*Journal of Dynamics and Differential Equations, Vol. 14, No. 3, July 2002 (© 2002)*

# Jacobian Free Computation of Lyapunov Exponents

**Luca Dieci**[1]

*Received January 16, 2002*

The purpose of this paper is to present new algorithms to approximate Lyapunov exponents of nonlinear differential equations, without using Jacobian matrices. We first derive first order methods for both continuous and discrete QR approaches, and then second order methods. Numerical testing is given, showing considerable savings with respect to existing implementations.

**KEY WORDS:** Lyapunov exponents; large systems; Jacobian free; continuous QR; discrete QR.



## 1. INTRODUCTION

Lyapunov exponents—LEs, for short—are a common tool to explore stability properties of nonlinear differential equations, and provide valuable information on the statistical properties of the system under study. However, existing techniques to approximate LES are expensive, and this fact may have precluded extensive use of the LES in large dimensional problems of practical interest, such as those arising in biology, molecular dynamics, or spatially discretized time dependent partial differential equations.

An excerpt from the recent review article [5]: *For small scale problems, numerical methods are well developed for "long time" dynamical processes, for example Lyapunov exponents. A major area for future work is to extend these techniques to large scale problems.* The present work is a step in that direction.

Unarguably, the computational bottleneck of existing techniques to approximate LEs of nonlinear differential equations is the need to evaluate (and store) the linearized vector field, the Jacobian, and to perform matrix vector multiplications with the Jacobian. For large problems, this may be

[1] School of Mathematics, Georgia Tech, Atlanta, Georgia 30332. E-mail: dieci@math.gatech.edu



1040-7294/02/0700-0697/0 

impossible, or impractical, or just too inefficient. It is our purpose in this paper to provide new algorithms to approximate LEs which bypass the need for the Jacobian, and which are shown to be far less expensive than existing techniques.

In the rest of this Introduction we review basic concepts and algorithms. In Section 2 we propose Jacobian free discrete and continuous QR first order methods to approximate the LEs. In Section 3 we give second order methods for both discrete and continuous QR approaches, and point out how generalizations to higher order techniques may be made. In Section 4 we discuss implementation issues and show numerical performance of the new methods on a couple of examples.

Given the differential equation

$$\dot{x} = f(x), \qquad x(0) = x_0, \tag{1.1}$$

the LEs are a characterization of the asymptotic properties of the solution $x(t, x_0)$ via analysis of the linearized problem (for ease of notation, the dependence of the solution on $x_0$ is suppressed)

$$\dot{y} = f_x(x(t))\, y. \tag{1.2}$$

Formally, the LEs associated to (1.2) are defined as follows. Let $\{p_i\}$ be the columns of an initial conditions (full rank) matrix $Y_0$, and define the numbers $\lambda_i$, $i = 1,\ldots, n$,

$$\lambda_i(p_i) = \limsup_{t\to\infty} \frac{1}{t} \log \|Y(t)\, p_i\|, \tag{1.3}$$

where $Y(t)$ is the solution of

$$\dot{Y} = f_x(x(t))\, Y, \qquad Y(0) = Y_0. \tag{1.4}$$

When the sum of these numbers is minimized as we vary over all possible ICs (initial conditions) $Y_0$, the numbers are called Lyapunov exponents of the system.

Naturally, there are $n$ LEs, $\{\lambda_i\}$, counted with their multiplicity, for a given $n$-dimensional system. However, in many circumstances, one does not need to approximate all $n$ LEs of a system. Often, only the $p$ most dominant ones are needed (and $p$ can be much smaller than $n$). For example, see [12], in order to approximate the entropy of a system only all the positive LEs are needed, and to estimate the dimension of an attractor only the most dominant (positive and negative) having positive sum are needed. Also, a commonly adopted criterion to assess chaotic dynamics on an

attractor rests upon detection of a positive Lyapunov exponents, and thus only the largest LE needs to be tracked. Finally, there are situations where one knows before hand that the LEs enjoy special symmetries (see [15, 11], and also [7]), which reduce the number of LEs one needs to approximate. From the practical point of view, in case one needs only the $p$ most dominant LEs of the system, then the matrix $Y_0$ in (1.4) is made up by just $p$ columns, which are typically chosen to be $[\frac{I_p}{0}]$. With this in mind, in (1.4), $Y: t \to \mathbb{R}^{n \times p}$.

To approximate the LEs, the most widely adopted techniques rest on the QR factorization of the solution $Y(t)$ of (1.4)

$$Y(t) = Q(t)\, R(t),$$

where $Q$ and $R$ are as smooth as $Y$, $Q: t \to \mathbb{R}^{n \times p}$ is orthogonal, $Q^T Q = I$ for all $t$, and $R: t \to \mathbb{R}^{p \times p}$ is upper triangular with positive diagonal entries. Then, one extract approximation to the LEs as time averages of the logarithms of the diagonal of $R$:

$$\lambda_i = \lim_{t \to \infty} \frac{1}{t} \log(R_{ii}(t)), \qquad i = 1,\ldots, p. \tag{1.5}$$

The existence of smooth factors in the QR factorization of a fundamental matrix solution has been known at least since Perron, [21], and the role of the QR factorization in the study of Lyapunov exponents has also been known for a long time; see [17]. From the numerical point of view, there are two ways in which the QR factorization of $Y$ is traditionally found, and these lead to so-called discrete and continuous QR approaches, respectively. We recall them next.

*Discrete QR approach*: [2, 11, 13, 22]. Say we want the QR factorization of $Y(t_{k+1}) \in \mathbb{R}^{n \times p}$. With $t_0 = 0$, the idea is to write $Y(t_{k+1})$ as composition of transition matrices:

$$Y(t_{k+1}) = Y(t_{k+1}, t_k)\, Y(t_k, t_{k-1}) \cdots Y(t_2, t_1)\, Y(t_1, 0)\, Y_0,$$

where $Y(t, t_j)$, $j = 0,\ldots, k$, is the solution of

$$\begin{cases} \dot{Y}(t, t_j) = f_x(x(t))\, Y(t, t_j), & t_j \leqslant t \leqslant t_{j+1} \\ Y(t_j, t_j) = I_n. \end{cases}$$

Now, let $Y_0 = Q_0 R_0$, and progressively update the QR factorizations as

$$Y(t_{j+1}, t_j)\, Q_j = Q_{j+1} R_{j+1}, \qquad j = 0,\ldots, k,$$

so that

$$Y(t_{k+1}) = Q_{k+1}[R_{k+1}R_k \cdots R_1 R_0] \tag{1.6}$$

gives the sought QR factorization of $Y(t_k+1)$. In (1.6), $Q_{k+1} \in \mathbb{R}^{n\times p}$ and $\prod_{j=k+1}^{0} R_j \in \mathbb{R}^{p\times p}$.

