LESLIS and LESLIL: Codes for approximating Lyapunov exponents of linear systems [1]

Luca Dieci[2] Erik S. Van Vleck[3]

December 9, 2004

[1]This work was supported in part under NSF Grants DMS-FRG 0139895 and 0139824.

[2]School of Mathematics, Georgia Institute of Technology, Atlanta, Georgia 30332 (dieci@math.gatech.edu).

[3]Department of Mathematics, University of Kansas, Lawrence, Kansas 66045 (evanvleck@math.ukans.edu).

Abstract

In this technical report we present two suites of `FORTRAN` codes, `LESLIS` and `LESLIL`, for approximating Lyapunov exponents of nonautonomous linear differential systems by QR methods. The two codes are very similar, the main difference being that `LESLIS` is apt to solve *small* systems for which we can store the coefficients' matrix, while `LESLIL` is geared for *large* systems for which the coefficients' matrix is not stored and only its action on a vector is required. We summarize options, capabilities, and limitations, of the codes. Examples are given to show how to setup drivers, and to show performance of the codes. A limited comparison of the different options is also provided. Finally, we highlight how these codes are used to approximate other spectral information of linear systems, such as the Exponential Dichotomy spectrum.

Keywords: Lyapunov Exponents, Linear Systems, Spectra, Exponential Dichotomy, Numerical Methods, Scientific Computation, Software.

AMS Subject Classification: 65-04, 65L, 65P, 65Y

Contents

Chapter 1

Approximating Lyapunov Exponents

1.1 Disclaimer

Using the results of our codes require sensible interpretation. Why? Lyapunov exponents provide the fundamental quantities by which we are able to measure the asymptotic exponential behavior of solutions of differential equations. By the very nature of the limiting process intrinsic in the definition of Lyapunov exponents, the approximation of these quantities is bound to be limited in extent, and perhaps the approximations themselves may be considered of dubious validity. On the other hand, enough progress has been made at least in identifying which tools are at our disposal if we are interested in approximating Lyapunov exponents. For this reason, we believe that the time is ripe for providing the scientific community with some tools to approximate Lyapunov exponents, much in the same way as about 50 to 40 years ago codes begun being developed for approximating eigenvalues of matrices. Indeed, our scope in developing `LESLIS` and `LESLIL` (and `LESNLS` and `LESNLL`) has been to create a platform for growth, and a benchmark: We believe that some of the choices we have adopted will be refined and improved by us and others in the years ahead as more complete numerical analysis of the task becomes available. At the same time, our codes are a sensible implementations of methods which **are state of the art** for approximating Lyapunov exponents.

1.2 Introduction

Since their inception –about 100 years ago, in the thesis of Lyapunov [34]– Lyapunov exponents (LEs, for short) have proved to be an extremely valuable tool to study dynamical systems, and nowadays Lyapunov exponents (in some of their several adaptations) are routinely used in many scientific studies. To name some of their uses, they are used to analyze asymptotic stability of continuous and discrete systems, to analyze time series, to test the limits of predictability of models, to indicate chaotic behavior of a dynamical system, to assess effects of random perturbations of a system, to estimate entropy, dimension of attractors, as well as to force desired stabilization of a system.

Lyapunov exponents lead naturally to a definition of spectrum of a linear non-autonomous system, but there are also other (non equivalent) useful definition of spectra, for example the

Exponential Dichotomy (or Sacker-Sell) spectrum. Both these spectra are useful in the study of dynamical systems.

In spite of such widespread use, the numerical approximation of spectra remains a very delicate and computationally demanding task. For one thing, the spectra provide information on the asymptotic behavior of a system, and it is obviously impossible to compute on an infinite time interval. Unsurprisingly, in the literature one often finds the concept of so–called *finite time* Lyapunov exponents, and it is argued that these are the quantities of physical interest. Moreover, in general, there are (at least) two Lyapunov exponents associated to a given trajectory of a dynamical system, one defined as lim sup, the other as lim inf. These two values make up the endpoints of the so-called Lyapunov spectral interval relative to the given trajectory.

In this report, we will restrict to linear problem. In the accompanying report [13], we consider nonlinear dynamical systems. In all cases, we are exclusively concerned with the numerical approximation of spectra for **continuous dynamical systems**, defined by a system differential equations and we focus on implementation of **methods to approximate spectra** by means of information which can be retrieved from time integration of the differential equations.

We refer to [1, 6] for monographs on Lyapunov exponents, and to [17, 21] for recent recounts with an eye to numerical approximation. Here below we provide just minimal background information.

1.3 The Problem

Consider the m-dimensional linear system

$$\dot{y} = A(t)y\,, \quad t \geq 0\,, \tag{1.1}$$

where A is continuous and a fundamental matrix solution has "log-bounded" growth. For example, A may be bounded: $\sup_t \|A(t)\| < \infty$. Throughout, $\|\cdot\|$ is the 2-norm.

Define the numbers μ_j, $j = 1, \ldots, m$, as

$$\mu_j = \limsup_{t\to\infty} \frac{1}{t} \log \|Y(t)e_j\|\,, \tag{1.2}$$

where the e_j's are the standard unit vectors and $Y(t)$ is the solution of

$$\dot{Y} = A(t)Y\,, \quad Y(0) = Y_0 \ \text{ full rank.}$$

When the sum of these numbers μ_j is minimized as we vary over all possible ICs (initial conditions) Y_0, the numbers are called the upper LEs of the system, and the ICs are said to form a *normal basis*. We will write λ_j^s, $j = 1, \ldots, m$, for the ordered upper LEs of (1.1). By working with the adjoint system

$$\dot{Z} = -A^T(t)Z\,,$$

we analogously can define (lower) LEs, λ_j^i, $j = 1, \ldots, m$, which again we consider ordered. The λ_j^i and λ_j^s then make up the endpoints of the so-called Lyapunov spectral intervals. In case in which $\lambda_j^i = \lambda_j^s = \lambda_j$, for all $j = 1, \ldots, m$, and

$$\lim_{t\to\infty} \frac{1}{t} \log\big(\det(Y(t))\big) = \sum_{j=1}^{m} \lambda_j\,,$$

then the system is called **regular**, and in this case the exponents are found as limits. A most important consequence of regularity is (already in [34]): *If we have a regular system with upper*

triangular coefficient matrix $B(t)$: $\dot{y} = B(t)y$, *then, for* $j = 1, \ldots, m$, *its Lyapunov exponents are given by*

$$\lambda_j \;=\; \lim_{t\to\infty} \frac{1}{t} \int_0^t B_{jj}(s)ds\,. \tag{1.3}$$

Nearly all numerical works of which we are aware (see references at the end) assume that the system is regular. Although a convenient[1] assumption, regularity is not sufficient to guarantee stability of the Lyapunov exponents, which is what we need to have in order to pursue computational procedures for their approximation.

Stability for the LEs means that small perturbations in the function of coefficients, A, produce small changes in the LEs. Millionschikov (see [36, 35]) and Bylov and Izobov (see [5]) gave conditions under which the LEs are stable, and further proved that these conditions are *generic* in the class of linear systems with continuous bounded coefficients. The key assumption needed is **integral separation**: "A fundamental matrix solution (written columnwise) $Y(t) = [Y_1(t), \ldots, Y_m(t)]$ is integrally separated if for $j = 1, \ldots, m-1$, there exist $a > 0$ and $d > 0$ such that

$$\frac{||Y_j(t)||}{||Y_j(s)||} \cdot \frac{||Y_{j+1}(s)||}{||Y_{j+1}(t)||} \geq de^{a(t-s)}\,, \tag{1.4}$$

for all $t, s \;:\; t \geq s$". In the cited works, it is proved that "If the system (1.1) has different characteristic exponents $\lambda_1 > \ldots > \lambda_m$, then they are stable if and only if there exists a fundamental matrix solution with integrally separated columns".

Often, only the n most dominant (outermost to the right) spectral intervals are needed (and n can be much smaller than m). In these cases, the matrix Y_0 of initial conditions is made up by just n columns (often taken to be $\begin{pmatrix} I_n \\ 0 \end{pmatrix}$) and thus $Y : t \in I\!R \to I\!R^{m\times n}$. With this in mind, we will henceforth restrict to the case in which n LEs are desired for an m-dimensional linear system.

[1] and quite reasonable in many practical situations, see [38]

Chapter 2

QR approaches

The chief difficulty in approximating the LEs is that direct integration of a fundamental matrix solution of (1.1) is a well known unadvisable thing to do. Indeed, the most successful general techniques rest on transformation of the matrix solution (or of the coefficient function) to simpler forms from which the exponents may be extracted. `LESLIS` and `LESLIL` are based upon transformation to triangular form. This is clearly advantageous. In fact, if the system is triangular and regular[1]: $\dot{R} = B(t)R$ with B upper triangular, then the exponents are attained from the time averages of the diagonal of B (see (1.3)):

$$\lambda_i \;=\; \lim_{t\to\infty} \frac{1}{t} \int_0^t B_{ii}(s)ds\,,\; i = 1, \ldots, n\,, \tag{2.1}$$

or equivalently as

$$\lambda_i \;=\; \lim_{t\to\infty} \frac{1}{t} \log R_{ii}(t)\,,\; i = 1, \ldots, n\,. \tag{2.2}$$

Naturally, one cannot rely on transformations which do not have guaranteed stability, or which are not globally defined for all times. For this reason, the successful techniques all involve use of orthogonal transformations, in particular giving the QR factorization of the matrix solution. These are the methods we implemented in `LESLIS` and `LESLIL`. In a nutshell, these techniques are based on the well known fact that Y can be uniquely smoothly decomposed as product of an orthonormal function and an upper triangular function with positive diagonal. Then, in case the system is regular, (2.1-2.2) can be used to approximate the exponents. Two flavors of QR techniques have been developed, *discrete* and *continuous*. We recall them next.

Note. In what follows, when we will talk about the QR factorization of a (full rank) matrix, or of a matrix valued function, we will always refer to the unique QR factorization for which the diagonal of R has positive entries.

2.1 Discrete QR method

Let $t_0 = 0$, and $Y_0 = Q_0 R_0$. Suppose we want the QR factorization of Y at the point t_{k+1}. For $j = 0, \ldots, k$, progressively define $Y_{j+1}(t) = Y(t, t_j)Q_j$, $Y_{j+1} :\; t \in [t_j, t_{j+1}] \to I\!R^{m\times n}$, as the solution of

$$\begin{cases} \dot{Y}_{j+1} = A(t)Y_{j+1}\;,\;\; t_j \le t \le t_{j+1} \\ Y_{j+1}(t_j) = Q_j\,, \end{cases} \tag{2.3}$$

[1] recall that (1.1) is *regular*, if the LEs exist as limits and $\lim_{t\to\infty} \frac{1}{t}\int_0^t (\mathrm{trace} A(s))ds = \sum_{i=1}^n \lambda_i$

and update the QR factorization as

$$Y_{j+1}(t_{j+1}) = Q_{j+1}R_{j+1}\,, \tag{2.4}$$

so that

$$Y(t_{k+1}) = Q_{k+1}\ [R_{k+1}R_k\cdots R_1R_0] \tag{2.5}$$

is the sought QR factorization of $Y(t_{k+1})$: $Q_{k+1}\in I\!R^{m\times n}$ and $\prod_{j=k+1}^{0} R_j \in I\!R^{n\times n}$. The upper LEs are given by the relation (cfr. with (2.2))

$$\lambda_i = \limsup_{k\to\infty} \frac{1}{k}\sum_{j=0}^{k} \log(R_j)_{ii}\,,\ i=1,\ldots,n\,. \tag{2.6}$$

2.2 Continuous QR method

Let $t_0=0$, and $Y_0=Q_0R_0$. The idea of the continuous QR method is to solve the differential equations governing the evolution of the Q and R factors in the QR factorization of Y, without explicitly finding Y. Moreover, R is not really needed, and only Q can be approximated. We see this next.

Differentiating the relation $Y=QR$ one gets $\dot{Q}R+Q\dot{R}=A(t)QR$, and multiplying by Q^T on the left one gets the equation for R:

$$\dot{R} \;=\; B(t)R\,,\quad R(0)=R_0\,,\ \ B(t):=Q^TA(t)Q-S\ , \tag{2.7}$$

where we have set $S:=Q^T\dot{Q}$. Since $Q^TQ=I$, then S is skew-symmetric with values in $I\!R^{n\times n}$, and since R is triangular then S is given by

$$S_{ij}=\begin{cases} (Q^T(t)A(t)Q(t))_{ij}, & i>j,\\ 0, & i=j,\\ -S_{ji}, & i<j. \end{cases} \tag{2.8}$$

Next, multiplying $\dot{Q}R+Q\dot{R}=A(t)QR$ by R^{-1} on the right, and using (2.7), we get the differential equation for Q:

$$\dot{Q} \;=\; (I-QQ^T)A(t)Q+QS\,,\quad Q(0)=Q_0\,. \tag{2.9}$$

From (2.7), the (upper) LEs are defined as (cfr. (2.1))

$$\lambda_i = \limsup_{t\to\infty}\frac{1}{t}\int_0^t (Q^T(s)A(s)Q(s))_{ii}ds = \limsup_{t\to\infty}\frac{1}{t}\int_0^t B_{ii}(s)ds\,,\quad i=1,\ldots,n\,. \tag{2.10}$$

Chapter 3

Implementation

Here we describe how we implemented the QR methods outlined in the previous chapter. The overall goal is to compute approximations –for given values of t– to the quantities $\lambda_i(t), i = 1, \ldots, n$, (see (2.1, 2.2)) herein defined as

$$\lambda_i(t) = \frac{1}{t}\nu_i(t)\,, \quad i = 1, \ldots, n, \tag{3.1}$$

where the ν_i's are defined as

$$\nu_i(t) = \log R_{ii}(t)\,,\ i = 1, \ldots, n\,, \tag{3.2}$$

or equivalently as

$$\nu_i(t) = \int_0^t B_{ii}(s)ds\,,\ i = 1, \ldots, n\,. \tag{3.3}$$

Of course, (3.2) is more natural when one uses a discrete QR method, while (3.3) is more natural when one uses a continuous QR method. In a typical use, our codes will proceed with adaptive stepsizes obtained by controlling local errors on the $\nu_i(t)$, $i = 1, \ldots, n$, with respect to a preassigned tolerance vector `TOL` (`TOL` is a vector of length n).

