Improved Numerical Floquet Multipliers

Kurt Lust[1]
Department of Computer Science, K.U.Leuven, Heverlee, Belgium

Abstract

This paper studies numerical methods for linear stability analysis of periodic solutions in codes for bifurcation analysis of small systems of ordinary differential equations (ODEs). Popular techniques in use today (including the AUTO97 method) produce very inaccurate Floquet multipliers if the system has very large or small multipliers. These codes compute the monodromy matrix explicitly or as a matrix pencil of two matrices. The monodromy matrix arises naturally as a product of many matrices in many numerical methods, but this is not exploited. In this case, all Floquet multipliers can be computed with very high precision by using the periodic Schur decomposition and corresponding algorithm [Bojanczyk *et al.*, 1992]. The time discretisation of the periodic orbit becomes the limiting factor for the accuracy. We present just enough of the numerical methods to show how the Floquet multipliers are currently computed and how the periodic Schur decomposition can be fitted into existing codes but omit all details. However, we show extensive test results for a few artificial matrices and for two four-dimensional systems with some very large and very small Floquet multipliers to illustrate the problems experienced by current techniques and the better results obtained using the periodic Schur decomposition. We use a modified version of AUTO97 [Doedel *et al.*, 1997] in our experiments.

1 Introduction

In this paper, we study algorithms for the computation of Floquet multipliers in codes for computation of periodic orbits of systems of ordinary differential equations (ODEs) based on multiple shooting (e.g., BIFPACK [Seydel, 1999]) or Gauss-Legendre collocation (AUTO97 [Doedel *et al.*, 1997] and CONTENT [Kuznetsov, 1998].) The ideas in this paper also apply to some other numerical methods. We will show that the techniques used in popular codes sometimes fail and we will present a better way to compute Floquet multipliers with very high accuracy. Our method works even in the presence of very large or very small Floquet multipliers, and does not require the solution of additional boundary-value problems.

Consider the autonomous system of ODEs

$$\frac{dx}{dt} = f(x, \lambda). \tag{1}$$

In this system, $x \in \mathbb{R}^N$ is the state and $\lambda \in \mathbb{R}^p$ are the parameters of the model. The numerical methods in this paper are all for N relatively small (not more than, say, 50), but the ideas can be easily extended to methods for large systems such as the multiple shooting Newton–Picard method of Lust [1997]. The details of continuation methods are not important for this paper. Hence we will pretend that we compute solutions at predetermined values of the parameters and drop λ from our notation. A periodic solution of (1) is a closed trajectory, i.e.,

$$\exists T \in \mathbb{R}_0^+ \ni x(T) = x(0).$$

[1]E-mail: Kurt.Lust@cs.kuleuven.ac.be. Kurt Lust is a postdoctoral fellow of the Fund for Scientific Research – Flanders.

Accepted for publication in International Journal of Bifurcation and Chaos.

The smallest value of T ($T > 0$) for which this holds is called the **period**. Let $\phi_t(x(0))$ denote the solution of (1) at time t starting from $x(0)$ at time $t = 0$. The linear stability of the periodic orbit is determined by the eigenvalues of the **monodromy matrix**

$$M = \left.\frac{\partial \phi_T(x)}{\partial x}\right|_{x = x(0)} . \tag{2}$$

Its eigenvalues are also called **Floquet multipliers**. The monodromy matrix is the solution at time T of the variational equation

$$\frac{dM(t)}{dt} = \left.\frac{\partial f(x)}{\partial x}\right|_{x(t)} M(t),\ M(0) = I. \tag{3}$$

The Floquet multipliers are independent of the choice of the point $x(0)$ on the limit cycle although the monodromy matrix and its eigenvectors do depend on this choice. For the autonomous ODE system (1), one of the Floquet multipliers is always 1. Its eigenvector is the tangent vector to the limit cycle in the point $x(0)$, i.e., $f(x(0))$. This Floquet multiplier is also called the trivial Floquet multiplier. A periodic solution is asymptotically stable if all Floquet multipliers except the trivial Floquet multiplier are strictly smaller than one in modulus. Bifurcations occur when one or more Floquet multipliers cross the unit circle in the complex plane. In numerical studies of dynamical systems, one is mostly interested in detecting bifurcations and in determining the number of Floquet multipliers on or outside the unit circle. However, more precise information about the exact values of the Floquet multipliers is often desired.

The Floquet multipliers μ can also be computed from the boundary value problems

$$\begin{cases} \dfrac{dv(t)}{dt} = \left.\dfrac{\partial f(x)}{\partial x}\right|_{x(t)} v(t), \\ v(T) = \mu v(0) \end{cases} \tag{4}$$

or

$$\begin{cases} \dfrac{dw(t)}{dt} = \left(\left.\dfrac{\partial f(x)}{\partial x}\right|_{x(t)} - \sigma I \right) w(t), \\ w(T) = w(0) \end{cases} \tag{5}$$

with

$$\mu = \exp(\sigma T).$$

σ is also called the Floquet exponent. The methods presented in this paper can be extended easily to solve these boundary value problems. Eq. (5) in particular is interesting since it avoids excessive growth or shrinking of $w(t)$ along the orbit. However, we want to show in this paper that there is no need to solve these boundary value problems to compute the Floquet multipliers with high accuracy. In fact, we expect only a minor to no improvement at all from such methods compared to the method we will present in this paper.

The monodromy matrix appears naturally in many numerical methods for the computation of periodic solutions. One of the simplest methods is single shooting (used, e.g., in the package Locbif [Khibnik *et al.*, 1993].) Since a periodic orbit is just a special trajectory of an ODE system, it is determined entirely by one point $x(0)$ of the orbit. This point satisfies the equation

$$r(x(0), T) = \phi_T(x(0)) - x(0) = 0. \tag{6}$$

Assuming that we look for a periodic solution at a given parameter value, (6) has $N+1$ unknowns: the N-dimensional vector $x(0)$ and the period T. (6) has discrete solutions for T, but any point $x(0)$ of the periodic orbit satisfies this equation. An additional scalar constraint

$$s(x(0), T) = 0 \tag{7}$$

has to be added to remove this indeterminacy. This constraint is also known as the **phase condition**. Dependence on T is not required for this equation. Examples of phase conditions can be found in [Seydel, 1994]. The nonlinear system (6)-(7) can be solved using Newton's method. In this iterative method, Eqs. (6)-(7) are first linearised around the current approximation $(x(0), T)$, next the linearised system

$$\begin{aligned} &\left.\frac{\partial(r,s)(x,T)}{\partial(x,T)}\right|_{(x(0),T)} \begin{bmatrix} \Delta x(0) \\ \Delta T \end{bmatrix} \\ &= \begin{bmatrix} M - I & f(\phi_T(x(0))) \\ \frac{\partial s(x(0),T)}{\partial x(0)} & \frac{\partial s(x(0),T)}{\partial T} \end{bmatrix} \begin{bmatrix} \Delta x(0) \\ \Delta T \end{bmatrix} = - \begin{bmatrix} r(x(0),T) \\ s(x(0),T) \end{bmatrix} \end{aligned} \tag{8}$$

is solved and finally the current approximation $(x(0), T)$ is updated with $(\Delta x(0), \Delta T)$. When $(x(0), T) = (x(0)^*, T^*)$, a solution of Eqs. (6)-(7) with x^* a point on the limit cycle, M in (8) is again the monodromy matrix (2). Note that we cannot use the exact $\phi_T(x(0))$ in a numerical method. Instead we must rely on a numerical time integrator producing only an approximation $\tilde{\phi}_T(x(0))$. M in (8) should be the Jacobian of our numerical scheme $\tilde{\phi}_T(x(0))$ and $f(\phi_T(x(0)))$ in (8) should be replaced with the partial derivative of $\tilde{\phi}_T(x(0))$ with respect to T. However, in practice one can easily use the approximation $f(\tilde{\phi}_T(x(0)))$ and compute M by solving the variational equation (3) using the same numerical scheme as for (1). In fact, for most popular time integration schemes, the results are identical except for the effect of rounding errors. Integration of the variational equations can be relatively cheap if the structure of the system is fully exploited. As the Newton iteration converges, increasingly accurate approximations for the monodromy matrix are generated and approximations to the Floquet multipliers are easily computed from the last approximation for the monodromy matrix using a standard eigenvalue solver, e.g., from Lapack [Anderson *et al.*, 1999].

Several factors influence the accuracy of the Floquet multipliers. One factor is the time discretisation error introduced by our numerical scheme. However, this is not the only source of errors. Further numerical errors are made during the computation of the trajectory and the monodromy matrix. The simple fact that the monodromy matrix is represented in floating point numbers already implies an error $\|\Delta M\|$ on the order of $\varepsilon_{mach} \|M\|$ with $\|\cdot\|$ an arbitrary matrix norm and ε_{mach} the machine precision. For the double precision IEEE arithmetic provided by most workstations and personal computers, ε_{mach} is around 10^{-16}. Although symmetries in the ODE system (1) will translate into special properties or structure of the monodromy matrix, for general systems, the monodromy matrix has usually no special structure. It is usually a full, nonnormal matrix, even if $\partial f / \partial x$ is a sparse matrix. Let us assume for simplicity that M is diagonalisable. According to the Bauer–Fike theorem (see, e.g., [Golub & Van Loan, 1996] p. 321) the eigenvalues $\tilde{\mu}$ of the perturbed matrix $M + \Delta M$ satisfy

$$\min_{\mu \in \lambda(M)} |\mu - \tilde{\mu}| \leq \kappa_p(S) \|\Delta M\|_p = \|S\|_p \left\|S^{-1}\right\|_p \|\Delta M\|_p \tag{9}$$

where $\lambda(M)$ denotes the spectrum of M and where S is the matrix of eigenvectors of M, i.e., $S^{-1}MS$ is a diagonal matrix with the eigenvalues of M on the diagonal. For a normal matrix, $\kappa_2(S) = 1$ and a perturbation of size $\varepsilon_{mach} \|M\|_2$ may cause absolute perturbations of the eigenvalues of the same order. This bound is sharp: it is sufficient to perturb M with a multiple of the rank-one matrix vv^T where v is an eigenvector of M. For a nonnormal matrix, $\kappa_2(S) > 1$ and the results are even more pessimistic. Hence we cannot expect to be able to compute eigenvalues smaller than $\varepsilon_{mach} \|M\|_2$. As a rule of thumb, we have found that it is impossible to compute any eigenvalue smaller than roughly $\varepsilon_{mach} \max|\lambda(M)|$ with any accurate digits, and the number of digits lost for an eigenvalue μ is on the order of

$$d = \log_{10} \frac{\max|\lambda(M)|}{|\mu|}. \tag{10}$$

This result is not a strong limitation for single shooting methods. Indeed, single shooting methods are not capable of computing very unstable periodic orbits, so eigenvalues close to the unit circle will always be quite accurate assuming that a good time integration scheme is used and that time steps are done with enough accuracy. Moreover, we cannot realistically expect to be able to compute very small eigenvalues (say, on the order of machine precision) since a unit perturbation in the direction of a corresponding eigenvector would be very small after integrating the variational equations over one period and be of the same order of the round-off errors. More advanced methods such as multiple shooting or Gauss–Legendre collocation are intrinsically capable of returning any Floquet multiplier with very high precision (almost only limited by the time discretisation errors), but inadequate linear algebra techniques in current codes impose single shooting-like limitations on the accuracy of the Floquet multipliers. We shall discuss this further in the next section.