Of course, there is no need to compute the full transition matrices $Y(t_{j+1}, t_j)$, and the following compact reformulation must be preferred. With above notation, we let $Y_{j+1}(t) = Y(t, t_j)\, Q_j$, $t \in [t_j, t_{j+1}] \to \mathbb{R}^{n\times p}$. For $j = 0,\ldots, k$, we solve

$$\begin{cases} \dot{Y}_{j+1} = f_x(x(t))\, Y_{j+1}, & t_j \leqslant t \leqslant t_{j+1} \\ Y_{j+1}(t_j) = Q_j, \end{cases} \tag{1.7}$$

and then let

$$Y_{j+1}(t_{j+1}) = Q_{j+1}R_{j+1}, \tag{1.8}$$

leading to (1.6) as before.

*Continuous QR approach*: [3, 8, 10, 13]. Here one derives—and integrates—the differential equations governing the evolution of the $Q$ and $R$ factors in the QR factorization of $Y$. Differentiating the relation $Y = QR$ and using (1.4) one gets $\dot{Q}R + Q\dot{R} = f_x(x(t))\, QR$, and multiplying by $Q^T$ on the left one gets the equation for $R$:

$$\dot{R} = (Q^T f_x(x(t))\, Q - S)\, R, \qquad R(0) = R_0, \tag{1.9}$$

where we have set $S := Q^T\dot{Q}$. Observe that since $Q^TQ = I$, then $S$ is skew-symmetric. Further, since $R$ is triangular, then the coefficient $Q^T f_x(x(t))\, Q - S$ in (1.9) must be upper triangular. This fact, and skew-symmetry, then give the following form for $S$:

$$S_{ij} = \begin{cases} (Q^T(t)\, f_x(x(t))\, Q(t))_{ij}, & i > j, \\ 0, & i = j, \\ -S_{ji}, & i < j. \end{cases} \tag{1.10}$$

Next, multiplying $\dot{Q}R + Q\dot{R} = f_x(x(t))\, QR$ by $R^{-1}$ on the right, and using (1.9), we get the differential equation for $Q$:

$$\dot{Q} = (I - QQ^T)\, f_x(x(t))\, Q + QS, \qquad Q(0) = Q_0. \tag{1.11}$$

Finally, observe that no explicit integration for $R$ needs to be done in order to approximate the LEs. In fact, from (1.9) and (1.5), one has

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} \int_0^t (Q^T(s)\, f_x(x(s))\, Q(s))_{ii}\, ds, \qquad i = 1,\ldots, p. \tag{1.12}$$

From (1.12), we further define the new functions (as in [10]) for $i = 1,\ldots, p$,

$$v_i(t) = \int_0^t (Q^T(s)\, f_x(x(s))\, Q(s))_{ii}\, ds \quad \text{so that} \quad \dot{v}_i = (Q^T(t)\, f_x(x(t))\, Q(t))_{ii}, \tag{1.13}$$

and these can be "integrated" directly along with (1.11). [Of course, the result is a quadrature rule on (1.12)].

**Remark 1.1.** Of course, the original differential equation (1.1) must be integrated along with (1.7) or (1.11) and the needed values for $f_x(\cdot)$ must be supplied at the appropriate order of accuracy. The simplest way to do this in practice is to use the same integration rule for all differential equations involved, say the same Runge–Kutta scheme.

**Remark 1.2.** In general, the LEs of (1.1) will depend on the initial conditions $x_0$. Further, it is generally not clear why the limits in (1.5) exist (and equal the LEs) and to what extent they depend on the initial conditions $Y_0$ for (1.4). Extensive theoretical studies address these concerns. For example, use of limits in (1.5) is justified for so-called *regular* systems (see [1]), which further are prevalent in a measure theoretic sense; see [19] and [20]. The dependence of the LEs upon the initial conditions is the domain of ergodic theory, and we again refer to the work of Oseledec, [20], for statements in this case.

**Remark 1.3.** There is numerical and theoretical evidence (e.g., see [8]) that the continuous QR approach is preferable to the discrete QR approach. While this is generally true, a couple of considerations are in order. First, the analysis showing the shortcoming of the discrete QR approach in [8] highlights that difficulties are chiefly caused by the QR factorization at the end of each step, and affect the (large) negative LEs; thus, if only a few dominant LEs are needed, this should not be a concern. Second, the numerical evidence highlights that controlling the local error (for either the Q-factor or the transition matrix itself) while integrating (1.7) tends to require (much) smaller stepsizes than for the continuous QR approach. This is certainly a concern—and a true drawback—in case the

original problem is linear (i.e., (1.1) is really $\dot{x} = A(t)\, x$, and thus $f_x(x(t))$ in (1.7) is just $A(t)$), because no other simple mechanism of error control is in place if one wants to approximate the LEs. However, for nonlinear problems, controlling the error on the trajectory should enforce error control on the linearized problem as well, and no further need to control local errors while integrating (1.7) ought to be required.

## 2. JACOBIAN FREE EXPONENTS: 1ST ORDER METHODS

Here we present our approach in the simplest setting possible. This will facilitate understanding of the simple idea we exploited, and will lead to appropriate ways to generalize to higher order methods. We again find it convenient to separate between discrete and continuous QR approaches.

**Remark 2.1.** We call our methods "Jacobian free" in analogy with so-called *matrix free* methods which are used in implicit schemes for stiff systems of differential equations to bypass the need for a formal Newton iteration (e.g., see [4, 6]). In that context, however, the "matrix free" formulation is conceptually only a means to solve the nonlinear system, and has no impact on the order of the scheme used, nor on the formulation itself (i.e., the scheme itself is not changed). Instead, as we will clarify below, our "Jacobian free" reformulation effectively modifies the scheme one is using and may alter its order if one is not careful.

### 2.1. Discrete QR: Forward Euler

Suppose we use the forward Euler method with step $h$ to approximate (1.7). Thus, the basic step is

$$Y_{j+1} = Q_j + hf_x(x_j)\, Q_j, \qquad \text{then} \quad Y_{j+1} = Q_{j+1} R_{j+1}. \tag{2.1}$$

Here, $x_j$ is the approximation to $x(t_j)$ which may have been obtained with forward Euler method or any other method of order at least one. Clearly, the resulting method is of first order of accuracy (second order locally). The expensive part of this scheme is given by the need for $f_x(x_j)$ and the matrix multiplication $f_x(x_j)\, Q_j$. To avoid these, we reason as follows.

