Nonlinear Model Order Reduction for Optimal Control of Burgers' Equation

- Master thesis -

Manuel M. Baumann

July 15, 2013

Table of Contents

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 the nonlinear part
- 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 the nonlinear part
- 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 the nonlinear part
- 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 the nonlinear part
- 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 consists of 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 $\ell \times \ell$ system of equations:

$$U_\ell^T (U_\ell \dot{\tilde{\mathbf{y}}} - A U_\ell \tilde{\mathbf{y}} - \mathbf{F}(t, U_\ell \tilde{\mathbf{y}})) = 0$$
$$\Rightarrow \quad \dot{\tilde{\mathbf{y}}} = \underbrace{U_\ell^T A U_\ell}_{=:\tilde{A}} \tilde{\mathbf{y}} + U_\ell^T \mathbf{F}(t, U_\ell \tilde{\mathbf{y}})$$

Two questions are left...

Considering the reduced model

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

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

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

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

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

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 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 projection $\mathcal{P}$ 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 projection $\mathcal{P}$ 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:

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

Therefore, suppose $\mathbf{F}(\cdot)$ 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

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

FEM-discretization in space leads to:

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

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

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

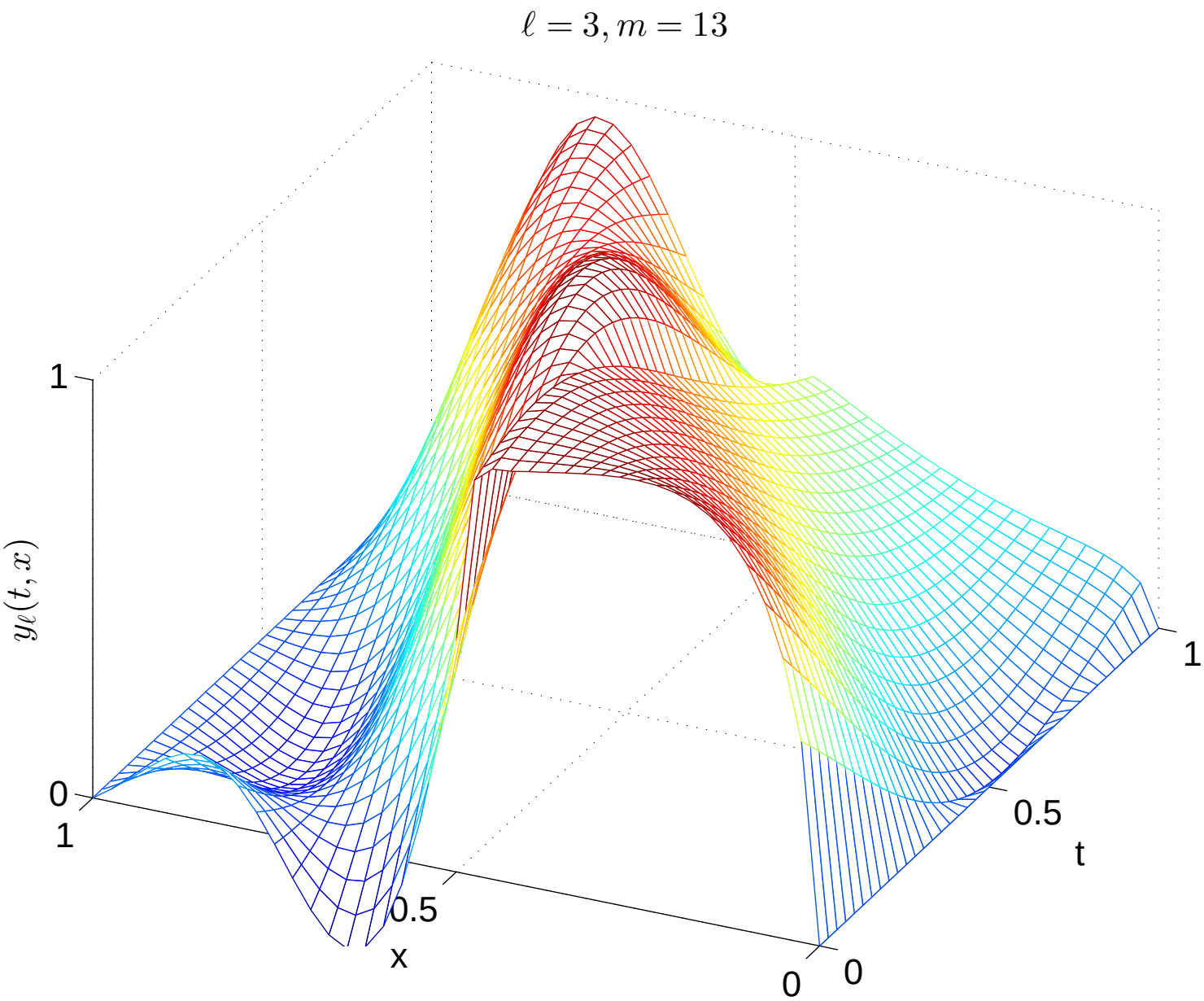
$$\Rightarrow \quad \dot{\tilde{y}}(t) = -\frac{1}{2} B_\ell(\Phi_\ell \tilde{\mathbf{y}}(t))^2 - \nu C_\ell \tilde{\mathbf{y}}(t)$$

