Orthonormal Integrators Based on Householder and Givens Transformations ⋆

Luca Dieci [a] and Erik S. Van Vleck [b]

[a] *School of Mathematics, Georgia Tech, Atlanta, GA 30332 U.S.A.*

[b] *Department of Mathematics, University of Kansas, Lawrence, KS 66045 U.S.A.*

AMS(MOS) subject classifications. 65L

Abstract

We consider refined implementations of algorithms based on Householder and Givens transformations to find the Q-factor in the QR factorization of a matrix solution of linear time dependent differential systems. After discussing the algorithms, we introduce a suite of integrators, `QRINT`, and provide numerical testing to show the efficiency and accuracy of our techniques.

Key words: Householder and Givens transformations, orthonormal integrators.

1 Introduction

Recently there has been interest in techniques to integrate differential equations with orthonormal solutions; e.g., see [2,3,5,6]. Our interest in this work is in the approximation of the orthonormal factor Q in the QR-factorization of p columns of a matrix solution of an n-dimensional linear differential equation. An important application of such a time dependent factorization is that it allows for the computation of Lyapunov exponents; e.g., see [1,4].

In this paper we continue the development begun in [5] of methods for finding Q based upon continuous Householder reflectors and Givens rotations: we introduce new variables for the Householder reflectors, devise inexpensive checks to determine when a change of coordinates is required, and provide complete algorithms and a suite of code, `QRINT`, to find time dependent Householder reflectors and Givens rotations. This provides an efficient and accurate means

⋆ Work supported under NSF Grants #DMS-9973266 and #DMS-9973393.

of approximating the orthonormal matrix function Q that automatically preserves orthonormality of Q even in the more difficult case when $p < n$.

Consider the the initial value problem for $X \in \mathbb{R}^{n\times p}$, $p \leq n$:

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

where $A : \;\; t \to \mathbb{R}^{n\times n}$ is of class C^{k-1}, $k \geq 1$. Our aim is to efficiently and accurately approximate the time dependent factor $Q \in \mathbb{R}^{n\times p}$ in a QR factorization of $X(t)$. It is well understood that the QR-factorization of X is unique only up to a choice of signs for the diagonal of R. Once a fixed choice of signs for the diagonal of R is selected, the matrix valued functions Q and R exist and are as smooth as X. A common approach is to approximate Q (and R) by applying numerical integration schemes to the differential equation it satisfies. Let Q_0R_0 be the QR-factorization of X_0. To derive a differential equation for Q, 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:

$$\begin{cases} \dot{Q} & = AQ - QQ^TAQ + QS\,, \\ Q(t_0) & = Q_0\,, \end{cases} \quad S_{ij} = \begin{cases} (Q^TAQ)_{ij}, & i > j, \\ 0, & i = j, \\ -(Q^TAQ)_{ji}, & i < j. \end{cases} \tag{1}$$

In this work we take further the approach introduced in [5]: instead of solving (1) directly, we solve for elementary factors in the form of continuous Householder reflectors or Givens rotations that, in analogy with their use in linear algebra, may then be used to form Q. However, there are some important differences between the linear algebra context and the time dependent context we consider here. In particular, in the present context, the order in which Givens rotators are applied is of utmost importance in order to avoid singularities in the differential equations. Similarly, when using Householder reflectors, attention must be paid in order to avoid singularities in the resulting differential equations. We will clarify these aspects in Section 2.

In Section 2, we discuss the methods we put forward. In Section 3 we discuss numerical issues that we confront when implementing techniques based on these elementary transformations. We also give a new formulation in the Householder case. In Section 4, we present `QRINT`, a suite of `FORTRAN` integrators, and we illustrate performance on a number of examples. Conclusions are in Section 5.

We note here that our focus is on the case of coefficient matrices $A(\cdot)$ which are **dense**; that is, A has no particularly exploitable structure. In case A is structured (symmetric, banded, etc.) some computational savings ought to be possible, but these cases will be considered in future work.

