POMULT: A Program for Computing Periodic Orbits in Hamiltonian Systems Based on Multiple Shooting Algorithms

Stavros C. Farantos
Institute of Electronic Structure and Laser,
Foundation for Research and Technology - Hellas, and
Department of Chemistry
University of Crete,
Iraklion, Crete 711 10, Greece.

Abstract

POMULT is a FORTRAN code for locating Periodic Orbits and Equilibrium Points in Hamiltonian systems based on 2-point boundary value solvers which use multiple shooting algorithms. The code has mainly been developed for locating periodic orbits in molecular Hamiltonian systems with many degrees of freedom and it utilizes a damped Newton-Raphson method and a secant method. Graphical User Interface has also been written in tcl-tk script language for interactively manipulating the input and output data.

Keywords: molecular dynamics, periodic orbits, multiple shooting algorithm, damped Newton-Raphson method

PROGRAM SUMMARY

Title of the program: POMULT (Periodic Orbit MULTishooting)

Catalogue number: POMT

Program obtainable form: CPC Program Library, Queen's University of Belfast, N. Ireland

Licensing provisions: none

Computer: Tested on workstations HP-9000/735, IBM-7030/3CT

Installation: IESL-FORTH, Iraklion, Crete, Greece

Operating system: UNIX

Programming language used: FORTRAN 77 with extensions, lower case, implicit, include

Memory required to execute with typical data: 5 Mbytes

No. of bits in a word: 32

No. of bytes in distributed program, included test data etc: 4014080

Distribution format: gzip compressed tar file

Keywords: molecular dynamics, periodic orbits, multiple shooting algorithm, damped Newton-Raphson method

Nature of the physical problem:

Given a multidimensional highly coupled molecular potential energy surface we want to compute families of periodic solutions of Hamilton equations. These families of periodic orbits reveal the structure of the classical phase space by detecting the regions of phase space with regular and chaotic motions. Furthermore, periodic orbits point out possible localization of the quantum wavefunctions, and explain/predict spectroscopic features.

Method of solution:

The location of periodic orbits is based on damped Newton-Raphson methods or secant-Quasi Newton methods. Simple or Multiple shooting algorithms are employed which are robust in cases of long period or highly unstable periodic orbits.

Restrictions on the complexity of the problem:

The program has been tested with 2-, 3-, 5-, and 6-dimensional molecular potential functions. Limitations are observed in cases of high instability or in regions of phase space densely occupied by periodic orbits. The above difficulties cause also limitations in the continuation of a family of periodic orbits with a parameter.

Typical running time:

This depends on the complexity of the potential function and the number of periodic orbits asked.

Remarks:

Standard numerical actions like integration of ordinary differential equations and solution of linear algebraic equations are carried out with routines from the package "Numerical Recipes". The program can be interfaced with ODESSA or other available programs which carry out sensitivity analysis of differential or algebraic equations. Graphical User Interface for providing the input data has been written in tcl-tk script language. The user should ensure that the versions tcl7.0 and tk4.0 or higher are installed in his system.

LONG WRITE-UP

I. Introduction

The application of non-linear mechanics into molecular systems and particularly the study of vibrational spectroscopic resonances with periodic orbits has increased the interest of having robust algorithms to locate periodic orbits (POs) in Hamiltonian systems.[1] The location of POs sometimes may be a difficult task, since the molecular potential energy surfaces are multiple dimensional functions and the strong coupling among the degrees of freedom introduces large instabilities.[2]

The problem of finding periodic orbits may be seen as a *2-point boundary value problem*. The boundary conditions are the relations of closing the trajectory in phase space after the period of time T;

$$\vec{q}(T) = \vec{q}(0), \qquad \vec{p}(T) = \vec{p}(0). \tag{1}$$

$\vec{q} = (q_1, q_2, ..., q_N)^+$ are the generalized coordinates of a dynamical system of N degrees of freedom, and $\vec{p} = (p_1, p_2, ..., p_N)^+$ their conjugate momenta. + denotes a column vector.

