &= \mathbf{x}_0 \end{aligned} \tag{1}$$

The operator $\mathbf{F}(\cdot)$ in our case is linear, even though in general it could be nonlinear. Such an time dependent ODE describe a dynamical evolution in $\mathbb{R}^n$, which is time costly if we want to solve it numerically by a computer. What we are seeking for is the corresponding operator $\hat{\mathbf{F}}(\cdot)$ that evolves our reduced state vector $\mathbf{r}$ in the kernel space $\mathbb{R}^m$,

$$\begin{aligned} \dot{\mathbf{r}(t)} &= \hat{\mathbf{F}}(\mathbf{r}(t)) \\ \mathbf{r}(0) &= \mathbf{r}_0 \end{aligned} \tag{2}$$

To reach a formulation like Eqn. 2, we can start our derivation from Eqn. 1. Assume we already know the projection mapping $\mathbf{P}$, we can have

$$\begin{aligned} \mathbf{P}\mathbf{x}(t) &= \mathbf{r}(t) \\ \mathbf{x}(t) \approx \mathbf{P}^{-1}\mathbf{r}(t) &\approx \mathbf{P}^T\mathbf{r}(t) \end{aligned} \tag{3}$$

By plugging the approximation of $\mathbf{x}$(t) into Eqn. 1, we have

$$\mathbf{P}^T\dot{\mathbf{r}}(t) = \mathbf{F}(\mathbf{P}^T\mathbf{r}(t)) + \mathbf{e}(t) \tag{4}$$

$\mathbf{e}$(t) is the error term caused by the approximation and it's time dependent, while the projection operator is time invariant. Assume $\mathbf{e}$(t) is constrained in a subspace of $\mathbb{R}^n$, we then can introduce another projection operator $\mathbf{W}$ satisfying

$$\mathbf{W}\mathbf{e}(t) = \mathbf{0} \tag{5}$$

Range space of $\mathbf{W}$, $R(\mathbf{W})$, is simply the orthogonal complement of the subspace that $\mathbf{e}$(t) lies in. Such assumption is valid for most linear model $\mathbf{F}(\cdot)$ due to the fact that their trajectories are contained in low dimension subspaces. For a thorough description, we refer you to []. By applying projection operator $\mathbf{W}$ onto both sides of Eqn. 4, it becomes

$$\mathbf{W}\mathbf{P}^T\dot{\mathbf{r}}(t) = \mathbf{W}\mathbf{F}(\mathbf{P}^T\mathbf{r}(t)) + \mathbf{W}\mathbf{e}(t) = \mathbf{W}\mathbf{F}(\mathbf{P}^T\mathbf{r}(t)) \tag{6}$$

Eqn. 6 is known as Petrov-Galerkin projection based equation. In particular, when $\mathbf{W} = \mathbf{P}$, this is called Galerkin projection based equation. When $\mathbf{P}$ is orthogonal, this leads to

$$\dot{\mathbf{r}}(t) = \mathbf{P}\mathbf{F}(\mathbf{P}^T\mathbf{r}(t)) \tag{7}$$

Thus we obtain the expression for the operator $\hat{\mathbf{F}}(\cdot)$,

$$\hat{\mathbf{F}}(\cdot) = \mathbf{P} \circ \mathbf{F} \circ \mathbf{P}^T(\cdot) \tag{8}$$

The above derivation is not only valid in vector space but also valid in general Hilbert space like Sobolev or Lebesgue space.

3 Model Reduction of Fluid Simulation

We now show how we can apply the above model reduction technique to fluid simulation. In fluid simulation, the fluid state at each time t is described by the velocity field $\mathbf{u}$(t). For example, in 2D case, it is just a 2D vector field. The evolution of such state is governed by well known nonlinear partial differential equations (PDE), Navier-Stokes equations

$$\dot{\mathbf{u}} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \nu\nabla^2\mathbf{u} + \nabla p + \mathbf{f} \tag{9}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{10}$$

Since we only consider incompressible flow, momentum (Eqn. 9) and mass (Eqn. 10) conservation are enough. Moreover, as we just focus on invisicid flow and IVP, we can get rid of the viscosity term $-\nu\nabla^2\mathbf{u}$ and external force $\mathbf{f}$ at the moment. Finally, the dynamical PDE we are actually solving is

$$\begin{aligned} \dot{\mathbf{u}} &= -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned} \tag{11}$$

