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

Manuel M. Baumann

Tea talk - October 11, 2013

TU Delft
Delft University of Technology

COSSE
Outline

1. What is Model Order Reduction (MOR)?
2. MOR algorithms for nonlinear dynamical systems
   - Proper Orthogonal Decomposition (POD)
   - Discrete Empirical Interpolation Method (DEIM)
3. Application to Burgers’ equation
   - POD-DEIM for Burgers’ equation
   - POD-DEIM for optimal control of Burgers’ equation
4. Numerical results
5. Summary and future research
Consider the nonlinear dynamical system

\[
\begin{aligned}
\dot{y}(t) &= Ay(t) + F(t, y(t)), \quad y(t) \in \mathbb{R}^N \\
y(0) &= y_0
\end{aligned}
\]  

(1)

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

\[
\begin{aligned}
\dot{y}(t) &= Ay(t) + F(t, y(t)), \quad y(t) \in \mathbb{R}^N \\
y(0) &= y_0
\end{aligned}
\] (1)

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

\[
\begin{aligned}
\dot{y}(t) &= Ay(t) + F(t, y(t)), \quad y(t) \in \mathbb{R}^N \\
y(0) &= y_0
\end{aligned}
\]  

(1)

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

\[
\begin{aligned}
\dot{y}(t) &= Ay(t) + F(t, y(t)), \quad y(t) \in \mathbb{R}^N \\
y(0) &= y_0
\end{aligned}
\]  

(1)

- arises in many applications, e.g. mechanical systems, fluid dynamics, neuron modeling, ...
- the matrix $A$ represents linear dynamical behavior and the function $F$ represents nonlinear dynamics
- often large dimension of (1) leads to huge computational work
The idea of model order reduction

Approximate the state via

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

where the matrix \( U_\ell \) has orthonormal columns, the so-called \textit{principal components} of \( y \), and \( \ell \ll N \).

Galerkin projection of the original full-order system leads to a reduced system of \( \ell \) equations:

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

\[
\Rightarrow \quad \dot{\tilde{y}} = U_\ell^T AU_\ell \tilde{y} + U_\ell^T F(t, U_\ell \tilde{y})
\]
The idea of model order reduction

Approximate the state via

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

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

Galerkin projection of the original full-order system leads to a reduced system of $\ell$ equations:

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

$$\Rightarrow \quad \ddot{\tilde{y}} = U_\ell^T A U_\ell \tilde{y} + U_\ell^T F(t, U_\ell \tilde{y})$$

$$=: \tilde{A}$$
Two questions are left...

Considering the reduced model

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

two questions are left:

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

Considering the reduced model

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

two questions are left:

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

Considering the reduced model

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

Two questions are left:

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

During the numerical simulation, build up the snapshot matrix

\[ Y := [y(t_1), \ldots, y(t_{ns})] \in \mathbb{R}^{N \times ns}, \]

with \( ns \) being the number of snapshots.

Perform a Singular Value Decomposition (SVD)

\[ Y = U \Sigma V^T \]

and let \( U_\ell := U(:,1:1) \) consist of those left singular vectors of \( Y \) that correspond to the \( \ell \) largest singular values in \( \Sigma \).
Consider the nonlinearity

\[ N := \begin{pmatrix} U^T \ell & F(t, U^\ell \tilde{y}(t)) \end{pmatrix} \]

\[ \ell \times N \quad N \times 1 \]

The approximation

\[ F \approx Wc, \quad W \in \mathbb{R}^{N \times m}, \quad c \in \mathbb{R}^m \]

is over-determined. Therefore, find projector \( P \) such that:

\[ P^T F = (P^T W)c \quad \Rightarrow \quad F \approx Wc = W(P^T W)^{-1}P^T F \]

\[ \Rightarrow \quad N \approx U^T \ell W (P^T W)^{-1}P^T F(t, U^\ell \tilde{y}(t)) \]

\[ m \times m \]
Consider the nonlinearity

\[ N := U^T_\ell F(t, U_\ell \tilde{y}(t)) \]

The approximation

\[ F \approx Wc, \quad W \in \mathbb{R}^{N \times m}, \quad c \in \mathbb{R}^m \]

is over-determined. Therefore, find projector \( P \) such that:

\[ P^T F = (P^T W)c \quad \Rightarrow \quad F \approx Wc = W(P^T W)^{-1}P^T F \]
\[ \Rightarrow \quad N \approx U^T_\ell W (P^T W)^{-1}P^T F(t, U_\ell \tilde{y}(t)) \]
Algorithm 1 The DEIM algorithm