In what follows, the superscript "c" will refer to *computed*, hence approximate, quantities. For example, we will write $\nu_i^c(t)$ to mean an approximation to either one of (3.2) or (3.3).

3.1 Discrete QR method

On the surface, implementing this method is rather straightforward, and indeed this is the most widely adopted technique amongst practitioners. However, it may not be the most efficient or reliable technique; e.g., for linear problems, the argument in [14] shows that it is not. And, regardless, careful implementation of the approach is warranted.

On a typical step from t_j to t_{j+1}, there are three tasks to carry out: (i) To integrate (2.3) (ii) To compute the QR factorization (2.4), and (iii) To update the approximation to $\lambda_i(t_{j+1})$, $i = 1, \ldots, n$. These are dealt with as follows.

(i) The differential equation (2.3) is integrated with embedded RK schemes. We implemented two such pairs. The first is the well known Dormand-Prince pair of order (5,4), the other is the Runge-Kutta 3/8 rule with embedded formula in the FSAL style, thereby giving a pair of order (4,3). We will henceforth refer to these pairs as `DP5` and `RK38`. The Runge-Kutta coefficients are given in the Appendix, and see [31]. Thus, at the end of a step, we have two

approximations to $Y_{j+1}(t_{j+1})$ in (2.4), call them Y_{j+1} (the higher order formula) and $\widehat{Y}_{j+1}$. It is quite important to point out immediately that **no** error control is performed on these approximations; see (iii) below.

(ii) As far as the QR factorizations of Y_{j+1} and $\widehat{Y}_{j+1}$, we implemented our own modified Gram-Schmidt procedure, as well as tested it against the QR routine from `LAPACK` which is based on Householder transformations (and which we post processed in order to guarantee that the R-factor had positive diagonal). In our tests, our routine was consistently faster, and thus we adopted it as default.

(iii) From the QR factorization of Y_{j+1} and $\widehat{Y}_{j+1}$, we obtain two approximations to R_{j+1} in (2.4), call them R^c_{j+1} and $\widehat{R}^c_{j+1}$ which can be used to perform error control. We use a mixed error control on the one step error to ν_i, $i = 1, \ldots, n$, as follows. Let h be the current stepsize, and $h_{\rm new}$ the new stepsize. Then, subject to the restriction that $h_{\rm new} \leq 5h$, we choose $h_{\rm new}$ as follows.

- Estimate the worse error componentwise with respect to the desired tolerance:

$$\texttt{err} = \max_{1\leq i\leq n} \left\{ |(R^c_{j+1})_{ii} - (\widehat{R}^c_{j+1})_{ii}| / \left[(1 + |(R^c_{j+1})_{ii}|)\texttt{TOL}_i \right] \right\}. \tag{3.4}$$

- Now estimate

$$h_{\rm new} = \texttt{safe}\ |h\ (1/\texttt{err})^l| \tag{3.5}$$

  where `safe` is a safety factor we set at 0.8, and $l = 1/5$ for `DP5` and $l = 1/4$ for `RK38`.

- If $\texttt{err} \leq 1$ the step is successful and accepted, otherwise is rejected. In the latter case, we never allow $h_{\rm new}$ to be less than 1/5 of h. [Though this may theoretically lead to repeated failures, in our experience it is a safeguard against drastic, but unwarranted, stepsizes' reductions].

Finally, after a successful step, we are ready to update the approximations:

$$\lambda^c_i(t_{j+1}) = \frac{t_j}{t_{j+1}} \lambda^c_i(t_j) + \frac{1}{t_{j+1}} \log(R^c_{j+1})_{ii}\,, \quad i = 1, \ldots, n\,. \tag{3.6}$$

Remark 3.1.1 The adaptive stepsize strategy above is similar to, but more conservative than, the one explained in [31]. A difference is that we advance the solution with the high order scheme, but estimate the local errors and new stepsize according to the lower order approximation.

3.2 Continuous QR method

On the surface, this method also looks rather simple. On a step from t_j to $t_{j+1} = t_j + h_j$, we need to: (i) Integrate (2.9) with initial conditions given by Q_j (i.e., our approximation $Q^c(t_j)$), and (ii) Update the approximations to $\nu_i(t_{j+1})$ in (3.3). These two tasks are actually quite delicate. In particular, for (i) there are a host of possibilities which deliver approximations which are orthogonal at the grid points; references [4, 7, 14, 15, 16, 19, 23, 32] all address this issue, and there are also public domain codes, [20], for this task. However, and in spite of us being the authors of the code `QRINT` in [19], we ended up revisiting some old ideas, and proposing also a new method. The reason we did not adopt `QRINT` is that the structure of the code made it hard to exploit sparsity structure of A for large problems, and we wanted methods which were flexible enough to accommodate sparse, and large, A.

Now, our task is to approximate (3.3). Thus, since

$$\nu_i(t_{j+1}) = \nu_i(t_j) + \int_{t_j}^{t_{j+1}} B_{ii}(t)dt \ , \quad i = 1, \ldots, n \, ,$$

we really need to approximate the local integrals

$$\mu_i = \int_{t_j}^{t_{j+1}} (Q^T(t)A(t)Q(t))_{ii}dt \ , \quad i = 1, \ldots, n \, . \tag{3.7}$$

To approximate (3.7) requires a quadrature, and approximations to Q in the subinterval $[t_j, t_{j+1}]$. Which quadrature we use depends on the scheme adopted to approximate Q. We now discuss this latter issue in details. As consequence of how we approximate Q, we will see how the μ_i's are also approximated. Of course, (3.7) is the same as solving on the step $[t_j, t_{j+1}]$ the differential equation

$$\dot{\mu}_i = (Q^T(t)A(t)Q(t))_{ii} \ , \quad \mu_i(t_j) = 0 \ , \quad i = 1, \ldots, n \, . \tag{3.8}$$

3.2.1 Q-integration

We explored several possibilities. Let $\dot{Q} = F(t, Q)$ be a shorthand notation for (2.9).

(1) In [15], the authors proposed a straightforward method to approximate the solution of (2.9): integrate (2.9) with an explicit Runge-Kutta scheme (for us, the pairs `DP5` or `RK38`), and orthogonalize the obtained solution(s) at the end of the step. The orthogonalization is done by replacing the obtained approximation with the Q factor of its QR factorization. We call this the *simple projected* scheme. In standard RK notation, the scheme can be written as follows:

$$U_{j+1} = Q_j + h_j(b_1K_1 + \ldots + b_sK_s) \ , \quad U_{j+1} \to Q_{j+1} \, , \tag{3.9}$$

where the arrow denotes replacement (via the QR factorization) with an orthonormal basis. Above, we have set

$$K_l = F(t_j + c_l h_j, U_{jl}) \ , \quad U_{jl} = Q_j + h_j(a_{l1}K_1 + \ldots + a_{l,l-1}K_{l-1}) \, . \tag{3.10}$$

In this case, the μ_i's can be approximated by the trapezoidal rule. That is, letting Q_{j+1} be the higher order approximation obtained, we would form

$$\mu_i^c = \frac{h_j}{2}\big((Q_j^T A(t_j)Q_j + Q_{j+1}A(t_{j+1})Q_{j+1})_{ii}\big) \, . \tag{3.11}$$

This way of proceeding is given as an option in `LESLIS` and `LESLIL`, though it is not our recommended choice. If this choice is adopted, error control can only be performed on the Q-factor. That is, calling $\widehat{Q}_{j+1}$ the lower order approximation obtained, an error control similar to the one of the discrete QR method is used for the max-element norm on each column of the Q-factor. More precisely, the estimate `err` is obtained as

$$\texttt{err} = \max_{1 \le i \le n} \big\{ \big(\|(Q_{j+1} - \widehat{Q}_{j+1})_{:,i}\|_\infty \big) / \big[(1 + \|(Q_{j+1})_{:,i}\|_\infty)\texttt{TOL}_i \big] \big\} \tag{3.12}$$

which is then used to select h_{new} in (3.5).

(2) The default integrator for (2.9) implemented in our codes is a slightly different scheme than the *simple* projected scheme above, and we will call it the *complete projected* scheme. Upon using an explicit Runge-Kutta scheme (`DP5` or `RK38`) to integrate (2.9), we orthogonalize all stage values as well. Again, the orthogonalization is carried out by using the QR factorization of the stage values and retaining the Q factor. So doing, we can use the same quadrature

as the basic scheme for approximating the μ_i's. In standard RK notation, the basic step we take is

$$U_{j+1} = Q_j + h_j(b_1K_1 + \ldots + b_sK_s) \;, \quad U_{j+1} \to Q_{j+1} \,, \tag{3.13}$$

but now (cfr. with (3.9-3.10))

$$K_l = F(t_j + c_l h_j, Q_{jl}) \;, \quad U_{jl} \to Q_{jl} \,, \tag{3.14}$$

and the μ_i's are approximated (see (3.8)) as

$$\mu_i^c = h_j \; (b_1 f_1 + \ldots b_s f_s) \,, \quad f_l = \big(Q_{jl}^T A(t_j + c_l h_j) Q_{jl}\big)_{ii} \,. \tag{3.15}$$

Now we will have obtained two approximations to the Q factor, Q_{j+1} and $\widehat{Q}_{j+1}$, and two approximations to the μ_i's, call them μ_i^c and $\widehat{\mu}_i^c$. We can control the error on the Q-factor as in (3.12), or can use μ_i^c and $\widehat{\mu}_i^c$ to perform error control directly. That is, we can now estimate `err` as

$$\texttt{err} = \max_{1 \le i \le n} \left\{ \big(|\mu_i^c - \widehat{\mu}_i^c|\big) / \big[(1 + |\mu_i^c|)\texttt{TOL}_i\big] \right\} \tag{3.16}$$

and again use (3.5) to select the new stepsize.

(3) Finally, there is one more integration scheme in `LESLIS-LESLIL` for solving (2.9). We will call it the *hybrid* scheme. Apparently, this is a new scheme[1]. The idea is very simple: We find approximation to the solution of (2.9) by projecting (via the QR factorization) the computed approximation to (2.3). In other words, on the step $[t_j, t_{j+1}]$, we integrate with `DP5` or `RK38` the differential equation

$$\dot{Y} = A(t)Y \;, \quad Y(t_j) = Q_j \,,$$

and only retain the Q-factor of the obtained approximations. Again, this can be done for all stage values and then the quadrature rule given by the integrators is used to obtain μ_i^c and $\widehat{\mu}_i^c$, otherwise if we only orthogonalize at the end of the step then (3.11) is used. In the first case we have a *complete hybrid projected* scheme and error control can be performed on the μ_i's and/or on Q, in the latter case we have a *simple hybrid projected* scheme and error control can only be performed on Q. To exemplify, the *simple hybrid projected* method looks like

$$\begin{array}{l} Y_{j+1} = Q_j + h_j(b_1K_1 + \ldots + b_sK_s) \;, \quad Y_{j+1} \to Q_{j+1} \,, \\ K_l = A(t_j + c_l h_j)Y_{jl} \;, \quad Y_{jl} = Q_j + h_j(a_{l1}K_1 + \ldots + a_{l.l-1}K_{l-1}) \,, \end{array} \tag{3.17}$$

and μ_i^c from (3.11). The *complete hybrid projected* method, instead, has as only difference that in (3.17), after forming Y_{jl}, we do

$$Y_{jl} \to Q_{jl} \,, \tag{3.18}$$

and the μ_i's are then approximated from (3.15). Notice that the projection step (3.18) is carried out only to provide the needed values for using (3.15), but the values Y_{jl} are not modified in (3.17).

Remark 3.2.1 In spite of solving the same differential equations, the hybrid scheme and the discrete QR method are distinctly different. The key difference is that **only** the Q-factor in the QR factorization of the transition matrices is retained in the hybrid scheme and the R-factor is never used, unlike in the discrete QR method. The hybrid scheme is also distinctly different from the projected schemes, since the hybrid scheme advances the solution by projecting the solution of

[1] We had used this technique in the past, but neglected it for many years since our unrefined implementations at the time misled us.

the linear system $\dot{Y} = A(t)Y$, not by projecting the solution of (2.9). On a given step, solving the linear system is less expensive than integrating (2.9). This fact becomes apparent in case we use (3.11), less so if we also project the stage values approximations and perform the full RK approximation for the μ_i's: Indeed, in this case, the key expense is in forming the integrand and this is obtained as a bonus with the complete projected schemes.

Remark 3.2.2 In case we can control either the error on Q or on the μ_i's directly, according to either (3.12) or (3.16), one must appreciate in which way these two error controls differ. In case we use (3.16), or (3.4), then we are effectively performing a local *error-per-step* control on the values μ_i's in (3.7). In case we use (3.12), then we are effectively trying to control the error in the integrand of (3.7). Though this resembles a local *error-per-unit-step* control on the values μ_i's in (3.7), and it often acts that way, it may occasionally be misleading. For example, think of the case of A being triangular and Q the identity! In our experience, enforcing (3.16) without also enforcing (3.12) is potentially inaccurate.

