Solving Orthogonal Matrix Differential Systems in *Mathematica*

Mark Sofroniou[1] and Giulia Spaletta[2]

[1] Wolfram Research, Champaign, Illinois, USA. marks@wolfram.com
[2] Mathematics Department, Bologna University, Italy. giulia@cs.unibo.it

Abstract. A component of a new environment for the numerical solution of ordinary differential equations in *Mathematica* is outlined. We briefly describe how special purpose integration methods can be constructed to solve structured dynamical systems. In particular we focus on the solution of orthogonal matrix differential systems using projection. Examples are given to illustrate the advantages of a projection scheme over conventional integration methods.

Keywords. Geometric numerical integration; orthogonal projection; ordinary differential equations; computer algebra systems.
AMS. 65L05; 65L06; 68Q40.

1 Features of the framework

The *Mathematica* function **NDSolve** can be used to find numerical solutions of a wide range of ordinary as well as some partial differential equations. A drawback with the current function is that the design and implementation are in the form of a 'black-box' and there is only a single one-step numerical method available, an outdated explicit Runge-Kutta pair of Fehlberg. Since the function is not modular, it cannot be used to take advantage of new research developments.

One of the aims of an ongoing overhaul is to make a number of differential equations solvers available in a uniform, integrated and extensible environment. Many one-step integrators are being developed: explicit, implicit, linearly implicit Euler and Midpoint; embedded explicit Runge Kutta pairs of various orders; Gauss, Lobatto (IIIA, IIIB, IIIC), Radau (IA, IIA) implicit Runge Kutta methods; extrapolation methods.

In recent years there has been a considerable growth of interest in studying and numerically preserving a variety of dynamical properties, leading to so called geometric integrators (see for example [9], [10], [17], [18], [21]). The new **NDSolve** allows built-in methods to be exploited as building blocks for the efficient construction of special purpose (compound) integrators. The framework is also hierarchical, meaning that one method can call another at each step of an integration. These features facilitate the construction of geometric integrators and the implementation of one specific method is given here as demonstration.

This paper is organized as follows. Section 2 describes the class of problems of interest and various strategies for their numerical solution. Amongst the possible

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2331, pp. 496–505, 2002.

choices, an iterative method is selected and an algorithm for the implementation is discussed together with appropriate stopping criteria. Section 3 describes the implementation of a projected integration method, **OrthogonalProjection**, written in top-level *Mathematica* code. Examples of improved qualitative behavior over conventional integrators are given by considering the solution of square and rectangular orthogonal matrix differential systems in Section 4 and Section 5. Some issues relating to potential extensions are given in Section 6 together with a motivating example.

2 Orthogonal projection

Consider the matrix differential equation:

$$y'(t) = f(t, y(t)), \quad t > 0\,, \tag{1}$$

where the initial value $y_0 = y(0) \in \mathbb{R}^{m\times p}$ is given and satisfies $y_0^T\, y_0 = I$, where I is the $p \times p$ identity matrix. Assume that the solution preserves orthonormality, $y^T\, y = I$, and that it has full rank $p < m$ for all $t \geq 0$.

From a numerical perspective, a key issue is how to integrate (1) in such a way that the approximate solution remains orthogonal. Several strategies are possible. One approach, presented in [4], is to use an implicit Runge-Kutta method such as the Gauss scheme. These methods, however, are computationally expensive and furthermore there are some problem classes for which no standard discretization scheme can preserve orthonormality [5]. Some alternative strategies are described in [3] and [6]. The approach that will be taken up here is to use any reasonable numerical integrator and then post-process using a projective procedure at the end of each integration step. It is also possible to project the solution at the end of the integration instead of at each step, although the observed end point global errors are often larger [13].

Given a matrix, a nearby orthogonal matrix can be found via a direct algorithm such as QR decomposition or singular value decomposition (see for example [4], [13]). The following definitions are useful for the direct construction of orthonormal matrices [8].

