

COMPUTING INVARIANT TORI AND CIRCLES IN DYNAMICAL SYSTEMS

VOLKER REICHELT*

Key words. Dynamical system, invariant manifold, torus, parametrization, graph transform, bifurcation.

Abstract. We present several algorithms to compute invariant tori in a family of dynamical systems using a continuation strategy. The algorithms are based on the discretization of the graph transform. To circumvent two problems of the standard approach (which requires solving ordinary BVPs) we modify the discretized graph transform. This results in faster and more robust methods.

1. Introduction. A variety of software packages deals with the computation of invariant manifolds in continuous dynamical systems. Many of them (like AUTO [5] for example) consider the one-dimensional case of single trajectories (and their bifurcations). Since the phase space is completely filled with trajectories, only those with special properties are of interest like periodic, homoclinic and heteroclinic orbits. Most of these special trajectories are characterized by their boundary conditions which suggests the use of BVP-solvers for their computation.

Computing higher-dimensional manifolds (or more general: manifolds that consist of more than one trajectory) is the next logical step. The dynamics on such a manifold can be quite complex so that we need a different approach than computing a large number of trajectories via BVP-problems to approximate the manifold.

We look at *hyperbolic manifolds*, i. e., manifolds that fulfill a *gap condition*. This gap characterizes the difference of the dynamics on and outside the manifold (see Section 2.1). The hyperbolicity ensures local uniqueness of the object and usually persists under small perturbations of the system.

The *graph transform* (see Section 2.3), our main tool, is constructed to exploit the gap condition: A manifold is represented as the graph of a function. The graph transform maps this function to another function, using the dynamics of the system, such that the invariant manifold is a fixed point of this operator. The hyperbolicity implies that the operator is well-defined and contracting. This gives rise to a simple algorithm: Start with a rough approximation and iterate the graph transform to obtain the manifold in the limit. In a continuation process this method can be used to compute invariant manifolds in a whole family of systems.

Our main task is to *discretize* the graph transform for numerical purposes (see Section 3, especially 3.3). Unfortunately, the graph transform is computationally expensive. There are mainly three reasons for this: First, the graph transform converges only linearly. Second, the manifolds we want

*Lehrstuhl für Numerische Mathematik, IGPM, RWTH Aachen, Germany

to handle are multi-dimensional. And third, for each graph transform (even in the one-dimensional case) we have to solve several BVPs (at least with the standard approach). Algorithms as in [3] and [2] suffer severely from the slow speed, especially when trying to locate bifurcations. Therefore, we aim for fast algorithms without neglecting robustness (for the latter see Section 3.3.3).

1.1. Setting. Let us consider a family of dynamical systems

$$(1.1) \quad \dot{x} = f_\lambda(x)$$

parametrized by $\lambda \in \mathbb{R}$. For simplicity we assume that each $f_\lambda : \mathbb{R}^N \rightarrow \mathbb{R}^N$ is C^1 , which ensures uniqueness of the solutions. Furthermore we require that all solutions $x(t)$ are defined for all $t \in \mathbb{R}$. The *flow* of the system, $F_\lambda : \mathbb{R}^{1+N} \rightarrow \mathbb{R}^N$, is such that $F_\lambda(t, x_0)$ is the solution of the ODE (1.1) at time t with initial value x_0 . We will write $F_\lambda^t(x_0)$ instead of $F_\lambda(t, x_0)$ so that $F_\lambda^t : \mathbb{R}^N \rightarrow \mathbb{R}^N$, for fixed $t, \lambda \in \mathbb{R}$, is the *time- t -map* of the system. The *invariance condition* for a properly embedded compact C^1 -manifold $M \subset \mathbb{R}^N$ can be written in the form

$$F_\lambda^t(x) \in M \quad \text{for all } x \in M, t \in \mathbb{R}$$

or as a fixed point equation:

$$(1.2) \quad F_\lambda^t(M) = M \quad \text{for all } t \in \mathbb{R}.$$

We want to compute (approximations of) the invariant manifolds M_λ for f_λ in a continuation process. I. e., we start with (an approximation of) an invariant manifold M_0 for f_0 and try to follow the manifold as we change the parameter. One goal is to follow the manifold as long as possible, namely up to a bifurcation, where it is destroyed.

1.2. Discrete systems. Although we focus on continuous dynamical systems, some of the results can also be applied to discrete ones: The time- t -map F_λ^t is a special case of a diffeomorphism $\varphi_\lambda : \mathbb{R}^N \rightarrow \mathbb{R}^N$ in a discrete dynamical system

$$x_{i+1} = \varphi_\lambda(x_i).$$

Here the invariance condition (1.2) transforms to $\varphi_\lambda^n(M) = M$ for all $n \in \mathbb{Z}$. By using only F_λ^t and exploiting (1.2) for our algorithms we can adapt them to discrete systems. Algorithms which try to discretize the equivalent invariance condition

$$f_\lambda(x) \in T_x M \quad \text{for all } x \in M$$

(as in [9]) cannot be adapted to maps as easily.

2. Analytical background. In this section we first discuss a theoretical result due to Fenichel [6]: a perturbation theorem for invariant manifolds. Then, we will have a closer look at the graph transform, which is the main ingredient in the proof of the theorem as well as in our algorithms.

2.1. Normal hyperbolicity, gap condition. Let M be an invariant manifold of the system $\dot{x} = f(x)$ with flow F^t . In order to compare the dynamics inside and outside the manifold we need a splitting $\mathbb{R}^N = TM \oplus NM$ with the tangent space TM of M and some transversal space NM . The tangent space is invariant under DF^t , because M is invariant under F^t . This is in general not true for the transversal space NM , which means that we have for $x \in M$:

$$\begin{aligned} DF^t(x)|_{T_x M} &: T_x M \rightarrow T_{F^t(x)} M, \\ DF^t(x)|_{N_x M} &: N_x M \rightarrow N_{F^t(x)} M \oplus T_{F^t(x)} M. \end{aligned}$$

There are two ways to achieve invariance also for NM . One possibility is to select a special transversal space which is invariant under DF^t (this might be convenient for the proof of the theorem, but would require extra computations in the algorithm, not to mention the question of its existence). The other possibility is to use the projector

$$\Pi(x) : N_x M \oplus T_x M \rightarrow N_x M$$

along $T_x M$ so that the following holds:

$$\Pi(F^t(x)) DF^t(x)|_{N_x M} : N_x M \rightarrow N_{F^t(x)} M.$$

This gives us the freedom to choose a more convenient transversal bundle.

In this work we consider only attractive manifolds. Similar to Lyapunov exponents we define the contraction rate $\theta(x)$ in the transversal direction and the contraction ratio $\rho(x)$ between the transversal and the tangential direction for $x \in M$ as

$$\begin{aligned} \theta(x) &:= \overline{\lim}_{t \rightarrow \infty} \frac{1}{t} \log \sup_{v \in N_x M, \|v\|=1} \|\Pi(x) \cdot DF^t(F^{-t}(x))v\| \\ &= \overline{\lim}_{t \rightarrow \infty} \frac{1}{t} \log \|\Pi(x) \cdot DF^t(F^{-t}(x))\|, \\ \rho(x) &:= \overline{\lim}_{t \rightarrow \infty} \frac{\frac{1}{t} \log \inf_{w \in T_x M, \|w\|=1} \|DF^t(F^{-t}(x))w\|}{\frac{1}{t} \log \sup_{v \in N_x M, \|v\|=1} \|\Pi(x) \cdot DF^t(F^{-t}(x))v\|} \\ &= \overline{\lim}_{t \rightarrow \infty} \frac{\log \|[D(F^t|_M)(F^{-t}(x))]^{-1}\|^{-1}}{\log \|\Pi(x) \cdot DF^t(F^{-t}(x))\|} \\ &= - \overline{\lim}_{t \rightarrow \infty} \frac{\log \|D(F^{-t}|_M)(x)\|}{\log \|\Pi(x) \cdot DF^t(F^{-t}(x))\|}. \end{aligned}$$

Measuring the ratio $\rho(x)$ this way allows a more subtle investigation of the dynamics than just taking the ratio of two limits. To describe the behavior of the flow in a neighborhood of the whole manifold we take the suprema:

$$\theta := \sup_{x \in M} \theta(x) \quad \text{and} \quad \rho := \sup_{x \in M} \rho(x).$$

Both the attractivity of the manifold and the gap condition can be written in terms of these expressions:

$$\theta < 0 \quad \text{and} \quad \rho < 1.$$

A manifold M with this property is called *attractive* and *eventually normally hyperbolic* (in the following referred to as *hyperbolic*). The manifold is said to be *eventually k -normally hyperbolic* for $k \in \mathbb{N}$ if $\rho < \frac{1}{k}$, which means that the gap between the contraction rates in the tangential and in the normal direction is even larger. The term “eventually” describes the fact that the dynamics is characterized in the limit $t \rightarrow \infty$. Due to the compactness of the manifold there is a fixed $t_0 > 0$ and an $\varepsilon > 0$ such that

$$\frac{1}{t} \log \|\Pi(x) \cdot DF^t(F^{-t}(x))\| < -\varepsilon, \quad -\frac{\log \|D(F^{-t}|_M)(x)\|}{\log \|\Pi(x) \cdot DF^t(F^{-t}(x))\|} < 1 - \varepsilon$$

holds for all $t \geq t_0$, $x \in M$. The manifold M is then *immediately normally hyperbolic* under those F^t .

The choice of the transversal bundle or the norms on TM and NM have no effect on θ and ρ (see Proposition 1 in [6, p. 203]). But the transversal bundle is important for (the discretization of) the graph transform (see Sections 2.3 and 3.3) and hence for the algorithm.

2.2. Perturbation theorem. We can now formulate the existence theorem for invariant manifolds of perturbed systems according to [6]. Theorem 1 in [6, p. 205] deals with the more general concept of an overflowing invariant manifold with boundary. For our purposes a slightly simplified version is sufficient. It could also be regarded as a version of Theorem 3 in [6, p. 215] without expanding directions:

THEOREM 2.1. *Let \tilde{f} be a C^k -vector field on \mathbb{R}^N . Let \tilde{M} be a compact, connected C^k -manifold, properly embedded in \mathbb{R}^N and invariant under \tilde{f} . Suppose $\theta(x) < 0$ and $\rho(x) < \frac{1}{k}$ for all $x \in \tilde{M}$. Then for any C^k -vector field f in some C^1 -neighborhood of \tilde{f} there is a manifold M invariant under f and C^k -diffeomorphic to \tilde{M} .*

The proof of the theorem is based on the graph transform (see Section 2.3). To show the existence of a C^k -manifold M we need a transversal bundle of \tilde{M} which is also C^k . Because the algorithms deal with approximations of manifolds, it will be difficult to obtain any smoothness results from the numerical data. Hence, it is sufficient for us to consider only the case $k = 1$. If M exists and if we are not interested in smoothness results,

it is not important which bundle we chose for the computation. In the algorithms, for computational purposes including speed, we can use the normal bundle, which is only continuous.

2.3. Graph transform. Let M be the (unknown) invariant manifold of the system $\dot{x} = f(x)$ and let \tilde{M} be a (known) approximation for M (diffeomorphic to M). To define a local coordinate system around \tilde{M} we need a splitting $\mathbb{R}^N = T\tilde{M} \oplus N\tilde{M}$, where $N\tilde{M}$ is some sufficiently smooth transversal bundle. The domain $D \subset \mathbb{R}^N$ of the coordinate system is a neighborhood of \tilde{M} . The graph of a differentiable function

$$m_i : \tilde{M} \rightarrow N\tilde{M}$$

is a manifold $M_i \subset D$ diffeomorphic to \tilde{M} . Or vice versa: A manifold $M_i \subset D$ which is diffeomorphic to \tilde{M} and intersects $N\tilde{M}$ transversally can be represented as the graph of such a function m_i .