Important. Many of the above possibilities implemented in `leslis` and `leslis` are not foolproof. They have been maintained in the codes to provide the user with a variety of testing options, and because each has potentially distinct advantages. Nevertheless, it is our experience that the **best general** strategy consists of using the complete projected scheme with associated full RK approximations of the integrals and error control performed via both (3.12) and (3.16).

3.3 Order results

It might not be immediately clear that all schemes we implemented for discrete and continuous QR methods maintain the order of the underlying RK scheme. However, it is not hard to convince oneself that this is the case. The reasoning relatively to the complete projection and hybrid schemes has not been reported elsewhere, so we now review it.

For the complete projection schemes, the basic fact is that the projection of the stage values is **not** changing the order of approximation of the stage values themselves. As a consequence, the order of the scheme is not changing either.

For the hybrid schemes, the basic fact is that the unique Q-factors in the QR factorizations of the Y-values in (3.17)-(3.18) are of the same order of accuracy of the Y-values themselves.

Chapter 4

LESLIS and LESLIL: The Codes

The basic mindframe of LESLIS and LESLIL is that the user gives a time T, and the codes return approximations to

$$\lambda_i(T) = \frac{1}{T} \log R_{ii}(T)\,,\ i = 1, \ldots, n\,, \tag{4.1}$$

when one chooses a discrete QR method, or to

$$\lambda_i(T) = \frac{1}{T} \int_0^T B_{ii}(s)ds\,,\ i = 1, \ldots, n\,, \tag{4.2}$$

when one uses a continuous QR method. If the user decides to continue for some larger values of T, then must simply call the codes again.

The information in (4.1-4.2) is immediately conducive to approximation of the Lyapunov exponents and can also be used to approximate other spectral intervals, as we will elucidate in Chapter 5.

4.1 LESLIS and LESLIL: Differences

Before calling either LESLIS or LESLIL, the user must call the subroutine INIT: this will be called only once, regardless of integration options. The user must call INIT after having set the input data specified in IPAR as well as the initial and final times, and error tolerances (see below, and see the Appendix for sample calls). A call to INIT will have the form

```
CALL INIT(M,N,IPAR,TO,TE,DT,TOLQ,TOLL,FWORK,IFLAG)
```

In INIT, the code sets up the workspace, check error tolerances, initial and final times, and sets dafault quantities. After having called INIT, the user can call LESLIS or LESLIL.

The major difference between LESLIS and LESLIL is the way the coefficient function A is defined by the user.

In LESLIS, the user must provide a subroutine where $A(t)$ is given, at any t. In LESLIL, the user must provide a subroutine where the action $A(t)v$, at any t, is given (here, v is a generic vector). This is the only difference the user will see between LESLIS and LESLIL.

(a) A call to LESLIS will be done as follows:

```
CALL LESLIS(GETA,M,N,APPLES,TO,TE,DT,YO,TOLQ,TOLL,IPAR,FWORK,IFLAG,INARR,REARR)
```

where GETA, declared EXTERNAL, is the subroutine where the user defines $A(t)$ and must obey the syntax

```
SUBROUTINE GETA(M,T,A,INARR,REARR)
INTEGER M, INARR(*)
DOUBLE PRECISION T, A(M,M), REARR(*)
```

(b) Instead, a call to LESLIL will be done as:

```
CALL LESLIL(GETAV,M,N,APPLES,TO,TE,DT,YO,TOLQ,TOLL,IPAR,FWORK,IFLAG,INARR,REARR)
```

where GETAV, declared EXTERNAL, is the subroutine where the user defines $\dot{v} = A(t)v$ and must obey the syntax

```
SUBROUTINE GETAV(M,T,V,VDOT,INARR,REARR)
INTEGER M, INARR(*)
DOUBLE PRECISION T, V(M), VDOT(M), REARR(*)
```

In the above, INARR, REARR are integer and double precision arrays the user can use for communication between the driver and the subroutines.

In the Appendix, we attach sample drivers to illustrate how the codes are called. Next, we briefly review the meaning of the INPUT/OUTPUT to the codes. More extensive documentation can be found in the interface to the codes themselves.

4.2 LESLIS and LESLIL: Input and Options

Typically, the user will write a driver where integration options are specified. In a nutshell, and subject to compatibilities between his/her requests, the user must specify: (1) whether to use a discrete or continuous QR method and which formulas, (2) how to approximate the ν's in (3.2-3.3), (3) if to proceed with fixed or variable stepsizes, and how to perform error control. This communication between the user and the codes is handled through the array IPAR, and by monitoring IFLAG the user knows if the required tasks have been accomplished. The following is an explanation of the INPUT/OUTPUT to LESLIS-LESLIL.

- M: INPUT, the dimension of the problem, must be ≥ 1.
- N: INPUT, the number of the approximate quantities in (4.1-4.2) desired, must have $1 \leq$ N $\leq$ M
- APPLES: OUTPUT only, the approximations to (3.1).
- TO: Initial time on INPUT, final time reached on OUTPUT.
- TE: Final time where approximations to (3.1) are desired. Can have TE $<$ TO or TE $>$ TO.
- DT: INPUT/OUTPUT, the stepsize in fixed stepsize mode, stores the last chosen stepsize in variable stepsize mode.
- YO: An (m, n) matrix which on INPUT contains the ICs to the fundamental matrix (see IPAR(4)). On OUTPUT, it contains the Q-factor at the point we reached.
- TOLQ: INPUT, scalar. This is the local error tolerance on the sup-norm error for the columns of Q.
- TOLL: INPUT, N-dimensional array. These are local error tolerances on the approximation to the Lyapunov exponents.
- IPAR: communication array. Most options can be defaulted, and the most relevant options are the following.
  - IPAR(1): specifies fixed or variable stepsize.

- IPAR(2): specifies if want code to return control to the driver after every successful step, or only at TE.
- IPAR(4): specifies (seeting IPAR(4).NE.0) if user provided ICs on the fundamental matrix Y_0. If not, the code sets the default: $\begin{pmatrix} I_n \\ 0 \end{pmatrix}$.
- IPAR(8): specifies which method and RK formula to use:
  - IPAR(8)=0 (default) is Continuous projected QR method with DP5
  - IPAR(8)=1 is Continuous hybrid QR method with DP5
  - IPAR(8)=2 is Continuous projected QR method with RK38
  - IPAR(8)=3 is Continuous hybrid QR method with RK38
  - IPAR(8)=4 is Discrete QR method with DP5
  - IPAR(8)=5 is Discrete QR method with RK38
- IPAR(9): specifies if the quantities in (3.3) are found by the RK integration on the ν-variables (IPAR(9)=0) or by the composite trapezoidal rule (IPAR(9)=1). Naturally, IPAR(9)=1 can only be required when IPAR(8)=0,1,2,3.
- IPAR(10): subject to compatibility with the chosen method, IPAR(10) specifies on which variables to perform error control. [See Remark 3.2.2]. In the most general form, one may be able to control the local errors on the approximation to the factor Q and/or to the Lyapunov exponents, with error tolerances specified by TOLQ and TOLL. To be precise, IPAR(10)=0 means error control on the Lyapunov exponents only, IPAR(10)=1 means error control on Q only, and IPAR(10)=10 means error control on both the Lyapunov exponents and Q. Naturally, IPAR(10)=1,10 can only be required when IPAR(8)=0,1,2,3.

- FWORK: this array contains all work space.
- IFLAG: communicates to the user if the integration was successfully completed or not (various diagnostics are then produced).
- INARR, REARR: user-defined arrays to communicate parameters between the code and the subroutines GETA or GETAV.

Chapter 5

Getting the spectra

In this Chapter, we explain how one can post-process the information produced by `LESLIS` or `LESLIL` in order to approximate Lyapunov exponents, and Lyapunov spectrum, as well as the Exponential Dichotomy spectrum Σ_{ED}. We also discuss how we can use the information produced by the codes to test for the stability of Lyapunov exponents. Of course, there may be other ways to proceed for these tasks, but our goal is to explain how one can accomplish them using only the averages of the diagonal elements over the current time step, which is information obtained from `LESLIS` and `LESLIL`. To reiterate, the codes return (4.2) or equivalently (4.1). Below, we will often refer just to $\int_0^T B_{ii}(s)ds$, but of course this is the same as $\log(R_{ii}(T))$.

We are interested in the approximation of the Lyapunov and the Exponential Dichotomy spectra, Σ_{ED}. Recall that Σ_{ED} is defined as the set of real values λ for which the shifted system

$$\dot{y}_\lambda \;=\; \big[A(t) - \lambda I\big]\, y$$

does not have Exponential Dichotomy. Recall also that Σ_{ED} is given by the union of at most m non overlapping intervals. In general, we will be interested in approximating the n most dominant intervals. The reason why the approximation of this spectrum is possible is explained in [17, 18]. Presently, it suffices to recall that Σ_{ED} can be obtained from the diagonal of the transformed upper triangular system (2.7): That is, using information which we have been able to compute (either B in (2.7) or the diagonal of R).

Here below, we discuss how in a post-processing step, one can obtain:

- Approximation of Lyapunov spectrum,
- Steklov averages of diagonal elements to approximate Σ_{ED} and assess variability of Lyapunov exponents,
- Steklov differences to infer stability of Lyapunov exponents.

5.1 Post-Processing for the LEs

Approximate Lyapunov exponents are basically returned by the codes `leslis` and `leslil` in the variable `APPLES`. In fact, in `APPLES` the codes return the values (4.1)-(4.2). So, a standard use of the codes is to monitor these values to see if convergence is taking place. If these values are converging, we may conclude that the system is regular and that we have approximated its Lyapunov exponents. On the other hand, if convergence is not taking place, one may want to monitor the variation in the values of `APPLES` on a sufficiently large interval to approximate the

Lyapunov spectral intervals; this is essentially the approach we used in [17] to approximate the spectral intervals.

5.2 Other spectra

To approximate $\Sigma_{\rm ED}$, one may compute Steklov averages. We adopted this approach in [17], and it has met with some success.

First, recall that, for a continuous bounded function f, the Steklov function or Steklov average of f with step $H > 0$ is defined as (see [1, Definition 5.4.1] and [6])

$$f^H(t) = \frac{1}{H}\int_t^{t+H} f(\tau)d\tau. \tag{5.1}$$

Now, given any $H > 0$, for $i = 1, \ldots, n$, consider

$$\alpha_i^H = \inf_t \frac{1}{H}\int_t^{t+H} B_{ii}(s)ds \quad \text{and} \quad \beta_i^H = \sup_t \frac{1}{H}\int_t^{t+H} B_{ii}(s)ds\ . \tag{5.2}$$

We showed in [17, Theorem 8.4] that $[\alpha_i^H, \beta_i^H]$ can be used to approximate the i-the spectral interval of $\Sigma_{\rm ED}$, $i = 1, \ldots, n$. So, here below we explain how one can use the output of our codes to approximate the intervals $[\alpha_i^H, \beta_i^H]$.

Remark 5.2.1 To facilitate approximation of the Sacker-Sell or Exponential Dichotomy spectrum and infer integral separation between consecutive diagonal elements of the upper triangular coefficient matrix function, B, the values returned in `APPLES` at consecutive time levels are employed. In particular, in one step mode we form

$$\int_{t_k}^{t_{k+1}} B_{ii}(s)ds = t_{k+1}[\frac{1}{t_{k+1}}\int_0^{t_{k+1}} B_{ii}(s)ds] - t_k[\frac{1}{t_k}\int_0^{t_k} B_{ii}(s)ds]$$

for all consecutive step levels t_k, t_{k+1}. Postprocessing may then be performed given the value of the current time step $h_k := t_{k+1} - t_k$ and approximations to the values $\int_{t_k}^{t_{k+1}} B_{ii}(s)ds$ for $i = 1, ..., n$. The first step is to use an interpolant so that the integrals of the B_{ii} are approximated over a fixed stepsize. Then appropriate Steklov averages are computed to approximate $\Sigma_{\rm ED}$ and integral separation between consecutive B_{ii}.

We assume now that we have approximations to the integrals of the desired B_{ii} with respect to some fixed step size h. Then choose a Steklov window length H such that $H = N \cdot h$ for some positive integer N. For each i we store and sum the first N consecutive approximations to the integrals of the B_{ii}. When each subsequent integral is read in it the first integral in the sum is subtracted from the sum, the new integral s added to the sum, the old integral is overwritten with the new integral, and (5.2) is easily approximated.

5.3 Integral separation

One more use of the output from `LESLIS`/`LESLIL` is to assess the degree of integral separation of the transformed coefficients' function B in (2.7), and therefore assess the stability of the Lyapunov exponents. Recall that distinct Lyapunov exponents $\lambda_1 > \lambda_2 > \ldots > \lambda_n$ are stable if and only if the functions $B_{ii}, B_{i+1,i+1}$, $i = 1, \ldots, n-1$, are integrally separated.

Recall also that for two bounded functions f_1 and f_2 (say, two consecutive diagonal elements of the upper triangular function B in (2.7)) to be integrally separated means that

$$\int_s^t (f_1(\tau) - f_2(\tau))d\tau \geq a(t-s) - d,\ a > 0,\ d \in I\!R,\ t \geq s\ . \tag{5.3}$$

The importance of Steklov functions resides in the fact that (5.3) can be inferred from the Steklov average of the difference $f_1 - f_2$. This is the content of [1, Lemma 5.4.1]. In other words, f_1 and f_2 are integrally separated if and only if for sufficiently large H their Steklov functions are separated:

$$f_1^H(t) - f_2^H(t) = \frac{1}{H}\int_t^{t+H} (f_1(\tau) - f_2(\tau))d\tau \geq a > 0,\ t \geq 0\,. \tag{5.4}$$

We can use the output of our codes to infer whether or not (5.4) holds in a very similar way to how we approximated Σ_{ED}: The only difference is that the integrals of the B_{ii} are replaced with the difference in the integrals of B_{ii} and $B_{i+1,i+1}$ for $i = 1, ..., n-1$.