Definition 1 (Thin Singular Value Decomposition (SVD)). *Given a matrix $A \in \mathbb{R}^{m\times p}$ with $m \geq p$, there exist two matrices $U \in \mathbb{R}^{m\times p}$ and $V \in \mathbb{R}^{p\times p}$ such that $U^T A V$ is the diagonal matrix of singular values of A, $\Sigma = diag(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{p\times p}$, where $\sigma_1 \geq \cdots \geq \sigma_p \geq 0$. U has orthonormal columns and V is orthogonal.*

Definition 2 (Polar Decomposition). *Given a matrix A and its singular value decomposition $U\,\Sigma\,V^T$, the polar decomposition of A is given by the product of two matrices Z and P where $Z = U\,V^T$ and $P = V\,\Sigma\,V^T$. Z has orthonormal columns and P is symmetric positive semidefinite.*

If A has full rank then its polar decomposition is unique. The *orthonormal polar factor* Z of A is the matrix that, for any unitary norm, solves the minimization problem [16]:

$$\|A - Z\| = \min_{Y \in \mathbb{R}^{m \times p}} \{\|A - Y\| \ : \ Y^T Y = I\}. \tag{2}$$

QR decomposition is cheaper than SVD, roughly by a factor of two, but it does not provide the best orthonormal approximation.

Locally quadratically convergent iterative methods for computing the orthonormal polar factor also exist, such as Newton or Schulz iteration [13]. For a projected numerical integrator, the number of iterations required to accurately approximate (2) varies depending on the local error tolerances used in the integration. For many differential equations solved in IEEE double precision, however, one or two iterations are often sufficient to obtain convergence to the orthonormal polar factor. This means that Newton or Schulz methods can be competitive with QR or SVD [13]. Iterative methods also have an advantage in that they can produce smaller errors than direct methods (see Figure 2 for example).

The application of Newton's method to the matrix function $A^T A - I$ leads to the following iteration for computing the orthonormal polar factor of $A \in \mathbb{R}^{m \times m}$:

$$Y_{i+1} = (Y_i + Y_i^{-T})/2, \quad Y_0 = A.$$

For an $m \times p$ matrix with $m > p$ the process needs to be preceded by QR decomposition, which is expensive. A more attractive scheme, that works for any $m \times p$ matrix A with $m \geq p$, is the Schulz iteration [15]:

$$Y_{i+1} = Y_i + Y_i\,(I - Y_i^T\,Y_i)/2, \quad Y_0 = A. \tag{3}$$

The Schulz iteration has an arithmetic operation count per iteration of $2\,m^2\,p + 2\,m\,p^2$ floating point operations, but is rich in matrix multiplication [13]. In a practical implementation, gemm level 3 BLAS of LAPACK [19] can be used in conjunction with architecture specific optimizations via the Automatically Tuned Linear Algebra Software (ATLAS) [22]. Such considerations mean that the arithmetic operation count of the Schulz iteration is not necessarily an accurate reflection of the observed computational cost.

A useful bound on the departure from orthonormality of A in (2) is [14]:

$$\|A^T A - I\|_F\,. \tag{4}$$

By comparing (4) and the term in parentheses in (3), a simple stopping criterion for the Schulz iteration is $\|A^T A - I\|_F \leq \tau$ for some tolerance τ.

Assume that an initial value y_n for the current solution is given, together with a solution $y_{n+1} = y_n + \Delta y_n$ from a one-step numerical integration method. Assume that an absolute tolerance τ for controlling the Schulz iteration is also prescribed. The following algorithm can be used for implementation.

Algorithm 1 (Standard formulation)

1. *Set* $Y_0 = y_{n+1}$ *and* $i = 0$.
2. *Compute* $E = I - Y_i^T Y_i$
3. *Compute* $Y_{i+1} = Y_i + Y_i E/2$.
4. *If* $\|E\|_F \leq \tau$ *or* $i = imax$ *then return* Y_{i+1}.
5. *Set* $i = i + 1$ *and go to step 2.*

NDSolve uses compensated summation to reduce the effect of rounding errors made in repeatedly adding the contribution of small quantities Δy_n to y_n at each integration step [16]. Therefore the increment Δy_n is returned by the base integrator. An appropriate orthogonal correction ΔY_i for the projective iteration can be determined using the following algorithm.