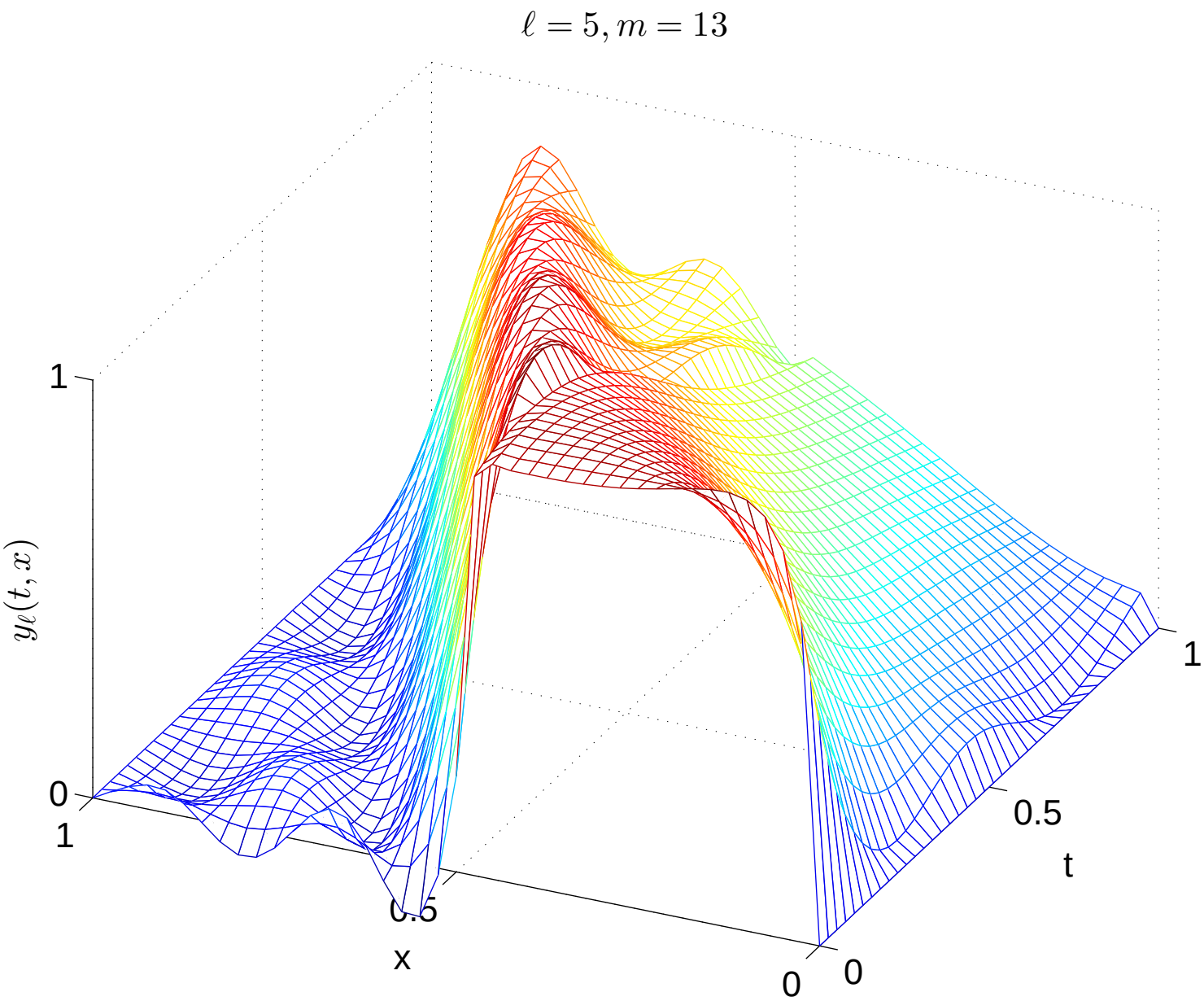
Next, obtain W via a truncated SVD of $[\mathbf{y}^2(t_1), ..., \mathbf{y}^2(t_{n_s})]$ and apply DEIM to the columns of W.