Let $Q_j = [q_1^j, \ldots, q_p^j]$. Then, for $k = 1, \ldots, p$, we approximate the action $hf_x(x_j)\, q_k^j$ as

$$hf_x(x_j)\, q_k^j \approx f(x_j + hq_k^j) - f(x_j), \qquad k = 1, \ldots, p.$$

The resulting scheme becomes

$$Z_{j+1} = Q_j + B_j, \qquad B_j := [f(x_j + hq_1^j) - f(x_j), \ldots, f(x_j + hq_p^j) - f(x_j)],$$

$$\text{then} \quad Z_{j+1} = Q_{j+1} R_{j+1}. \tag{2.2}$$

**Expense Comparison**. We delay until Section 4 a careful comparison of the cost of each scheme. Momentarily, we observe that computing $Z_{j+1}$ from (2.2) rather than $Y_{j+1}$ from (2.1) avoids the Jacobian evaluation and the multiplication $f_x(x_j)\, Q_j$.

Upon obtaining $R_{j+1}$, we can update the approximation of the LEs. If $\lambda_i^j$, $i = 1, \ldots, p$, are the approximation of the LEs at $t_j$, then from (1.5) and (1.6) the approximate values $\lambda_i^{j+1}$ at $t_{j+1}$ are easily obtained as

$$\lambda_i^{j+1} = \frac{t_j}{t_j + h} \lambda_i^j + \frac{1}{t_j + h} \log(R_{j+1})_{ii}, \qquad i = 1, \ldots, p. \tag{2.3}$$

It is trivial to assess the error caused by the Jacobian free replacement, and to appreciate that (2.2) is a first order scheme. However, since we will need the explicit error expression in the next section, we now derive it. Because of linearity of the problem (1.7), it suffices to look at the error on a single column. Further, we look at the local error on a single step, and thus can consider the first step. So, let $y_0$ be the initial condition (that is, $y_0$ is any of the $q_k^0$, $k = 1, \ldots, p$). For later use, recall that the local error for (2.1) is given by

$$y(t_1) - y_1 = \tfrac{1}{2} h^2 \ddot{y}_0 + O(h^3). \tag{2.4}$$

For (2.2), instead, we have

$$f(x_0 + hy_0) - f(x_0) = h f_x(x_0) + \tfrac{1}{2} h^2 f_{xx}(x_0)(y_0, y_0) + O(h^3),$$

so that

$$z_1 = y_1 + \tfrac{1}{2} h^2 f_{xx}(x_0)(y_0, y_0) + O(h^3)$$

and thus

$$y(t_1) - z_1 = \tfrac{1}{2} h^2 \ddot{y}_0 - \tfrac{1}{2} h^2 f_{xx}(x_0)(y_0, y_0) + O(h^3). \tag{2.5}$$

Therefore, (2.2) and (2.3) is a first order scheme.

### 2.2. Continuous QR: Forward Euler

The issue here is to integrate (1.11) and approximate the integral in (1.12). Integration of (1.11) has received a lot of attention in recent times (e.g., see [3] and [9]), and sophisticated choices exist for this task. All of these choices can be applied in the present context, but for the sake of simplicity here we consider the simplest possibility: integrate (1.11) with a forward Euler step $h$ and then orthogonalize the solution (in the current terminology, a first order projected integrator). Orthogonalization is handily carried out by the QR factorization (though the polar factorization may also be used). In other words, the basic step is

$$P_1 = Q_0 + h[f_x(x_0)\, Q_0 - Q_0(Q_0^T f_x(x_0)\, Q_0 - S_0)], \qquad \text{then} \quad Q_1 : P_1 = Q_1 R_1. \tag{2.6}$$

Clearly, the resulting method is of first order of accuracy (second order locally), and the expensive part is again given by the need for $f_x(x_0)$ and the matrix multiplication $f_x(x_0)\, Q_0$. The same reasoning as in the discrete QR case can however be applied.

So, if $Q_0 = [q_1^0, \ldots, q_p^0]$, we approximate the action $hf_x(x_0)\, Q_0$ as

$$hf_x(x_0)\, Q_0 \approx B_0 := [f(x_0 + hq_1^0) - f(x_0), \ldots, f(x_0 + hq_p^0) - f(x_0)],$$

and further use this approximation in (2.6) also to approximate $S_0$ (recall (1.10)); in other words, we form $Q_0^T B_0$ and use it to approximate $hS_0$, call it $H_0$, which is thus defined as $(H_0)_{ij} = (Q_0^T B_0)_{ij}$, $i > j$, and by skew-symmetry. The resulting scheme becomes

$$V_1 = Q_0 + B_0 - Q_0(Q_0^T B_0 - H_0), \qquad \text{then} \quad Q_1 : V_1 = Q_1 R_1. \tag{2.7}$$

It is again immediate to appreciate that (2.7) is a first order scheme. To approximate the LEs, one can replace the integral in (1.12) by a simple quadrature rule. Since we are using the scheme (2.7) to approximate $Q$, a rectangle rule is adequate (i.e., a forward Euler step for (1.13)). That is, if $\lambda_i^j$, $i = 1, \ldots, p$, are the approximation of the LEs at $t_j$, then from (1.12) we update the approximations at $t_{j+1}$ as

$$\lambda_i^{j+1} = \frac{t_j}{t_j + h}\lambda_i^j + \frac{1}{t_j + h}(Q_j^T B_j)_{ii}, \qquad i = 1, \ldots, p, \tag{2.8}$$

and obtain a first order approximation for the LEs.

Computing $V_1$ from (2.7), rather than $P_1$ from (2.6), avoids the Jacobian evaluation and the multiplication $f_x(x_0)\, Q_0$.

## 3. SECOND ORDER METHODS

Often, the LEs are used to infer qualitative properties of the differential system (1.1) and high accuracy approximation of the LEs may not be really needed. However, there are situations where one needs more accurate approximations than those delivered by the first order schemes of the previous section.[2] Here we extend the simple methods of the previous section to second order techniques, and point out how to generalize to higher order.

In the literature of numerical differential equations there are many ways to produce high order schemes (see [16]). We consider two of them here: second order Runge–Kutta (RK) schemes, and extrapolation. As before, we differentiate between discrete and continuous QR approaches.

### 3.1. Discrete QR: 2nd Order

We look at two ways to obtain second order schemes: (a) using the explicit midpoint rule, or (b) extrapolating forward Euler approximations. Neither of these choices can be implemented naively in a Jacobian free way, and some care must be paid in order to obtain order 2.

**(a) Explicit Midpoint Rule.** The basic scheme on one step $h$ from $Q_0$ to $Q_1$ is

$$\begin{aligned} Y_{1/2} &= Q_0 + \frac{h}{2} f_x(x_0)\, Q_0, \\ Y_1 &= Q_0 + h f_x(x_{1/2})\, Y_{1/2}, \qquad \text{then} \quad Q_1 \colon Y_1 = Q_1 R_1. \end{aligned} \tag{3.1}$$

Above, $x_0$ and $x_{1/2}$ are approximations to the solution of (1.1) which may have been obtained by the midpoint rule, so that $x_{1/2} = x_0 + \frac{h}{2} f(x_0)$, or also by some other scheme, as long as they have the appropriate order (e.g., we can use $x_{1/2} = 1/2(x_0 + x_1)$ if we are using a high order scheme to integrate (1.1)). Naturally, (3.1) is a 2nd order scheme ([16]).