Algorithm 2 (Increment formulation)

1. *Set* $\Delta Y_0 = 0$ *and* $i = 0$.
2. *Set* $Y_i = \Delta Y_i + y_{n+1}$
3. *Compute* $E = I - Y_i^T Y_i$
4. *Compute* $\Delta Y_{i+1} = \Delta Y_i + Y_i E/2$.
5. *If* $\|E\|_F \leq \tau$ *or* $i = imax$ *then return* $\Delta Y_{i+1} + \Delta y_n$.
6. *Set* $i = i + 1$ *and go to step 2.*

This modified algorithm is used in **OrthogonalProjection** and shows an advantage of using an iterative process over a direct process, since it is not obvious how an orthogonal correction can be derived for direct methods.

3 Implementation

The projected orthogonal integrator **OrthogonalProjection** has three basic components, each of which is a separate routine:

- initialize the basic numerical method to use in the integration;
- invoke the base integration method at each step;
- perform an orthogonal projection.

Initialization of the base integrator involves constructing its 'state'. Each method in the new **NDSolve** framework has its own data object which encapsulates information that is needed for the invocation of the method. This includes, but is not limited to, method coefficients, workspaces, step size control parameters, step size acceptance/rejection information, Jacobian matrices. The initialization phase is performed once, before any actual integration is carried out, and the resulting data object is validated for efficiency so that it does not need to be checked at each integration step.

Options can be used to modify the stopping criteria for the Schulz iteration. One option provided by our code is **IterationSafetyFactor** which allows control over the tolerance τ of the iteration. The factor is combined with a Unit in the

Last Place, determined according to the working precision used in the integration (ULP $\approx 2.22045\,10^{-16}$ for IEEE double precision). The Frobenius norm used for the stopping criterion can be efficiently computed via the LAPACK LANGE functions [19]. An option **MaxIterations** controls the maximum number of iterations $imax$ that should be carried out.

The integration and projection phase are performed sequentially at each time step. During the projection phase various checks are performed, such as confirming that the basic integration proceeded correctly (for example a step rejection did not occur). After each projection, control returns to a central time stepping routine which is a new component of **NDSolve**. The central routine advances the solution and reinvokes the integration method.

An important feature of our implementation is that the basic integrator can be any built-in numerical method, or even a user-defined procedure. An explicit Runge-Kutta pair is often used as the basic time stepping integrator but if higher local accuracy is required an extrapolation method could be selected by simply specifying an appropriate option.

All numerical experiments in the sequel have been carried out using the default options of **NDSolve**. The appropriate initial step size and method order are selected automatically by the code (see for example [1], [7] and [9]). The step size may vary throughout the integration interval in order to satisfy local absolute and relative error tolerances. Order and tolerances can also be specified using options. With the default settings the examples of Section 4 and Section 5 require exactly two Schulz iterations per integration step.

4 Square systems

Consider the orthogonal group $O_m(\mathbb{R}) = \{Y \in \mathbb{R}^{m\times m} \,:\, Y^T Y = I\}$. The following example involves the solution of a matrix differential system on $O_3(\mathbb{R})$ [23].

$$\begin{aligned} Y' &= F(Y)\,Y \\ &= \big(A + \big(I - Y\,Y^T\big)\big)\,Y \end{aligned} \quad \text{where} \quad A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}. \tag{5}$$

The matrix A is skew-symmetric. Setting $Y(0) = I$, the solution evolves as $Y(t) = \exp[t\,A]$ and has eigenvalues:

$$\lambda_1 = 1,\ \lambda_2 = \exp\big(t\,i\,\sqrt{3}\big),\ \lambda_3 = \exp\big(-t\,i\,\sqrt{3}\big).$$

As t approaches $\pi/\sqrt{3}$ two of the eigenvalues of $Y(t)$ approach -1. The interval of integration is $[0, 2]$.

The solution is first computed using an explicit Runge-Kutta method. Figure 1 shows the orthogonal error (4) at grid points in the numerical integration. The error is of the order of the local accuracy of the numerical method.