Now we have the operator $\mathbf{F}(\cdot)$ which is the right hand side of momentum conservation equation, while mass conservation equation simply plays the role of removing ambiguities when shock happens due to nonlinear advection. To construct the kernel space operator $\hat{\mathbf{F}}(\cdot)$, we still need the projection operator $\mathbf{P}$. We adopt the PCA based method described in [8] to construct the projection operator. In order to apply PCA, we first need to collect state samples from the full order space. Before we move onto the kernel space, we perform a training step by solving Eqn. 11 in the full order space and collect a set of example velocity fields $\{\mathbf{u}_i\}$. We applied *Stable Fluid* algorithm in the training step, simply followed the time splitting scheme and did Semi-Lagrangian advection as well as Poisson divergence free solve. For more details, we refer you to Jos Stam's paper [7].

3.1 PCA Based Projection

In this section, we will provide the mathematical proof regarding how the PCA based projection matrix comes from. In general, one can use any projection matrix as $\mathbf{P}$. However, to make the projection operator meaningful, one need to specify certain metric or error measurement that the projection can minimize. Otherwise, the projection error will be unbounded. In our case, we use the reconstruction error as our measurement which is

$$\mathbf{U} - \mathbf{B}\mathbf{B}^T\mathbf{U} \tag{12}$$

Here $\mathbf{U}$ is a n by s matrix whose column vectors are from our example velocity set $\{\mathbf{u}_i\}, \mathbf{u}_i \in \mathbb{R}^n$. $\mathbf{B}$ is an unknown n by m orthonormal matrix which is the projection matrix $\mathbf{P}$ we are seeking for. $\mathbf{B}\mathbf{B}^T\mathbf{U}$ just means reconstruction from the subspace and we want to measure its distance from the original one. We use squared Frobenius norm, which is the sum of elementwise square, as our metric.

$$\zeta = \|\mathbf{U} - \mathbf{B}\mathbf{B}^T\mathbf{U}\|_F^2 \tag{13}$$

Then we will show that in order to minimize ζ, $\mathbf{B}$ need to be constructed by the first m eigen vectors of $\mathbf{U}\mathbf{U}^T$.

$$
\begin{aligned}
\zeta &= \|\mathbf{U} - \mathbf{B}\mathbf{B}^T\mathbf{U}\|_F^2 = tr[(\mathbf{U} - \mathbf{B}\mathbf{B}^T\mathbf{U})(\mathbf{U} - \mathbf{B}\mathbf{B}^T\mathbf{U})^T] \\
&= tr[\mathbf{U}\mathbf{U}^T - \mathbf{U}\mathbf{U}^T\mathbf{B}\mathbf{B}^T - \mathbf{B}\mathbf{B}^T\mathbf{U}\mathbf{U}^T + \mathbf{B}\mathbf{B}^T\mathbf{U}\mathbf{U}^T\mathbf{B}\mathbf{B}^T] \\
&= tr[\mathbf{U}\mathbf{U}^T] - tr[\mathbf{U}\mathbf{U}^T\mathbf{B}\mathbf{B}^T] - tr[\mathbf{B}\mathbf{B}^T\mathbf{U}\mathbf{U}^T] + tr[\mathbf{B}\mathbf{B}^T\mathbf{U}\mathbf{U}^T\mathbf{B}\mathbf{B}^T] \\
&= tr[\mathbf{U}\mathbf{U}^T] - tr[\mathbf{U}\mathbf{U}^T\mathbf{B}\mathbf{B}^T] - tr[\mathbf{U}\mathbf{U}^T\mathbf{B}\mathbf{B}^T] + tr[\mathbf{B}\mathbf{B}^T\mathbf{B}\mathbf{B}^T\mathbf{U}\mathbf{U}^T] \\
&= tr[\mathbf{U}\mathbf{U}^T] - 2tr[\mathbf{U}\mathbf{U}^T\mathbf{B}\mathbf{B}^T] + tr[\mathbf{B}\mathbf{I}\mathbf{B}^T\mathbf{U}\mathbf{U}^T] \\
&= tr[\mathbf{U}\mathbf{U}^T] - 2tr[\mathbf{U}\mathbf{U}^T\mathbf{B}\mathbf{B}^T] + tr[\mathbf{U}\mathbf{U}^T\mathbf{B}\mathbf{B}^T] \\
&= tr[\mathbf{U}\mathbf{U}^T] - tr[\mathbf{U}\mathbf{U}^T\mathbf{B}\mathbf{B}^T] \\
&= tr[\mathbf{U}\mathbf{U}^T] - tr[\mathbf{B}^T\mathbf{U}\mathbf{U}^T\mathbf{B}]
\end{aligned}
\tag{14}
$$