Chapter 6

Examples

We present some examples to highlight performance of the codes. Many more examples have been used to test the codes, and are part of the collection of problems in the routine `user.f` which comes with our codes.
`CPU` **times**. All results in this Section refer to computations made on one of the two different computational platforms below.

(1) A Dell Dimension, with a Pentium 4, 2.5 GHz processor, using `g77` GNU compiler with -O optimization option. The `CPU` times in Tables 6.1, 6.2, 6.6, and 6.7, refer to computations done on this environment.

(2) A Dell Inspiron 8200, with a Pentium 4, 1.8 GHz processor, using Watcom Fortrtan 77 compiler with no optimization. The results in Table 6.8, as well as all Figures 6.2–6.13 were generated on this environment.

6.1 Small systems

These are problems of small size, and are used to illustrate comparative performance of the different options.

Example 6.1.1 Consider a linear periodic coefficient problem due to Markus and Yamabe. The linear system

$$\dot{y} = \begin{pmatrix} -1 + \frac{3}{2}\cos^2(t) & 1 - \frac{3}{2}\cos(t)\sin(t) \\ -1 - \frac{3}{2}\sin(t)\cos(t) & -1 + \frac{3}{2}\sin^2(t) \end{pmatrix} y \equiv A(t)y$$

is regular, with Lyapunov exponents $\lambda_1 = \frac{1}{2}$ and $\lambda_2 = -1$, which also form the point spectrum of the system. The eigenvalues of $A(t)$ are constant, $(-1 \pm i\sqrt{7})/4$ and suggest that the zero solution is asymptotically stable, which of course is not true. If we make the orthogonal change of variables $Q(t)w = y$ with

$$Q(t) = \begin{pmatrix} \cos(t) & \sin(t) \\ -\sin(t) & \cos(t) \end{pmatrix}$$

we obtain the system

$$\dot{w} = \begin{pmatrix} 1/2 & 0 \\ 0 & -1 \end{pmatrix} w.$$

We tabulate results in Table 6.1 and illustrate the dependence on `IPAR(8)`, the method employed, `IPAR(10)`, the error control, and the number of steps taken `STEPS`, the number of rejections `REJS`,

Markus and Yamabe Problem. $T = 10^3$.

IPAR(8)	IPAR(10)	TOL	Errμ_1	Errμ_2	STEPS	REJS	CPU
0	0	1.E-4	4.E-4	4.E-4	1957	977	.054
1	0	1.E-4	4.E-4	4.E-4	1608	0	.030
2	0	1.E-4	2.E-4	2.E-4	2105	0	.027
3	0	1.E-4	3.E-4	3.E-4	3099	1	.036
4	0	1.E-4	2.E-6	2.E-6	2418	0	.032
5	0	1.E-4	2.E-5	2.E-5	5501	0	.054
0	1	1.E-4	2.E-5	2.E-5	1323	48	.014
1	1	1.E-4	1.E-4	1.E-4	2109	0	.033
2	1	1.E-4	2.E-6	2.E-6	4234	0	.055
3	1	1.E-4	3.E-4	3.E-4	3099	0	.026
0	10	1.E-4	2.E-5	2.E-5	1323	48	.023
0	0	1.E-8	1.E-7	1.E-7	2706	3	.050
0	1	1.E-8	1.E-9	1.E-9	5005	0	.110
0	10	1.E-8	1.E-9	1.E-9	5005	0	.103

Table 6.1: Comparison of error, steps, rejections, and CPU times for different methods, error control, and tolerances.

and the CPU usage CPU. In the table, the value TOL refer to the common value used for all error tolerances. In this table, as well as in all other tables, we use "exponential notation"; e.g., 1.E-2 means 10^{-2}. It is especially noteworthy to observe the large number of rejections that occur for IPAR(8)$= 0$ with IPAR(10)$= 0$ and TOL$= 10^{-4}$. This highlights the need to control the error on Q as well as the error on the exponents, at least for this value of the tolerance.

Example 6.1.2 We have

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

where

$$D(t) = \mathrm{diag}(\lambda_1, \cos(t), -\frac{1}{2\sqrt{t+1}}, \lambda_4)\,,$$

$\lambda_1 > 0$, $\lambda_4 < 0$, and

$$Q(t) = \mathrm{diag}(1, Q_\beta(t), 1) \cdot \mathrm{diag}(Q_\alpha(t), Q_\alpha(t))\,.$$

We set

$$Q_\gamma(t) = \begin{pmatrix} \cos(\gamma t) & \sin(\gamma t) \\ -\sin(\gamma t) & \cos(\gamma t) \end{pmatrix},\ \alpha = 1,\ \beta = \sqrt{2}.$$

We fix $\lambda_1 = 1$, $\lambda_4 = -10$. This is a regular system, and the Lyapunov exponents are $1, 0, 0, -10$. In Table 6.2 we vary the method and the tolerance. The need to control the error on Q as well as the exponents is again highlighted by the large number of rejections that occur with tolerance of 10^{-4} with the projected DP5 and projected RK38 methods.

We highlight the behavior of the codes on this problem with several figures. In Figure 6.1 we plot the computed Lyapunov exponents as a function of time in a semilog scale. In Figures 6.2–6.11, the values on the x-axis are the negative logarithms (in base 10) of the required error tolerance; all these results were obtained with TOLL=TOLQ and values between 10^{-2} and 10^{-13}. In all figures the legend refers to the combination of IPAR(8)/IPAR(10) we adopted. For example, the heading 2/1 means that the projected method of order 4 was used with error control on Q. For

Quasi Periodic Problem. $T = 10^3$.

IPAR(8)	IPAR(10)	TOL	STEPS	REJS	CPU
0	0	1.E-4	7913	3910	1.306
1	0	1.E-4	7177	3585	1.015
2	0	1.E-4	7544	680	.573
3	0	1.E-4	7634	525	.449
4	0	1.E-4	19470	0	1.405
5	0	1.E-4	42992	0	2.055
0	0	1.E-8	16771	159	1.859
1	0	1.E-8	17674	188	1.616
2	0	1.E-8	35791	0	2.504
3	0	1.E-8	38808	0	2.104
4	0	1.E-8	117341	0	8.511
5	0	1.E-8	418392	0	20.080
0	10	1.E-4	8953	119	.899
0	10	1.E-8	52416	0	5.418

Table 6.2: Comparison of error, steps, rejections, and CPU times for different methods, error control, and tolerances.

these particular problem, all results with `IPAR(10)=10` were identical to those with `IPAR(10)=1`. Finally, all results refer to computation from 0 to time $T = 100$.

In Figures 6.2 and 6.3, we give the error (for the 5th and 4th order methods, respectively) in the truncated exponents $\frac{1}{T}\int_0^T B_{ii}(s)ds$, $i = 1, \ldots, n$, see (4.2). The y-axis is the negative logarithm in base 10 of the error (actually, the absolute error for quantities of magnitude less than 1 and the relative error otherwise). In general, for the continuous QR methods, it is necessary to require error control on Q to obtain approximation of order of accuracy the same as the error tolerance. For tight tolerances, the only methods capable to achieve the desired accuracy are the continuous QR methods implemented with projected schemes and error tolerance on Q.

In Figures 6.4 and 6.5, we give the error in Q. It is interesting to observe that the discrete QR methods achieve a good error control on Q although this is not explicitly enforced. For all but the tighter tolerances, the continuous QR methods give error on Q of size of the error tolerance.

Figures 6.6 and 6.7 give the number of successful steps in function of the error tolerances, and for the different error control options, for methods of order 4 and 5. Figures 6.8 and 6.9, instead, give the percentage of rejections. We remark that there were no rejections for the discrete QR methods, both at order 4 and 5, and for all different error tolerances. By looking at Figures 6.6 and 6.7, we can see that the discrete QR methods take more steps than the continuous ones, and that the continuous methods implemented either as projected or hybrid take about the same number of steps. It is revealing to observe that the number of steps grows as the theory would demand for methods of order 5, respectively 4: In fact, for the runs with the dsicrete QR method, or the continuous ones with error control on Q, the slopes of the graphs in Figures 6.6 and 6.7 are almost exactly 1/5 and 1/4.

Looking at Figures 6.8 and 6.9, it is apparent that the continuous methods, without requiring error control on Q are not efficient for large/intermediate values of the error tolerances. Indeed, in these cases, the large number of rejections leads to inefficient computations as reflected in Figures 6.10 and 6.11 which show the total CPU time. To be precise, on the y-axis is the $\log_{10}$ of the `CPU` times, normalized by the minimum `CPU` time observed (it is 0.07" for these figures). Thus, for

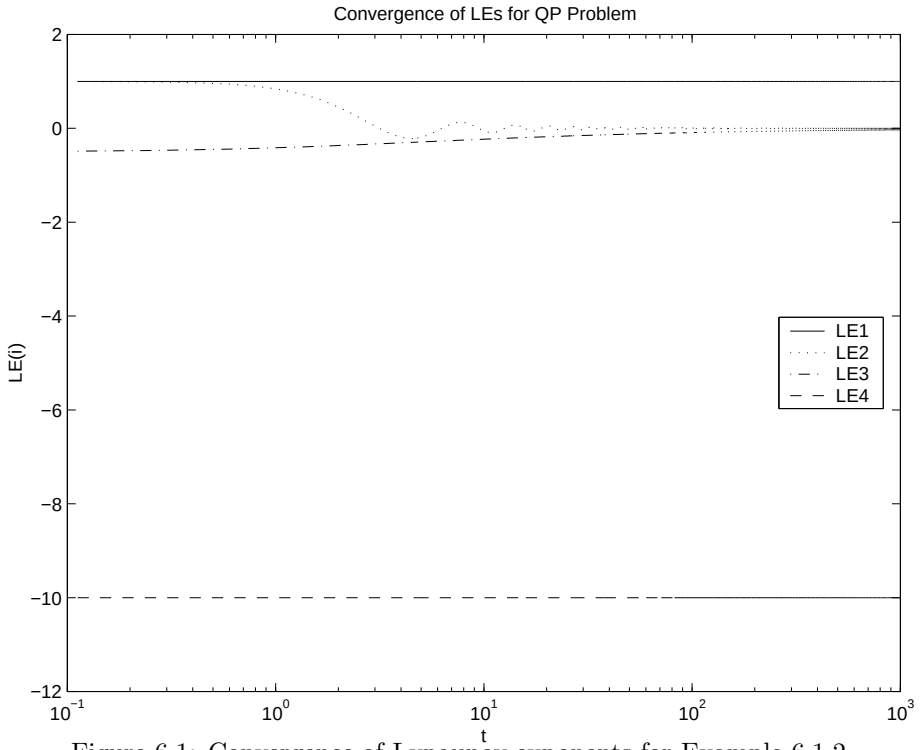

Figure 6.1: Convergence of Lypaunov exponents for Example 6.1.2.

example, the most expensive run (in Figure 6.11, the discrete QR method with `TOL`=1.E-13) takes about 76.7 true seconds: $0.07 \times 10^{3.04}$. Looking at these Figures, we observe that the expense (as measured by the `CPU` times) grows exponentially as the tolerances decrease, for all methods and options.

Finally, for comparison purposes, we report on results for the 5th order schemes with the continuous QR methods using the trapezoidal rule option to approximate the integrals in (3.3). In Figures 6.12 and 6,13, these results are compared with those in which the integrals are approximated also at 5th order by the same RK scheme used to approximate Q. Using the trapezoidal rule only gives the accuracy of a second order method. Naturally, using the trapezoidal rule is less expensive, but the savings are somewhat marginal: It appears that the cost of using the trapezoidal rule is about the same as that of not using it, but with tolerances values 10 times as large!

Although different problems produce different shadings in the outcomes, the overall picture on many problems is similar to that we presented in this example. Thus, we recommend that:
• With adaptive error control, the continuous QR methods must be used enforcing error control on Q (we recommend `IPAR(10)=10`).
• For tight error tolerances, the best option is `IPAR(8)=0` while for low error tolerances `IPAR(8)=2` is appropriate.
• The trapezoidal rule option is of interest only for truly large time T, in which case the convergence to the exponents is anyhow dominated by the factor $\frac{1}{T}$.