2 Background and Householder and Givens representations

Many algorithms have been proposed for approximating the solution of (1). E.g., see [3,9,10,6,1,5]. Whichever technique and/or reformulation of the problem one chooses to adopt, in our mind the following are desirable properties to recover: (i) To obtain an orthonormal approximation, ideally not just at grid points, but everywhere desired, at no extra cost; (ii) To be able to handle without modifications both cases $p = n$ and $p < n$; (iii) To have a cost per step of $O(n^2 p)$–flops, and never of $O(n^3)$–flops when $p < n$; (iv) To be able to integrate the relevant differential equations with theoretical order restrictions given only by the degree of differentiability of Q; (v) To be able to efficiently proceed in adaptive step-size mode.

We can view the solution Q of (1) 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 we have to resolve. Naturally, the curve Q may be parametrized in many different ways, and the choice of parametrization turns out to be important from the numerical point of view. Furthermore, although the solution Q of (1) is a globally defined and smooth function, a representation of Q need not be unique nor globally smooth; we could choose to have a representation for Q on an interval $[t_0, t_1]$, another one on $[t_1, t_2]$, etc.. Each representation of Q must be smooth in the interval where is used, but may even fail to extend for all t. Finally, we should appreciate that if $Q_1 R_1$ and $Q_2 R_2$ are any two QR factorization of X at a given point, then $Q_2 = Q_1 D$, where D is a (p, p) diagonal matrix of ± 1, chosen so that the diagonal of DR_2 equals that of R_1 (and hence $DR_2 = R_1$).

In the present work, and in [5], our approach is to **represent** a matrix Q which gives **a** QR factorization of X as products of Householder or Givens transformations. We resolve the lack of uniqueness inherent in a QR factorization of X by selecting the signs for Householder reflectors in a certain way, or applying the rotators in a certain order. The precise way this is done will be explained below. Still, it should be kept in mind that our choices on selecting such signs and/or ordering are made so that the chosen representation can be formed in a numerically stable way, and these choices can be made without explicit knowledge of X (or R). It must be also realized that enforcing our criteria on the choice of signs for Householder reflectors might very well end up giving different factors Q on different subintervals: a trivial post-multiplication by a diagonal matrix of ± 1 will allow us to recover any Q we want; e.g., the one for which the diagonal of R is positive.

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 $Q^T(t) = Q_p(t)\cdots Q_1(t)$, with $Q_i(t) = Q_i^T(t)$, $i = 1\ldots, p$, the Householder matrices

$$Q_i(t) = \begin{bmatrix} I_{i-1} & 0 \\ 0 & P_i(t) \end{bmatrix}, \quad P_i(t) = I - 2v_i(t)v_i(t)^T, \ ||v_i||_2 = 1.$$

After $(i-1)$ transformations, the matrix X got transformed into $Q_{i-1}\ldots Q_1 X$ 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 Q_i. So, we will set

$$u_i(t) = x_i(t) - \sigma_i \|x_i\| e_1\,, \quad v_i(t) = \frac{u_i(t)}{\|u_i(t)\|}\,, \tag{2}$$

and continue the triangularization process. To make sure that no loss of precision occurs when forming (2), the textbook choice for σ_i is (see [7]):

$$\sigma_i := \begin{cases} -1, & \text{if } e_1^T x_i(t_k) \geq 0, \\ 1, & \text{if } e_1^T x_i(t_k) < 0. \end{cases} \tag{3}$$

In [5] we derived differential equations for the v_i's, which we now recall.

For simplicity, we omit the subindices, and thus use the notation v for v_i, etc., and also use A for $A(i:n, i:n)$, where the matrix A has been progressively modified by the accumulated transformations:

$$(A, Q_j)-\text{update}: \quad A(t) := Q_j(t)A(t)Q_j(t) - Q_j(t)\dot{Q}_j(t)\ , \quad j = 1, \ldots, i-1\ . \tag{4}$$

