Commun. Comput. Phys.
doi: 10.4208/cicp.190311.270112a

Vol. **12**, No. 5, pp. 1520-1540
November 2012

System Reduction Using an LQR-Inspired Version of Optimal Replacement Variables

Alex Solomonoff[1,2,*]

[1] *Camberville Research Institute, Somerville, MA, USA.*
[2] *Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong.*

Received 19 March 2011; Accepted (in revised version) 27 January 2012

Communicated by Wei Cai

Available online 4 June 2012

Abstract. Optimal Replacement Variables (ORV) is a method for approximating a large system of ODEs by one with fewer equations, while attempting to preserve the essential dynamics of a reduced set of variables of interest. An earlier version of ORV [1] had some issues, including limited accuracy and in some rare cases, instability. Here we present a new version of ORV, inspired by the linear quadratic regulator problem of control theory, which provides better accuracy, a guarantee of stability and is in some ways easier to use.

AMS subject classifications: 65M06, 65M30, 65M70

Key words: System reduction, optimal replacement variables, resolved variables, optimal prediction.

1 Introduction

The problem studied in this paper is that of approximating a large system of ordinary differential equations (ODEs) by one with a smaller number of equations.

This is desirable in many situations. For example, many partial differential equations, when discretized into a system of ODEs, require millions of degrees of freedom (DOF) to adequately approximate the dynamics. However, most of these DOF are usually of no interest. Examples of such PDEs include weather simulations and many simulations of fluid or aerodynamic flows. In the flow-around-an-aircraft example, an engineer would be mainly interested in bulk features such as the total lift and drag, or average vorticity as a function of time. A flow field detailed enough to actually resolve all of the dynamics would not be needed in many situations. Examples of linear systems of interest include

[*]Corresponding author. *Email address:* alex.solomonoff@yahoo.com (A. Solomonoff)

the discretized versions of any linear PDE with complex geometry, such as heat flow in electronic devices or structural dynamics of skyscrapers or aircraft. Calculating solutions to these equations can require quite large amounts of computing resources, and a system reduction method such as the one studied here has the potential to reduce the resources required, or allow the fast solution of more complex problems.

The version of the problem studied here is a system of ODEs

$$z_t = Fz = \begin{pmatrix} F_0 & F_1 \\ F_2 & F_3 \end{pmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \quad t \geq 0, \quad z = \begin{bmatrix} x \\ y \end{bmatrix}. \tag{1.1}$$

Only the case of linear systems of ODEs is considered here. Nonlinear ODEs are a topic for future work.

The state vector $z \in \mathbb{R}^{m+n}$ is divided into a set of resolved variables $x \in \mathbb{R}^m$ that one wants to observe or calculate, and a set of unresolved variables $y \in \mathbb{R}^n$ that one doesn't need to observe, but which the dynamics of the resolved variables x depend on. Furthermore, it may be that $m \ll n$ or even n infinite, in which case it will not be computationally feasible to solve the full system of equations.

An overview of different approaches to the problem can be found in Givon, Kupferman, Stuart [9]. One approach for system reduction has been developed by Chorin et al. [5–7] and Chertock, Gottlieb & Solomonoff [8], which has been called the T-System or Optimal Prediction.

Another approach, called Optimal Replacement Variables (ORV), see Solomonoff & Don [1] attempts to find a low-dimensional replacement for the unresolved variables such that the influence of the unresolved variables on the resolved ones is adequately approximated by the replacement variables. This is the approach which the present paper builds upon.

1.1 Other types of system reduction

There are several versions of the system reduction problem. We will call the one of this paper, the Optimal Prediction Problem (OPP). Others are similar in flavor, but differ enough in details that techniques used for one problem are of little use in another.

1.1.1 Control theory system reduction problem

The canonical control theory system is

$$x_t = Ax + Bu, \qquad y = Cx, \tag{1.2}$$

where u is a low-dimensional control input, y is a low-dimensional output and x is a high-dimensional internal state, and the goal is to construct a low-dimensional system that approximates the relation between the inputs and the outputs. Long-time behavior is emphasized and initial conditions tend to be ignored. This problem is quite well studied [15,16].

This control theory system reduction problem has the concept of control inputs, which the problem of this paper lacks. In OPP, the initial conditions are the only inputs to the problem. There are intriguing similarities between the two problems — the control theory outputs look a lot like the resolved variables of the current problem. Nevertheless, I do not know of any way of using the control theory techniques for OPP.

1.1.2 Proper orthogonal decomposition

This approach, due to Sirovich [13] makes the assumption that the system solution always lies close to some low-dimensional linear subspace of the phase space. It finds the subspace by examining many values of the system solution, then projects the full system onto the subspace. It requires that the low-dimensional-subspace assumption be valid. It has no concept of resolved variables and attempts to approximate the entire system.

1.1.3 No-replacement-variables approach of Chorin's T-system

Chorin's T-System attacks a very similar problem to OPP, but attempts to approximate the system using only the resolved variables. Since the unresolved variables clearly influence the resolved ones, this must be a limited approach.

The framework of Chorin's T-System includes a probability distribution placed on the initial conditions, and then only assumes that their resolved part is available. Instead of approximating the solution, the conditional expectation of the solution, given only the resolved initial conditions is desired. But the unresolved variables influence the expected-resolved variables as well.

1.2 Drawbacks of ORV

The version of ORV presented in [1] had some issues needing to be addressed:

1. It required the user to provide an approximate probability distribution for the solution of the ODE on its phase space, and assumed that it was valid for all time. In most cases the true probability varies with time, of course, but a crude approximation of it might very well be able to be time-invariant. Nevertheless, providing the approximate probability is a burden on the user, and the fact that only a strange, crude approximation is possible is unappealing.
2. Stability. The reduced system generated by the ORV technique was not guaranteed to be stable, and in some rare cases it was slightly unstable. It would be desirable to be able to guarantee the stability of the system.
3. Performance. In [1], ORV was examined using two test problems. In one problem it gave more accurate dynamics than all the competing methods used, but on the other, it was only able to equal the accuracy of the best competing method, and that best competitor was quite simple. A method with better accuracy would be desirable.

