Draft version of February 1, 2008

ORTHONORMAL INTEGRATORS BASED ON HOUSEHOLDER AND GIVENS TRANSFORMATIONS*

LUCA DIECI† AND ERIK S. VAN VLECK‡

Abstract. We carry further our work [DV2] on orthonormal integrators based on Householder and Givens transformations. We propose new algorithms and pay particular attention to appropriate implementation of these techniques. We also present a suite of `Fortran` codes and provide numerical testing to show the efficiency and accuracy of our techniques.

AMS(MOS) subject classifications. Primary 65L

Key words. Continuous Householder and Givens transformations, orthonormal integrators.

1. Introduction. In recent times, there has been a lot of interest in techniques for solving differential equations whose exact solution is an orthonormal matrix, in such a way that the computed solution is also orthonormal; [CIZ], [DRV1], [DV1], [DV2], [DLP], [GSO], [H], [MR], are a representative sample of references. In this paper, we will restrict attention to a particular class of differential equations with orthonormal solution, the *QR equations*. These arise in many seemingly unrelated applications (e.g., see [BGGS], [C], [DNT], [DV2], [Me]), but can all be tracked back to the following setup.

1. We are given the function $A: \ t \to \mathbb{R}^{n\times n}$, $A \in C^{k-1}$, $k \geq 1$, and consider the associated initial value problem for $X \in \mathbb{R}^{n\times p}$, $p \leq n$:

$$\dot{X} \ = \ A(t)X \, , \ t \in [t_0, t_f] \ , \ X(t_0) = X_0 \ \text{full rank} \, .$$

2. We need the QR factorization of the matrix X: $X = QR$, $Q, R \in C^k$, $Q \in \mathbb{R}^{n\times p}$, $R \in \mathbb{R}^{p\times p}$. For stability reasons, we want to find the factors Q and R without first finding X. Once Q is known, the equation for R becomes:

$$\dot{R} \ = \ \tilde{A}R \, , \ R(t_0) = R_0 \, , \ \ \tilde{A}(t) = (Q^T A Q - Q^T \dot{Q})R \, , \tag{1.1}$$

where –since $X = QR$– $\tilde{A}$ of (1.1) is upper triangular. Computation of R reduces to a backward substitution algorithm for the entries of R, coupled with quadratures, which is *conceptually* simple. Therefore, in this work we will focus on finding Q.

3. Let $Q_0 R_0 = X_0$ be a QR factorization of X_0. Differentiate the relation $X = QR$, make use of triangularity of R, let $S = Q^T \dot{Q}$, observe that $S \in \mathbb{R}^{p\times p}$ must be skew-symmetric, and obtain the following differential equation for Q:

$$\dot{Q} = AQ - QQ^T AQ + QS \, , \ \ S_{ij} = \begin{cases} (Q^T AQ)_{ij}, & i > j \\ 0, & i = j \\ -(Q^T AQ)_{ji}, & i < j \end{cases} \ , \ Q(t_0) = Q_0 \, . \tag{1.2}$$

*This work was supported in part under NSF Grants #DMS-9973266 and #DMS-9973393.

†School of Mathematics and CDSNS, Georgia Institute of Technology, Atlanta, GA 30332-0160 U.S.A..

‡ Dept. of Mathematical & Computer Sciences, Colorado School of Mines, Golden, CO 80401 U.S.A..

Remark 1.1. Different rewritings of (1.2) are possible, in the case of $p < n$, which present distinct advantages with respect to the given form (1.2). In particular, the following rewriting proposed by Bridges and Reich in [BR] is interesting:

$$\dot{Q} = [(A - QQ^T A) - (A^T - A^T QQ^T) + QSQ^T]Q\,. \tag{1.3}$$

Remark 1.2. Triangularity of $\tilde{A}$ is the key point of a successful and widely used technique for computing Lyapunov exponents of the A-system in decreasing order, i.e., the first p dominant ones; e.g., see [BGGS] and [DRV2]. It is worth pointing out that one may not know a priori how many Lyapunov exponents are needed, i.e., the value of p; for example, one may need to compute all the positive ones[1]. In this case, it would be desirable to have techniques which are capable (at least, in principle) of increasing the number of exponents to be calculated without having to restart the entire computation from scratch, and that can profitably use the computations done so far.

In the next section, we discuss the design choices we face when devising methods to solve (1.2). Then, we revisit the setup we proposed in [DV2] where we integrate (1.2) by seeking Q as product of elementary Householder or Givens transformations. In Section 3 we discuss numerical issues which must be confronted when implementing techniques based on these elementary transformations. We also put forward new formulations in the Householder case. In Section 4, we present `FORTRAN` codes we have written for the task, and we illustrate their performance on a number of examples. Conclusions are in Section 5.

We must point out right away that in this work we are exclusively concerned with the case of coefficient matrices $A(\cdot)$ which are **dense**; that is, they have no particularly exploitable structure. In case A is structured (symmetric, banded, etc.) some computational savings ought to be possible, and we will consider these cases in future work.

2. Background and Householder and Givens representations. Integrating (1.2) is an example of integration on a manifold. Quite clearly, we can view the solution Q as a curve on the compact manifold of orthonormal (orthogonal if $p = n$) matrices. The dimension of this manifold is $p\frac{2n-p-1}{2}$, which is therefore the number of degrees of freedom one has to resolve. In general, a smooth manifold may be parametrized in many different ways, and the choice of parametrization may turn out to be of utmost importance from the numerical point of view. Clearly, a minimal parametrization requires $p\frac{2n-p-1}{2}$ parameters. We notice that a typical construction in differential geometry is to parametrize a smooth manifold by overlapping local charts; this may be a convenient way to seek the solution of a differential equation evolving on a manifold even when the manifold itself may happen to be globally parametrizable. Indeed, from the numerical point of view, it is important that the parametrization lead to a stable numerical procedure. For example, it is not enough to know that the solution of a differential equation on a manifold has a global parametrization to make adopting this parametrization numerically legitimate. In particular, if we choose a representation for the solution Q of (1.2), even when it happens to be globally defined, it may not give us a sound numerical procedure.

Since the solution of (1.2) exists for all times, one may think that direct integration of the IVP for Q is the right way to proceed. However, a standard numerical method

[1] e.g., this is needed in the study of ergodic dynamical systems to approximate the so-called Lyapunov dimension and physical measure, see [ER]

used on (1.2) will not deliver a solution which is orthonormal at the meshpoints. For this reason, several techniques have been proposed to obtain an orthonormal numerical solution.

[$p = n$] In this so-called "square case", there are many competing approaches. In our opinion, the ones below are the most natural and/or interesting. All of these choices have been implemented to varying degree of sophistication, and all present intrinsic implementation difficulties: without entering in the specifics of these methods, it is enough to state that none of the methods below is trivial to implement, with the exception of (ii), and that all them can be implemented so to have an expense proportional to $O(n^3)$ per step, which is the expense of evaluating the right hand side of (1.2)[2].

(i) Runge-Kutta schemes at Gauss points used directly on (1.2); [DRV1].

(ii) Projected methods. There are at least two of these: (a) *bad* and (b) *good*.

(a) These methods use the original differential equation for X, integrate this, and then form its QR factorization at each step. They are numerically unsound, since integration of X is quite often an unstable procedure (as when the problem is exponentially dichotomic).

(b) These consist in using any scheme to integrate (1.2), and then projecting the solution onto the manifold of the orthonormal matrices. For example, the projection step can be done with a modified Gram-Schmidt procedure (as in [DRV1]) or by using the orthogonal polar factor (as in [H]).

(iii) Transformation methods. Here, one transforms the equation for Q, which is an element of the (Lie) group of the orthogonal matrices, into an equation for a skew-symmetric matrix Y, an element of the underlying (Lie) algebra. The equation for Y now evolves in a linear space, and standard discretizations will deliver a skew-symmetric approximation: transforming this back to the group gives the desired approximation for Q. There have been at least two ways in which this design has been carried out:

(a) using near the identity transformations and the matrix exponential to get back on the group (e.g., see [Mu]), and

(b) using the Cayley transform to get to the algebra and then back to the group (e.g., see [DLP]).

[$p < n$] Here, the solution belongs to the manifold of rectangular orthonormal matrices, the so-called *Stiefel manifold*. We must appreciate that often one has $p \ll n$, and thus we must insist that a well conceived method should have cost proportional to $O(n^2 p)$ per step, that is the cost of evaluating the right hand side of (1.2). As it turns out, and to the best of our understanding, this restriction rules out most of the methods we listed above for the $p = n$ case, with the exception of projection methods. However, the reasons why the other methods must be discarded are different from one another: Gauss RK methods applied directly to (1.2) do not maintain orthogonality in the case $p < n$ (see [DV1]), and transformation methods require going from the group structure to the algebra and back: apparently, this requires an $O(n^3)$ expense at some level. Notice that Gauss RK schemes on (1.3) do maintain

[2] we ignore the expense of computing $A(t)$, since this is problem dependent, and we only focus on the linear algebra expense; further, recall that we only consider the case of A "dense"

orthogonality, but it is not clear to us that they can be implemented so to require a $O(n^2p)$ expense per step without forcing a possibly severe stepsize restriction; for this reason in [BR] the authors adopt a stabilization procedure which allows for explicit schemes to be used on (1.3); in essence, their method becomes akin to inexact projection methods.

Based upon the above discussion, and with our present knowledge and understanding, it would seem that projection methods are the only survivors amongst the methods recalled above. We will see below that the techniques put forward in [DV2] and revisited in this paper are a better alternative, but before doing so, let us point out some inherent characteristics and potential limitations of projection methods.

(i) Perhaps the major obstacle to use the (*good*) projection methods is of theoretical nature: projection methods are hard to analyze. Of course, if any method of order s, say, is used for (1.2), then the projected solution is also of order s. However, the projection step itself is a discontinuous operator, and this has been cause for some worries.

