Nonlinear Model Order Reduction using POD/DEIM for Optimal Control of Burgers' Equation

Manuel M. Baumann

July 15, 2013

Outline

Nonlinear dynamical systems

Consider the nonlinear dynamical system

$$\begin{aligned} \dot{\mathbf{y}}(t) &= A\mathbf{y}(t) + \mathbf{F}(t, \mathbf{y}(t)), \quad \mathbf{y}(t) \in \mathbb{R}^N \\ \mathbf{y}(0) &= \mathbf{y}_0 \end{aligned} \tag{1}$$

- arises in many applications, e.g. mechanical systems, fluid dynamics, neuron modeling, ...
- the matrix A represents the linear dynamical behavior and the function $\mathbf{F}$ represents nonlinear dynamics
- often large dimension of (1) leads to *huge* computational work

Nonlinear dynamical systems

Consider the nonlinear dynamical system

$$\begin{aligned} \dot{\mathbf{y}}(t) &= A\mathbf{y}(t) + \mathbf{F}(t, \mathbf{y}(t)), \quad \mathbf{y}(t) \in \mathbb{R}^N \\ \mathbf{y}(0) &= \mathbf{y}_0 \end{aligned} \tag{1}$$

- arises in many applications, e.g. mechanical systems, fluid dynamics, neuron modeling, ...
- the matrix A represents the linear dynamical behavior and the function $\mathbf{F}$ represents nonlinear dynamics
- often large dimension of (1) leads to *huge* computational work

Nonlinear dynamical systems

Consider the nonlinear dynamical system

$$\begin{aligned} \dot{\mathbf{y}}(t) &= A\mathbf{y}(t) + \mathbf{F}(t, \mathbf{y}(t)), \quad \mathbf{y}(t) \in \mathbb{R}^N \\ \mathbf{y}(0) &= \mathbf{y}_0 \end{aligned} \tag{1}$$

- arises in many applications, e.g. mechanical systems, fluid dynamics, neuron modeling, ...
- the matrix A represents the linear dynamical behavior and the function $\mathbf{F}$ represents nonlinear dynamics
- often large dimension of (1) leads to *huge* computational work

Nonlinear dynamical systems

Consider the nonlinear dynamical system

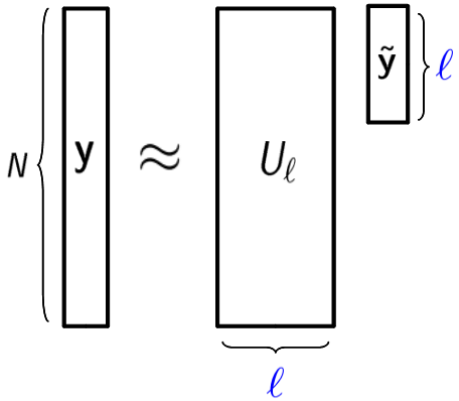
$$\begin{aligned} \dot{\mathbf{y}}(t) &= A\mathbf{y}(t) + \mathbf{F}(t, \mathbf{y}(t)), \quad \mathbf{y}(t) \in \mathbb{R}^N \\ \mathbf{y}(0) &= \mathbf{y}_0 \end{aligned} \tag{1}$$

- arises in many applications, e.g. mechanical systems, fluid dynamics, neuron modeling, ...
- the matrix A represents the linear dynamical behavior and the function $\mathbf{F}$ represents nonlinear dynamics
- often large dimension of (1) leads to *huge* computational work

The idea of model order reduction

Approximate the state via

$$\mathbf{y}(t) \approx U_\ell \tilde{\mathbf{y}}(t), \quad U_\ell \in \mathbb{R}^{N\times \ell}, \tilde{\mathbf{y}} \in \mathbb{R}^{\ell},$$



where the matrix U_ℓ has orthonormal columns, the so-called *principal components* of $\mathbf{y}$, and $\ell \ll N$.

Galerkin projection of the original full-order system leads to a reduced system of ℓ equations:

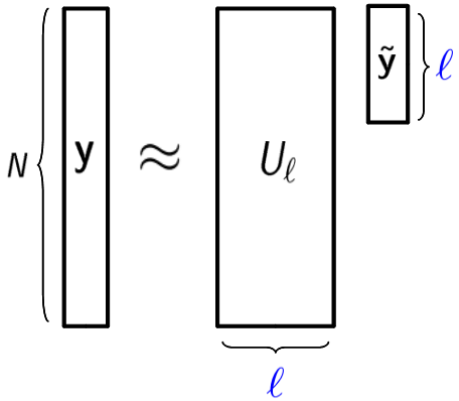
$$U_\ell^T \left[U_\ell \dot{\tilde{\mathbf{y}}} - A U_\ell \tilde{\mathbf{y}} - \mathbf{F}(t, U_\ell \tilde{\mathbf{y}}) \right] = 0$$

$$\Rightarrow \quad \dot{\tilde{\mathbf{y}}} = \underbrace{U_\ell^T A U_\ell}_{=:\tilde{A}} \tilde{\mathbf{y}} + U_\ell^T \mathbf{F}(t, U_\ell \tilde{\mathbf{y}})$$

The idea of model order reduction

Approximate the state via

$$\mathbf{y}(t) \approx U_\ell \tilde{\mathbf{y}}(t), \quad U_\ell \in \mathbb{R}^{N\times \ell}, \tilde{\mathbf{y}} \in \mathbb{R}^{\ell},$$



where the matrix U_ℓ has orthonormal columns, the so-called *principal components* of $\mathbf{y}$, and $\ell \ll N$.

Galerkin projection of the original full-order system leads to a reduced system of ℓ equations:

$$U_\ell^T \left[U_\ell \dot{\tilde{\mathbf{y}}} - A U_\ell \tilde{\mathbf{y}} - \mathbf{F}(t, U_\ell \tilde{\mathbf{y}}) \right] = 0$$

$$\Rightarrow \quad \dot{\tilde{\mathbf{y}}} = \underbrace{U_\ell^T A U_\ell}_{=:\tilde{A}} \tilde{\mathbf{y}} + U_\ell^T \mathbf{F}(t, U_\ell \tilde{\mathbf{y}})$$

Two questions are left...

Considering the reduced model

$$\dot{\tilde{\mathbf{y}}}(t) = \tilde{A}\tilde{\mathbf{y}}(t) + U_\ell^T \mathbf{F}(t, U_\ell \tilde{\mathbf{y}}(t)), \quad \tilde{\mathbf{y}}(t) \in \mathbb{R}^\ell$$

two questions are left:

1. How to obtain the matrix U_ℓ of principal components ?
2. Note that $U_\ell \tilde{\mathbf{y}}(t) \in \mathbb{R}^N$ is still large. How do we evaluate $\mathbf{F}(t, U_\ell \tilde{\mathbf{y}}(t))$ efficiently ?

Two questions are left...

Considering the reduced model

$$\dot{\tilde{\mathbf{y}}}(t) = \tilde{A}\tilde{\mathbf{y}}(t) + U_\ell^T \mathbf{F}(t, U_\ell \tilde{\mathbf{y}}(t)), \quad \tilde{\mathbf{y}}(t) \in \mathbb{R}^\ell$$

two questions are left:

1. How to obtain the matrix U_ℓ of principal components ?
2. Note that $U_\ell \tilde{\mathbf{y}}(t) \in \mathbb{R}^N$ is still large. How do we evaluate $\mathbf{F}(t, U_\ell \tilde{\mathbf{y}}(t))$ efficiently ?

Two questions are left...

Considering the reduced model

$$\dot{\tilde{\mathbf{y}}}(t) = \tilde{A}\tilde{\mathbf{y}}(t) + U_\ell^T \mathbf{F}(t, U_\ell \tilde{\mathbf{y}}(t)), \quad \tilde{\mathbf{y}}(t) \in \mathbb{R}^\ell$$

two questions are left:

1. How to obtain the matrix U_ℓ of principal components ?
2. Note that $U_\ell \tilde{\mathbf{y}}(t) \in \mathbb{R}^N$ is still large. How do we evaluate $\mathbf{F}(t, U_\ell \tilde{\mathbf{y}}(t))$ efficiently ?

The Proper Orthogonal Decomposition (POD)

During the numerical simulation, build up the snapshot matrix

$$Y := [\mathbf{y}(t_1), ..., \mathbf{y}(t_{n_s})] \in \mathbb{R}^{N \times n_s},$$

with n_s being the number of snapshots.

Perform a Singular Value Decomposition (SVD)

$$Y = U \Sigma V^T$$

and let $U_\ell := \texttt{U(:,1:l)}$ consist of those left singular vectors of Y that correspond to the ℓ largest singular values in Σ.