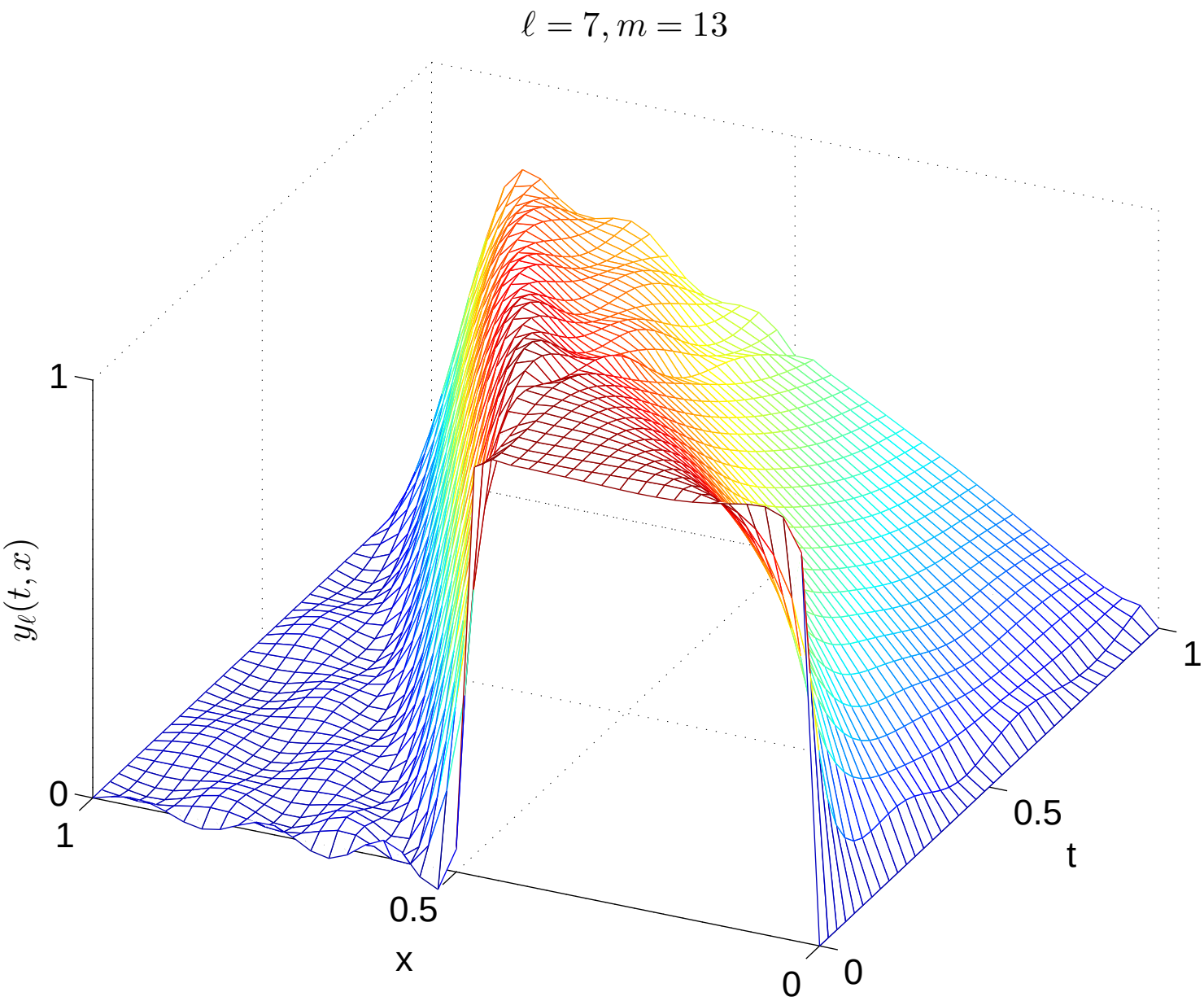
The POD-DEIM reduced Burgers' equation

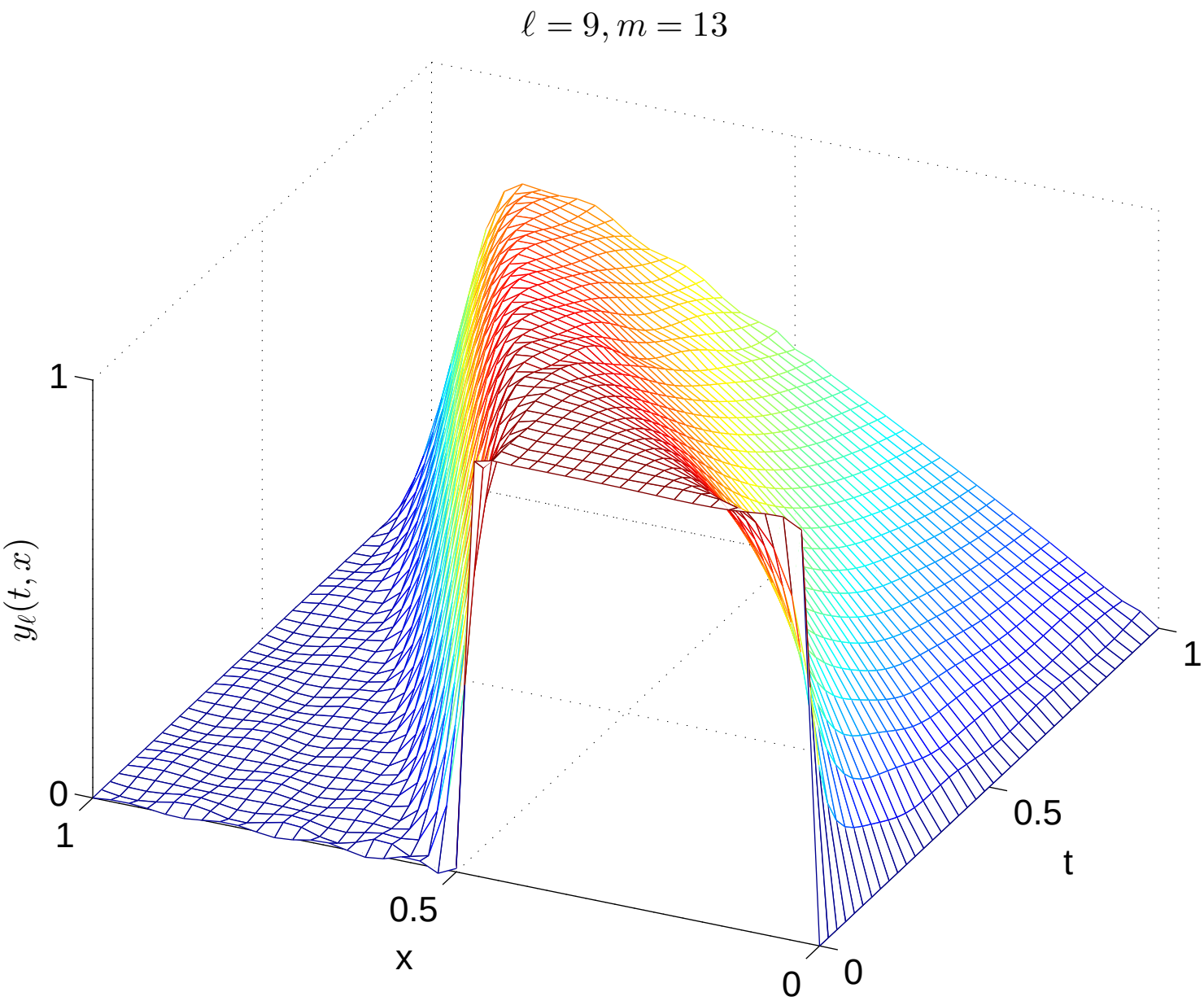
$$\dot{\tilde{y}}(t) = -\frac{1}{2}\tilde{B}(\tilde{F}\tilde{\mathbf{y}}(t))^2 - \nu \tilde{C}\tilde{\mathbf{y}}(t),$$

