Reduced Order Model Hessian Approximations in Newton Methods for Optimal Control

Matthias Heinkenschloss and Caleb Magruder

Abstract This paper introduces reduced order model (ROM) based Hessian approximations for use in inexact Newton methods for the solution of optimization problems implicitly constrained by a large-scale system, typically a discretization of a partial differential equation (PDE). The direct application of an inexact Newton method to this problem requires the solution of many PDEs per optimization iteration. To reduce the computational complexity, a ROM Hessian approximation is proposed. Since only the Hessian is approximated, but the original objective function and its gradient is used, the resulting inexact Newton method maintains the first-order global convergence property, under suitable assumptions. Thus even computationally inexpensive lower fidelity ROMs can be used, which is different from ROM approaches that replace the original optimization problem by a sequence of ROM optimization problem and typically need to accurately approximate function and gradient information of the original problem. In the proposed approach, the quality of the ROM Hessian approximation determines the rate of convergence, but not whether the method converges. The projection based ROM is constructed from state and adjoint snapshots, and is relatively inexpensive to compute. Numerical examples on semilinear parabolic optimal control problems demonstrate that the proposed approach can lead to substantial savings in terms of overall PDE solves required.

Keywords Model reduction · Optimization · Hessian approximation · Newton method

M. Heinkenschloss (✉)
Department of Computational and Applied Mathematics, Rice University, Houston, TX, USA
e-mail: heinken@rice.edu

C. Magruder
MathWorks, Natick, MA, USA
e-mail: cmagrude@mathworks.com

C. Beattie et al. (eds.), *Realization and Model Reduction of Dynamical Systems*,
https://doi.org/10.1007/978-3-030-95157-3_18

1 Introduction

We introduce reduced order model (ROM) based Hessian approximations for use in inexact Newton methods for the solution of large-scale smooth optimization problems

$$\min_{\mathbf{u}\in\mathbb{R}^m} \widehat{J}(\mathbf{u}) = J(\mathbf{y}(\mathbf{u}), \mathbf{u}), \tag{1}$$

where for given $\mathbf{u} \in \mathbb{R}^m$ the vector $\mathbf{y}(\mathbf{u}) \in \mathbb{R}^n$ is the solution of

$$\mathbf{c}(\mathbf{y}(\mathbf{u}), \mathbf{u}) = \mathbf{0}. \tag{2}$$

The optimization problem (1, 2) often comes from a discretization of optimal control problems and in this case $\mathbf{u} \in \mathbb{R}^m$ is the discretized control, $\mathbf{y} \in \mathbb{R}^n$ is the discretized state, and (2) is the discretized state equation. Our approach can be easily extended to a setting where the control space $\mathcal{U}$ and the state space $\mathcal{Y}$ are Hilbert spaces, but because of space restrictions we limit ourselves to the finite dimensional case. In our applications, the state Eq. (2) is a discretized parabolic partial differential equation (PDE). In this case, each objective function (1) evaluation requires the solution of a discretized PDE, the evaluation of the objective function gradient requires the solution of the (linear) adjoint PDE, and the evaluation of a Hessian-times-vector multiplication requires the additional solution of two (linear) PDE. This is computationally expensive. In addition, the additional PDEs that have to be solved for gradient and Hessian-times-vector computations depend on the solution of $\mathbf{y}(\mathbf{u}) \in \mathbb{R}^n$ of the state equation, which for time dependent PDEs is also memory intensive. ROMs can be used to reduce the computation time and memory requirements.

Previous approaches of using ROMs in optimization have approximated the mapping $\mathbf{u} \mapsto \mathbf{y}(\mathbf{u}) \in \mathbb{R}^n$ using a ROM applied to the state Eq. (2). Typically, the state Eq. (2) is approximated by a projection based ROM

$$\mathbf{V}^T\mathbf{c}(\mathbf{V}\widehat{\mathbf{y}}(\mathbf{u}), \mathbf{u}) = \mathbf{0}, \tag{3}$$

where $\mathbf{V} \in \mathbb{R}^{n\times r}$ is a matrix with rank $r \ll n$. The ROM state solution is then used to approximate (1) by

$$\min_{\mathbf{u}\in\mathbb{R}^m} J(\mathbf{V}\widehat{\mathbf{y}}(\mathbf{u}), \mathbf{u}). \tag{4}$$

For many problems, including example problems in Sect. 4 of this paper, the ROM approximation (4, 3) of the states implies via the optimality conditions that the controls $\mathbf{u}$ are also contained in a low dimensional subspace related to $\mathbf{V}$.

In special cases, one can extend ROM approaches for dynamical systems [5] to find one ROM such that the solution of (4, 3) is a good approximation of the solution of (1, 2). See, e.g., [2, 3], and the survey [7]. For general problems, however, the ROM is only valid in a potentially small neighborhood of the current control $\mathbf{u}_c$ and corresponding state $\mathbf{y}_c = \mathbf{y}(\mathbf{u}_c)$. In this case trust-region based model management approaches have been proposed. See [1, 10, 13, 16, 19, 20], and the surveys [7, 17].

While in principle, one can use trust-region based model management approaches that generate a new ROM at each new iterate, in practice their use is limited by the computational cost of ROM generation versus the computational savings resulting from that ROM.

Therefore, instead of generating ROMs for the optimization problem, we generate ROM approximations of the Hessians. These ROMs are computationally cheaper than a ROM that also has to approximate the objective function (1) and its gradient over a range of controls. Ideally the ROM Hessians still generate optimization steps that are close to the Newton steps, and therefore lead to fast local convergence of the optimization algorithm at a fraction of the cost of a corresponding Newton-type methods applied to the full order model (FOM) problem (1, 2).

In Sect. 2 we will outline our new approach and in Sect. 3 we will specify it further for a discretized parabolic optimal control problem. We demonstrate our approach on semilinear parabolic optimal control problems in Sect. 4.

2 Inexact Newton Methods and ROM Hessian Approximations

This section provides a general outline of our proposed approach. We begin with a review of gradient and Hessian computation and the line-search Newton- Conjugate-Gradient (Newton-CG) method. Then we will present the general structure of our ROM Hessian approximation, and a heuristic for choosing the ROM subspace within this approximation.

Inexact Newton Method and ROM Hessian Approximation. To make the definition of the objective function $\widehat{J}$ and its gradient and Hessian calculation rigorous, we make the following assumptions throughout.

(A1) There is an open set $D_u \subset \mathbb{R}^m$ such that for all $\mathbf{u} \in D_u$ the equation $\mathbf{c}(\mathbf{y}, \mathbf{u}) = 0$ has a unique solution $\mathbf{y} \in \mathbb{R}^n$.

(A2) There exists an open set $D_y \subset \mathbb{R}^n$ such that $\{\mathbf{y}(\mathbf{u}) : \mathbf{u} \in D_u\} \subset D_y$ and the functions J and $\mathbf{c}$ are twice continuously differentiable on $D_u \times D_y$.

(A3) The inverse $\mathbf{c}_\mathbf{y}(\mathbf{y}, \mathbf{u})^{-1}$ exists for all $(\mathbf{y}, \mathbf{u}) \in \{(\mathbf{y}, \mathbf{u}) \subset D_u \times D_y : \mathbf{c}(\mathbf{y}, \mathbf{u}) = \mathbf{0}\}$.

Here we use $\nabla_\mathbf{y} J(\mathbf{y}, \mathbf{u}) \in \mathbb{R}^n$, $\nabla_\mathbf{u} J(\mathbf{y}, \mathbf{u}) \in \mathbb{R}^m$ to denote the partial gradients and $\mathbf{c}_\mathbf{y}(\mathbf{y}, \mathbf{u}) \in \mathbb{R}^{n\times n}$, $\mathbf{c}_\mathbf{u}(\mathbf{y}, \mathbf{u}) \in \mathbb{R}^{n\times m}$ to denote the partial Jacobians. Similar notation is used for second derivatives.