4. ORV requires that an error measure be minimized, and this is done numerically. If the error measure has multiple local minima, the minimization can be quite difficult. In one of the two test problems used in [1], the error measure had no more than a few minima, but in the other it had a great many, requiring a time-consuming global minimization using multiple random starts. An error measure that had only a single local minimum would be desirable.

This paper presents an improved version of ORV that addresses these problems.

2 ORV setup and framework

Most of the framework is borrowed from [1]. The goal of the project is approximation of the system (1.1) with the unresolved variables replaced by a smaller set $u \in \mathbb{R}^k$, $k \ll n$, approximately preserving the dynamics. The output of the ORV process is two matrices G and R. The replacement variables u are related to the unresolved variables through R:

$$u \approx u_z = Ry, \qquad P = \begin{pmatrix} I & 0 \\ 0 & R \end{pmatrix}, \qquad w = \begin{bmatrix} x_* \\ u \end{bmatrix} \approx w_z = \begin{bmatrix} x \\ Ry \end{bmatrix} = Pz. \tag{2.1}$$

The matrix G defines a new, approximate system of ODEs

$$w_t = Gw, \qquad w(t=0) = Pz_0 \tag{2.2}$$

and G and R are chosen in such a way as to optimally reproduce the resolved part of the dynamics of (1.1). Note that G and R do not vary in time.

Define some "subvector-extracting" matrices:

$$J_{xz} = \begin{pmatrix} I_m & 0_{m\times n} \end{pmatrix}, \qquad J_{xw} = \begin{pmatrix} I_m & 0_{m\times k} \end{pmatrix}, \tag{2.3}$$

so for any appropriately-sized vectors x, y, x_* and u,

$$J_{xz} \begin{bmatrix} x \\ y \end{bmatrix} = x, \qquad J_{xw} \begin{bmatrix} x_* \\ u \end{bmatrix} = x_*. \tag{2.4}$$

The original error criterion in [1] was

$$e_{old} = \mathrm{E}_z \|PFz - GPz\|^2, \tag{2.5}$$

where E_z is expectation over the user-supplied, suitable-for-all-time probability density on z. This needs to weigh the error in the dynamics of u even though we only care about the accuracy of x. This is necessary because the dynamics of the replacement variables have to be accurate before their influence on the resolved variables has any hope of being correct.

The probability density on z was assumed to be a zero-mean multivariable Gaussian density, with the covariance matrix supplied by the user:

$$z \sim \mathcal{N}(0, S_{old}).$$

We now present a new, more natural dynamic error measure, inspired by the linear quadratic regulator problem of control theory, (see, e.g., Kwakernaak and Sivan [10]):

$$e = e_{new} = E_{z_0}\Big(\int_0^\infty e^{-2\alpha t}\|J_{xz}z(t) - J_{xw}w(t)\|^2 dt\Big), \tag{2.6}$$

where z and w are defined in (1.1) and (2.2).

The expectation E_{z_0} is taken over a probability density on the initial conditions of z. This is taken to be zero-mean and have a covariance matrix

$$E(z_0 z_0^T) = S = \begin{pmatrix} S_0 & S_1 \\ S_1^T & S_3 \end{pmatrix}, \tag{2.7}$$

which is provided by the user. Two examples of user-provided covariance matrices are given in the test problems in Section 6.

The error e_{new} is exactly a weighted time-average of the error in the resolved variables.

The $e^{-2\alpha t}$ weight is a way of telling the scheme you are not interested in the solution far in the future. $\alpha \geq 0$ is a user-chosen parameter, reflecting how far in the future the user is interested in. α can be zero if desired.

In computations in this paper, α was usually chosen so that the temporal weight had the same expected value as the mean of the time interval, i.e.,

$$\int_0^\infty t e^{-2\alpha t} dt \Big[\int_0^\infty e^{-2\alpha t} dt\Big]^{-1} = \frac{1}{2} t_{final} \longrightarrow \alpha = \frac{1}{t_{final}}. \tag{2.8}$$

Note that in the case of $\alpha = 0$, if F is a stable[†] matrix, then G must also be stable or else $e_{new} = \infty$. Since e_{new} is finite for any stable G, it follows that the G that minimizes e_{new} is guaranteed to be stable. If $\alpha > 0$ then only $G - \alpha I$ is guaranteed to be stable.

The probability density needs to be provided by the user, but it requires no intuition about the dynamics of the system and does not require that a single density represent a density that changes with time. One can argue that the initial density is legitimately part of the problem definition. Some issues surrounding the probability density are presented in the discussion section of [1].

We don't actually need to assume z_0 is Gaussian — the calculations only ever depend on the covariance matrix. Any distribution having the same covariance would generate the same reduced system. (This is true for the original version of ORV as well.)

[†]A matrix M is called stable if its eigenvalues all lie in the left half-plane, or equivalently, if $\lim_{t\to\infty} e^{Mt} = 0$.

3 Reducing the error expression to something computable

The error, in the form of (2.6) with its expectation and integral in time, would be very difficult to evaluate computationally. The next few sections derive an expression for the error that is suitable for numerical computations.