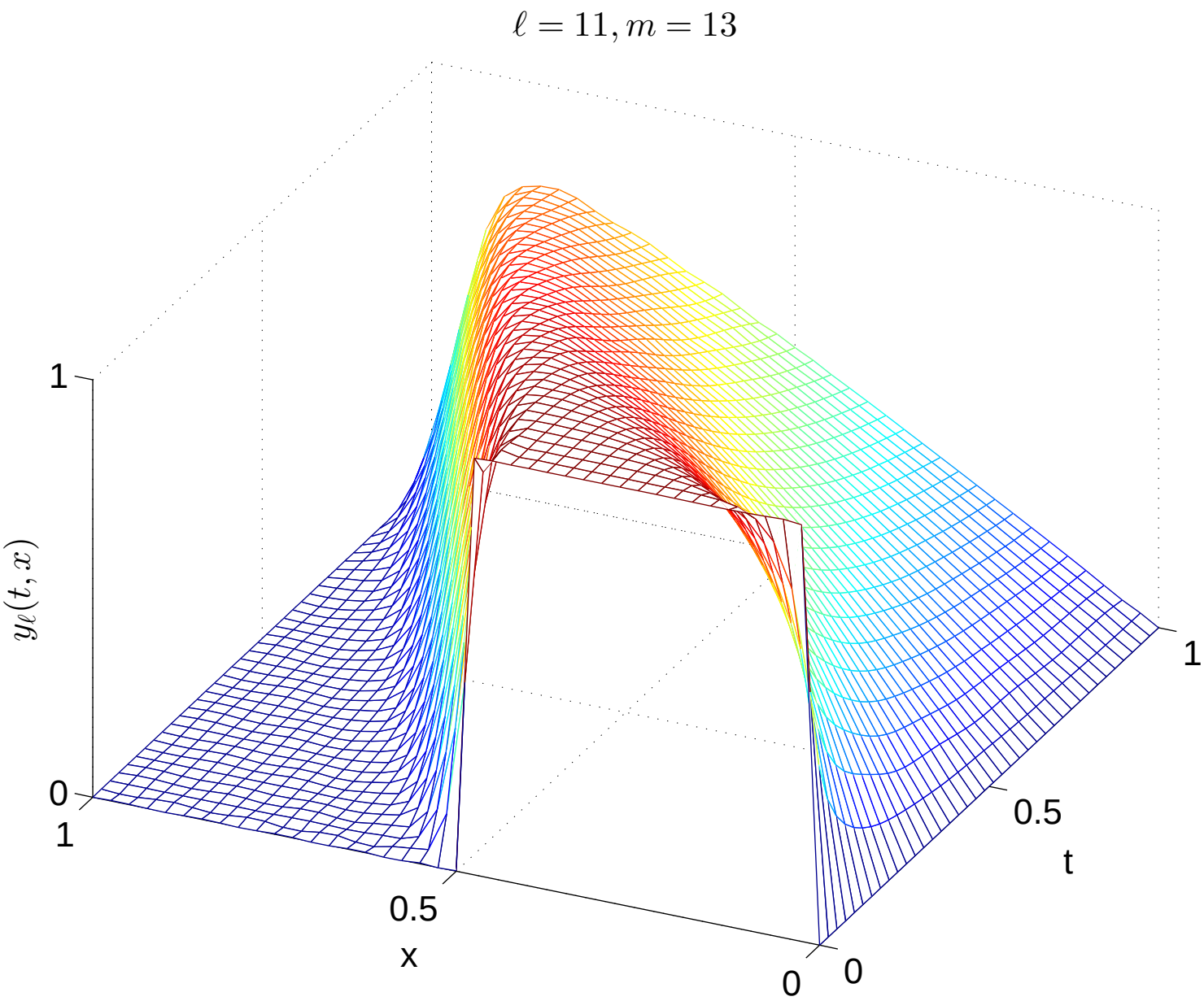
with $\tilde{B} = \Phi_\ell^T B W (\mathcal{P}^T W)^{-1} \in \mathbb{R}^{\ell \times m}$, $\tilde{F} = \mathcal{P}^T \Phi_\ell \in \mathbb{R}^{m \times \ell}$, and $\tilde{C} = C_\ell \in \mathbb{R}^{\ell \times \ell}$.