To make (3.1) a Jacobian free method, we must approximate the actions $f_x(x_0)\, Q_0$ and $f_x(x_{1/2})\, Y_{1/2}$ by appropriate directional derivatives. For $f_x(x_0)\, Q_0$, this is as before, since an $O(h^2)$ approximation to $Y_{1/2}$ is all that is needed. Thus, if $Q_0 = [q_1^0, \ldots, q_p^0]$, we let

$$h f_x(x_0)\, Q_0 \approx B_0 := \left[ f\left(x_0 + \frac{h}{2} q_1^0\right) - f(x_0), \ldots, f\left(x_0 + \frac{h}{2} q_p^0\right) - f(x_0) \right],$$

[2] Still, we stress that the trajectory itself can be approximated at any—higher—order of accuracy.

and notice that $f_x(x_0)\,Q_0 - B_0 = O(h^2)$. We thus obtain

$$Z_{1/2} = Q_0 + B_0$$

instead of $Y_{1/2}$. Next, we need to approximate Jacobian free the term $hf_x(x_{1/2})\,Z_{1/2}$. We cannot use the simple difference quotient above to approximate this term, since we would get stuck with terms of $O(h^2)$. For this reason, we use a higher order centered difference approximation. So, if we let $Z_{1/2} = [z_1^{1/2},\ldots, z_p^{1/2}]$, we then use

$$hf_x(x_{1/2})\,Z_{1/2} \approx M_0 \qquad \text{where}$$
$$M_0 := \tfrac{1}{2}[f(x_{1/2}+hz_1^{1/2}) - f(x_{1/2}-hz_1^{1/2}),\ldots, f(x_{1/2}+hz_p^{1/2}) - f(x_{1/2}-hz_p^{1/2})].$$

With the above notation, the Jacobian free midpoint scheme is

$$Z_{1/2} = Q_0 + B_0, \qquad Z_1 = Q_0 + M_0, \qquad \text{then} \quad Q_1 \colon Z_1 = Q_1 R_1. \tag{3.2}$$

It is simple to appreciate that the scheme (3.2) is a second order scheme, since the local error is $O(h^3)$. This is because $Z_{1/2}$ is an $O(h^2)$ approximation to $Y_{1/2}$ and $M_0 - hf_x(x_{1/2})\,Z_{1/2} = O(h^3)$. The latter follows by straightforward Taylor expansion; for $j = 1,\ldots, p$, one has

$$f(x_{1/2}+hz_j^{1/2}) - f(x_{1/2}-hz_j^{1/2}) = 2hf_x(x_{1/2})\,z_j^{1/2} + \frac{h^3}{6} f_{xxx}(x_{1/2})\,z_j^{1/2} z_j^{1/2} z_j^{1/2} + O(h^5).$$

Clearly, computing $Z_1$ from (3.2), rather than $Y_1$ from (3.1), avoids two Jacobian evaluations and the matrix multiplications with the Jacobians.

**Remark 3.1.** The above second order scheme (3.2) requires three evaluations per step of the vector field $f$ for each LE. We grew accustomed to second order RK-like schemes requiring just two $f$ evaluations. Whether or not this is actually possible in the present context is not clear to us, but in our—admittedly, limited—efforts we have not succeeded.

**Remark 3.2.** It is in principle possible to extend the above reasoning to obtain Jacobian free methods of even higher order. For example, for explicit RK schemes, one needs to replace all actions $hf_x(\cdot)\,Z_j$ by suitable difference quotients so that the order of the RK scheme is retained. However, for our scopes, second order schemes are sufficient, and we did not spend any time in trying to obtain high order analogs of (3.2), leaving this task to future work.

**(b) Extrapolating Forward Euler.** The basic extrapolation scheme on one step $h$ from $Q_0$ to $Q_1$ is

$$\begin{aligned} &Y_1 = Q_0 + hf_x(x_0)\, Q_0 \qquad \text{forward Euler step} \\ &Y_{1/2} = Q_0 + \frac{h}{2} f_x(x_0)\, Q_0, \qquad \hat{Y}_1 = Q_0 + \frac{h}{2} f_x(x_{1/2})\, Y_{1/2}, \qquad \text{two half steps} \\ &\text{then} \quad Y_1^* \leftarrow Y_1 + 2(\hat{Y}_1 - Y_1), \qquad \text{and} \quad Q_1 \colon Y_1^* = Q_1 R_1. \end{aligned} \tag{3.3}$$

It is well known that (3.3) is a second order scheme, see [16]. Our scope here is to derive a Jacobian free second order scheme from (3.3).

In what follows, we let $Q_0 = [q_1^0, \ldots, q_p^0]$ and $Z_{1/2} = [z_1^{1/2}, \ldots, z_p^{1/2}]$. We propose the following scheme

$$\begin{aligned} Z_1 &= Q_0 + [f(x_0 + hq_1^0) - f(x_0), \ldots, f(x_0 + hq_p^0) - f(x_0)] \\ Z_{1/2} &= Q_0 + \left[ f\left(x_0 + \frac{h}{2} q_1^0\right) - f(x_0), \ldots, f\left(x_0 + \frac{h}{2} q_p^0\right) - f(x_0) \right] \\ \hat{Z}_1 &= Q_0 + \left[ f\left(x_{1/2} + \frac{h}{2} z_1^{1/2}\right) - f(x_{1/2}), \ldots, f\left(x_{1/2} + \frac{h}{2} z_p^{1/2}\right) - f(x_{1/2}) \right] \\ &\text{then} \quad Z_1^* \leftarrow Z_1 + 2(\hat{Z}_1 - Z_1), \qquad \text{and} \quad Q_1 \colon Z_1^* = Q_1 R_1. \end{aligned} \tag{3.4}$$

It is not obvious that this is a second order scheme, so we prove it.