(ii) Part of the worries are probably caused by the difficulty of getting a backward error statement for projection methods. In the numerical analysis of differential equations, a backward error analysis consists in the realization that one has solved (at any given order) a problem which is $O(h^s)$ close to the original one. We refer to [HL] for this point of view, which has proven valuable for many problems, and most notably for Hamiltonian systems. However, this point of view is perhaps less relevant in the context under examination here. After all, suppose we obtain some orthonormal $\tilde{Q}$ at the grid points t_k instead of the exact Q, and some $\tilde{R}$, instead of the exact R. Naturally, we can assume that both $\tilde{Q}$ and $\tilde{R}$ are $O(h^s)$ approximation to the exact matrices. Therefore, we have triangularized some $\tilde{Y}(t_k)$ instead of $Y(t_k)$, and obviously $\tilde{Y}(t_k)$ is $O(h^s)$ close to $Y(t_k)$. In general, should we expect something better?

(iii) In adaptive integration mode, there are some undesirable features of projection methods. For the sake of clarity, suppose that the time stepping strategy is based on two formulas of different orders. If we control the error on the unprojected solution, then it may occasionally happen that we will reject a step which would not have been rejected if we had checked the error on the projected values (or, similarly, we may accept a step which would have not been accepted for the projected values). On the other hand, if we control the error on the projected values, then we essentially increase the work, because now two projections have to be performed. Finally, and regardless of how we control the error, it is undesirable that when we end up rejecting a step we had to approximate all of Q in the first place. It would have been preferable to realize that the step was not going to be successful ahead of computing all of Q. Admittedly, this may sound as a strange request, but we'll see below that it is possible to achieve it with well designed methods.

(iv) To conclude, and in spite of the above observations, we must say that our experience with projection methods has been quite positive. For this reason, in Section 4, we will compare performance of our new codes with a projection method.

We are now ready to list which properties –in our opinion– a method to solve (1.2) should have. At this point, the word "method" must be read as: "formulation of the task in a form which in principle allows for the properties below to be satisfied". In so doing, we maintain the freedom of delaying discussion of which formulas will be

used in practice.

1. The method must deliver an orthonormal approximation, at the very least at the mesh points. If a RK type integrator is adopted, the method must deliver (at no added cost) an orthonormal approximation also at the internal abscissas. This will guarantee that forming the matrix $\tilde{A}$ at the internal abscissas is a numerically stable procedure.
2. The method must be flexible enough to handle without modifications both cases $p = n$ and $p < n$.
3. The method must have a cost per step of $O(n^2 p)$, and never require a $O(n^3)$ expense when $p < n$.
4. The method must be based on numerically sound coordinates.
5. The method must allow for integration of the relevant differential equations with theoretical order restrictions given only by the degree of differentiability of Q.
6. The method must be well suited also in adaptive step-size mode: (i) we want to be able to increase the stepsize if Q evolves slowly, (ii) we want to be able to reject a step which is going to be unsuccessful as quickly as possible, possibly (much) earlier than having completed approximation for all of Q.
7. The method must be flexible enough to allow for increasing the number of columns of A which we want to triangularize, without having to restart the entire triangularization process from the beginning, and it should be powerful enough to exploit the work already done.

As we will see, the methods we laid down in [DV2] achieve the above points. These methods are based upon writing Q as product of Householder or Givens transformations, and since, in general, these elementary transformations do not vary smoothly on the whole interval of interest, we have to adopt a "reimbedding" strategy in order to obtain a well defined process. We refer to [DV2] for the original derivation of this approach, which we will review in Sections 2.1 and 2.2.

Now we consider a different interpretation of these methods based on Householder or Givens transformations. In what follows, we will assume that we have to integrate (1.2) when $p < n$; trivial simplifications take place if $p = n$.

We begin observing that, in principle, the sought solution Q can always be written in the redundant way $Q(t) = [\, Q(t) \quad Q^{\perp}(t) \,] \begin{bmatrix} I_p \\ 0 \end{bmatrix} =: U(t) \begin{bmatrix} I_p \\ 0 \end{bmatrix}$, $\forall t$, where $Q^{\perp}$ is the orthogonal complement of Q. Thus, we can think –at least in theory– to have the following representation for the sought solution Q:

$$Q(t) \;=\; U(t_k, t_0) Q(t, t_k) \;,\; t \geq t_k \;,\; Q(t_k, t_k) \;=\; \begin{bmatrix} I_p \\ 0 \end{bmatrix} . \tag{2.1}$$

In (2.1), $U(t_k, t_0) \in \mathbb{R}^{n \times n}$ and $Q(t, t_k) \in \mathbb{R}^{n \times p}$. The matrix $U(t_k, t_0)$ is at once comprising Q, at t_k, and its orthogonal complement. Now, let $U_k := U(t_k, t_0)$, and set

$$\hat{A}(t) \;=\; U_k^T A(t) U_k \,,\; t \geq t_k \,.$$

With this, we have that the equation satisfied by $Q(t, t_k)$ is again (1.2) with $\hat{A}$ replacing A there. With abuse of notation, if we still call $Q(t)$ what is really $Q(t, t_k)$, for

$t \geq t_k$, we would then have

$$\dot{Q} = \hat{A}Q - QQ^T\hat{A}Q + Q\hat{S}\,, \;\; \hat{S}_{ij} = \begin{cases} (Q^T\hat{A}Q)_{ij}, & i > j \\ 0, & i = j \\ -(Q^T\hat{A}Q)_{ji}, & i < j \end{cases}\,, \; Q(t_k) = \begin{bmatrix} I_p \\ 0 \end{bmatrix}. \tag{2.2}$$

(The equation for R, (1.1), is also modified trivially with $\hat{A}$ replacing A.)

Thus, quite clearly, knowledge of U_k would allow us to integrate the equation (2.2), starting "near the identity"; ostensibly, this can be done with a number of choices for representing $Q(t, t_k)$ valid locally, if not globally (recall our discussion on local charts). Now, suppose we can solve (2.2) from t_k to t_{k+1}, obtaining $Q(t_{k+1})$, and that at the same time we are able to obtain also the orthogonal complement of $Q(t_{k+1}, t_k)$, $Q^\perp(t_{k+1}, t_k)$; then, we could form the matrix $U_{k+1} = [\, Q(t_{k+1}) \quad Q^\perp(t_{k+1})\,]\; U_k$, redefine $\hat{A}$ and $Q(t, t_{k+1})$ accordingly for $t \geq t_{k+1}$ and then again formulate a differential equation for $Q(t, t_{k+1})$ identical to (2.2) for $t \geq t_{k+1}$. We could then repeat this basic setup until we arrive at t_f. Proceeding this way, we would always have to solve differential equations with initial conditions $\begin{bmatrix} I_p \\ 0 \end{bmatrix}$. Naturally, in practice we will only have computed approximations, and not exact values, but the setup remains unchanged.

Notice that, unless we explicitly compute also $Q^\perp$ so to triangularize all of A (and this would cost $O(n^3)$), the transformed matrix $\hat{A}$, at the t_k's, is upper triangular only in its left (n,p) part, in particular its bottom (n-p,n-p) block is not triangular. But, suppose that we now decide that we really needed to triangularize more columns of the fundamental matrix solution X (or, which is the same, wanted to transform larger part of A to upper triangular). If, somehow, we managed to keep track not only of $Q(t)$, but also of its orthogonal complement, then we could work only on the bottom (n-p,n-p) piece of the matrix $\hat{A}$. In other words, at the price of added memory of course, we would avoid having to restart from scratch. But, is it possible to obtain at once information on $Q(t_{k+1}, t_k)$ and $Q^\perp(t_{k+1}, t_k)$, for a cost proportional to $O(n^2 p)$ per step? This is where we can in principle exploit the representation for $Q(t, t_k)$ in terms of either Householder or Givens transformations. In fact, with these choices, one does get information on both $Q(t, t_k)$ and its orthogonal complement at the same time, at the expense of computing only $Q(t_{k+1}, t_k)$.

Warning. It should be stressed right away that Householder (or Givens) transformations are not globally defined; this means that if we want to represent $Q(t)$ in (2.1) by using these transformations, in general we cannot expect to have a globally smooth representation. As we showed in [DV2], it is a trivial matter to recover for free a smooth representation for $Q(t)$. However, it is **not trivial** at all to recover for free a smooth representation for $Q^\perp(t)$ (see also [CS]). After all, it is hardly imaginable that one can obtain a smooth representation for the orthogonal complement at the price of only getting Q! Regardless, lack of smoothness in the obtainable representation of $Q^\perp$ is not a concern in the contexts of which we are aware. For example, if the orthogonal complement is needed because we intend to carry further the triangularization process of A, then this can be done without restarting the entire computation from the beginning (but it is not a trivial implementation to do, since we need to save in memory all quantities which have been computed thus far).

Now, since Householder and Givens transformations are not unique (there is a sign ambiguity for each Householder reflection, and a decision to be made on the order in which we apply the Givens rotations), we must specify the way in which this

lack of uniqueness is resolved. This extra freedom means that a choice must be made between different coordinates systems to represent the same object. The best way to resolve this ambiguity is to make sure that we pick up the soundest coordinates from the numerical point of view.

These ideas have been carried out (in a different notation) in [DV2]. There, we wrote the solution $Q(t)$ of (1.2), locally, as product of elementary Householder or Givens transformations. We are ready to review the basic algorithms based on Householder and Givens transformations. We will not explicitly take advantage of the representation (2.1), but, by writing $Q(t)$ solution of (1.2) as product of elementary orthogonal matrices of Householder or Givens type, we are conceptually representing Q as in (2.1), with nonsmooth U.