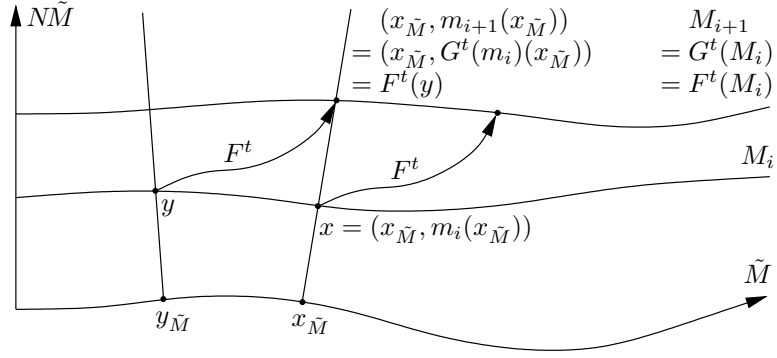


FIG. 1. Graph transform.

The graph transform $G^t = G^{t, \tilde{M}, N\tilde{M}}$, for some fixed $t > 0$, maps m_i to a function $m_{i+1} := G^t(m_i)$ of the same type such that

$$\text{graph}(m_{i+1}) = \text{graph}(G^t(m_i)) = F^t(\text{graph}(m_i))$$

holds. Using the projectors

$$\begin{aligned} P_{\tilde{M}} : D &\rightarrow \tilde{M}, \\ P_{N\tilde{M}} : D &\rightarrow N\tilde{M} \end{aligned}$$

this can be written in the local coordinates for each point: The value $m_{i+1}(x_{\tilde{M}})$ for $x_{\tilde{M}} \in \tilde{M}$ is defined by

$$m_{i+1}(x_{\tilde{M}}) := P_{N\tilde{M}} F^t(y) \quad \text{with} \quad y = (y_{\tilde{M}}, m_i(y_{\tilde{M}})) \in M_i$$

where $y_{\tilde{M}} \in \tilde{M}$ is chosen such that the condition $x_{\tilde{M}} = P_{\tilde{M}} F^t(y)$ is satisfied. For simplicity we will denote $\text{graph}(m_{i+1})$ with $G^t(M_i)$.

The normal hyperbolicity of the invariant manifold has several important consequences. First, it implies that the graph transform is well-defined for all m_i from a subset of functions whose slope is limited by some fixed constant: The attractivity ensures that the image $F^t(\text{graph}(m_i))$ stays close to M for $t \geq t_0$. This prevents the image from leaving D , because M is close to \tilde{M} . The gap condition limits the slope of the image, because the attraction in the direction of $N\tilde{M}$ or NM is stronger than in the direction of \tilde{M} or M . In particular, $F^t(M_i)$ is transversal to $N\tilde{M}$ so that it can be represented as a function m_{i+1} in the local coordinate system and m_{i+1} is in the same subset of functions as m_i .

Second, the attractivity of M ensures that G^t is a contracting operator in C^0 -topology, so that the existence of a fixed point is guaranteed. Obviously we have $G^t(M_i) = F^t(M_i)$ for any M_i , which implies

$$G^t(M) = M,$$

i. e., M is the fixed point. As another consequence of the contraction operator theorem we can start with some M_0 in the neighborhood of M (for example $M_0 := \tilde{M}$) and apply the graph transform iteratively to obtain M as the fixed point of G^t in the limit.

And third, using the gap condition one can show C^1 - or even C^k -smoothness for the manifold (for details see [6]).

Sometimes it is useful to regard the graph transform in the following way: At first, the time- t -map F^t is applied to M_i . The image points are parametrized by their inverse image. In a second step the graph transform reparametrizes the image of F^t so that the points are parametrized by their position in the local coordinate system and not by their initial value. This second step prevents the parametrization from getting worse in each step of the iteration. If we just iterate the time- t -map F^t , the parametrization usually breaks down in the limit.

The computed manifold is independent of the local coordinate system. The latter only changes the parametrization. Therefore, it is not even required to choose the same coordinate system in each iteration. We just need a reparametrization strategy against the breakdown. A method to achieve this without using the local coordinate system is presented in Section 3.3.3. This is very useful if the “ordinary” graph transform is not well-defined. The latter can have two reasons: $F^t(M_i)$ could partially lie outside the domain of the local coordinate system or it is completely inside, but it is not the graph of a function. Either situation can happen if changing the parameter causes quick changes of the shape of the manifold or if only a rough approximation for the initial step is known.

For our first two algorithms (see Sections 3.3.1 and 3.3.2) we chose the normal bundle, because it is easy (fast) to handle and gives a fairly good parametrization. To use a foliation that is (almost) invariant under the flow (see [2]) is another — much more expensive — possibility. To prove smoothness results, the normal bundle is not sufficient, because we would

lose one derivative. In this case a transversal bundle as smooth as \tilde{M} is needed.

Because the graph transform converges only for \tilde{M} close to M , choosing an appropriate \tilde{M} is of major importance. In a continuation setting a natural choice for \tilde{M} is the manifold computed for the previous λ value.

3. Numerics. The main task is to discretize the manifold and the graph transform in such a way that the resulting algorithm is robust, accurate and fast.

From now on we will restrict ourselves to invariant 2-tori, although this is necessary only for the Sections 3.1 and 3.2. Section 3.3 can easily be adapted to more general manifolds.

3.1. Discretizing the torus. In order to discretize a given torus M we need a suitable representation. To achieve this, we introduce a diffeomorphism from the unit-square $\Omega := [0, 1]^2 \cong (\mathbb{R} \bmod 1)^2$ with “wrap-around” onto $M \subset \mathbb{R}^N$:

$$\omega_M : \Omega \rightarrow M.$$

However, this mapping is not unique. For analytical purposes, it might not matter, but for numerical reasons we want $\text{cond}(D\omega_M)$ to be small. Often the mapping ω_M is given for the initial torus. For example

$$\omega_M(z) = (\sin(2\pi z_1), \cos(2\pi z_1), \sin(2\pi z_2), \cos(2\pi z_2))^T \quad \text{for } z \in [0, 1]^2$$

is a natural choice in the case of the coupled oscillators (Section 5.1) for $\lambda = 0$. Unfortunately, the parametrization of M worsens in general for each new value of the continuation parameter. How to compute a better and somehow unique parametrization is discussed in Section 3.2.

To discretize M we replace the function ω_M by a spline ω . For simplicity we take a rectangular grid (equidistant in each direction) on the unit-square Ω for bicubic spline interpolation. The set of grid points will be denoted by Γ . The simplicity of the data structure is essential for the performance of the algorithm.¹ In addition the bicubic spline offers the possibility to refine the grid locally, but this feature has not been implemented yet.

The spline is uniquely determined by prescribing the values $\omega(z)$ for $z \in \Gamma$ together with the derivatives $\frac{\partial}{\partial z_1}$, $\frac{\partial}{\partial z_2}$ and $\frac{\partial}{\partial z_1} \frac{\partial}{\partial z_2}$. It will be at least C^1 for any choice of the derivatives.

If only the $\omega(z)$ are given, one can compute values for the derivatives such that the spline will be C^2 . Experiments have shown however, that this is not a good choice: Computing these “artificial” derivatives is slow and they are most likely responsible for additional interpolation errors.

¹Evaluating the spline ω is one of the two most expensive parts of the algorithm, because it is used very often. The other part is solving the ODE.

Instead, we follow [3] and compute the derivatives $\frac{\partial}{\partial z_1}$, $\frac{\partial}{\partial z_2}$ and $\frac{\partial}{\partial z_1} \frac{\partial}{\partial z_2}$ of ω as finite differences of the $\omega(z)$ with $z \in \Gamma$ to approximate the derivatives of ω_M . This results in a bicubic C^1 -spline that does not suffer from the problems above.

3.2. Reparametrizing the torus. The parametrization of the manifold plays an important role for the algorithm. The better the parametrization, the more accurate the results. The consequences of a very bad parametrization are dramatic: In the worst case it can cause a breakdown of the discretized graph transform. In many cases a good parametrization is analytically given for the initial manifold, but during the continuation it will gradually get worse with each λ step. To avoid this we have to improve the grid via some reparametrization strategy.

For curves we can use arclength to get a canonical and unique parametrization (modulo the choice of an initial point and the direction). In higher dimensions the situation is more difficult. A generalization to the torus is a conformal parametrization. The following approach was already used by Moore [9], and we recall it for completeness.

Let M be a torus that is parametrized by a function $\omega : \Omega \rightarrow M$. Consider two curves in Ω both passing through the point z where they have tangent vectors v_1 and v_2 . The images of both curves under ω intersect in $\omega(z)$ at an angle α which is given by (for simplicity we suppress z in the notation):

$$\cos \alpha = \frac{\langle D\omega \cdot v_1, D\omega \cdot v_2 \rangle}{\|D\omega \cdot v_1\|_2 \|D\omega \cdot v_2\|_2} = \frac{\langle v_1, Av_2 \rangle}{\sqrt{\langle v_1, Av_1 \rangle \langle v_2, Av_2 \rangle}}$$

$$\text{with } A = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \frac{1}{\sqrt{\det(D\omega^T D\omega)}} D\omega^T D\omega.$$

Of course, any multiple of A will do, but this way we get a unique A with $\det(A) = 1$. The idea is to construct a diffeomorphism $\bar{\omega} : \Omega \rightarrow \Omega$ which has the same fundamental functions E , F and G as ω . Then $\omega_{\text{new}} := \omega \circ \bar{\omega}^{-1} : \Omega \rightarrow M$ will map two curves in Ω intersecting at the angle α to curves on M intersecting at the same angle α . Thus, ω_{new} is conformal.

One can easily check that the function $\bar{\omega} := (x, y)^T$, whose components $x = x(z)$ and $y = y(z)$ are the solutions of the Laplace-Beltrami-system

$$\nabla x = \begin{pmatrix} -F & E \\ -G & F \end{pmatrix} \nabla y,$$

has the desired property. As boundary conditions for x and y we require periodicity in one direction and a difference of 1 in the other (the coordinates remain constant in one direction, but increase in the other). Of course, x and y are only determined up to a constant, but this corresponds to the freedom that we have in the choice of $\bar{\omega}(0) = (x(0), y(0))^T$. By using

Schwarz's lemma we see that both x and y fulfill the elliptic equation

$$\frac{\partial}{\partial z_2} \left[-F \frac{\partial u}{\partial z_1} + E \frac{\partial u}{\partial z_2} \right] = \frac{\partial}{\partial z_1} \left[-G \frac{\partial u}{\partial z_1} + F \frac{\partial u}{\partial z_2} \right],$$

but with different boundary conditions. The associated variational problem with test-function v reads

$$(3.1) \quad \int_{\Omega} \langle \nabla u, A^{-1} \nabla v \rangle dz = 0.$$

Let us go back to the numerics at this point: The function ω is a (bicubic) spline on a rectangular grid. To solve (3.1) we use the finite element method on the same grid with bilinear splines for u and v . The 9-point-stencil for the discretization that we use differs slightly from the one used in [9, p. 2347], because we approximate A^{-1} by a bilinear spline, too. We then have to solve two linear systems with different right sides due to the two boundary conditions. The matrix is n -by- n , where n is the number of grid-points $|\Gamma|$, and has a single zero-eigenvalue (corresponding to the free choice of $x(0)$ and $y(0)$ respectively). Because the matrix is large but sparse (9 entries per row), we use the conjugate gradients method to solve the linear systems.