***Theorem 3.3.*** *The scheme (3.4) is a second order scheme. That is, if $Y(h)$ is the exact solution at $h$ of $\dot{Y} = f_x(x(t))\, Y$, $Y(0) = Q_0$, and $x_{1/2} = x_0 + \frac{h}{2} f(x_0) + O(h^2)$,[3] then $Y(h) - Z_1^* = O(h^3)$.*

**Proof.** Since it suffices to consider one single column, we just use lower case letters $q_0$, $y_{1/2}$, $z_{1/2}$, etc., where the notation is inherited from (3.3) and (3.4). Recall that (see (2.5))

$$y(h) - z_1 = \frac{h^2}{2} \ddot{y}_0 - \frac{h^2}{2} f_{xx}(x_0)(q_0, q_0) + O(h^3), \qquad y(h) - y_1 = \frac{h^2}{2} \ddot{y}_0 + O(h^3).$$

We have

$$z_{1/2} = y_{1/2} + \frac{h^2}{8} f_{xx}(x_0)(q_0, q_0) + O(h^3),$$

[3] I.e., $x_{1/2}$ is a second order approximation to $x(h/2)$.

and thus

$$\hat{z}_1 = y_{1/2} + \frac{h^2}{8} f_{xx}(x_0)(q_0, q_0) + \frac{h}{2} f_x(x_{1/2})\, z_{1/2} + \frac{h^2}{8} f_{xx}(x_{1/2})(z_{1/2}, z_{1/2}) + O(h^3)$$

$$= y_{1/2} + \frac{h}{2} f_x(x_{1/2})\, y_{1/2} + \frac{h^2}{8} f_{xx}(x_0)(q_0, q_0) + \frac{h^2}{8} f_{xx}(x_{1/2})(y_{1/2}, y_{1/2}) + O(h^3).$$

But

$$\frac{h^2}{8} f_{xx}(x_{1/2})(y_{1/2}, y_{1/2}) = \frac{h^2}{8} f_{xx}(x_0)(q_0, q_0) + O(h^3),$$

and so

$$\hat{z}_1 = \hat{y}_1 + \frac{h^2}{4} f_{xx}(x_0)(q_0, q_0) + O(h^3).$$

But also

$$z_1 = y_1 + \frac{h^2}{2} f_{xx}(x_0)(q_0, q_0) + O(h^3),$$

and therefore

$$\hat{z}_1 - z_1 = \hat{y}_1 - y_1 - \frac{h^2}{4} f_{xx}(x_0)(q_0, q_0) + O(h^3).$$

Now, since $\hat{y}_1 - y_1 = \frac{h^2}{4}\ddot{y}_0 + O(h^3)$, we have that

$$2(\hat{z}_1 - z_1) = \frac{h^2}{2}\ddot{y}_0 - \frac{h^2}{2} f_{xx}(x_0)(q_0, q_0) + O(h^3),$$

and so

$$z_1^* = z_1 + 2(\hat{z}_1 - z_1) = y(h) + O(h^3). \qquad \square$$

**Remark 3.4.** In (3.4), it is tempting to replace the terms $f(x_0 + \frac{h}{2} q_j^0) - f(x_0)$, $j = 1,\ldots, p$, in the definition of $Z_{1/2}$ by $\frac{1}{2}[f(x_0 + hq_j^0) - f(x_0)]$, $j = 1,\ldots, p$, which are of the same order of accuracy, and would save us $p$ function evaluations (the term in brackets was already computed to obtain $Z_1$). However, if we do so, the extrapolation procedure does not increase the order.

Exactly as for (3.2) and (3.1), to compute $Z_1$ from (3.4) rather than $Y_1$ from (3.3) avoids the Jacobian evaluations and the matrix multiplications with the Jacobians.

**Remark 3.5.** Regardless of whether one uses (3.2) or (3.4), of course the LEs are still updated as in (2.3), which now gives a 2nd order approximation for the LEs.

### 3.2. Continuous QR: 2nd Order

We adapt to this case the projected integrator based on the second order midpoint rule.

**(a) Explicit Midpoint Rule.** The procedure is the same as what we did for the explicit midpoint in the case of the discrete QR method. The true midpoint rule projected integrator would be

$$\begin{aligned} P_{1/2} &= Q_0 + \frac{h}{2}\,[f_x(x_0)\,Q_0 - Q_0(Q_0^T f_x(x_0)\,Q_0 - S_0)], \\ P_1 &= Q_0 + h[f_x(x_{1/2})\,P_{1/2} - P_{1/2}(P_{1/2}^T f_x(x_{1/2}P_{1/2} - S_{1/2})], \\ &\text{then orthogonalize } P_1 \text{ to get} \quad Q_1\colon P_1 = Q_1 R_1. \end{aligned} \tag{3.5}$$

To make this scheme Jacobian free, and to retain second order, we proceed as follows. The terms $hf_x(x_0)\,Q_0$ and $hS_0$ in the definition of $P_{1/2}$ can be approximated as we did to get to (2.7); that is, we let $B_0$ to be our approximation to $hf_x(x_0)\,Q_0$ and $H_0$ the approximation to $hS_0$: $B_0 := [f(x_0 + hq_1^0) - f(x_0), \ldots, f(x_0 + hq_p^0) - f(x_0)]$, and $(H_0)_{ij} = (Q_0^T B_0)_{ij}$, $i > j$, plus skew-symmetry. So doing, we obtain a value $V_{1/2}$ which is an $O(h^2)$ approximation to $P_{1/2}$:

$$V_{1/2} = Q_0 + \tfrac{1}{2}[B_0 - Q_0(Q_0^T B_0) + Q_0 H_0].$$

Next, we need to approximate at 3rd order the action $hf_x(x_{1/2})\,V_{1/2}$ and further use this in forming $hS_{1/2}$ (here, $S_{1/2}$ is defined by using $V_{1/2}$). This is accomplished by centered differences. We let $V_{1/2} = [v_1^{1/2}, \ldots, v_p^{1/2}]$, and let

$$\begin{aligned} hf_x(x_{1/2})\,V_{1/2} &\approx B_{1/2}, \\ B_{1/2} &:= \tfrac{1}{2}[f(x_{1/2} + hv_1^{1/2}) - f(x_{1/2} - hv_1^{1/2}), \ldots, \\ &\qquad f(x_{1/2} + hv_p^{1/2}) - f(x_{1/2} - hv_p^{1/2})], \end{aligned}$$

and also use this to approximate $hS_{1/2}$ by $H_{1/2}$, which is thus given by $(H_{1/2})_{ij} = (V_{1/2}^T B_{1/2})_{ij}$, $i > j$, and then using skew-symmetry. In the end, the resulting scheme is

$$\begin{aligned} V_{1/2} &= Q_0 + \tfrac{1}{2}[B_0 - Q_0(Q_0^T B_0) + Q_0 H_0], \\ V_1 &= Q_0 + B_{1/2} + V_{1/2}(H_{1/2} - V_{1/2}^T B_{1/2}), \\ &\text{then orthogonalize } V_1 \text{ to get} \quad Q_1 : V_1 = Q_1 R_1. \end{aligned} \tag{3.6}$$

By construction, this is a second order Jacobian free discretization of (1.11). To take advantage of the increased accuracy, we now update the LEs using the midpoint rule rather than the forward Euler as in (2.8). Thus, if $\lambda_i^j$, $i = 1,\ldots, p$, are the approximations of the LEs at $t_j$, we update these approximations at $t_{j+1}$ as

$$\lambda_i^{j+1} = \frac{t_j}{t_j + h}\lambda_i^j + \frac{1}{t_j + h}(Q_{j+1/2}^T B_{j+1/2} Q_{j+1/2})_{ii}, \qquad i = 1,\ldots, p, \tag{3.7}$$

which—given the way we approximate $B_{j+1/2}$—gives a 2nd order approximation to the LEs.