Example 6.1.3 This is an example where there are continuous spectra $\Sigma_{\rm L}$ and $\Sigma_{\rm ED}$. We replace $D(t)$ in Example 6.1.2 with

$$D(t) = \mathrm{diag}(f(\tau(t)) + 4, f(\tau(t)), f(\tau(t)) - 1, f(\tau(t)) - 4)$$

where $f(x) = \cos(x) + \sin(x)$, and $\tau(t) = \ln(t + 1)$. Then we have

$$\Sigma_{\rm L} = \bigcup_{i=1}^{4}[a_i, b_i] = [3, 5] \cup [-1, 1] \cup [-2, 0] \cup [-5, -3],$$
$$\Sigma_{\rm ED} = \bigcup_{i=1}^{4}[\alpha_i, \beta_i] = [4 + \sqrt{2}, 4 - \sqrt{2}] \cup [-\sqrt{2}, \sqrt{2}] \cup [-1 - \sqrt{2}, -1 + \sqrt{2}] \cup [-4 - \sqrt{2}, -4 + \sqrt{2}].$$

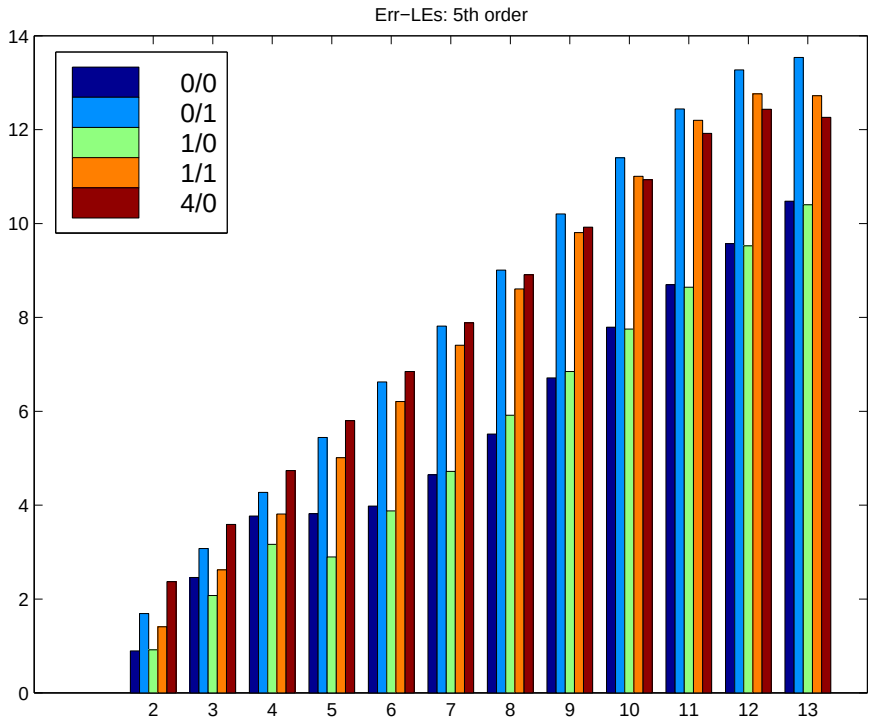

Figure 6.2: Error on truncated exponents for Example 6.1.2: 5th order methods.

In this problem we compare the exact and computed values of the spectral intervals. To approximate $\Sigma_{\rm L}$ we compute for $0 < \tau_0 < T$,

$$\inf_{T \geq t \geq \tau_0} \frac{1}{t} \int_0^t B_{ii}(s)ds \quad \text{and} \quad \sup_{T \geq t \geq \tau_0} \frac{1}{t} \int_0^t B_{ii}(s)ds$$

and to approximate $\Sigma_{\rm ED}$ we compute for $0 < H < T$,

$$\inf_{T-H \geq t \geq 0} \frac{1}{H} \int_t^{t+H} B_{ii}(s)ds \quad \text{and} \quad \sup_{T-H \geq t \geq 0} \frac{1}{H} \int_t^{t+H} B_{ii}(s)ds$$

Tables 6.3, 6.4, and 6.5 record our approximations to $\Sigma_{\rm L}$, $\Sigma_{\rm ED}$, and the integral separation of consecutive diagonal elements of B, respectively. We vary τ_0 for $\Sigma_{\rm L}$, and H for $\Sigma_{\rm ED}$ and integral separation of the diagonal elements. From Table 6.4, we observe that a time interval of length $T = 10^6$ is required to get a good approximation of the exponential dichotomy spectrum with Steklov window length of $H = 10^3$; smaller values of T and/or H lead to inaccurate results. This highlights the general difficulty of selecting the time interval and the Steklov window: We are not aware of any systematic, foolproof, way to select these values.

6.2 Large systems

These are large problems for which we are interested in assessing the relative merits of `LESLIL` versus `LESLIS`.

Example 6.2.1 We consider linearization about a traveling wave solution of the parabolic Nagumo equation

$$u_t = \epsilon^2 u_{xx} - f(u),\ x \in I\!R, t \geq 0,\ f(u) = u(u-1)(u-a),\ a \in (0,1). \tag{6.1}$$

Σ_{L}. $T = 10^5$. IPAR(8)=0. IPAR(10)=10. TOL=10^{-4}

τ_0	i	$[a_i, b_i]$
10^1	1	$[2.99105481, 5.00054981]$
10^1	2	$[-1.0090923, 1.00040134]$
10^1	3	$[-2.00916475, 0.000329240246]$
10^1	4	$[-5.00922297, -2.99972667]$
10^3	1	$[3.00014447, 5.00054981]$
10^3	2	$[-1.00000376, 1.00040134]$
10^3	3	$[-2.00007593, 0.000329240246]$
10^3	4	$[-5.00013189, -2.99972667]$

Table 6.3: Computed approximations to Lyapunov spectrum for Example 6.1.3.

Σ_{ED}. IPAR(8)=0. IPAR(10)=10. TOL=10^{-4}

T/H	i	$[\alpha_i, \beta_i]$
$10^5/10^1$	1	$[2.58592912, 5.41437546]$
$10^5/10^1$	2	$[-1.41420998, 1.41423018]$
$10^5/10^1$	3	$[-2.41428279, 0.414156513]$
$10^5/10^1$	4	$[-5.4143401, -2.58589528]$
$10^5/10^3$	1	$[2.58602723, 5.37346259]$
$10^5/10^3$	2	$[-1.41412084, 1.37331392]$
$10^5/10^3$	3	$[-2.41419304, 0.373241827]$
$10^5/10^3$	4	$[-5.41424902, -2.62681384]$
$10^6/10^3$	1	$[2.58602723, 5.41437509]$
$10^6/10^3$	2	$[-1.41412084, 1.41422661]$
$10^6/10^3$	3	$[-2.41419304, 0.414154302]$
$10^6/10^3$	4	$[-5.41424902, -2.58590173]$

Table 6.4: Computed approximations to exponential dichotomy spectrum for Example 6.1.3.

Integral Separation. IPAR(8)=0. IPAR(10)=10. TOL=10^{-4}

T/H	a_1	a_2	a_3
$10^5/10^1$	4.00011782	1.00006544	3.00003553
$10^5/10^3$	4.00014751	1.00007192	3.00005541
$10^6/10^3$	4.00014751	1.00007186	3.00005531

Table 6.5: Computed approximations to integral separation for Example 6.1.3.

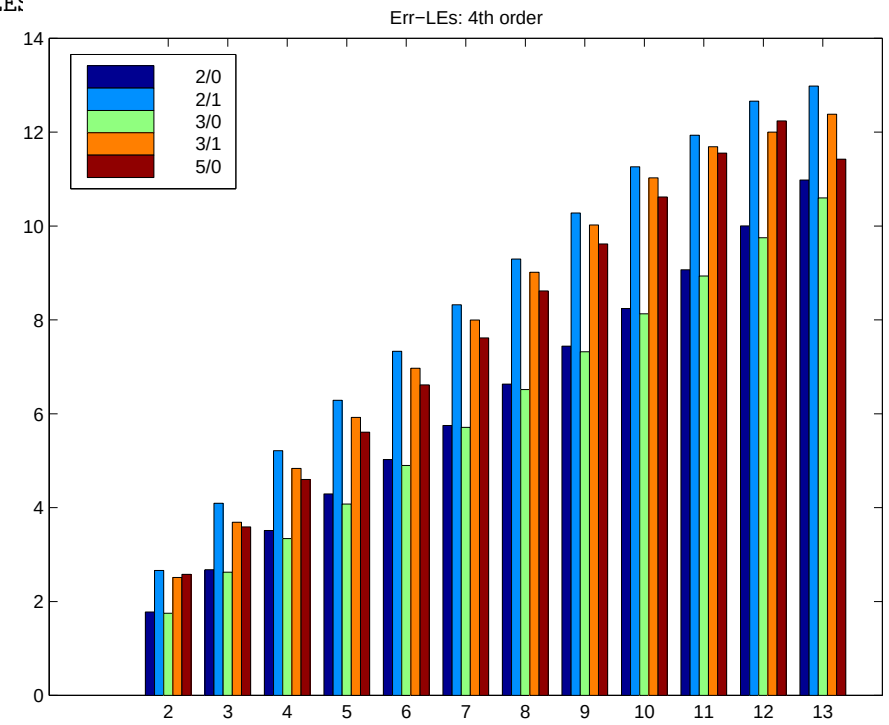

Figure 6.3: Error on truncated exponents for Example 6.1.2: 4th order methods.

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)

$$\left\{ \begin{array}{rcl} \phi(\xi) & = & \frac{1}{2}[1 \mp \tanh(\sqrt{\frac{1}{8\epsilon^2}} \cdot \xi)] \\ c & = & \pm(1-2a)\sqrt{\frac{\epsilon^2}{2}} \ . \end{array} \right. \tag{6.2}$$

If we linearize (6.1) about (6.2) we obtain the linear partial differential equation

$$\eta_t = \epsilon^2 \eta_{xx} - f'(u_{TW}(x,t))\eta \, . \tag{6.3}$$

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 two discretizations: (i) standard finite (centered) differences (large, sparse, matrices), and (ii) spectral (large, but dense, matrices). Thus, for the grid points $x_j = -1 + 2(j-1)/m$, $j = 1, \ldots, m$, we seek an approximate solution $z(t)$ by requiring that (6.3) is satisfied at the grid points:

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

For spectral discretizations, 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 $Z = (z(x_1,t), \ldots, z(x_m,t))^T$:

$$\frac{dZ}{dt} = A_D(t)Z \ , \ A_D(t) = \epsilon^2 A - \mathrm{diag}(f'(u_{TW}(x_j,t)), \ j = 1, \ldots, m \ ) \, . \tag{6.4}$$

For our experiments we set the wave speed $c = 10^{-1}$, the detuning parameter $a = 9/16$, and the diffusion coefficient $\epsilon^2 = 1.28$.

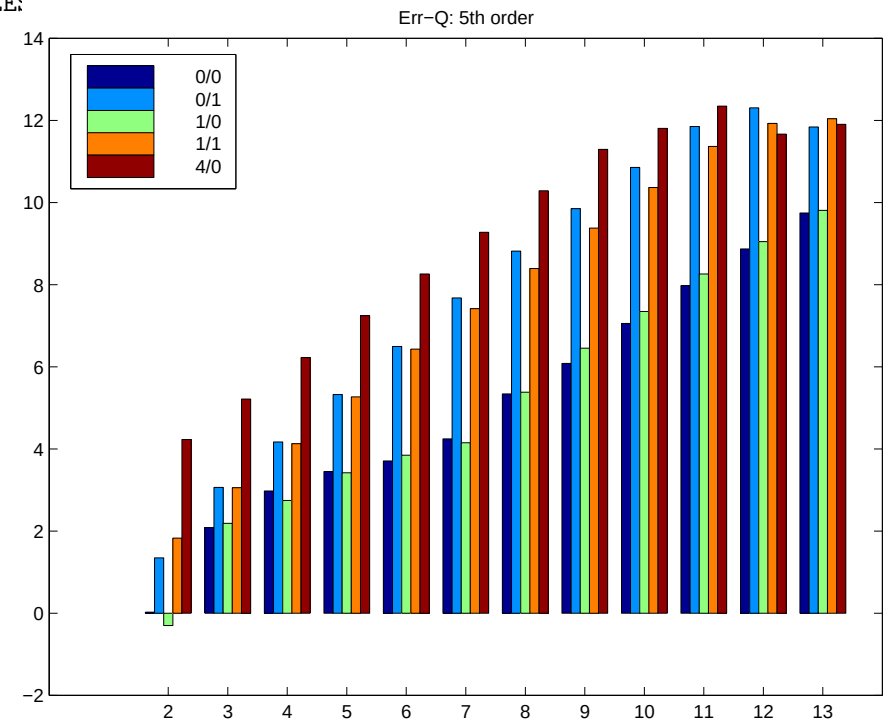

Figure 6.4: Error in Q for Example 6.1.2: 5th order methods.

Results are tabulated in Table 6.6 The savings due to using `LESLIL`, the code for large systems, are clear. Finally, in Figure 6.14, we show the convergence behavior of the two largest Lyapunov exponents of the spectral, respectively finite differences, discretization in fucntion of time. Notice that all these Lyapunov exponents converge nicely in time, but the second Lyapunov exponent of the finite difference discretization is far (for the given value of $m = 32$) from the exact value since the finite difference approximation to the eigenvalues of the second derivative are far from exact.

Example 6.2.2 Consider

$$\dot{u} = a(t)Au + b(t)u, \;\; t \geq t_0 \equiv 1, \;\; u(\cdot) \in I\!R^m \tag{6.5}$$

where A is orthogonally diagonalizable and $a(\cdot), b(\cdot)$ are bounded and continuous. Then we can write $A = QDQ^T$, $D = \mathrm{diag}(\lambda_j, \;\; j = 1, ..., m)$ and after change of variables, consider the scalar problems

$$\dot{x}_j = [a(t)\lambda_j + b(t)]x_j \;\; , \;\; j = 1, \ldots, m \;\; . \tag{6.6}$$

As a specific example, we take λ_j's to be the different eigenvalues of the spectral, respectively finite difference, discretization of Example 6.2.1. That is, we take

- Spectral: $\lambda_j = -(j\pi)^2, j = 0, 1, \ldots, m-1$.
- Finite Difference: $\lambda_j = \frac{m^2}{2}(\cos(2j\pi/m) - 1), j = 0, \ldots, m-1$.

Moreover, we take $a(t) = \kappa + f(\theta_1(t))$ and $b(t) = f(\theta_2(t))$ where

$$f(t) = \cos(t) + \sin(t), \;\; \theta_1(t) = \ln(t), \;\; \theta_2(t) = \ln(\omega t), \;\; \omega > 0 \tag{6.7}$$

so that $\theta_2(t) = \theta_1(t) + \ln(\omega)$. Here $\kappa > \sqrt{2}$ implies $a(t) > 0$.