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 where $A_s = 1/2(A + A^T)$ and e_1 is the first unit vector (of appropriate dimension). Then, we have

$$\frac{d}{dt}\begin{bmatrix} (e_1^T v) \\ \hat{v} \end{bmatrix} = \begin{bmatrix} 0 & c^T - b^T \\ b - c & S - S^T \end{bmatrix} \begin{bmatrix} (e_1^T v) \\ \hat{v} \end{bmatrix}, \tag{5}$$

where we have set $b = \frac{2(e_1^T v)^2 - 1}{2}\hat{a}_1 + \hat{A}\hat{v}(e_1^T v)$, $c^T := (-\hat{a}_{1,a}^T \hat{v} + \alpha)\hat{v}^T$, $S = (\frac{2(e_1^T v)^2 - 1}{2(e_1^T v)}\hat{a}_1 + \hat{A}\hat{v})\hat{v}^T$ 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 (5) can be easily supplied with initial conditions at t_0 since X_0 is known, and we can find a Q_0 via Householder transformations (in the v formulation, with the σ's satisfying (3)). However, to describe the typical step between t_k and $t_{k+1} = t_k + h_k$ the expression (3) used for choosing the sign of σ must be modified since in practice we do not have $X(t_k)$. We enforce (3) as follows, by only keeping track of the transformations.

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

$$\sigma_i^{(k)} := \begin{cases} -1, & \text{if } \sigma_i^{(k-1)} e_1^T K_{i-1}^{(k)} P_i^{(k-1)} e_1 \geq 0\,, \\ +1, & \text{if } \sigma_i^{(k-1)} e_1^T K_{i-1}^{(k)} P_i^{(k-1)} e_1 < 0\,, \end{cases} \tag{6}$$

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

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

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

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

Thus, we need to find a 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 component of $v_i^{(k-1)}(t_k)$. Thus, we can prescribe new ICs for the $v_i^{(k)}(t_k)$. It should be stressed that "*the choice (6) is the same as (3), but (6) does not require knowledge of X*".

Remark 1 In linear algebra, see [7], the choice of signs as in (3) 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. To find the v_i's, we integrate (5) and in (5) there is a division by $e_1^T v_i$ when forming the vector $\frac{2(e_1^T v_i)^2 - 1}{2e_1^T v_i}\hat{v}$ of S. Since v_i has length one, we must make sure that $e_1^T v_i$ is not a small number. But, (6) is equivalent to having

$$(e_1^T v_i)^2 \geq \sum_{j=2}^{n-i+1} (e_j^T v_i)^2\,,\ i = 1, \ldots, p\,. \tag{7}$$

In particular, the vector $\frac{2(e_1^T v_i)^2-1}{2e_1^T v_i}\hat{v}$ is well scaled. Moreover, (7) can be used to decide if the current Householder frames are numerically stable or not.

Householder 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., the vectors $v_i^{(k-1)}$ at t_k), and the $\sigma_i^{(k-1)}$.

(1) For $i = 1, \ldots, p$, check to see if (7) holds true. If it fails, redefine $\sigma_i^{(k)}$ according to (6) and determine new $Q_i^{(k)}(t_k) = \begin{bmatrix} I_{i-1} & 0 \\ 0 & P_l^{(k)}(t_k) \end{bmatrix}$ accordingly (redefine $v_i^{(k)}(t_k)$), for $i = 1, \ldots, p$

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

(2) Let $A = A(i : n, i : n)$

(3) Find the Householder transformation $P_i^{(k)}(t)$ by integrating (5) on $[t_k, t_{k+1}]$

(4) Do an (A, Q_i) update (4)

- Endfor i.

OUTPUT: $Q^{(k)}(t_{k+1})^T = Q_p^{(k)}(t_{k+1}) \cdots Q_1^{(k)}(t_{k+1})$, 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 [7]), 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 key for numerical stability, and therefore for accuracy and efficiency.

Order of rotators. Our strategy is

$$\text{First rotator must annihilate largest entry of } x_i(2:n-i+1)\ . \tag{8}$$

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

$$l\ :\ X_{i+l-1,i} = \max_{2\le j\le n-i+1} |X_{i+j-1,i}|\,, \tag{9}$$

and define the index array π_i as

$$\pi_i = [1, l, 2, \ldots, l-1, l+1, \ldots, n-i+1]\,. \tag{10}$$

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

$$G_i \ = \ Q_{i,\pi_i(2)} \cdots Q_{i,\pi_i(n-i+1)}\,. \tag{11}$$

Remark 2 There seems to be no need to further refine (8) by selecting the second rotator to annihilate the largest entry of the unreduced part, etc..