The gradient of the objective $\widehat{J}$ in (1, 2) can be computed using the so-called adjoint equation approach. See, e.g., [12, Sect. 1.6]. Define the Lagrangian

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

corresponding to (1, 2). Given a control $\mathbf{u}$ and corresponding state $\mathbf{y}(\mathbf{u})$ the gradient of the objective $\widehat{J}$ is computed by first solving the adjoint equation

$$\mathbf{c}_{\mathbf{y}}(\mathbf{y}(\mathbf{u}), \mathbf{u})^T \boldsymbol{\lambda} = -\nabla_{\mathbf{y}} J(\mathbf{y}(\mathbf{u}), \mathbf{u}) \tag{5a}$$

for $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\mathbf{u})$ and then setting

$$\nabla \widehat{J}(\mathbf{u}) = \nabla_{\mathbf{u}} J(\mathbf{y}(\mathbf{u}), \mathbf{u}) + \mathbf{c}_{\mathbf{u}}(\mathbf{y}(\mathbf{u}), \mathbf{u})^T \boldsymbol{\lambda}(\mathbf{u}). \tag{5b}$$

Given a control $\mathbf{u}$, the corresponding state $\mathbf{y}(\mathbf{u})$, and the corresponding adjoint $\boldsymbol{\lambda}(\mathbf{u})$, the Hessian-times-vector product $\nabla^2 \widehat{J}(\mathbf{u})\,\mathbf{d}$ is computed by solving the linearized state equation

$$\mathbf{c}_{\mathbf{y}}(\mathbf{y}(\mathbf{u}), \mathbf{u})\,\mathbf{w} = \mathbf{c}_{\mathbf{u}}(\mathbf{y}(\mathbf{u}), \mathbf{u})\,\mathbf{d} \tag{6a}$$

for $\mathbf{w}$, then the second order adjoint equation

$$\mathbf{c}_{\mathbf{y}}(\mathbf{y}(\mathbf{u}), \mathbf{u})^T \mathbf{p} = \nabla_{yy} L(\mathbf{y}(\mathbf{u}), \mathbf{u}, \lambda(\mathbf{u}))\mathbf{w} - \nabla_{\mathbf{yu}} L(\mathbf{y}(\mathbf{u}), \mathbf{u}, \lambda(\mathbf{u}))\,\mathbf{d} \tag{6b}$$

for $\mathbf{p}$, and then computing

$$\begin{aligned} \nabla^2 \widehat{J}(\mathbf{u})\,\mathbf{d} =& \mathbf{c}_{\mathbf{u}}(\mathbf{y}(\mathbf{u}), \mathbf{u})^T \mathbf{p} - \nabla_{\mathbf{uy}} L(\mathbf{y}(\mathbf{u}), \mathbf{u}, \boldsymbol{\lambda}(\mathbf{u}))\,\mathbf{w} \\ &+ \nabla_{\mathbf{uu}} L(\mathbf{y}(\mathbf{u}), \mathbf{u}, \boldsymbol{\lambda}(\mathbf{u}))\,\mathbf{d}. \end{aligned} \tag{6c}$$

For optimal control problems, the computation of $\mathbf{y}(\mathbf{u})$ requires the solution of a nonlinear discretized PDE. The gradient computation requires the solution of a linear discretized adjoint PDE to compute $\boldsymbol{\lambda}(\mathbf{u})$. Each Hessian-times-vector operation $\nabla^2 \widehat{J}(\mathbf{u})\,\mathbf{d}$ requires the solution of two linear discretized PDEs to compute $\mathbf{w}$ and $\mathbf{p}$, respectively.

The problem (1) is solved using a line-search Newton-CG Method. However, a trust-region method could be used as well, and the proposed ROM Hessian approximation provides the same computational benefits in a trust-region setting.

Given a current iterate $\mathbf{u}_c$ the line-search Newton-CG Method [15, Algorithm 7.1] applies the truncated CG method to approximately solve the Newton subproblem

$$\nabla^2 \widehat{J}(\mathbf{u}_c)\,\mathbf{d} = -\nabla \widehat{J}(\mathbf{u}_c). \tag{7}$$

Then it computes a suitable step-size $\alpha_c \in (0, 1]$ such that the sufficient decrease condition

$$\widehat{J}(\mathbf{u}_c + \alpha_c \mathbf{d}) \leq \widehat{J}(\mathbf{u}_c) + 10^{-4} \alpha_c \mathbf{d}^T \nabla \widehat{J}(\mathbf{u}_c) \tag{8}$$

is satisfied. The new iterate is given by $\mathbf{u}_+ = \mathbf{u}_c + \alpha_c \mathbf{d}$.

The ROM Hessian approximation is computed by applying a projection ROM to the Eq. (6a, b). Let $\mathbf{V} \in \mathbb{R}^{n \times r}$, $r \ll n$, with linearly independent columns such that $\mathbf{V}^T \mathbf{c}_{\mathbf{y}}(\mathbf{y}(\mathbf{u}), \mathbf{u})\mathbf{V}$ is invertible.

The ROM Hessian-times-vector product is computed by first solving the projected linearized state equation

$$\mathbf{V}^T \mathbf{c}_{\mathbf{y}}(\mathbf{y}(\mathbf{u}), \mathbf{u}) \mathbf{V}\, \widehat{\mathbf{w}} = \mathbf{V}^T \mathbf{c}_{\mathbf{u}}(\mathbf{y}(\mathbf{u}), \mathbf{u})\, \mathbf{d} \tag{9a}$$

for $\widehat{\mathbf{w}}$, then solving the projected second order adjoint equation

$$\mathbf{V}^T \mathbf{c}_{\mathbf{y}}(\mathbf{y}(\mathbf{u}), \mathbf{u})^T \mathbf{V}\, \widehat{\mathbf{p}} = \mathbf{V}^T \nabla_{yy} L(\mathbf{y}(\mathbf{u}), \mathbf{u}, \lambda(\mathbf{u})) \mathbf{V} \widehat{\mathbf{w}} - \mathbf{V}^T \nabla_{\mathbf{yu}} L(\mathbf{y}(\mathbf{u}), \mathbf{u}, \lambda(\mathbf{u}))\, \mathbf{d} \tag{9b}$$

for $\widehat{\mathbf{p}}$, and then computing

$$\begin{aligned} \widetilde{\nabla^2 \widehat{J}(\mathbf{u})}\, \mathbf{d} =& \mathbf{c}_{\mathbf{u}}(\mathbf{y}(\mathbf{u}), \mathbf{u})^T \mathbf{V} \widehat{\mathbf{p}} - \nabla_{\mathbf{uy}} L(\mathbf{y}(\mathbf{u}), \mathbf{u}, \boldsymbol{\lambda}(\mathbf{u}))\, \mathbf{V} \widehat{\mathbf{w}} \\ &+ \nabla_{\mathbf{uu}} L(\mathbf{y}(\mathbf{u}), \mathbf{u}, \boldsymbol{\lambda}(\mathbf{u}))\, \mathbf{d}. \end{aligned} \tag{9c}$$

The ROM Hessian approximation (9) is used to compute the direction in the inexact Newton method. Instead of solving (7), given a current iterate $\mathbf{u}_c$ the direction $\widetilde{\mathbf{d}}$ is computed by using the truncated CG method to approximately solve

$$\widetilde{\nabla^2 \widehat{J}(\mathbf{u}_c)}\, \widetilde{\mathbf{d}} = -\nabla \widehat{J}(\mathbf{u}_c). \tag{10}$$

The step size α_c is computed as before using (8) with $\mathbf{d}$ replaced by $\widetilde{\mathbf{d}}$. The new iterate is given by $\mathbf{u}_+ = \mathbf{u}_c + \alpha_c \widetilde{\mathbf{d}}$.

The global convergence of the original line-search Newton method and the line-search Newton method with ROM Hessian approximation can be proven using standard arguments. See, e.g., [15, Theorem 3.2]. While first order global convergence can be ensured under standard conditions if the ROM Hessian approximation is used, the speed with which the inexact Newton method converges in this case depends on the quality of the ROM Hessian approximation, and in particular on $\mathbf{V}$. Next we motivate our choice of $\mathbf{V}$.

Basic Properties of the ROM Hessian. Using (6) it can be seen that the Hessian of $\widehat{J}$ is given by

$$\nabla^2 \widehat{J}(\mathbf{u}) = \begin{pmatrix} \mathbf{c}_{\mathbf{y}}(\mathbf{y}, \mathbf{u})^{-1} \mathbf{c}_{\mathbf{u}}(\mathbf{y}, \mathbf{u}) \\ \mathbf{I} \end{pmatrix}^T \begin{pmatrix} \nabla_{\mathbf{yy}} L(\mathbf{y}, \mathbf{u}, \boldsymbol{\lambda}) & \nabla_{\mathbf{yu}} L(\mathbf{y}, \mathbf{u}, \boldsymbol{\lambda}) \\ \nabla_{\mathbf{uy}} L(\mathbf{y}, \mathbf{u}, \boldsymbol{\lambda}) & \nabla_{\mathbf{uu}} L(\mathbf{y}, \mathbf{u}, \boldsymbol{\lambda}) \end{pmatrix} \begin{pmatrix} \mathbf{c}_{\mathbf{y}}(\mathbf{y}, \mathbf{u})^{-1} \mathbf{c}_{\mathbf{u}}(\mathbf{y}, \mathbf{u}) \\ \mathbf{I} \end{pmatrix}, \tag{11}$$

where we have set $\mathbf{y} = \mathbf{y}(\mathbf{u})$ and $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\mathbf{u})$ to simplify notation. We will use this simplified notation frequently during the remainder of this section.