The Discrete Empirical Interpolation Method (DEIM)

Consider the nonlinearity

$$\mathbf{N} := \underbrace{U_\ell^T}_{\ell \times N} \underbrace{\mathbf{F}(t, U_\ell \tilde{\mathbf{y}}(t))}_{N \times 1}$$

The approximation

$$\mathbf{F} \approx W\mathbf{c}, \quad W \in \mathbb{R}^{N \times m}, \mathbf{c} \in \mathbb{R}^m$$

is over-determined. Therefore, find projector $\mathcal{P}$ such that:

$$\begin{aligned} \mathcal{P}^T\mathbf{F} = (\mathcal{P}^T W)\mathbf{c} \quad &\Rightarrow \quad \mathbf{F} \approx W\mathbf{c} = W(\mathcal{P}^T W)^{-1}\mathcal{P}^T\mathbf{F} \\ &\Rightarrow \quad \mathbf{N} \approx U_\ell^T W \underbrace{(\mathcal{P}^T W)}_{m \times m}{}^{-1}\mathcal{P}^T\mathbf{F}(t, U_\ell \tilde{\mathbf{y}}(t)) \end{aligned}$$

The Discrete Empirical Interpolation Method (DEIM)

Consider the nonlinearity

$$\mathbf{N} := \underbrace{U_\ell^T}_{\ell \times N} \underbrace{\mathbf{F}(t, U_\ell \tilde{\mathbf{y}}(t))}_{N \times 1}$$

The approximation

$$\mathbf{F} \approx W\mathbf{c}, \quad W \in \mathbb{R}^{N \times m}, \mathbf{c} \in \mathbb{R}^m$$

is over-determined. Therefore, find projector $\mathcal{P}$ such that:

$$\begin{aligned} \mathcal{P}^T\mathbf{F} = (\mathcal{P}^T W)\mathbf{c} \quad &\Rightarrow \quad \mathbf{F} \approx W\mathbf{c} = W(\mathcal{P}^T W)^{-1}\mathcal{P}^T\mathbf{F} \\ &\Rightarrow \quad \mathbf{N} \approx U_\ell^T W \underbrace{(\mathcal{P}^T W)}_{m \times m}{}^{-1}\mathcal{P}^T\mathbf{F}(t, U_\ell \tilde{\mathbf{y}}(t)) \end{aligned}$$

Algorithm 1 The DEIM algorithm [Chaturantabut, Sorensen, 2010]

INPUT: $\{\mathbf{w}_i\}_{i=1}^m \subset \mathbb{R}^N$ linear independent

OUTPUT: $\vec{\wp} = [\wp_1, ..., \wp_m]^T \in \mathbb{R}^m$, $\mathcal{P} \in \mathbb{R}^{N \times m}$

$[|\rho|, \wp_1] = \max\{|\mathbf{w}_1|\}$

$W = [\mathbf{w}_1], \mathcal{P} = [\mathbf{e}_{\wp_1}], \vec{\wp} = [\wp_1]$

for $i = 2$ to m **do**

  Solve $(\mathcal{P}^T W)\mathbf{c} = \mathcal{P}^T \mathbf{w}_i$ for $\mathbf{c}$

  $\mathbf{r} = \mathbf{w}_i - W\mathbf{c}$

  $[|\rho|, \wp_i] = \max\{|\mathbf{r}|\}$

  $W \leftarrow [W\ \mathbf{w}_i], \mathcal{P} \leftarrow [\mathcal{P}\ \mathbf{e}_{\wp_i}], \vec{\wp} \leftarrow \begin{bmatrix} \vec{\wp} \\ \wp_i \end{bmatrix}$

end for

The product $\mathcal{P}^T\mathbf{F}$ is a selection of entries

Let $m = 3$. Suppose the DEIM-algorithm has chosen indices $\wp_1, \ldots, \wp_m$ such that:

$$\mathcal{P}^T\mathbf{F} = \begin{bmatrix} 0 & \ldots & \overset{\wp_1}{\downarrow} 1 & \ldots & 0 \\ 0 & 1 & \ldots & \ldots & 0 \\ 0 & \ldots & 0 & \underset{\wp_m}{\uparrow} 1 & 0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_N \end{bmatrix} = \begin{bmatrix} F_{\wp_1} \\ F_{\wp_2} \\ F_{\wp_m} \end{bmatrix}$$

Assuming that $\mathbf{F}(\cdot)$ acts pointwise, we obtain:

$$\begin{aligned} \mathbf{N} &\approx U_\ell^T W (\mathcal{P}^T W)^{-1} \mathcal{P}^T \mathbf{F}(t, U_\ell \tilde{\mathbf{y}}(t)) \\ &= \underbrace{U_\ell^T W (\mathcal{P}^T W)^{-1}}_{\ell \times m} \underbrace{\mathbf{F}(t, \mathcal{P}^T U_\ell \tilde{\mathbf{y}}(t))}_{m \times 1} \end{aligned}$$

The nonlinear 1D Burgers' model

$$y_t + \left(\frac{1}{2}y^2 - \nu y_x\right)_x = f, \quad (x,t) \in (0,L)\times(0,T),$$
$$y(t,0) = y(t,L) = 0, \quad t \in (0,T),$$
$$y(0,x) = y_0(x), \quad x \in (0,L).$$

1. FEM-discretization in space leads to:

$$M\dot{\mathbf{y}}(t) = -\frac{1}{2}B\mathbf{y}^2(t) - \nu C\mathbf{y}(t) + \mathbf{f}, \quad t > 0$$
$$\mathbf{y}(0) = \mathbf{y}_0$$

2. Time integration via implicit Euler + Newton's method

POD-DEIM for Burgers' equation

Suppose, Φ_ℓ is an M-orthogonal POD basis.

The POD reduced Burgers' equation

$$\overbrace{\Phi_\ell^T M \Phi_\ell}^{=I_\ell} \dot{\tilde{y}}(t) = -\frac{1}{2}\Phi_\ell^T B(\Phi_\ell \tilde{\mathbf{y}}(t))^2 - \nu \Phi_\ell^T C \Phi_\ell \tilde{\mathbf{y}}(t)$$

$$\Rightarrow \quad \dot{\tilde{y}}(t) = -\frac{1}{2} B_\ell(\Phi_\ell \tilde{\mathbf{y}}(t))^2 - \nu C_\ell \tilde{\mathbf{y}}(t)$$

Next, obtain W via a truncated SVD of $[\mathbf{y}^2(t_1), \ldots, \mathbf{y}^2(t_{n_s})]$ and apply DEIM to the columns of W.

The POD-DEIM reduced Burgers' equation

$$\dot{\tilde{y}}(t) = -\frac{1}{2}\tilde{B}(\tilde{F}\tilde{\mathbf{y}}(t))^2 - \nu \tilde{C}\tilde{\mathbf{y}}(t),$$

with $\tilde{B} = \Phi_\ell^T B W (\mathcal{P}^T W)^{-1} \in \mathbb{R}^{\ell \times m}$, $\tilde{F} = \mathcal{P}^T \Phi_\ell \in \mathbb{R}^{m \times \ell}$, and $\tilde{C} = C_\ell \in \mathbb{R}^{\ell \times \ell}$.

POD-DEIM for Burgers' equation

Suppose, Φ_ℓ is an M-orthogonal POD basis.

The POD reduced Burgers' equation

$$\overbrace{\Phi_\ell^T M \Phi_\ell}^{=I_\ell} \dot{\tilde{y}}(t) = -\frac{1}{2}\Phi_\ell^T B(\Phi_\ell \tilde{\mathbf{y}}(t))^2 - \nu \Phi_\ell^T C \Phi_\ell \tilde{\mathbf{y}}(t)$$