Finally, we have to compute $\omega_{\text{new}}(\Gamma) = \omega(\bar{\omega}^{-1}(\Gamma))$ and interpolate through these new grid points. The first step is to determine the pre-images of the n grid-points under $\bar{\omega}$ by Newton's method. Afterwards ω is applied. The resulting spline ω_{new} and ω will in general represent slightly different manifolds. The difference is usually the bigger the worse the parametrization ω was.

In most cases, the graph transform is used before the reparametrization. To reduce the errors caused by a "bad" parametrization ω some steps of the graph transform should be applied after reparametrizing.

3.3. Discretizing the graph transform. Having discretized the manifold we can now focus on the graph transform. Let \tilde{M} , represented by the spline $\tilde{\omega}$, be a (rough) approximation for the unknown invariant manifold M . As transversal bundle $N\tilde{M}$ we choose the normal bundle. For a chosen time-step $t > 0$ the discretized graph transform will be denoted by G_{Discr}^t . It is applied to the manifold M_i , represented by ω_i , in the neighborhood of \tilde{M} giving $M_{i+1} := G_{\text{Discr}}^t(M_i)$. This is done by computing a new spline $G_{\text{Discr}}^t(\omega_i) = \omega_{i+1}$ such that

$$(3.2) \quad \omega_{i+1}(z_{\Gamma}) \in F^t(M_i) \quad \text{and} \quad \omega_{i+1}(z_{\Gamma}) \in N_{\tilde{\omega}(z_{\Gamma})} \tilde{M}$$

holds at least approximately for every grid point $z_{\Gamma} \in \Gamma \subset \Omega$. The first condition ensures $M_{i+1} \approx F^t(M_i)$ and the second gives the proper parametrization in the local coordinate system. Computing the normal bundle for

the second condition explicitly (as in [3]) is rather expensive. Instead, we check the latter via the standard inner product. We only need to satisfy

$$(3.3) \quad \left\langle \omega_{i+1}(z_\Gamma) - \tilde{\omega}(z_\Gamma), \frac{\partial \tilde{\omega}}{\partial z_j}(z_\Gamma) \right\rangle = 0, \quad j = 1, 2.$$

The partial derivatives of $\tilde{\omega}$ are needed for the spline interpolation anyway, so we get them “for free”. The different algorithms presented here use different strategies to satisfy these conditions approximately.

3.3.1. Shooting. This is the “standard” method: To satisfy the first condition of (3.2) we write $G_{\text{Shoot}}^t(\omega_i)(z_\Gamma) = \omega_{i+1}(z_\Gamma) = F^t(\omega_i(z))$ and solve

$$\left\langle F^t(\omega_i(z)) - \tilde{\omega}(z_\Gamma), \frac{\partial \tilde{\omega}}{\partial z_j}(z_\Gamma) \right\rangle = 0, \quad j = 1, 2,$$

via Newton’s method for $z \in \Omega$ to satisfy the second one. The two derivatives of $F^t \circ \omega_i$ with respect to z_j which we need can be approximated by finite differences. Essentially, this is a shooting algorithm to solve a

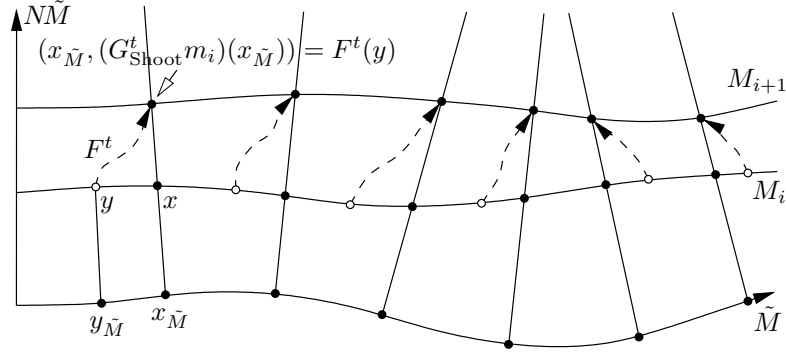


FIG. 2. Discretized graph transform based on shooting with $x = \omega_i(z_\Gamma)$, $y = \omega_i(z)$, $x_{\tilde{M}} = P_{\tilde{M}}(x) = \tilde{\omega}(z_\Gamma)$ and $y_{\tilde{M}} = P_{\tilde{M}}(y) = \tilde{\omega}(z)$.

BVP. Unfortunately, shooting is very expensive with respect to computation time, because we have to solve three IVPs in each step (one for F^t and two for the finite differences that approximate $\frac{\partial}{\partial z_j}(F^t \circ \omega_i)$).

However, in case of a discrete dynamical system we only have to apply the diffeomorphism three times. Maybe we even know the derivatives and can use them instead of the finite differences. Then the algorithm becomes very inexpensive given the quadratic convergence of Newton’s method.

3.3.2. Two steps. In order to significantly speed up the computation we now present an algorithm based on IVPs instead of BVPs. The idea is to split the graph transform in two separate steps: following the flow and reparametrizing the image (as in Section 2.3). At first we compute the

spline $\bar{\omega}_{i+1}$ through the points $F^t(\omega_i(\Gamma))$. The image \bar{M}_{i+1} of the spline approximates $F^t(M_i)$. The task of finding points $G_{\text{TwoSteps}}^t(\omega_i)(z_\Gamma) = \omega_{i+1}(z_\Gamma) := \bar{\omega}_{i+1}(z)$ on \bar{M}_{i+1} that satisfy the second condition of (3.2) is again performed by Newton's method: It is applied for every grid point $z_\Gamma \in \Gamma$ to the equation

$$\left\langle \bar{\omega}_{i+1}(z) - \tilde{\omega}(z_\Gamma), \frac{\partial \tilde{\omega}}{\partial z_j}(z_\Gamma) \right\rangle = 0, \quad j = 1, 2,$$

with unknown $z \in \Omega$. For every step of Newton's method we now only have to evaluate the spline $\bar{\omega}_{i+1}$ (together with two derivatives) at one point instead of solving three IVPs. In typical examples this reduces the computation time by one order of magnitude.

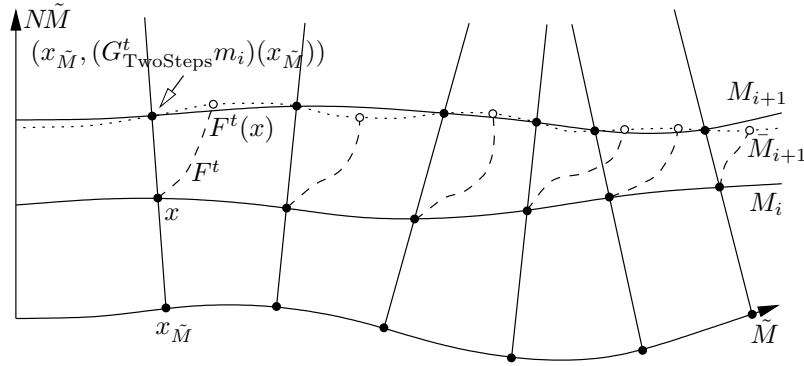


FIG. 3. Discretized graph transform based on IVPs.

Unfortunately, we have to pay for the speed-up: Every iteration of the graph transform in the shooting algorithm G_{Shoot}^t needs two interpolations: One to evaluate ω_i and the other one to get M_{i+1} . This introduces two errors in each step. The error of the first interpolation is damped by the application of F^t , since F^t is a contraction in the normal direction (and we do not bother about the error in the tangent direction).

In every iteration of G_{TwoSteps}^t we also have to interpolate twice: The first time to obtain \bar{M}_{i+1} and the second time to obtain M_{i+1} . But here the application of F^t does not have any damping effects, because it is performed prior to both interpolations. Lacking the damping effect in the two steps method we will get slightly larger errors. Fortunately, they are of minor importance compared to the gained speed-up with respect to the shooting method (see Section 5.1.3).

A much greater problem occurs if the distance between two points $F^t(\omega_i(z_1))$ and $F^t(\omega_i(z_2))$ for two neighbors z_1, z_2 on the grid Γ becomes too large or too small. In these cases \bar{M}_{i+1} is only a bad approximation for $F^t(M_i)$. This can lead to a breakdown of the algorithm, because the discretized graph transform might lose its contractivity.

Roughly speaking, this behavior is a sign of an ill-conditioned problem. In this situation the flow is very sensitive with respect to the initial data: Small changes of $x \in M_i$ lead to large changes of $F^t(x)$. Of course, the shooting algorithm suffers also from this fact, but in a different way (which might be less harmful): The inverse images of the flow (on M_i) lie closer together. But if small errors on M_i cause big errors on the image under the flow, the error caused by the interpolation of ω_i becomes a problem.

The sensitivity can be controlled to some extent by choosing smaller values for t such that F^t shows only moderate sensitivity. However, we must not choose t too small, otherwise M could have repelling parts under the action of F^t . Because of the gap condition there is fortunately an interval for t where M is attractive, but where the time- t -map F^t does not show too much sensitivity. This is an important difference with respect to discrete dynamical systems, where the map φ is fixed. Hence, for maps the shooting algorithm is recommended, because it suffers less from sensitivity of the map. In addition, it is less expensive, if the evaluation of the map is not too costly.

3.3.3. Floating. For the algorithms described above it is essential that the computed manifolds lie in the domain of the local coordinate system. But there are three important cases where this assumption is violated: First, if \tilde{M} is “rough”, the domain where the coordinate system is well-defined can be very small. Second, the shape of the manifold can change very quickly with the variation of λ . This phenomenon can often be seen in the neighborhood of a bifurcation. And third, sometimes only a very rough approximation of the initial manifold is known.

In the first two cases we could decrease the step size $\Delta\lambda$ of the continuation parameter to stay inside the domain of the coordinate system. Sometimes however, this leads to unreasonably small step sizes which, of course, slow down the algorithm. Here a different approach could reduce the computation time considerably. In the third case we need a new approach to find the manifold at all.

The idea is that we do not need a fixed coordinate system for the reparametrization step in the graph transform at first. We just have to find a good parametrization. We therefore modify the previous algorithm in the following fashion: Having computed \tilde{M}_{i+1} we do not parametrize the manifold with respect to the local coordinate system, but use the reparametrization strategy explained in Section 3.2 to obtain $M_{i+1} = G_{\text{Floating}}^t(M_i)$.

Due to the reparametrization step in every iteration this method is quite slow. To speed up the computation we recommend to apply only some iterations of this version of the graph transform and switch back to one of the previous algorithms when M_i is close enough to M .

Lacking a theorem like Theorem 2.1 there is no guarantee that the floating algorithm will converge towards anything. Hence (and because it is slow), it should only be used when other algorithms fail or perform badly.

Numerical experiments have shown that this method is capable of following a torus closer to a bifurcation in situations where the shape of the torus changes rapidly (close to a Hopf-bifurcation for example, see Section 5.2). In this sense the floating algorithm is more robust than the other two.

4. Implementation. The continuation algorithm mainly consists of two loops. In the inner loop the graph transform is iterated. It ends successfully if the difference between two iterates M_i and M_{i+1} is smaller than some tolerance. If this is not the case after some prescribed number of iterations or if Newton's method fails to determine the grid points for M_{i+1} , it finishes unsuccessfully. The outer loop controls the parameter λ : We start with an initial value λ_0 and a step size $\Delta\lambda$. After each successful iteration of the inner loop we increase λ by $\Delta\lambda$. If the inner loop finishes without success, we halve the step size $\Delta\lambda$ and restart with the last successfully computed manifold. If the step size is decreased beyond some minimal size or λ reaches a given λ_{end} , the outer loop terminates.