2.1. Householder transformations. Suppose we are at t_k and that we know $X(t_k)$ (e.g., $t_k = t_0$). Then, to find $Q(t)$ such that $Q^T(t)X(t) = R(t)$, for $t \geq t_k$, we look for $Q^T(t) = P_p(t)\cdots P_1(t)$, with $P_i(t) = P_i^T(t)$ the Householder matrices

$$P_i(t) = \begin{bmatrix} I_{i-1} & 0 \\ 0 & Q_i(t) \end{bmatrix}\,,\quad Q_i(t) = I - \frac{2}{u_i^T(t)u_i(t)} u_i(t)u_i(t)^T\,.$$

After $(i-1)$ transformations, the matrix X got transformed into $P_{i-1}\dots P_1 X$ and its first $(i-1)$ columns have been triangularized. Let us still call X the transformed matrix, and let $x_i = X(i:n,i)$ be its i-th column we need to triangularize. This is the role of P_i. So, we will set

$$u_i(t) = x_i(t) - \sigma_i \|x_i\| e_1\,, \tag{2.3}$$

and continue the triangularization process. The standard textbook choice for the value of σ_i is (see [GVL]):

$$\sigma_i := \begin{cases} -1, & \text{if } e_1^T x_i(t_k) \geq 0, \\ 1, & \text{if } e_1^T x_i(t_k) < 0. \end{cases} \tag{2.4}$$

This is the idea. But, of course, we do not have X except at t_0. In [DV2] we showed that differential equations can be set for the u_i's directly, and for the norm of the vectors x_i, requiring only knowledge of the coefficient matrix A. Moreover, we also derived differential equations for the Householder transformations in different variables, namely for

$$v_i := \frac{u_i}{\|u_i\|}\,,\qquad Q_i = I - 2v_i v_i^T\,, \tag{2.5}$$

which are better scaled variables, since $v_i^T v_i = 1$.

We now recall the differential equations for the u and v-variables. For simplicity, we omit the subindices, and thus use the notation u for u_i, etc., and also use A for $A(i:n, i:n)$, where the matrix A has been progressively modified by the accumulated transformations:

$$(A,P_j)-\text{update}:\quad A(t) := P_j(t)A(t)P_j(t) - P_j(t)\dot{P}_j(t)\,,\ j = 1,\dots,i-1\,. \tag{2.6}$$

With this understanding, for the u-variables we have:

$$\frac{d}{dt}\begin{bmatrix} u \\ \sigma\|x\| \end{bmatrix} = \begin{bmatrix} A - 2e_1e_1^TA_s & (A - e_1e_1^TA_s)e_1 \\ 2e_1^TA_s & e_1^TA_se_1 \end{bmatrix}\begin{bmatrix} u \\ \sigma\|x\| \end{bmatrix} - \frac{u^TA_su}{\sigma\|x\|}\begin{bmatrix} e_1 \\ -1 \end{bmatrix}\,. \tag{2.7}$$

Here, $A_s = 1/2(A + A^T)$, e_1 is the first unit vector (of appropriate dimension), and (2.7) must be understood as a differential equation for the $u_i,\ i = 1, \ldots, p$.

For the v-variables, we have the following. Partition v as $v = \begin{bmatrix} (e_1^T v) \\ \hat{v} \end{bmatrix}$, let $\begin{bmatrix} a_{11} \\ \hat{a}_1 \end{bmatrix}$ be the first column of A, $(0, \hat{a}_{1,a}^T)$ be the first row of $\frac{1}{2}(A - A^T)$, $\begin{bmatrix} a_{11} \\ \hat{a}_{1,s} \end{bmatrix}$ be the first column of A_s, and let $\hat{A}$ and $\hat{A}_s$ be the submatrices obtained from A and A_s, respectively, by deleting the first row and column. Then, we have

$$\frac{d}{dt} \begin{bmatrix} (e_1^T v) \\ \hat{v} \end{bmatrix} = \begin{bmatrix} 0 & c^T - b^T \\ b - c & S - S^T \end{bmatrix} \begin{bmatrix} (e_1^T v) \\ \hat{v} \end{bmatrix}, \tag{2.8}$$

where we have set

$$b = \frac{2(e_1^T v)^2 - 1}{2} \hat{a}_1 + \hat{A}\hat{v}(e_1^T v)\,,\ c^T := (-\hat{a}_{1,a}^T \hat{v} + \alpha)\hat{v}^T\,,\ S = (\frac{2(e_1^T v)^2 - 1}{2(e_1^T v)} \hat{a}_1 + \hat{A}\hat{v})\hat{v}^T\,,$$
$$\text{and } \alpha := (a_{11}(e_1^T v) + \hat{a}_{1,s}^T \hat{v})(2(e_1^T v)^2 - 1) + 2(e_1^T v)\hat{v}^T(\hat{a}_{1,s}(e_1^T v) + \hat{A}_s \hat{v})\,.$$

The differential equations (2.7) and (2.8) can easily be supplied with initial conditions at t_0 since X_0 is known, and we can find Q_0 via Householder transformations (in either u or v formulation, with the σ's satisfying (2.4)). However, to describe the typical step between t_k and $t_{k+1} = t_k + h_k$, and regardless of the choice adopted between u or v-variables, the expression (2.4) used for choosing the sign of σ must be modified since in practice we do not have $X(t_k)$. The way it is done is to enforce (2.4), but by only keeping track of the transformations. It goes as follows.

Suppose we have found the Householder matrices at t_k, coming from t_{k-1}, call them $P_i^{(k-1)}(t_k)$. Call $P_i^{(k)}(t_k) = \begin{bmatrix} I_{i-1} & 0 \\ 0 & Q_i^{(k)}(t_k) \end{bmatrix}$ the possibly different initial condition for the Householder matrices (that is, different u's or v's and σ's) we need in order to step past t_k. Let $K_0^{(k)} = I_n$. Inductively define $\sigma_i^{(k)}\,,\ i = 1, \ldots, p$, as follows:

$$\sigma_i^{(k)} := \begin{cases} -1, & \text{if } \sigma_i^{(k-1)} e_1^T K_{i-1}^{(k)} Q_i^{(k-1)} e_1 \geq 0\,, \\ +1, & \text{if } \sigma_i^{(k-1)} e_1^T K_{i-1}^{(k)} Q_i^{(k-1)} e_1 < 0\,, \end{cases} \tag{2.9}$$

where the matrices $K_{i-1}^{(k)} \in \mathbb{R}^{n-i+1, n-i+1}\,,\ i = 2, \ldots, p$, are defined by

$$Q_{i-1}^{(k)}(t_k) K_{i-2}^{(k)} Q_{i-1}^{(k-1)}(t_k) = \begin{bmatrix} \sigma_{i-1}^{(k-1)} \sigma_{i-1}^{(k)} & 0 \\ 0 & K_{i-1}^{(k)} \end{bmatrix},$$

and, for $i = 1, \ldots, p$, the matrix $Q_i^{(k)}(t_k)$ is obtained so that

$$K_{i-1}^{(k)} Q_i^{(k-1)}(t_k) \sigma_i^{(k-1)} e_1 = Q_i^{(k)}(t_k) \sigma_i^{(k)} e_1\,.$$

Thus, we need to find a[3] Householder transformation which transforms the left-hand side of this last equation into $\sigma_i^{(k)} e_1$. This trivially gives new ICs for the $v_i^{(k)}(t_k)$; the sign ambiguity in the vector $v_i^{(k)}(t_k)$ is resolved by forcing the sign to that of the first

[3] not unique, there is typo in [DV2]

component of $v_i^{(k-1)}(t_k)$. Thus, we can prescribe new ICs for the $v_i^{(k)}(t_k)$, and from these it is simple to write ICs for the u_i's:

$$u_i^{(k)}(t_k) = -2\sigma_i^{(k)}\|x_i(t_k)\|(e_1^T v_i^{(k)}(t_k))v_i^{(k)}(t_k)\,.$$

The proof of the following Lemma is omitted since it amounts to a simple verification.

LEMMA 2.1. *The choice (2.9) is the same as (2.4).*

Remark 2.1. Let us substantiate the claim that the choice (2.4) ensures that we are using sound coordinates from a numerical point of view. In linear algebra, see [GVL], the choice of signs as in (2.4) is justified in order to avoid subtraction of (possibly) nearly equal numbers. Of course, this is still true in our context, since we will need to form the reflectors. But there is also another aspect to take into account in the present context. We restrict to the v-coordinates, since these must be generally preferred to the u-coordinates (but see the w-coordinates of (3.1) in Section 3). We seek an orthonormal function Q, and we are choosing local coordinates to represent it. Obviously, every column of Q, Qe_i, has unit length, and we represent these columns as $P_i \dots P_1 e_i$; to do this, we find the v_i's, each of which has itself unit length, by integrating (2.8). From the differential equations (2.8), we observe that there is a division by $e_1^T v_i$ in forming the vector $\frac{2(e_1^T v_i)^2-1}{2e_1^T v_i}\hat{v}$ of S (here, $e_1 = (1,0,\dots,0)^T \in \mathbb{R}^{n-i+1}$). For stability, we must ensure to avoid division by small numbers. But, with some simple algebra, one can see that:
The choice (2.9) is equivalent to having

$$(e_1^T v_i)^2 \geq \sum_{j=2}^{n-i+1} (e_j^T v_i)^2\ ,\ i = 1,\dots,p\,. \tag{2.10}$$

In particular, the vector $\frac{2(e_1^T v_i)^2-1}{2e_1^T v_i}\hat{v}$ is as well scaled as generally possible. Moreover, (2.10) can be used to decide if the current Householder frames are numerically stable or not.

In the u-variables, it is easier to check (2.4) directly, since $e_1^T x_i = e_1^T u_i + \sigma_i\|x_i\|$; thus, as long as

$$\sigma_i(e_1^T u_i + \sigma_i\|x_i\|) < 0\ ,\ i = 1,\dots,p\,, \tag{2.11}$$

there is no need to change the current frame.

We now summarize the overall strategy on a typical step from t_k to $t_{k+1} = t_k + h_k$. In the next section, we will pay closer attention to all aspects of this strategy.

Householder on $[t_k, t_{k+1}]$.

INPUT: t_k, $h_k > 0$, initial conditions $P_i^{(k-1)}(t_k)\,,\ i = 1,\dots,p$ (i.e., the vectors $u_i^{(k-1)}$ and $\|x_i^{(k-1)}\|$ or the vectors $v_i^{(k-1)}$ at t_k), and the $\sigma_i^{(k-1)}$.