Applying the Implicit Function Theorem to the state Eq. (2) shows that the sensitivity of $\mathbf{u} \mapsto \mathbf{y}(\mathbf{u})$ is given by

$$\mathbf{y}_{\mathbf{u}}(\mathbf{u}) = \mathbf{c}_{\mathbf{y}}(\mathbf{y}, \mathbf{u})^{-1} \mathbf{c}_{\mathbf{u}}(\mathbf{y}, \mathbf{u}) \in \mathbb{R}^{n \times m}. \tag{12}$$

Similarly, applying the Implicit Function Theorem to the adjoint Eq. (5a) shows that the derivative of $\mathbf{u} \mapsto \boldsymbol{\lambda}(\mathbf{u})$ is given by

$$\begin{aligned}\boldsymbol{\lambda}_{\mathbf{u}}(\mathbf{u}) &= \mathbf{c}_{\mathbf{y}}(\mathbf{y},\mathbf{u})^{-T}\left(\nabla_{\mathbf{yy}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda}),\ \nabla_{\mathbf{yu}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda})\right)\begin{pmatrix}\mathbf{c}_{\mathbf{y}}(\mathbf{y},\mathbf{u})^{-1}\mathbf{c}_{\mathbf{u}}(\mathbf{y},\mathbf{u})\\ \mathbf{I}\end{pmatrix}\\ &= \mathbf{c}_{\mathbf{y}}(\mathbf{y},\mathbf{u})^{-T}\left(\nabla_{\mathbf{yy}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda}),\ \nabla_{\mathbf{yu}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda})\right)\begin{pmatrix}\mathbf{y}_{\mathbf{u}}(\mathbf{u})\\ \mathbf{I}\end{pmatrix} \in \mathbb{R}^{n\times m}. \end{aligned} \tag{13}$$

Using these derivatives, the Hessian can be written as

$$\nabla^2\widehat{J}(\mathbf{u}) = \mathbf{c}_{\mathbf{u}}(\mathbf{y},\mathbf{u})^T\boldsymbol{\lambda}_{\mathbf{u}}(\mathbf{u}) + \left(\nabla_{\mathbf{uy}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda}),\ \nabla_{\mathbf{uu}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda})\right)\begin{pmatrix}\mathbf{y}_{\mathbf{u}}(\mathbf{u})\\ \mathbf{I}\end{pmatrix}. \tag{14}$$

Using (9) it can be seen that the ROM Hessian approximation is given by

$$\begin{aligned}\widetilde{\nabla^2\widehat{J}(\mathbf{u})} = &\begin{pmatrix}\mathbf{V}\left(\mathbf{V}^T\mathbf{c}_{\mathbf{y}}(\mathbf{y},\mathbf{u})\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{c}_{\mathbf{u}}(\mathbf{y},\mathbf{u})\\ \mathbf{I}\end{pmatrix}^T\begin{pmatrix}\nabla_{\mathbf{yy}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda}) & \nabla_{\mathbf{yu}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda})\\ \nabla_{\mathbf{uy}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda}) & \nabla_{\mathbf{uu}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda})\end{pmatrix}\\ &\times\begin{pmatrix}\mathbf{V}\left(\mathbf{V}^T\mathbf{c}_{\mathbf{y}}(\mathbf{y},\mathbf{u})\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{c}_{\mathbf{u}}(\mathbf{y},\mathbf{u})\\ \mathbf{I}\end{pmatrix}. \end{aligned} \tag{15}$$

If we define

$$\widetilde{\mathbf{y}_{\mathbf{u}}(\mathbf{u})} = \mathbf{V}\left(\mathbf{V}^T\mathbf{c}_{\mathbf{y}}(\mathbf{y},\mathbf{u})\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{c}_{\mathbf{u}}(\mathbf{y},\mathbf{u}) \in \mathbb{R}^{n\times m} \tag{16}$$

and

$$\widetilde{\boldsymbol{\lambda}_{\mathbf{u}}(\mathbf{u})} = \mathbf{V}\left(\mathbf{V}^T\mathbf{c}_{\mathbf{y}}(\mathbf{y},\mathbf{u})\mathbf{V}\right)^{-T}\mathbf{V}^T\left(\nabla_{\mathbf{yy}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda}),\ \nabla_{\mathbf{yu}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda})\right)\begin{pmatrix}\widetilde{\mathbf{y}_{\mathbf{u}}(\mathbf{u})}\\ \mathbf{I}\end{pmatrix} \in \mathbb{R}^{n\times m}, \tag{17}$$

then the ROM Hessian approximation can be written as

$$\widetilde{\nabla^2\widehat{J}(\mathbf{u})} = \mathbf{c}_{\mathbf{u}}(\mathbf{y},\mathbf{u})^T\widetilde{\boldsymbol{\lambda}_{\mathbf{u}}(\mathbf{u})} + \left(\nabla_{\mathbf{uy}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda}),\ \nabla_{\mathbf{uu}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda})\right)\begin{pmatrix}\widetilde{\mathbf{y}_{\mathbf{u}}(\mathbf{u})}\\ \mathbf{I}\end{pmatrix}. \tag{18}$$

The next result ensures the positive definiteness of the Hessian and its ROM approximation in situations that are often encountered in classes of applications, such as those in Sect. 4.

Proposition 1 *Let* $\mathbf{V}\in\mathbb{R}^{n\times r}$ *have rank* $r<n$ *and satisfy that* $\mathbf{V}^T\mathbf{c}_{\mathbf{y}}(\mathbf{y}(\mathbf{u}),\mathbf{u})\mathbf{V}$ *is invertible. If* $\nabla_{\mathbf{uu}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda})$ *is positive definite,* $\nabla_{\mathbf{yy}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda})$ *is positive semi-definite, and* $\nabla_{\mathbf{yu}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda})=\mathbf{0}$*, then* $\nabla^2\widehat{J}(\mathbf{u})$ *and* $\widetilde{\nabla^2\widehat{J}(\mathbf{u})}$ *are positive definite with smallest eigenvalue bounded from below by the smallest eigenvalue of* $\nabla_{\mathbf{uu}}L(\mathbf{y},\mathbf{u},\boldsymbol{\lambda})$.

Proof The statement follows immediately from (11) and (15).

Comparing (14) with (18) shows that $\nabla^2 \widehat{J}(\mathbf{u}) \approx \widetilde{\nabla^2 \widehat{J}(\mathbf{u})}$ if $\mathbf{y_u}(\mathbf{u}) \approx \widetilde{\mathbf{y_u}(\mathbf{u})}$ and $\boldsymbol{\lambda}_\mathbf{u}(\mathbf{u}) \approx \widetilde{\boldsymbol{\lambda}_\mathbf{u}(\mathbf{u})}$. The latter two approximation conditions motivate our choice for the ROM matrix $\mathbf{V}$.

Proposition 2 *Let $\mathbf{V} \in \mathbb{R}^{n\times r}$ have rank $r < n$ and satisfy that $\mathbf{V}^T \mathbf{c}_\mathbf{y}(\mathbf{y}(\mathbf{u}), \mathbf{u})\mathbf{V}$ is invertible. If $\mathbf{y}(\mathbf{u}+\mathbf{d}) \in \mathcal{R}(\mathbf{V})$ for all sufficiently small $\mathbf{d}$, then*

$$\widetilde{\mathbf{y_u}(\mathbf{u})} = \mathbf{y_u}(\mathbf{u}). \tag{19}$$

If, in addition, $\boldsymbol{\lambda}(\mathbf{u}+\mathbf{d}) \in \mathcal{R}(\mathbf{V})$ for all sufficiently small $\mathbf{d}$, then

$$\widetilde{\boldsymbol{\lambda}_\mathbf{u}(\mathbf{u})} = \boldsymbol{\lambda}_\mathbf{u}(\mathbf{u}) \quad \textit{and} \quad \widetilde{\nabla^2 \widehat{J}(\mathbf{u})} = \nabla^2 \widehat{J}(\mathbf{u}). \tag{20}$$