The orthogonal error in the solution computed using **OrthogonalProjection**, with the same explicit Runge-Kutta method as the base integration scheme,

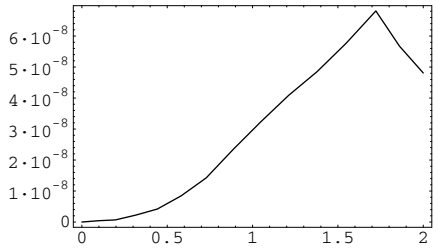

Fig. 1. Orthogonal error $\|Y^T Y - I\|_F$ vs time for an explicit Runge Kutta method applied to (5).

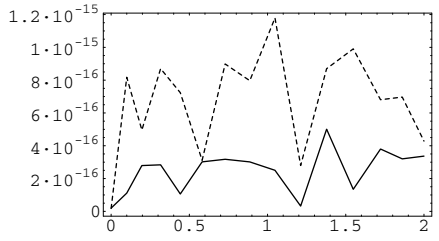

Fig. 2. Orthogonal error $\|Y^T Y - I\|_F$ vs time for projected orthogonal integrators applied to (5). The dashed line corresponds to forming the polar factor directly via SVD. The solid line corresponds to the Schulz iteration in **OrthogonalProjection**.

is illustrated in Figure 2. The errors in the orthonormal polar factor formed directly from the SVD is also given. The initial step size and method order are the same as in Figure 1, but the step size sequences in the integration are different. The orthogonal errors in the direct decomposition are larger than those of the iterative method, which are reduced to approximately the level of roundoff in IEEE double precision arithmetic.

5 Rectangular systems

OrthogonalProjection also works for rectangular matrix differential systems. Formally stated, we are interested in solving ordinary differential equations on the Stiefel manifold $V_{m,p}(\mathbb{R}) = \{Y \in \mathbb{R}^{m\times p} \ : \ Y^T Y = I\}$ of matrices of dimension $m \times p$, with $1 \le p < m$. Solutions that evolve on the Stiefel manifold find numerous applications such as eigenvalue problems in numerical linear algebra, computation of Lyapunov exponents for dynamical systems and signal processing. Consider an example adapted from [3]:

$$q'(t) = A\,q(t), \quad t > 0, \quad q(0) = \frac{1}{\sqrt{m}}\,[1, \ldots, 1]^T\,, \tag{6}$$

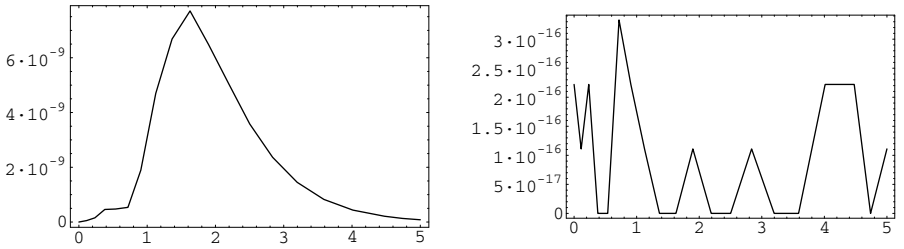

Fig. 3. Orthogonal error $\|Y^T Y - I\|_F$ vs time for (6) using **ExplicitRungeKutta** (left) and **OrthogonalProjection** (right).

where $A = diag(a_1, \ldots, a_m) \in \mathbb{R}^{m\times m}$ with $a_i = (-1)^i \alpha,\ \alpha > 0$. The normalized exact solution is given by:

$$Y(t) \ = \ \frac{q(t)}{\|q(t)\|} \in \mathbb{R}^{m\times 1}, \quad q(t) \ = \ \frac{1}{\sqrt{m}} \left[\exp(a_1 t), \ldots, \exp(a_m t)\right]^T .$$

$Y(t)$ therefore satisfies the following weak skew-symmetric system on $V_{m,1}(\mathbb{R})$:

$$\begin{aligned} Y' = & \quad F(Y)\, Y \\ = & \left(I - Y\, Y^T\right) A\, Y \end{aligned}$$

The system is solved on the interval $[0, 5]$ with $\alpha = 9/10$ and dimension $m = 2$.