(1) For $i = 1,\dots,p$, check to see if (2.10) (or (2.11)) holds true. If it fails, redefine $\sigma_i^{(k)}$ according to (2.9) and determine new $P_i^{(k)}(t_k) = \begin{bmatrix} I_{i-1} & 0 \\ 0 & Q_l^{(k)}(t_k) \end{bmatrix}$ accordingly (redefine $u_i^{(k)}(t_k)$ or $v_i^{(k)}(t_k)$), for $i = 1,\dots,p$

- For $i = 1,\dots,p$

(2) Let $A = A(i:n, i:n)$
(3) Find the Householder transformation $P_i^{(k)}(t)$ by integrating (2.7) or (2.8) on $[t_k, t_{k+1}]$
(4) Do an (A, P_i) update (2.6)

- Endfor i.

OUTPUT: $Q^{(k)}(t_{k+1})^T = P_p^{(k)}(t_{k+1}) \cdots P_1^{(k)}(t_{k+1})$, is such that $Q^{(k)}(t_{k+1})^T X(t_{k+1})$ is triangular.

2.2. Givens transformations. Suppose at t_k we know $X(t_k)$ (e.g., $t_k = t_0$). To find $Q(t)$ such that $Q^T(t)X(t) = R(t)$, for $t \geq t_k$, we look for $Q(t) = Q_1(t) \cdots Q_p(t)$, where $Q_i(t)$ is of the form $Q_i(t) = \begin{bmatrix} I_{i-1} & 0 \\ 0 & G_i(t) \end{bmatrix}$, and each G_i, $i = 1, \ldots, p$, is the product of elementary planar rotations (Givens, or Jacobi, transformations) of the type

$$Q_{ij}(t) = I - (e_1 e_1^T + e_j e_j^T) + G_{ij} \ , \ \ G_{ij} = c_{ij}(e_1 e_1^T + e_j e_j^T) - s_{ij}(e_1 e_j^T - e_j e_1^T) \ ,$$

for $j = 2, \ldots, n - i + 1$. Above, we have used c_{ij} and s_{ij} to express $\cos(\theta_{ij}(t))$ and $s_{ij} = \sin(\theta_{ij}(t))$, respectively, where the function $\theta_{ij}(t)$ needs to be found. Now, suppose we have triangularized the first $i-1$ columns of X, and still call X the transformed matrix. The role of G_i is to triangularize $x_i := X(i:n, i)$, the i-th column of the unreduced part of X.

In the standard linear algebra setting (see [GVL]), the rotators are safely applied in their natural sequence $Q_{i,i+1}, \ldots, Q_{i,n}$. But this may lead to instabilities in our time dependent setting! In fact, the specification of the order in which the rotators $Q_{i,i+1}, \ldots, Q_{i,n}$ are applied turns out to be of utmost importance for numerical stability, and therefore for accuracy and efficiency. We adopted the following strategy. *Order of rotators.* Our strategy is

(2.12) First rotator must annihilate largest entry of $x_i(2 : n - i + 1)$.

To be precise, define l to be the largest entry in absolute value of $x_i(2 : n - i + 1)$:

$$l \ : \ X_{i+l-1,i} = \max_{2 \leq j \leq n-i+1} |X_{i+j-1,i}| \, , \tag{2.13}$$

and define the index array π_i as

$$\pi_i = [1, l, 2, \ldots, l-1, l+1, \ldots, n-i+1] \, . \tag{2.14}$$

Then, define (the ordering of the rotators) G_i as

$$G_i \ = \ Q_{i,\pi_i(2)} \cdots Q_{i,\pi_i(n-i+1)} \, . \tag{2.15}$$

Remark 2.2. There seems to be no need to further refine (2.12) by selecting the second rotator to annihilate the largest entry of the unreduced part, and so forth.

In [DV2], we derived differential equations for the elementary rotators, that is for the θ_{ij} or for the corresponding $(\cos, \sin)$ pairs. To recall, omitting the row index i (i.e., using θ_j for θ_{ij}, etc.), these are

$$\begin{bmatrix} c_{\pi_i(3)} \cdots c_{\pi_i(n-i+1)} \ \frac{d}{dt}\theta_{\pi_i(2)} \\ c_{\pi_i(4)} \cdots c_{\pi_i(n-i+1)} \ \frac{d}{dt}\theta_{\pi_i(3)} \\ \vdots \\ c_{\pi_i(n-i+1)} \ \frac{d}{dt}\theta_{\pi_i(n-i)} \\ \frac{d}{dt}\theta_{\pi_i(n-i+1)} \end{bmatrix} = \begin{bmatrix} \alpha_{\pi_i(2)} \\ \alpha_{\pi_i(3)} \\ \vdots \\ \alpha_{\pi_i(n-i)} \\ \alpha_{\pi_i(n-i+1)} \end{bmatrix} . \tag{2.16}$$

Here, for $j = 2, \ldots, n-i+1$, we have set

$$\alpha_{\pi_i(j)}(t) = e_{\pi_i(j)}^T \big[Q_{i,\pi_i(n-i+1)}^T \cdots Q_{i,\pi_i(2)}^T A Q_{i,\pi_i(2)} \cdots Q_{i,\pi_i(n-i+1)} \big] e_1 \,,$$

and A is really $A(i:n, i:n)$, which has been progressively modified by the accumulated transformations:

$$(A, Q_j)-\text{update} : \; A(t) := Q_j(t)^T A(t) Q_j(t) - Q_j^T(t) \dot{Q}_j(t) \,, \;\; j = 1, \ldots, i-1 \,. \tag{2.17}$$

Of course, since $\frac{d}{dt} \begin{bmatrix} \cos(\theta_{ij}) \\ \sin(\theta_{ij}) \end{bmatrix} = \begin{bmatrix} -\sin(\theta_{ij}) \\ \cos(\theta_{ij}) \end{bmatrix} \frac{d}{dt}\theta_{ij}$, from (2.16) it is trivial to write differential equations for the $(\cos, \sin)$ pairs directly:

$$\begin{bmatrix} c_{\pi_i(3)} \cdots c_{\pi_i(n-i+1)} \frac{d}{dt} \begin{bmatrix} c_{\pi_i(2)} \\ s_{\pi_i(2)} \end{bmatrix} \\ \vdots \\ c_{\pi_i(n-i+1)} \frac{d}{dt} \begin{bmatrix} c_{\pi_i(n-i)} \\ s_{\pi_i(n-i)} \end{bmatrix} \\ \frac{d}{dt} \begin{bmatrix} c_{\pi_i(n-i+1)} \\ s_{\pi_i(n-i+1)} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 & -\alpha_{\pi_i(2)} \\ \alpha_{\pi_i(2)} & 0 \end{bmatrix} \begin{bmatrix} c_{\pi_i(2)} \\ s_{\pi_i(2)} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} 0 & -\alpha_{\pi_i(n-i)} \\ \alpha_{\pi_i(n-i)} & 0 \end{bmatrix} \begin{bmatrix} c_{\pi_i(n-i)} \\ s_{\pi_i(n-i)} \end{bmatrix} \\ \begin{bmatrix} 0 & -\alpha_{\pi_i(n-i+1)} \\ \alpha_{\pi_i(n-i+1)} & 0 \end{bmatrix} \begin{bmatrix} c_{\pi_i(n-i+1)} \\ s_{\pi_i(n-i+1)} \end{bmatrix} \end{bmatrix} . \tag{2.18}$$

To supply initial conditions at t_0 for the differential equations (2.16) and/or (2.18), we enforce (2.12); that is, use (2.13), (2.14), and apply the rotators in the order specified by (2.15). At each application of an elementary rotator on x_i at t_0, there is a sign ambiguity; we resolve this ambiguity by forcing the first entry of x_i to be always positive as the rotators are applied (and thus, equal to $\|x_i\|$ after all the elementary rotators $Q_{i,\pi_i(2)}, \ldots, Q_{i,\pi_i(n-i+1)}$, are applied). This way, we can start the integration.

But, to describe the typical step between t_k and $t_{k+1} = t_k + h_k$, the strategy just outlined, in particular the expressions (2.13) and (2.14), must be modified, since we do not have X at t_k. To understand the way we do this, we first need the following remark.

Remark 2.3. Givens transformation provide another example of the situation we described at the beginning of this work, when we discussed how a trajectory on a smooth manifold can be parametrized by overlapping local charts. Then, we said that an appropriate way to choose local coordinates was to enforce numerical stability. We now justify how this can be achieved with the choices we have adopted. Without loss of generality, assume that $\pi_i = [1, 2, \ldots, n-i+1]$ (if not, relabel the indices accordingly). From the differential equations (2.16) and (2.18), we notice that multiplication by the inverse of the diagonal matrix

$$\operatorname{diag}\big(c_3 \cdots c_{n-i+1},\; c_4 \cdots c_{n-i+1},\; \ldots,\; c_{n-i+1},\; 1\big)$$

is taking place. For numerical stability, we need to avoid division by small numbers. Clearly, the smallest number by which we are dividing (in absolute sense) is $c_3 \cdots c_{n-i+1}$. Let $x_{i,j}$ denote the jth component of x_i, $j = 1, 2, \ldots, n-i+1$. Then, using rotators to triangularize x_i, we have

$$c_j^2 = \frac{(\prod_{l=1}^{j-1} x_{i,l}^2)}{(\prod_{l=1}^{j} x_{i,l}^2)} \,,$$

and so

$$c_3^2 \cdots c_{n-i+1}^2 = \frac{(x_{i,1}^2 + x_{i,2}^2)}{\|x_i\|^2}\,.$$

Now, as long as

$$x_{i,1}^2 + x_{i,2}^2 \ \geq x_{i,j}^2,\ j = 3, \ldots, n-i+1\ ,\ i = 1, \ldots, p\,, \tag{2.19}$$

no division by small number is taking place, since when (2.19) is satisfied we have

$$c_3^2 \cdots c_{n-i+1}^2 \geq \frac{1}{(n-i)}\,,$$

which is as nicely bounded away from 0 as any product of cosines for Givens' transformations can generally be. Furthermore, with some simple algebra, one can see that: *The inequality (2.19) is equivalent to having*

$$\prod_{j=3}^{k} c_j^2 \ \geq\ s_k^2\ ,\ \ k = 3, \ldots, n-i+1\ ,\ \ i = 1, \ldots, p\,. \tag{2.20}$$

Clearly, (2.20) can be used to decide if the current ordering of Givens' transformations is numerically stable or not, without knowledge of X.