Notice $tr[\mathbf{U}\mathbf{U}^T]$ is the sum of all the eigen values of matrix $\mathbf{U}\mathbf{U}^T$, which is nonegative as $\mathbf{U}\mathbf{U}^T$ is semi positive definite. $\mathbf{B}^T\mathbf{U}\mathbf{U}^T\mathbf{B}$ is also symmetric and thus semi positive definite. As a result, its trace is also nonegative. Moreover, ζ is also nonegative. Thus we have

$$
0 \leq tr[\mathbf{B}^T\mathbf{U}\mathbf{U}^T\mathbf{B}] \leq tr[\mathbf{U}\mathbf{U}^T] \tag{15}
$$

Thus to minimize ζ we need to maximize $tr[\mathbf{B}^T\mathbf{U}\mathbf{U}^T\mathbf{B}]$ as $tr[\mathbf{U}\mathbf{U}^T]$ is fixed. Then the problem becomes to a constrained optimization problem. To see it more clearly, let's define $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m]$. And we have the following

$$
tr[\mathbf{B}^T\mathbf{U}\mathbf{U}^T\mathbf{B}] = \sum_{i=1}^{m} \mathbf{b}_i^T\mathbf{U}\mathbf{U}^T\mathbf{b}_i \tag{16}
$$

Then we can formulate our optimization problem as following

$$
\begin{aligned}
&\textbf{maximize: } f(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m) = \sum_{i=1}^{m} \mathbf{b}_i^T\mathbf{U}\mathbf{U}^T\mathbf{b}_i \\
&\textbf{s.t. } \mathbf{b}_i^T\mathbf{b}_j = \delta_{ij},\ \textit{i, j = 1, 2, ..., m}
\end{aligned}
\tag{17}
$$

δ is the Kronecker delta. To solve such an optimization problem, we are going to use Lagrange multiplier.

$$
\begin{aligned}
L(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m) &= \sum_{i=1}^{m} \mathbf{b}_i^T\mathbf{U}\mathbf{U}^T\mathbf{b}_i + \sum_{i=1}^{m} \lambda_i(1 - \mathbf{b}_i^T\mathbf{b}_i) - \sum_{i \neq j} \mu_{ij}\mathbf{b}_i^T\mathbf{b}_j \\
\frac{\partial L}{\partial \mathbf{b}_i} &= 2\mathbf{U}\mathbf{U}^T\mathbf{b}_i - 2\lambda_i\mathbf{b}_i - \sum_{i \neq j} \mu_{ij}\mathbf{b}_j = 0
\end{aligned}
\tag{18}
$$

So far the problem is still hard to solve especially when we take the nonlinear constraints into account. However, we can solve a less constrained problem instead getting rid of constraints $\mathbf{b}_i^T\mathbf{b}_j = 0, i \neq j$. And then we have

$$
\begin{aligned}
\frac{\partial L}{\partial \mathbf{b}_i} &= 2\mathbf{U}\mathbf{U}^T\mathbf{b}_i - 2\lambda_i\mathbf{b}_i = 0 \\
\mathbf{U}\mathbf{U}^T\mathbf{b}_i &= \lambda_i\mathbf{b}_i
\end{aligned}
\tag{19}
$$

This means we can obtain optimal for the less constrained problem when $\mathbf{b}_i$ are eigen vectors of $\mathbf{U}\mathbf{U}^T$. Since those eigen vectors also satisfy the orthogonal constraints, they are also optimal solutions to our original problem. To maximize our objective, we simply pick the first m eigen vectors corresponding to the largest m eigen values, when our maximum is the sum of largest m eigen values.