Define

$$D = \begin{pmatrix} J_{xz} & -J_{xw} \end{pmatrix}, \tag{3.1a}$$

$$H = \begin{pmatrix} F & 0 \\ 0 & G \end{pmatrix}, \qquad H_\alpha = H - \alpha I = \begin{pmatrix} F-\alpha I & 0 \\ 0 & G-\alpha I \end{pmatrix}, \tag{3.1b}$$

$$K_{vz} = \begin{bmatrix} I \\ P \end{bmatrix}, \quad \text{so that } v = \begin{bmatrix} z \\ w \end{bmatrix} = K_{vz} z. \tag{3.1c}$$

Note the double partitioning of v:

$$v = \begin{bmatrix} x \\ y \\ x_* \\ u \end{bmatrix}. \tag{3.2}$$

Define the instantaneous error vector

$$d(t) \equiv J_{xz} z(t) - J_{xw} w(t) = D e^{Ht} K_{vz} z_0, \tag{3.3a}$$

$$e^{-\alpha t} d(t) = D e^{H_\alpha t} K_{vz} z_0. \tag{3.3b}$$

The error for a given initial condition z_0 is

$$\begin{aligned} e(z_0) &= \int_0^\infty e^{-2\alpha t} \|d(t)\|^2 dt = \int_0^\infty e^{-2\alpha t} z_0^T K_{vz}^T e^{H^T t} D^T D e^{Ht} K_{vz} z_0 dt \\ &= \int_0^\infty z_0^T K_{vz}^T e^{H_\alpha{}^T t} D^T D e^{H_\alpha t} K_{vz} z_0 dt = z_0^T K_{vz}^T Q K_{vz} z_0, \end{aligned} \tag{3.4}$$

where

$$Q = \int_0^\infty e^{H_\alpha{}^T t} A e^{H_\alpha t} dt, \qquad A = D^T D. \tag{3.5}$$

The expected error (2.6) is

$$\begin{aligned} e &= E_{z_0}\big(e(z_0)\big) = \mathrm{E}\big(z_0^T K_{vz}^T Q K_{vz} z_0\big) \\ &= \mathrm{tr}(S K_{vz}^T Q K_{vz}) = \mathrm{tr}(K_{vs} S K_{vz}^T Q) = \mathrm{tr}(S_v Q), \end{aligned} \tag{3.6}$$

where

$$S_v = K_{vz} S K_{vz}^T = \begin{pmatrix} S & SP^T \\ PS & PSP^T \end{pmatrix}. \tag{3.7}$$

Dependencies

- S_v and K_{vz} depend on R and not on G;
- H and Q depend on G and not R;
- D and A do not depend on either R or G;
- None of them depend on t.

4 Necessary tools and identities

This section presents various identities and tools which are used in the presentation of the rest of the paper.

Notation

If M is any square matrix, we implicitly partition it 2×2 into submatrices and label the submatrices in the following way:

$$M = \begin{pmatrix} M_0 & M_1 \\ M_2 & M_3 \end{pmatrix}.$$

This applies both to named matrices and matrix-valued expressions, so $(\cdots)_3$ refers to the lower-right submatrix of $(\cdots)$. This gives, for example

$$\mathrm{tr}\big(S_v Q\big) = \mathrm{tr}(SQ_0) + 2\mathrm{tr}(PSQ_1) + \mathrm{tr}(PSP^T Q_3).$$

In addition, since

$$w = \begin{bmatrix} x_* \\ u \end{bmatrix}$$

is partitioned, the matrix Q_3 is also partitioned:

$$Q_3 = \begin{pmatrix} Q_{30} & Q_{31} \\ Q_{31}^T & Q_{33} \end{pmatrix}. \tag{4.1}$$

Facts about the trace

Most of these can be found in Wikipedia [12]:

$$\mathrm{tr}(M) \equiv \sum_i M_{ii} \qquad \text{(definition)}, \tag{4.2a}$$

$$\|A\|_F^2 \equiv \sum_{ij} A_{ij}^2 = \mathrm{tr}(A^T A) = \mathrm{tr}(AA^T), \tag{4.2b}$$

$$\mathrm{tr}(AB^T) = \mathrm{tr}(A_0 B_0^T) + \mathrm{tr}(A_1 B_1^T) + \mathrm{tr}(A_2 B_2^T) + \mathrm{tr}(A_3 B_3^T), \tag{4.2c}$$

$$\mathrm{tr}(AB) = \mathrm{tr}(BA) = \mathrm{tr}(A^T B^T) = \mathrm{tr}(B^T A^T). \tag{4.2d}$$

This is true for vectors too:

$$x^T Ax = \mathrm{tr}(x^T Ax) = \mathrm{tr}(Axx^T), \qquad \mathrm{E}(x^T Ax) = \mathrm{tr}\big(\mathrm{E}(xx^T)A\big).$$

Matrix inner product:

$$\langle A,B\rangle \equiv \mathrm{tr}(AB^T) = \mathrm{tr}(A^T B) = \sum_{ij} A_{ij}B_{ij}$$

is the matrix analog of the vector inner product $\langle a,b\rangle = a^T b$.

Trace and derivative

The following is key algebraic tool for later derivations: Let $g(M)$ be a scalar-valued function of a matrix. Note that the gradient $\nabla g(M)$ is a matrix. Write the directional derivative of $g(M)$, in the direction M', as $\nabla_{M'} g(M)$. This can be defined in a couple of different ways:

$$\nabla_{M'} g(M) = \frac{\partial}{\partial s} g(M+sM')\Big|_{s=0} = \langle M', \nabla g(M)\rangle = \mathrm{tr}\big((M')^T \nabla g\big). \tag{4.3}$$

Note that for fixed M', $\nabla_{M'} g(M)$ is a scalar. Also note that s is a dummy variable here — the directional derivative does not depend on s.

Now if for every M', $\nabla_{M'} g(M)$ should happen to take the form

$$\nabla_{M'} g(M) = \mathrm{tr}\big(G(M)(M')^T\big) \tag{4.4}$$

for some matrix-valued function $G(M)$, then

$$G(M) = \nabla g(M). \tag{4.5}$$

Moreover, if

$$M = \begin{pmatrix} M_0 & M_1 \\ M_2 & M_3 \end{pmatrix}, \quad \text{then } \frac{dg}{d(M_3)_{kl}} = \big[G_3(M)\big]_{kl}, \tag{4.6}$$

and we write

$$\frac{dg}{d(M_3)} = G_3(M) \tag{4.7}$$

to refer to every k, l at once.

4.1 Lyapunov and Sylvester equations

This subsection presents some (mostly well-known, see for example Horn & Johnson [11]) results about the Lyapunov equation necessary for derivations later.