Again, computing $V_1$ from (3.6) rather than $P_1$ from (3.5) avoids the Jacobian evaluations and multiplications with the Jacobians.

## 4. IMPLEMENTATION AND EXAMPLES

We implemented all first and second order schemes given in the previous sections. For simplicity, we report on results where integration for the trajectory was done with the same scheme used to approximate the LEs. However, we also made experiments in which integration for the trajectory was carried out by a fourth order RK scheme; this had *no* impact on the quality of the answers obtained for the LEs, which continued to be approximated at first or second order, according to the schemes adopted for their approximation.

### 4.1. Expense

Before reporting on our experiments, we give a breakdown of the computational costs of the different methods. We monitor the cost per integration step in terms of the number of required flops,[4] and the number

[4] A *flop* is the cost of an arithmetic operation.

of required function evaluations (i.e., evaluations of the vector field $f$ in (1.1)). As it is customary, we will consider the cost of evaluating the Jacobian $f_x(\bar{x})$ to be $n$ function evaluations. This is appropriate in general, as evidenced by the common case in which the Jacobian is approximated by divided differences, whereby for each of its columns one uses

$$f_x(\bar{x})\, e_j \approx \frac{f(\bar{x}+\epsilon e_j)-f(\bar{x})}{\epsilon}, \qquad j=1,\ldots,n,$$

with $\epsilon \neq 0$ sufficiently small (typically, $\epsilon$ is the square root of the machine precision).

We will use the following naming conventions for the schemes we implemented

- FE is "Euler" method matrix Free, JE is Euler method using the Jacobian. We further differentiate between FED and FEC for the Discrete or Continuous QR approaches.
- Similarly, FEXD is the "EXtrapolation" method used matrix free for the discrete QR approach, JMPC is the "MidPoint" method using the Jacobian for the continuous QR approach, etc..

To arrive at the results summarized in Table I, we have adopted the following choices: (i) the schemes have been implemented in the form in which they have been written in the previous sections; (ii) to perform the QR factorization of a $(n, p)$-matrix (required by all schemes), we implemented the modified Gram–Schmidt algorithm, whose cost is reported as $2np^2$ flops in [14]; (iii) the cost of adding two $(n, p)$-matrices is $np$ flops, and of multiplying a $(m, q)$-matrix by a $(q, s)$-matrix is $2mqs$ flops (again,

**Table I.** Computational Costs

| Method | Flops | $f$-evaluations |
|---|---|---|
| JED, (2.1) | $2n^2p+2np^2+np$ | $n$ |
| FED, (2.2) | $2np^2+2np$ | $p$ |
| JEC, (2.6) | $2n^2p+5np^2+3np$ | $n$ |
| FEC, (2.7) | $5np^2+4np$ | $p$ |
| JMPD, (3.1) | $4n^2p+2np^2+2np$ | $2n$ |
| FMPD, (3.2) | $2np^2+4np$ | $3p$ |
| JEXD, (3.3) | $4n^2p+2np^2+7np$ | $2n$ |
| FEXD, (3.4) | $2np^2+7np$ | $3p$ |
| JMPC, (3.5) | $4n^2p+8np^2+6np$ | $2n$ |
| FMPC, (3.6) | $6np^2+9np$ | $3p$ |

see [14]); (iv) for the continuous methods, we have made use of the form of $S$ in (1.10) (and similarly for the Jacobian free versions), to save on arithmetic operations; (v) we have not counted the costs of updating the LEs, which is the same for all methods of same order, and is negligible for $p$ small.

Quite clearly, in the important case of $p \ll n$, the Jacobian-free methods have a cost per step proportional to $O(np^2)$ flops, and $O(p)$ function evaluations, while the methods requiring the Jacobian have a cost of $O(n^2p)$ flops and $O(n)$ function evaluations.

### 4.2. Examples

All experiments were made with FORTRAN codes, without any optimization option, on a Workstation with clock speed of 300 Mhz/s. Notation used in the tables later is as follows.

- Method is the method used.
- $h$ is the stepsize used.
- CPU are the computing times.
- $\lambda_i$, $i = 1,\ldots,$ are the approximated exponents.

**Example 4.1.** This is a problem adapted from one in [11], for which in [10] we computed accurate approximations of the LEs, and thus we will use it to test order of convergence of the schemes, and perform a moderate comparison.

We have a ring of oscillators with an external force proportional to the position component of the limit cycle of the van der Pol oscillator:

$$\begin{aligned} &\ddot{y}+\alpha(y^2-1)\,\dot{y}+\omega^2 y=0 \\ &\ddot{x}_i+d_i\dot{x}_i+\gamma[\Phi'(x_i-x_{i-1})-\Phi'(x_{i+1}-x_i)]=\sigma y\delta_{i1}, \qquad i=1,\ldots,m. \end{aligned} \tag{4.1}$$

Above, we set $\alpha = 1$, $\omega = 1.82$, $\gamma = 1$ and $\sigma = 4$. Also, $\Phi(x) = (x^2/2) + (x^4/4)$ is the single well Duffing potential, $x_i$ is the displacement of the $i$th particle, $d_i$ is the damping coefficient, and we have periodic boundary conditions to be used in the expressions for $\Phi'$ ($x_0 = x_m$ and $x_{m+1} = x_1$). For the present set of experiments, we take 5 oscillators, and $d_i = 0.25$ for $i$ odd and $d_i = 0.15$ for $i$ even. Initial conditions are taken $y(0) = 0$, $\dot{y}(0) = -2$, $x_i(0) = \dot{x}_i(0) = 1$, $i = 1,\ldots, m$. We integrate to $t = 1000$, approximating 4 Lyapunov exponents (i.e., in the previous notation, $n = 12$ and $p = 4$). The