We are ready to describe how we modify the strategy which led us to (2.13) and (2.14) after the first step. First of all, as long as (2.20) holds, we do not change the present ordering of the rotators. In case (2.20) fails, we enforce (2.12), but by only keeping track of the transformations, in the following way. Suppose we have found the Givens transformation matrices at t_k, coming from t_{k-1}, call them $Q_i^{(k-1)}(t_k)$. Call $Q_i^{(k)}(t_k) = \begin{bmatrix} I_{i-1} & 0 \\ 0 & G_i^{(k)}(t_k) \end{bmatrix}$ the possibly different initial condition for the new Givens matrices (that is, different θ's or cosines/sines) we need in order to step past t_k. Define $K_0^{(k)} = I_n$, and inductively define

$$w_i := K_{i-1}^{(k)} G_i^{(k-1)}(t_k) e_1\,,\ \text{ for } i = 1, \ldots, p\,. \tag{2.21}$$

Let l be the index of the largest entry (in absolute value) of $w_i(2 : n-i+1)$. Accordingly, define π_i as in (2.14) and the ordering for the rotators relative to $G_i^{(k)}(t_k)$ as in (2.15). Find initial conditions for the $Q_{i,j}^{(k)}(t_k)$ by enforcing that all vectors $\prod_{j=2}^{n-m+1} (Q_{i,\pi_i(j)}^{(k)}(t_k))^T w_i$, $m = n-1, \ldots, i$, have positive first component. This way, we define $G_i^{(k)}(t_k)$, hence $Q_i^{(k)}(t_k)$. Finally, for $i = 2, \ldots, p$, we define $K_{i-1}^{(k)} \in \mathbb{R}^{n-i+1, n-i+1}$ from

$$Q_{i-1}^{(k)}(t_k)^T \begin{pmatrix} I_{i-2} & 0 \\ 0 & K_{i-2}^{(k)} \end{pmatrix} Q_{i-1}^{(k-1)}(t_k) = \begin{pmatrix} I_{i-1} & 0 \\ 0 & K_{i-1}^{(k)} \end{pmatrix}.$$

Again we omit the proof of the following Lemma, since it is a direct verification.

LEMMA 2.2. *The choice just described for providing initial conditions on the rotators at t_k is equivalent to the strategy based on (2.15) and first paragraph after (2.18).*

We summarize the overall strategy on a typical step from t_k to $t_{k+1} = t_k + h_k$. We assume that integration has been successful up to t_k, and that the step h_k has been selected.

Givens on $[t_k, t_{k+1}]$.

INPUT: t_k, $h_k > 0$, initial conditions $Q_i^{(k-1)}(t_k)$, $i = 1, \ldots, p$ (i.e., either the $(\cos, \sin)$ pairs or the θ values, and the ordering in which they had been applied).

(1) For $i = 1, \ldots, p$, check to see if (2.20) holds true. If it fails, redefine the index array π_i and initial conditions for the rotators at t_k. That is, for $i = 1, \ldots, p$, let $Q_i^{(k)} = Q_{i,\pi_i(2)} \cdots Q_{i,\pi_i(n+i-1)}$, and find initial conditions for $Q_i^{(k)}(t_k)$ by bringing $w_i(i:n)$ into e_1 so that all rotated vectors $\prod_{j=2}^{n-m+1}(Q_{i,\pi_i(j)}^{(k)}(t_k))^T w_i$, $m = n-1, \ldots, i$, have positive first component.

- For $i = 1, \ldots, p$
  (2) Let $A = A(i:n, i:n)$.
  (3) Find the transformation $Q_i^{(k)}(t)$ by integrating the differential equations (2.16) or (2.18) on $[t_k, t_{k+1}]$.
  (4) Do an (A, Q_i) update (2.17).
- Endfor i.

OUTPUT: $Q^{(k)}(t_{k+1}) = Q_1^{(k)}(t_{k+1}) \cdots Q_p^{(k)}(t_{k+1})$, is such that $(Q^{(k)}(t_{k+1}))^T X(t_{k+1})$ is triangular with positive diagonal entries.

Remark 2.4. We observe that –even through a change of initial conditions– with our strategy the signs on the diagonal of R, see (1.1), remain fixed when using rotators to find Q.

3. Implementation. Here we describe how we implemented the algorithm put forward in the previous section. Before doing so, however, we want to derive a different formulation for the Householder transformations, which present some distinct advantages over both the u and v formulations. The motivation comes from Remark 2.1.

We experimented with two different rewritings of the Householder vectors.

(a) With the notation of (2.8), we observe that $(e_1^T v) = \sigma(1 - \hat{v}^T \hat{v})^{1/2}$. Thus, one can write differential equations just for $\hat{v}$, and then recover all of v by using $(e_1^T v) = \sigma(1 - \hat{v}^T \hat{v})^{1/2}$. On the surface, this seems to present some advantages, since there is one less differential equation to solve. However, there is a potential loss of precision which occurs with this approach when forming back $e_1^T v$. After all, we know that the first component of v dominates all others; thus, we expect that some of the other components will be small (and we often observed this to be true in practice). Indeed, in our experiments, this formulation gave consistently less precise results than the v formulation or the w formulation to be introduced next. For this reason, henceforth we will not present computational results obtained with this formulation.

(b) Consider the new variable w:

$$w \;=\; \frac{1}{e_1^T v}\, v\,, \text{ thus } w \;=\; \begin{bmatrix} 1 \\ \hat{w} \end{bmatrix}. \tag{3.1}$$

Observe that

$$(e_1^T v) \;=\; \frac{-\sigma}{\|w\|}\,,$$

so that the discussion relative to the choice of σ's (see (2.9)) stays unchanged. Since $\dot{w} = \frac{dw}{dt} = \begin{bmatrix} 0 \\ \frac{d\hat{w}}{dt} \end{bmatrix}$, we have one less differential equation to solve and a simplified form for the (A, P_i)-updates. Omitting the indices for simplicity, the form of the typical update becomes

$$PAP - P\dot{P} = \tag{3.2}$$
$$= A - \frac{2}{w^T w}(w(w^T A) + (Aw)w^T) + 4\frac{w^T Aw}{(w^T w)^2} ww^T - \frac{2}{w^T w}(w\dot{w}^T - \dot{w}w^T) \, .$$

From (3.2), it is easy to derive the differential equation satisfied by $\hat{w}$. In fact, this can be obtained from the requirement that $(PAP - P\dot{P})e_1$ has only its first entry not 0. Using the form of $\dot{w}$, and the same notation used to derive (2.8), this requirement becomes

$$\hat{a}_1 - \frac{2}{w^T w}((a_{11} + \hat{w}^T \hat{a}_1)\hat{w} + \hat{a}_1 + \hat{A}\hat{w}) + 4\frac{w^T Aw}{(w^T w)^2}\hat{w} + \frac{2}{w^T w}\dot{\hat{w}} = 0 \, ,$$

from which we get the equation for $\hat{w}$:

$$\frac{d\hat{w}}{dt} = \big[a_{11} + \hat{w}^T \hat{a}_1 - 2\frac{w^T Aw}{w^T w}\big]\hat{w} + (1 - \frac{w^T w}{2})\hat{a}_1 + \hat{A}\hat{w} \, . \tag{3.3}$$

Remark 3.1. Notice that the division by $w^T w$ in (3.3) is perfectly safe, given the form of w. Of course, if one uses the w-variables, obvious modifications take place in the skeleton of the algorithm on page 9. E.g., (2.10) now would read

$$1 - (w_{i+1,i}^2 + \ldots + w_{n,i}^2) \geq 0 \, . \tag{3.4}$$

To sum up, for Householder methods, we have considered three possibilities: (i) u-variables, (ii) v-variables, and (iii) w-variables. In the v-variables, we need to integrate (2.8) whose solution has norm 1 for all t, and we must maintain this property under discretization, at gridpoints. In principle, we could use Gauss RK schemes for this task; we did this in [DV2], by using Newton's method to solve the resulting nonlinear system, but this gives a cost of $O(pn^3)$ per step which is more than what we are willing to pay. Use of the linearly convergent scheme of [DRV1] forces a severe stepsize restriction, which is clearly incompatible with adaptive time stepping. For these reasons, for the v-variables, in this work we propose integrating (2.8), with an explicit scheme followed by a renormalization at each step. The resulting method is a "local projection" method for the v-variables and has cost of $O(n^2 p)$ per step. As far as the integration for the u or w-variables, integration can be carried out with explicit schemes, with no need of renormalization. Notice, though, that when forming the Householder transformations, and hence Q, one is in fact normalizing things so to keep them orthogonal; this is obvious: in the matrix $I - 2ww^T/w^T w$, the term $ww^T/w^T w$ is indeed an orthogonal projection.

As far as methods based on Givens transformations, so far, we spent more time trying to obtain good codes only for the formulation in terms of the θ-variables, see (2.16), and not for the $(\cos, \sin)$-variables. There are several reasons for our choice. First, a simplicity reason: it is simpler to work with the θ-variables, and then form the rotators. Second, if we work with $(\cos, \sin)$, then we must integrate (2.18) maintaining

the solution of norm 1: again, we could use Gauss RK schemes with Newton's method, or a "local projection" technique. But, in all cases, (2.18) is twice the dimension of the system to be solved with the θ-variables. Repeated use of the trigonometric identity $\cos^2\alpha + \sin^2\alpha = 1$ would allow us to half the dimensions, of course, but at the price of added nonlinearities. Third, there is an accuracy reason to prefer the θ-variables; in all our tests, they gave more accurate results than their $(\cos, \sin)$ counterpart. An explanation for this fact is the content of the next Lemma.