Consider the Sylvester equation

$$A^T X + XB + C = 0. \tag{4.8}$$

The important special case of $A = B$ is called the Lyapunov equation. It has some special properties: if C is symmetric then X is also symmetric. Also, if C is positive definite then X is positive definite if and only if A is stable — this is an important theorem in control theory.

Define the linear matrix operator

$$L_{A,B}(X) = -(A^T X + XB), \tag{4.9}$$

and then (4.8) is written as

$$L_{A,B}(X) = C, \qquad X = L^{-1}_{A,B}(C).$$

If $A = B$, then we write $L_{A,A}(X) = L_A(X)$. If A and B are stable matrices then it can be shown that

$$L^{-1}_{A,B}(C) = \int_0^\infty e^{A^T t} C e^{Bt} dt. \tag{4.10}$$

Derivative of Lyapunov

Suppose A and C depend on some scalar parameter s and let $'$ refer to the derivative with respect to s. Then

$$(A')^T X + A^T X' + X'A + XA' + C' = 0, \tag{4.11a}$$
$$L_A(X') + L_{A'}(X) = C', \tag{4.11b}$$
$$X' = L_A^{-1}\big(C' - L_{A'}(X)\big). \tag{4.11c}$$

Partitioned equations

Suppose A and B are block diagonal and C can be partitioned:

$$A = \begin{pmatrix} A_0 & 0 \\ 0 & A_3 \end{pmatrix}, \qquad B = \begin{pmatrix} B_0 & 0 \\ 0 & B_3 \end{pmatrix}, \qquad C = \begin{pmatrix} C_0 & C_1 \\ C_2 & C_3 \end{pmatrix}.$$

Then Eq. (4.8) can be partitioned as

$$X = \begin{pmatrix} L^{-1}_{A_0,B_0}(C_0) & L^{-1}_{A_0,B_3}(C_1) \\ L^{-1}_{A_3,B_0}(C_2) & L^{-1}_{A_3,B_3}(C_3) \end{pmatrix}. \tag{4.12}$$

Numerical solution of the Sylvester equation

It can be shown that if M, N are two invertible matrices then (4.8) can be transformed as follows:

$$L_{A,B}^{-1}(C)=M^{-T}L_{M^{-1}AM,N^{-1}BN}^{-1}(M^THN)N^{-1}.$$

In particular, if A and B are diagonalizable then (4.8) can be transformed into a diagonal form, or it can be transformed into a triangular form using a Schur decomposition of A and B.

The standard algorithm for solving (4.8), due to Bartels and Stewart [2], reduces A and B to (almost-) triangular form using a Schur decomposition, and then solves the triangular Sylvester equation directly, in a process somewhat analogous to the backsubstitution process of solving a triangular linear system of equations. This is an $\mathcal{O}(n^3)$ algorithm, available in a number of computing environments, such as LAPACK and Matlab. Note that the problem has $\mathcal{O}(n^2)$ inputs, so the $\mathcal{O}(n^3)$ operation count is unlikely to be improved upon unless the matrices are structured.

Large sparse problems

Algorithms for solving large sparse Sylvester and Lyapunov equations exist, see [14,15], usually iterative methods involving Krylov subspace projections or the matrix sign function. They usually assume the solution is a matrix approximately of low rank.

While these methods can probably be applied to the present computations, the Lyapunov equations are only part of the computation of the reduced system, and so extending the current work to large sparse problems is far outside the scope of this paper.

5 The computable error function, part II

The goal is to find matrices G and R that minimize the error e. This requires a numerical minimization process, and so e and its gradient with respect to G and R need to be easily computable. Start with the computation of e:

The matrix Q is the solution of the Lyapunov equation

$$Q=L_{H_\alpha}^{-1}(A), \tag{5.1}$$

where H_α is defined by Eq. (3.1b). The error is then computed by

$$e=\mathrm{tr}(S_vQ)=\mathrm{tr}(SQ_0)+2\mathrm{tr}(PSQ_1)+\mathrm{tr}(PSP^TQ_3). \tag{5.2}$$

Computing the gradient of e is slightly more complex. Let $Q'=\nabla_{H_\alpha'}Q$ be the derivative of Q with respect to H_α in a (any) direction $H_\alpha{}'$. In a minimization procedure, F and α are fixed, so the only part of H_α we are interested in varying is G,

$$H_\alpha{}'=\begin{pmatrix} 0 & 0 \\ 0 & G' \end{pmatrix} \tag{5.3}$$

so essentially differentiation with respect to H_α and G are the same. Then,

$$\begin{aligned}\nabla_{G'}e=&\mathrm{tr}\big(S_vQ'\big)=-\mathrm{tr}\big(S_vL_{H_\alpha}^{-1}\big(L_{H'_\alpha}(Q)\big)\big)=\mathrm{tr}\big(S_vL_{H_\alpha}^{-1}(H_\alpha{}'^TQ+QH_\alpha{}')\big)\\ =&\mathrm{tr}\big(S_v\int_0^\infty e^{H_\alpha{}^Tt}(H_\alpha{}'^TQ+QH_\alpha{}')e^{H_\alpha t}dt\big)=2\mathrm{tr}\big(S_v\int_0^\infty e^{H_\alpha{}^Tt}QH_\alpha{}'e^{H_\alpha t}dt\big)\\ =&2\mathrm{tr}\big(H_\alpha{}'\int_0^\infty e^{H_\alpha t}S_ve^{H_\alpha{}^Tt}dtQ\big)=2\mathrm{tr}\big(H_\alpha{}'L_{H_\alpha{}^T}^{-1}(S_v)Q\big).\end{aligned}\tag{5.4}$$

Define

$$Z=L_{H_\alpha{}^T}^{-1}(S_v),\tag{5.5}$$

and then

$$\frac{\partial e}{\partial G}=2\mathrm{tr}\big(H_\alpha{}'(QZ)^T\big)=2\big(QZ\big)_3=2\big(Q_1^TZ_1+Q_3Z_3\big).\tag{5.6}$$

Note that Z and Q are symmetric matrices. If H_α is stable, then in addition Z is positive definite and Q is at least positive semidefinite.