The plan of this paper is as follows. In Sec. 2, we will discuss multiple shooting methods and the Gauss–Legendre collocation method as implemented in AUTO97. We will show that in these methods, the monodromy matrix appears as a product of matrices and/or inverses of matrices and also indicate how Floquet multipliers are computed in current codes. In Sec. 3 we first demonstrate using a few synthetic examples why the current strategies can produce wrong results. Next we discuss the periodic Schur decomposition. This is a decomposition of a product of matrices (with or without some inverses of matrices) and was first introduced by Bojanczyk, Golub & Van Dooren [1992]. We demonstrate the improved results using a modified version of AUTO97 in Sec. 4 and conclude in Sec. 5.

The purpose of this paper is to point out a deficiency in current state-of-the-art codes for bifurcation analysis of periodic solutions. As such, we will limit the discussion of numerical methods to the minimum required to understand the source of the problem and why the proposed solution should be more reliable. We will illustrate our point with ample synthetic examples and tests with AUTO97. The reader is referred to [Bojanczyk *et al.*, 1992] and [Lust, 2000] for more details on the periodic QR algorithm used to compute the periodic Schur decomposition.

2 Current Numerical Techniques

The single shooting method presented in Sec. 1 cannot always compute very unstable solutions or solutions that follows an unstable structure in phase space for some time (such as the unstable part of the slow manifold in a multiple time scale system with canard solutions, see, e.g., [Guckenheimer & Meloon, 2000].) Numerical time integration over long time intervals is hard or impossible in such cases, since round-off and discretisation errors build up along at least part of the trajectory. Even if time integration is successful, the domain of attraction of Newton's method for Eqs. (6)-(7) is often very small. More sophisticated methods are needed for robust, general-purpose codes. Such methods belong to one of two classes:

- (Multiple) shooting methods generate parts of trajectories using a numerical time integrator and determine the initial conditions and time intervals for the trajectory parts such that all parts join into a closed curve.

- Global methods look for the function in a linear space of periodic functions that minimizes some measure for the residual
$$\frac{dx}{dt} - f(x).$$
Methods differ in the space of functions used, the measure for the residual and in the choice of additional constraints such as the phase condition.

We will now discuss two popular methods in more detail. We will present one variant of multiple shooting and explain Gauss–Legendre collocation as an example of a global method.

2.1 Multiple shooting

Multiple shooting for boundary value problems was first proposed by Keller [1968]. In the context of periodic solutions, the method computes several points on the orbit. Time integration is only needed from one point to the next point on the orbit, reducing numerical problems with the time integration and increasing the domain of attraction of Newton's method. There are several variants of the method, see, e.g., [Guckenheimer & Meloon, 2000]. We will present one typical variant (similar to the one implemented in BIFPACK) and show how the monodromy matrix can be recovered from the computations. In other variants, the monodromy matrix can be recovered in a similar way.

Consider a partition of the unit interval
$$s_i,\ i = 0, \ldots, m \tag{11}$$
with
$$0 = s_0 < s_1 < \cdots < s_{m-1} < s_m = 1.$$
The limit cycle is represented by the points
$$x_i = x(s_i T),\ i = 0, \ldots, m \tag{12}$$
where $x_m = x_0$ for a periodic solution. Let
$$\nabla s_i = s_i - s_{i-1}.$$
One variant of multiple shooting computes the points (12) and the period T from the nonlinear system
$$\left\{\begin{array}{rcl} \phi_{\nabla s_1 T}(x_0) - x_1 & = & 0, \\ & \vdots & \\ \phi_{\nabla s_{m-1} T}(x_{m-2}) - x_{m-1} & = & 0, \\ \phi_{\nabla s_m T}(x_{m-1}) - x_m & = & 0, \\ x_m - x_0 & = & 0 \text{ (periodicity constraint,)} \\ s(x_0, \ldots x_{m-1}, T) & = & 0. \end{array}\right. \tag{13}$$
ϕ represents our numerical time integrator. We have omitted the tilde from the notation used in Sec. 1. The last equation in (13) is again the phase condition. It depends on one or more of the points x_i. The variable x_m can be easily eliminated using the periodicity constraint. Newton's

method requires the repetitive solution of linear systems

$$\begin{bmatrix} M_1 & -I & & & & b_1 \\ & \ddots & \ddots & & & \vdots \\ & & M_{m-1} & -I & & b_{m-1} \\ & & & M_m & -I & b_m \\ -I & & & & I & \\ s_{x_0} & \cdots & s_{x_{m-2}} & s_{x_{m-1}} & & s_T \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \vdots \\ \Delta x_{m-2} \\ \Delta x_{m-1} \\ \Delta x_m \\ \Delta T \end{bmatrix} = - \begin{bmatrix} \phi_{\nabla s_1 T}(x_0) - x_1 \\ \vdots \\ \phi_{\nabla s_{m-1} T}(x_{m-2}) - x_{m-1} \\ \phi_{\nabla s_m T}(x_{m-1}) - x_0 \\ x_m - x_0 \\ s(x_0, \ldots x_{m-1}, T) \end{bmatrix}. \tag{14}$$

Here

$$M_i = \frac{\partial \phi_{\nabla s_i T}(x_{i-1})}{\partial x_{i-1}} \text{ and } b_i = \frac{\partial \phi_{\nabla s_i T}(x_{i-1})}{\partial T} \approx \nabla_{s_1} f(\phi_{\nabla s_i T}(x_{i-1})).$$

Instead of computing the derivatives of the numerical time integrator for M_i, it is more common to integrate the variational equation (3) together with (1) as we showed for single shooting. Several methods have been proposed to solve (14) efficiently, see, e.g., [Deuflhard & Bader, 1993] and [Lust, 1997]. Using forward recursion, (14) can be reduced to a system in the unknowns Δx_0 and T having the same structure as (8), but this method can be unstable.

Since at convergence

$$x_m = \phi_{\nabla s_m T}(\cdots(\phi_{\nabla s_2 T}(\phi_{\nabla s_1 T}(x_0)))\cdots) = \phi_T(x(0)),$$

the monodromy matrix (or at least, the numerical approximation to it) is

$$M = \frac{\partial x_m}{\partial x_0} = M_m \cdots M_1, \tag{15}$$

so it is a product of submatrices of (14) at convergence of the Newton process. Codes based on forward recursion will compute (15) during the solution procedure. Note that the monodromy matrix in the point x_i is

$$M(x_i) = M_i \cdots M_1 M_m \cdots M_{i+1}. \tag{16}$$

Hence it is also easy to compute the monodromy matrix in the other points of the periodic orbit from (14).

Multiple shooting can also be used to compute the Floquet multipliers from the linear boundary value problem (4) using the trajectory computed from blocks in Eq. (13). Setting

$$v_i = v(s_i T), \ i = 0, \ldots, m,$$

one obtains the linear system

$$\begin{bmatrix} M_1 & -I & & \\ & \ddots & \ddots & \\ & & M_m & -I \\ -\mu I & & & I \end{bmatrix} \begin{bmatrix} v_0 \\ \vdots \\ v_{m-1} \\ v_m \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \tag{17}$$

i.e., a system very similar to (14). The periodicity constraint has been replaced with the new boundary conditions, and the row and column of bordering corresponding to the phase condition and the period have been dropped. (17) is a special eigenvalue problem: we are interested in values of μ for which the system has nonzero solutions for the vectors v_i. From the first m

blocks of equations one easily gets $v_m = M_m \cdots M_1 v_0$ and after substituting this expression in the boundary conditions, we obtain the eigenvalue problem

$$M_m \cdots M_1 v_0 = \mu v_0.$$

This shows once more that the monodromy matrix in x_0 is $M_m \cdots M_1$.

The Floquet multipliers are usually computed by first computing the product (15) if it is not already computed in the linear system solver for Eq. (14) and then applying a standard eigenvalue solver just as in the single shooting case. There are two problems with this method. First, all information is condensed in one matrix and our analysis based on the Bauer–Fike theorem in Sec. 1 applies again. We cannot expect to compute any Floquet multiplier smaller than roughly $\varepsilon_{mach} \max|\lambda(M)|$ with any accurate digits. Second, we will obtain eigenvector information for only one point of the periodic orbit, while it is natural to require such information in all points (12) (e.g., for branch switching.) *We essentially get the same type of information as in a single shooting method: limited accuracy for the Floquet multipliers and eigenvector information in only one point of the periodic orbit, a clear mismatch with the representation of a periodic orbit by multiple points in the multiple shooting method.*

2.2 Gauss–Legendre collocation

Gauss–Legendre collocation is probably the most popular method at the moment for computation and bifurcation analysis of (1) thanks to its implementation in AUTO97 and CONTENT. It uses a space of spline functions and requires that the ODE be satisfied exactly in a number of well-chosen points on the periodic orbit.

In AUTO, the ODE (1) is first transformed using the transformation $t = Ts$ to

$$\frac{dx(s)}{ds} = T f(x(s), \lambda) \tag{18}$$

and the boundary condition for a periodic solution becomes

$$x(s=1) = x(s=0).$$

First a mesh (11) is defined on the interval $[0,1]$. The space of functions used by AUTO is the space of generalised spline functions of degree n, continuous on the interval $[0,1]$ but there are no continuity requirements for derivatives. I.e., on each interval the spline is a (different) polynomial of degree n and the piecewise polynomial is continuous over the whole interval $[0,1]$. This is not a space of periodic functions. Instead, periodicity is expressed as a separate set of constraints. The Gauss–Legendre collocation strategy as implemented in AUTO uses a basis of Lagrange polynomials. To define these polynomials, an equidistant submesh

$$s_{i+j/n} = s_i + \frac{j}{n}(s_{i+1} - s_i),\ j = 0, \ldots, n \tag{19}$$

is defined on each mesh interval $[s_i, s_{i+1}]$. The Lagrange polynomials on $[s_i, s_{i+1}]$ are polynomials of degree n given by

$$l_{i,j}(s) = \prod_{\substack{l=0\\ l\neq j}}^{l=n} \frac{s - s_{i+l/n}}{s_{i+j/n} - s_{i+l/n}}\ ,\ j = 0, \ldots, n, \tag{20}$$

one for each point of the submesh. Remark that

$$l_{i,j}(s_{i+l/n}) = \delta_{l,j}$$

with $\delta_{l,j}$ the Kronecker delta. Therefore the coefficients of an arbitrary polynomial of degree n expressed in the basis (20) are precisely the values of the polynomial in the points $s_{i+j/n}$ of the submesh (19). Let $p(s)$ denote a function in our space of generalised spline functions and $p_i(s)$ be the spline $p(s)$ restricted to the mesh interval $[s_i, s_{i+1}]$. Then $p_i(s)$ can be expanded to

$$p_i(s) = \sum_{j=0}^{n} x_{i+j/n} l_{i,j}(s). \tag{21}$$

Our function space has dimension $N(mn+1)$. The periodicity constraint $x_0 = x_m$ removes N degrees of freedom. To determine the other coefficients, Gauss–Legendre collocation imposes that the spline function $p(s)$ should satisfy (18) in n points on each mesh interval. These n points are the zeros of the Legendre polynomial of degree n shifted and scaled to the interval $[s_i, s_{i+1}]$ and are also called the Gauss points. They will be denoted by $z_{i,k}$. The subscript i denotes the mesh interval and k is the index of the Gauss point on mesh interval i. Using Eq. (21), the collocation requirements are

$$\begin{gathered}\sum_{j=0}^{n} \left.\frac{dl_{i,j}}{ds}\right|_{s\,=\,z_{i,k}} x_{i+j/n} = Tf\left(\textstyle\sum_{j=0}^{n} l_{i,j}(z_{i,k})\, x_{i+j/n}\right), \\ i = 0, \ldots, m-1,\ k = 1, \ldots, n.\end{gathered} \tag{22}$$

(We again dropped the parameter from our notations.) This type of collocation is also used in COLSYS [Ascher *et al.*, 1981], a package for multipoint boundary value problems. Ascher *et al.* [1995] and Weiss [1974] show that this method is equivalent to a particular n-step implicit Runge–Kutta formula which is therefore also called a Gauss–Legendre Runge–Kutta formula. Gauss–Legendre collocation is essentially equivalent to the multiple shooting using the same mesh (11) and a single step with the corresponding Gauss–Legendre Runge–Kutta time integrator on each mesh interval.