In [5], 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

$$\begin{bmatrix} c_{\pi_i(3)} \cdots c_{\pi_i(n-i+1)}\ \frac{d}{dt}\theta_{\pi_i(2)} \\ c_{\pi_i(4)} \cdots c_{\pi_i(n-i+1)}\ \frac{d}{dt}\theta_{\pi_i(3)} \\ \vdots \\ c_{\pi_i(n-i+1)}\ \frac{d}{dt}\theta_{\pi_i(n-i)} \\ \frac{d}{dt}\theta_{\pi_i(n-i+1)} \end{bmatrix} = \begin{bmatrix} \alpha_{\pi_i(2)} \\ \alpha_{\pi_i(3)} \\ \vdots \\ \alpha_{\pi_i(n-i)} \\ \alpha_{\pi_i(n-i+1)} \end{bmatrix}\ . \tag{12}$$

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

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

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

$$(A, Q_j)-\text{update}:\ A(t) := Q_j^T(t) A(t) Q_j(t) - Q_j^T(t)\dot{Q}_j(t),\ j = 1, \ldots, i-1. \tag{13}$$

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 (12) it is simple to

write differential equations for the $(\cos, \sin)$ pairs directly:

$$\begin{bmatrix} c_{\pi_i(3)} \cdots c_{\pi_i(n-i+1)} \frac{d}{dt} \begin{bmatrix} c_{\pi_i(2)} \\ s_{\pi_i(2)} \end{bmatrix} \\ \vdots \\ c_{\pi_i(n-i+1)} \frac{d}{dt} \begin{bmatrix} c_{\pi_i(n-i)} \\ s_{\pi_i(n-i)} \end{bmatrix} \\ \frac{d}{dt} \begin{bmatrix} c_{\pi_i(n-i+1)} \\ s_{\pi_i(n-i+1)} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 & -\alpha_{\pi_i(2)} \\ \alpha_{\pi_i(2)} & 0 \end{bmatrix} \begin{bmatrix} c_{\pi_i(2)} \\ s_{\pi_i(2)} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} 0 & -\alpha_{\pi_i(n-i)} \\ \alpha_{\pi_i(n-i)} & 0 \end{bmatrix} \begin{bmatrix} c_{\pi_i(n-i)} \\ s_{\pi_i(n-i)} \end{bmatrix} \\ \begin{bmatrix} 0 & -\alpha_{\pi_i(n-i+1)} \\ \alpha_{\pi_i(n-i+1)} & 0 \end{bmatrix} \begin{bmatrix} c_{\pi_i(n-i+1)} \\ s_{\pi_i(n-i+1)} \end{bmatrix} \end{bmatrix}. \quad (14)$$

To supply initial conditions at t_0 for the differential equations (12) and/or (14), we enforce (8); that is, use (9), (10), and apply the rotators in the order specified by (11). 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.

To describe the typical step between t_k and $t_{k+1} = t_k + h_k$, the strategy just outlined, in particular the expressions (9) and (10), 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 3 Suppose we are triangularizing the i-th column, and, without loss of generality, assume that $\pi_i = [1, 2, \ldots, n-i+1]$ (if not, relabel the indices accordingly). In the differential equations (12) or (14), multiplication by the inverse of the diagonal matrix

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

is taking place. To avoid numerical instabilities, we would like to guarantee that the smallest diagonal entry (in absolute value) is away from 0; clearly, such entry is $c_3 \cdots c_{n-i+1}$. Now, 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

$$c_j^2 = \frac{(\prod_{l=1}^{j-1} x_{i,l}^2)}{(\prod_{l=1}^{j} x_{i,l}^2)}, \quad \text{so} \quad c_3^2 \cdots c_{n-i+1}^2 = \frac{(x_{i,1}^2 + x_{i,2}^2)}{\|x_i\|^2}.$$