5.1 Computing the optimal R

Recall that

$$e=\mathrm{tr}(SQ_0)+\mathrm{tr}(2PSQ_1+PSP^TQ_3).$$

Since for optimization, R is the only part of P that varies, perturbations in P and R are related:

$$P'=\frac{\partial P}{\partial R}=\begin{pmatrix}0&0\\0&R'\end{pmatrix},\tag{5.7}$$

and then one has

$$\frac{1}{2}\frac{\partial e}{\partial R}=\mathrm{tr}(P'SQ_1)+\mathrm{tr}(P'SP^TQ_3)=\big(SQ_1+SP^TQ_3\big)_3.\tag{5.8}$$

Note that for any suitably-sized matrix or matrix-valued expression M, M_3 is the submatrix of M corresponding to R in P.

Recall that S, Q_1, and Q_3 are all partitioned matrices, so (5.8) can be written as

$$\begin{aligned}\frac{1}{2}\frac{\partial e}{\partial R}=&\left(\begin{pmatrix}S_0&S_1\\S_1^T&S_3\end{pmatrix}\begin{pmatrix}Q_{10}&Q_{11}\\Q_{12}&Q_{13}\end{pmatrix}+\begin{pmatrix}S_0&S_1\\S_1^T&S_3\end{pmatrix}\begin{pmatrix}I&0\\0&R^T\end{pmatrix}\begin{pmatrix}Q_{30}&Q_{31}\\Q_{31}^T&Q_{33}\end{pmatrix}\right)_3\\ =&S_1^TQ_{11}+S_3Q_{13}+S_1^TQ_{31}+S_3R^TQ_{33}.\end{aligned}\tag{5.9}$$

Then $\partial e/\partial R=0$ gives the optimal R for a given G,

$$R_* = \arg\min_R e(R,G) = -Q_{33}^{-1}\left(Q_{13}^T + (Q_{11}^T + Q_{31}^T)S_1 S_3^{-1}\right). \tag{5.10}$$

Note that A_3 is only positive semidefinite, so Q_3 and Q_{33} are only guaranteed to be positive semidefinite and Q_{33} is not guaranteed to be invertible or well-conditioned.

We can easily compute R_* for any G and then take

$$e(G) = e\big(G, R_*(G)\big).$$

The derivative of e would seem to have to take the variation of R with G into account:

$$\frac{de}{dG} = \frac{\partial e}{\partial G} + \frac{\partial e}{\partial R}\frac{\partial R}{\partial G} \tag{5.11}$$

but with this choice of $R=R_*$, $\partial e/\partial R=0$ and the second term is zero.

Calculating e and e'

The error e and its gradient de/dG are calculated as follows:

First a preprocessing step:

1. Compute the Schur decomposition of F: $F = UTU^T$.
2. Calculate Q_0 using U, T and the Bartels-Stewart (B-S) algorithm.
3. Calculate $e_0 = \mathrm{tr}(SQ_0)$.

Then for each G,

1. Compute the Schur decomposition of G: $G = VWV^T$.
2. Calculate Q_1 and Q_3 using U, T, V, W, the B-S algorithm and (5.1).
3. Calculate the optimum R using Q_1 and Q_3 and (5.10).
4. Calculate e using e_0, Q_1, Q_3, R, and Eq. (5.2).
5. Calculate Z_1 and Z_3 using R, U, T, V, W, the B-S algorithm, and (5.5).
6. Calculate $\partial e/\partial G$ using Eq. (5.6).

6 Numerical results

All computations were done using Octave [4], an open-source Matlab-like software, except the error $e(G)$ was minimized using the NLopt [3] numerical minimization package.

In order to get a full picture of the performance of the ORV system, we compare it to various competing systems on two different test problems:

6.1 Test problems

6.1.1 Random-matrix ODE

A linear system of ODEs $z_t = Fz$, where F is a random (neutrally) stable matrix. Its entries are IID 0-1 Gaussian random numbers. It is stabilized by computing its eigenvalues and adding the multiple of the identity that just makes the matrix stable. Initial conditions are an IID 0-1 Gaussian random vector, i.e., $S = I$.

For readers wishing to duplicate this ODE in their own computations, the details of which random numbers were used are described in Appendix A.

Properties of the Random-matrix problem:

1. No special qualities; less likely to have anomalously good or bad results than other single problems.
2. The different modes are strongly coupled.
3. It is neither strongly damped nor strongly stable.
4. There is no simple way (that I know of) of constructing a good set of replacement variables.

Single random problem vs. ensemble

One can argue that good results on a single random matrix problem might just be luck, and to be sure, we should test an ensemble of random matrix problems and compute average error, max error, etc.

This is a valid (and demanding) position, but not unassailable. One could make the same objection to any single system of ODEs, especially one that had a free parameter that could be varied.

The opposite position also seems valid, that an ODE system constructed from the output of a (pseudo-) random number generator is just as valid a single example as any other. My intuition is that this random matrix problem is less likely to have anomalously good or bad performance than other systems of ODEs.

6.1.2 Cosine transform of the variable-coefficient heat equation

The expressions

$$u_t = \big(2 + \cos(x)\big) u_{xx}, \quad \text{with} \quad u(x,t) = \sum_{k=0}^{\infty} u_k(t)\cos(kx).$$

The cosine expansion of the solution implies periodic boundary conditions, a domain of length 2π, and a solution that is even about $x = 0$ for all time.

The PDE can be converted into the (infinite) tridiagonal system of ODEs

$$\frac{d}{dt}u_k = -\frac{1}{2}(k-1)^2 u_{k-1} - 2k^2 u_k - \frac{1}{2}(k+1)^2 u_{k+1}. \tag{6.1}$$

This infinite system is truncated at a fixed number $m+n$ of equations. The truncation is in the Galerkin manner, by forcing $u_k=0, \forall k \geq n+m$. Initial coefficients are independent zero-mean Gaussian random numbers with variance

$$S_{ij}=\delta_{ij}\gamma^j, \tag{6.2}$$

with $0<\gamma<1$ a user-settable parameter. The exponential decay of S_{ii} corresponds to the assumption that the initial conditions $u(x,t=0)$ is a smooth function, (analytic in fact) and faster decay with respect to j corresponds to greater smoothness.