The system (22) is further augmented with the periodic boundary condition

$$x_m = x_0 \tag{23}$$

and a phase condition. The precise phase condition is not important for this paper and we refer the reader to [Doedel, 1986] for details. The phase condition does involve all points $x_{i+j/n}$. AUTO also uses pseudo-arclength continuation, but we will ignore the continuation aspects for the sake of clarity.

Equations (22)–(23) and the phase condition together form a nonlinear system of $N(nm+1)+1$ equations and an equal number of unknowns. This system is again solved using a Newton process. For the purpose of this paper it is important to understand the structure of the various matrices that arise during this process. We shall show the structure of the matrices for the case $n = 2$ and $m = 3$. The structure of the linearised system in this case is

$$\left[\begin{array}{cccccccc} * & * & * & & & & & \mid \\ * & * & * & & & & & \mid \\ & & * & * & * & & & \mid \\ & & * & * & * & & & \mid \\ & & & & * & * & * & \mid \\ & & & & * & * & * & \mid \\ -I & & & & & & I & \\ - & - & - & - & - & - & - & \end{array}\right] \left[\begin{array}{c} \Delta x_0 \\ \Delta x_{1/2} \\ \Delta x_1 \\ \Delta x_{1+1/2} \\ \Delta x_2 \\ \Delta x_{2+1/2} \\ \Delta x_3 \\ \Delta T \end{array}\right] = \left[\begin{array}{c} \mid \\ \mid \\ \mid \\ \mid \\ \mid \\ \mid \\ \mid \\ \cdot \end{array}\right]. \tag{24}$$

The last column in this matrix comes from the derivatives with respect to T; the last row contains the partial derivatives of the phase condition. Each $*$ in this matrix represents a $N \times N$-matrix

$$A_{i,j,k} = Tl_{i,j}(z_{i,k}) \left.\frac{\partial f}{\partial x}\right|_{p_i(z_{i,k})} - \left.\frac{dl_{i,j}}{ds}\right|_{s\,=\,z_{i,k}} I_N \tag{25}$$

for some i, j and k. In (24), the three blocks of 2×3 stars correspond to the mesh intervals $[s_i, s_{i+1}]$, $i = 0$, $i = 1$ and $i = 2$ respectively. Within each 2×3 block, row k corresponds to the collocation requirements in the Gauss point $z_{i,k}$ and column j corresponds with the unknown $x_{i+(j-1)/n}$, i.e., $A_{i,j,k}$ is the $(k, j+1)$ subblock of the ith 2×3 block. Note that intrinsically, Eq. (24) again contains a mapping from a perturbation of the point x_0 to the corresponding perturbation of the point x_m one period later.

AUTO has a specialized linear system solver that exploits the block structure of Eq. (24) and solves Eq. (24) in such a way that the Floquet multipliers can also be computed. In the first phase, the internal unknowns $x_{i+j/n}$, $j = 1, \ldots, n-1$ of each mesh interval are eliminated. This operation can be done in parallel for each mesh interval and is known as the "condensation of the parameters". Equation (24) is reduced using Gaussian elimination with row pivoting within each 2×3-block to

$$\left[\begin{array}{cccccccc}
* & \nabla & * & & & & & | \\
\boxed{G_1} & & \boxed{-H_1} & & & & & \boxed{|} \\
 & & * & \nabla & * & & & | \\
 & & \boxed{G_2} & & \boxed{-H_2} & & & \boxed{|} \\
 & & & & * & \nabla & * & | \\
 & & & & \boxed{G_3} & & \boxed{-H_3} & \boxed{|} \\
\boxed{-I} & & & & & & \boxed{I} & \\
\boxed{-} & & \boxed{-} & & \boxed{-} & & \boxed{-} &
\end{array}\right]
\left[\begin{array}{c}
\boxed{\Delta x_0} \\
\Delta x_{1/2} \\
\boxed{\Delta x_1} \\
\Delta x_{1+1/2} \\
\boxed{\Delta x_2} \\
\Delta x_{2+1/2} \\
\boxed{\Delta x_3} \\
\boxed{\Delta T}
\end{array}\right]
=
\left[\begin{array}{c}
\boxed{|} \\
| \\
\boxed{|} \\
| \\
\boxed{|} \\
| \\
\boxed{|} \\
\boxed{\cdot}
\end{array}\right]. \tag{26}$$

∇ denotes an upper triangular matrix. This system can be solved by first computing T and the unknowns Δx_i at the mesh points (11) from the boxed subsystem and then computing the internal unknowns of each mesh interval from the remaining equations. Note that if we would use Gauss–Legendre collocation to solve the eigenvalue problem (4) for the Floquet multipliers, we would obtain essentially the same matrix but without the bordering from the phase condition and derivative with respect to the period and with a block $-\mu I$ at the lower left instead of the block $-I$. This is an eigenvalue problem very similar to Eq. (17). It can be reduced to the standard eigenvalue problem

$$H_m^{-1} G_m \ldots H_2^{-1} G_2 H_1^{-1} G_1 v_0 = \mu v_0.$$

Hence the monodromy matrix in x_0 can be recovered from (26) at convergence and is

$$M = H_m^{-1} G_m \ldots H_2^{-1} G_2 H_1^{-1} G_1. \tag{27}$$

Similarly, the monodromy matrix in the point x_i is given by the product

$$M(x_i) = H_i^{-1} G_i \ldots H_1^{-1} G_1 H_m^{-1} G_m \ldots H_{i+1}^{-1} G_{i+1}. \tag{28}$$

AUTO proceeds by further reducing the boxed subsystem of Eq. (26) to

$$\begin{bmatrix} * & \nabla & & & | \\ * & & \nabla & & | \\ \boxed{G} & & & \boxed{-H} & \boxed{|} \\ \boxed{-I} & & & \boxed{I} & \\ \boxed{-} & & & \boxed{-} & \end{bmatrix} \begin{bmatrix} \boxed{\Delta x_0} \\ \Delta x_1 \\ \Delta x_2 \\ \boxed{\Delta x_3} \\ \boxed{\Delta T} \end{bmatrix} = \begin{bmatrix} | \\ | \\ \boxed{|} \\ \boxed{|} \\ \boxed{\cdot} \end{bmatrix}. \tag{29}$$

The blocks G_2 and G_3 have been eliminated using row operations with the first and second block rows respectively. During this operation, further nonzero elements where created in the first block column. This system can be solved by first solving the boxed subsystem for the unknowns x_0, x_m and T and then computing the unknowns at the other mesh points. Using similar ideas as before, it is easy to show that the monodromy matrix in x_0 is now

$$M = H^{-1}G. \tag{30}$$

However, the boxed subsystem does no longer contain information about the monodromy matrix in the other mesh points. In AUTO86, the Floquet multipliers were computed by first computing M — this is done by solving the linear system $HM = G$ — and then solving the standard eigenvalue problem

$$Mv = \mu v. \tag{31}$$

However, as was the case for multiple shooting, this method suffers from the same limitations as explained in Sec. 1 for single shooting. Fairgrieve and Jepson [1991] suggested to solve the generalised boundary value problem

$$Gv = \mu Hv \tag{32}$$

instead for which they use a routine from EISPACK [Smith *et al.*, 1976]. This suggestion is incorporated in AUTO94 and AUTO97. We will discuss the effects of this strategy in the next section. The range of Floquet multipliers that can be computed accurately does improve. In fact, in AUTO94/97 Floquet multipliers close to the unit circle usually turn out to be the most accurate ones, independent of the presence of very large or very small Floquet multipliers, though there is no guarantee that this will always be the case. However, very large or very small Floquet multipliers are usually inaccurate. The method can also only generate eigenvector information for one point on the orbit at a time and not for all mesh point simultaneously.

2.3 Other numerical methods

To compute Floquet multipliers, all one really needs is that the linearised system intrinsically contains a map from a perturbation at one point of the limit cycle to the perturbation one period later, i.e., using appropriate linear algebra techniques, the system could be reduced to a system similar to Eq. (8). This is probably the case for any method (global method or shooting method) based on a timestepper.

In Secs. 2.1 and 2.2, we have discussed a multiple shooting method using integration forward in time and the Gauss–Legendre collocation method. In both cases, it was easy to recover the monodromy matrix during the computations in all mesh points. The structure observed in (14) and (26) is quite typical. We distinguish two cases which we will call type-I and type-II structure:

- The type-I structure is the structure of the boxed subset of Eq. (26),

$$\begin{bmatrix} G_1 & -H_1 & & & | \\ & G_2 & -H_2 & & | \\ & & G_3 & -H_3 & | \\ -I & & & I & \\ - & - & - & - & \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ \Delta T \end{bmatrix} = \begin{bmatrix} | \\ | \\ | \\ | \\ \cdot \end{bmatrix}. \tag{33}$$

  It has a bidiagonal structure with one or more rows and columns of bordering from other constraints. In this case, the monodromy matrix at each mesh point is the product (28) of certain blocks and inverses of blocks of (33). We call the product (27) a type-I matrix product.

- The type-II structure is the structure of Eq. (14). This is essentially a special case of the type-I matrix with $H_i = I$. The monodromy matrix at each mesh point is the product (16) of blocks of (14). If all matrices H_i are nonsingular, a type-I matrix can also be reduced to a type-II matrix by premultiplying the ith blockrow of (33) with H_i^{-1}. We call the product (15) a type-II matrix product.

In many other numerical methods for periodic solutions, it is possible to reduce and transform the linearised system to a type-I or type-II system and the numerical methods for Floquet multipliers presented in this paper will apply to those cases too. The linearised system in the multiple shooting method of Guckenheimer & Meloon [2000] involving integration forwards and backwards in time can be transformed into a type-I system. This is also the case for the collocation method of Choe & Guckenheimer [1999] and Guckenheimer and Meloon [2000]. Finite difference methods based on one-step methods will also lead to a type-I or type-II system. E.g., the simple method obtained by using the mesh (11) and the trapezoidal rule

$$\frac{x_{i+1} - x_i}{s_{i+1} - s_i} = \frac{1}{2}\left(f(x_i) + f(x_{i+1})\right)$$

results in a type-I system after linearisation. In all these cases we can do better than simply computing the matrix products (27) or (15) and computing its eigenvalues.

3 The Periodic Schur Decomposition

3.1 Problems with traditional methods

From the previous discussion, two problems of the classical approaches clearly arise:

- In the classical methods, we can only get eigenvector information at one point of the periodic orbit at a time. If eigenvector information is required at m different points, the method needs to be applied m times. This is not a problem with single shooting methods, where the limit cycle is represented by only one point. However, in multiple shooting or collocation methods, eigenvector information in all mesh points is very useful, e.g., to implement branch switching.