Therefore, as long as

$$x_{i,1}^2 + x_{i,2}^2 \ \geq x_{i,j}^2,\ j = 3, \ldots, n-i+1\ ,\ i = 1, \ldots, p\,, \quad (15)$$

is satisfied, we have

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

Furthermore, one can see that (15) is equivalent to having

$$\prod_{j=3}^{k} c_j^2 \;\geq\; s_k^2 \;,\;\; k = 3, \ldots, n-i+1 \;,\;\; i = 1, \ldots, p \,, \tag{16}$$

and (16) 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 (9) and (10) after the first step. First of all, as long as (16) holds, we do not change the present ordering of the rotators. In case (16) fails, we enforce (8) as follows, by only keeping track of the transformations. 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

$$w_i := K_{i-1}^{(k)} G_i^{(k-1)}(t_k) e_1 \,, \;\; \text{for } i = 1, \ldots, p \,. \tag{17}$$

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 (10) and the ordering for the rotators relative to $G_i^{(k)}(t_k)$ as in (11). 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

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

It should be stressed that "*the choice just described for providing initial conditions on the rotators at t_k is equivalent to the strategy based on (11) and first paragraph after (14) but does not require knowledge of X*".

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 (16) 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 $\prod_{j=2}^{n-m+1}(Q^{(k)}_{i,\pi_i(j)}(t_k))^T w_i$, $m = n-1,\ldots,i$, 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 (12) or (14) on $[t_k, t_{k+1}]$.

(4) Do an (A, Q_i) update (13).

- 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 4 We observe that –even through a change of initial conditions– with our strategy the signs on the diagonal of R remain fixed when using rotators to find Q.

3 Implementation

Here we describe how we implemented the algorithms put forward in the previous section. Before doing so, however, we derive a different formulation for the Householder transformations.

Consider the new variable w:

$$w \;=\; \frac{1}{e_1^T v}\, v\,, \;\text{thus}\; w \;=\; \begin{bmatrix} 1 \\ \hat{w} \end{bmatrix}. \tag{18}$$

Observe that $(e_1^T v) \;=\; \frac{-\sigma}{\|w\|}$, so that the discussion relative to the choice of σ's (see (6)) 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, Q_i)-updates. Omitting the indices for simplicity, the form of the typical update becomes

$$\begin{aligned} QAQ - Q\dot{Q} \;&=\; \\ =\; A - \frac{2}{w^T w}(w(w^T A) + (Aw)w^T) + 4\frac{w^T A w}{(w^T w)^2} w w^T - \frac{2}{w^T w}(w\dot{w}^T - \dot{w}w^T)\;. \end{aligned} \tag{19}$$

From (19), it is easy to derive the differential equation satisfied by $\hat{w}$. In fact, this can be obtained from the requirement that $(QAQ - Q\dot{Q})e_1$ has only its first entry not 0. Using the form of $\dot{w}$, and the same notation used to derive

(5), this requirement becomes

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

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

$$\frac{d\hat{w}}{dt} \;=\; [a_{11} + \hat{w}^T \hat{a}_1 - 2\frac{w^T A w}{w^T w}]\hat{w} + (1 - \frac{w^T w}{2})\hat{a}_1 + \hat{A}\hat{w}\,. \tag{20}$$

Remark 5 Division by $w^T w$ in (20) is perfectly safe, given the form of w. Of course, if one uses the w-variables, obvious modifications are required in the skeleton of the algorithm on page 6. E.g., (7) now will read

$$1 - (w_{i+1,i}^2 + \ldots + w_{n,i}^2) \;\geq\; 0\,. \tag{21}$$

To summarize, for Householder methods, we have considered two possibilities: (i) v-variables, and (ii) w-variables. In the v-variables, we need to integrate (5) whose solution has norm 1 for all t, and we must maintain this property under discretization, at gridpoints. We simply integrate (5) 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 $O(n^2 p)$–flops per step. As far as the integration for the w-variables, this can be carried out with explicit schemes, with no need of renormalization.