3.2 Advection

In the kernel space, to linearize the advection operator, we use the conservation form of the advection step.

$$\dot{u}^* = -(\mathbf{u} \cdot \nabla)u^* = -\nabla \cdot (\mathbf{u}u^*) + u^* \nabla \cdot \mathbf{u} \tag{20}$$

As described in the original paper, our example velocities are divergence free. The linearity of the projection operation keeps such property. Thus the conservation form becomes

$$\dot{u}^* = -\nabla \cdot (\mathbf{u}u^*) \tag{21}$$

Fig. 1 shows the MAC grid we used for storing velocities (on grid face) and other properties (on grid center) like pressure, density or temperature.

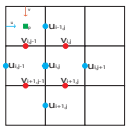


Figure 1: Nine Point Scheme

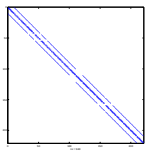


Figure 2: Sparsity Pattern

By discretizing the conservation form about each grid face, we can formulate our method by using Nine Point discretization scheme

$$\dot{u}^*_{i,j} = \frac{1}{\Delta h}((u^x u^*)_{i-1/2,j} - (u^x u^*)_{i+1/2,j} + (u^y u^*)_{i,j-1/2} - (u^y u^*)_{i,j-1/2}) \tag{22}$$

Here Δh is the grid spacing. By assuming u^x and u^y are known constant, we can linearize Eqn. 22 and formulate a linear system after grouping the equations together

$$\dot{\mathbf{u}} = \mathbf{A}_\mathbf{u}\mathbf{u} \tag{23}$$

As shown in Fig. 2, $\mathbf{A}_\mathbf{u}$ is a low bandwidth sparse matrix.

Then we need to derive the advection operator in the kernel space. Based on the Galerkin Projection rule we show in the last section, the operator should look like

$$\hat{\mathbf{A}} = \mathbf{B}^T\mathbf{A}\mathbf{B} \tag{24}$$

However, as the advection operator itself depends on the velocity fluxes, the advection operator built on the kernel space actually related to the basis of the subspace. Finally, we actually get an order three tensor $\check{\mathbf{A}}$ as our advection operator

$$\check{\mathbf{A}} = \mathbf{B}^T\mathbf{A}_{\hat{\mathbf{u}}_i}\mathbf{B} \tag{25}$$

Here we use the Eistein summation convention and notice $\mathbf{A}_{\hat{\mathbf{u}}_i}$ itself is an order 2 tensor, which is a matrix. To reconstruct this order 2 advection tensor in the kernel space, we need to contract it by the kernel space state vector **r**, where **u = Br**

$$\hat{\mathbf{A}} = \check{\mathbf{A}} \otimes_1 \mathbf{r} = r_i\mathbf{B}^T\mathbf{A}_{\hat{\mathbf{u}}_i}\mathbf{B} \tag{26}$$

Then we have the advection equation in the kernel space, which is

$$\dot{\mathbf{r}} = \hat{\mathbf{A}}\mathbf{r} \tag{27}$$

Thus at each timestep we will just take a linear combination of the m precomputed reduced advection matrices $\mathbf{B}^T\mathbf{A}_{\hat{\mathbf{u}}_i}\mathbf{B}$ to form the reduced advection matrix $\hat{\mathbf{A}}$. Then we can show how to solve this ODE symbolically as it has closed form analytical solution given initial condition $\mathbf{r}(0)$. The result is obvious with the knowledge of matrix calculus, it's

$$\mathbf{r} = e^{\hat{\mathbf{A}}t}\mathbf{r}(0) \tag{28}$$

To see it more clearly, you can take the Taylor expansion of $e^{\hat{\mathbf{A}}t}$ and plug it into the ODE and you will get it proved. This is already enough for the IVP ODE problem actually. Given an initial velocity field condition, one can get future velocity field at any time slice. However, the authors provided an iterative form of the solution in the original paper, because they also want to handle BVP problem, which means the velocity field might be changed due to external force pulse by user interaction or whatever. Here we also follow their method and we will show how that iterative form comes from.