Proof If $\mathbf{y}(\mathbf{u}+\mathbf{d}) \in \mathcal{R}(\mathbf{V})$ for all sufficiently small $\mathbf{d}$. Then $\mathbf{y_u}(\mathbf{u})\mathbf{d} \in \mathcal{R}(\mathbf{V})$ for all $\mathbf{d}$. Furthermore, for every $\mathbf{u}+\mathbf{d}$ with $\mathbf{d}$ sufficiently small there exists $\widetilde{\mathbf{y}}(\mathbf{u}+\mathbf{d})$ such that $\mathbf{y}(\mathbf{u}+\mathbf{d}) = \mathbf{V}\widetilde{\mathbf{y}}(\mathbf{u}+\mathbf{d})$. Inserting this into the state Eq. (2) implies

$$\mathbf{V}^T \mathbf{c}(\mathbf{V}\widetilde{\mathbf{y}}(\mathbf{u}+\mathbf{d}), \mathbf{u}+\mathbf{d}) = \mathbf{0} \quad \text{for all sufficiently small } \mathbf{d}.$$

Applying the Implicit Function Theorem to this equation shows that

$$\widetilde{\mathbf{y}}_\mathbf{u}(\mathbf{u}) = \left(\mathbf{V}^T \mathbf{c}_\mathbf{y}(\mathbf{y}(\mathbf{u}), \mathbf{u})\mathbf{V}\right)^{-1} \mathbf{V}^T \mathbf{c}_\mathbf{u}(\mathbf{y}(\mathbf{u}), \mathbf{u})$$

and (16) can be written as $\widetilde{\mathbf{y_u}(\mathbf{u})} = \mathbf{V}\widetilde{\mathbf{y}}_\mathbf{u}(\mathbf{u})$. Since $\mathbf{y}(\mathbf{u}+\mathbf{d}) = \mathbf{V}\widetilde{\mathbf{y}}(\mathbf{u}+\mathbf{d})$ for all sufficiently small $\mathbf{d}$,

$$\widetilde{\mathbf{y_u}(\mathbf{u})} = \mathbf{V}\widetilde{\mathbf{y}}_\mathbf{u}(\mathbf{u}) = \mathbf{y_u}(\mathbf{u}),$$

which is (19).

If, in addition, $\boldsymbol{\lambda}(\mathbf{u}+\mathbf{d}) \in \mathcal{R}(\mathbf{V})$ for all sufficiently small $\mathbf{d}$, then we can use similar argument to show the first equality in (20).

The second identity in (20) follows immediately from (14), (18), (19), and first identity (20).

In general there is no matrix $\mathbf{V} \in \mathbb{R}^{n\times r}$ with small rank $r \ll n$, such that the conditions in Proposition 2 hold. However, if there exists $\mathbf{V} \in \mathbb{R}^{n\times r}$, $r \ll n$, with orthogonal columns such that the projections of $\mathbf{y}(\mathbf{u}+\mathbf{d})$ and $\boldsymbol{\lambda}(\mathbf{u}+\mathbf{d})$ onto $\mathcal{R}(\mathbf{V})$ are close to $\mathbf{y}(\mathbf{u}+\mathbf{d})$ and $\boldsymbol{\lambda}(\mathbf{u}+\mathbf{d})$, respectively, or equivalently such that

$$\|(\mathbf{I} - \mathbf{V}\mathbf{V}^T)\,\mathbf{y}(\mathbf{u}+\mathbf{d})\| \le \text{tol} \quad \text{and} \quad \|(\mathbf{I} - \mathbf{V}\mathbf{V}^T)\,\boldsymbol{\lambda}(\mathbf{u}+\mathbf{d})\| \le \text{tol} \tag{21}$$

for a smal tolerance tol $\ll 1$, then we expect $\mathbf{y_u}(\mathbf{u}) \approx \widetilde{\mathbf{y_u}(\mathbf{u})}$, $\boldsymbol{\lambda}_\mathbf{u}(\mathbf{u}) \approx \widetilde{\boldsymbol{\lambda}_\mathbf{u}(\mathbf{u})}$, and consequently that the ROM Hessian $\widetilde{\nabla^2 \widehat{J}(\mathbf{u})}$ is a good approximation of the original FOM Hessian $\nabla^2 \widehat{J}(\mathbf{u})$.

However, finding $\mathbf{V} \in \mathbb{R}^{n\times r}$ such that (21) is guaranteed to hold for *all* sufficiently small $\mathbf{d}$ would be computationally expensive. In practice we compute $\mathbf{V} \in \mathbb{R}^{n\times r}$ using proper orthogonal decomposition (POD) applied to the state $\mathbf{y}(\mathbf{u})$ and the adjoint $\boldsymbol{\lambda}(\mathbf{u})$ at the current control.

ROM Hessians versus ROM Optimization Proposition 2 and even (21) describe optimistic scenarios. If we were able to find a $\mathbf{V}$ such that (21) holds at the current control $\mathbf{u} = \mathbf{u}_c$, then it could be used to approximate the original problem (1, 2) in a trust-region based model management framework and approximately solve (4, 3) over all $\mathbf{u}$ in a neighborhood (trust-region) around the current control $\mathbf{u}_c$. Rather than using the ROM model $\mathbf{u} \mapsto J(\mathbf{V}\widehat{\mathbf{y}}(\mathbf{u}), \mathbf{u})$ in a neighborhood of the current control $\mathbf{u}_c$, our approach uses the approximate quadratic Taylor expansion $\mathbf{d} \mapsto \widehat{J}(\mathbf{u}_c) + \nabla\widehat{J}(\mathbf{u}_c)^T\mathbf{d} + \mathbf{d}^T\widetilde{\nabla^2\widehat{J}(\mathbf{u}_c)}\,\mathbf{d}$ as a model at controls $\mathbf{u} = \mathbf{u}_c + \mathbf{d}$. The true function $\widehat{J}(\mathbf{u})$ and this Taylor model have the same function and gradient value at $\mathbf{u}_c$, no matter how good $\mathbf{V}$ is. If $\widetilde{\nabla^2\widehat{J}(\mathbf{u}_c)}$ well approximates $\nabla^2\widehat{J}(\mathbf{u}_c)$, we will essentially compute a Newton step. If the resulting $\mathbf{V}$ is less good, our approach can still generate a descent direction that is much better than a simple gradient step. Computationally inexpensive, but possibly less accurate ROMs can still be used to accelerate the overall optimization.

3 ROM Hessian Approximations for Discrete Time Optimal Control Problems

In this section we apply the ROM Hessian approach to a minimization problem that arises, e.g., from a discretized parabolic optimal control problem.

Model Problem. Given functions $\ell : \mathbb{R}^{n_y} \to \mathbb{R}$, $\sigma : \mathbb{R}^{n_u} \to \mathbb{R}$, $\mathbf{F} : \mathbb{R}^{n_y} \to \mathbb{R}^{n_y}$, and $\mathbf{G} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$, and given $\mathbf{y}_0 \in \mathbb{R}^{n_y}$ consider the following minimization problem in the variables $\mathbf{u} = (\mathbf{u}_1^T, \ldots, \mathbf{u}_{n_t}^T)^T \in \mathbb{R}^{n_u n_t}$, $\mathbf{y} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_{n_t}^T)^T \in \mathbb{R}^{n_y n_t}$.

$$\text{Minimize} \quad \sum_{k=1}^{n_t} \ell(\mathbf{y}_k) + \sigma(\mathbf{u}_k), \tag{22a}$$

where $\mathbf{y}$ and $\mathbf{u}$ satisfy an implicit constraint,

$$\mathbf{M}\left(\frac{\mathbf{y}_{k+1} - \mathbf{y}_k}{\Delta t}\right) + \mathbf{F}(\mathbf{y}_{k+1}) = \mathbf{G}(\mathbf{u}_{k+1}), \quad k = 0, \ldots, n_1 - 1, \qquad \mathbf{y}_0 \text{ given.} \tag{22b}$$

It is easily possible to extent our ROM Hessian approach to generalizations of (22), e.g., replace $\ell(\mathbf{y}_k)$ and $\mathbf{G}(\mathbf{u}_k)$ by $\ell(\mathbf{y}_k, \mathbf{u}_k)$ and $\mathbf{G}(\mathbf{y}_k, \mathbf{u}_k)$. We consider (22) to simplify notation.

Throughout this section we make the following assumptions, which are adaptations of the assumptions (A1)–(A3) to the problem (22).

(A1') There exists an open set $D_u \subset \mathbb{R}^{n_u n_t}$ such that for every $\mathbf{u} = (\mathbf{u}_1^T, \ldots, \mathbf{u}_{n_t}^T)^T \in D$ the state Eq. (22b) has a unique solution $\mathbf{y}(\mathbf{u}) = (\mathbf{y}_1(\mathbf{u})^T, \ldots, \mathbf{y}_{n_t}(\mathbf{u})^T)^T$.