- The accuracy that can be obtained from a procedure condensing all information in one or two matrices is limited. There is usually no problem if

$$\varepsilon_{mach} \left|\mu_{\max}\right| \ll \left|\mu_{\min}\right|.$$

```
randn('state', 100);
D = diag( [1e5 1 1e-5] );
D1 = D; D2 = inv(D);
Y = randn(3); Z = randn(3);
cond(Y), cond(Z)
M = Y*(D*D)*inv(Y);
G = Z*D1*inv(Y); H = Z*D2*inv(Y);
Ex = diag(D).^2;
Em = flipud(sort(eig(M)));
Egh = flipud(sort(eig(G,H)));
fprintf( 1, ['%7.1e %24.16e %7.1e %24.16e %7.1e' 13], ...
[Ex Em abs(Em-Ex)./Ex Egh abs(Egh-Ex)./Ex].'  );
```

Figure 1: MATLAB code to illustrate the differences between the AUTO86 and AUTO94 approaches to the computation of Floquet multipliers.

intended value	**AUTO86 (Eq. (31))**		**AUTO94 (Eq. (32))**	
	eigenvalue	**rel. err.**	**eigenvalue**	**rel. err.**
1.0e+10	9.9999999999999943e+09	5.7e-16	1.0000005002215284e+10	5.0e-07
1.0e+00	9.9999970240833602e-01	3.0e-07	9.9999999992005295e-01	8.0e-11
1.0e-10	1.4838430602293062e-07	1.5e+03	9.9999654891207026e-11	3.5e-06

Table 1: Comparison between the AUTO86 and AUTO94 approaches to the computation of Floquet multipliers.

> However, for periodic solutions with a very large range of Floquet multipliers, some or all of the Floquet multipliers can be very inaccurate. From the Bauer–Fike theorem we learned that the linear algebra will not constrain the accuracy of the largest Floquet multipliers computed from the eigenvalue problem (31), but eigenvalues that are very small compared to the larger ones will be inaccurate. Analysis for the generalised eigenvalue problem (32) is harder, but as is observed by Fairgrieve & Jepson [1991] and as we shall see in Sec. 4, the Floquet multipliers closest to the unit circle are usually the most accurate, although there is no guarantee that this will always be the case, and very large and very small Floquet multipliers are usually inaccurate.

Let us illustrate the last point with a few synthetic test cases using random matrices and MATLAB. We used MATLAB 5.3 on a Pentium PC running Windows NT and used the new Lapack-based numerics library [Moler, 2000].

Our first test illustrates the differences one can expect between the AUTO86 strategy (using (31)) and the AUTO94 strategy (solving (32).) From two diagonal matrices, we constructed the standard eigenvalue problem (31) and the generalised eigenvalue problem (32) with known eigenvalues. The code is shown in Fig. 1. Note that due to round-off errors, the eigenvalues are not exact and there are more stable methods to build a matrix with given eigenvalues, but the errors caused by round-off errors during the construction of the matrices are small compared to the error of some of the eigenvalues for the results presented in Table 1. Computing the Floquet multipliers from Eq. (31), the largest eigenvalue is very accurate. The error is only a few times the machine precision. However, we have lost 9-10 digits for the eigenvalue at 1 and the smallest eigenvalue is totally inaccurate. These results are in line with our heuristic

diag(Q^TGZ)	diag(Q^THZ)	eigenvalue
`9.4372672222576934e+04`	`9.4372643316842431e-06`	`1.0000003062935772e+10`
`3.3782305753441708e-06`	`3.3782435005041771e+04`	`9.9999617399988945e-11`
`1.0895476436804528e+00`	`1.0895476437672922e+00`	`9.9999999992029776e-01`

Table 2: Diagonal elements of the upper triangular matrices Q^TGZ and Q^THZ produced by the MATLAB QZ function and the quotient of those numbers. Note that the eigenvalues are slightly different from those produced by the MATLAB `eig()` function used before, but the relative errors are comparable.

intended	computed result		QZ decomposition	
eigenvalue	eigenvalue	rel. err.	diag(Q^TGZ)	diag(Q^THZ)
`1.0e+10`	`1.0000021695301685e+10`	`2.2e-06`	`9.437267e-01`	`9.437231e-11`
`1.0e+00`	`1.0000000000000002e+00`	`2.2e-16`	`1.026180e+00`	`1.026180e+00`
`1.0e-10`	`9.9999706914013111e-11`	`2.9e-06`	`3.586842e-11`	`3.586853e-01`

Table 3: Eigenvalues computed using the QZ algorithm and the diagonal elements after the reduction.

(10). The picture is different when using the generalised eigenvalue problem (32) to compute the Floquet multipliers. Now none of the eigenvalues is very accurate, and the eigenvalue at 1 is best approximated. The real QZ algorithm used to solve (32) reduces both matrices to upper triangular structure in this case (since there are no pairs of complex conjugate eigenvalues.) The eigenvalues are computed by dividing corresponding diagonal elements. Table 2 shows the diagonal elements after running the QZ algorithm for the same matrix as used for Table 1. It is not possible to check in this case what the exact values should be. However, it is reasonable to expect the largest diagonal elements to be the most accurate ones and a loss of relative accuracy for any given element proportional to the ratio of the modulus of the largest diagonal element to the modulus of that element. Since the maximal relative error for the quotient of two numbers is the sum of the relative errors, the least accurate of the two diagonal elements from which an eigenvalue is computed will determine the accuracy of that eigenvalue. Which eigenvalues are most accurate depends on the diagonals of Q^TGZ and Q^THZ after reduction by the QZ algorithm, but if the ratio of the moduli of the largest to the smallest Floquet multiplier is very large, some Floquet multipliers will be inaccurate. It is not always the case that the eigenvalues with amplitude somewhere in the middle are the most accurate ones or that no eigenvalue is very accurate. Consider, e.g., the matrices obtained using

```
D1 = diag( [1 1 1e-10] );
D2 = diag( [1e-10 1 1] );
```

in the script of Fig. 1. The results for these matrices are shown in Table 3. In this case, the eigenvalue at 1 is computed very accurately, but ten digits have been lost computing the eigenvalues 10^{10} and 10^{-10} (more than for the result in Table 1.) A look at the diagonal elements of the upper triangular factors Q^TGZ and Q^THZ shows why. The eigenvalue 1 is computed from the largest elements of both diagonals and these elements are very accurate. However, computation of the two other eigenvalues involves a number on the order of 10^{-10} in both cases, and these diagonal elements are expected to have a large relative error. Likewise, with $G = I$ and $H = M^{-1}$, the smallest eigenvalue would be the most accurate one. It appears that in computations with AUTO94 and AUTO97, we usually find ourselves in the case of Table 3 and

```
randn('state', 10001);
D = diag( [100 10 1 0.1 0.01] );
Y1 = randn(5,5); Y2 = randn(5,5); Y3 = randn(5,5); Y4 = randn(5,5); Y5
= randn(5,5);
[cond(Y1) cond(Y2) cond(Y3) cond(Y4) cond(Y5)]
M1 = Y2*D*inv(Y1); M2 = Y3*D*inv(Y2); M3 = Y4*D*inv(Y3); M4 =
Y5*D*inv(Y4); M5 = Y1*D*inv(Y5);
M = M5*M4*M3*M2*M1;
Ex = diag(D).^5;
Em = flipud(sort(eig(M)));
fprintf( 1, ['%7.1e %24.16e %7.1e' 13], [Ex Em abs(Em-Ex)./Ex].'  );
```

Figure 2: MATLAB code to illustrate the loss of accuracy caused by explicitly constructing the product of matrices and solving a standard eigenvalue problem.

intended eigenvalue	**computed eigenvalue**	**relative error**
1.0e+10	1.0000000000000017e+10	1.7e-15
1.0e+05	1.0000000000435553e+05	4.4e-11
1.0e+00	9.9998938811090576e-01	1.1e-05
1.0e-05	1.9501410255903883e-05	9.5e-01
1.0e-10	-3.4881676070753112e-06	3.5e+04

Table 4: Eigenvalues of $M_5M_4M_3M_2M_1$ and the relative error.

the Floquet multipliers close to the unit circle are quite accurate. Fairgrieve and Jepson [1991] attribute this to the pivoting strategy used in AUTO, but there is no guarantee that the Floquet multipliers close to the unit circle will always be accurate.

Our second test illustrates the effect of explicitly computing the product (15). We constructed five matrices such that the eigenvalues of the product are known except for small round-off errors during the construction. Next we computed the product of those matrices and the eigenvalues of the product. The script used to do this is shown in Fig. 2 and the results are in Table 4. The results are completely in line with the expectations for a single shooting method. The largest Floquet multiplier is very accurate but the smallest eigenvalue is totally wrong. Instead of the expected value 10^{-10}, we get a value around $\varepsilon_{mach}10^{10}$. Given that a multiple shooting method can compute orbits with very large Floquet multipliers accurately, this is not a good result. We would be unable to detect further bifurcations on such a branch. We will use the matrix from Fig. 2 again in Sec. 4 and show that the eigenvalues can be computed much more accurately provided we avoid the explicit construction of the product.

3.2 Two matrix decompositions for matrix products

As stated before, the key to improved Floquet multipliers is avoiding the construction of the monodromy matrix (15) or (27). To this end, we will now study orthogonal matrix decompositions for type-I and type-II matrix products that allow us to easily compute eigenvalues of these products. The decompositions are a generalisation of the real QZ decomposition and real Schur decomposition respectively (see, e.g., [Golub & Van Loan, 1996] for these decompositions.) The periodic Schur decomposition (as our decomposition is called) was first proposed in [Bojanczyk *et al.*, 1992]. We will present a slight variant of the decomposition. (The changes were adopted for use in a generalisation of the Newton–Picard method for computing periodic solutions of large-scale systems to multiple shooting, see [Lust, 1997].)

First we study a decomposition of a type-I matrix product.

Lemma 3.1 Periodic Schur decomposition of a type-I matrix product. *Let G_i, $i = 1, \ldots, m$ and H_i, $i = 1, \ldots, m$ be real $N \times N$-matrices. Then there exist orthogonal $N \times N$-matrices Q_i, $i = 0, \ldots, m-1$ and Z_i, $i = 1, \ldots, m$ such that*

$$\hat{G}_i = Z_i^T G_i Q_{i-1}, \quad \hat{H}_i = Z_i^T H_i Q_i, \quad Q_m := Q_0,$$

where all matrices $\hat{G}_i$ and $\hat{H}_i$ are upper triangular except $\hat{G}_m$. $\hat{G}_m$ is quasi-upper triangular with a 1×1 block for each real eigenvalue of (27) and a 2×2-block for each pair of conjugate eigenvalues.

In fact, one could chose another matrix $\hat{G}_i$ to be quasi-upper triangular instead. In the case $m = 1$, this decomposition reduces to the QZ decomposition except for the exchange of the roles of Q and Z. The periodic Schur decomposition is not unique. Its existence is proven in [Bojanczyk *et al.*, 1992]. In fact, the decomposition is also possible if one or more of the matrices H_i is singular. One then says that the eigenvalue problem has eigenvalues at infinity. Floquet multipliers are always nonzero and finite and the matrices G_i and H_i will usually be nonsingular, though singular or near-singular matrices could occur with a bad choice of the mesh. Note that

$$\begin{aligned} Q_i^T M(x_i) Q_i &= Q_i^T \left(H_i^{-1} G_i \cdots H_1^{-1} G_1 H_m^{-1} G_m \cdots H_{i+1}^{-1} G_{i+1} \right) Q_i \\ &= Q_i^T (H_i^{-1} Z_i Z_i^T G_i Q_{i-1} Q_{i-1}^T \cdots) Q_i \\ &= \hat{H}_i^{-1} \hat{G}_i \cdots \hat{H}_1^{-1} \hat{G}_1 \hat{H}_m^{-1} \hat{G}_m \cdots \hat{H}_{i+1}^{-1} \hat{G}_{i+1} =: T_i \end{aligned}$$

is a quasi-upper triangular matrix with the same structure as $\hat{G}_m$, i.e., $M(x_i) Q_i = Q_i T_i$ is the real Schur decomposition of $M(x_i)$ and we have solved the first problem mentioned at the beginning of Sec. 3.1: obtaining information about eigenspaces at all points simultaneously. A real eigenvalue at position (j, j) in the decomposition is computed as the product

$$\hat{H}_m(j,j)^{-1} \hat{G}_m(j,j) \cdots \hat{H}_1(j,j)^{-1} \hat{G}_1(j,j).$$

Computing this product gives only a moderate growth of the relative error. Hence the accuracy of a real eigenvalue will be determined by the relative errors of the corresponding diagonal elements in each of the matrices $\hat{G}_i$ and $\hat{H}_i$. A pair of complex conjugate eigenvalues at rows and columns j and $j+1$ is computed by first computing the product of the corresponding 2×2-blocks

$$\hat{H}_m(j{:}j{+}1, j{:}j{+}1)^{-1} \hat{G}_m(j{:}j{+}1, j{:}j{+}1) \cdots \hat{H}_1(j{:}j{+}1, j{:}j{+}1)^{-1} \hat{G}_1(j{:}j{+}1, j{:}j{+}1)$$

and then computing the eigenvalues of this 2×2-matrix. The relative error of the modulus and the absolute error of the argument will usually be very small if the corresponding diagonal blocks are accurate.