As far as methods based on Givens transformations, we worked on obtaining good codes only for the formulation in terms of the θ-variables, see (12), and not for the $(\cos, \sin)$-variables. Naturally, it is simpler to work with the θ-variables, and then form the rotators. But, there is also an accuracy reason to prefer the θ-variables, since 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 6 *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 the general case of $Q \in \mathbb{R}^{n \times p}$, with notation similar to Lemma 6, writing Q as product of rotators, and using Lemma 6 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 7 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 too large and cause numerical difficulties, we always renormalize their values to $[-\pi, \pi]$, which we do by computing inverse tangents.

Remark 8 Householder methods based on the w-variables, and Givens' methods based on the θ-variables parametrize Q using exactly $p\frac{2n-p-1}{2}$ parameters: the minimal number required.

`QRINT`. We wrote a suite of `FORTRAN` codes, `QRINT`, which approximate Q by exploiting its representation in terms of Householder or Givens transformations. These codes are public and we encourage others to experiment with them. [1] Briefly, in `QRINT` the user specifies which formulation is desired, which integration formulas to use, variable or fixed stepsize integration, and `OUTPUT` specifications. The rest is carried out according to the previous exposition with some care for efficiency. We next outline some of the choices adopted in `QRINT`, more extensive documentation can be found in the interface to the codes.

Discretization Schemes. In `QRINT`, two explicit integrators of Runge-Kutta type are adopted as the basic schemes. The range of accuracy of interest is between 10^{-2} and 10^{-10}, and our schemes have been chosen with this accuracy demands in mind. We implemented 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 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; see [8]. Notice that with our schemes it is a trivial matter to form orthogonal approximations also at the internal RK points, not just at the grid points, at negligible extra cost.

Updates. To perform the updates (4) and (13), no extra function evaluations are needed, since `QRINT` stores and uses the values obtained at the Runge-Kutta stages.

Error Control. For variable time-stepping integration, we adopted the standard mixed absolute/relative error control of [8, II-4] with some slightly different heuristics. But, the most relevant changes to the standard strategies are due to the nature of our methods, which triangularize one column of X at the time: (i) we decide on step-size changes by monitoring the behavior of the error on all columns independently, and then take the most conservative estimate for the next step; (ii) there is no need to complete the entire integration step

[1] `www.math.gatech.edu/~dieci` or `www.math.ukans.edu/~evanvleck`

for Q 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.

Changes of signs or orderings. We have used the construction based on (6), or (17), and following discussion, only if the tests (7), or (21), or (16), for the 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

In the examples, we will refer to the performance of several options which are available in `QRINT`, by using the following first letter convention: `v`=v-variables, `w`=w-variables, `t`=θ-variables.

- Fixed stepsize codes.
  - 3/8th rule: `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: `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: `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: `vvdp5, vwdp5, vtdp5`.

All runs of which below have been made on a workstation by using the `f77` compiler with four optimization options: (1) No optimization, (2) `O1`, (3) `O3`, and (4) `O3` plus the options `fast-math, unroll-loops, expensive-optimizations`. In our tables, we report on the following measures of performance:

- `err`: the error between computed and exact Q, if the exact Q is known.
- `reimb`: the number of *reimbeddings* needed to complete a given run. Here, a reimbedding is a change of signs for Householder reflectors or a change of orderings for Givens rotations. We increment the counter every time any such change is performed.
- `rejs/first`: the total number of rejections (in variable stepsize mode) followed by the rejections occurring while triangularizing the first column.
- `cpu`: the `CPU` time needed to complete a given run normalized to 1 for the fastest run on the given problem. Here, the `CPU` time is the average of the times obtained with the four different optimization options.
- `nsteps`: the total number of steps taken (in variable stepsize mode).

For the variable stepsize codes, we also include comparison with a tried and true projected integrator, `prk45`, which integrates (1) 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 solver of order 5/4 whose performance is comparable with the Dormand-Prince 5/4 pair we adopted here.