We set $\kappa = 2$ and $\omega = e^\pi$.

Σ_L:

$$\overline{\mu}_j - \kappa\lambda_j = \kappa\lambda_j - \underline{\mu}_j = \sqrt{\alpha^2 + \beta^2} = (\lambda_j^2 + 2\lambda_j \cos(\ln(\omega)) + 1)^{1/2}. \tag{6.8}$$

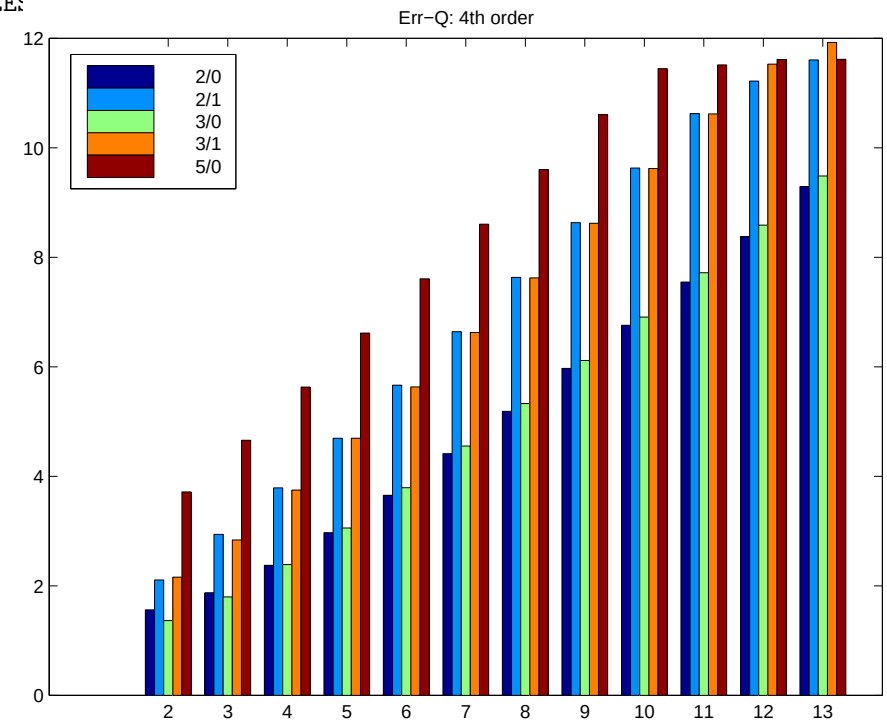

Figure 6.5: Error in Q for Example 6.1.2: 4th order methods.

Σ_{ED}:

$$\overline{\gamma}_j - \kappa\lambda_j = \kappa\lambda_j - \underline{\gamma}_j = \sqrt{2}\sqrt{\alpha^2+\beta^2} = \sqrt{2}(\lambda_j^2 + 2\lambda_j \cos(\ln(\omega)) + 1)^{1/2}. \tag{6.9}$$

Table 6.7 illustrates the savings of using the large option. In Figure 6.15 we show the convergence in time of the four largest upper Lyapunov exponents, $b_i = \sup_{t\geq\tau_0} \int_0^t B_{ii}(s)ds$ for $i = 1, ..., 4$ with $\tau_0 = 10$, for the finite difference discretization.

6.3 Systems with Symmetries

At times, one has to find Lyapunov exponents of systems that have special symmetries, e.g. Hamiltonian systems. A typical byproduct of these symmetries is that the Lyapunov exponents enjoy themelseves special symmetries. For example, Hamiltonian systems have Lyapunov spectra which are symmetric with respect to the origin. Of course, in these cases, one should take full advantage of the symmetries and approximate fewer Lyapunov exponents, while recovering the entire set of exponents exploiting the underlying symmetries.

Example 6.3.1 This is an example from [12]. We have the coefficients' matrix

$$A(t) = \begin{pmatrix} 0 & 2 & -1 & \frac{1}{1+t} & 1 & 2 \\ -2 & 0 & \frac{1}{1+t} & 5 & \cos(t) & 4 \\ 1 & -\frac{1}{1+t} & 0 & 2 & -2 & 1 \\ -\frac{1}{1+t} & -5 & -2 & 0 & -4 & \cos(t) \\ 1 & \cos(t) & -2 & -4 & 0 & \sin(t) \\ 2 & 4 & 1 & \cos(t) & -\sin(t) & 0 \end{pmatrix}, \qquad t \geq 0\ .$$

Observe that we have

$$A^T(t)C + CA(t) = 0$$

Traveling Wave Problem. $T = 10^1$. IPAR(8)=0. IPAR(10)=10. TOL=10^{-4}.

DISC.	L/S	M	N	CPU
SPEC	L	128	4	5.65
FD	L	128	4	1.15
SPEC	S	128	4	58.30
FD	S	128	4	3.80
SPEC	L	64	4	0.70
FD	L	64	4	0.15
SPEC	S	64	4	4.05
FD	S	64	4	0.28
SPEC	L	64	16	3.70
FD	L	64	16	0.87
SPEC	S	64	16	6.15
FD	S	64	16	1.03

Table 6.6: Comparison of CPU times for different discretizations, large/small option, and number of exponents.

Linear Parabolic Problem. $T = 10^2$. $M = 32$. IPAR(8)=0. IPAR(10)=10.

DISC.	L/S	N	TOL	CPU
SPEC	L	4	1.E-4	35.39
FD	L	4	1.E-4	7.32
SPEC	S	4	1.E-4	111.76
FD	S	4	1.E-4	11.88
SPEC	L	4	1.E-8	35.38
FD	L	4	1.E-8	7.32
SPEC	S	4	1.E-8	111.71
FD	S	4	1.E-8	11.87
FD	L	8	1.E-8	21.49
FD	L	16	1.E-8	48.34
FD	S	8	1.E-8	193.58
FD	S	16	1.E-8	225.29
FD/TRAP	L	8	1.E-8	16.19
FD/TRAP	L	16	1.E-8	33.09

Table 6.7: Comparison of CPU times for different discretizations, large/small option, number of exponents, and tolerance.

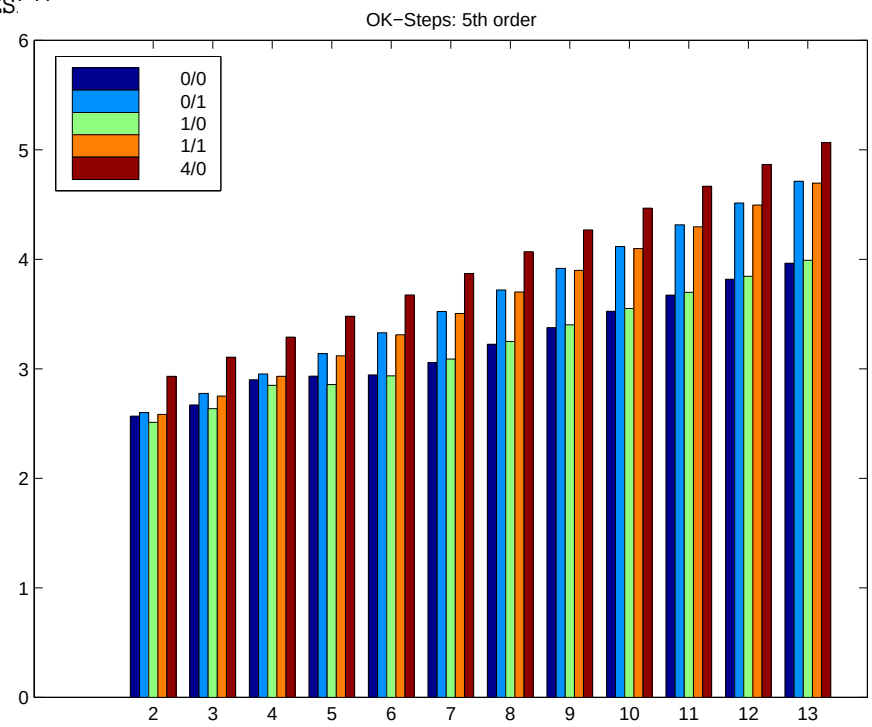

Figure 6.6: Successful steps for Example 6.1.2: 5th order methods.

with

$$C = \begin{pmatrix} Q \otimes I_2 & 0 \\ 0 & -Q \end{pmatrix}$$

where $Q = \begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{pmatrix}$, $0 < \phi < \frac{\pi}{2}$.

The system is regular, and there are two zero and two possibly nonzero Lyapunov exponents, symmetric with respect to the origin, each of multiplicity 2. That is, only one Lyapunov exponent really needs to be computed. At four digits, the two nonzero Lyapunov exponents are $\{\pm 3.027\}$. Results of selected runs are summarized in Table 6.8. All these results have been obtained with error tolerances all equal to 10^{-8}. It is worthwhile noticing that the time required to compute one Lyapunov exponents is almost exactly 1/6-th of the time required to approximate all six exponents.

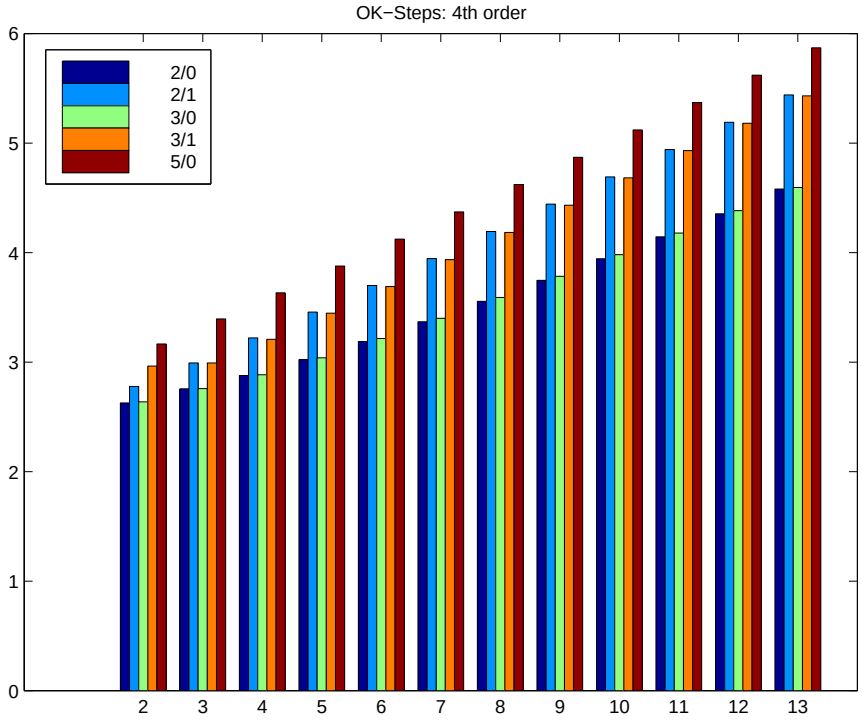

Figure 6.7: Successful steps for Example 6.1.2: 4th order methods.

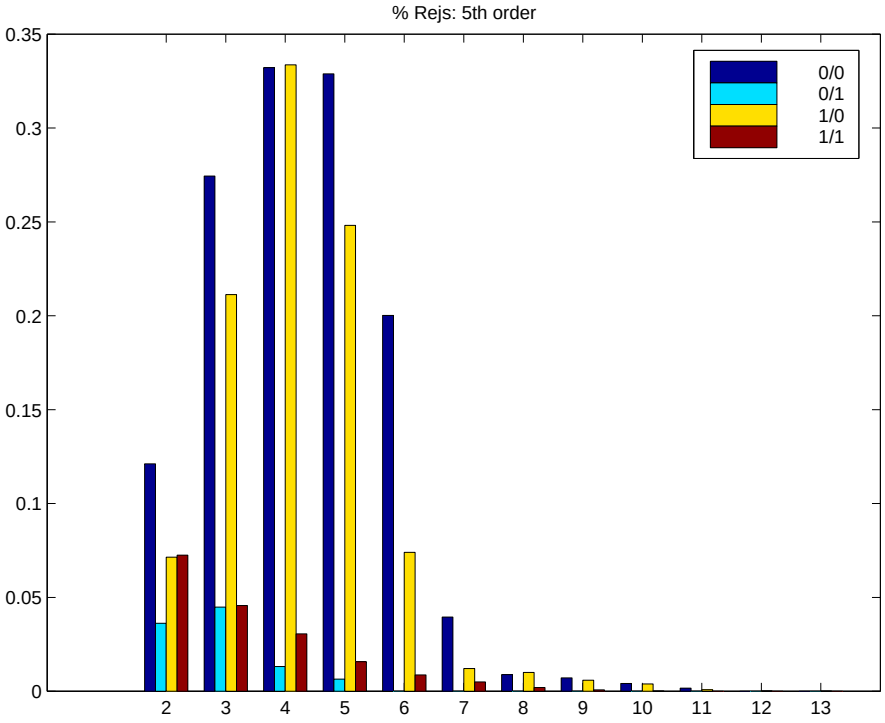

Figure 6.8: Percentage of rejections for Example 6.1.2: 5th order methods.