LEMMA 3.1. *Let θ be the angle associated to the elementary rotation $Q(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$. Let ϕ be an approximation to θ, and let $Q(\phi)$ be the elementary rotator associated to ϕ. Consider the error matrix $E := Q^T(\theta)(Q(\phi) - Q(\theta))$. If $\theta - \phi = \eta$, sufficiently small, then the error on the diagonal of E is $O(\eta^2)$:*

$$E = \begin{bmatrix} -\frac{\eta^2}{2} + O(\eta^4) & \eta + O(\eta^3) \\ -\eta + O(\eta^3) & -\frac{\eta^2}{2} + O(\eta^4) \end{bmatrix}.$$

Proof. The proof follows from $Q^T(\theta)Q(\phi) = \begin{bmatrix} \cos(\theta-\phi) & \sin(\theta-\phi) \\ -\sin(\theta-\phi) & \cos(\theta-\phi) \end{bmatrix}$. □

In particular, this Lemma implies that $\|E\| = |\eta| + O(\eta^2)$. Instead, an error equal to η on $\cos(\theta) - \cos(\phi)$ and on $\sin(\theta) - \sin(\phi)$, would have rendered $\|E\| = |\eta|\sqrt{2} + O(\eta^2)$. In the general case of $Q \in \mathbb{R}^{n\times p}$, with notation similar to Lemma 3.1, by taking into account the general form of the matrix Q as product of rotators, and using Lemma 3.1 over and over, it is simple to realize that an error of order η on the angles still gives an $O(\eta^2)$ error term on the diagonal of E, whereas an error $O(\eta)$ for the cos and sin would give an $O(\eta)$ error term for all entries of E.

Remark 3.2. Of course, the θ-variables are more properly seen as variables on a torus. Indeed, also to avoid pathological cases in which the θ values would grow unbounded, we always renormalize their values to $[-\pi, \pi]$, which we do by computing inverse tangent. This represents a drawback with respect to working directly with the $(\cos, \sin)$ pairs.

Remark 3.3. Householder methods based on the w-variables, and Givens' methods based on the θ-variables parametrize the sought Q using exactly $p\frac{2n-p-1}{2}$ parameters: the minimal number required.

Linear Algebra Involved. An advantage of using Householder and Givens transformations to represent Q is that we can take advantage of the many clever ways in which these transformations can be manipulated, see [GVL], and thus obtain efficient procedures. For example, we keep the matrix Q in factored form, and never form it (except when required for output purposes). Of course, we never perform matrix-matrix multiplications either, but we exploit inner product arithmetic for Householder matrices and the simple structure of the rotators for Givens matrices.

Discretization Schemes. We have chosen to use explicit integrators of RK type as the basic schemes. We are not interested in extremely accurate computations; the range of practical interest for us is between 10^{-2} and 10^{-10}, and our schemes have been chosen with this accuracy demands in mind. If one needs more accurate computation, then different choices would be more appropriate. We chose formulas of order 4 and of order 5 with an associated embedded formula, to be used in variable stepsize mode. The formulas used are the well known Runge-Kutta 3/8-th rule, a scheme of order 4 with an embedded scheme of order 3, and the formulas of Dormand-Prince of order

5 with the embedded scheme of order 4. For convenience, we give these pairs below. The lower order scheme is the one relative to the weights $\hat{b}$.

0	0	0	0	0	0
$\frac{1}{3}$	$\frac{1}{3}$	0	0	0	0
$\frac{2}{3}$	$-\frac{1}{3}$	1	0	0	0
1	1	-1	1	0	0
b	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$	0
1	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$	0
$\hat{b}$	$\frac{1}{12}$	$\frac{1}{2}$	$\frac{1}{4}$	0	$\frac{1}{6}$

3/8-th Runge-Kutta 4-3 pair

0	0	0	0	0	0	0	0
$\frac{1}{5}$	$\frac{1}{5}$	0	0	0	0	0	0
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$	0	0	0	0	0
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$	0	0	0	0
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$	0	0	0
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$	0	0
b	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
$\hat{b}$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Dormand–Prince 5-4 pair.

Remark 3.4. With our schemes based on elementary transformations, one can form orthogonal approximations also at the internal RK points, not just at the grid points, at negligible extra cost. This is a nice fact, both for dense output purposes, and for forming $\tilde{A}$ at the internal points.

Updates. Whenever the updates are required, see (2.6) and (2.17), we do not setup the updated matrices by using exact derivatives (as we did in [DV2]), since we can (and do) use the values obtained at the Runge-Kutta stages; this way, the update does not require extra function evaluations.

Error Control. Finally, about error control for variable time-stepping integration. Most of the criteria we used are standard. We do mixed absolute/relative error control with respect to the input value `TOL`, in the way explained in [HNW, II-4], with some slightly different heuristics. For example we use a safety factor of 0.8 (it is `fac` in [HNW]), we never allow a new stepsize to be bigger than four times the current stepsize, and we choose the initial stepsize to be $\texttt{TOL}^{1/(q+1)}$, where q is the order of the estimator (i.e., $q = 3$ for the embedded 3/8th rule, and $q = 4$ for the embedded Dormand-Prince rule). But the most relevant changes to the standard strategies are due to the nature of our methods. In fact, triangularizing one column of X at the time, as we do, present some important advantages. First of all, we decide on stepsize changes by monitoring the behavior of the error (in the ∞-norm) on all columns independently, and then take the most conservative estimate for the next step. But, most importantly, there is no need to complete the entire integration step prior to rejecting the step! This is a pleasant outcome of the present implementation, since often (see Section 4) a step failure occurs ahead of having completed computation of all of Q.

Reimbedding. We have used the construction based on (2.9), or (2.21), and following discussion, only if the tests (2.11), or (2.10), or (3.4), or (2.20), for the u, v, w, or θ

variables, respectively, failed. We have not succeeded in finding less expensive ways to obtain new initial conditions in case these tests failed.

4. Codes & Examples. We have written `FORTRAN` codes using the previous ideas. In the examples, we will refer to the performance of each of these codes, which differ by which formulas are used, and whether or not they are implemented in constant or variable stepsize modes. We use the following first letter convention: `u`=u-variables, `v`=v-variables, `w`=w-variables, `t`=θ-variables.

- Fixed stepsize codes.
  - 3/8th rule: `urk38, vrk38, wrk38, trk38`. Thus, for example, `wrk38` is a fixed stepsize implementation using the Runge-Kutta 3/8th rule of the Householder method based on the w-variables.
  - Dormand-Prince rule: `udp5, vdp5, wdp5, tdp5`.
- Variable stepsize codes. The naming convention is as above, but now the first letter is a `v` to signify "variable stepsize".
  - 3/8th pair: `vurk38, vvrk38, vwrk38, vtrk38`. For example, `vvrk38` is the variable stepsize implementation of the 3/8th pair for the Householder method in v-variables.
  - Dormand-Prince pair: `vudp5, vvdp5, vwdp5, vtdp5`.

We report on several measures of performance of the above codes.

- `err`: the error between computed and exact Q, if the exact Q is known.
- `reimb`: the number of reimbeddings needed for a given method; we increment the counter every time a change of coordinates is performed.
- `rejs/first`: the number of total rejections (in variable stepsize mode) followed by the rejections occurring while triangularizing the first column.
- `cpu`: the total `CPU` time needed to complete a given run normalized to 1 for the fastest run on the given problem.
- `nsteps`: the total number of steps taken (in variable stepsize mode).

For the variable stepsize codes, we also include comparison with a projected integrator, `prk45`, which integrates (1.2) with the well known (and sophisticated) integrator `RKF45` of `Netlib`, and then uses modified Gram-Schmidt for the projection. We recall that `RKF45` is a RK solver of order 5/4 whose performance is comparable with the Dormand-Prince 5/4 pair we adopted here.

Example 4.1. This is a problem chosen because the underlying fundamental solution is both exponentially dichotomic and fast rotating. Bad projection methods have difficulties on this problem. Also Gauss RK schemes without Newton method run into serious stepsize restrictions. We have the coefficient matrix

$$A(t) = \begin{bmatrix} \beta\cos(2\alpha t) & -\alpha + \beta\sin(2\alpha t) \\ \alpha + \beta\sin(2\alpha t) & -\beta\cos(2\alpha t) \end{bmatrix},$$

and we seek Q associated to the QR factorization of X : $X' = AX$, $X(0) = I$. The exact solution is $X(t) = \begin{bmatrix} \cos(\alpha t) & \sin(\alpha t) \\ \sin(\alpha t) & -\cos(\alpha t) \end{bmatrix} \begin{bmatrix} e^{\beta t} & 0 \\ 0 & -e^{-\beta t} \end{bmatrix}$. We fix $\beta = 100$, $\alpha = 100$, and consider integration on the interval $[0, b]$ with $b = 10$. The problem should cause difficulties to methods based on Householder transformations, because of the fast rotation of Q. This does indeed produce several reimbeddings for Householder methods, but no appreciable deterioration in accuracy.

Observe that for the `t` methods there is only one equation to integrate:

$$\dot{\theta} = \alpha - \beta \sin(2\theta(t)) \cos(2\alpha t) + \beta \sin(2\alpha t) \cos(2\theta(t)) \ , \ \ \theta(0) = 0 \, ,$$

and this has the exact solution $\theta(t) = \alpha t$. So, one may expect no error while integrating for θ. However, the reason why the `t` methods do not recover the exact solution is because we automatically renormalize angles to $[-\pi, \pi]$ and this causes roundoff errors to enter in the picture; if we do not renormalize the angles, then the exact solution is recovered.

Tables 4.1 and 4.2 summarize the results of our numerical experiments. For the fixed step methods in Table 4.1 the small error for the `t` methods is notable and this is mirrored by the superior performance for the variable step `t` methods (see Table 4.2). Note that the `u` methods fail on this problem. Also, observe how `prk45` is considerably more expensive than the competing 5/4 codes (`vvdp5, vwdp5`).

Table 4.1. Example 4.1: fixed stepsize $\Delta t = 1.E-3$.

Meth	err	reimb	cpu
tdp5	$3.1E-13$	0	17.5
trk38	$3.9E-13$	0	14.2
udp5	–	–	–
urk38	–	–	–
vdp5	$2.5E-9$	318	24.0
vrk38	$1.6E-6$	318	18.9
wdp5	$3.9E-8$	318	18.9
wrk38	$2.4E-6$	318	15.5

Table 4.2. Example 4.1: variable stepsize, `tol`$= 1.E-8$.