(A2') There exists an open set $D_y \subset \mathbb{R}^{n_y n_t}$ such that $\{\mathbf{y}(\mathbf{u}) : \mathbf{u} \in D_u\} \subset D_y$, and the functions $\ell : D_y \to \mathbb{R}$, $\sigma : D_u \to \mathbb{R}$, $\mathbf{F} : D_y \to \mathbb{R}^{n_y}$, and $\mathbf{G} : D_u \to \mathbb{R}^{n_y}$ are twice continuously differentiable.

(A3') For every $\mathbf{y} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_{n_t}^T)^T \in \{\mathbf{y}(\mathbf{u}) : \mathbf{u} \in D_u\}$ and every $\mathbf{r}_1, \ldots, \mathbf{r}_{n_t} \in \mathbb{R}^{n_y}$ there exists a unique solution $\mathbf{w}_1, \ldots, \mathbf{w}_{n_t} \in \mathbb{R}^{n_y}$ of the equations

$$\mathbf{w}_0 = \mathbf{0}, \qquad \mathbf{w}_{k+1} - \mathbf{w}_k = \Delta t\, \mathbf{F}_y(\mathbf{y}_{k+1})\mathbf{w}_{k+1} + \mathbf{r}_{k+1}, \quad k = 0, \ldots, n_t - 1.$$

Under the assumptions (A1')-(A3') we can define the function $\widehat{J} : D_u \to \mathbb{R}$, $\widehat{J}(\mathbf{u}) = \sum_{k=1}^{K} \ell(\mathbf{y}_k(\mathbf{u})) + \sigma(\mathbf{u}_k)$, so that (22), is a special case of (1, 2).

Gradient and Hessian Computation. Under the assumptions (A1')–(A3') the function $\widehat{J} : D_u \to \mathbb{R}$ is twice continuously differentiable. Its gradient can be computed using the adjoint equation approach in Algorithm 1. Hessian-times-vector operations are computed using Algorithm 2.

Algorithm 1: Gradient Computation

1: Given $\mathbf{u} = (\mathbf{u}_1^T, \ldots, \mathbf{u}_{n_t}^T)^T$ let $\mathbf{y} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_{n_t}^T)^T$ solve the state Eq. (22b).

2: Solve the adjoint equations for $\boldsymbol{\lambda}_{n_t}, \ldots, \boldsymbol{\lambda}_1$:

$$[\mathbf{M} + \Delta t\, \mathbf{F}_y(\mathbf{y}_{n_t})]^T \boldsymbol{\lambda}_{n_t} = -\nabla_y \ell(\mathbf{y}_{n_t}), \tag{23a}$$

$$[\mathbf{M} + \Delta t\, \mathbf{F}_y(\mathbf{y}_k)]^T \boldsymbol{\lambda}_k = \mathbf{M}\boldsymbol{\lambda}_{k+1} - \nabla_y \ell(\mathbf{y}_k), \quad k = n_t - 1, \ldots, 1. \tag{23b}$$

3: Compute the gradient

$$\nabla J(\mathbf{u}) = \begin{pmatrix} \nabla_u \sigma(\mathbf{u}_1) - \Delta t\, \mathbf{G}_u(\mathbf{u}_1)^T \boldsymbol{\lambda}_1 \\ \vdots \\ \nabla_u \sigma(\mathbf{u}_{n_t}) - \Delta t\, \mathbf{G}_u(\mathbf{u}_{n_t})^T \boldsymbol{\lambda}_{n_t} \end{pmatrix}.$$

Note that since the objective function and the implicit constraints in the model problem (22) are separable, $\nabla_{\mathbf{uy}} L(\mathbf{y}, \mathbf{u}, \boldsymbol{\lambda}) = \mathbf{0}$.

Hessian Approximation by Model Order Reduction. As mentioned towards the end of Sect. 2, we compute a ROM from state $\mathbf{y}$ and adjoint $\boldsymbol{\lambda}$ information. In this example the state $\mathbf{y}$ and the adjoint $\boldsymbol{\lambda}$ are vectors of vectors $\mathbf{y}_k \in \mathbb{R}^{n_y}$ and $\boldsymbol{\lambda}_k \in \mathbb{R}^{n_y}$ respectively. We compute a matrix $\mathbf{V} \in \mathbb{R}^{n_y \times r}$ such that the components $\mathbf{y}_k$ and $\boldsymbol{\lambda}_k$ are approximately contained in the range of $\mathbf{V}$ (this $\mathbf{V}$ is a component of the projection matrix in Sect. 2) The projection matrix for $\mathbf{y}$ and $\boldsymbol{\lambda}$ is the $n_t n_y \times n_t r$ block diagonal matrix with identical diagonal blocks given by $\mathbf{V}$.

Algorithm 2: Hessian-Times-Vector Computation – $\nabla^2 \widehat{J}(\mathbf{u})\mathbf{v}$

1: Given $\mathbf{u} = (\mathbf{u}_1^T, \ldots, \mathbf{u}_{n_t}^T)^T$ let $\mathbf{y} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_{n_t}^T)^T$ solve the state Eq. (22b) and let $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1^T, \ldots, \boldsymbol{\lambda}_{n_t}^T)^T$ solve the adjoint Eq. (23).

2: Set $\mathbf{w}_0 = \mathbf{0}$ and solve the linearized state equation $\mathbf{w}_1, \ldots, \mathbf{w}_{n_t}$:

$$[\mathbf{M} + \Delta t \mathbf{F}_y(\mathbf{y}_{k+1})]\mathbf{w}_{k+1} = \mathbf{M}\mathbf{w}_k + \Delta t\, \mathbf{G}_u(\mathbf{u}_{k+1})\mathbf{v}_{k+1}, \qquad k = 0, \ldots, n_t - 1.$$

3: Solve the second order adjoint equations for $\mathbf{p}_{n_t}, \ldots, \mathbf{p}_1$.

$$\begin{aligned}[\mathbf{M} + \Delta t\, \mathbf{F}_y(\mathbf{y}_{n-t})]^T \mathbf{p}_{n_t} &= -\left[\Delta t\, (\boldsymbol{\lambda}_{n_t}^T \mathbf{F}(\mathbf{y}_{n_t}))_{yy} + \ell_{yy}(\mathbf{y}_{n_t})\right]\mathbf{w}_{n_t},\\ [\mathbf{M} + \Delta t\, \mathbf{F}_y(\mathbf{y}_k)]^T \mathbf{p}_k &= \mathbf{M}\mathbf{p}_{k+1} - \left[\Delta t\, (\boldsymbol{\lambda}_k^T \mathbf{F}(\mathbf{y}_k))_{yy} + \ell_{yy}(\mathbf{y}_k)\right]\mathbf{w}_k,\ k = n_t - 1, \ldots, 1.\end{aligned}$$

4: Compute the action of the Hessian

$$\nabla^2 \widehat{J}(\mathbf{u})\mathbf{v} = \begin{pmatrix} -\Delta t\, \mathbf{G}_u(\mathbf{u}_1)^T \mathbf{p}_1 + [\sigma_{uu}(\mathbf{u}_1) - \Delta t\, (\boldsymbol{\lambda}_1^T \mathbf{G}(\mathbf{u}_1))_{uu}]\mathbf{v}_1 \\ \vdots \\ -\Delta t\, \mathbf{G}_u(\mathbf{u}_{n_t})^T \mathbf{p}_{n_t} + [\sigma_{uu}(\mathbf{u}_{n_t}) - \Delta t\, (\boldsymbol{\lambda}_{n_t}^T \mathbf{G}(\mathbf{u}_{n_t}))_{uu}]\mathbf{v}_{n_t} \end{pmatrix}.$$

We apply POD to the computed state and adjoint to compute $\mathbf{V} \in \mathbb{R}^{n_y \times r}$. Since states and adjoint typically have very different scales, we do not apply POD to the combined snapshots, but individually, as stated in Algorithm 3.

Algorithm 3: Construction of POD Subspace

1: Collate the solution $\mathbf{y}_1, \ldots, \mathbf{y}_{n_t}$ to the state equation and he solution $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{n_t}$ to the adjoint equation into snapshot matrices,

$$\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_{n_t}], \qquad \boldsymbol{\Lambda} = [\boldsymbol{\lambda}_1 \ldots, \boldsymbol{\lambda}_{n_t}].$$

2: Construct by truncated SVD matrices $\mathbf{V}_y \in \mathbb{R}^{n_y \times r_y}$ and $\mathbf{V}_\lambda \in \mathbb{R}^{n_y \times r_\lambda}$ so that

$$\frac{\|(\mathbf{I} - \mathbf{V}_y \mathbf{V}_y^T)\mathbf{Y}\|}{\|\mathbf{Y}\|} < \text{tol}_{\text{POD}} \qquad \text{and} \qquad \frac{\|(\mathbf{I} - \mathbf{V}_\lambda \mathbf{V}_\lambda^T)\boldsymbol{\Lambda}\|}{\|\boldsymbol{\Lambda}\|} < \text{tol}_{\text{POD}}.$$

3: Orthogonalize. $\text{orth}([\mathbf{V}_y\ \mathbf{V}_\lambda]) \mapsto \mathbf{V}$.