Properties of the heat-equation problem:

1. The high modes are strongly damped, and the system is somewhat stiff.
2. Different modes are somewhat weakly coupled.
3. Has a simple way of constructing good replacement variables.
4. Of special interest because of its physical origin.

6.2 Competing systems

1. Galerkin:

   $$x_t=F_0x, \tag{6.3}$$

   where F_0 is as defined in Eq. (1.1).

2. The $m+k$ scheme: $G=F(1:m+k,1:m+k)$, that is, G is the square principal submatrix of F of size $m+k$. This amounts to a very simplistic way of picking replacement variables.

   Note that for the random matrix problem, this choice of replacement variables has no special properties and can be thought of as randomly selected. Later we will see that they provide little improvement in accuracy over the Galerkin method.

   But in the heat equation problem, these replacement variables are the next k Fourier coefficients in the expansion of the solution. Because the solution is a smooth function, and the matrix F is not too far away from diagonal, they are a very good choice of replacement variable. In fact, for this problem, they equal the performance of the original ORV method. In [1] this was presented as "Equalling the performance of a strong competitor is not too bad, especially for a new technique", but improving on this mediocre result is one reason this new version of ORV was developed.

3. The T-System of Chorin et al. [6,7] and Gottlieb [8]:

   $$x_t=F_0x+tF_1F_2x. \tag{6.4}$$

4. The original ORV scheme from [1].

5. First Order Optimal Prediction (FOOP). This is just

$$x_t = \mathrm{E}\big(F_0 x + F_1 y | x\big) = F_0 x + F_1 E(y|x) \tag{6.5}$$

with the expectation taken with respect to the distribution on the initial conditions. For both test problems x and y are independent under their initial distribution, so this reduces to the Galerkin method.

A related technique was used as a competitor in [1], in which a probability distribution for the exact solution was integrated in time and then averaged over the integration time interval. Under this distribution x and y are not usually independent. This distribution can be used to create a nontrivial FOOP-like competitor. But, in [1], it was found to work only about as well as Galerkin.

Because of this poor performance, and because the plots of system performance are already somewhat crowded, this technique is not chosen as a competitor.

Recall that $(m,n,k) =$ the number of resolved, unresolved and replacement variables, respectively. The curve labeled "Resolved Norm" is the l_2-norm of the resolved part of the exact solution. The errors shown are all l_2-norm of the resolved part of the error, and are absolute errors rather than relative errors.

Performance on the random matrix problem

Fig. 1 shows the error of the new ORV scheme on the random matrix problem, along with the competing schemes. The new ORV scheme achieves about 1 extra digit of accuracy over the original ORV scheme. The Galerkin and $m+k$ schemes both have $\mathcal{O}(1)$ error. The T-System has $\mathcal{O}(1)$ error even for small times, and is unstable. Instability is a weakness the T-System sometimes exhibits.

Performance on the heat equation

Fig. 2 shows the error of the new ORV scheme on the heat equation. In this case the original ORV scheme and the $m+k$ scheme both are about 1.5 digits more accurate than Galerkin, and the new ORV scheme is about 1.5 digits more accurate than that. The new ORV scheme is more accurate than all of the competitors presented on these two problems.

Stability of the reduced dynamics

Recall that only $G_\alpha = G - \alpha I$ is guaranteed stable, and G itself might not be stable. However given the stability of G_α, one can show fairly easily that there is a norm $\|\cdot\|_*$ such that

$$\|w(t)\|_* \le e^{\alpha t} \|w(t=0)\|_*. \tag{6.6}$$

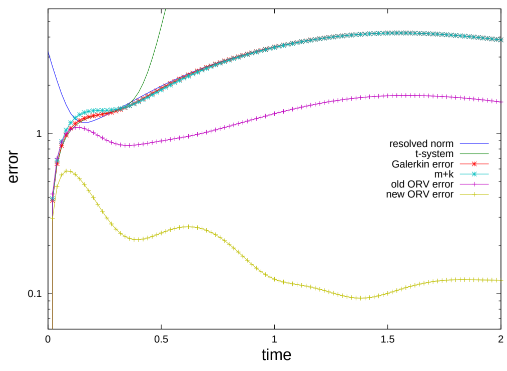

Figure 1: l_2 error of the resolved variables on log scale vs time for the Random Matrix Problem. $(m,n,k)=(10,80,12)$, $\alpha=0.5$.

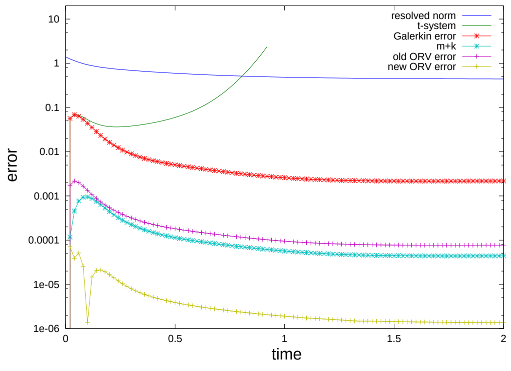

Figure 2: L_2 error of the resolved variables on log scale vs. time on the Heat Equation (6.1), $(m,n,k)=(4,24,2)$, $\alpha=0.5$, $\gamma=0.8$.

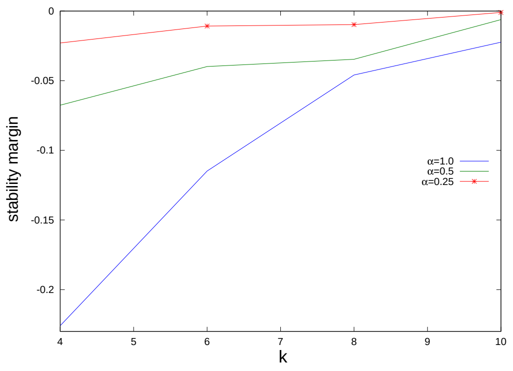

Figure 3: Stability Margin of the reduced system matrix G for different choices of k and α. Random matrix problem, $(m,n)=(4,16)$.