**Table II.** Example 1. Discrete QR Method

| Method | $h$ | CPU | $\lambda_1$ | $\lambda_2$ | $\lambda_3$ | $\lambda_4$ |
|---|---|---|---|---|---|---|
| FED | 1.0E−2 | 6” | 2.9E−2 | 1.45E−2 | 3.5E−3 | 1.3E−4 |
| JED | 1.0E−2 | 12” | 2.8E−2 | 1.6E−2 | 3.3E−3 | 1.2E−3 |
| FED | 1.0E−4 | 10’ 8” | 1.85E−3 | 8.8E−4 | −9.9E−2 | −9.7E−2 |
| JED | 1.0E−4 | 20’ | 1.85E−3 | 8.8E−4 | −9.9E−2 | −9.7E−2 |
| FEXD | 1.0E−2 | 8” | 1.6E−3 | 8.6E−4 | −9.7E−2 | −1.0E−1 |
| JEXD | 1.0E−2 | 19.5” | 1.6E−3 | 8.6E−4 | −9.7E−2 | −1.0E−1 |
| FMPD | 1.0E−2 | 7.4” | 1.6E−3 | 8.6E−4 | −9.7E−2 | −1.0E−1 |
| JMPD | 1.0E−2 | 20” | 1.6E−3 | 8.6E−4 | −9.7E−2 | −1.0E−1 |

following values of these first 4 approximate exponents are believed to be accurate to the two digits shown (cf. [10]):

$$1.7\text{E}-3, \qquad 8.7\text{E}-4, \qquad -9.7\text{E}-2, \qquad -1.0\text{E}-1.$$

As it can be observed from Table II, the results of the Jacobian free versions agree very closely with those using the Jacobian. The first order scheme requires $h$ to be too small to deliver decent accuracy, while the second order schemes are all fairly accurate. On such small problem, the savings achieved with the Jacobian free versions are a bit more than 50%.

From Table III, we again observe that the results of the Jacobian free versions agree very closely with those using the Jacobian, the first order scheme requires $h$ to be too small to become accurate, and the savings in the Jacobian free version are a bit less than 50%. The continuous QR approach is slightly less accurate than the discrete QR counterpart, and 2 to 3 times as expensive.

**Table III.** Example 1. Continuous QR Method

| Method | $h$ | CPU | $\lambda_1$ | $\lambda_2$ | $\lambda_3$ | $\lambda_4$ |
|---|---|---|---|---|---|---|
| FEC | 1.0E−2 | 10.2” | −1.5E−1 | −5.6E−2 | −1.9E−1 | −7.2E−2 |
| FEC | 1.0E−2 | 20” | −1.6E−1 | −4.4E−2 | −1.9E−1 | −7.3E−3 |
| FEC | 1.0E−4 | 17’ 13” | 1.4E−3 | 7.0E−4 | −1.0E−1 | −9.8E−2 |
| JEC | 1.0E−4 | 33’ | 1.4E−3 | 7.0E−3 | −1.0E−1 | −9.8E−2 |
| FMPC | 1.0E−2 | 21.7” | 1.4E−3 | 6.9E−4 | −9.9E−2 | −9.8E−2 |
| JMPC | 1.0E−2 | 38.4” | 1.4E−3 | 7.0E−4 | −9.8E−2 | −9.9E−2 |
| FMPC | 1.0E−3 | 3’ 39” ” | 1.7E−3 | 8.7E−4 | −9.7E−2 | −1.0E−1 |
| JMPC | 1.0E−3 | 6’ 22” | 1.7E−3 | 8.7E−4 | −9.7E−2 | −1.0E−1 |

**Table IV.** Example 1. All 12 Exponents: CPU

| FMPD | JMPD | FMPC | JMPC |
|---|---|---|---|
| 36.3" | 1' 10" | 2' 38" | 3' 25" |

In Table IV, we report the CPU times when we approximate all 12 exponents at $t = 1000$ by the midpoint rule methods. Notice that the Jacobian free continuous QR method scales poorly with increasing $p$ (here, $p = n$), because of the need for the matrix multiplications $Q^T B$ (recall (3.6)).

**Example 4.2.** This is the same as Example 4.1 except that we now increase the number of oscillators, taking $m = 15$ and $m = 150$, respectively. Comparison values in this case are not known. However, we perform this experiment by taking the following parameter values (as in [11]): $\alpha = 1$, $\omega = 1.6$, $\gamma = 1$, $\sigma = 2$, and $d_i = 0.4$ for all $i = 1,\ldots, m$. With these values, we expect one LE equal to 0, and all other LEs negative. Initial conditions are as in Example 4.1. In all runs below integration is done up to $t = 1000$, and $p = 4$ LEs are approximated.

From Table V, observe that the first order method for the discrete QR approach delivers qualitatively correct answers, while for the continuous QR approach doesnot. The Jacobian free and Jacobian based methods give nearly identical results.with the Jacobian free methods costing about 20 to 25% of the methods requiring the Jacobian. The Jacobian free second order methods for the discrete and continuous QR approaches give nearly identical results in terms of accuracy, with the continuous approach being 3 times as expensive.

Finally, in Table VI we report on the CPU times for the method when $m = 150$. The approximate LEs show the same quality as in Table V: those

**Table V.** Example 2, $m = 15$ ($n = 32$), $h = 1.0\text{E}-2$

| Method | CPU | $\lambda_1$ | $\lambda_2$ | $\lambda_3$ | $\lambda_4$ |
|---|---|---|---|---|---|
| FED | 15.3" | 9.3E−4 | −7.6E−4 | −9.1E−2 | −9.4E−2 |
| JED | 1' 2" | 1.4E−3 | −7.6E−4 | −9.3E−2 | −9.4E−2 |
| FMPD | 18.2" | 1.6E−3 | −7.3E−4 | −8.6E−2 | −8.75E−2 |
| JMPD | 1' 47" " | 1.6E−3 | −7.3E−4 | −8.6E−2 | −8.75E−2 |
| FEC | 15.6" " | −2.7E−2 | −1.8E−3 | −1.3E−1 | −1.4E−1 |
| JEC | 1' 26" | −2.7E−2 | −1.8E−3 | −1.3E−1 | −1.4E−1 |
| FMPC | 55" | 1.4E−3 | −7.3E−4 | −8.6E−2 | −8.75E−2 |
| JMPC | 3' 11" | 1.5E−3 | −7.3E−4 | −8.6E−2 | −8.75E−2 |

**Table VI.** Example 2, $m = 150$ ($n = 302$), $h = 1.0\text{E}-2$. CPU times

| FED | JED | FMPD | JMPD | FEC | JEC | FMPC | JMPC |
|---|---|---|---|---|---|---|---|
| 2' 15" | 1h 48' | 3' | 3h 35' 25" | 4' 6" | 2h 14' 18" | 9' | 4h 38' |

of FED and JED are qualitatively correct (7.3E−4, −1.9E−3, −1.1E−2, −2.8E−2), those of FEC and JEC are all negative (and seemingly inaccurate), and those for FMPD, JMPD, FMPC, JMPC are all nearly identical to one another (1.5E−3, −1.9E−3, −1.2E−2, −2.8E−2).

## 5. CONCLUSIONS

We have proposed schemes to approximate LEs of nonlinear differential equations which do not need the Jacobian matrix. The basic idea is really very simple, and consists in replacing the product of the Jacobian matrix $f_x$ times a matrix $Z$ by approximate directional derivatives. We gave first order and second order schemes for both discrete and continuous QR methods, and showed the reliability and effectiveness of our choices on two examples. It is our hope that these choices will prove valuable to those interested in approximation of LEs of large dimensional systems, and who have been reluctant to do so because of computational expense.