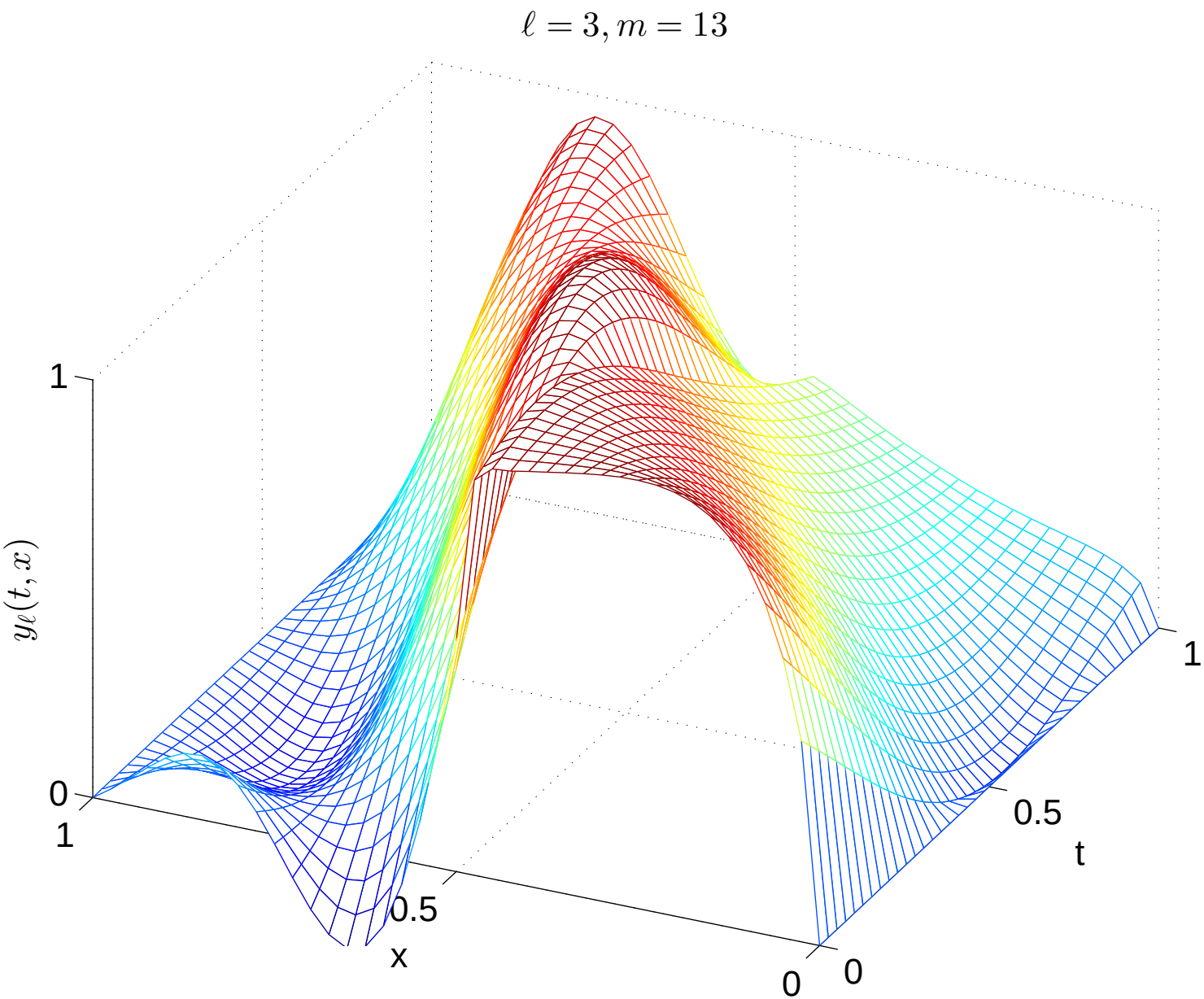
$$\Rightarrow \quad \dot{\tilde{y}}(t) = -\frac{1}{2} B_\ell(\Phi_\ell \tilde{\mathbf{y}}(t))^2 - \nu C_\ell \tilde{\mathbf{y}}(t)$$

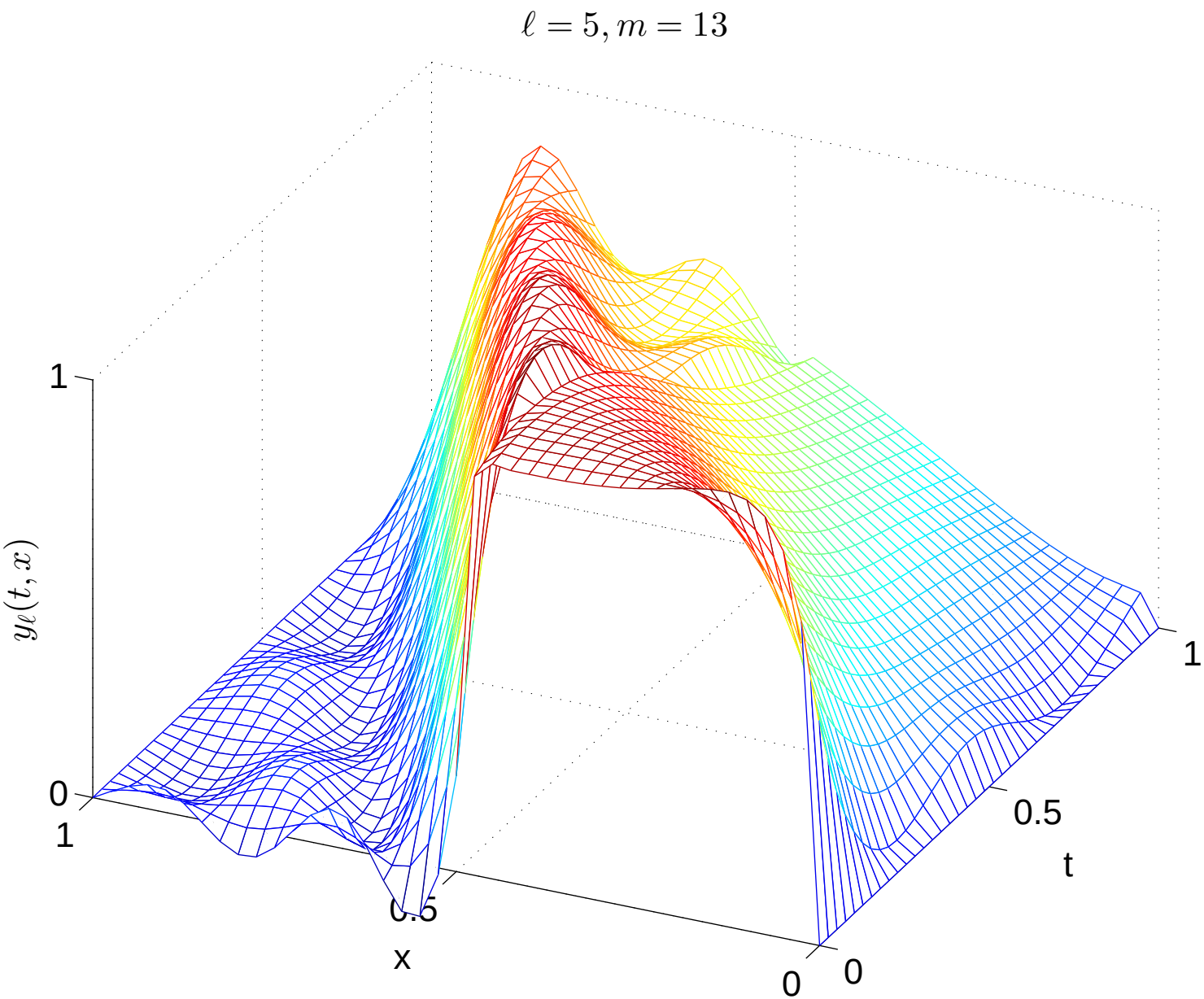
Next, obtain W via a truncated SVD of $[\mathbf{y}^2(t_1), ..., \mathbf{y}^2(t_{n_s})]$ and apply DEIM to the columns of W.