Example 9 This is a problem chosen because the underlying fundamental solution is both exponentially dichotomic and fast rotating. 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 $[0, 10]$. 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. For the `t` methods there is only one equation to integrate:

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

which has the exact solution $\theta(t) = \alpha t$. So, one may expect no error while integrating for θ. However, we automatically renormalize angles to $[-\pi, \pi]$ and this causes roundoff errors to enter in the picture.

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). Observe that `prk45` is considerably more expensive than the competing 5/4 codes (`vvdp5, vwdp5`).

Table 4.1. Example 9: fixed stepsize $\Delta t = 1.E-3$.

Meth	err	reimb	cpu
tdp5	$2.4E-13$	0	15.0
trk38	$3.4E-13$	0	11.0
vdp5	$2.5E-9$	318	16.0
vrk38	$1.6E-6$	318	12.5
wdp5	$3.9E-8$	318	14.0
wrk38	$2.4E-6$	318	11.0

Table 4.2. Example 9: variable stepsize, tol= $1.E-8$.					
Meth	err	reimb	rejs	cpu	nsteps
prk45	$1.4E-8$	–	–	22.5	20803
vtdp5	$3.8E-8$	0	172	1	596
vtrk38	$1.5E-8$	0	155	1	695
vvdp5	$3.4E-9$	318	0	17.0	9535
vvrk38	$7.3E-9$	318	0	51.0	37883
vwdp5	$4.2E-9$	318	637	17.0	10821
vwrk38	$6.3E-9$	318	1380	36.0	31293

Example 10 This example was chosen because it exemplifies a class of systems of arbitrarily large dimension with no particularly exploitable sparsity structure in the function $A(t)$. We consider linearization about a traveling wave solution of the parabolic Nagumo equation

$$u_t = \epsilon^2 u_{xx} - f(u),\ x \in I\!R, t \geq 0,\ f(u) = u(u-1)(u-a),\ a \in (0,1). \quad (22)$$

The traveling wave ansatz $u_{TW}(x,t) = \phi(x-ct)$ gives a 2nd order boundary value problem with exact solution (unique up to translation)

$$\begin{cases} \phi(\xi) = \frac{1}{2}[1 \mp \tanh(\sqrt{\frac{1}{8\epsilon^2}} \cdot \xi)] \\ \quad c = \pm(1-2a)\sqrt{\frac{\epsilon^2}{2}}\ . \end{cases} \quad (23)$$

If we linearize (22) about (23) we obtain the linear partial differential equation

$$\eta_t = \epsilon^2 \eta_{xx} - f'(u_{TW}(x,t))\eta\,. \quad (24)$$

To obtain a linear system of ordinary differential equations from this PDE, we take x in the truncated interval $[-1,+1]$, impose periodic boundary conditions, and consider a Fourier collocation approach. Thus, for the grid points $x_j = -1 + 2(j-1)/n\ ,\ j = 1,\ldots,n$, we seek an approximate solution $z(t)$ by requiring that (24) is satisfied at the grid points:

$$[z_t = \epsilon^2 \partial_{xx} z - f'(u_{TW}(x,t))z]_{x=x_j}\,,\ j = 1,\ldots,n\,.$$

We replace ∂_{xx} with the spectral approximation $A = F^{-1}DF$ where F and F^{-1} are the forward and backward Fourier transforms, respectively, and D is the appropriate diagonal matrix containing the eigenvalues of ∂_{xx}, and we end up with the following system for $Z = (z(x_1,t),\ldots,z(x_n,t))^T$:

$$\frac{dZ}{dt} = A_D(t)Z\ ,\ A_D(t) = \epsilon^2 A - \mathrm{diag}(f'(u_{TW}(x_j,t)),\ j = 1,\ldots,n\ )\,. \quad (25)$$