To achieve high accuracy we use a high order solver for the initial value problems. Because we integrate only over small time intervals, we took an explicit Runge-Kutta method by Dormand and Prince (see [8, p. 181]). We modified the code DOP853 based on it from Hairer and Wanner [8, p. 481] to gain some speed.

Since the algorithms have not been implemented for invariant circles yet, the circles are computed as tori: We add two equations with a hyperbolic periodic orbit to the system. The product of both circles forms a torus, which can be computed without any modification of the program. This is of course very inefficient and introduces additional interpolation errors (compared to an implementation designed for circles). Therefore, we do not discuss the computation time of the algorithms for circles in the following section as we do for tori.

5. Examples. As first example, we consider two oscillators with linear coupling. This system has several symmetries, which allow us to determine some bifurcations exactly so that we can compare them with the numerical results. In addition, there is not only an invariant torus, but also two invariant circles which we can try to follow.

The second example is the forced van der Pol oscillator. This system allows us to study several scenarios for the breakdown of a torus. This includes one scenario where the floating method can follow the torus much closer to a bifurcation than the other methods.

5.1. Coupled oscillators. The system

$$\begin{aligned}\dot{x}_1 &= \beta x_2 + (1 - x_1^2 - x_2^2)x_1 - \lambda(x_1 + x_2 - x_3 - x_4), \\ \dot{x}_2 &= -\beta x_1 + (1 - x_1^2 - x_2^2)x_2 - \lambda(x_1 + x_2 - x_3 - x_4), \\ \dot{x}_3 &= \beta x_4 + (1 - x_3^2 - x_4^2)x_3 + \lambda(x_1 + x_2 - x_3 - x_4), \\ \dot{x}_4 &= -\beta x_3 + (1 - x_3^2 - x_4^2)x_4 + \lambda(x_1 + x_2 - x_3 - x_4)\end{aligned}$$

with $\beta > 0$ is thoroughly investigated in [1]. Numerical computations of the invariant torus of this system can be found in [3] and [9] for example. The reason for the breakdown of the torus is discussed in [1] and [4].

Without coupling ($\lambda = 0$) the system consists of two independent oscillators, each with a circle of radius 1 (S^1) as attractive periodic orbit of period $\frac{2\pi}{\beta}$. Together both oscillators have an attractive invariant torus $M \cong S^1 \times S^1$ consisting of a one-parameter family of periodic orbits. On the torus we have no attractivity so that two (and not just one) of the Floquet multipliers of each orbit are 1. The other two multipliers are smaller than 1, due to the attractivity of the torus. As a consequence we have an attractive, normally hyperbolic invariant torus, which will persist under small perturbations (for small λ). For a detailed analysis we fix $\beta = 0.55$ in the following.

5.1.1. Invariant circles. Because of symmetry the system has two invariant planes $P_1 = \{x \in \mathbb{R}^4 \mid x_1 = x_3, x_2 = x_4\}$ and $P_2 = \{x \in \mathbb{R}^4 \mid x_1 = -x_3, x_2 = -x_4\}$ independent of λ which both intersect the torus transversally. The invariant circles of these intersections will persist under small perturbations of the system. They are the only circles that survive the change of the parameter. For simplicity we will suppress the last two coordinates x_3, x_4 in our notation when considering the planes P_1 or P_2 .

The *in-phase orbit* (see [1]) lies in the plane P_1 where the ODE reduces to

$$\begin{aligned}\dot{x}_1 &= \beta x_2 + (1 - x_1^2 - x_2^2)x_1, \\ \dot{x}_2 &= -\beta x_1 + (1 - x_1^2 - x_2^2)x_2.\end{aligned}$$

This attractive periodic orbit exists for all λ , because the ODE is independent of the parameter. Although its shape does not change, its Floquet multipliers do (the ones related to the flow outside P_1). For positive but small λ the second Floquet multiplier (which is 1 for $\lambda = 0$) is smaller than one, so that the in-phase orbit is attractive on the torus (thus in the whole system). For negative but small λ the second multiplier is larger than one, and the invariant circle is repulsive on the torus, but still attractive from the outside.

The second orbit that survives is the *out-of-phase orbit* (see [1]) in P_2 where the ODE reduces to

$$\begin{aligned}\dot{x}_1 &= \beta x_2 + (1 - x_1^2 - x_2^2)x_1 - 2\lambda(x_1 + x_2), \\ \dot{x}_2 &= -\beta x_1 + (1 - x_1^2 - x_2^2)x_2 - 2\lambda(x_1 + x_2).\end{aligned}$$

For small $\lambda > 0$ its second Floquet multiplier (with respect to the four-dimensional system) is larger than 1. Hence, the out-of-phase orbit is repulsive on the torus. Any orbit that starts in its neighborhood will converge towards the in-phase orbit.

In the following we restrict the system to the plane P_2 . As λ tends to $\lambda_{\text{SN}} = \frac{\beta}{2} = 0.275$ the period of the out-of-phase orbit tends to infinity.

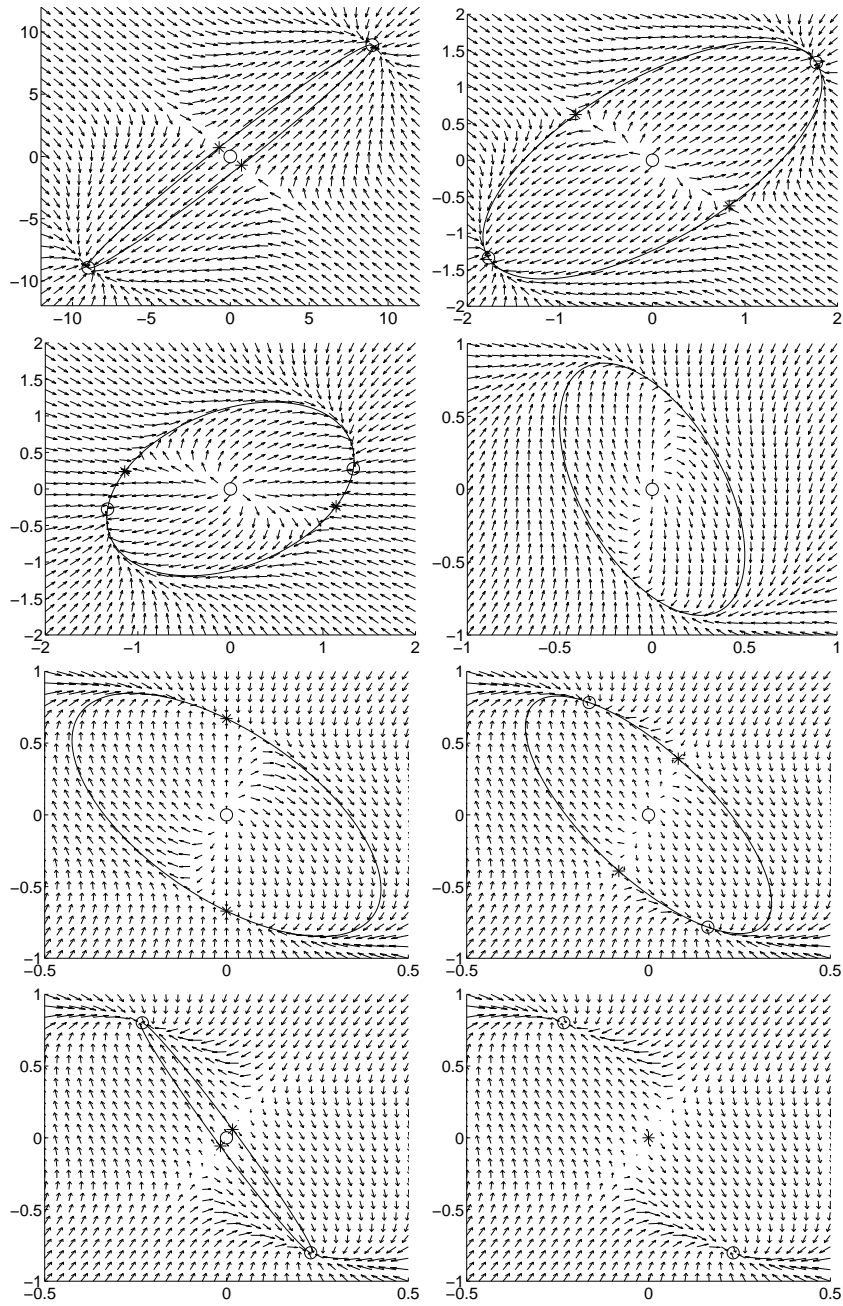


FIG. 4. The flow on P_2 for $\lambda = -40, -1, -0.3, 0.25, 0.275, 0.3, 0.325, 0.325625$ with its nodes ("o"), saddles ("*") and invariant circles.

Finally, at λ_{SN} two fixed points with the coordinates $(0, \pm\sqrt{1-\beta})$ appear. Instead of a periodic orbit the invariant circle now consists of two fixed points connected by two heteroclinic orbits. This is the point where algorithms that compute only one trajectory break down. In fact, there are two saddle-node bifurcations on the circle: Both fixed points split for $\lambda > \lambda_{\text{SN}}$ into a sink and a saddle which are connected by (in total four) heteroclinic orbits.

The dynamics on the circle is closely related to the dynamics in the neighborhood of the origin: For $\lambda \in (-\lambda_{\text{SN}}, \lambda_{\text{SN}})$ the origin is a rotating source (with respect to P_2), since both eigenvalues of $D(f_\lambda|_{P_2})(0)$ are complex conjugate with real part $1 - 2\lambda > 0$. When the fixed points appear on the invariant circle, also the rotation of the origin stops: $D(f_{\lambda_{\text{SN}}}|_{P_2})(0)$ has a double real eigenvalue. For λ slightly larger than λ_{SN} the origin is a source without rotation.

The invariant circle is finally destroyed in a pitch-fork bifurcation where the saddles collide with the origin at $\lambda_{\text{PF}} = \frac{1+\beta^2}{4} = 0.325625$. When λ tends to this value, the circle gets flattened until it degenerates to a curve consisting of two heteroclinic orbits that connect the origin with the sinks. For $\lambda > \lambda_{\text{PF}}$ those two orbits form the unstable manifold of the origin which is a saddle for those values of λ . For $\lambda \rightarrow \infty$ the sinks tend to $\pm(\sqrt{1/2}, -\sqrt{1/2})$. The origin stays a saddle so that there is probably no further qualitative change of the dynamics on P_2 .

For λ slightly smaller than 0 the second multiplier (with respect to the four-dimensional system) of the out-of-phase orbit is smaller than 1 and the orbit is attractive. At $-\lambda_{\text{SN}}$ two fixed points $(\pm\sqrt{1+\beta}, 0)$ appear. As in the case $\lambda > 0$ we have two saddle-node bifurcations and the origin stops rotating. But here the emerging four fixed points do not disappear for any λ . The invariant circle probably exists for all $\lambda < -\lambda_{\text{SN}}$ although growing larger, since the sinks tend to infinity for $\lambda \rightarrow -\infty$.

Examples of the flow in the plane P_2 for different parameter values as well as the invariant circle and its fixed points are shown in Figure 4. Figure 5 shows the invariant circles and fixed points in the plane P_2 for $\lambda \geq 0$. The fixed points form the s-shaped curve. The saddles lie on the dotted part and the sinks on dash-dotted part. Their limit for $\lambda \rightarrow \infty$ is marked with “+”. The locations of the two saddle-node bifurcations and the pitch-fork bifurcation are marked with “*”. The saddle-node bifurcations occur on the dashed circle ($\lambda = \lambda_{\text{SN}} = 0.275$). The outermost circle of radius 1 corresponds to $\lambda = 0$. For larger values of λ the circles get smaller. As an example, the sinks (“o”) and saddles (“x”) of the circle for $\lambda = 0.3$ are plotted here. Figure 6 shows the same scenario for $\lambda \leq 0$.