The ROM Hessian-time-vector computation is specified in Algorithm 4. The subspace matrix $\mathbf{V} \in \mathbb{R}^{n_y \times n_r}$ used in steps 2 and 3 is computed via Algorithm 3.

Computational Efficiency of ROM Hessian. While matrices like $\mathbf{V}^T \mathbf{F}_y(\mathbf{y}_k)\mathbf{V}$ in the ROM Hessian computation are smaller than $\mathbf{F}_y(\mathbf{y}_k)$, the dependence of $\mathbf{F}_y$ on $\mathbf{y}_k$, which changes with time step k makes the computation of $\mathbf{V}^T \mathbf{F}_y(\mathbf{y}_k)\mathbf{V}$ expensive. This well-known issue [4, 6, 9, 11, 18] is addressed via hypereduction. Specifically, we use a so-called unassembled form of the Discrete Empirical Interpolation Method (DEIM), which originates from the (D)EIM [6, 9] and is described in more detail

Algorithm 4: ROM Hessian-Times-Vector Computation—$\widetilde{\nabla^2 \widehat{J}(\mathbf{u})\mathbf{v}}$

1: Given $\mathbf{u} = (\mathbf{u}_1^T, \ldots, \mathbf{u}_{n_t}^T)^T$ let $\mathbf{y} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_{n_t}^T)^T$ be the solution of the state Eq. (22b) and let $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1^T, \ldots, \boldsymbol{\lambda}_{n_t}^T)^T$ be the solution of the adjoint Eq. (23).

2: Set $\widehat{\mathbf{w}}_0 = \mathbf{0}$ and solve the POD linearized state equation for $\widehat{\mathbf{w}}_k$,

$$[\mathbf{I} + \Delta t\, \mathbf{V}^T \mathbf{F}_y(\mathbf{y}_{k+1})\mathbf{V}]\widehat{\mathbf{w}}_{k+1} = \widehat{\mathbf{w}}_k + \Delta t \mathbf{V}^T \mathbf{G}_u(\mathbf{u}_{k+1})\mathbf{v}_{k+1}, \quad k = 0, \ldots, n_t - 1.$$

3: Solve the POD second order adjoint equation for $\widehat{\mathbf{p}}_k$,

$$\begin{aligned}
[\mathbf{I} + \Delta t\, \mathbf{V}^T \mathbf{F}_y(\mathbf{y}_{n_t})\mathbf{V}]^T \widehat{\mathbf{p}}_{n_t} &= -\mathbf{V}^T [\Delta t\, (\boldsymbol{\lambda}_{n_t}^T \mathbf{F}(\mathbf{y}_{n_t}))_{yy} + \ell_{yy}(\mathbf{y}_{n_t})]\mathbf{V}\widehat{\mathbf{w}}_{n_t}, \\
[\mathbf{I} + \Delta t\, \mathbf{V}^T \mathbf{F}_y(\mathbf{y}_k)\mathbf{V}]^T \widehat{\mathbf{p}}_k &= \widehat{\mathbf{p}}_{k+1} - \mathbf{V}^T [\Delta t\, (\boldsymbol{\lambda}_k^T \mathbf{F}(\mathbf{y}_k))_{yy} + \ell_{yy}(\mathbf{y}_k)]\mathbf{V}\widehat{\mathbf{w}}_k, \\
& \qquad k = n_t - 1, \ldots, 1.
\end{aligned}$$

4: Compute the approximation to the action of the Hessian,

$$\widetilde{\nabla^2 \widehat{J}(\mathbf{u})\mathbf{v}} = \begin{pmatrix} -\Delta t\, \mathbf{G}_u(\mathbf{u}_1)^T \mathbf{V}\widehat{\mathbf{p}}_1 + [\sigma_{uu}(\mathbf{u}_1) - \Delta t\, (\boldsymbol{\lambda}_1^T \mathbf{G}(\mathbf{u}_1))_{uu}]\mathbf{v}_1 \\ \vdots \\ -\Delta t\, \mathbf{G}_u(\mathbf{u}_{n-t})^T \mathbf{V}\widehat{\mathbf{p}}_{n_t} + [\sigma_{uu}(\mathbf{u}_{n_t}) - \Delta t\, (\boldsymbol{\lambda}_1^T \mathbf{G}(\mathbf{u}_1))_{uu}]\mathbf{v}_{n_t} \end{pmatrix}$$

in [4, 18]. This leads to a further approximation of the ROM Hessian computed by Algorithm 4. The error in this additional approximation is controlled by the tolerance tol_{DEIM}) used in DEIM. We refer to [14, Sects. 5.4, 6.2] for further details. We will use $\widetilde{\nabla^2 \widehat{J}(\mathbf{u})}$ to denote the final ROM Hessian approximation. We can replace $\mathbf{y}_k \approx \mathbf{V}\widehat{\mathbf{y}}_k$, where $\widehat{\mathbf{y}}_k = \mathbf{V}^T \mathbf{y}_k \in \mathbb{R}^r$, and $\boldsymbol{\lambda}_k \approx \mathbf{V}\widehat{\boldsymbol{\lambda}}_k$, where $\widehat{\boldsymbol{\lambda}}_k = \mathbf{V}^T \boldsymbol{\lambda}_k \in \mathbb{R}^r$ to reduce storage requirements.

4 Numerical Results

We apply the ROM Hessian approximation to semi-linear parabolic optimal control problems of the form

$$\min_{u \in L^2(\Omega \times (0,1))} \frac{1}{2} \int_0^1 \int_\Omega (y(\mathbf{x}, t; u) - y_d(\mathbf{x}, t))^2 d\mathbf{x}dt + \frac{\alpha}{2} \int_0^1 \int_\Omega u(\mathbf{x}, t)^2 d\mathbf{x}dt, \tag{25a}$$

where for given function $u \in L^2(\Omega \times (0, 1))$ the function $y(\cdot, \cdot; u)$ is the solution of

$$y_t(\mathbf{x}, t) - \Delta y(\mathbf{x}, t) + f(y(\mathbf{x}, t)) = u(\mathbf{x}, t) \qquad (\mathbf{x}, t) \in \Omega \times (0, T), \tag{25b}$$

$$y(\mathbf{x}, t) = 0 \qquad (\mathbf{x}, t) \in \partial\Omega \times (0, T), \tag{25c}$$

$$y(\mathbf{x}, 0) = y_0(\mathbf{x}) \qquad \mathbf{x} \in \Omega, \tag{25d}$$

Table 1 Example 1. Optimization using FOM Hessian (left) and ROM Hessian $\text{tol}_{\text{POD}} = 10^{-3}$, $\text{tol}_{\text{DEIM}} = 10^{-3}$ (right). Stopping criteria in both approaches $\|\nabla \widehat{J}(\mathbf{u})\| < 10^{-7}$. Both optimization approaches show nearly identical convergence behavior. Iteration i, objective function value ($\|\widehat{J}(\mathbf{u})\|$), gradient norm ($\|\nabla \widehat{J}(\mathbf{u})\|$), step-size ($\alpha$), and number of CG iterations (CG) are shown

i	$\widehat{J}(\mathbf{u}_i)$	$\|\nabla \widehat{J}(\mathbf{u}_i)\|$	α_i	CG
0	4.397932	2.7878e-04	1.00	18
1	1.665062	2.5098e-05	1.00	20
2	1.598802	4.7573e-06	1.00	25
3	1.595127	2.9272e-07	1.00	35
4	1.595110	1.4527e-09		
i	$\widehat{J}(\mathbf{u}_i)$	$\|\nabla \widehat{J}(\mathbf{u}_i)\|$	α_i	CG
0	4.397932	2.7878e-04	1.00	18
1	1.665253	2.5120e-05	1.00	20
2	1.598817	4.7625e-06	1.00	25
3	1.595127	2.9294e-07	1.00	39
4	1.595110	1.7659e-08		

and $\Omega = (0, 1)^2$. The problem is discretized in space using piecewise linear finite elements on a triangulation obtained by dividing $\Omega = (0, 1)^2$ into 40×40 squares and then dividing each square into two triangles. This results in a semi-discretization with $n_u = 1681$ and $n_y = 1521$ degrees of freedom in the control and state respectively. The resulting semi-discretization is then discretized in time using the backward Euler method using $n_t = 100$ times steps. This results in a problem of the type (22) with $\mathbf{G}(\mathbf{u}_k) = \mathbf{M}_u \mathbf{u}_k$ and $\sigma(\mathbf{u}_k) = (\alpha/2)\mathbf{u}_k^T \mathbf{M}_u \mathbf{u}_k$, where $\mathbf{M}_u$ is the mass matrix, and $\ell(\mathbf{y}_k) = (\mathbf{y}_k - \mathbf{y}_{k,d})^T \mathbf{M}(\mathbf{y}_k - \mathbf{y}_{k,d})$, where $\mathbf{M}$ is the mass matrix that also appears in (22b) ($\mathbf{M}_u$ and $\mathbf{M}$ differ in size because of bounday conditions for y) and $\mathbf{y}_{k,d}$ comes from a discretization of the desired state y_d in (25a).