A type-II product can be seen as a special case of a type-I product with $H_i = I$. The periodic Schur decomposition can be simplified in this case.

Corollary 3.2 Periodic Schur decomposition of a type-II matrix product. *Let M_i, $i = 1, \ldots, m$ $\in \mathbb{R}^{N \times N}$. Then there exist orthogonal $N \times N$-matrices Q_i, $i = 0, \ldots, m-1$, such that the matrices*

$$\hat{M}_i = Q_i^T \hat{M}_i Q_{i-1}, \; Q_m := Q_0$$

are all upper triangular except $\hat{M}_m$. $\hat{M}_m$ is a quasi-upper triangular matrix with a 1×1 diagonal block for each real eigenvalue of (15) and a 2×2-block for each pair of complex conjugate eigenvalues.

Again, we can chose another factor $\hat{G}_i$ to be quasi-upper triangular instead and

$$M(x_i)Q_i = G_i \cdots G_1 G_m \cdots G_{i+1} Q_i = Q_i \left(\hat{M}_i \cdots \hat{M}_1 \hat{M}_m \cdots \hat{M}_{i+1} \right) =: Q_i T_i$$

is the real Schur decomposition of $M(x_i)$. The eigenvalues are also easily computed.

3.3 Computational aspects

In this section, we will present some information about the algorithms used to compute the periodic Schur decomposition and about our code and we will discuss the cost of the approach.

An overview of the algorithms to compute the above decompositions is given in [Bojanczyk *et al.*, 1992]. The algorithms are an extension of the QR algorithm for the computation of the real Schur decomposition. The ideas are very similar to the extension of the QR algorithm to the QZ algorithm used to solve the generalised eigenvalue problem (32). The algorithms consist of two steps. In the first step, the matrices are reduced to a simpler form to make further computations cheaper. The resulting form is a generalisation of the upper Hessenberg form used in combination with the QR algorithm and is called the periodic upper Hessenberg form in [Bojanczyk *et al.*, 1992]. All factors are reduced to upper triangular matrices except the matrix G_m which is reduced to upper Hessenberg structure. This can be accomplished using orthogonal transformations. The second step is an iterative algorithm to compute the actual periodic Schur decomposition. It is a generalisation of the QR iterations. At each step of the algorithm, only the entries of the product that are actually needed for the QR step are computed. We have implemented the algorithm for type-II matrices. Implementation details will be discussed in a forthcoming report [Lust, 2000] and our code psSchur will be made publicly available. The code is written in Fortran 77 using a few extensions available on all modern compilers (such as variable names longer than six characters) and can also be called from MATLAB 5.3. Our code can cope with singular matrices M_i. Though the monodromy matrix and hence each of the matrices M_i are nonsingular in theory, in actual finite-precision computations, the matrices M_i can be numerically singular in extreme cases. We have observed this in the multiple shooting code of Guckenheimer and Meloon [2000] when doing extremely accurate time integrations for a multiple time scale system. There are also routines to reorder the eigenvalues in the decomposition. We do not yet provide routines to compute eigenvectors. However, since the first k columns of Q_i in the real Schur decomposition are an orthogonal basis for the space spanned by the first k generalised eigenvalues (provided $T_0(k+1,k) = 0$), we can compute some eigenvector information by reordering the Schur decomposition and this is usually all that is needed for applications. Computation of eigenvectors is troublesome if there is not a complete set of eigenvectors and generalised eigenvectors have to be used instead. This is the generic case for matrices with eigenvalues of multiplicity larger than one.

We have also integrated our implementation with AUTO97 for testing purposes and will use this modified version of AUTO for our tests in Sec. 4. Therefore the type-I structure arising in AUTO is converted to a type-II structure by computing all matrices $M_i = H_i^{-1} G_i$ using routines from Lapack.

Let us now study the cost of using the periodic Schur decomposition. We will restrict ourselves to type-II matrix products, but a similar argument can be made for type-I products. The periodic QR algorithm is certainly more expensive than traditional algorithms, even when eigenvector information is desired. Note that if v is an eigenvector of $M_m \cdots M_1$ for the eigenvalue μ, then $M_k \cdots M_1 v$ is an eigenvector of $M_k \cdots M_1 M_m \cdots M_{k+1}$ for the same eigenvalue provided all matrices M_i are nonsingular. This relationship can be used to compute eigenvector information at all grid points given eigenvector information at a single grid point. Although we do obtain information in all grid points with such a procedure, this is still done by first condensing the problem into a single shooting-style problem. The condensation into a single matrix limits the accuracy of Floquet multipliers and the eigenvectors and the forward recursion used to compute eigenvectors at one mesh point from eigenvectors at another mesh point also has some numerical pitfalls. If only the Floquet multipliers are needed, the cost of computing this information from the matrices $M_1, \ldots, M_m$ is $2mN^3$ for the computation of the product and approximately $10N^3$ for the computation of the eigenvalues using the QR algorithm (see [Golub & Van Loan, 1996].) If the matrices M_i are not evaluated again at convergence, the computation of the product has probably already been done during the Newton process to compute the orbit. If full eigenvector information in all grid points is required, the costs amount to $2mN^3$ (computation of the monodromy matrix) $+25N^3$ (QR algorithm) $+2mN^3$ (computation of all eigenvectors.) In comparison, the periodic QR algorithm is roughly m times as expensive as the QR algorithm. Convergence is often slightly slower and therefore the constants tend to be slightly larger than the 10 or 25 from the QR algorithm. However, the condensation and recursion to compute all eigenvectors are no longer needed. For large m, the periodic Schur decomposition will be considerably more expensive. However, to put this in perspective, the cost of computing the eigenvalues and eigenvectors should be compared with the cost of computing the periodic orbit and computing the matrices G_i and H_i. E.g., in AUTO, the cost of the first step of the linear system solver, the condensation of parameters, is on the order of $2/3mn^3N^3$ (i.e., the cost of solving m linear systems of size nN using Gaussian elimination) where n is the degree of the polynomials used. Fourth degree polynomials being typical in computations with AUTO, this amounts to $O(43mN^3)$, twice as much as computing the complete periodic Schur decomposition. Note also that the cost of using the periodic Schur decomposition could be reduced somewhat by first condensing all the information onto a coarser mesh before computing the eigenvector information and by transferring eigenvector information again to the original mesh at the end using forward recursion. In AUTO, this condensation process would correspond to a two-step condensation instead of the current second step of the linear system solver. This will cause some loss of accuracy, but the loss can be controlled by a good selection of the coarse mesh. The periodic Schur decomposition is certainly an attractive option if it does indeed offer a higher accuracy and robustness, which we shall now study in Sec. 4.

4 Test cases

In this section, we will present results for two four-dimensional dynamical systems obtained with our modified version of AUTO97, but let us first reconsider the second test case from Sec. 3.1. Using the same matrices as before, Table 5 shows the eigenvalues computed from the explicit computation of the product of the matrices and the eigenvalues computed by our code psSchur. The improvement is spectacular. Only two or three digits were lost computing the eigenvalue at 1 and the smallest eigenvalue is computed with ten correct digits. As we suggested before, the loss of accuracy appears to be proportional to ratios of diagonal elements of the factors $\hat{M}_i$ and not to the ratio of diagonal elements of the product of the factors. Hence the second problem mentioned in Sec. 3.1 (single shooting-like accuracy) has also been solved. We can now compute all Floquet multipliers with much higher precision than before. It is sufficient to use enough grid points and to chose the position of the grid points well to assure that the accuracy of the Floquet multipliers is determined by the accuracy of the time integration scheme and no longer limited by linear algebra operations.

Intended value	**Product $M_5M_4M_3M_2M_1$**		**psSchur**	
	Eigenvalue	**rel. err.**	**Eigenvalue**	**rel. err.**
1.0e+10	1.0000000000000017e+10	1.7e-15	1.0000000000000000e+10	0.0e+00
1.0e+05	1.0000000000435553e+05	4.4e-11	9.9999999999997133e+04	2.9e-14
1.0e+00	9.9998938811090576e-01	1.1e-05	9.9999999999975586e-01	2.4e-13
1.0e-05	1.9501410255903883e-05	9.5e-01	9.9999999999390771e-06	6.1e-12
1.0e-10	-3.4881676070753112e-06	3.5e+04	1.0000000000290037e-10	2.9e-11

Table 5: Comparison of eigenvalues computed by explicit construction of the matrix product and by the code psSchur.

Our second test example is the system of two coupled oscillators

$$\left\{\begin{array}{rcl}\frac{dx_1}{dt} &=& x_1 + \beta y_1 - x_1\left(x_1^2+y_1^2\right) + \delta\left(x_2 - x_1 + y_2 - y_1\right),\\ \frac{dy_1}{dt} &=& -\beta x_1 + y_1 - y_1\left(x_1^2+y_1^2\right) + \delta\left(x_2 - x_1 + y_2 - y_1\right),\\ \frac{dx_2}{dt} &=& x_2 + \beta y_2 - x_2\left(x_2^2+y_2^2\right) + \delta\left(x_1 - x_2 + y_1 - y_2\right),\\ \frac{dy_2}{dt} &=& -\beta x_2 + y_2 - y_2\left(x_2^2+y_2^2\right) + \delta\left(x_1 - x_2 + y_1 - y_2\right)\end{array}\right. \tag{34}$$

also used in [Fairgrieve and Jepson, 1991]. We computed the same branch of periodic solutions but continued to a larger value of the period (500 instead of 350.) On this branch, the two oscillators are π radians out of phase. We used δ as our continuation parameter while β is fixed at 0.5 and computed periodic solutions near the heteroclinic loop at $\delta = 0.25$. The Floquet multipliers change rapidly in terms of the parameter δ near the heteroclinic loop. For clearness of the figures, we will plot the Floquet multipliers versus the period of the limit cycle. Since the behaviour of the periodic orbit is largely determined by the fixed points on the heteroclinic loop, exponential growth of the Floquet multipliers in terms of the period is expected. Therefore Fig. 3 shows the modulus of the computed Floquet multipliers on a logarithmic scale versus the period. For this figure, we used $NCOL = 4$ (degree of the polynomials,) $NTST = 100$ (number of mesh intervals) and $EPS = 10^{-10}$ (relative error tolerance for the collocation equations). The value of $NTST$ and $NCOL$ is the same as in [Fairgrieve and Jepson, 1991]. All Floquet multipliers computed from the periodic Schur decomposition are real and positive. We connected them using purple lines in Fig. 3. These connections are somewhat arbitrary around the suspected bifurcation points as the Floquet multipliers change very rapidly in a very narrow interval. We could not compute enough points to resolve the picture completely using double precision arithmetic. The Floquet multipliers computed by AUTO97 are indicated using a blue x (real positive), a red + (real negative) and green circles. The circles denote Floquet multipliers at infinity, i.e., the matrix H in Eq. (32) is numerically singular. The three largest Floquet multipliers computed using the periodic Schur decomposition lie on very smooth curves as one would expect. This indicates that these multipliers are probably quite accurate. The curve for the smallest Floquet multiplier remains smooth until $T \approx 120$, but then becomes rather jagged. This is because we have reached the limits of the discretisation. We study this further in Fig. 4, zooming in on the smallest Floquet multiplier. We used mostly the same parameters as Fig. 3 except for a larger stepsize in the continuation code. The mesh was adapted every three points. This corresponds exactly to the jumps of the smallest Floquet multiplier in the figure and proves that the discretisation is the cause of errors in this case and not the convergence of the nonlinear system solver or the periodic Schur procedure. As can be seen in Fig. 3, AUTO97 (using the procedure from [Fairgrieve and Jepson, 1991]) computes the three larger Floquet