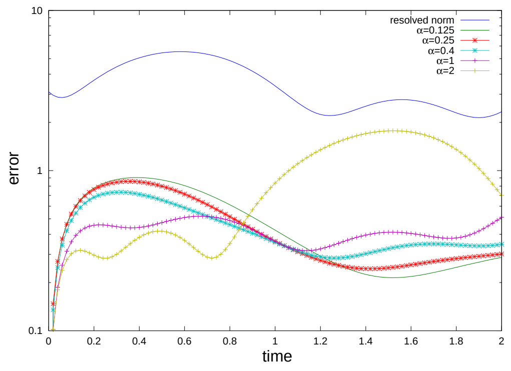

Figure 4: L_2 error of the resolved variables on log scale vs. time for different values of the time decay parameter α. Random matrix problem, $(m,n,k)=(4,20,4)$, $\alpha=0.5$.

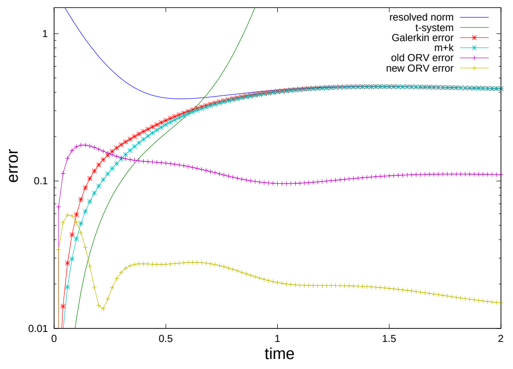

Figure 5: L_2 error of the resolved variables on log scale vs. the expected solution, for the random matrix problem, $(m,n,k)=(5,30,5)$, $\alpha=1/2$.

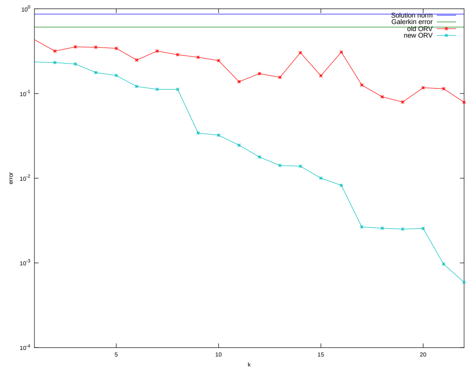

Figure 6: Error for the random matrix problem, as k, the number of replacement variables changes. $(m,n)=(5,30)$, $\alpha=1/2$.

If we take $\alpha = 1/t_{final}$ then we have for all $t \le t_{final}$

$$\|w(t)\|_* \le e\|w(t=0)\|_*, \tag{6.7}$$

which seems quite satisfactory.

Fig. 3 shows the stability margin of G, i.e., the real part of the rightmost eigenvalue of the reduced system matrix G, for several values of α and k (the number of replacement variables). We find that G is stable for all the test cases, even when $\alpha > 0$. If G were found to be unstable for some problem, it could always be stabilized by reducing α, to zero if necessary.

Note that the random matrix problem tested here is neutrally stable, and so it is a hard test for the stability of a reduced system.

Influence of the time decay parameter α

Fig. 4 shows the error of the new ORV scheme on the random matrix problem, using several different values of the time decay parameter α. Increasing α tells the scheme it should improve accuracy for t small and allow larger errors for t large. The figure shows exactly this behavior.

Error compared to the expected solution

None of these schemes approximating Eq. (1.1) are able to make use of all of the information in the initial conditions. The ORV and $m+k$ schemes use only part of the information in the unresolved initial conditions, and the Galerkin and T-System schemes do not use any unresolved initial data at all. So an arguably fairer way to test the accuracy of the different schemes is to compare them to the expected solution

$$z_{\exp}(t) = \mathrm{E}_{z_0}\big(z(t)|x(t=0)\big) = e^{Ft}\mathrm{E}(z_0|x_0). \tag{6.8}$$

In the case of S a diagonal matrix, which is the case for both test problems,

$$\hat{z}_0 = \mathrm{E}(z_0|x_0) = \begin{bmatrix} x_0 \\ 0 \end{bmatrix}. \tag{6.9}$$

The vector $\hat{z}_0$ was used as the initial conditions for both the exact solution and all of the approximation schemes.

Fig. 5 shows this version of the error for the ORV scheme and competitors. The results are somewhat similar to Fig. 1. The biggest difference is that the T-System gives good accuracy for small times, which it does not do when compared to the true solution. This is expected, since it is designed specifically to approximate the expected solution. Also the $m+k$ and Galerkin schemes also give a little bit of accuracy for small times, which they do not do when compared to the true solution.

An interesting property of the expected solution is that since it doesn't depend on y_0, the ORV approximation of it does not use the matrix R.

6.2.1 Convergence with respect to k

Fig. 6 shows the decay of the solution error for the random matrix problem, as the number of replacement variables increases. The error of the new ORV method is better than the old ORV method for all k, and the error decays more rapidly as k increases. The error displayed is integrated over time, i.e.,

$$\text{error}^2 = \int_0^{T_{final}} \sum_{i=1}^{m} \big(x_i(t) - (x_*)_i(t)\big)^2 dt. \tag{6.10}$$

Multiple local minima

The original version of ORV [1] used an error measure that sometimes had many local minima, which greatly complicates any numerical minimization process. Not so for this new version of ORV. Its error measure seems to have only a single minima.

This was investigated by computing the minimum many times, initializing the minimization software with a different random starting vector each time. In every case, it converged to the same solution.

This is a welcome result, but I cannot prove that the local minima is always unique, or even provide any motivation why might be so.

7 Discussion and conclusions

This paper builds upon the ORV method presented in [1] by introducing a new, more natural criterion for the dynamic accuracy of a reduced system, and develops the machinery necessary to optimize the new criterion. The resulting new reduced system works well, at least on the problems it has been tested against:

1. The new ORV scheme achieves better accuracy than the original version. In the case of the heat equation, it has better accuracy than both the original ORV and the $m+k$ schemes. The original version of ORV could only equal the accuracy of $m+k$.