The calculations in the reduced system were done with 1000 grid points per circle (although less would have been sufficient, especially for the circles with $\lambda \geq -2.8$ displayed in Figure 5 and 6). For positive λ we can follow the circles almost to the bifurcation at $\lambda_{\text{PF}} = 0.325625$ where they degenerate to a curve (the flattest circle shown here corresponds to $\lambda = 0.325$). For

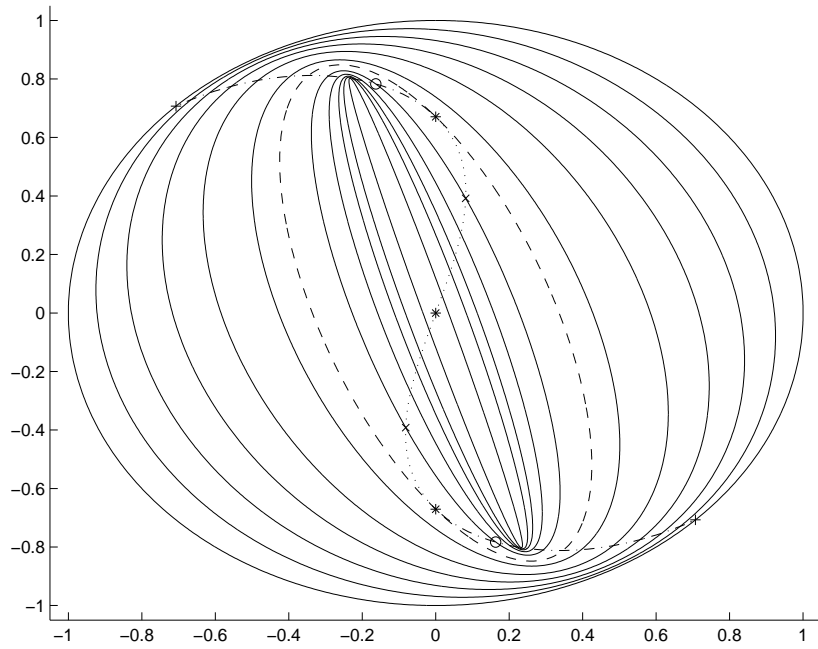


FIG. 5. The invariant circles and fixed points in the plane P_2 for $\lambda \geq 0$.

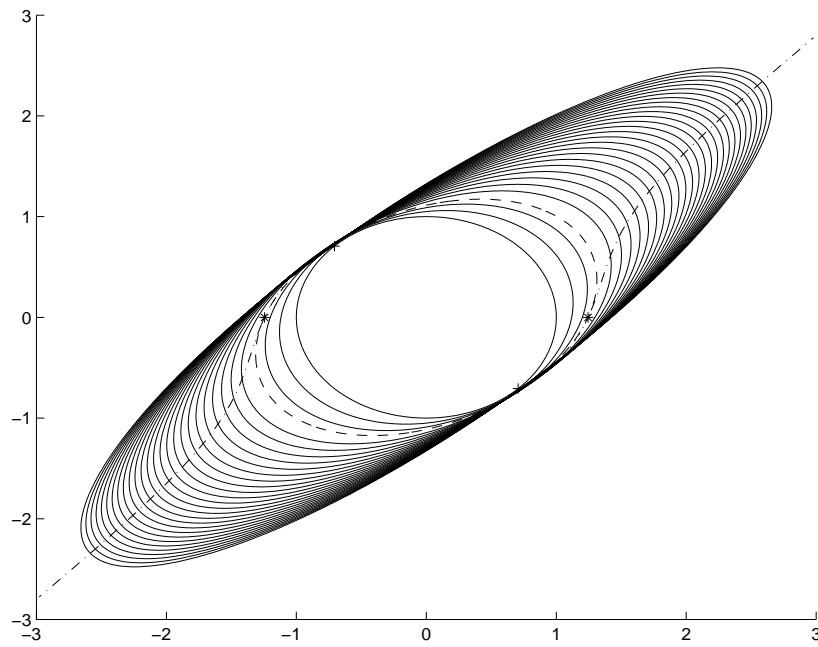


FIG. 6. The invariant circles and fixed points in the plane P_2 for $\lambda \leq 0$.

negative λ we can follow the circles beyond -40 . Here we have to reduce the time-step for the graph transform several times (from 1 to 0.01) to achieve convergence. This is caused by the acceleration of the dynamics: The eigenvalues of the sinks change from about -3.68 and -0.48 at $\lambda = -0.3$ to about -322 and -160 at $\lambda = -40$ for example.

5.1.2. Invariant torus. But now for our main goal, the invariant torus: As mentioned above it will exist for λ in a neighborhood of 0 because of Theorem 2.1. Let us first consider the case $\lambda > 0$:

The fact that there is no bifurcation in the reduced systems on the planes P_1 and P_2 between 0 and λ_{SN} could suggest that the invariant torus exists up to λ_{SN} . However, this is not the case (see [4]): Close to zero the first Floquet multiplier of the out-of-phase orbit is larger than one (because the orbit is repelling within the torus) and the next two are smaller than one (because the whole torus has two attractive transversal directions). The last multiplier is of course 1. If we increase λ beyond $\lambda_{\text{TO}} \approx 0.260524$, also the second multiplier becomes larger than 1 (see Figure 7). Thus, the orbit becomes repelling not only within the torus,

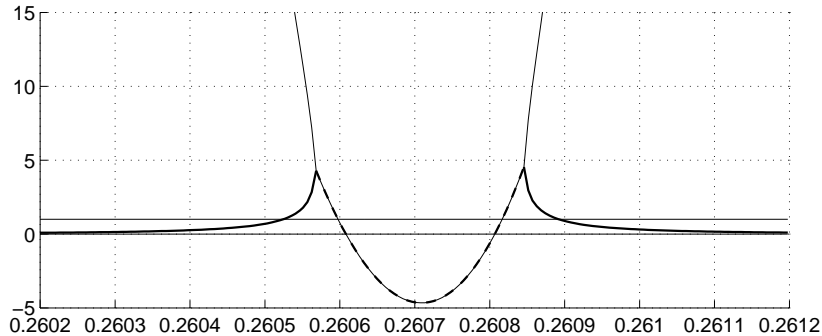


FIG. 7. The Floquet multipliers (computed with AUTO [5]) of the out-of-phase orbit. The dashed line represents the real part of the two complex conjugate multipliers.

but also in a direction transversal to the torus. Hence, the torus loses its attractivity. In a small interval — approximately $(0.26056, 0.26085)$ — there are two complex conjugate multipliers and beyond 0.2609 we have the same situation as before: one multiplier is smaller and two are larger than 1. Because the torus loses its attractivity at λ_{TO} , it cannot be computed any further. It seems to be destroyed completely, since the algorithm fails to find it back for λ larger than 0.2609.

Where the torus is destroyed for negative λ , if it is destroyed at all, is not clear. As the computations show, it survives at least the saddle-node bifurcation at $-\lambda_{\text{SN}}$ on the out-of-phase orbit and can be computed for quite some time. It finally breaks down around -0.565 . Some comments on this case can be found at the end of Section 5.1.3.

For the computations we started with the (analytically given) torus at $\lambda = 0$ and tried to follow the torus as far as possible in both directions. Figures 8 and 9 show the development of the torus and the circles from two positions. The visualization is somewhat difficult, since the dynamical system is four-dimensional. Because we omitted the last dimension in the plots, it seems as if the tori intersect themselves, but that is only an effect of the projection. In addition, some parts are hidden by others. Nevertheless, in order to show these parts we only display some 50 percent of the surface.

Finally, the flow on the torus is shown in Figure 10 for some selected values of λ . It is plotted modulo 1. On the main diagonal we see the in-phase orbit, on the sub- and superdiagonal we see the out-of-phase orbit (or the invariant circle which emerges from it). As a result of the symmetry of the system and the parametrization they are straight lines.

5.1.3. Performance. The performance² of the algorithms with varying parameters can be seen in Table 1. We start at $\lambda = 0$ with the initial step size $\Delta\lambda = 0.1$. If the graph transform does not converge, we halve the step size and try again starting from the last iteration that converged. If the step size decreases below 10^{-6} , we stop.

Already with a small grid of 20×20 points we get quite close to the bifurcation at λ_{TO} (within only 60 to 90 seconds). For just a quick glance at a problem this is the recommended choice. As expected, the computation time grows almost linearly with the number of grid points. With an increasing number of points the breakdown gets closer to the bifurcation. But due to the relatively high accuracy of the result on the coarse grid the improvement is only small. The fact that almost the same results are obtained for different grid sizes (and values of t and even different algorithms) suggests strongly the existence of a bifurcation close to $\lambda = 0.2605$ (which indeed is the case as seen before).

The better performance of the two steps algorithm compared to the shooting method is rather obvious. Even on a 100×100 grid it is faster than shooting on a 40×40 grid for most values of t .

The range for the parameter t where useful results are obtained grows — as expected — with increasing grid size. The strongly varying computation time is somewhat confusing at first sight and needs further explanation: Most of the computation time is spent for λ values close to the breakdown where the contraction rate of the graph transform is almost 1. Many iterations are needed for each λ step. Sometimes the contraction is so weak that the graph transform does not seem to converge. Consequently, $\Delta\lambda$ is decreased and two steps are performed instead of one, which slows down the algorithm considerably. On the other hand, choosing $\Delta\lambda$ too large, can also slow down the algorithm: If λ is increased beyond the point of breakdown, several steps with non-converging (and therefore slow) graph transforms

²The timings were made on an SGI workstation with a 200 MHz MIPS R4400 processor under IRIX 5.3. They are not very accurate due to other load on the system.