Meth	err	reimb	rejs	cpu	nsteps
prk45	$1.4E-8$	–	–	31.1	20803
vtdp5	$4.6E-8$	0	162	1.2	599
vtrk38	$2.5E-8$	0	158	1	705
vudp5	–	–	–	–	–
vurk38	–	–	–	–	–
vvdp5	$3.3E-9$	318	0	20.4	9557
vvrk38	$7.2E-9$	318	1	61.4	37931
vwdp5	$3.0E-9$	318	718	20.4	11623
vwrk38	$4.6E-9$	318	1835	46.0	34317

Example 4.2. Here we take the coefficients matrix

$$A(t) = \alpha(\theta(t) - \sin(t)) \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} ,$$

where $\theta(t) = \frac{\alpha}{1+\alpha^2}(\exp(-\alpha t) + \alpha \sin(t) - \cos(t))$ and we fixed $\alpha = 100$. Interval of integration is $[0, 10]$ and exact solution is $Q(t) = \begin{bmatrix} \cos(\theta(t))) & \sin(\theta(t)) \\ -\sin(\theta(t)) & \cos(\theta(t)) \end{bmatrix}$. This is a difficult problem for all the methods considered, but all methods except `prk45`

obtain accurate solutions as seen in Tables 4.3 and 4.4. The integrator `RKF45` fails to integrate in one-step mode on the desired interval, though it succeeds on $[0, 9]$, much less efficiently than `vtdp5, vvdp5, vwdp5`. We expected the θ methods to run into difficulty in variable stepsize mode, because of stiffness of the θ differential equation, but did not notice that.

Table 4.3. Example 4.2: fixed stepsize $\Delta t = 1.E-3$.

Meth	err	reimb	cpu
tdp5	$1.5E-12$	0	169.
trk38	$1.5E-10$	0	136.
vdp5	$6.2E-12$	0	209.
vrk38	$1.5E-10$	0	170.
wdp5	$6.2E-12$	0	163.
wrk38	$1.6E-10$	0	131.

Table 4.4. Example 4.2: variable stepsize, tol$= 1.E-8$.

Meth	err	reimb	rejs	cpu	nsteps
prk45	–	–	–	–	–
vtdp5	$5.3E-9$	0	8	1	53
vtrk38	$5.1E-9$	0	8	3.0	206
vudp5	$1.5E-8$	0	10	2.0	93
vurk38	$2.6E-8$	0	1	4.0	280
vvdp5	$6.9E-9$	0	10	2.9	106
vvrk38	$5.9E-8$	0	8	5.0	263
vwdp5	$1.3E-8$	0	12	1.2	66
vwrk38	$6.4E-9$	0	20	3.1	238

Example 4.3. This is a coefficients' matrix arising from a stiff two-point boundary value problem having both boundary and interior layers. We have

$$A(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ t/(2\epsilon) & 0 & 1 & 1/2 \\ 1/\epsilon & 0 & 0 & 0 \\ 0 & 1/\epsilon & 1/\epsilon & -t/(2\epsilon) \end{bmatrix}$$

and we take $\epsilon = 10^{-2}$. Interval of integration is $[-1, 1]$. Exact solution is not known. The u-methods fail to complete the integration, but all other methods perform well. For this problem, see Tables 4.6-(i) and 4.6(ii), most rejections occur after having computed all of Q.

Table 4.5. Example 4.3: fixed stepsize $\Delta t = 1.E-3$.

Meth	reimb	cpu
tdp5	2	8.7
trk38	2	6.2
vdp5	3	12.2
vrk38	3	8.8
wdp5	3	10.0
wrk38	3	7.2

Table 4.6-(i). Example 4.3: variable stepsize, tol$= 1.E-8$.

Meth	reimb	rejs/first	cpu	nsteps
prk45	–	–	1.4	252
vtdp5	2	11/2	1	221
vtrk38	2	15/4	1.7	628
vudp5	–	–	–	–
vurk38	–	–	–	–
vvdp5	3	9/1	1.3	217
vvrk38	3	14/2	2.7	612
vwdp5	3	10/2	1.1	228
vwrk38	3	13/3	2.3	649

Table 4.6-(ii). Example 4.3: column-rejections, tol$= 1.E-8$.

Meth	1st	2nd	3rd
vtdp5	2	1	8
vtrk38	4	0	11
vvdp5	1	0	8
vvrk38	2	0	12
vwdp5	2	1	7
vwrk38	3	0	10

Example 4.4. This is a fairly simple problem quite useful for testing purposes. We have the coefficient matrix

$$A(t) = Q(t)D(t)Q^T(t) + \dot{Q}(t)Q^T(t)\,,$$

where $D(t) = \mathrm{diag}(1, \cos(t), -\frac{1}{2\sqrt{t+1}}, -10)$, $Q(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & Q_\beta(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Q_\alpha(t) & 0 \\ 0 & Q_\alpha(t) \end{bmatrix}$, and $Q_\gamma(t) = \begin{bmatrix} \cos(\gamma t) & \sin(\gamma t) \\ -\sin(\gamma t) & \cos(\gamma t) \end{bmatrix}$. We report on results for $\alpha = 1$ and $\beta = \sqrt{2}$ and integrate on $[0, 100]$. Notice the inadequacy of the u-variables. Such inadequacy is hidden in constant stepsize mode, but it becomes apparent in variable stepsize mode: the u-vector becomes very poorly scaled. In variable stepsize, nearly all rejections

occur while triangularizing the first column. From Table 4.8, observe how `prk45` takes more steps and is less accurate than `vtdp5, vvdp5, vwdp5`.

Table 4.7. Example 4.4: fixed stepsize $\Delta t = 1.E-3$.

Meth	err	reimb	cpu
tdp5	$1.6E-10$	27	22.7
trk38	$1.5E-10$	27	16.9
vdp5	$1.6E-10$	77	24.3
vrk38	$1.5E-10$	77	19.1
wdp5	$1.6E-10$	77	22.2
wrk38	$1.5E-10$	77	17.3

Table 4.8. Example 4.4: variable stepsize, tol= $1.E-8$.

Meth	err	reimb	rejs/first	cpu	nsteps
prk45	$2.1E-7$	–	–	1.1	5053
vtdp5	$7.7E-9$	27	94/89	1	4533
vtrk38	$1.2E-8$	27	107/100	2.1	13010
vudp5	–	–	–	–	–
vurk38	–	–	–	–	–
vvdp5	$1.2E-8$	77	80/80	1.1	3967
vvrk38	$1.2E-8$	77	99/99	2.1	11169
vwdp5	$1.4E-8$	77	103/92	1.1	4370
vwrk38	$2.8E-8$	77	125/115	2.3	12694

Example 4.5. Here we take the diagonal coefficient matrix

$$A(t) = \mathrm{diag}(-\frac{1}{2\sqrt{t+1}}, -10, \cos(t), 1)\,,$$

which is just a permutation of $D(t)$ in Example 4.4. We take both $Q(0) = I$ and a random initial condition $Q(0)$, to verify if the diagonal elements of D will become reordered. When $Q(0) = I$, then the computed solution remains $Q(t) = I$ for all t. For randomly chosen $Q(0)$, the change of variables Q does instead reorder the diagonal of $\tilde{A}$: $\tilde{A}_{11} \geq \cdots \geq \tilde{A}_{44}$ (see Figure 4.1; results obtained with `vtdp5`).

Example 4.6. This is an example of a constant coefficients matrix. The equation (1.2) reduces to a so-called isospectral flow. For these problems, it is known that the diagonal of the upper (p, p) block of the transformed matrix must eventually converge to the real parts of the leading eigenvalues of A.Far from advocating the ideas set forth in this paper as a mean to solve the eigenvalue problem, we included this problem because we wanted to highlight (on a problem of arbitrarily high dimensions) how most stepsize rejections occur ahead of having found all of Q; for this reason, we report on results obtained with the variable stepsize codes `vtdp5, vvdp5, vwdp5`, and, for comparison, with `prkf45`, in spite of the fact that probably constant coefficients problems can be efficiently solved also with constant stepsize. Also, we wanted to see how ill conditioning (i.e., poor separation) of the eigenvalues affected convergence.

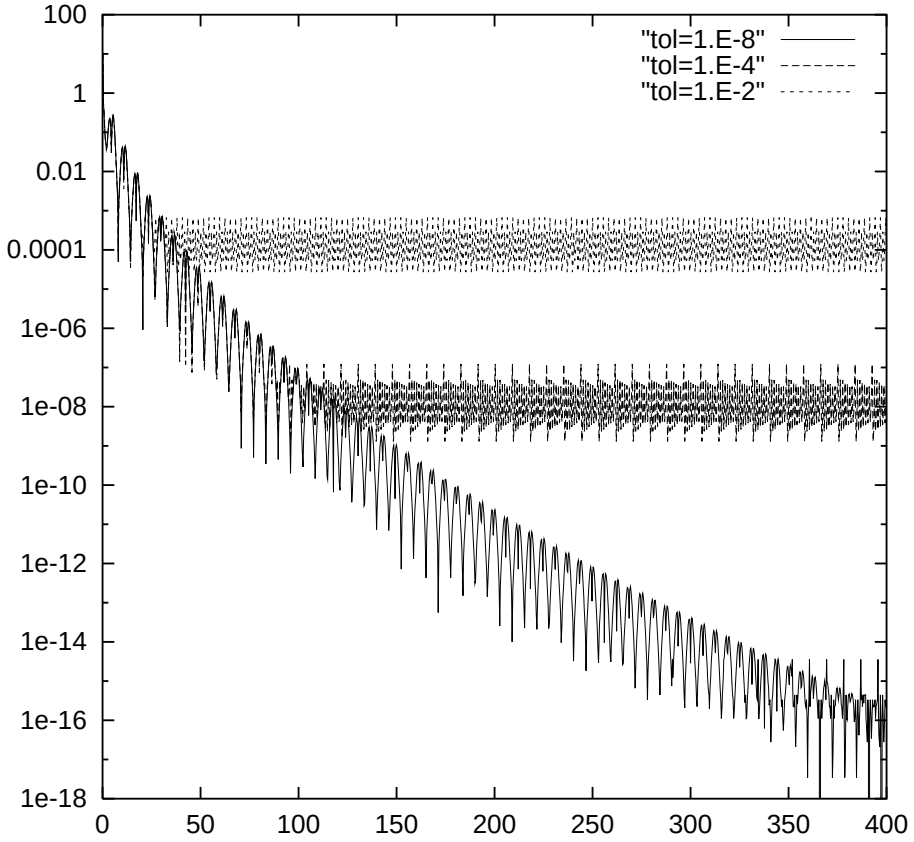

FIG. 4.1. *Plot of t vs.* $\log_{10}(||D(t) - \mathrm{diag}(\tilde{A}(t))||_\infty)$ *for* `vtdp5` *applied to Example 4.5 for different tolerances using a "random" $Q(0)$.*