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

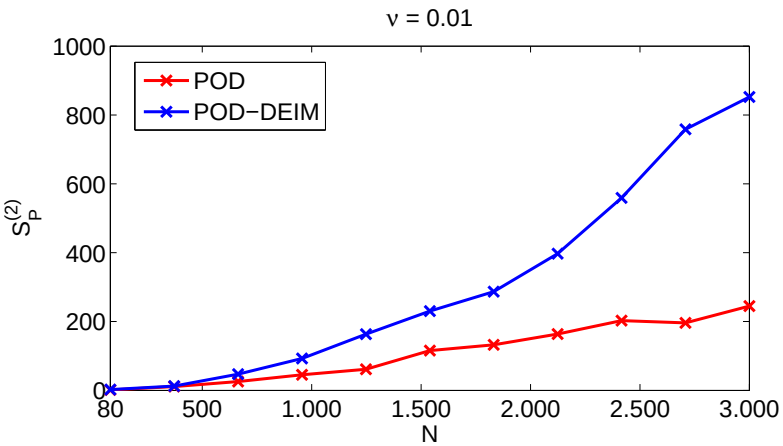

$\ell = 7, m = 13$
$y_\ell(t, x)$
1
0
1
0.5
x
0
0
0.5
t
1

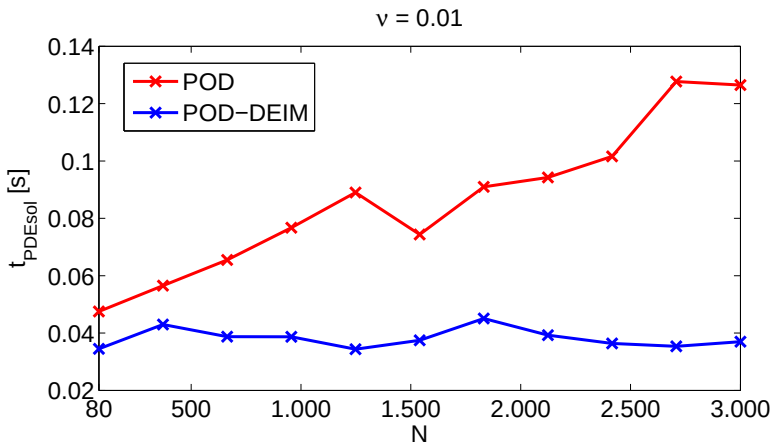

$\ell = 9, m = 13$
1
0
$y_\ell(t,x)$
1
0.5
x
0
0
0.5
t
1


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

Computational Speedup

	$\nu = 0.01$			$\nu = 0.001$			$\nu = 0.0001$		
	Full	POD	DEIM	Full	POD	DEIM	Full	POD	DEIM
$N/\ell/m$	80	11	(11, 13)	200	35	(35, 40)	800	40	(40, 55)
$t_{setup}[s]$	-	0.003	0.011	-	0.009	0.021	-	0.0729	0.182
$t_{PDEsol}[s]$	0.068	0.047	0.040	0.232	0.12	0.055	4.416	0.276	0.085
$\bar{e}[10^{-4}]$	-	6.13	6.46	-	6.12	6.85	-	7.43	7.66
$S_P^{(1)}$	-	1.35	1.34	-	1.74	3.03	-	12.54	16.52
$S_P^{(2)}$	-	1.44	1.71	-	1.88	4.21	-	15.85	51.85





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 $y(u)$,
- first-order and second-order optimization algorithms.

A second-order optimization algorithm

Minimize $\mathcal{J}(y(\cdot),\cdot)$ over u taking information of the first and second derivative into account.

- Choose u_0 and $k = 0$
- Repeat until convergence
  - Solve the PDE-constraint $c(y_k, u_k) = 0$ for y_k
  - Solve the Newton equation $\nabla^2 \mathcal{J}(y_k, u_k)s_k = -\nabla \mathcal{J}(y_k, u_k)$ via the **truncated CG-method**
  - Obtain $\alpha^* = \operatorname{argmin}_{\alpha \in \mathbb{R}_+} \mathcal{J}(y(u_k + \alpha s_k), u_k + \alpha s_k)$ via **Armijo line search**
  - Calculate new control $u_{k+1} = u_k + \alpha^* s_k$
  - Stop if $\|\nabla \mathcal{J}(y_k, u_k)\|$ is small or $\mathcal{J}(y_k, u_k)$ does not change anymore
  - $k = k + 1$

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 for 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

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

where y is a solution to the nonlinear Burgers' equation

$$\begin{aligned} y_t + \left(\frac{1}{2}y^2 - \nu y_x\right)_x &= f + u, \quad (x,t) \in (0,L) \times (0,T), \\ y(t,0) = y(t,L) &= 0, \quad t \in (0,T), \\ y(0,x) &= y_0(x), \quad x \in (0,L). \end{aligned} \tag{3}$$

- 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 $z(t, \cdot) = y_0, \forall t$:

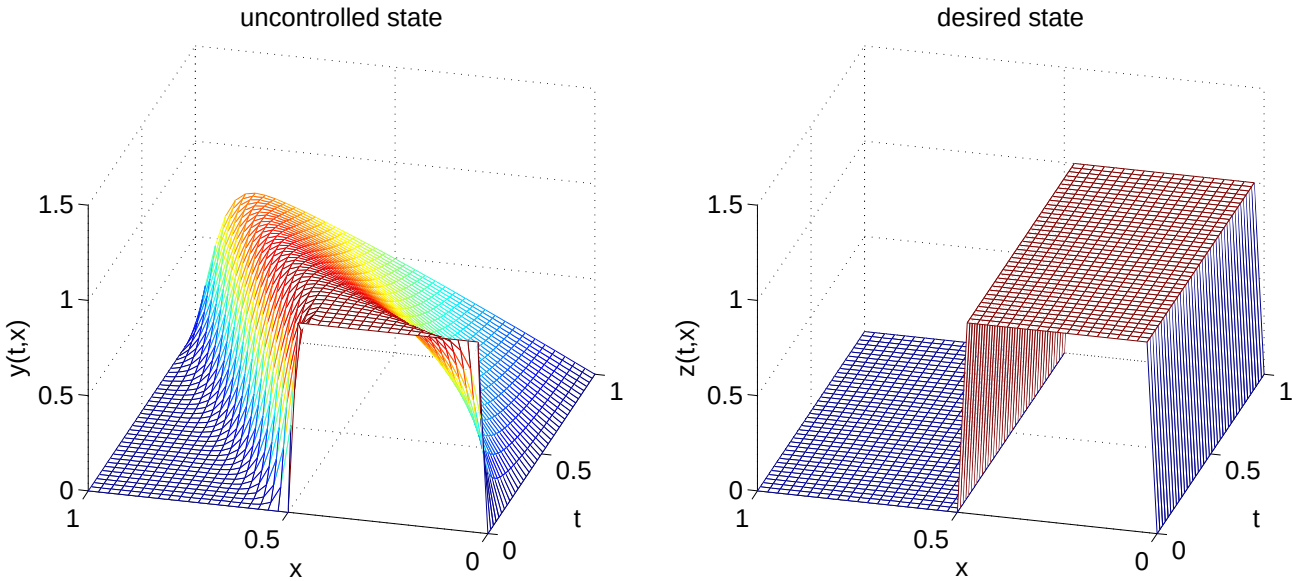


Figure: Uncontrolled and desired state for $\nu = 0.01$.