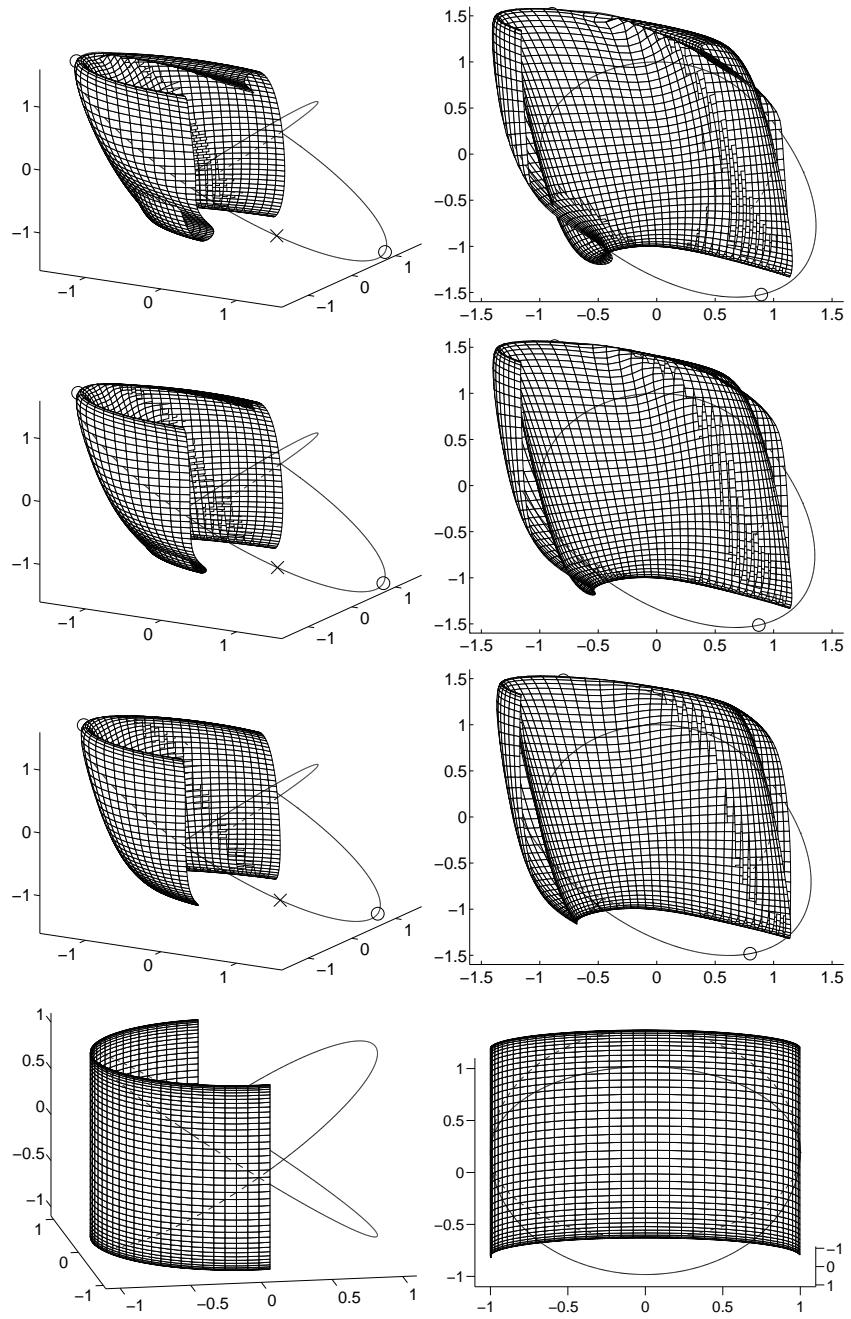


FIG. 8. Invariant torus for $\lambda = -0.565625, -0.55, -0.5$ and 0 (front and side view).

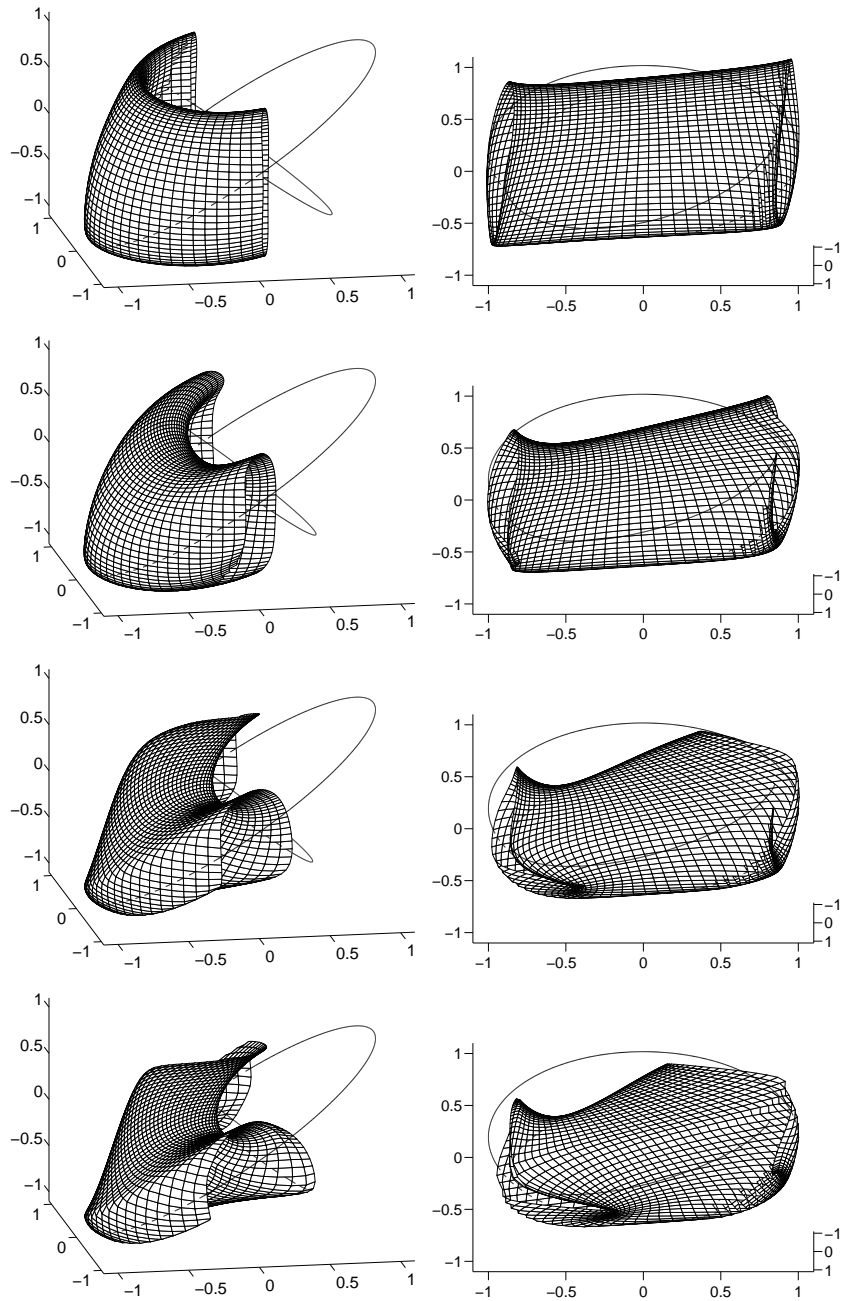


FIG. 9. *Invariant torus for $\lambda = 0.2, 0.25, 0.259375$ and 0.2605 (front and side view).*

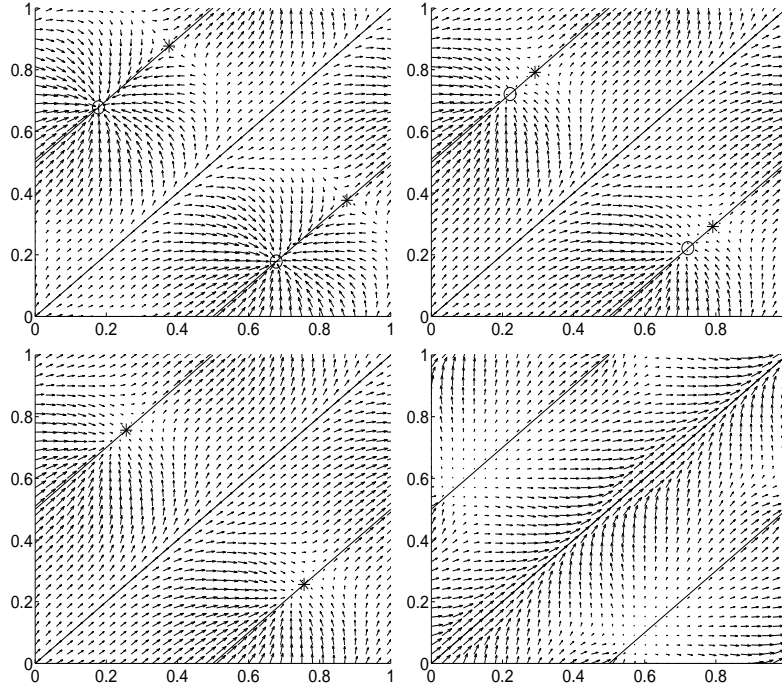


FIG. 10. Vector field on the torus for $\lambda = -0.565625, -0.3, -0.275$ and 0.2605 .

might be needed to decrease $\Delta\lambda$ enough to get back before this point. The selection of λ steps does not depend continuously on t and neither does the computation time. To improve performance, it would be important to analyze how t should be chosen. In addition a more sophisticated strategy for the choice of $\Delta\lambda$ might be helpful.

In a second performance test (see Table 2) we choose a fixed $\lambda_{\text{end}} = 0.25$. It is far enough away from the breakdown so that the algorithms will not have to decrease $\Delta\lambda$ for most values of t to achieve convergence of the graph transform. This allows us to compare the performance for each graph transform. To estimate the error of the computation we calculate the maximum distance of the grid points from an “exact” torus computed on a much finer grid (400×400).

The computations are about 20 times faster than the previous ones where most of the time is spent to find the point of breakdown with high accuracy. Here only three λ steps are necessary. Again, the increase of the computation time with the grid size is almost linear. However, the time needed for solving the PDE in the reparametrization step grows super-linearly with the grid size. This is responsible for the slight slow-down on the 100×100 grid. In the second test the performance of the shooting algorithm is better than in the previous one. Far away from the breakdown

TABLE 1
Performance of the algorithms close to the bifurcation at $\lambda_{\text{TO}} \approx 0.260524$.

Method	Grid	t	Breakdown	Time
Two steps	20×20	0.5	0.260595	94
Two steps	20×20	0.75	0.260937	67
Two steps	20×20	1	0.261193	78
Two steps	20×20	1.5	0.261035	85
Two steps	20×20	2	0.261132	80
Two steps	20×20	2.5	0.260754	42
Two steps	20×20	3	0.258056	34
Two steps	20×20	4	0.250000	15
Two steps	40×40	0.5	0.260437	325
Two steps	40×40	0.75	0.260546	284
Two steps	40×40	1	0.260546	248
Two steps	40×40	1.5	0.260412	389
Two steps	40×40	2	0.260403	390
Two steps	40×40	2.5	0.260571	107
Two steps	40×40	3	0.260269	108
Two steps	40×40	4	0.258035	103
Shooting	40×40	0.5	0.260412	1488
Shooting	40×40	0.75	0.260485	1689
Shooting	40×40	1	0.260464	2070
Shooting	40×40	1.5	0.260406	5517
Shooting	40×40	2	0.260449	3595
Shooting	40×40	2.5	0.260559	2002
Shooting	40×40	3	0.260504	2535
Shooting	40×40	4	0.260156	493
Two steps	100×100	0.5	0.260565	1974
Two steps	100×100	0.75	0.260571	1657
Two steps	100×100	1	0.260552	1778
Two steps	100×100	1.5	0.260543	2066
Two steps	100×100	2	0.260505	1262
Two steps	100×100	2.5	0.260510	1276
Two steps	100×100	3	0.260430	1275
Two steps	100×100	4	0.253393	4490

the manifold is still smooth, therefore, only a few iterations of Newton's method in each graph transform are necessary to satisfy the orthogonality condition (3.3). The advantage of the two steps method is that the iterations of Newton's method are much less expensive than in the shooting algorithm. Since the first step of the two steps method is the most expensive one, the gap between both strategies widens with the increase of the number of iterations. In other words: The worse the grid, the faster the two steps method.

The accuracy of the results improves with the number of grid points as expected. In general, the shooting algorithm achieves slightly smaller errors than the two steps method. But, if we slightly refine the grid, the two steps method is more accurate and still faster than the shooting algorithm. The range for t , where we get fairly accurate results is smaller for the two steps method.

TABLE 2
Performance and accuracy of the algorithms with fixed $\lambda_{\text{end}} = 0.25$.