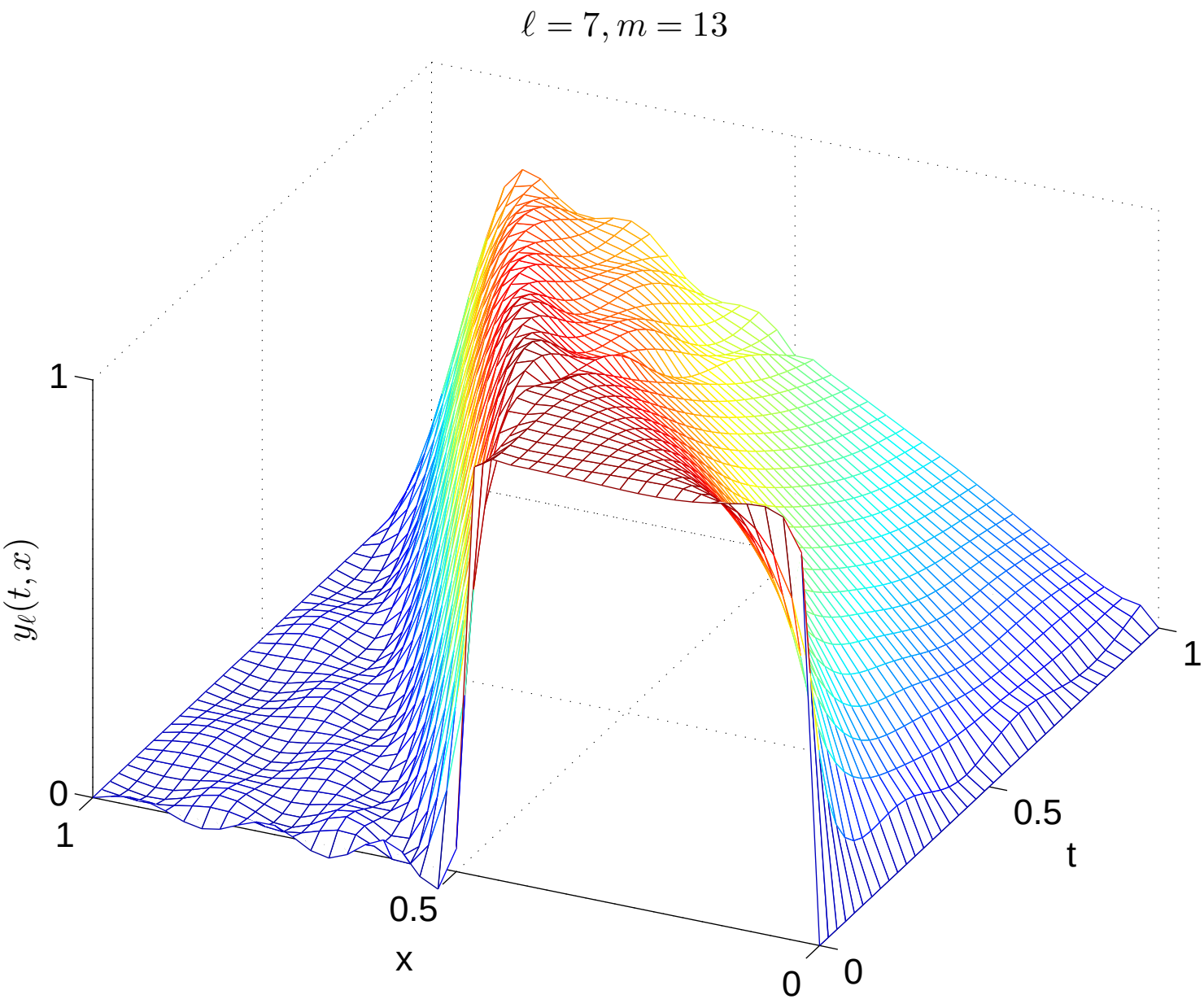
The POD-DEIM reduced Burgers' equation

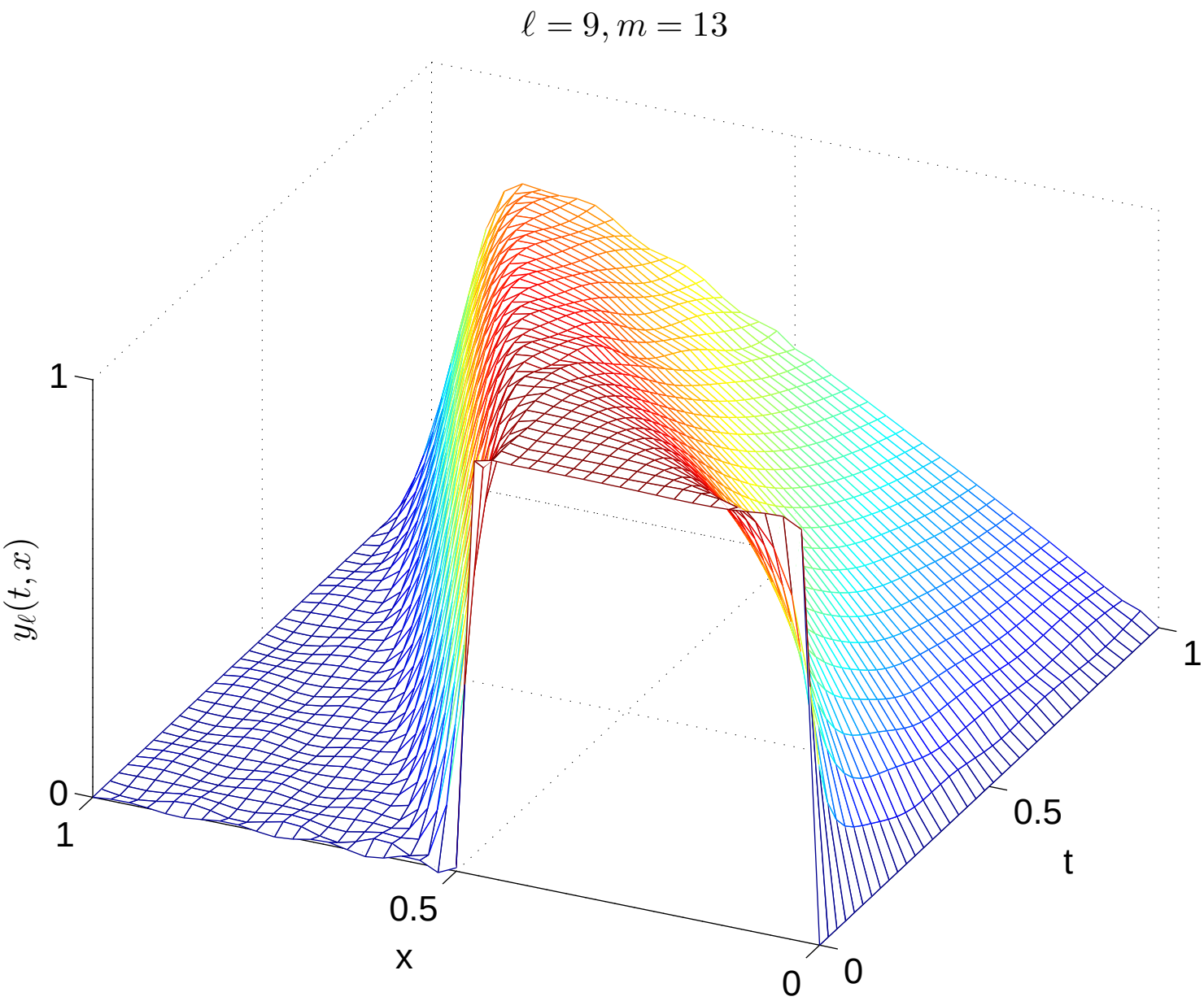
$$\dot{\tilde{y}}(t) = -\frac{1}{2}\tilde{B}(\tilde{F}\tilde{\mathbf{y}}(t))^2 - \nu \tilde{C}\tilde{\mathbf{y}}(t),$$

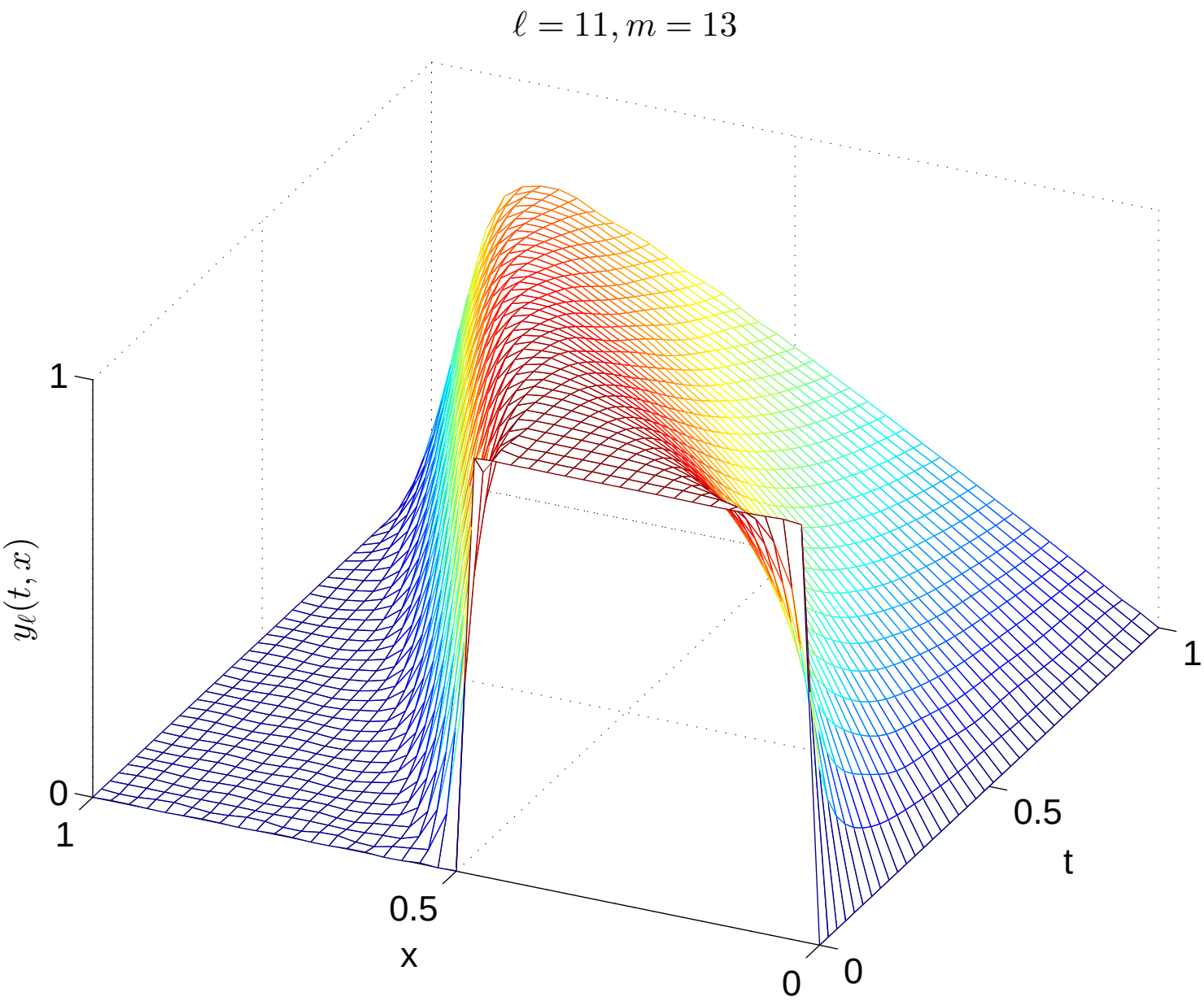
with $\tilde{B} = \Phi_\ell^T B W (\mathcal{P}^T W)^{-1} \in \mathbb{R}^{\ell \times m}$, $\tilde{F} = \mathcal{P}^T \Phi_\ell \in \mathbb{R}^{m \times \ell}$, and $\tilde{C} = C_\ell \in \mathbb{R}^{\ell \times \ell}$.