The full-order discretized optimal control problem

$$\min_u \mathcal{J}(y(u),u) \equiv \min_{\mathbf{u}_0,\dots,\mathbf{u}_{N_t}} \sum_{i=0}^{N_t} \delta t \left(\frac{1}{2}\mathbf{y}_i^T M \mathbf{y}_i - \mathbf{z}^T \mathbf{y}_i + \frac{\omega}{2}\mathbf{u}_i^T M \mathbf{u}_i \right),$$

where $\mathbf{y}_i$ is the solution of the full-order Burgers' equation

$$c(y,u) \equiv \frac{1}{\delta t} M \mathbf{y}_{i+1} - \frac{1}{\delta t} M \mathbf{y}_i + \frac{1}{2} B \mathbf{y}_{i+1}^2 + \nu C \mathbf{y}_{i+1} - \mathbf{f} - M \mathbf{u}_{i+1} = 0$$

The POD-DEIM reduced optimal control problem

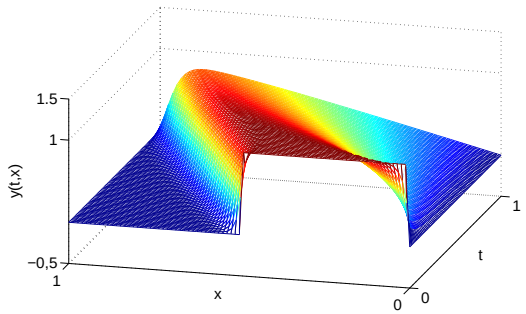
$$\min_u \tilde{\mathcal{J}}(\tilde{y}(u),u) \equiv \min_{\mathbf{u}_1,\dots,\mathbf{u}_{N_t}} \sum_{i=0}^{N_t} \delta t \left(\frac{1}{2}\tilde{\mathbf{y}}_i^T \tilde{\mathbf{y}}_i - \tilde{\mathbf{z}}^T \tilde{\mathbf{y}}_i + \frac{\omega}{2}\mathbf{u}_i^T M \mathbf{u}_i \right),$$

where $\tilde{\mathbf{y}}_i$ is the solution of the POD-DEIM reduced Burgers' equation

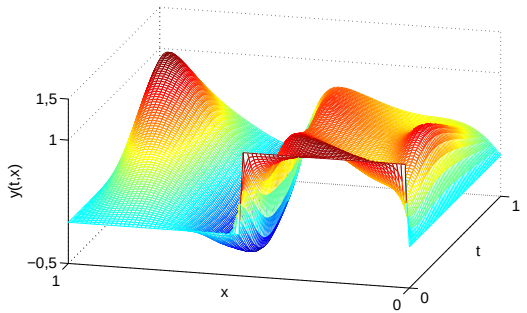
$$\tilde{c}(\tilde{y},u) \equiv \frac{1}{\delta t} \tilde{\mathbf{y}}_{i+1} - \frac{1}{\delta t} \tilde{\mathbf{y}}_i + \frac{1}{2} \tilde{B}(\tilde{F}\tilde{\mathbf{y}}_{i+1})^2 + \nu \tilde{C} \tilde{\mathbf{y}}_{i+1} - \tilde{\mathbf{f}} - \tilde{M} \mathbf{u}_{i+1} = 0$$

Numerical tests

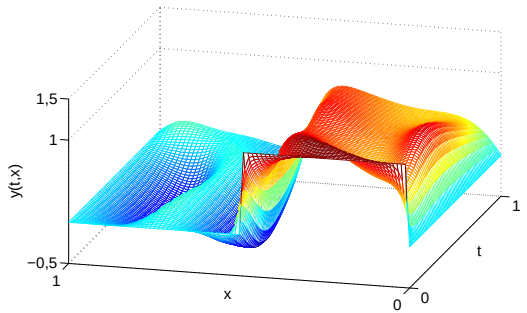
Newton-type method for the full-order Burgers' model:



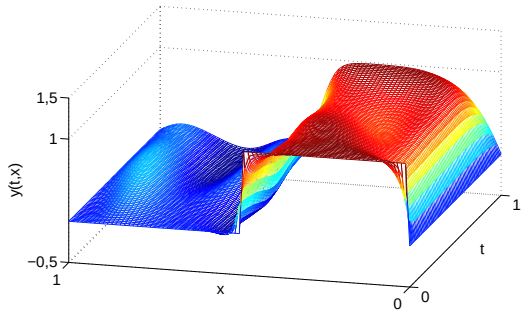
(a) $k = 0$ (uncontrolled)