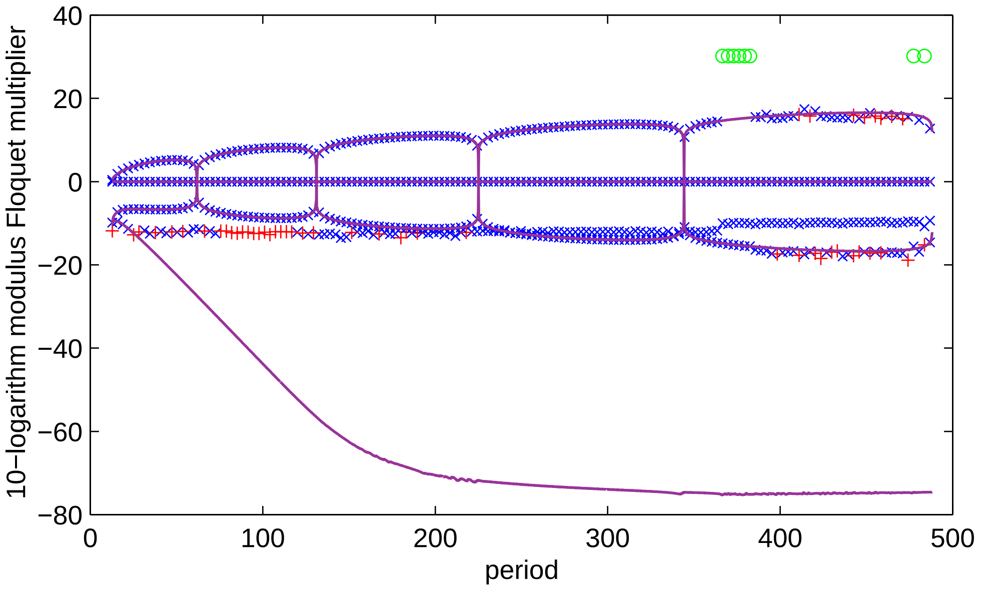

Figure 3: Floquet multipliers versus period for Eq. (34). The Floquet multipliers computed using the periodic Schur decomposition are plotted using purple lines and the multipliers computed by AUTO97 are denoted by a blue x (real positive), a red + (real negative) or a green circle (eigenvalue at infinity.) All Floquet multipliers computed from the periodic Schur decomposition are real and positive. The Floquet multipliers computed using the AUTO97 algorithm are plotted every 25 points.

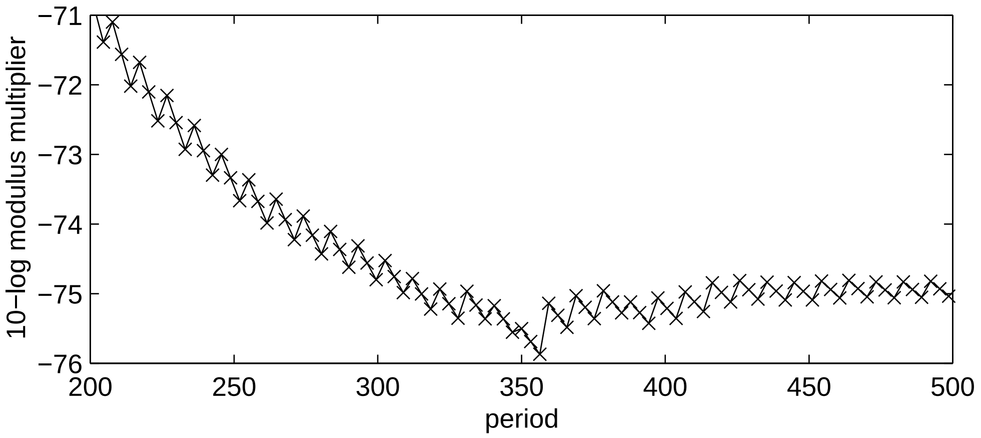

Figure 4: Detail of the smallest Floquet multiplier (computed using the periodic Schur decomposition) versus period for Eq. (34). The computed values are marked with an x and joined in the order of computation.

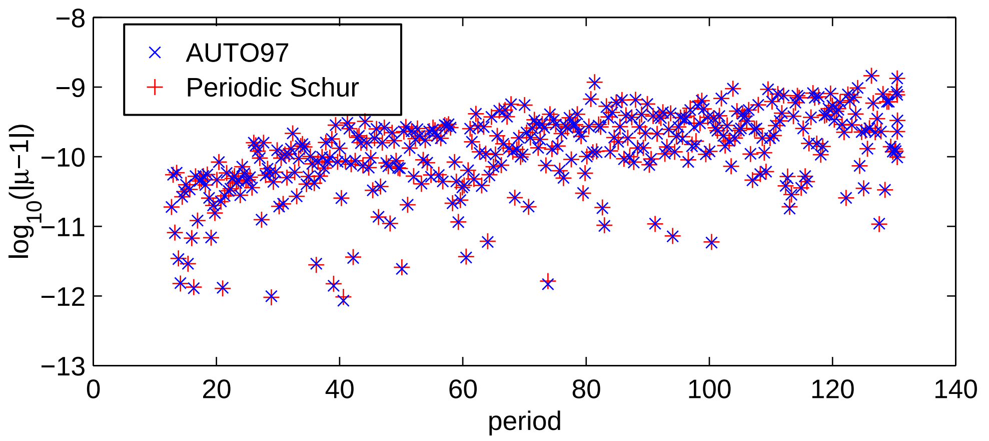

Figure 5: Comparison of the error of the trivial Floquet multiplier for Eq. 34.

$\delta \approx 0.24885$ and $T \approx 131$			$\delta \approx 0.2498551$ and $T \approx 369.17$		
Q^TGZ	Q^THZ	Eigenvalue	Q^TGZ	Q^THZ	Eigenvalue
5.19e-14	1.08e+03	4.8240049249e-17	8.88e-15	8.13e+01	1.0918418108e-16
1.11e-02	3.03e-02	3.6631991184e-01	3.56e+01	3.56e+01	1.0000000119e+00
5.85e+02	7.18e+02	8.1544734383e-01	9.61e-15	1.40e+02	6.8759051187e-17
6.23e+02	6.23e+02	1.0000000006e+00	1.34e+02	0	Infinity

Table 6: Diagonal elements after the QZ decomposition for two matrix pencils.

multipliers correctly until $T \approx 350$. For larger values of the period, the largest and the third Floquet change sign in a random way and lie no longer on a smooth curve. For the smallest Floquet multiplier we find a random pattern of positive and negative multipliers that lie more or less on a line at all parameter values. This random pattern is clearly different from the jagged curve characteristic for discretisation errors and observed when using the periodic Schur decomposition. The nonlinear system solver cannot be the cause either since we obtained much better results with the periodic Schur decomposition. Hence the errors must be inherent to the way AUTO97 computes the Floquet multipliers. We observe the limitations of computing the Floquet multipliers from the generalised eigenvalue problem (32).

Figure 5 shows the error of the trivial Floquet multiplier computed with AUTO97 (blue x) and with the periodic Schur decomposition (red +). Here we used $NCOL = 7$, $NTST = 1000$ and $EPS = 10^{-12}$ to minimize the discretisation errors and errors from inaccurate solution of the collocation equations. The results are essentially the same for both methods. AUTO97 computes the trivial Floquet multiplier with very high accuracy, also for orbits with larger period (where the other Floquet multipliers are inaccurate.) In Sec. 3.1, we already indicated that AUTO tends to compute the Floquet multipliers close to the unit circle from the largest diagonal elements of the QZ-reduced matrices Q^TGZ and Q^THZ. Table 6 shows the diagonal elements and computed Floquet multipliers for two orbits, now again using $NCOL = 4$ and $NTST = 100$. The first orbit is at $\delta \approx 0.24885$ and $T \approx 131$, close to one of the bifurcations along the branch. The second orbit is at $\delta \approx 0.2498551$ and $T \approx 369.17$ and is a point where an eigenvalue at infinity is reported. We did not use the eigenvalue solver in AUTO for this table (which uses a fancy deflation to increase the accuracy of Floquet multipliers close to one in certain cases) but wrote the matrix pencil to disk and processed the matrices with MATLAB.

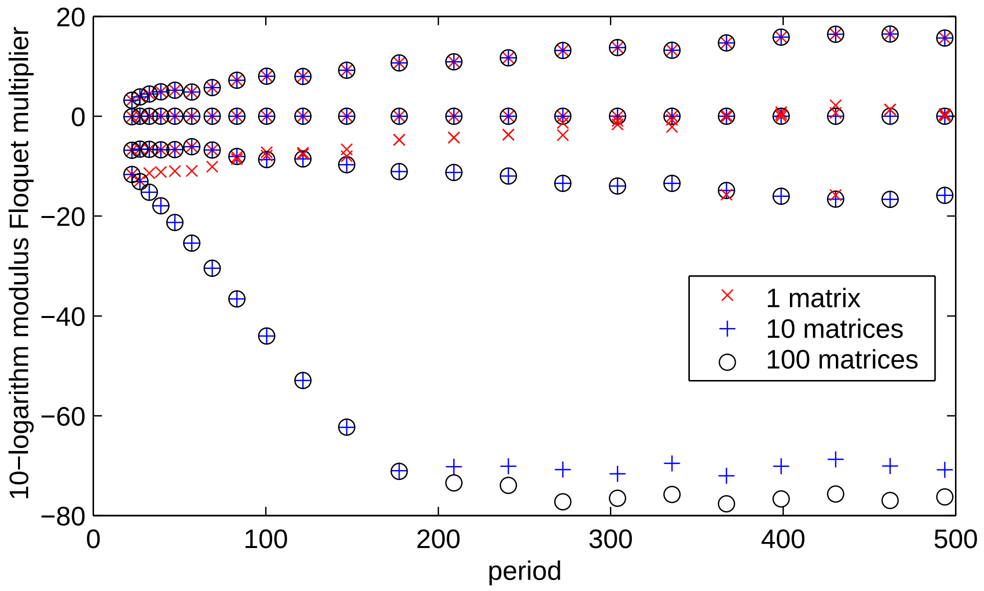

Figure 6: Floquet multipliers computed from the monodromy matrix and a product of 10 and 100 matrices. We used $NTST = 100$, $NCOL = 10$ and $EPS = 10^{-10}$.

In both cases the trivial Floquet multiplier is computed from rather large diagonal elements. The accuracy is high and only limited by the discretisation.