$\ell = 3, m = 13$
$y_\ell(t, x)$
1
0
1
0.5
x
0
0
0.5
t
1


$\ell = 5, m = 13$
$y_\ell(t, x)$
1
0
1
x
0.5
0
0
t
0.5
1


$\ell = 7, m = 13$
$y_\ell(t, x)$
1
0
1
0.5
x
0
0
0.5
t
1

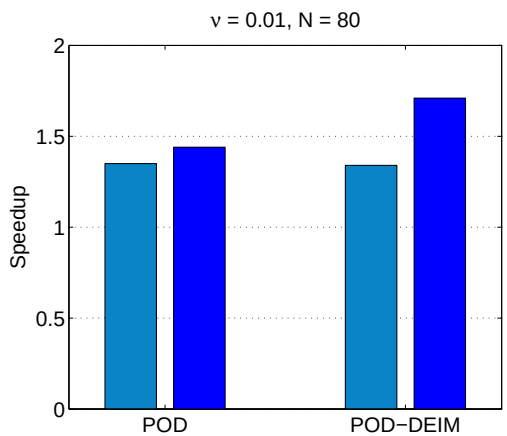

$\ell = 9, m = 13$
$y_\ell(t, x)$
1
0
1
0.5
x
0
0
0.5
t
1

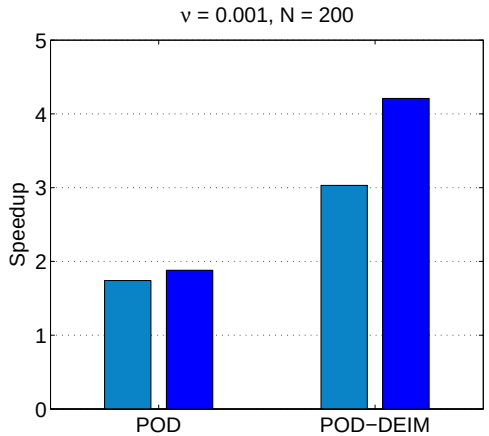

$\ell = 11, m = 13$
$y_\ell(t,x)$
1
0
1
0.5
x
0
0
0.5
t
1

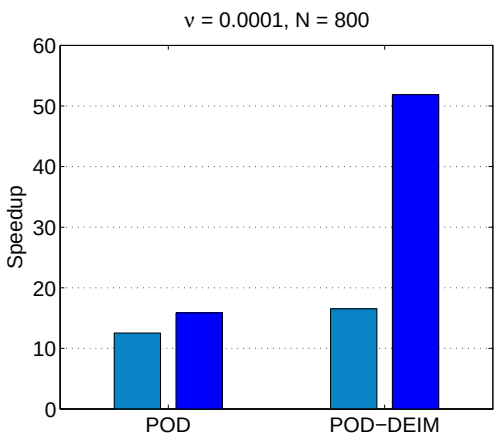
Computational Speedup [1]

Conclusion: High accuracy of the POD-DEIM reduced model.

But is it also faster?



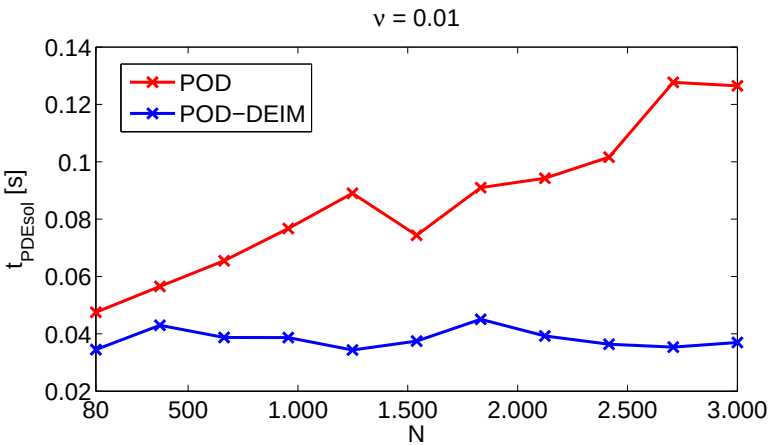


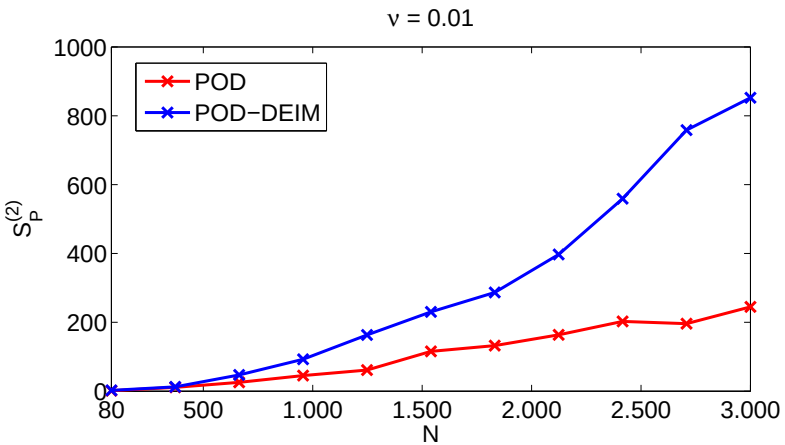


- Spatial discretization of the full model depends on viscosity parameter ν
- choose ℓ, m such that relative L_2-error in $\mathcal{O}(10^{-4})$

Computational Speedup [2]

For a fixed $\nu = 0.01$, we could show the independence of the POD-DEIM reduced model of the full-order dimension N.





- Computation time for solving the POD-DEIM reduced Burgers' equation is almost constant (right)
- POD-DEIM almost 4 times faster than pure POD (left)

PDE-constrained optimization

Minimize

$$\min_u \mathcal{J}(y(u), u),$$

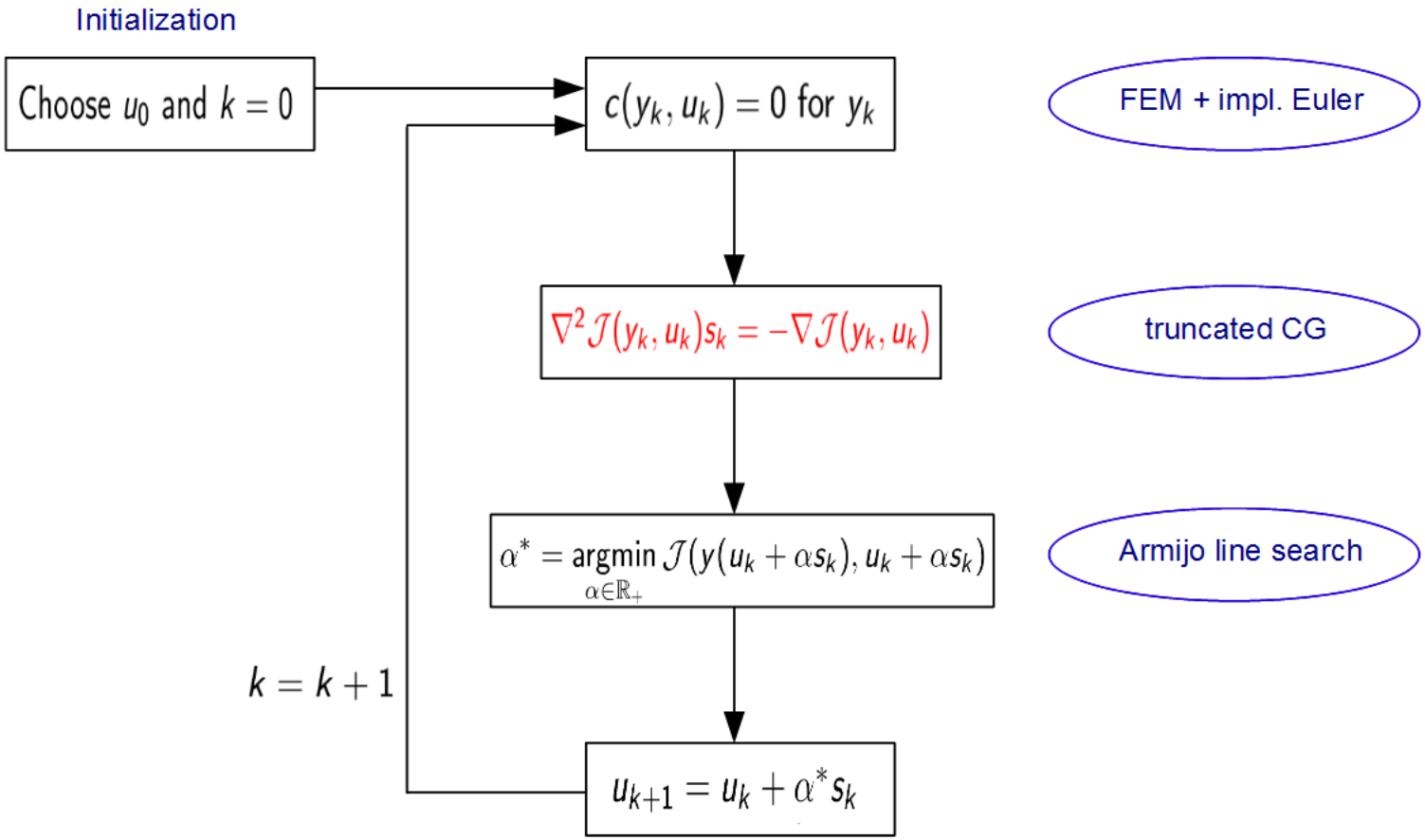
where y is the solution to a nonlinear, possibly time-dependent partial differential equation,

$$c(y, u) = 0.$$

- $\mathcal{J}$ is called objective function,
- in order to evaluate $\mathcal{J}$, we need to solve $c(y, u) = 0$ for $y(u)$,
- solve with algorithms for unconstrained minimization problems.