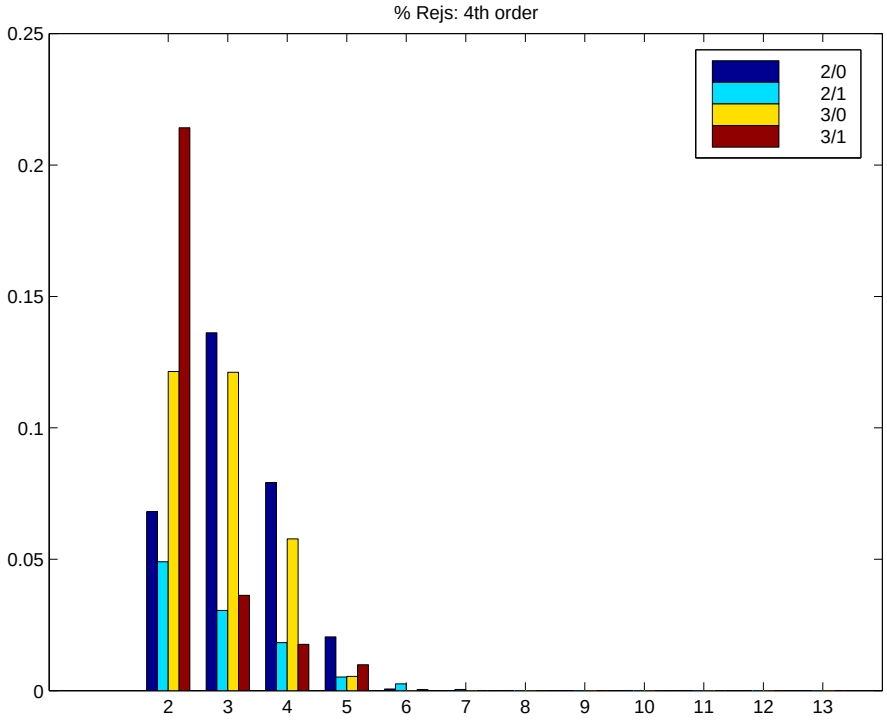

Figure 6.9: Percentage of rejections for Example 6.1.2: 4th order methods.

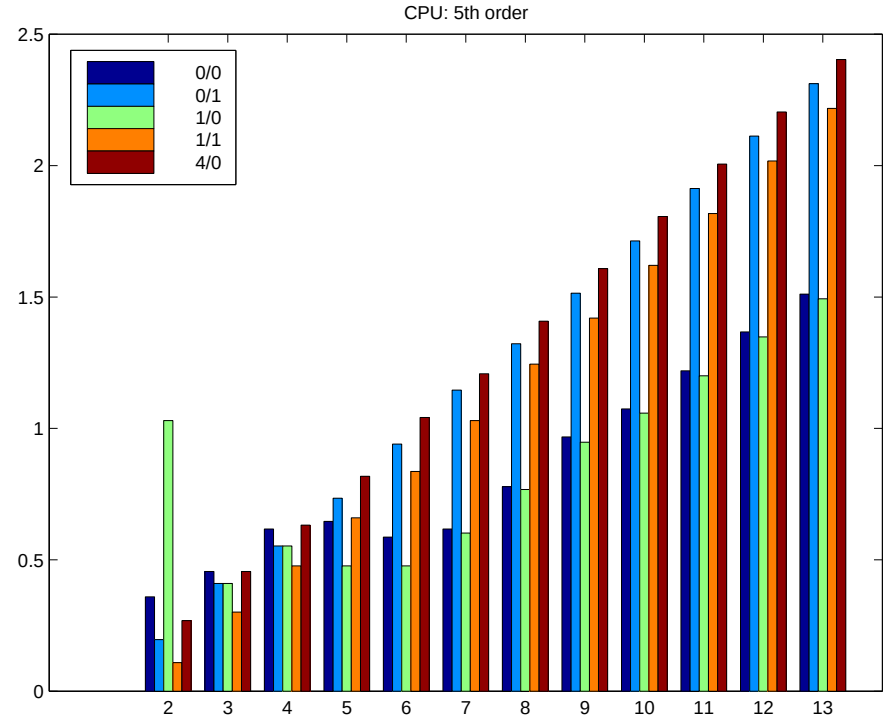

Figure 6.10: CPU usage for Example 6.1.2: 5th order methods.

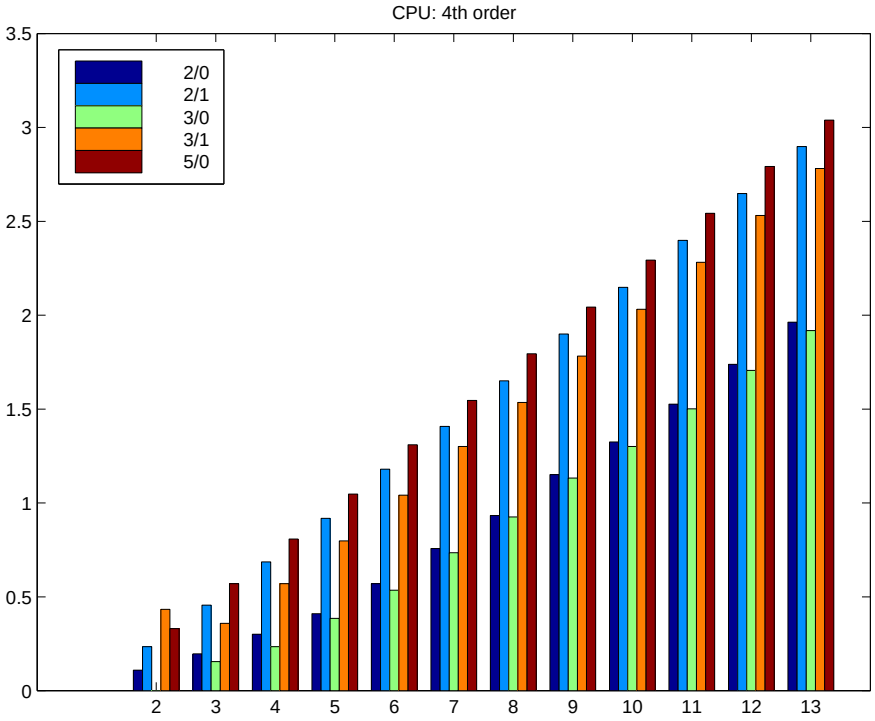

Figure 6.11: CPU usage for Example 6.1.2: 4th order methods.

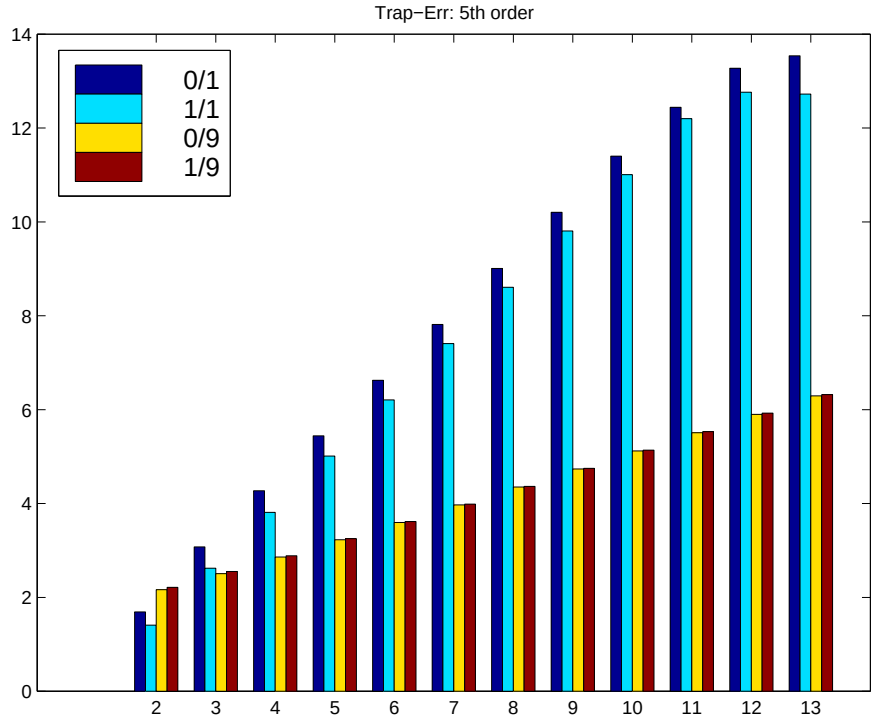

Figure 6.12: Error comparison when using the trapezoidal rule. Example 6.1.2, 5th order methods.

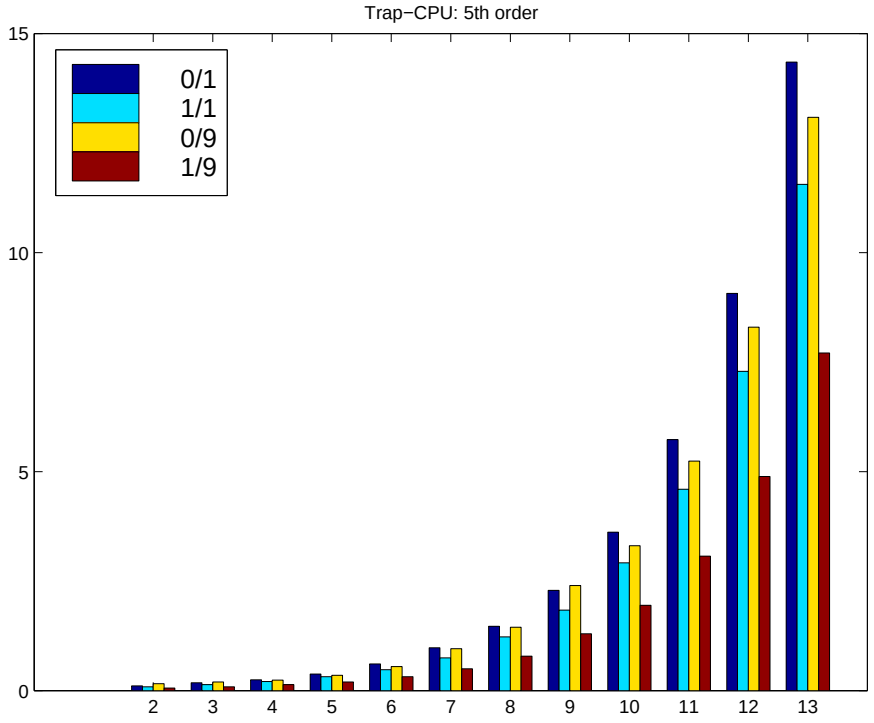

Figure 6.13: CPU comparison when using the trapezoidal rule. Example 6.1.2, 5th order methods.

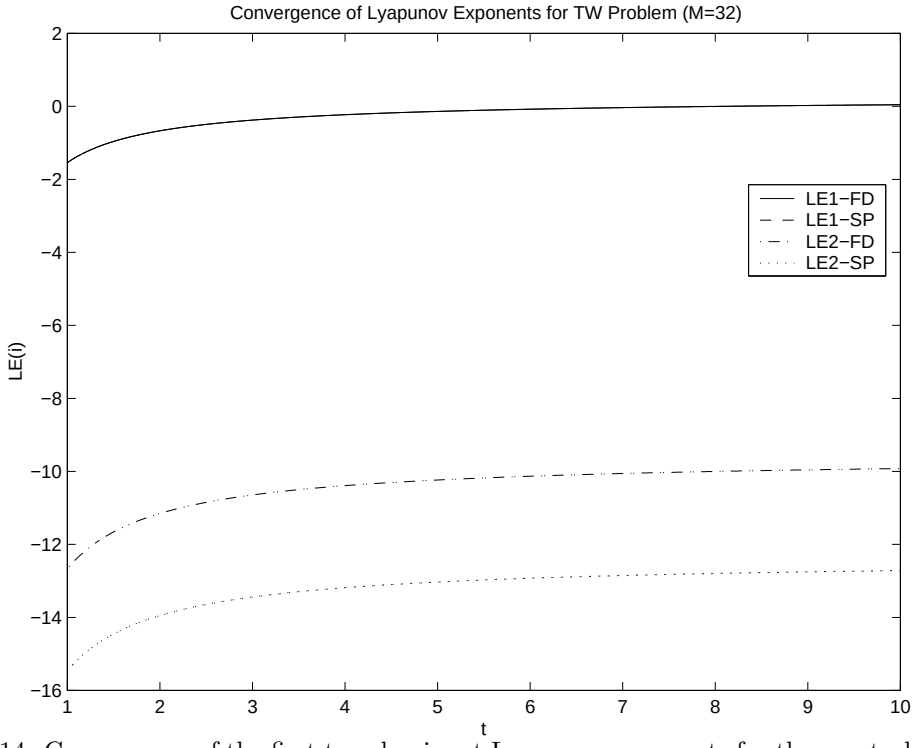

Figure 6.14: Convergence of the first two dominant Lyapunov exponents for the spectral and finite difference discretizations: Example 6.2.1.