Figure 6 shows the Floquet multipliers computed using the periodic Schur decomposition from the original product of 100 matrices $H_i^{-1}G_i$ and from a product of 10 matrices obtained by explicitly multiplication of the products $H_{10i+10}^{-1}G_{10i+10}\cdots H_{10i+1}^{-1}G_{10i+1}$, $i = 1,\ldots,10$. We have also plotted the eigenvalues computed from the product $H_{100}^{-1}G_{100}\cdots H_1^{-1}G_1$. The latter are equivalent to the Floquet multipliers as they would be computed by AUTO86. The results for 10 matrices illustrate what would happen if one would first condense the blocked subsystem of (26) to a system on the coarse mesh consisting of the mesh points s_0, s_{10}, ... of the original grid and then compute the Floquet multipliers. This would reduce the cost of the computation of the Floquet multipliers somewhat (as discussed in Sec. 3.3,) but also potentially reduce the accuracy. Using 10 matrices, all Floquet multipliers can be computed quite accurately except the smallest one when it becomes smaller than 10^{-70}. When computing the Floquet multipliers from a single matrix, the largest Floquet multiplier is still accurate, but as this Floquet multiplier approaches 10^{20}, the approximations for all other Floquet multipliers including the trivial Floquet multiplier lie far outside the unit circle. One cannot even determine the dimension of the unstable manifold correctly when the period becomes large. Note that AUTO97 produces inaccurate approximations for all but the trivial Floquet multiplier for orbits with a period larger than 350, but the dimension of the unstable manifold was correct in all our computations.

Our last test case is the multiple time scale system consisting of two asymmetric oscillators

$$\left\{\begin{array}{rcl}
\dfrac{dv_1}{dt} & = & -\left(v_1 - a_1\tanh\dfrac{\sigma_1 v_1}{a_1} + q_1 + \omega_{12}\dfrac{v_1-\rho_{12}}{1+\exp\left(-4\gamma_2(v_2-\theta_2)\right)}\right), \\
\dfrac{dq_1}{dt} & = & \varepsilon\left(-q_1 + s_1 v_1\right), \\
\dfrac{dv_2}{dt} & = & -\left(v_2 - a_2\tanh\dfrac{\sigma_2 v_2}{a_2} + q_2 + \omega_{21}\dfrac{v_2-\rho_{21}}{1+\exp\left(-4\gamma_1(v_1-\theta_1)\right)}\right), \\
\dfrac{dq_2}{dt} & = & \varepsilon\left(-q_2 + s_2 v_2\right),
\end{array}\right.$$

studied in [Guckenheimer *et al.*, 2000]. These equations model a pair of reciprocally inhibited neurons in the gastric mill circuit of a lobster. Our continuation parameter is σ_2. The difference in time scales is fixed at $\varepsilon = 10^{-3}$ and the other parameters are set at $\sigma_1 = 3$, $a_1 = a_2 = 1$. $\omega_{12} = \omega_{21} = 0.03$, $r_{12} = r_{21} = -4$, $\gamma_1 = \gamma_2 = 10$, $\theta_1 = \theta_2 = 0.013333$ and $s_1 = s_2 = 1$. The continuation was always started in the direction of decreasing values of σ_2. We used 25000 mesh intervals and polynomials of degree four for the results in the figures. To check the accuracy of the results, we also did some computations using 50000 mesh intervals and polynomials of degree four and 20000 mesh intervals and polynomials of degree seven. The branch of periodic orbits we computed has two very small Floquet multipliers — small enough to cause underflow in IEEE double precision arithmetic — but our code deals with this by representing the modulus of the Floquet multipliers on a logarithmic scale. Comparison with results on the finer mesh or with the higher degree polynomials shows that our results are surprisingly accurate. We believe that our Floquet multipliers have at least three or four correct digits in all points on the computed branch.

Figure 7 shows the Floquet multipliers computed using the periodic Schur decomposition. We display the 10-logarithm of the modulus of the Floquet multipliers and split the vertical axis in three sections with different scaling because of the enormous difference in magnitude of the Floquet multipliers. The left part shows the computed branch. There are three fold points at the left but this can hardly be seen in this figure. There also appears to be an interaction between the second and third largest Floquet multipliers around $\sigma_2 = 1.15690226$ (labeled I_1.) Both Floquet multipliers change sign simultaneously. Since we cannot presently compute eigenvectors, it not completely clear what exactly is happening. The right part of Fig. 7 zooms in onto the Floquet multipliers close to the fold points. It appears that there is an interaction between the two largest non-trivial Floquet multipliers at the fold points. At a fold point, we would expect a second Floquet multiplier 1, but the transition occurs in such a narrow parameter interval that it cannot be computed using double precision arithmetic. Eigenvectors could probably shed more light on what exactly is happening in that area. There is also some interaction between the two smallest Floquet multipliers at the point labeled I_2. Both Floquet multipliers change sign at the same point.

Figure 8 shows the Floquet multipliers computed by AUTO97 during the same run. In the left part of the figure, we have plotted the Floquet multipliers at only one point every twenty points along the branch. The right part again shows a zoom around the fold points. The two largest Floquet multipliers are computed with reasonable accuracy until the largest Floquet multiplier becomes larger than 10^6. We still observe the correct behaviour around the first fold point, but from then on, things start to go wrong. Instead of the two smallest Floquet multipliers, two clouds of values are computed, one around 10^{-14} and one around 10^{-19} and we lose all information about the behaviour around the fold points and the possible interactions between Floquet multipliers.

Finally, Fig. 9 shows the error for the trivial Floquet multiplier. The errors for the periodic Schur decomposition and AUTO97 are comparable. On most of the interval, the error is around 10^{-10} - 10^{-9}, corresponding to the accuracy of the periodic orbits themselves. As we approach the period doubling bifurcation from the right, the error increases to 10^{-6} for both methods. Very near the fold points, the error for the periodic Schur decomposition even grows to 10^{-4} and the periodic Schur decomposition performs worse than the method in AUTO97. The eigenvalue problem is very ill-conditioned near a fold point. In fact, a more detailed plot of the data shows that local maxima of the error are reached at the fold points. The method in AUTO97 uses a special deflation technique (also discussed in [Fairgrieve & Jepson, 1991]) to cope with some of the effects caused by ill-conditioning of the eigenvalue problem near fold points. We have

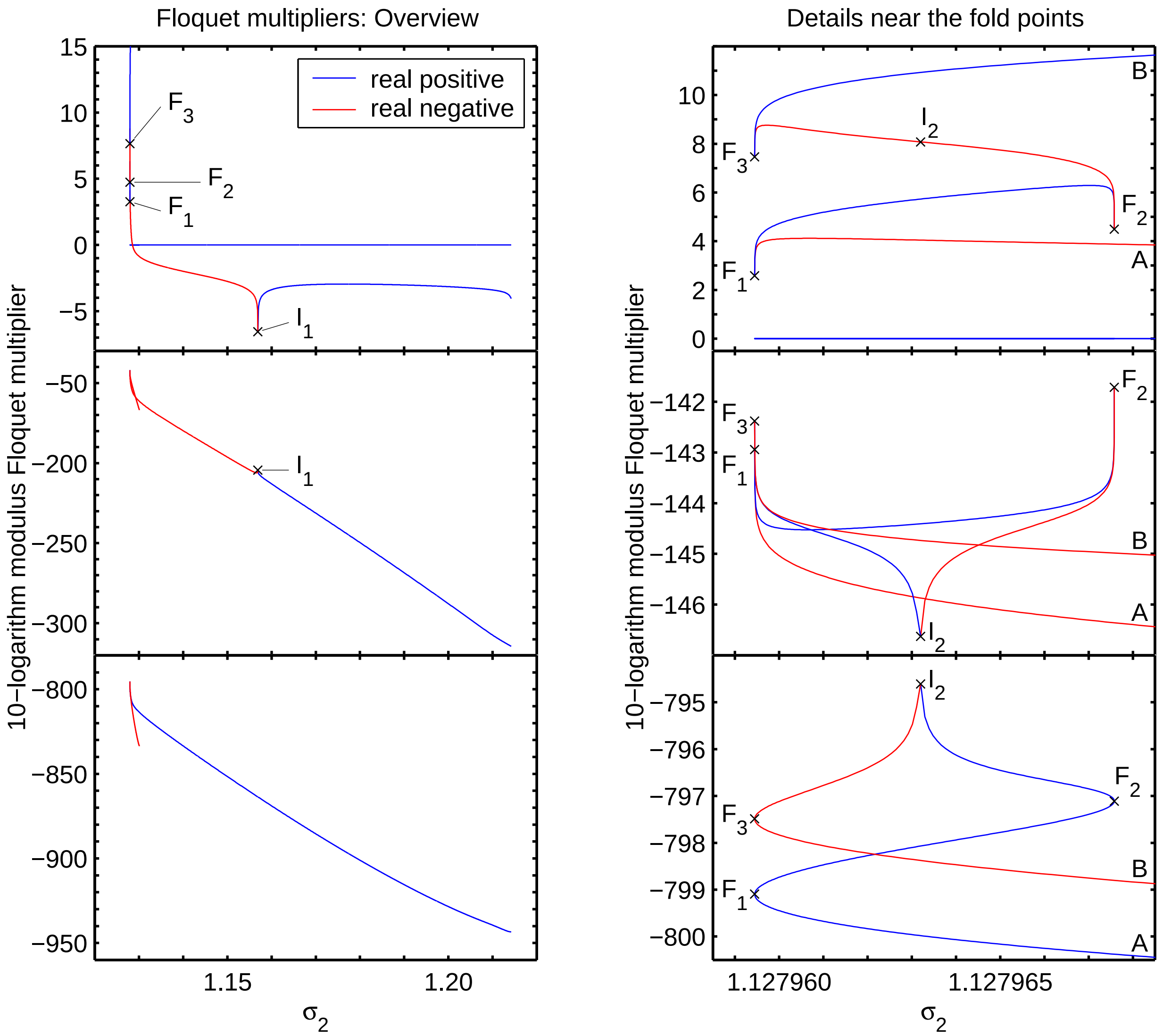

Figure 7: Floquet multipliers computed using the periodic Schur decomposition.

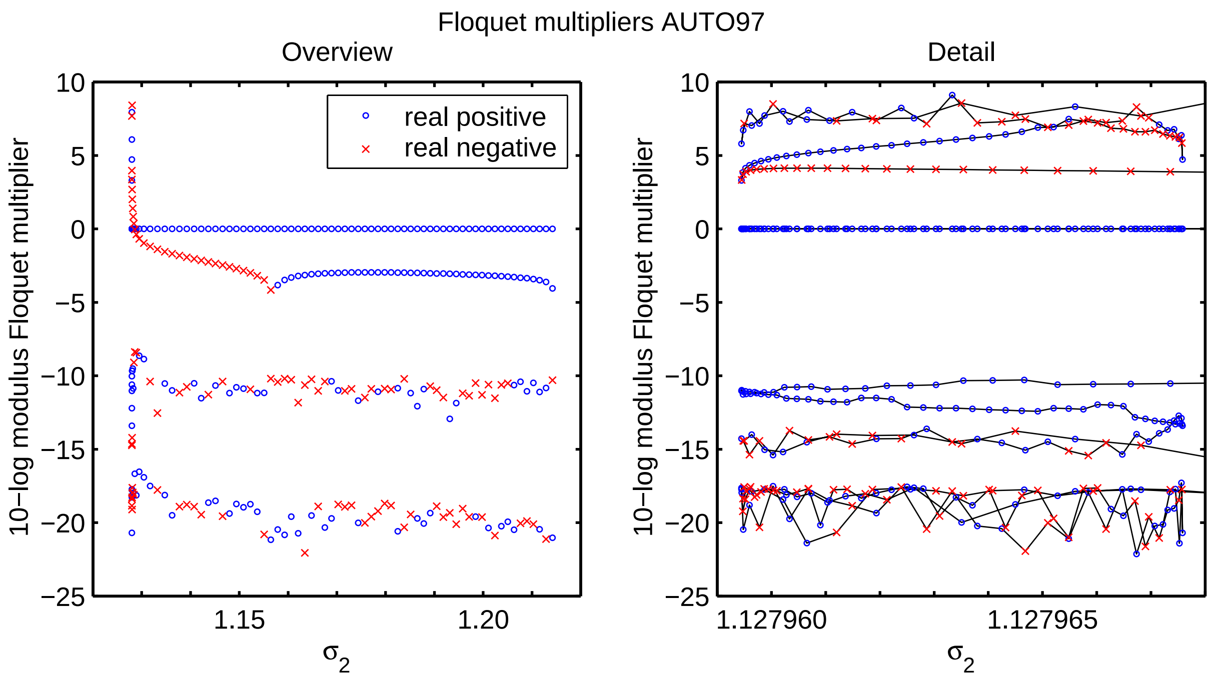

Figure 8: Floquet multipliers computed by the standard AUTO97 routine.