A Newton-type optimization algorithm

Minimize $\mathcal{J}(y(u), u)$ in u using information of the first and second derivative.



Gradient computation via adjoints

Consider the Lagrangian function

$$\mathcal{L}(y, u, \lambda) = \mathcal{J}(y, u) + \lambda^T c(y, u)$$

and **impose** the zero-gradient condition $\nabla_y \mathcal{L}(y, u, \lambda) = 0$.

We derive the *adjoint equation*:

$$c_y(y(u), u)^T \lambda = -\nabla_y \mathcal{J}(y(u), u)$$

Algorithm 2 Computing $\nabla \hat{\mathcal{J}}(u)$ via adjoints [Heinkenschloss, 2008]

1: For a given control u, solve $c(y, u) = 0$ for the state $y(u)$
2: Solve the adjoint equation $c_y(y(u), u)^T \lambda = -\nabla_y \mathcal{J}(y(u), u)$ for $\lambda(u)$
3: Compute $\nabla \hat{\mathcal{J}}(u) = \nabla_u \mathcal{J}(y(u), u) + c_u(y(u), u)^T \lambda(u)$

Gradient computation via adjoints

Consider the Lagrangian function

$$\mathcal{L}(y, u, \lambda) = \mathcal{J}(y, u) + \lambda^T c(y, u)$$

and **impose** the zero-gradient condition $\nabla_y \mathcal{L}(y, u, \lambda) = 0$.

We derive the *adjoint equation*:

$$c_y(y(u), u)^T \lambda = -\nabla_y \mathcal{J}(y(u), u)$$

Algorithm 3 Computing $\nabla \hat{\mathcal{J}}(u)$ via adjoints [Heinkenschloss, 2008]

1: For a given control u, solve $c(y, u) = 0$ for the state $y(u)$
2: Solve the adjoint equation $c_y(y(u), u)^T \lambda = -\nabla_y \mathcal{J}(y(u), u)$ for $\lambda(u)$
3: Compute $\nabla \hat{\mathcal{J}}(u) = \nabla_u \mathcal{J}(y(u), u) + c_u(y(u), u)^T \lambda(u)$

First-order optimization algorithms

Instead of solving

$$\nabla^2 \mathcal{J}(y_k, u_k) s_k = -\nabla \mathcal{J}(y_k, u_k),$$

first-order methods approximate the Hessian via H_k and solve

$$H_k s_k = -\nabla \mathcal{J}(y_k, u_k).$$

- We have used Matlab implementations of the BFGS and the SPG method,
- Evaluation of $\mathcal{J}$ and gradient computation as seen before,
- SPG easily allows to include bounds on the control, i.e. $u_{lower} \leq u(t,x) \leq u_{upper}$ which is used in many applications

Optimal Control problem for Burgers' equation

Minimize

$$\min_{u} \frac{1}{2} \int_0^T \int_0^L [y(t,x) - z(t,x)]^2 + \omega u^2(t,x) \, dx \, dt,$$

where y is a solution to the nonlinear Burgers' equation

$$\begin{aligned} y_t + \left(\frac{1}{2} y^2 - \nu y_x \right)_x &= f + u, \quad (x,t) \in (0,L) \times (0,T), \\ y(t,0) = y(t,L) &= 0, \quad t \in (0,T), \\ y(0,x) &= y_0(x), \quad x \in (0,L). \end{aligned}$$

- u is the control that determines y
- z is the desired state

Control goal

We want to control the solution of Burgers' equation in such a way that it stays in the desired state $z(t,\cdot) = y_0, \forall t$:

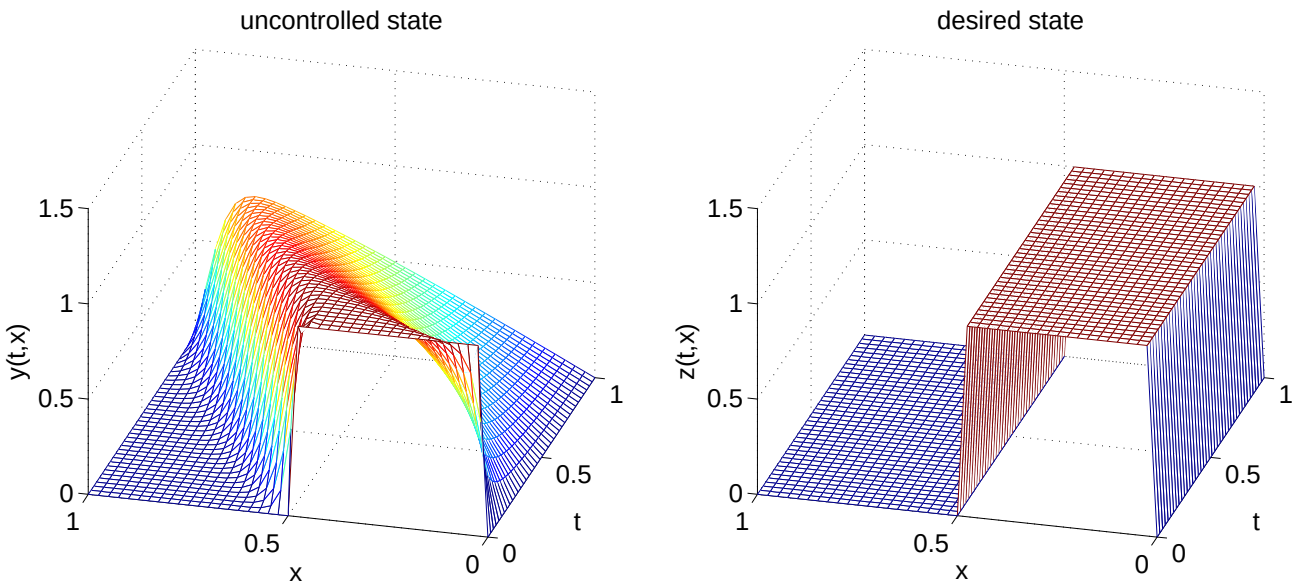


Figure: Uncontrolled ($u \equiv 0$) and desired state for $\nu = 0.01$.

Numerical treatment

1. Discretize the objective function and Burgers' equation in time and space
2. Apply adjoints in order to compute gradient and Hessian
3. Apply first-order or second-order optimization algorithm
4. Explore the usage of a POD-DEIM reduced model

Numerical treatment

1. Discretize the objective function and Burgers' equation in time and space
2. Apply adjoints in order to compute gradient and Hessian
3. Apply first-order or second-order optimization algorithm
4. Explore the usage of a POD-DEIM reduced model

Numerical treatment

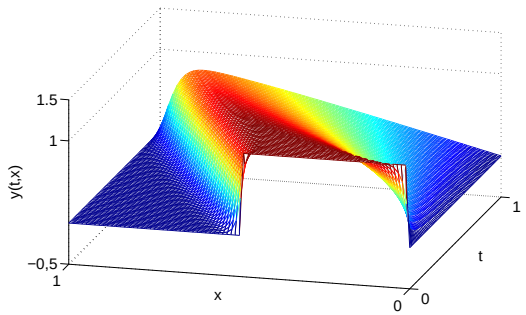
1. Discretize the objective function and Burgers' equation in time and space
2. Apply adjoints in order to compute gradient and Hessian
3. Apply first-order or second-order optimization algorithm
4. Explore the usage of a POD-DEIM reduced model

Numerical treatment

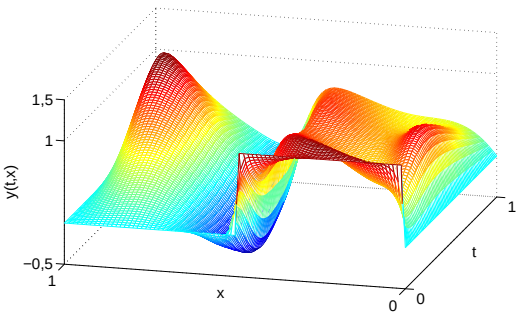
1. Discretize the objective function and Burgers' equation in time and space
2. Apply adjoints in order to compute gradient and Hessian
3. Apply first-order or second-order optimization algorithm
4. Explore the usage of a POD-DEIM reduced model

Numerical tests

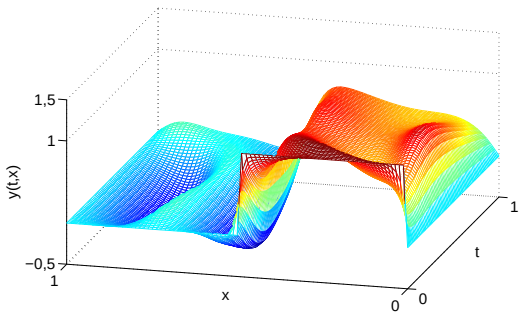
Newton-type method for the full-order Burgers' model:



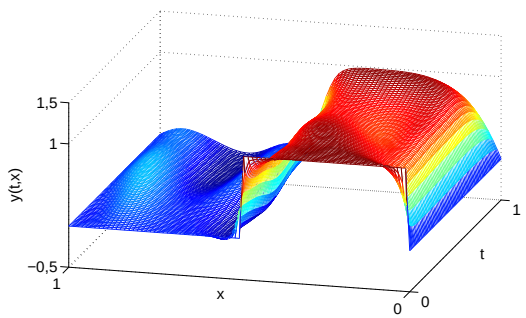
$k = 0$ (uncontrolled)



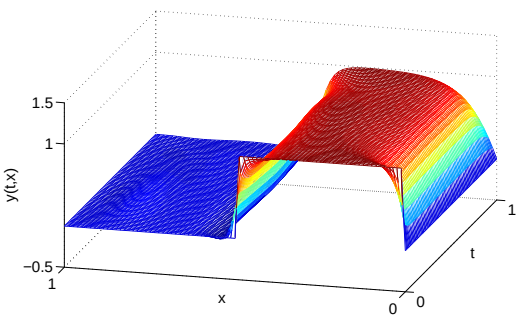
$k = 1$



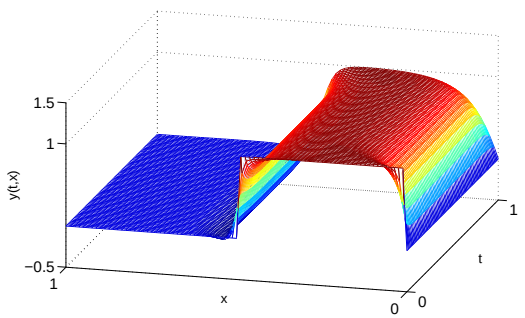
$k = 2$



$k = 3$

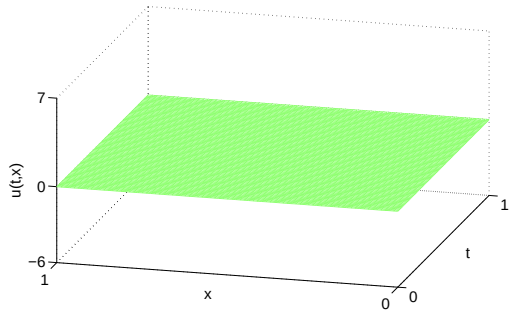


$k = 4$

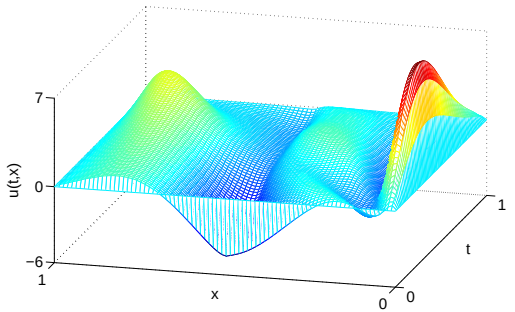


$k = 5$

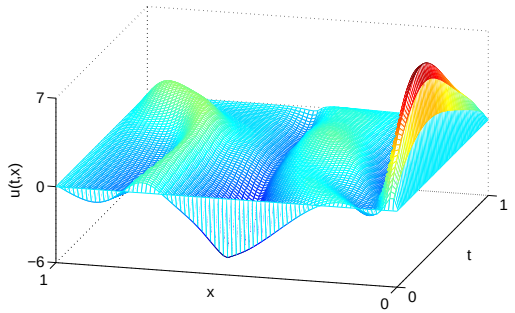
The corresponding optimal control at each iteration:



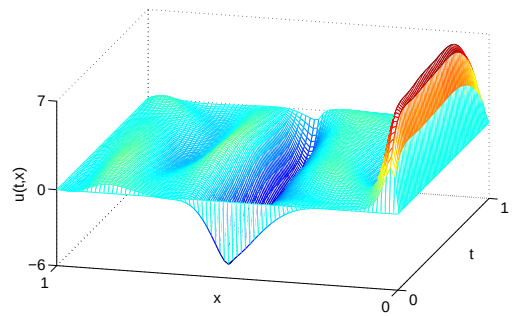
$k = 0$ (initial)