The orthogonal error in the solution has been computed using an explicit Runge-Kutta pair and using **OrthogonalProjection** with the same explicit Runge-Kutta pair for the basic integrator. Figure 3 gives the orthogonal error at points sampled during the numerical integration. For **ExplicitRungeKutta** the error is of the order of the local accuracy. Using **OrthogonalProjection** the deviation from the Stiefel manifold is reduced to the level of roundoff.

Since the exact solution in known, it is possible to compute the componentwise absolute global error at the end of the integration interval. The results are displayed in Table 1.

Method	Errors
ExplicitRungeKutta	$(-2.38973\,10^{-9},\ 4.14548\,10^{-11})$
OrthogonalProjection	$(-2.38974\,10^{-9},\ 2.94986\,10^{-13})$

Table 1. Absolute global integration errors for (6).

6 Future work

OrthogonalProjection indicates how it is possible to extend the developmental **NDSolve** environment to add new numerical integrators. The method works by numerically solving a differential system and post-processing the solution at each step via an orthogonal projective procedure.

In some systems there may be constraints that are not equivalent to the conservation of orthogonality. An example is provided by Euler's equations for rigid body motion (see [12] and [20]):

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{pmatrix} = \begin{pmatrix} 0 & \frac{y_3}{I_3} & -\frac{y_2}{I_2} \\ -\frac{y_3}{I_3} & 0 & \frac{y_1}{I_1} \\ \frac{y_2}{I_2} & -\frac{y_1}{I_1} & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} . \tag{7}$$

Two quadratic first integrals of the system are:

$$I(y) = y_1^2 + y_2^2 + y_3^2, \tag{8}$$

and

$$H(y) = \frac{1}{2} \left(\frac{y_1^2}{I_1} + \frac{y_2^2}{I_2} + \frac{y_3^2}{I_3} \right) . \tag{9}$$

Constraint (8) is conserved by orthogonality and has the effect of confining the motion from $\mathbb{R}^3$ to a sphere. Constraint (9) represents the kinetic energy of the system and, in conjunction with (8), confines the motion to ellipsoids on the sphere. Certain numerical methods, such as the implicit midpoint rule or 1-stage Gauss implicit Runge-Kutta scheme, preserve quadratic invariants exactly (see for example [2]). Figure 4 shows three solutions of (7) computed on the interval $[0, 32]$ with constant step size $1/10$, using the initial data:

$$I_1 = 2,\ I_2 = 1,\ I_3 = \frac{2}{3},\ y_1(0) = \cos(\frac{11}{10}),\ y_2(0) = 0,\ y_3(0) = \sin(\frac{11}{10}) .$$

For the explicit Euler method solutions do not lie on the unit sphere. **OrthogonalProjection**, with the explicit Euler method as the base integrator, preserves orthogonality but not the quadratic invariant (9), so that trajectories evolve on the sphere but are not closed. The implicit midpoint method conserves both (8) and (9) so that solutions evolve as ellipsoids on the sphere.

Runge-Kutta methods cannot conserve all polynomial invariants that are neither linear or quadratic [12, Theorem 3.3]. In such cases, however, the local solution from any one-step numerical scheme can be post-processed using a generalized projection based on Newton iteration (see for example [12, Section IV.4] and [10, Section VII.2]). In order to address these issues, a multiple constraint method, **Projection**, is currently under development. If the differential system is ρ-reversible in time then a *symmetric* projection process has also been shown to be beneficial [11].

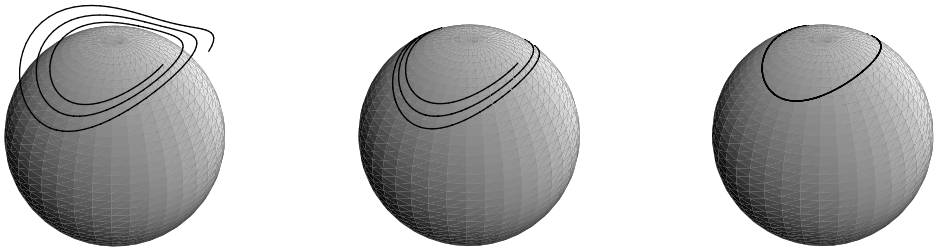

Fig. 4. Solutions of (7) using the explicit Euler method (left), **OrthogonalProjection** (center) and the implicit midpoint method (right).