All optimization runs are initialized at $\mathbf{u} = \mathbf{0}$. The truncated CG is stopped when negative curvature is detected or when the CG residual satisfies

$$\|\mathbf{H}_i\, \mathbf{d} + \nabla \widehat{J}(\mathbf{u}_i)\| \leq \min\{\|\nabla \widehat{J}(\mathbf{u}_i)\|^2, 0.01\|\nabla J(\mathbf{u}_i)\|\},$$

$\mathbf{H}_i = \nabla^2 \widehat{J}(\mathbf{u}_i)$ or $\widetilde{\nabla^2 \widehat{J}(\mathbf{u}_i)}$ depending on whether a FOM or ROM Hessian is used.

Example 1: Cubic Reaction. In the first example the nonlinearity in (25b) is cubic, $f(y) = y^3$. The desired state is $y_d(\mathbf{x}, t) = 2e^t + 2\mathbf{x}_1(\mathbf{x}_1 - 1) + 2\mathbf{x}_2(\mathbf{x}_2 - 1)$, the initial state is $y_0(\mathbf{x}, t) = \sin(2\pi \mathbf{x}_1)$, and the control penalty is $\alpha = 10^{-4}$.

Both optimization with FOM Hessians and with ROM Hessians converged to virtually the same solution. The computed optimal control u at $t = 0.1, 0.5, 1$ is shown in Fig. 2. Table 1 shows that the optimization histories for the two approaches are nearly the same.

While the optimization histories for the two approaches are nearly the same, the ROM Hessian approach is much faster as shown in Table 2. The timing reported

Table 2 Example 1. The FOM Hessian approach requires 2*98 PDE solves, which are replaced by 2*102 ROM PDE solves in the ROM Hessian approach, resulting in an overall speedup of 5.5

	FOM	ROM
State solves	5	5
Adjoint solves	5	5
Hess-Vec mult	98	102
Total time (s)	410.42	74.13

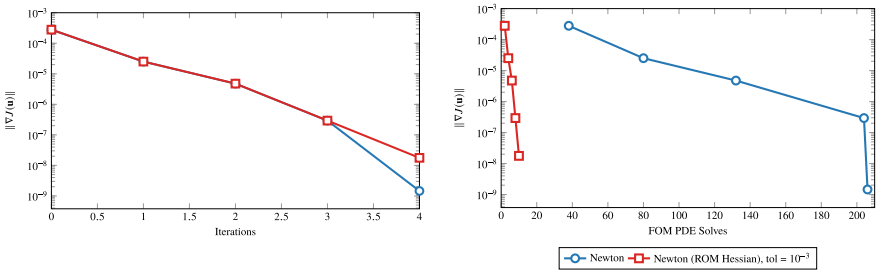

Fig. 1 Convergence history for Newton-CG with and without POD+DEIM approximations in the Hessian-vector computation

in Table 2 is for computations using MATLAB running on a MacBook Pro (13-inch, Retina, Mid 2014). The difference in computing time is due to the cost of Hessian-times-vector multiplications. In this example, using FOM Hessian requires 98 Hessian-times-vector multiplications and therefore 196 linear discretized PDE solves. In contrast, using ROM Hessian requires 102 Hessian-times-vector multiplications and therefore 204 linear ROM PDE solves.

Figure 1 is another presentation of the convergence results in Tables 1 and 2.

The left plot shows that both approaches have nearly the same iteration history. However, when convergence history is plotted against computational work performed, measured in terms of PDE solves, the ROM Hessian approach is much faster. Note that here we do not differentiate between the nonlinear state PDE solve and the linear PDE solves needed for gradient, adjoint and FOM Hessian-times-vector computations. While the discretized state Eq. (22b) is nonlinear and computing $\mathbf{y}_{k+1}$ is performed via Newton's method, only 1–2 Newton iterations per time step in the state computation are needed in this example. Therefore the difference between a nonlinear state PDE solve and linear PDE solve is small.

Example 2: Solid Fuel Ignition Model. This example is modeled after [8]. The nonlinearity in (25b) is $f(y) = -\delta e^y$ with $\delta = 5$. The desired state is $y_d(\mathbf{x}, t) = \pi^{-2}$ $\sin(\pi \mathbf{x}_1) \sin(\pi \mathbf{x}_2)$, the initial state is $y_0 \equiv \mathbf{0}$, and the penalty is $\alpha = 5 \cdot 10^{-3}$.

The numerical results for this example mirror those of the previous example. Optimization with FOM Hessians and with ROM Hessians converged to virtually the same solution, and the optimization histories for the two approaches are nearly

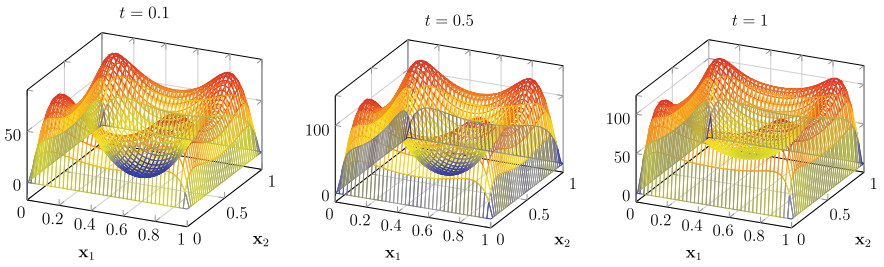

Fig. 2 Optimal control u at $t = 0.1, 0.5, 1$ for Example 1

Table 3 Example 2. Optimization using FOM Hessian (left) and and ROM Hessian $\text{tol}_{\text{POD}} = 10^{-3}$, $\text{tol}_{\text{DEIM}} = 10^{-3}$ (right). Stopping criteria in both approaches $\|\nabla \widehat{J}(\mathbf{u})\| < 10^{-8}$. Both optimization approaches show nearly identical convergence behavior. Iteration i, objetive function value ($\|\widehat{J}(\mathbf{u})\|$), gradient norm ($\|\nabla \widehat{J}(\mathbf{u})\|$), step-size ($\alpha$), and number of CG iterations (CG) are shown

i	$\widehat{J}(\mathbf{u}_i)$	$\|\nabla \widehat{J}(\mathbf{u}_i)\|$	α_i	CG
0	2.6981e-02	4.5436e-05	1.00	9
1	1.3028e-02	3.8587e-06	1.00	12
2	1.2917e-02	2.6076e-08	1.00	27
3	1.2917e-02	1.2552e-12		
i	$\widehat{J}(\mathbf{u}_i)$	$\|\nabla \widehat{J}(\mathbf{u}_i)\|$	α_i	CG
0	2.6981e-02	4.5436e-05	1.00	9
1	1.3028e-02	3.8601e-06	1.00	12
2	1.2917e-02	2.6116e-08	1.00	28
3	1.2917e-02	2.4637e-12		

Table 4 Example 2. The FOM Hessian approach requires 2*48 PDE solves, which are replaced by 2*49 ROM PDE solves in the ROM Hessian approach, resulting in an overall speedup of 4

	FOM	ROM
State solves	4	4
Adjoint solves	4	4
Hess-Vec mult	48	49
Total time (s)	234.78	56.73

the same, as shown in Table 3. The computed optimal control u at $t = 0.1, 0.5, 1$ is shown in Fig. 4.

The ROM Hessian approach is much faster as shown in Table 4. Again, the timing reported in Table 4 is for computations using MATLAB running on a MacBook Pro (13-inch, Retina, Mid 2014) and the reason for the difference in computing time is the cost of Hessian-times-vector multiplications. The ROM Hessian requires 96 ROM PDE solves for ROM Hessian-vector operations in contrast to the 96 FOM PDE solves in the FOM Hessian approach.