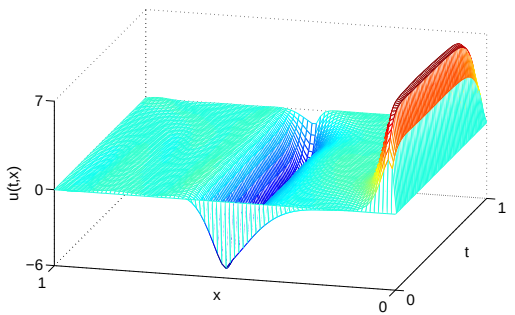
$k = 1$



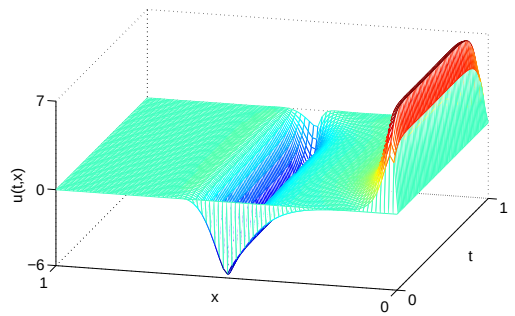
$k = 2$



$k = 3$

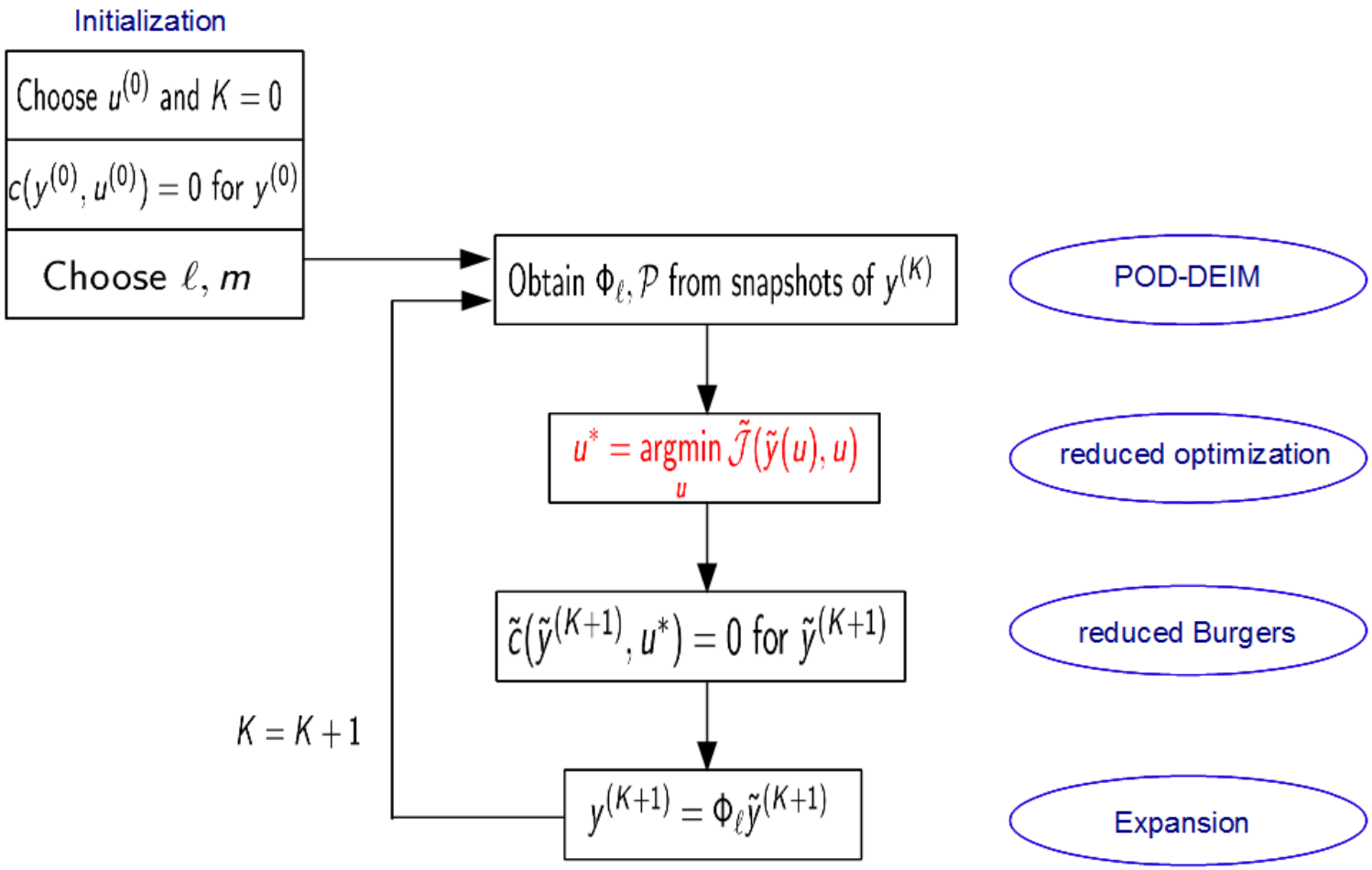


$k = 4$

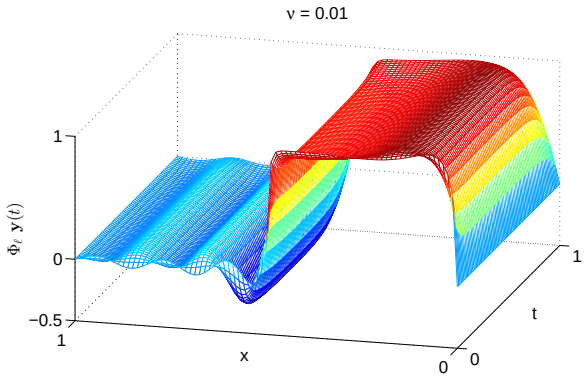


$k = 5$

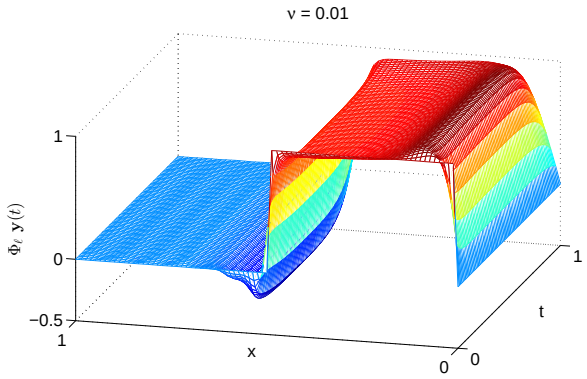
We propose the following algorithm for POD-DEIM reduced optimal control :



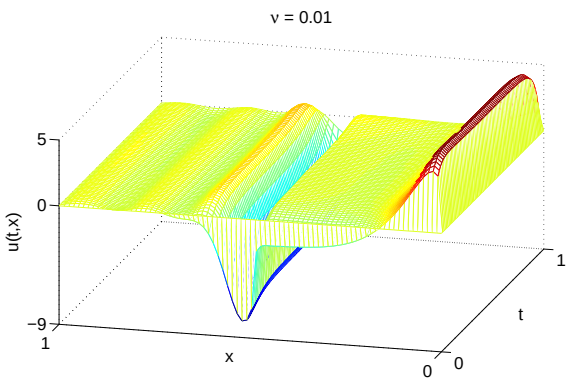
Final state and control of the POD-DEIM reduced optimal control problem:



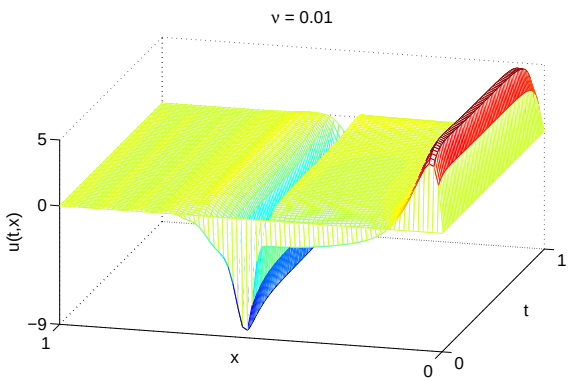
$\ell = m = 7$



$\ell = m = 15$



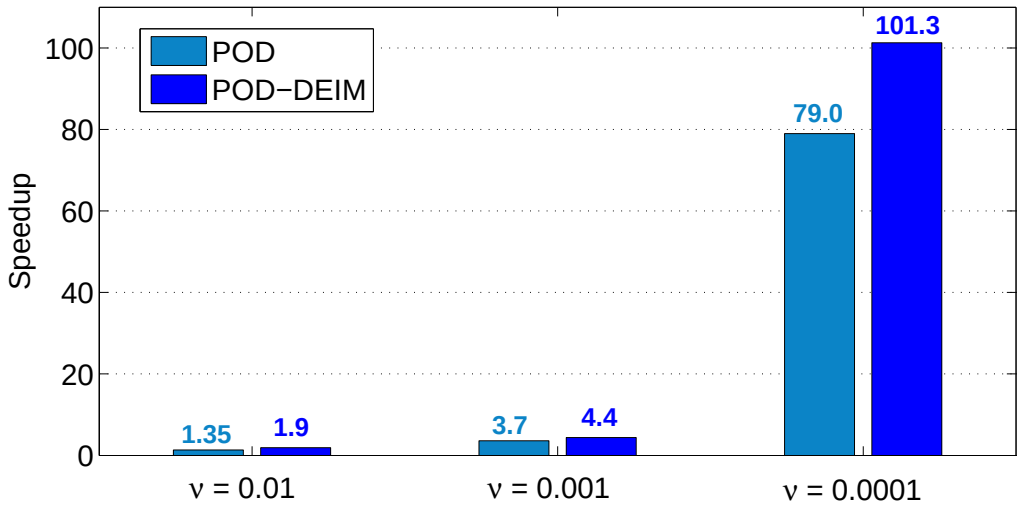
$\ell = m = 7$



$\ell = m = 15$

Computational Speedup [3]

Reduced optimal control using the Newton-type method:

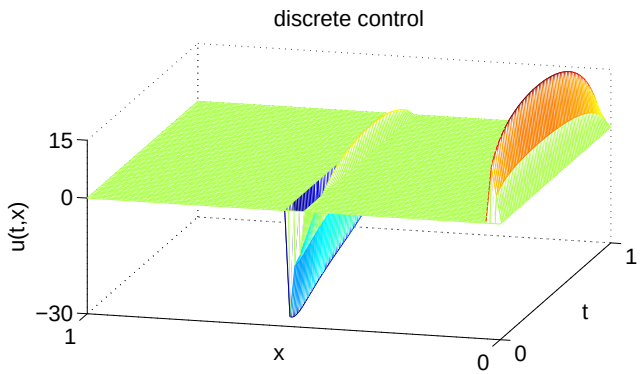


- at final state: relative L_2-error in $\mathcal{O}(10^{-2})$
- comparable value of the objective function at convergence
- use same stopping criteria for full-order and reduced model

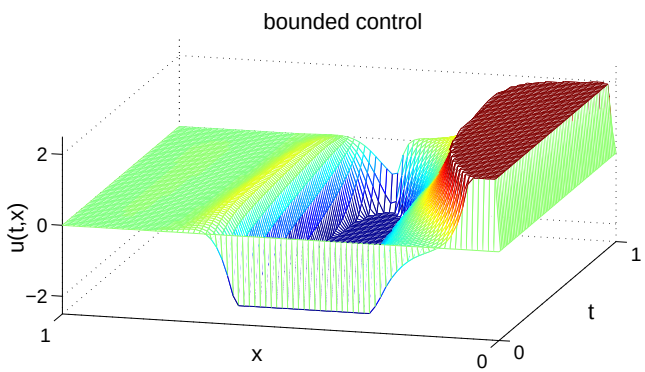
Computational Speedup [4]

Some other results.

For $\nu = 0.0001$, low-dimensional control leads to a speedup of ~ 20 for all three optimization methods.



SPG allows a bounded control $-2 \leq u(t,x) \leq 2$. For $\nu = 0.0001$, we obtained a speedup of 3.6 for POD and 8.8 for POD-DEIM.



Concluding Remarks

What I learnt:

- The accuracy of the reduced Burgers' model is of the same order when POD is extended by DEIM.
- Optimal Control of Burgers' equation using POD-DEIM leads to a speedup of ~ 100 for small ν.
- For the reduced model, all derivatives need to be computed in terms of the reduced variable. This can be quite hard in practice.

Future Research

What I still want to learn:

- Use the POD basis Φ_ℓ also for dimension reduction of the control, i.e.

$$\mathbf{u}(t) \approx \Phi_\ell \tilde{\mathbf{u}}(t) = \sum_{i=1}^{\ell} \varphi_i \tilde{u}_i(t)$$

- Extend Burgers' model to 2D/3D
- More sophisticated choice of reduced dimensions ℓ and m

This Master project was supervised by
Marielba Rojas and Martin van Gijzen.

Thank you for your attention!

Are there any questions or remarks?

https://github.com/ManuelMBaumann/MasterThesis

This Master project was supervised by
Marielba Rojas and Martin van Gijzen.

Thank you for your attention!

Are there any questions or remarks?

`https://github.com/ManuelMBaumann/MasterThesis`

Further information can be found in...

S. Chaturantabut and D. Sorensen
Nonlinear Model Reduction via Discrete Empirical Interpolation.
SIAM Journal of Scientific Computing, 2010.

M. Heinkenschloss
Numerical solution of implicitly constrained optimization problems.
Technical report, Department of Computational and Applied Mathematics, Rice University, 2008.

K. Kunisch and S. Volkwein
Control of the Burgers Equation by a Reduced-Order Approach Using Proper Orthogonal Decomposition.
Journal of Optimization Theory and Applications, 1999.