For our experiments we set the wave speed $c = 10^{-1}$, the detuning parameter $a = 9/16$, and the diffusion coefficient $\epsilon^2 = 1.28$. We integrate over the interval $[0, 10]$. Results are tabulated in Table 4.3. Exact solution is not known, so we measure the "error" by comparing the results at $t = 10$ of the different methods against those obtained with `TOL` $= 1.E-12$ (using `vwdp5` and `vtdp5` there is agreement to 13 digits); this "error" we list as `errd` in Table 4.3. We note that `prk45` failed with `iflag = 6` (requested accuracy could not be achieved using smallest allowable stepsize) for $(n, p) = (32, 4)$. The `w`-variables consistently proved most efficient for this problem. Interestingly, for the different runs made with the `t,v,w` methods, all rejections occurred in the first column except for the run with for $n = 32$ in which all except 14 for `vtdp5`, 37 for `vtrk38`, 22 for `vvdp5` and 33 for `vvrk38` occurred in the first column. For this particular example, the `rk38` codes performed extremely well with `vwrk38` easily performing most efficiently. Finally, we observe the large number of rejections of most codes for the case $(n, p) = (32, 4)$; we suspect that this is due to increased stiffness in the relevant differential equations, but remain baffled as to why no rejections occurred when using `vwrk38`.

Table 4.3. Example 10: variable stepsize; `tol`$= 1.E-6$.

Meth	n/p	errd	reimb	rejs	cpu	nsteps
prk45	8/8	$1.46E-6$	-	-	1.4	574
vtdp5	8/8	$3.33E-7$	4	143	1.9	640
vtrk38	8/8	$4.35E-7$	4	149	1.8	807
vvdp5	8/8	$1.00E-6$	2	145	1.6	630
vvrk38	8/8	$5.39E-7$	2	151	1.3	785
vwdp5	8/8	$2.32E-7$	2	0	1.1	641
vwrk38	8/8	$4.14E-7$	2	0	1	798
prk45	32/4	-	-	-	-	-
vtdp5	32/4	$4.93E-7$	25	2398	438.8	9747
vtrk38	32/4	$1.34E-6$	27	2512	366.2	11714
vvdp5	32/4	$3.79E-7$	2	2402	279.8	9716
vvrk38	32/4	$2.46E-6$	2	2525	227.1	11644
vwdp5	32/4	$6.65E-7$	2	2249	219.8	9754
vwrk38	32/4	$1.74E-6$	2	0	169.1	11739

5 Conclusions

The purpose of this work has been two-fold: (i) To give new formulations and new algorithms for methods based on Householder and Givens transformations; (ii) To present `QRINT`, a suite of `FORTRAN` codes for finding Q by our techniques. We believe that the methods put forward in this work, and their implementation as laid down here, are a sensible way to find Q, and should

be used by people interested in comparative performance of other techniques. It is for this reason which we took up the task to write `QRINT`, and trust that it will prove useful to others. We should stress that we make no claim as far as our codes –nor our techniques, for what matters– being superior to other implementations.

While there is certainly room for improving `QRINT`, we believe that it is sophisticated enough that some conclusions about the relative merits of the techniques examined herein can be drawn.

(1) In variable stepsize the `dp5` codes generally outperform the `rk38` codes. The implementation in variable stepsize has proved very rewarding, since most rejections occur far ahead of having computed all of Q. In constant stepsize, however, the `rk38` codes are less expensive and this makes up for their lower order.
(2) Givens methods based on the θ-variables are accurate and efficient on problems of small size, but become less efficient than the Householder codes for larger problems because of the overhead associated to the frequent evaluations of trigonometric functions.
(3) With the present level of implementation, and all things considered, the methods based on the w variables are probably the best, followed by the θ methods and the v-methods in this order.
(4) Since our approaches do not integrate (1) directly, it is quite possible that at times our methods will outperform methods based on direct integration of (1), while on other occasions the opposite will be observed. On the tests presented, our methods seem generally superior to integration of (1) in that comparison with the projected integrator `prk45` appears favorable to our new codes. On the other hand, there are situations where `prk45` performs better than our codes: one such case is Example 5.4 of [5], where `prk45` is more accurate and less expensive.

References

[1] T. Bridges and S. Reich, *Computing Lyapunov exponents on a Stiefel manifold*, Physica D **156** (2001), pp. 219–238.

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

[3] 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.

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

pp. 275–291.

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

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

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

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

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

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