Method	Grid	t	Error	Time
Two steps	20×20	0.5	$9.37 \cdot 10^{-3}$	5.05
Two steps	20×20	0.75	$8.10 \cdot 10^{-3}$	3.84
Two steps	20×20	1	$5.28 \cdot 10^{-3}$	3.43
Two steps	20×20	1.5	$9.83 \cdot 10^{-3}$	3.27
Two steps	20×20	2	$23.95 \cdot 10^{-3}$	3.18
Two steps	20×20	2.5	$56.34 \cdot 10^{-3}$	3.75
Two steps	40×40	0.5	$1.15 \cdot 10^{-3}$	18.29
Two steps	40×40	0.75	$0.70 \cdot 10^{-3}$	14.15
Two steps	40×40	1	$0.62 \cdot 10^{-3}$	13.19
Two steps	40×40	1.5	$2.33 \cdot 10^{-3}$	12.99
Two steps	40×40	2	$5.02 \cdot 10^{-3}$	13.68
Two steps	40×40	2.5	$7.95 \cdot 10^{-3}$	14.63
Two steps	40×40	3	$11.78 \cdot 10^{-3}$	15.77
Shooting	40×40	0.5	$1.69 \cdot 10^{-3}$	70.14
Shooting	40×40	0.75	$1.64 \cdot 10^{-3}$	70.35
Shooting	40×40	1	$1.35 \cdot 10^{-3}$	70.95
Shooting	40×40	1.5	$0.85 \cdot 10^{-3}$	80.06
Shooting	40×40	2	$0.52 \cdot 10^{-3}$	87.16
Shooting	40×40	2.5	$0.98 \cdot 10^{-3}$	91.26
Shooting	40×40	3	$0.67 \cdot 10^{-3}$	106.00
Two steps	100×100	0.5	$0.05 \cdot 10^{-3}$	138.74
Two steps	100×100	0.75	$0.09 \cdot 10^{-3}$	117.45
Two steps	100×100	1	$0.06 \cdot 10^{-3}$	109.88
Two steps	100×100	1.5	$0.20 \cdot 10^{-3}$	109.56
Two steps	100×100	2	$0.39 \cdot 10^{-3}$	117.07
Two steps	100×100	2.5	$0.53 \cdot 10^{-3}$	126.19
Two steps	100×100	3	$0.75 \cdot 10^{-3}$	136.70

The results for negative λ (see Table 3) are not that satisfactory. The point of breakdown varies strongly with the chosen algorithm, the grid size and the parameter t . Plots of the situation show that the difficulties which lead to the breakdown are located in the neighborhood of the sinks in the plane P_2 . Here the torus seems to lose its smoothness. The fixed points themselves are hyperbolic and hence not responsible for the breakdown.

Maybe it is a global phenomenon in the sense that the torus does not lose its hyperbolicity at a fixed point or a periodic orbit as in the examples before, but that the whole torus loses its smoothness. This would explain why a definite point of breakdown seems to be missing: The interpolating spline through a finite number of points on a “rough” manifold is very sensitive with respect to the choice of the grid. For example, if we take our grid points in the “valleys”, the interpolating spline might be almost flat, if we chose them on the “hills” as well as in the “valleys”, the spline will be rough. In the first case the algorithm might continue for some iterations (seeing only the smooth spline), whereas it will break down in the second. However, we cannot rule out the possibility that the breakdown is only an effect of numerical problems.

TABLE 3
Results for negative λ .

Method	Grid	t	Breakdown	Time
Two steps	20×20	0.1	-0.542163	1223
Two steps	20×20	0.2	-0.553036	701
Two steps	20×20	0.3	-0.559330	743
Two steps	20×20	0.5	-0.562109	41
Two steps	20×20	0.75	-0.562500	15
Two steps	20×20	1.0	-0.550000	11
Two steps	40×40	0.1	-0.564770	2706
Two steps	40×40	0.2	-0.568019	220
Two steps	40×40	0.3	-0.562500	54
Two steps	40×40	0.5	-0.562500	42
Two steps	40×40	0.75	-0.561718	84
Two steps	40×40	1.0	-0.559375	55
Shooting	40×40	0.1	-0.565393	11045
Shooting	40×40	0.2	-0.568750	518
Shooting	40×40	0.3	-0.566015	397
Shooting	40×40	0.5	-0.562500	212
Shooting	40×40	0.75	-0.561718	357
Shooting	40×40	1.0	-0.559375	325
Two steps	100×100	0.1	-0.566406	1453
Two steps	100×100	0.2	-0.565722	800
Two steps	100×100	0.3	-0.567968	804
Two steps	100×100	0.5	-0.559375	503
Two steps	100×100	0.75	-0.555126	671
Two steps	100×100	1.0	-0.556250	728
Shooting	100×100	0.1	-0.565820	3248
Shooting	100×100	0.2	-0.564062	1832
Shooting	100×100	0.3	-0.565722	1968
Shooting	100×100	0.5	-0.562500	1943
Shooting	100×100	0.75	-0.556250	2277
Shooting	100×100	1.0	-0.556268	2917

5.2. Forced van der Pol oscillator. Although it is originally a second-order ODE, we write the forced van der Pol oscillator as first order system (see [7, p. 67ff]):

$$(5.1) \quad \begin{aligned} \dot{x}_1 &= x_2 + \alpha x_1 \left(1 - \frac{x_1^2}{3}\right), \\ \dot{x}_2 &= -x_1 + \lambda \cos(\omega t) \end{aligned}$$

with fixed parameters $\alpha > 0$, $\omega > 0$ and the continuation parameter λ that controls the periodic forcing term. Without forcing ($\lambda = 0$) we have an oscillator with an attractive periodic orbit. For $\lambda \neq 0$ the ODE is non-autonomous. By adding a third equation $\dot{t} = 1$ for the time variable we can change it into an autonomous system. By identifying t with $t + \frac{2\pi}{\omega}$ (so that $\cos(\omega t)$ is still well-defined) this equation becomes periodic in an abstract fashion. The trivial oscillator corresponding to $\dot{t} = 1$ satisfies the gap condition, because its normal bundle is zero-dimensional (which implies $\theta = -\infty$ and $\rho = 0$). As in the previous example we now have two

independent oscillators for $\lambda = 0$. Again, their cross product has a torus as invariant manifold that will persist under small perturbations of λ .

Enabling the algorithms to deal with the abstract periodicity would require some reprogramming. Thus, for computational purposes, it is easier to consider the following system:

$$\begin{aligned}\dot{x}_1 &= x_2 + \alpha x_1 \left(1 - \frac{x_1^2}{3}\right), \\ \dot{x}_2 &= -x_1 + \lambda x_3, \\ \dot{x}_3 &= \omega x_4 + (1 - x_3^2 - x_4^2)x_3, \\ \dot{x}_4 &= -\omega x_3 + (1 - x_3^2 - x_4^2)x_4\end{aligned}$$

where a two-dimensional oscillator provides the forcing term. This system is similar to the coupled oscillators with the main difference that here the second oscillator remains unperturbed for $\lambda \neq 0$. We also have less symmetry, which has two consequences: First, this example shows that the numerical results achieved by the algorithms do not depend on the symmetry of a system. Second, it is more difficult to obtain analytical data for the comparison with the numerical data. We circumvent this problem in two ways: For a qualitative analysis we consider the *averaged* system that is reduced by one dimension, but also slightly perturbed. For a quantitative analysis we exploit the fact that the bifurcations of the torus coincide with the ones of the periodic orbits lying on the torus. Computing the latter with fairly high accuracy (using AUTO) we can take these data as a basis for the comparison.

The averaged system (for details see [7, p. 70]) has the following form:

$$(5.2) \quad \begin{aligned}\dot{u} &= \sigma v + (1 - u^2 - v^2)u, \\ \dot{v} &= -\sigma u + (1 - u^2 - v^2)v - \gamma\end{aligned}$$

with $\sigma = \frac{1-\omega^2}{\alpha\omega}$ and $\gamma = \frac{\lambda}{2\alpha\omega}$. Its construction is based on the following idea: The invariant torus of the forced oscillator (5.1) can be regarded as a circle which slowly changes its shape with increasing time. By averaging the vector field over one period the motion of the circle is suppressed leading to an autonomous system. Unfortunately, this process slightly changes the dynamics so that the resulting system can only be used for qualitative analysis. It is sufficient to investigate the circle, and we can reduce the three-dimensional autonomous system by one dimension.

The diagrams in Figure 11 show the saddle-node bifurcations (curves *a* and *b*) and the Hopf bifurcations (curve *c*) of the averaged system. Let us first consider the region for $0 < \sigma < 1/2$: Starting at $\gamma = 0$ the oscillator (5.2) has an attractive periodic orbit and a source in its center. If we increase γ , the period of the orbit also increases. As we hit the first curve *a*, coming from below, a saddle-node bifurcation occurs: A saddle appears on the invariant circle which splits into a sink and a saddle for larger γ . As we continue, the distance between the saddle on the circle and the source in

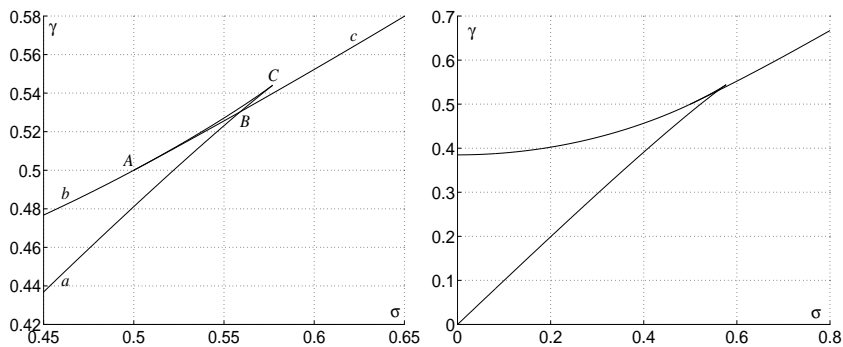


FIG. 11. Bifurcation diagram for the averaged system.

the circle becomes smaller. When γ reaches the second curve b the source hits the saddle, giving another saddle-node bifurcation. The circle loses its attractivity and vanishes for larger γ ; the saddle also vanishes so that only the sink survives.

For $\sigma > 1/\sqrt{3}$ we have only one bifurcation: Again, we start at $\gamma = 0$ with an attractive periodic orbit and a source in its center. As γ increases the invariant circle gets smaller. At the curve c the circle is destroyed in a Hopf bifurcation which turns the source into a sink.

The situation for $\sigma \in [1/2, 1/\sqrt{3}]$ is more complex: In the neighborhood of B two of the fixed points lie outside the invariant circle, but in the neighborhood of the origin they lie on the circle. A bifurcation curve, let us denote it by d , separates those situations. It is located below c and connects the point A with the curve a . On d one of the fixed points (the saddle) lies on the invariant circle producing a homoclinic orbit. Here the circle is not C^1 anymore.

If we decrease γ , we observe that the heteroclinic orbit that connected the saddle and the sink splits up and unites with the homoclinic orbit, which simultaneously opens. In this way also the second fixed point is included in the invariant circle, which now consists of two heteroclinic orbits. This looks almost like the situation in the origin with one major difference: The circle is still not C^1 , because both heteroclinic orbits meet the sink from the same direction and not from the opposite ones. Therefore, there is another bifurcation curve e below d where one heteroclinic orbit flips from the center-stable direction via the strong-stable direction to the opposite center-stable direction of the sink. Talking in terms of topological equiva-

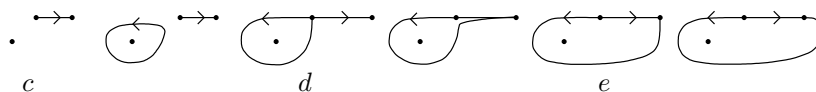


FIG. 12. Sketch of the bifurcations occurring for $\sigma \in (1/2, 1/\sqrt{3})$.

lence this is not a real bifurcation, but talking about C^1 -manifolds this is the point where they are born or destroyed. Since it is not easy to locate those two curves precisely, we omitted them in the diagram. A sketch of those bifurcations is given in Figure 12.