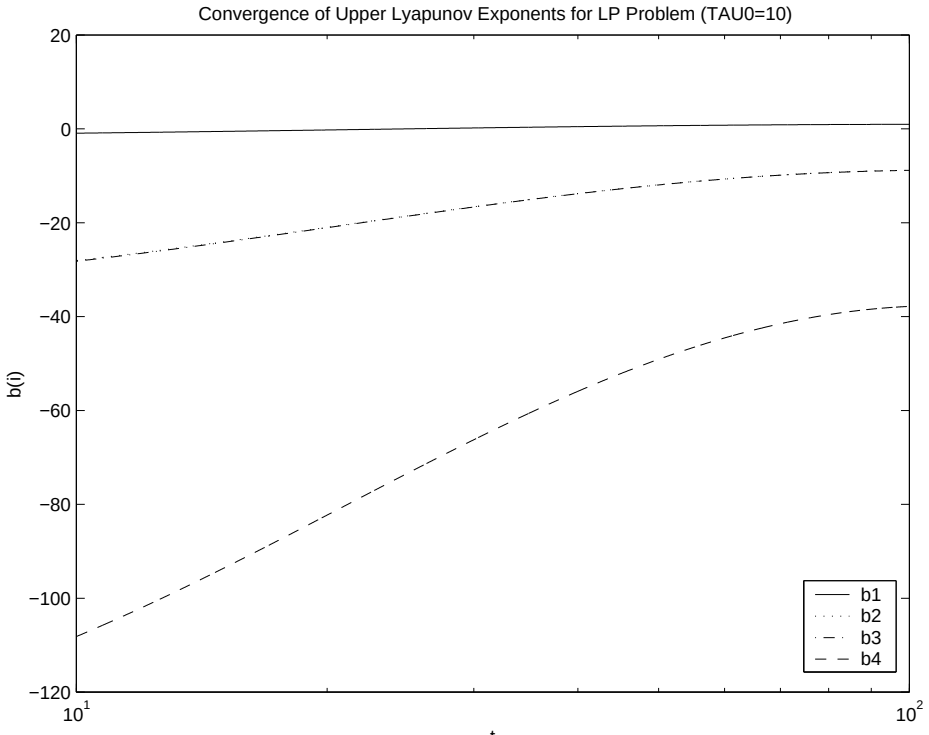

Figure 6.15: Convergence of the first four upper Lyapunov exponents for the finite difference discretization: Example 6.2.2.

Problem with Symmetries. `IPAR(10)`=10 when `IPAR(8)=0`.

`IPAR(8)`	`T`	N	λ_1	`CPU`
0	1.E2	4	3.0044611	1.6
0	1.E3	4	3.0260058	15.5
0	1.E4	4	3.0276900	150.3
0	1.E5	4	3.0276502	1490.9
0	1.E2	1	3.0044611	0.35
0	1.E3	1	3.0260058	2.7
0	1.E4	1	3.0276900	26.4
0	1.E5	1	3.0276502	250.1
4	1.E3	4	3.0260058	9.7
4	1.E5	4	3.0276502	958.7
4	1.E3	1	3.0260058	2.4
4	1.E5	1	3.0276502	233.54

Table 6.8: Comparison of λ_1 and CPU times varying the method used, the final time, and the number of exponents.

Chapter 7

Appendix

7.1 RK coefficients

These are the embedded RK pairs we used. The lower order scheme is the one relative to the weights $\widehat{b}$.

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
$\widehat{b}$	$\frac{1}{12}$	$\frac{1}{2}$	$\frac{1}{4}$	0	$\frac{1}{6}$

3/8-th Runge-Kutta 4-3 pair: `RK38`.

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
$\widehat{b}$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Dormand–Prince 5-4 pair: `DP5`.

7.2 Drivers

Here we give sample drivers for calling the codes. The drivers refer to the Marcus–Yamabe problem, Example 6.1.1.

7.2.1 Driver for LESLIS

```
C ... SAMPLE DRIVER FOR LESLIS ...
      PROGRAM DRIV
C DECLARE VARIABLES NEEDED BY LESLIS
      EXTERNAL GETA
      INTEGER M, N
      PARAMETER (M=2,N=2)
      DOUBLE PRECISION T0, TE, DT, TOLL(N), APPLES(N), Y0(M,N)
      INTEGER IPAR(13), IFLAG, IFDIM,  INARR
      PARAMETER(IFDIM=M*M+11*M*N+8*N+63)
      DOUBLE PRECISION FWORK(IFDIM), REARR, TOLQ
      INTEGER I
      DOUBLE PRECISION ACC

CCCCCCC INPUT: ARRAY IPAR, USE DEFAULT VALUES
C FOR IPAR(1,2,4,8,9)=0.  THAT IS:
C VARIABLE STEP SIZE, FROM T0 TO TE, DEFAULT ICS ON Y0,
C METHOD IS PROJECTION/DP5, AND
C APPROXIMATE EXPONENTS BY NU-INTEGRATION
      T0 = 0.0D0
      TE = 1.0D+1
CCCCCCC IT IS VERY FIRST CALL
      IPAR(3) = 1
CCCCCCC FWORK DIMENSION
      IPAR(5)=IFDIM
C USE ERROR CONTROL ON LEs AND ON Q
      IPAR(10)=10
C ERROR TOLERANCES
      DO 2 I=1,N
         TOLL(I) = 1.D-8
 2    CONTINUE
      TOLQ=1.D-8
C INITIALIZE CODE THE VERY FIRST TIME
      CALL INIT(M,N,IPAR,T0,TE,DT,TOLQ,TOLL,FWORK,IFLAG)
      IF (IFLAG.NE.0) THEN
         PRINT *, 'IFLAG = ', IFLAG
         STOP
      END IF
C INTEGRATE FROM T0 TO TE RETURNING OUTPUT EVERY 10 UNITS
 20   CONTINUE
C CALL MAIN ROUTINE
      CALL LESLIS(GETA,M,N,APPLES,T0,TE,DT,
     *            Y0,TOLQ,TOLL,IPAR,FWORK,IFLAG,INARR,REARR)
      IF (IFLAG.NE.0) THEN
         PRINT *, 'IFLAG = ', IFLAG
```

```
         PRINT*, ' CURRENT T = ', T0,'  LYAP-EXPS ARE '
         WRITE(*,25) (APPLES(I),I=1,N)
         STOP
      ENDIF
      PRINT *, 'AT T= ',T0,'  LYAP-EXPS ARE '
      WRITE(*,25) (APPLES(I),I=1,N)
 25   FORMAT (2X,4(E16.8,2X))
      PRINT *
      IF (T0.LT.100.0D0) THEN
         TE=TE+10.D0
         GOTO 20
      ENDIF
      ACC=0.0D0
      DO 26 I=1,N
 26      ACC=ACC+APPLES(I)
      PRINT*, ' SUM OF THE LES IS ',ACC
      PRINT *
C PRINT SOME STATISTICS
      PRINT *, 'STEPS REJECTED = ', IPAR(12)
      PRINT *, 'NUMBER OF STEPS = ', IPAR(13)
      STOP
      END

C ... and this is the sample GETA ....

      SUBROUTINE GETA(M,T,AMAT,INARR,REARR)
      INTEGER M, INARR(*)
      DOUBLE PRECISION T, REARR(*), AMAT(M,M)

CCCCCCC PROBLEM MARCUS-YAMABE (N=2)
      AMAT(1,1) = -1.D0+1.5D0*DCOS(T)*DCOS(T)
      AMAT(2,2) = -1.D0+1.5D0*DSIN(T)*DSIN(T)
      AMAT(1,2) = 1.D0-1.5D0*DSIN(T)*DCOS(T)
      AMAT(2,1) = -1.D0-1.5D0*DSIN(T)*DCOS(T)
      RETURN
      END
```

7.2.2 Driver for LESLIL

The driver for LESLIL is essentially identical to the one above with the call to LESLIL replacing that to LESLIS and the following routine GETAV replacing GETA.

```
C ... and this is the sample GETAV ....

      SUBROUTINE GETAV(M,T,V,VDOT,INARR,REARR)
      INTEGER M, INARR(*)
      DOUBLE PRECISION T, V(M), VDOT(M), REARR(*)

CCCCCCC PROBLEM MARCUS-YAMABE (N=2)
      VDOT(1) = (-1.D0+1.5D0*DCOS(T)*DCOS(T))*V(1)+
     1           (1.D0-1.5D0*DSIN(T)*DCOS(T))*V(2)
```

```
      VDOT(2) = (-1.D0-1.5D0*DSIN(T)*DCOS(T))*V(1)+
     1           (-1.D0+1.5D0*DSIN(T)*DSIN(T))*V(2)
      RETURN
      END
```

Bibliography

[1] L. Ya. Adrianova. *Introduction to Linear Systems of Differential Equations*, volume Translations of Mathematical Monographs 146. AMS, Providence R.I., 1995.

[2] E. Anderson. *LAPACK User's Guide*. SIAM, Philadelphia, 1992.

[3] G. Benettin, L. Galgani, A. Giorgilli and J.-M. Strelcyn, "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** (1980), pp. 9-20, 21-30.

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

[5] B.F. Bylov and N.A. Izobov. Necessary and sufficient conditions for stability of characteristic exponents of a linear system. *Differentsial'nye Uravneniya*, 5:1794–1903, 1969.

[6] B.F. Bylov, R.E. Vinograd, D.M. Grobman, and V.V. Nemyckii. *The theory of Lyapunov exponents and its applications to problems of stability*. Nauka Pub., Moscow, 1966.

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

[8] E. Celledoni, and B. Owren. A class of intrinsic schemes for orthogonal integration. *SIAM J. Numer. Anal.*, 40:2069–2084, 2002.

[9] F. Christiansen and H. H. Rugh. Computing Lyapunov spectra with continuous Gram-Schmidt orthonormalization. *Nonlinearity*, 10:1063–1072, 1997.

[10] W. A. Coppel. *Stability and Asymptotic Behavior of Differential Equations*, volume Heath Mathematical Monographs. Heath & Co., Boston, 1965.

[11] W. A. Coppel. *Dichotomies in Stability Theory*, volume Lecture Notes in Mathematics Vol. 629. Springer-Verlag, 1978.

[12] L. Dieci and L. Lopez. Lyapunov exponents of systems evolving on quadratic groups. *SIAM J. Mat. Anal. Applic.*, 24:1175–1185, 2003.

[13] L. Dieci, M. Jolly, and E. S. Van Vleck. `LESNLS` and `LESNLL`: Codes for approximating Lyapunov exponents of nonlinear systems. See: http://www.math.gatech.edu/∼dieci.

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

[15] L. Dieci, R. D. Russell, and E. S. Van Vleck. Unitary integrators and applications to continuous orthonormalization techniques. *SIAM J. Numer. Anal.*, 31:261–281, 1994.

[16] L. Dieci and E. Van Vleck. Computation of orthonormal factors for fundamental solution matrices. *Numerische Mathematik*, 83:599–620, 1999.

[17] L. Dieci and E. Van Vleck. Lyapunov spectral intervals: theory and computation. *SIAM J. Numer. Anal.*, 40:516–542, 2002.

[18] L. Dieci and E. Van Vleck. Lyapunov and Sacker-Sell spectral intervals, (2003) submitted.

[19] L. Dieci and E. Van Vleck. Orthonormal integrators based on Householder and Givens transformations. Future Generation Computer Systems, 19:363–373, 2003.

[20] L. Dieci and E. Van Vleck. `QRINT`: a collection of `FORTRAN` routines for orthonormal integration, 2001. Public Domain: http://www.math.gatech.edu/∼dieci.

[21] L. Dieci and E. Van Vleck. Lyapunov and other spectra: a survey. In D. Estep and S. Tavener, editors, *Preservation of Stability under Discretization*. SIAM, Philadelphia, 2002.

[22] L. Dieci and E. S. Van Vleck. Computation of a few Lyapunov exponents for continuous and discrete dynamical systems. *Applied Numerical Math.*, 17:275–291, 1995.

[23] F. Diele, L. Lopez, and R. Peluso. The Cayley transform in the numerical solution of unitary differential systems. *Advances in computational mathematics*, 8:317–334, 1998.

[24] S. P. Diliberto. On systems of ordinary differential equations. In *Contributions to the Theory of Nonlinear Oscillations*, volume Ann. of Math. Studies 20, pages 1–38, Princeton, 1950. Princeton Univ. Press.

[25] J. P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange attractors. *Reviews of Modern Physics*, 57:617–656, 1985.

[26] K. Geist, U. Parlitz, and W. Lauterborn. Comparison of different methods for computing Lyapunov exponents. *Prog. Theor. Phys.*, 83:875–893, 1990.

[27] 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*, 27:311–337, 1987.

[28] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 2nd edition, 1989.

[29] J.M. Greene and J-S. Kim. The calculation of Lyapunov spectra. *Physica D*, 24:213–225, 1987.

[30] D. Gupalo, A. S. Kaganovich, and E. G. D. Cohen. Symmetry of Lyapunov spectrum. *J. Statistical Physics*, 74:1145–1159, 1994.

[31] E. Hairer, S. P. Nœrsett, and G. Wanner. *Solving Ordinary Differential Equations I*. Springer-Verlag, Berlin-Heidelberg, 1993. Second edition.

[32] D. Higham. Time-stepping and preserving orthonormality. *BIT*, 37:24–36, 1997.

[33] R. A. Johnson, K. J. Palmer, and G. Sell. Ergodic properties of linear dynamical systems. *SIAM J. Mathem. Analysis*, 18:1–33, 1987.

[34] A. Lyapunov. Problém géneral de la stabilité du mouvement. *Int. J. Control*, 53:531–773, 1992.

[35] V. M. Millionshchikov. Structurally stable properties of linear systems of differential equations. *Differents. Uravneniya*, 5:1775–1784, 1969.

[36] V. M. Millionshchikov. Systems with integral division are everywhere dense in the set of all linear systems of differential equations. *Differents. Uravneniya*, 5:1167–1170, 1969.

[37] V. M. Millionshchikov. The stochastic stability of Lyapunov's characteristic exponent. *Differents. Uravneniya*, 11:581–583, 1975.

[38] V. I. Oseledec. A multiplicative ergodic theorem. Lyapunov characteristic numbers for dynamical systems. *Trans. Moscow Mathem. Society*, 19:197, 1968.

[39] O. Perron, Die Ordnungszahlen Linearer Differentialgleichungssystemen, *Math. Zeits*, 31:748–766, 1930.

[40] G. Rangarajan, S. Habib, and R. D. Ryne. Lyapunov exponents without rescaling and reorthogonalization. *Physical Review Letters*, 80:1747–1750, 1998.

[41] R. J. Sacker and G. R. Sell. A spectral theory for linear differential systems. *J. Diff. Equations*, 27:320–358, 1978.