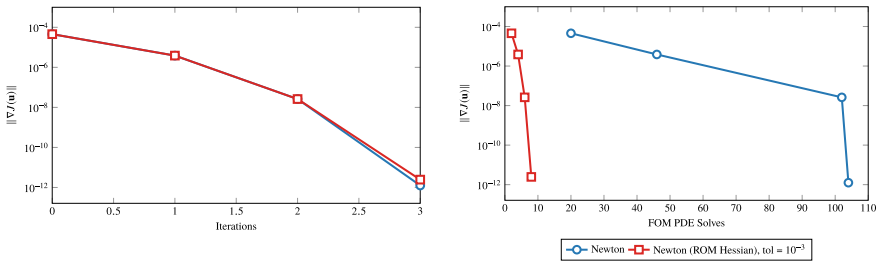

Fig. 3 Convergence history for Newton-CG with and without POD+DEIM approximations in the Hessian-vector computation

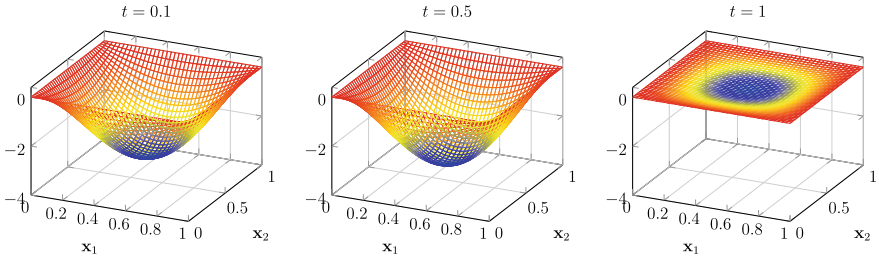

Fig. 4 Optimal control u at $t = 0.1, 0.5, 1$ for Example 2

Figure 3 shows that when convergence history is plotted not against iteration count, but against computational work performed, measured in terms of PDE solves, the ROM Hessian approach is much faster.

5 Conclusions

We have introduced ROM Hessian approximations for use in inexact Newton methods for large-scale smooth optimization problems obtained from discretizations of optimal control problems governed by parabolic PDEs. In contrast to other ROM approaches, which approximate the optimization problem, our approach retains the original FOM objective function and gradient. The Hessian ROM is computed by applying POD to state and adjoint snapshots that have to be computed anyway, and therefore ROM computation is relatively inexpensive. Since original objective functions and gradients are used, the qualitative global convergence properties of the original line-search Newton method and the line-search Newton method with ROM Hessian approximation are the same. However, since Hessian approximations are significantly cheaper, the ROM Hessian approach can lead to substantial savings. This is confirmed by numerical experiments with two semilinear parabolic optimal control problems. The two optimization approaches had essentially the same convergence behavior, but the ROM Hessian approach had a factor 5–8 speedup because

of computational savings due to the Hessian approximations. However, it is possible to construct examples, where the ROM Hessian approach does not lead to computational savings. When the ROM Hessian too poorly approximates the true one, the ROM Hessian approach can require substantially more optimization iterations, which erase savings enjoyed within an optimization iteration. Additional analysis and numerical tests are part of future work.

Acknowledgements The research in this paper was performed before Caleb Magruder joined MathWorks. This research was funded in part through a sponsored research agreement with the ExxonMobil Upstream Research Company and by NSF grants CCF-1816219 and DMS-1819144.

References

1. Afanasiev, K., Hinze, M.: Adaptive control of a wake flow using proper orthogonal decomposition. In: Shape Optimization and Optimal Design (Cambridge, 1999), Lecture Notes in Pure and Appl. Math., vol. 216, pp. 317–332. Dekker, New York (2001)
2. Antil, H., Heinkenschloss, M., Hoppe, R.H.W.: Domain decomposition and balanced truncation model reduction for shape optimization of the Stokes system. Optim. Methods Softw. **26**(4–5), 643–669 (2011). http://dx.doi.org/10.1080/10556781003767904
3. Antil, H., Heinkenschloss, M., Hoppe, R.H.W., Sorensen, D.C.: Domain decomposition and model reduction for the numerical solution of PDE constrained optimization problems with localized optimization variables. Comput. Vis. Sci. **13**, 249–264 (2010). https://doi.org/10.1007/s00791-010-0142-4
4. Antil, H., Heinkenschloss, M., Sorensen, D.C.: Application of the discrete empirical interpolation method to reduced order modeling of nonlinear and parametric systems. In: Quarteroni, A., Rozza, G. (eds.) Reduced Order Methods for Modeling and Computational Reduction, MS&A. Model. Simul. Appl., vol. 9, pp. 101–136. Springer Italia, Milan (2014). https://doi.org/10.1007/978-3-319-02090-7_4
5. Antoulas, A.C.: Approximation of Large-Scale Dynamical Systems. Advances in Design and Control, vol. 6. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2005). https://doi.org/10.1137/1.9780898718713
6. Barrault, M., Maday, Y., Nguyen, N.D., Patera, A.T.: An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. C. R. Math. Acad. Sci. Paris **339**(9), 667–672 (2004). https://doi.org/10.1016/j.crma.2004.08.006
7. Benner, P., Sachs, E., Volkwein, S.: Model order reduction for PDE constrained optimization. In: Leugering, G., Benner, P., Engell, S., Griewank, A., Harbrecht, H., Hinze, M., Rannacher, R., Ulbrich, S. (eds.) Trends in PDE Constrained Optimization, Internat. Ser. Numer. Math., vol. 165, pp. 303–326. Birkhäuser/Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05083-6_19
8. Borzì, A., Kunisch, K.: The numerical solution of the steady state solid fuel ignition model and its optimal control. SIAM J. Sci. Comput. **22**(1), 263–284 (electronic) (2000). https://doi.org/10.1137/S1064827599360194
9. Chaturantabut, S., Sorensen, D.C.: Nonlinear model reduction via discrete empirical interpolation. SIAM J. Sci. Comput. **32**(5), 2737–2764 (2010). https://doi.org/10.1137/090766498
10. Fahl, M., Sachs, E.: Reduced order modelling approaches to PDE–constrained optimization based on proper orthogonal decompostion. In: Biegler, L.T., Ghattas, O., Heinkenschloss, M., van Bloemen Waanders, B. (eds.) Large-Scale PDE-Constrained Optimization, Lecture Notes in Computational Science and Engineering, vol. 30, pp. 268–280. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-642-55508-4_16

11. Farhat, C., Chapman, T., Avery, P.: Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models. Int. J. Numer. Methods Eng. **102**(5), 1077–1110 (2015). https://doi.org/10.1002/nme.4820
12. Hinze, M., Pinnau, R., Ulbrich, M., Ulbrich, S.: Optimization with PDE Constraints. Mathematical Modelling, Theory and Applications, vol. 23. Springer, Heidelberg, New York, Berlin (2009). https://doi.org/10.1007/978-1-4020-8839-1
13. Kouri, D.P., Heinkenschloss, M., Ridzal, D., van Bloemen Waanders, B.G.: Inexact objective function evaluations in a trust-region algorithm for PDE-constrained optimization under uncertainty. SIAM J. Sci. Comput. **36**(6), A3011–A3029 (2014). https://doi.org/10.1137/140955665
14. Magruder, C.: Projection-based model reduction in the context of optimization with implicit PDE constraints. Ph.D. thesis, Department of Computational and Applied Mathematics, Rice University, Houston, TX (2017)
15. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer Verlag, Berlin, Heidelberg, New York (2006). https://doi.org/10.1007/978-0-387-40065-5
16. Qian, E., Grepl, M.A., Veroy, K., Wilcox, K.: A certified trust region reduced basis approach to PDE-constrained optimization. SIAM J. Sci. Comput. **39**(5), S434–S460 (2017). https://doi.org/10.1137/16M1081981
17. Sachs, E.W., Volkwein, S.: POD-Galerkin approximations in PDE-constrained optimization. GAMM-Mitteilungen **33**(2), 194–208 (2010). https://doi.org/10.1002/gamm.201010015
18. Tiso, P., Rixen, D.J.: Discrete empirical interpolation method for finite element structural dynamics. In: Kerschen, G., Adams, D., Carrella, A. (eds.) Topics in Nonlinear Dynamics, vol. 1, Conference Proceedings of the Society for Experimental Mechanics Series, vol. 35, pp. 203–212. Springer New York (2013). https://doi.org/10.1007/978-1-4614-6570-6_18
19. Yue, Y., Meerbergen, K.: Accelerating optimization of parametric linear systems by model order reduction. SIAM J. Optim. **23**(2), 1344–1370 (2013). https://doi.org/10.1137/120869171
20. Zou, Z., Kouri, D.P., Aquino, W.: An adaptive local reduced basis method for solving PDEs with uncertain inputs and evaluating risk. Comput. Methods Appl. Mech. Engrg. **345**, 302–322 (2019). https://doi.org/10.1016/j.cma.2018.10.028