1: **INPUT:** \( \{w_i\}_{i=1}^m \subset \mathbb{R}^N \) linear independent
2: **OUTPUT:** \( \vec{\phi} = [\phi_1, \ldots, \phi_m]^T \in \mathbb{R}^m, \ P \in \mathbb{R}^{N \times m} \)
3: \( [\sim, \phi_1] = \max\{|w_1|\} \)
4: \( W = [w_1], \ P = [e_{\phi_1}], \vec{\phi} = [\phi_1] \)
5: **for** \( i = 2 \) to \( m \) **do**
6: \( \text{Solve} \ (P^T W)c = P^T w_i \text{ for } c \)
7: \( r = w_i - Wc \)
8: \( [\sim, \phi_i] = \max\{|r|\} \)
9: \( W \leftarrow [W \ w_i], \ P \leftarrow [P \ e_{\phi_i}], \vec{\phi} \leftarrow [\vec{\phi} \ \phi_i] \)
10: **end for**

Reference

Algorithm 1 The DEIM algorithm

1: INPUT: \( \{w_i\}_{i=1}^m \subset \mathbb{R}^N \) linear independent
2: OUTPUT: \( \tilde{\phi} = [\phi_1, \ldots, \phi_m]^T \in \mathbb{R}^m, \mathcal{P} \in \mathbb{R}^{N \times m} \)
3: \( [\sim, \phi_1] = \max\{|w_1|\} \)
4: \( W = [w_1], \mathcal{P} = [e_{\phi_1}], \tilde{\phi} = [\phi_1] \)
5: for \( i = 2 \) to \( m \) do
6: Solve \((\mathcal{P}^T W)c = \mathcal{P}^T w_i\) for \( c \)
7: \( r = w_i - Wc \)
8: \( [\sim, \phi_i] = \max\{|r|\} \)
9: \( W \leftarrow [W \ w_i], \mathcal{P} \leftarrow [\mathcal{P} \ e_{\phi_i}], \tilde{\phi} \leftarrow \begin{bmatrix} \tilde{\phi} \\ \phi_i \end{bmatrix} \)
10: end for

Reference

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:

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

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

$$\mathbf{N} \approx U_\ell^T W (\mathcal{P}^T W)^{-1} \mathcal{P}^T \mathbf{F}(t, U_\ell \tilde{y}(t)) = U_\ell^T W (\mathcal{P}^T W)^{-1} \mathbf{F}(t, \mathcal{P}^T U_\ell \tilde{y}(t))$$

\[\ell \times m \quad m \times 1\]
Finally, some application...

Let’s consider

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:

\[ M\dot{y}(t) = -\frac{1}{2} B y^2(t) - \nu C y(t) + f, \quad t > 0 \]
\[ y(0) = y_0 \]

2. Time integration via implicit Euler + Newton’s method
Suppose, $\Phi_\ell$ is an $M$-orthogonal POD basis.

The POD reduced Burgers’ equation

\[
\Phi_\ell^T M \Phi_\ell \ddot{y}(t) = -\frac{1}{2} \Phi_\ell^T B(\Phi_\ell \tilde{y}(t))^2 - \nu \Phi_\ell^T C \Phi_\ell \tilde{y}(t)
\]

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

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

The POD-DEIM reduced Burgers’ equation

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

with $\tilde{B} = \Phi_\ell^T BW(PTW)^{-1} \in \mathbb{R}^{\ell \times m}$, $\tilde{F} = PT \Phi_\ell \in \mathbb{R}^{m \times \ell}$, and $\tilde{C} = C_\ell \in \mathbb{R}^{\ell \times \ell}$. 
Suppose, $\Phi_\ell$ is an M-orthogonal POD basis.

The POD reduced Burgers’ equation

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

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

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

The POD-DEIM reduced Burgers’ equation

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

with $\tilde{B} = \Phi_\ell^T BW(\mathcal{P}^T W)^{-1} \in \mathbb{R}^{\ell \times m}$, $\tilde{F} = \mathcal{P}^T \Phi_\ell \in \mathbb{R}^{m \times \ell}$, and $\tilde{C} = C_\ell \in \mathbb{R}^{\ell \times \ell}$. 
Burgers' equation

\( \ell = 3, m = 13 \)
$\ell = 5, m = 13$
\[ \ell = 7, m = 13 \]
\[ \ell = 9, m = 13 \]
\[ \ell = 11, \, m = 13 \]
Conclusion: High accuracy of the POD-DEIM reduced model.

But is it also faster?

- Spatial discretization of the full model depends on viscosity parameter $\nu$
- choose $\ell, m$ such that relative $L_2$-error in $\mathcal{O}(10^{-4})$
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 (left)
- POD-DEIM almost 4 times faster than pure POD (right)
Make use of this observation within the framework of PDE-constrained optimization

Find

\[ u^* = \arg\min_u J(y(u), u), \]

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

\[ c(y, u) = 0. \]

- \( J \) is called objective function,
- in order to evaluate \( J \), we need to solve \( c(y, u) = 0 \) for \( y(u) \),
- solve with algorithms for unconstrained minimization problems.
The optimal control $u^*$ is found in an iterative process.