To interpret the results in terms of system (5.1), we only have to add one dimension, a “circle” becomes a “torus” and a “fixed point” becomes a “periodic orbit”. Of course, the bifurcation diagram changes somewhat because of the errors introduced by the averaging. Using AUTO to follow the periodic orbits and their Floquet multipliers we can calculate the new diagram shown in Figure 13 for $\alpha = 0.4$ (in terms of ω and λ instead of σ and γ). It is also shown how close the algorithms are able to get to the bifurcation where the torus is destroyed. For several values of ω we started with $\lambda = 0$ and tried to follow the tori as far as possible (using a 40×40 grid).

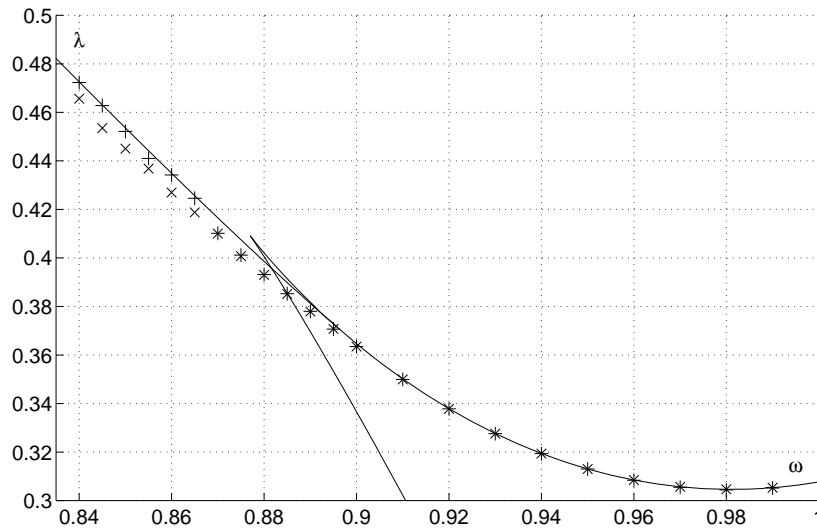


FIG. 13. Performance of the algorithms for the forced van der Pol oscillator.

In the right section (where the torus is destroyed in a “saddle-node” bifurcation) we observe that the two steps algorithm (“x”) and the floating algorithm (“+”) perform equally well. In the left section we see that the floating algorithm performs much better, getting almost up to the Hopf bifurcation. Here the torus changes its shape very quickly when the parameter λ changes, a situation for which the more robust method was designed.

For ω between 0.87 and 0.895 both algorithms do not (seem to) perform very well. This has two reasons: First, in Figure 13 we do not see the bifurcation curve of the flip bifurcation in the right part of this interval where the torus is destroyed as a C^1 manifold. Since this bifurcation curve is located below c the actual error is smaller than the picture suggests.

Second, the algorithms suffer from numerical problems: Although the torus is still smooth from the analytical point of view, the normal directions of two adjacent grid points change about 90° . This is a sign that the resolution of the grid is still not high enough. Maybe adaptive grid refinement could help in this situation, but that has not been implemented yet.

The Figures 14 and 15 show invariant tori for three scenarios leading to a flip bifurcation ($\omega = 0.885$), a saddle-node bifurcation ($\omega = 0.91$) and a Hopf bifurcation ($\omega = 0.86$). They were computed on a 100×100 grid, although they are plotted with a smaller resolution. The time variable is normalized in the pictures such that all tori are plotted modulo 2π . For $\omega = 0.885$ and $\omega = 0.91$ the invariant circles on and inside the torus are also shown. To be able to look inside the torus in the second scenario we cut away about 25% of the torus.

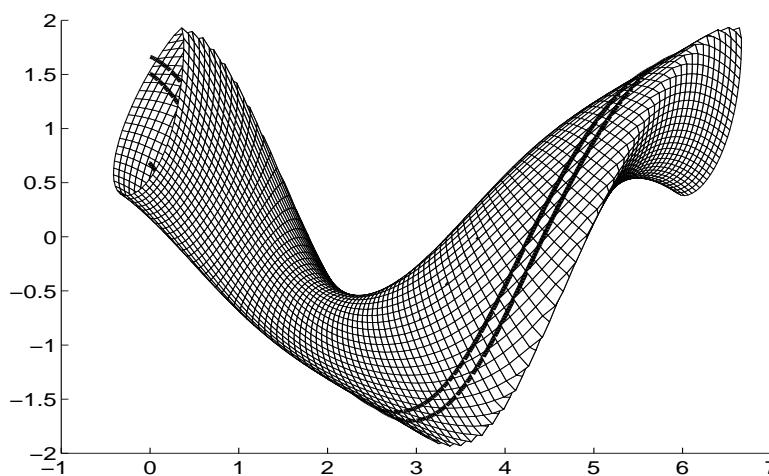


FIG. 14. *Invariant torus for $\alpha = 0.4$, $\omega = 0.885$ and $\lambda = 0.3858$ close to the loss of smoothness.*

In the first scenario the numerical difficulties before the torus loses its smoothness in the flip bifurcation can be seen. Although the torus is still smooth from the analytical point of view, it looks as if a cusp is developing, which causes the algorithms to break down. The torus is actually destroyed when the “cusp” meets the attracting period orbit. The first picture of the second scenario shows the torus shortly after the first saddle-node bifurcation, the two invariant orbits on the torus are lying close together. In the last picture their distance has become larger, but now we are very close to the second saddle-node bifurcation, where the invariant circle from the inside meets one on the torus. The third scenario with the Hopf bifurcation needs no further explanation, just a remark: We can actually follow the torus closer to the bifurcation than displayed here, but in this way one can at least see a torus.

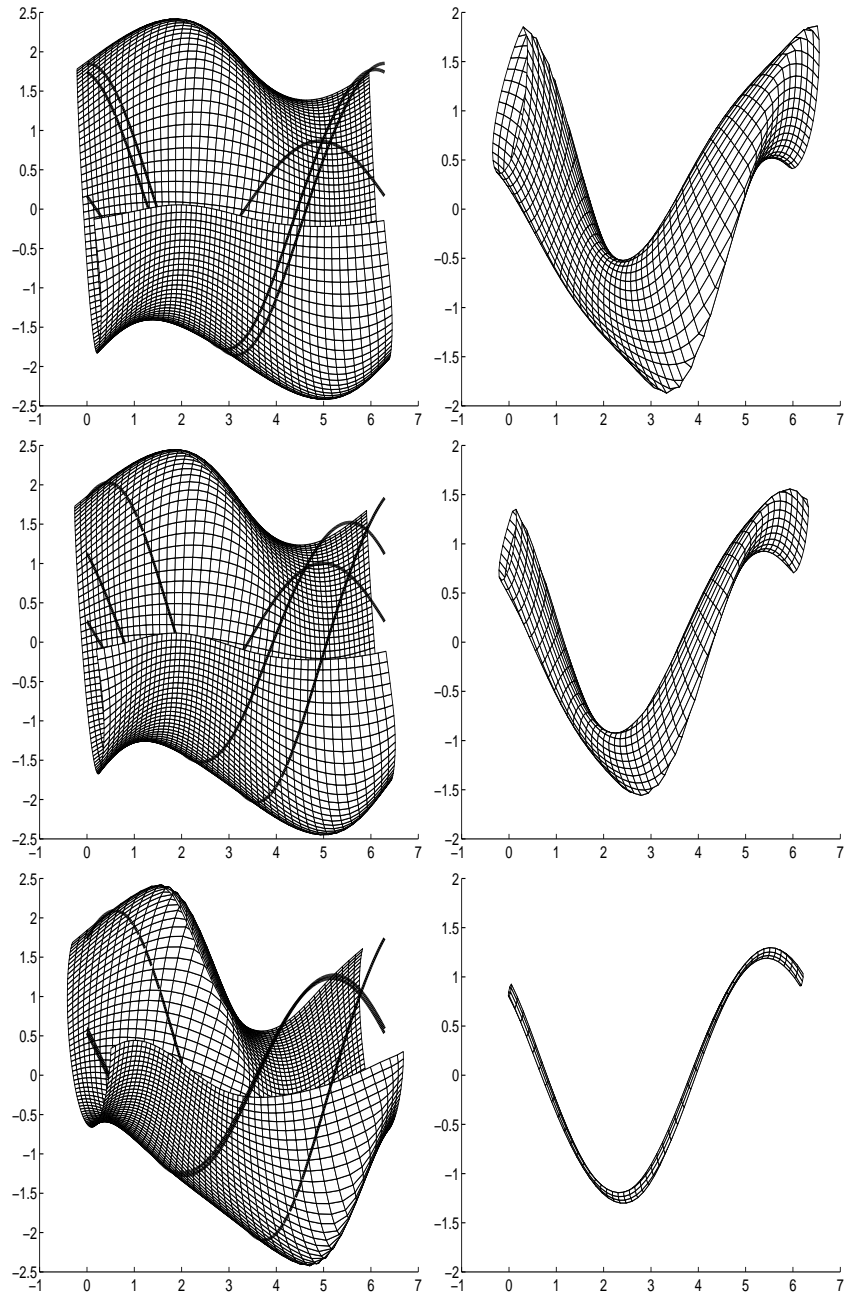


FIG. 15. *On the left: Invariant torus for $\alpha = 0.4$, $\omega = 0.91$ and $\lambda = 0.303, 0.33, 0.350027$ close to the saddle-node bifurcation. On the right: Invariant torus for $\alpha = 0.4$, $\omega = 0.86$ and $\lambda = 0.425, 0.43, 0.434636$ close to the Hopf bifurcation.*

Acknowledgments. I want to thank J. Lorenz for his support of my work during several years. Thanks to H. M. Osinga, B. Krauskopf and E. Doedel for some helpful discussions. And last, but not least, I want to thank the referee for several suggestions to improve this article.

REFERENCES

- [1] D. G. ARONSON, E. J. DOEDEL AND H. G. OTHMER, *An analytical and numerical study of the bifurcations in a system of linearly-coupled oscillators*, *Physica D*, **25** (1987), pp. 20-104.
- [2] H. W. BROER, H. M. OSINGA AND G. VEGTER, *Algorithms for computing normally hyperbolic invariant manifolds*, *Z. Angew. Math. Phys.*, **48**(3) (1997), pp. 480-524.
- [3] L. DIECI AND J. LORENZ, *Computation of invariant tori by the method of characteristics*, *SIAM J. Numer. Anal.*, **32**(5) (1995), pp. 1436-1474.
- [4] L. DIECI AND J. LORENZ, *Lyapunov-type numbers and torus breakdown: numerical aspects and a case study*, *J. Numer. Algorithms*, **14** (1997), pp. 79-102.
- [5] E. J. DOEDEL AND X. J. WANG, *AUTO94: Software for continuation and bifurcation problems in ordinary differential equations*, Technical report, CRPC-95-2, Center for Research on Parallel Computing, California Institute of Technology, Pasadena CA 91125, 1995.
- [6] N. FENICHEL, *Persistence and Smoothness of Invariant Manifolds for Flows*, *Indiana University Mathematics Journal*, **21**(3) (1971), pp. 193-226.
- [7] J. GUCKENHEIMER AND PH. HOLMES, *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, Springer Series in Applied Mathematical Sciences 42, Springer-Verlag, 1983.
- [8] E. HAIRER, S. P. NØRSETT AND G. WANNER, *Solving Ordinary Differential Equations I, Nonstiff Problems, Second revised edition*, Springer Series in Computational Mathematics 8, Springer-Verlag, 1993.
- [9] G. MOORE, *Computation and parametrisation of invariant curves and tori*, *SIAM J. Numer. Anal.*, **33**(6) (1996), pp. 2333-2358.