Based on the results of our experiments, we can draw the following conclusions and recommendations for future work.

(1) The Jacobian free version of both discrete and continuous QR approaches is considerably less expensive than the counterpart requiring the Jacobian, and it is at least equally accurate. Savings depend on whether one uses the discrete or continuous QR approach, on the dimension of the problem, and on the number of LEs one needs to monitor. In general, savings will also depend on how expensive it is to compute the Jacobian, and on its structure (which we have not taken into account in Examples 4.1 and 4.2). From our experiments, we observed that for small problems, and in case we need all LEs, savings areon the order of 50%, but already for moderate size problems (i.e., dimension 300), to compute a few LEs, the Jacobian free version may take up to only 1.5% of the time taken by the methods requiring the Jacobian. Needless to say, the Jacobian free versions have the major advantage of not requiring the derivative of the vector field in the first place.

(2) For the fixed stepsize low order schemes we have considered, the Jacobian free implementation of the discrete QR approach appears superior to the continuous QR analog. Clearly, the discrete QR approach is consistently less expensive, and this could have been easily anticipated just by looking at the differential equations one needs to solve with the two different approaches; but, perhaps more importantly, it is also at least equally accurate. For example, the Jacobian free "Euler" method works poorly for the continuous QR approach, while it is rather reliable for the discrete QR approach (see especially Example 4.2). Furthermore, the expense needed for the Jacobian free discrete QR method scales favorably with respect to the number $p$ of desired LEs, while—as $p$ increases—the continuous QR approach becomes progressively penalized by (dense) matrix multiplications of $(n, p) \times (p, n)$ matrices. Recalling also Remark 1.3, we lean towards recommending the Jacobian free discrete QR approach for large nonlinear systems, at least when a modest number of LEs are desired. In fact, either of the two second order Jacobian free schemes we introduced for the discrete QR approach would be our recommended choice.

(3) Our basic schemes to approximate the LEs have been simple low order explicit RK methods, but extensions to other schemes are certainly possible and perhaps warranted. In future work, we may investigate higher order schemes as well as adapt our choices to different basic discretizations. We stress once more that our choices pertain exclusively to the approximation of the LEs, and integration for the trajectory can be performed with any other appropriate scheme.

(4) Finally, we observe that all Jacobian free second order schemes we derived can be implemented with variable stepsizes, by monitoring local errors with respect to the Jacobian free Euler method (and no extra evaluations of $f$ are required). We also leave to future development the implementation of error control and variable time stepping strategies.

## ACKNOWLEDGMENTS

This work was supported in part under NSF Grant DMS-9973266. The author would like to thank Erik Van Vleck for many fruitful discussions on Lyapunov exponents.

## REFERENCES


1. Adrianova, L. Ya. (1995). *Introduction to Linear Systems of Differential Equations*, Translations of Mathematical Monographs, Vol. 146, AMS, Providence, R.I.
2. Benettin, G., Galgani, L., Giorgilli, A., and Strelcyn, J.-M. (1980). Lyapunov exponents for smooth dynamical systems and for Hamiltonian systems; a method for computing all of them. Part 1: Theory, and … Part 2: Numerical applications. *Meccanica* **15** 9–20, 21–30.
3. Bridges, T., and Reich, S. (2001). Computing Lyapunov exponents on a Stiefel manifold. *Physica D* **156**, 219–238.
4. Brown, P., and Hindmarsh, A. (1989). Reduced storage matrix methods in stiff ODE systems. *Appl. Math. Comp.* **31**, 40–91.
5. Cliffe, K. A., Spence, A., and Tavener, S. (2000). The numerical analysis of bifurcation problems with application to fluid mechanics. *Acta Numerica*, 39–131.
6. Brown, P., and Hindmarsh, A. (1986). Matrix free methods for stiff systems of ODES. *SIAM J. Numer. Anal.* **23**, 610–638.
7. Dieci, L., and Lopez, L. Lyapunov exponents on quadratic groups, submitted.
8. Dieci, L., Russell, R. D., and Van Vleck, E. S. (1997). On the computation of Lyapunov exponents for continuous dynamical systems. *SIAM J. Numer. Anal.* **34**, 402–423.
9. Dieci, L., and Van Vleck, E. S. (2001). Orthonormal Integrators Based on Householder and Givens Transformations, submitted.
10. Dieci, L., and Van Vleck, E. S. (2002). Lyapunov spectral intervals: Theory and computation, to appear in *SIAM J. Numer. Anal.*.
11. Dressler, U. (1988). Symmetry property of the Lyapunov spectra of a class of dissipative dynamical systems with viscous damping. *Phys. Rev. A* **38**(4), 2103–2109.
12. Eckmann, J. P., and Ruelle, D. (1985). Ergodic theory of chaos and strange attractors. *Rev. Modern Phys.* **57**, 617–656.
13. Geist, K., Parlitz, U., and Lauterborn, W. (1990). Comparison of different methods for computing Lyapunov exponents. *Prog. Theor. Phys.* **83**, 875–893.
14. Golub, G. H., and Van Loan, C. F. (1989). *Matrix Computations*, The Johns Hopkins University Press, 2nd ed.
15. Gupalo, D., Kaganovich, A. S., and Cohen, E. G. D. (1994). Symmetry of Lyapunov spectrum. *J. Statist. Phys.* **74**, 1145–1159.
16. Hairer, E., Nœrsett, S. P., and Wanner, G. (1993). *Solving Ordinary Differential Equations I*, Springer-Verlag, Berlin/Heidelberg, 2nd ed.
17. Johnson, R. A., Palmer, K. J., and Sell, S. (1987). Ergodic properties of linear dynamical systems. *SIAM J. Math. Anal.* **18**, 1–33.
18. Lyapunov, A. (1949). *Problém géneral de la stabilité du mouvement*, Annals of Mathematics Studies, Vol. 17, Princeton University Press.
19. Millionshchikov, V. M. (1971). Linear systems of ordinary differential equations. *Differ. Uravn.* **7-3**, 387–390.
20. Oseledec, V. I. (1968). A multiplicative ergodic theorem. Lyapunov characteristic numbers for dynamical systems. *Trans. Moscow Math. Soc.* **19**, 197–231.
21. Perron, O. (1930). Die Ordnungszahlen Linearer Differentialgleichungssysteme. *Math. Z.* **31**, 748–766.
22. Wolf, A., Swift, J. B., Swinney, H. L., and Vastano, J. A. (1985). Determining Lyapunov exponents from a time series. *Physica D* **16**, 285–317.


*Printed in Belgium*