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

Initialization

Choose $u_0$ and $k = 0$

$c(y_k, u_k) = 0$ for $y_k$

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

$\alpha^* = \arg\min_{\alpha \in \mathbb{R}_+} \mathcal{J}(y(u_k + \alpha s_k), u_k + \alpha s_k)$

$k = k + 1$

$u_{k+1} = u_k + \alpha^* s_k$
In particular, we consider the

**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

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

$$y(t, 0) = y(t, L) = 0, \quad t \in (0, T),$$

$$y(0, x) = y_0(x), \quad x \in (0, L).$$

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

![Graph showing uncontrolled and desired state for ν = 0.01.](image)

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

**Reference**

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 $\nu = 0.01$.](image)

Reference

Numerical treatment (sketch)

1. Discretize the objective function and Burgers’ equation in time and space
2. Use adjoint techniques in order to compute gradient (and Hessian) of the objective function
3. Apply first-order (or second-order) optimization algorithm
   - BFGS
   - SPG
   - Newton-type method
4. Explore the usage of a POD-DEIM reduced model

Reference

Numerical treatment (sketch)

1. Discretize the objective function and Burgers’ equation in time and space
2. Use *adjoint techniques* in order to compute gradient (and Hessian) of the objective function
3. Apply first-order (or second-order) optimization algorithm
   - BFGS
   - SPG
   - Newton-type method
4. Explore the usage of a POD-DEIM reduced model

Reference

Numerical treatment (sketch)

1. Discretize the objective function and Burgers’ equation in time and space
2. Use *adjoint techniques* in order to compute gradient (and Hessian) of the objective function
3. Apply first-order (or second-order) optimization algorithm
   - BFGS
   - SPG
   - Newton-type method
4. Explore the usage of a POD-DEIM reduced model

Reference

Numerical treatment (sketch)

1. Discretize the objective function and Burgers’ equation in time and space
2. Use *adjoint techniques* in order to compute gradient (and Hessian) of the objective function
3. Apply first-order (or second-order) optimization algorithm
   - BFGS
   - SPG
   - Newton-type method
4. Explore the usage of a POD-DEIM reduced model

Reference

Numerical results for the full-order model

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

- $k = 0$ (uncontrolled)
- $k = 1$
- $k = 2$
- $k = 3$
- $k = 4$
- $k = 5$
The corresponding 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:

- **Initialization**
  - Choose $\mathbf{u}^{(0)}$ and $K = 0$
  - $c(\mathbf{y}^{(0)}, \mathbf{u}^{(0)}) = 0$ for $\mathbf{y}^{(0)}$
  - Choose $\ell, m$

- **Expansion**
  - Obtain $\Phi, \mathcal{P}$ from snapshots of $\mathbf{y}^{(K)}$

- **Loop**
  - $\mathbf{u}^* = \text{argmin}_{\mathbf{u}} \tilde{J} (\tilde{\mathbf{y}}(\mathbf{u}), \mathbf{u})$
  - $\tilde{c}(\tilde{\mathbf{y}}^{(K+1)}, \mathbf{u}^*) = 0$ for $\tilde{\mathbf{y}}^{(K+1)}$
  - $\mathbf{y}^{(K+1)} = \Phi \tilde{\mathbf{y}}^{(K+1)}$

- **POD-DEIM**
- **Reduced optimization**
- **Reduced Burgers**
- **Expansion**
We propose the following algorithm for **POD-DEIM reduced optimal control**:

![Algorithm Diagram]

1. **Initialization**
   - Choose $u^{(0)}$ and $K = 0$
   - $c(y^{(0)}, u^{(0)}) = 0$ for $y^{(0)}$
   - Choose $\ell, m$
   - Obtain $\Phi, \mathcal{P}$ from snapshots of $y^{(K)}$

2. **Reduced Optimization**
   - $u^* = \arg\min_u \tilde{J}(\tilde{y}(u), u)$

3. **Full Burgers**
   - $c(y^{K+1}, u^*) = \text{for } y^{K+1}$

4. $K = K + 1$
Final state and control of the POD-DEIM reduced optimal control problem:

\[ \ell = m = 7 \]

\[ \ell = m = 15 \]
Reduced optimal control using the Newton-type method:

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

For $\nu = 0.0001$, low-dimensional control leads to a speedup of $\sim 20$ 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 $\sim 100$ for small $\nu$.
- For the reduced model, all derivatives need to be computed in terms of the reduced variable. This can be quite hard in practice.
What I still want to learn:

- Use the POD basis $\Phi_\ell$ also for dimension reduction of the control, i.e.

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

- Extend Burgers’ model to 2D/3D
- More sophisticated choice of reduced dimensions $\ell$ 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.  

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.  