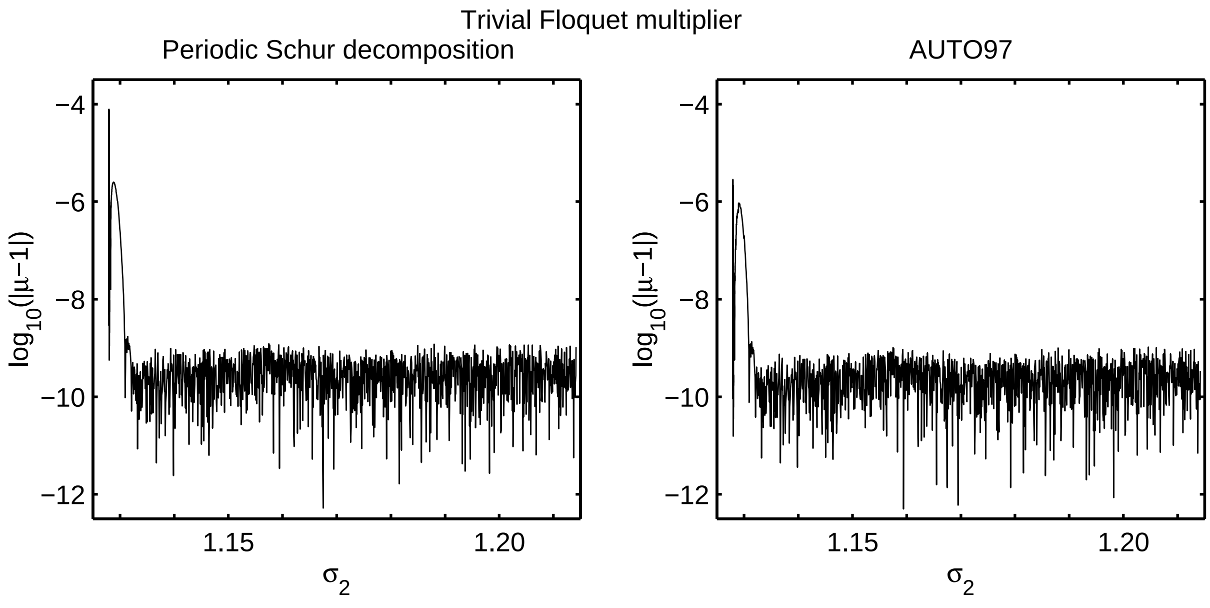

Figure 9: Error of the trivial Floquet multiplier.

no equivalent technique so far for the periodic Schur decomposition. This explains the better results of the AUTO97 algorithm. If we continue the branch beyond the limits used in Figs. 7–9, the error for the trivial Floquet multiplier decreases again to 10^{-10} – 10^{-8} for the periodic Schur decomposition and 10^{-11} — 10^{-9} for the algorithm in AUTO97. It is not clear at the moment why the AUTO97 algorithm performs slightly better in this case.

5 Conclusions

We discussed several algorithms to compute periodic solutions of ODE systems: single- and multiple shooting and global methods such as Gauss–Legendre collocation. In most codes, the Floquet multipliers are computed from the real Schur decomposition of the monodromy matrix which is often easily constructed. One notorious exception to this is AUTO94 and AUTO97. These packages compute the Floquet multipliers from an $N \times N$ generalised eigenvalue problem. All these strategies have similar disadvantages: they compute eigenvector information at only one point and are not capable of computing all Floquet multipliers accurately if some Floquet multipliers are very large or very small. The real Schur decomposition is an excellent match for a single shooting method, but a better method is needed for multiple shooting or global methods. We distinguished two matrix products that commonly arise in codes for periodic solutions. For both products, there is a corresponding variant of the periodic Schur decomposition. When well implemented, the periodic Schur decomposition allows one to compute extremely large and extremely small Floquet multipliers accurately, even if they are too large or small for the floating point representation being used. We developed a code to compute the periodic Schur decomposition of the so-called type-II structure. Our code was developed with multiple shooting methods in mind, but it can be used with AUTO too if the type-I matrix product arising from the linear system solver is first converted to a type-II product.

We have illustrated the effectiveness of the periodic Schur decomposition by studying two systems with very large and very small Floquet multipliers using a modified version of AUTO97. The algorithm in AUTO86 does compute the largest Floquet multiplier accurately and so is good to compute the first bifurcation point where the periodic solutions become unstable. However, it is not capable to compute further bifurcations if the largest Floquet multiplier becomes of the order of ε_{mach}^{-1}. AUTO94 and AUTO97 implement the approach based on a generalised eigenvalue problem proposed by Fairgrieve & Jepson [1991]. Experiments show that Floquet multipliers close to the unit circle are usually very accurate. However, there is no proof that this will always be the case. It just seems that the pivoting strategy used in AUTO generates a generalised eigenvalue problem for which eigenvalues close to the unit circle are well conditioned. Very large and very small Floquet multipliers are usually not well approximated, though in our tests, one could always determine the dimension of the unstable manifold correctly from the computed approximations. All Floquet multipliers are computed with very high precision when using the periodic Schur decomposition. The accuracy is no longer limited by the linear algebra, but by the discretisation of the periodic orbit. So far, there are no error bounds for eigenvalues computed from the periodic Schur decomposition, but our experiments indicate clearly that the accuracy is related to the size of the diagonal elements of the upper triangular factors in the periodic Schur decomposition and not to the size of the diagonal elements in the Schur decomposition of the product of the matrices, though the condition of the eigenvalue problem should also play a role.

AUTO97 does in most cases a good job of computing the Floquet multipliers close to the unit circle and one might wonder whether better techniques are really needed. The periodic Schur algorithm is more expensive than the generalised eigenvalue problem solver in AUTO97, but compared to the cost of computing the periodic orbit, the cost is still moderate. We feel that it is in some cases easier to interpret the evolution of the Floquet multipliers when all multipliers are computed accurately. This and the general increase in reliability make it well worth using a slightly more expensive technique. After all, as long as speed remains acceptable, reliability is the most desirable feature of a computer code. Multiple shooting codes often rely on a standard eigenvalue problem and there we expect a more spectacular improvement, comparable to the improvement obtained with respect to AUTO86. For large N, the periodic Schur decomposition can be combined with subspace iteration (see [Lust, 1997]) to compute a dominant subset of the Floquet multipliers.

Acknowledgment

Kurt Lust is a postdoctoral fellow of the Fund for Scientific Research – Flanders. Part of this work was done while the author was a postdoctoral fellow at the Institute of Mathematics and its Applications of the University of Minnesota and later at the Center of Applied Mathematics of Cornell University, and a visitor at the Mathematics Institute of the University of Warwick. The author acknowledges the support from AFOSR grant F49620-97-0135 at Cornell University. The author also thanks prof. E. Doedel, prof. J. Guckenheimer, prof. A. Spence and prof. P. Van Dooren for the helpful discussions. E. Doedel, J. Guckenheimer, K. Hoffman and W. Weckesser provided us with code for the AUTO test cases. This paper presents research results of the Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture. The scientific responsibility rests with its author.

References

E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney & D. Sorensen [1999] *Lapack User's Guide* (SIAM, Philadelphia,) third edition.

U. Ascher, J. Christiansen & R.D. Russell [1981] "Collocation software for boundary value ODE's," *ACM Transactions on Mathematical Software* **7**(2), 209–222.

U.M. Ascher, R.M.M. Mattheij & R.D. Russell [1995] *Numerical solution of boundary value problems for ordinary differential equations,* volume 13 of *Classics in Applied Mathematics* (SIAM, Philadelphia.)

A. Bojanczyk, G. Golub & P. Van Dooren [1992] "The periodic Schur decomposition. Algorithms and applications," in *Advanced Signal Processing Algorithms, Architectures and Implementations III,* ed. F.T. Luk, pp. 31–42.

W. G. Choe & J. Guckenheimer [1999] "Computing periodic orbits with high accuracy," *Computer Methods in Applied Mechanics and Engineering* **170**, 331–341.

P. Deuflhard & G. Bader [1983] "Multiple shooting techniques revisited," in *Numerical Treatment of Inverse Problems in Differential and Integral Equations,* eds. P. Deuflhard & E. Hairer, volume 2 of *Progress in Scientific Computing* (Birkhäuser Verlag.)

E.J. Doedel, A.R. Champneys, T.F. Fairgrieve, Y.A. Kuznetsov, B. Sandstede & X.J. Wang [1997] *AUTO97: Continuation and Bifurcation Software for Ordinary Differential Equations (with HomCont).*

E. Doedel [1986] "AUTO: Software for continuation and bifurcation problems in ordinary differential equations," report Applied Mathematics California Institute of Technology Pasadena, USA 1986.

T. F. Fairgrieve & A. D. Jepson [1991] "O.K. Floquet multipliers," *SIAM J. Numer. Anal.* **28**(5), 1446–1462.

G.H. Golub & C.F. Van Loan [1996] *Matrix computations* (The Johns Hopkins University Press, Baltimore,) third edition.

J. Guckenheimer & B. Meloon [2000] "Computing periodic orbits and their bifurcations with automatic differentiation," *SIAM J. Sci. Comput.* To appear.

J. Guckenheimer, K. Hoffman & W. Weckesser [2000] "Numerical computation of canards," *Int. J. of Bifurcation and Chaos* **10**(12).

H.B. Keller [1968] *Numerical methods for two-point boundary value problems* (Blaisdell, New-York).

A. I. Khibnik, Yu. A. Kuznetsov, V. V. Levitin & E. N. Nikolaev [1993] "Continuation techniques and interactive software for bifurcation analysis of ODEs and iterated maps," *Physica D* **62**, 360–371.

Yu.A. Kuznetsov [1998] "CONTENT – integrated environment for analysis of dynamical systems. Tutorial," Rapport de Recherche UPMA-98-224 Ecole Normale Superieure de Lyon.

K. Lust [1997] *Numerical bifurcation analysis of periodic solutions of partial differential equations,* Ph.D. thesis Department of Computer Science, K.U.Leuven.

K. Lust [2000] "psSchur: A code for computing the periodic Schur decomposition." In preparation.

C. Moler [2000] "MATLAB incorporates LAPACK. Increasing the speed and capabilities of matrix computing," MATLAB *newsletter winter 2000.*

R. Seydel [1994] *Practical Bifurcation and Stability Analysis. From Equilibrium to Chaos* (Springer-Verlag, New York,) second edition.

R. Seydel [1999] *BIFPACK — a program package for continuation, bifurcation and stability analysis* (First version: Buffalo, 1983, current version: Ulm, 1999.)

B.T. Smith, J.M. Boyle, J.J. Dongarra, B.S. Garbow, Y. Ikebe & V.C. Klema [1976] *Matrix Eigensystem Routines – EISPACK Guide,* volume 6 of *Lecture Notes in Computer Science* (Springer-Verlag, Berlin).

R. Weiss [1974] "The application of implicit Runge–Kutta and collocation methods to boundary value problems," *Math. Comp.* **28**, 449–464.