2. It has guaranteed stability, at least in the $\alpha = 0$ case.

3. It does not require the user provide a suitable-for-all-time probability density on the phase space.

4. The associated minimization problem has a unique local minimum.

Note that the last two points make the new ORV scheme easier to use.

This new version of ORV fixes many of the issues of the original ORV, but has some possibly-limiting qualities of its own:

1. It is probably more difficult to extend this version of ORV to nonlinear problems.

2. Evaluating the error function and its gradient is more complex than for the original ORV. Not necessarily in the sense of the amount of CPU time required, but in the sense of the number of equations and amount of software required. This may make it harder to generalize or extend.

Appendix

A Initializing the random matrix problem

Computations were done using version 3.0.1 of Octave and version 9.04 of Ubuntu Linux. Octave's random number generator was seeded with a constant value to insure the same problem was always used. Essentially the following code was used to generate the random matrix problem:

```
function [F, x0] = init_randprob(nm)
        randn("seed", 42);
        F0 = randn(nm,nm);
        r = max(real(eig(F0)));
        F = F0 - r\asteye(columns(F0))
        x0 = randn(nm, 1)
endfunction
```

and the function call

```
[F, x0] = init_randprob(2);
```

should produce the output:

```
F  = 0.13721  -0.12000
     0.22221  -0.13721
x0 = 1.8949
    -1.4611
```

Versions 3.0.5 and 3.2.3 of Octave were found to generate the same numbers.

B Near-simultaneous zero crossings

Suppose $u(t)$ is a vector-valued function of time with m components and $u_*(t)$ is an approximation of it. In many cases the difference components $d_i(t) = u_i(t) - (u_*)_i(t)$ will oscillate about zero. If most or all of the components happen to cross zero at almost the same time, the error $e(t) = \|u(t) - u_*(t)\|$ will have a momentary sharp drop, which we name a near-simultaneous zero-crossing (NSZC). If $m = 1$ then this must happen repeatedly, but it is actually not unusual for it to happen (with smaller reductions in error) even if $m = 2$ or 3 or even more.

One can support this with a crude Monte-Carlo simulation.

NSZC are observed to happen in Fig. 2 near $t=0.1$ with $m=4$ and a momentary error drop of 30x and in Fig. 5 near $t=0.2$ with m=5 with a momentary drop of about 3x.

Notes:

1. NSCZ is not unique to the ORV approximation of an ODE. The author has observed it to happen with the $m+k$ approximation and the old ORV method as well.
2. If $m=4$, then large drops in error are unlikely, but if most of the components are tiny, and (say) 2 are larger, then effectively $m=2$. This is the case with the heat equation, and a 30x drop with $m=2$ is not surprising.
3. A large momentary drop in error with $m=5$ is unlikely, but the error drop in Fig. 5 is only 3x, and that size of error drop can happen with $m=5$.

Acknowledgments

The author acknowledges the support of this work under RGC grant HKBU 200910, and appreciates the hospitality of the Department of Mathematics at Hong Kong Baptist University during his visits. He also appreciates the hospitality and useful discussions of his host, Professor Wai Sun Don.

References

[1] A. Solomonoff and Wai Sun Don, Reduction of linear systems of ODES with optimal replacement variables, Commun. Comput. Phys., 9 (2011), 756–779.

[2] R. H. Bartels and G. W. Stewart, Solution of the matrix equation $AX+XB=C$, Commun. ACM, 15(9) (1972), 820–826.

[3] Steven G. Johnson, The NLopt nonlinear-optimization package, `ab-initio.mit.edu/nlopt`.

[4] John W. Eaton, GNU Octave Manual, Network Theory Limited, 2002, ISBN=0-9541617-2-6, also `www.octave.org`.

[5] A. Chorin, O. H. Hald and R. Kupferman, Optimal prediction with memory, Phys. D, 166 (2002), 239–257.

[6] A. Chorin and P. Stinis, Problem reduction, renormalization and memory, Commun. Appl. Math. Comput. Sci., 1(1) (2005), 1–27.

[7] A. J. Chorin and O. H. Hald, Stochastic Tools in Mathematics and Science, Springer, 2006

[8] A. Chertock, D. Gottlieb and A. Solomonoff, Modified optimal prediction and its application to a particle-method problem, J. Sci. Comput., 37 (2008), 189–201.

[9] D. Givon, R. Kupferman and A. Stuart, Extracting macroscopic dynamics: model problems and algorithms, Nonlinearity, 17(6) (2004), R55–R127, MR 2097022.

[10] H. Kwakernaak and R. Sivan, Linear Optimal Control Systems, Wiley-Interscience, 1972.

[11] R. A. Horn and C. R. Johnson, Topics in Matrix Analysis, Cambridge University Press, 1994.

[12] Wikipedia contributors, Trace (linear algebra), Wikipedia, the free encyclopedia.

[13] L. Sirovich, Analysis of turbulent flows by means of the empirical eigenfunctions, Fluid Dyn. Res., 8 (1991), 85–100.

[14] J. Brandts, A comparison of subspace methods for sylvester equations, preprint No. 1183, Universiteit Utrecht Mathematics Institute, 2001.

[15] P. Benner, E. S. Quintana-Orti, G. Quintana-Orti, Efficient numerical algorithms for balanced stochastic truncation, 2001, see also `http://matwbn.icm.edu.pl/ksiazki/amc/amc11/amc1156.pdf`.

[16] A. Antoulas, D. Sorensen, K. A. Gallivan, P. Van Dooren, A. Grama, C. Hoffman and A. Sameh, Model Reduction of Large-Scale Dynamical Systems, Computational Science-ICCS 2004, 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part III, M. Bubak, G. Dick van Albada, P. M. A. Sloot and J. Dongarra, eds., Lecture Notes in Computer Science, 3038 (2004), 740–747.