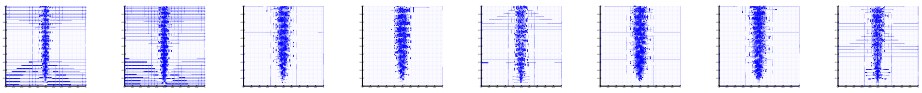
$$\begin{aligned} \mathbf{r}(t+\Delta t) &= e^{\hat{\mathbf{A}}(t+\Delta t)}\mathbf{r}(0) \\ \mathbf{r}(t) &= e^{\hat{\mathbf{A}}(t)}\mathbf{r}(0) \\ \Rightarrow \mathbf{r}(t+\Delta t) &= e^{\hat{\mathbf{A}}(\Delta t)}\mathbf{r}(t) \end{aligned} \tag{29}$$

$$\begin{aligned} & e^{\hat{\mathbf{A}}t} = \mathbf{I} + \hat{\mathbf{A}}t + \frac{1}{2}\hat{\mathbf{A}}^2t^2 + \frac{1}{3!}\hat{\mathbf{A}}^3t^3 + \dots \\ & \Rightarrow e^{\mathbf{E}\mathbf{\Lambda}\mathbf{E}^{-1}t} = \mathbf{I} + \mathbf{E}\mathbf{\Lambda}\mathbf{E}^{-1}t + \frac{1}{2}\mathbf{E}\mathbf{\Lambda}^2\mathbf{E}^{-1}t^2 + \frac{1}{3!}\mathbf{E}\mathbf{\Lambda}^3\mathbf{E}^{-1}t^3 + \dots = \mathbf{E}e^{\mathbf{\Lambda}t}\mathbf{E}^{-1} \\ & \Rightarrow \mathbf{r}(t+\Delta t) = \mathbf{E}e^{\mathbf{\Lambda}\Delta t}\mathbf{E}^{-1}\mathbf{r}(t) \end{aligned} \tag{30}$$

4 Results

We did all the experiment using Matlab including full order Stable Fluid simulation, PCA and kernel space simulation. We tried the popular *Kármán Vortex Street* as our verification test case. As our experiment, by using the first 8 eigen velocity fields, the reconstructed velocity field from the kernel space is able to cover most of the details in the full order simulation. With more eigen velocity fields included, more minor details will be recovered. Here we show our first 8 eigen velocity fields in Fig. 3. They are from the full order space $\mathbb{R}^{100^2}$.

Then in Fig. 4.a and 4.b, we show the full order simulation result compared with the reconstructed one from the kernel space. And in Fig. 4.c and 4.d, we visualize the two cases by advecting some material in those velocity fields and make a comparision.

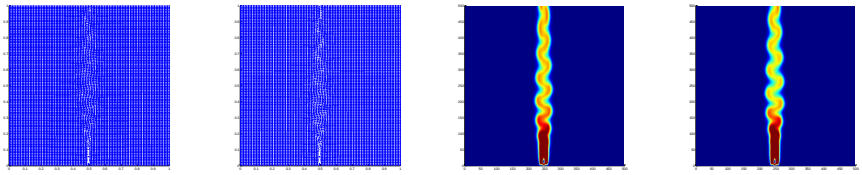


(a) (b) (c) (d) (e) (f) (g) (h)

Figure 3: First 8 Eigen Velocity Fields

5 Conclusion

As we have shown so far, the math of this method is quite beautiful. And the speedup achievement is absolutely promising. It's really a good example of incorporation of Machine Learning techniques



(a) Full Order Simulation (b) Kernel Space Simulation (c) Full Order Simulation (d) Kernel Space Simulation

Figure 4: Results Comparision

into physical based simulation. However, one unpleasant thing came up during the project is to decide how many eigen velocity fields are enough to achieve certain requirements is manually, which is quite annoying to us. Even though we have the error metric for the projection in general, there is no straightforward way to reach such explicit expression in our case. Moreover, such issue is not mentioned in the original paper, either, which makes us feel so disappointed. For future work, we are expecting to see solution to this problem.

References

[1] Jernej Barbič and Doug L. James. Real-time subspace integration for st. venant-kirchhoff deformable models. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 982–990, New York, NY, USA, 2005. ACM.

[2] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 522–531, New York, NY, USA, 2004. ACM.

[3] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. Neuroanimator: fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 9–20, New York, NY, USA, 1998. ACM.

[4] Wenzel Jakob and Steve Marschner. Manifold exploration: a markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph.*, 31(4):58:1–58:13, July 2012.

[5] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 759–769, New York, NY, USA, 2003. ACM.

[6] Dean Rubine. Specifying gestures by example. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '91, pages 329–337, New York, NY, USA, 1991. ACM.

[7] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[8] Adrien Treuille, Andrew Lewis, and Zoran Popović. Model reduction for real-time fluids. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 826–834, New York, NY, USA, 2006. ACM.

[9] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 65–76, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.