The coefficient matrix is the upper Hessenberg Frank matrix:

$$A = \begin{pmatrix} n & n-1 & n-2 & \dots & 1 \\ n-1 & n-1 & n-2 & \dots & 1 \\ 0 & n-2 & n-2 & \dots & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 1 \end{pmatrix}.$$

We take $n = 25$, and seek Q triangularizing $X \in \mathbb{R}^{n\times p}$, $X' = AX$, $X(0) = \begin{bmatrix} I_p \\ 0 \end{bmatrix}$, with $p = 13$. Interval of integration is $[0, b]$. Exact solution is not known, so we report on the defect of the diagonal of the upper (p, p) block of the transformed matrix $\tilde{A}(b)$ from the real parts of the leading eigenvalues; this we list as `errd`. At four digits, the eigenvalues are 77.9837, 60.5984, 47.7777, 37.5667, 29.2021, 22.2856, 16.5772, 11.9193, 8.2006, 5.3359, 3.2479, 1.8495, $1.1841 \pm 0.2634i$, $0.8147 \pm 0.6124i$, $0.4098 \pm 0.7755i$, $0.0051 \pm 0.7737i$, $-0.3373 \pm 0.6031i$, $-0.5452 \pm 0.3177i$, -0.6070.

In the next table, we report on selected runs with the variable stepsize codes for $b = 100$. The value of `errd` is entirely due to lack of convergence to the real part of the 13-th eigenvalue, 1.1841, all larger eigenvalues having already converged to several digits. Notice that (for most cases) `errd` does not change appreciably by decreasing the tolerance value, which indicates that only by increasing the length of integration one would be able to approximate the 13th eigenvalue more accurately. Also, we notice that our codes are clearly superior to `prkf45`. Finally, in our codes, a large majority of rejections occur while integrating for the 4th or 5th columns; for example, for `vtdp5` with `tol=1.E-6`, the rejection patterns is as follows: $[1, 0, 11, 372, 104, 4, 0, 0, 1, 14, 1, 4, 3]$, where each number in this vector refers to rejections occurred while triangularizing columns 1 to 13.

Table 4.9. Example 4.6: variable stepsize.

Meth	tol	errd	reimb	rejs	cpu	nsteps
prk45	$1.E-4$	$1.8E-1$	-	-	1.7	5365
vtdp5	$1.E-4$	$9.6E-2$	73	525	1.5	2391
vvdp5	$1.E-4$	$1.1E0$	77	516	1.2	2459
vwdp5	$1.E-4$	$3.0E0$	85	504	1	2462
prk45	$1.E-6$	$1.8E-1$	-	-	1.7	5430
vtdp5	$1.E-6$	$1.8E-1$	65	515	1.6	2459
vvdp5	$1.E-6$	$5.6E-2$	35	501	1.2	2491
vwdp5	$1.E-6$	$1.1E-1$	34	476	1.	2481

5. Conclusions. The purpose of this work has been manifold. We discussed general design choices people should have in mind when devising techniques for solving (1.2). We reinterpreted methods based on Householder and Givens transformations as methods which give a trajectory on the smooth manifold of orthonormal matrices, and use overlapping local charts to parameterize the manifold. We gave new formulations and new implementations of methods based on Householder and Givens transformations. Finally, we presented a suite of `FORTRAN` codes for solving (1.2) by our techniques. We believe that the methods put forward in this work, and their implementation as laid down here, are a very sensible way to solve (1.2), and should not be ignored by anyone interested in comparative performance of other techniques for solving (1.2). In this spirit, we welcome other people to use our codes[4].

Our study showed that all design scopes we had set at the beginning are satisfied by appropriate implementation of our techniques. There is certainly room for improving our codes, and we anticipate some work towards more efficient implementation, in particular insofar as data structure, memory requirement, and extensive use of the `BLAS` libraries. However, we believe that our implementations are now sophisticated enough that some conclusions and recommendations for future work can be given.

1. Much as one should have expected, in variable stepsize the `dp5` codes outperform the `rk38` codes. In constant stepsize, however, the `rk38` codes are less expensive and this makes up for their reduced accuracy.
2. A striking difference with the standard linear algebra situation occurs when using Householder methods. There, different representations of the Householder vectors are chiefly a matter of storage efficiency. In our time de-

[4]e-mail to either author

pendent setting, however, Householder methods in the u-variables should be discarded: they are prone to instabilities, and this is clearly betrayed in a variable stepsize implementation. Instead, in the v-variables, integrated with a local projection step, and - especially - the w-variables, Householder methods are robust and also handle well large problems.

3. Givens methods based on the θ-variables revealed very accurate and efficient. However, the frequent need to compute sines and cosines and inverse trigonometric functions is a potential drawback with this approach. A careful implementation based on direct integration of the $(\cos, \sin)$ values has not been carried out, but could be an interesting endeavor.
4. The implementation in variable stepsize has proved very rewarding, especially since it confirmed our expectation that often most rejections occur far ahead of having computed all of Q.
5. It would be interesting to implement our methods by exploiting the rewriting (2.1), and hence solving (2.2) always starting near $\begin{bmatrix} I_p \\ 0 \end{bmatrix}$. This may prove beneficial for all methods, also those based on the u-variables.
6. With the present level of implementation, and all things considered, the methods based on θ and w variables are probably the best, followed by the v-methods.
7. The relative comparison with the projected integrator `prk45` appears favorable to our new codes. In particular, our codes generally require fewer steps, and are more accurate and less expensive. Moreover, the `vtdp5, vvdp5, vwdp5` codes never failed to complete the integration; instead, `prk45`, in spite of the certainly more sophisticated implementation of the integrator `RKF45`, was occasionally unable to complete the integration.

REFERENCES

[BGGS] G. Benettin, G. Galgani, L. Giorgilli, J. M. Strelcyn, *Lyapunov exponents for smooth dynamical systems and for Hamiltonian systems; a method for computing all of them. Part I: Theory. ... Part II: Numerical applications*, Meccanica **15** (1980), pp. 9–20,21–30.

[BR] T. Bridges and S. Reich, *Computing Lyapunov exponents on a Stiefel manifold*, preprint, University of Surrey, (2000).

[CIZ] M. P. Calvo, A. Iserles, and A. Zanna, *Numerical solution of isospectral flows*, Math. Comp. **66** (1997), pp. 1461–1486.

[C] M.T. Chu, *On the continuous realization of iterative processes*, SIAM Review **30** (1988), 375–387.

[CS] T.F. Coleman and D.C. Sorensen, *A note on the computation of and orthonormal basis for the null space of a matrix*, Mathematical Programming **29** (1984), 234–242.

[Da] Davey, A., *An Automatic Orthonormalization Method for Solving Stiff BVPs*, J. Comp. Phys. **51** (1983), pp. 343-356.

[DNT] P. Deift, T. Nanda, and C. Tomei, *Ordinary differential equations and the symmetric eigenvalue problem*, SIAM J. Numer. Anal. **20** (1983), no. 1, 1–22.

[DRV1] L. Dieci, R. D. Russell, E. S. Van Vleck, *Unitary Integrators and Applications to Continuous Orthonormalization Techniques*, SIAM J. Numer. Anal. **31** (1994), pp. 261-281.

[DRV2] L. Dieci, R. D. Russell, and E. S. Van Vleck, *On the computation of Lyapunov exponents for continuous dynamical systems*, SIAM J. Numer. Anal. **34** (1997), 402–423.

[DV1] L. Dieci and E. S. Van Vleck, *Computation of a few Lyapunov exponents for continuous and discrete dynamical systems*, Appld. Numer. Math. **17** (1995), 275–291.

[DV2] L. Dieci and E. Van Vleck, *Computation of orthonormal factors for fundamental solution matrices*, Numer. Math. **83** (1999), 599–620.

[DV3] L. Dieci and E. Van Vleck, *Continuous orthonormalization for linear two-point boundary value problems revisited*, IMA Volumes in Math. Appl. **118** (1999), 69–90.

[DLP] F. Diele, L. Lopez, and R. Peluso, *The Cayley transform in the numerical solution of unitary differential systems*, Adv. Comp. Math. **8** (1998), 317–334.

[ER] J. P. Eckmann, D. Ruelle, *Ergodic theory of chaos and strange attractors*, Rev. Mod. Phys. **57** (1985), 617–656.

[GSO] I. Goldhirsch, P. L. Sulem, and S. A. Orszag, *Stability and Lyapunov stability of dynamical systems: a differential approach and a numerical method*, Physica D **27** (1987), 311–337.

[GVL] G. H. Golub and C. F. Van Loan, *Matrix computations*, 2nd ed., The Johns Hopkins University Press, 1989.

[HL] E. Hairer, C. Lubich, *The life-span of backward error analysis for numerical integrators*, Numer. Math. **76** (1997), 441–462.

[HNW] E. Hairer, S. P. Nœrsett, and G. Wanner, *Solving ordinary differential equations I*, Springer-Verlag, Berlin-Heidelberg, 1993, Second edition.

[H] D. Higham, *Time-stepping and preserving orthonormality*, BIT **37** (1997), 24–36.

[MR] V. Mehrmann and W. Rath, *Numerical methods for the computation of analytic singular value decompositions*, Elec. Trans. Numer. Anal. **1** (1993), 72–88.

[Me] Meyer, G. H. *Continuous Orthonormalization for Boundary Value Problems*, J. Comp. Phys. **62** (1986), pp. 248-262.

[Mu] H. Munthe-Kaas, *Runge-Kutta methods on Lie groups*, BIT **38** (1998), 92–111.