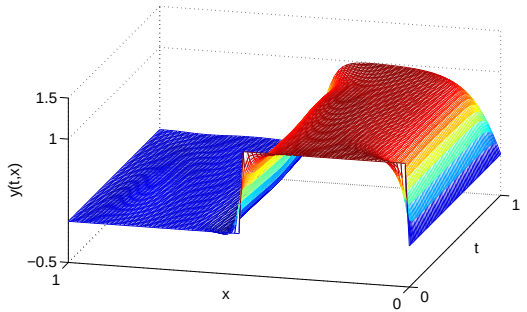
(b) $k = 1$



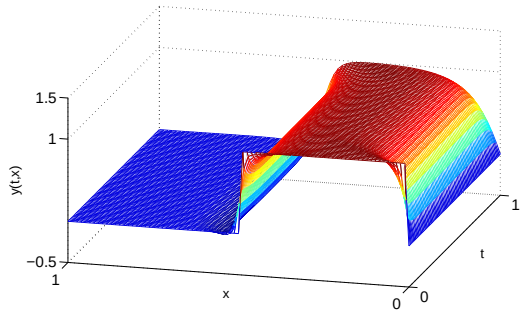
(c) $k = 2$



(d) $k = 3$

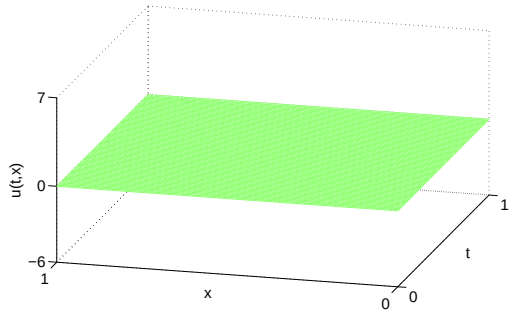


(e) $k = 4$

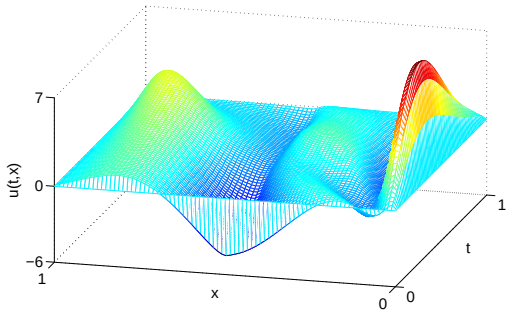


(f) $k = 5$

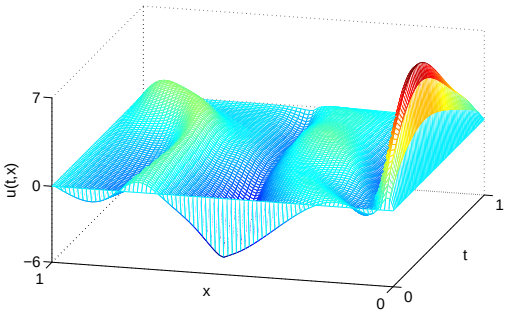
The corresponding optimal control at each iteration:



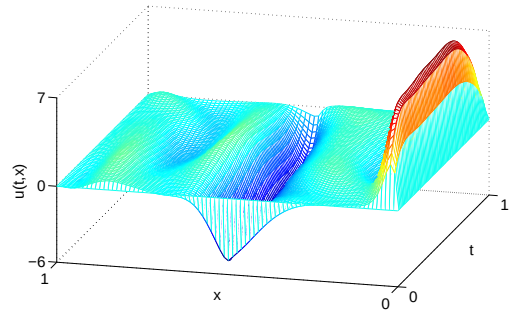
(g) $k = 0$ (initial)



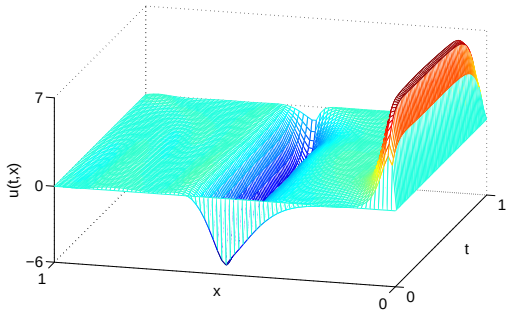
(h) $k = 1$



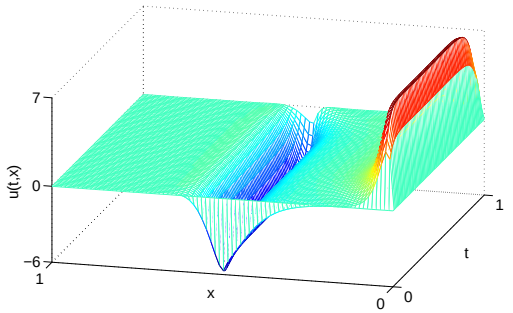
(i) $k = 2$



(j) $k = 3$



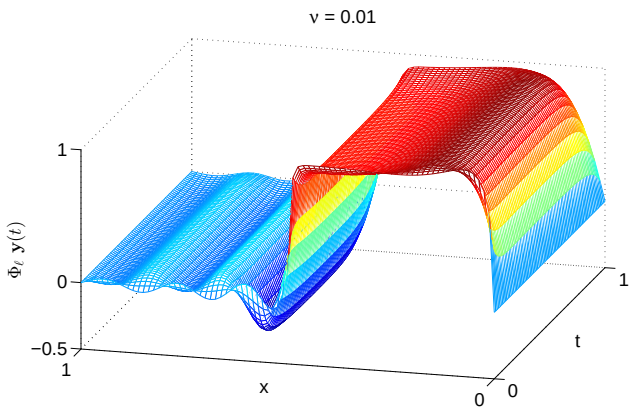
(k) $k = 4$



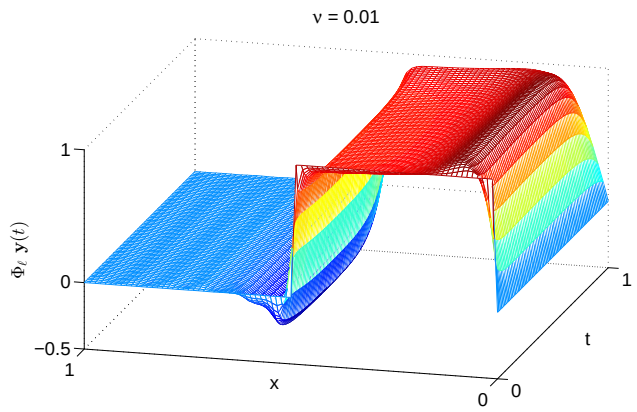
(l) $k = 5$

The following algorithm for POD-DEIM reduced optimal control has been developed:

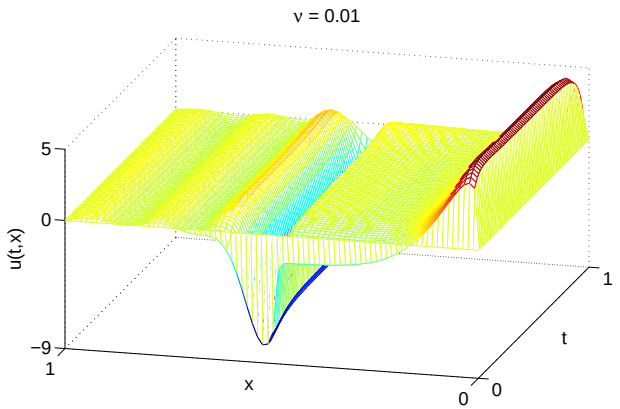
- Choose $u^{(0)}$ for the full-order model, $K = 0$
- Solve full-order Burgers' equation $c(y^{(0)}, u^{(0)}) = 0$ for $y^{(0)}$
- Choose POD-dimension ℓ and DEIM-dimension m
- Repeat until $\|y^{(K)} - z\|_{L_2(\Omega)} < tol$
  - Perform POD-DEIM algorithm using snapshots of $y^{(K)}$ $\hookrightarrow$ $\Phi_\ell, \mathcal{P}$
  - Solve $u^{(K+1)} = \operatorname{argmin}_u \tilde{\mathcal{J}}(\tilde{y}(u), u)$
  - Solve reduced Burgers' equation $\tilde{c}(\tilde{y}^{(K+1)}, u^{(K+1)}) = 0$ for $\tilde{y}^{(K+1)}$
  - Expand $y^{(K+1)} = \Phi_\ell \tilde{y}^{(K+1)}$
  - $K = K + 1$



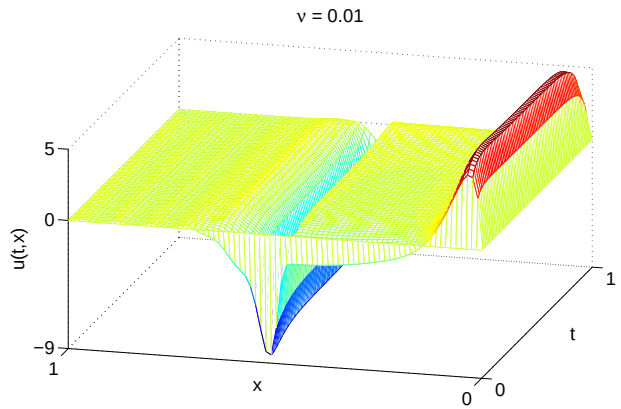
(m) $\ell = m = 7$



(n) $\ell = m = 15$



(o) $\ell = m = 7$



(p) $\ell = m = 15$

Computational benefit of the reduced optimization

Newton-type optimization for different values of ν:

	$\nu = 0.01$			$\nu = 0.001$			$\nu = 0.0001$		
	Full	POD	DEIM	Full	POD	DEIM	Full	POD	DEIM
$N/\ell/m$	80	9	(9, 25)	200	11	(11, 25)	800	15	(15, 25)
$t_{opt}[s]$	4.83	4.13	3.22	23.1	6.38	5.25	1,865.8	23.61	18.42
$\mathcal{J}(y^*, u^*)$	0.0241	0.0262	0.0233	0.0206	0.0198	0.0200	0.0202	0.0191	0.0233
$\bar{e}$	-	0.0097	0.0141	-	0.0194	0.0192	-	0.0255	0.0238
S_P	-	1.35	1.9	-	3.7	4.4	-	79.0	101.3

Further numerical tests have shown:

- For $\nu = 0.0001$ and $n_c = 3$ discrete control points in $[0, L]$, a speedup of ~ 20 for all three optimization methods has been obtained.
- For a bounded control $-2 \leq u(t, x) \leq 2$, optimization with SPG leads to a speedup of 8.8 using POD-DEIM and $\nu = 0.0001$.

Computational benefit of the reduced optimization

Newton-type optimization for different values of ν:

	$\nu = 0.01$			$\nu = 0.001$			$\nu = 0.0001$		
	Full	POD	DEIM	Full	POD	DEIM	Full	POD	DEIM
$N/\ell/m$	80	9	(9, 25)	200	11	(11, 25)	800	15	(15, 25)
$t_{opt}[s]$	4.83	4.13	3.22	23.1	6.38	5.25	1,865.8	23.61	18.42
$\mathcal{J}(y^*, u^*)$	0.0241	0.0262	0.0233	0.0206	0.0198	0.0200	0.0202	0.0191	0.0233
$\bar{e}$	-	0.0097	0.0141	-	0.0194	0.0192	-	0.0255	0.0238
S_P	-	1.35	1.9	-	3.7	4.4	-	79.0	101.3

Further numerical tests have shown:

- For $\nu = 0.0001$ and $n_c = 3$ discrete control points in $[0, L]$, a speedup of ~ 20 for all three optimization methods has been obtained.
- For a bounded control $-2 \leq u(t, x) \leq 2$, optimization with SPG leads to a speedup of 8.8 using POD-DEIM and $\nu = 0.0001$.

Overview and main results

- 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.
- The accuracy of the reduced Burgers' model is of the same order when POD is extended by DEIM.

Future research questions

- 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 elaborated choice of reduced dimensions ℓ and m

This Master project is 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 is 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.