Acknowledgements

The authors are grateful to Robert Knapp for his work on many aspects of **NDSolve** and to Ernst Hairer for his lectures on geometric integration and for pointing out the system (7).

References

1. Butcher, J. C.: Order, stepsize and stiffness switching. Computing. **44** (1990) 209–220.
2. Cooper, G. J.: Stability of Runge-Kutta methods for trajectory problems. IMA J. Numer. Anal. **7** (1987) 1–13.
3. Del Buono, N., Lopez, L.: Runge-Kutta type methods based on geodesics for systems of ODEs on the Stiefel manifold. BIT. **41** (5) (2001) 912–923.
4. Dieci, L., Russel, R. D., Van Vleck, E. S.: Unitary integrators and applications to continuous orthonormalization techniques. SIAM J. Num. Anal. **31** (1994) 261–281.
5. Dieci, L., Van Vleck, E. S.: Computation of a few Lyapunov exponents for continuous and discrete dynamical systems. Appl. Numer. Math. **17** (3) (1995) 275–291.
6. Dieci, L., Van Vleck, E. S.: Computation of orthonormal factors for fundamental solution matrices. Numer. Math. **83** (1999) 591–620.
7. Gladwell, I., Shampine, L. F., Brankin, R. W.: Automatic selection of the initial step size for an ODE solver. J. Comp. Appl. Math. **18** (1987) 175–192.
8. Golub, G. H., Van Loan, C. F.: Matrix computations. 3rd edn. Johns Hopkins University Press, Baltimore (1996).
9. Hairer, E., Nørsett, S. P., Wanner, G.: Solving ordinary differential equations I: nonstiff problems. 2nd edn. Springer-Verlag, New York (1993).
10. Hairer, E., Wanner, G.: Solving ordinary differential equations II: stiff and differential algebraic problems. 2nd edn. Springer-Verlag, New York (1996).
11. Hairer, E.: Symmetric projection methods for differential equations on manifolds. BIT. **40** (4) (2000) 726–734.

12. Hairer, E., Lubich, C., Wanner, G.: Geometric numerical integration: structure preserving algorithms for ordinary differential equations. Springer-Verlag, New York, draft version June 25 (2001).
13. Higham, D.: Time-stepping and preserving orthonormality. BIT. **37** (1) (1997) 241–36.
14. Higham, N. J.: Matrix nearness problems and applications. In: Gover, M. J. C., Barnett, S. (eds.): Applications of Matrix Theory. Oxford University Press, Oxford (1989) 1–27.
15. Higham, N. J., Schreiber, R. S.: Fast polar decomposition of an arbitrary matrix. SIAM J. Sci. Stat. Comput. **11** (4) (1990) 648–655.
16. Higham, N. J.: Accuracy and stability of numerical algorithms. SIAM, Philadelphia (1996).
17. Iserles, A., Munthe-Kaas, H. Z., Nørsett, S. P., Zanna, A.: Lie-group methods. Acta Numerica. **9** (2000) 215–365.
18. McLachlan, R. I., Quispel, G. R. W.: Six lectures on the geometric integration of ODEs. In: DeVore, R. A., Iserles, A., Süli, E. (eds.): Foundations of Computational Mathematics. Cambridge University Press. Cambridge. (2001) 155–210.
19. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorenson, D.: LAPACK Users' Guide. 3rd edn. SIAM, Philadelphia (1999).
20. Marsden, J. E., Ratiu, T.: Introduction to mechanics and symmetry. Texts in Applied Mathematics, Vol. 17. 2nd edn. Springer-Verlag, New York (1999).
21. Sanz-Serna, J. M., Calvo, M. P.: Numerical Hamiltonian problems. Chapman and Hall, London (1994).
22. Whaley, R. C., Petitet, A., Dongarra, J. J.: Automatated empirical optimization of software and the ATLAS project. available electronically from http://math-atlas.sourceforge.net/
23. Zanna, A.: On the numerical solution of isospectral flows. Ph. D. Thesis, DAMTP, Cambridge University (1998).