There are two classes of numerical methods for solving in general 2-point boundary value problems.[3] The first includes the *shooting methods* in which the 2-point boundary value problem is converted to an *initial value* one. Choosing an initial value for the trajectory we integrate the equations of motion and check the discrepancy in the boundary conditions. By varying the initial conditions or some free parameters we successively approach the trajectory which satisfies the boundary conditions.

The second class includes the *relaxation methods*. In these methods the differential equations are replaced with difference equations by choosing an appropriate mesh of points for the variables. Then, starting with an approximate solution we try to bring it into successively closer agreement with the finite difference equations, and with the boundary conditions.

Both techniques have been applied to locate periodic orbits in molecular systems or model potentials. The shooting methods are the most popular.[4] The *Monodromy Method* of Baranger and coworkers[5–7] is a technique which is classified in the relaxation methods.

An extension of the shooting techniques which tries to incorporate the benefits of the relaxation technique is the *multiple shooting method*.[8–13] In this case the 1-point initial value problem is converted to $(m-1)$ initial value problems by choosing m nodes along the period of time. We do not take a finite difference representation of the equations of motion, but instead, we integrate $(m-1)$ trajectories and by varying their $(m-1)$ initial conditions we approach to a smooth trajectory which satisfies the closure conditions (Eqs (1)).

The program presented here, POMULT, is an implementation of the multiple shooting algorithm which uses global convergence techniques in the Newton-Raphson iterations. The package is also enriched with tools for calculating Power Spectra with Fast Fourier Transform (FFT), Maximum Lyapunov Exponents[14](MLE), the Classical Correlation Function[15] (CCF), and the construction of Poincaré Surfaces of Section[14] (PSS) for the exploration of the geometry of phase space of a dynamical system. Graphical User Interface is provided for an easy way to alter the input data.

II. Computational methods

For the simplification of the mathematical equations we define the column vector

$$\vec{x} = (\vec{q}, \vec{p})^{+}. \tag{2}$$

Using $\vec{x}$ we can write Hamilton equations in the form

$$\frac{d\vec{x}(t)}{dt} = \dot{\vec{x}}(t) = J\nabla H[\vec{x}(t)] \quad (0 \leq t \leq T), \tag{3}$$

where H is the Hamiltonian function, and J is the symplectic matrix

$$J = \begin{pmatrix} 0_N & I_N \\ -I_N & 0_N \end{pmatrix}. \tag{4}$$

0_N and I_N are the zero and unit $N \times N$ matrices respectively. $J\nabla H(\vec{x})$ is a vector field, and J satisfies the relations,

$$J^{-1} = -J \quad \text{and} \quad J^2 = -I_{2N}. \tag{5}$$

To find periodic solutions it is necessary to solve Eqs (3) subject to the 2-point boundary conditions,

$$\vec{x}(T) - \vec{x}(0) = 0. \tag{6}$$

The above boundary value problem is converted to an *initial value problem* by considering the initial values of the coordinates and momenta, $\vec{s}$,

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

as independent variables in the nonlinear functions

$$\vec{B}(\vec{s}; T) = \vec{x}(T; \vec{s}) - \vec{s}. \tag{8}$$

We denote the roots of Eqs (8) as $\vec{s}_*$, i.e.,

$$\vec{B}(\vec{s}_*; T) = 0. \tag{9}$$

Hence, if $\vec{s}$ is a nearby value to the solution $\vec{s}_*$ we can compute the functions $\vec{B}(\vec{s})$ by integrating Hamilton equations for the period T. By appropriately modifying the initial values $\vec{s}$ we hope to converge to the solution, that is, $\vec{s} \rightarrow \vec{s}_*$ and $B \rightarrow 0$.

The common procedure to find the roots of Eqs (8) is the Newton-Raphson method. This is an iterative scheme and at each iteration, k, we update the initial conditions of the orbit

$$\vec{s}_{k+1} = \vec{s}_k + \Delta\vec{s}_k. \tag{10}$$

The corrections $\Delta\vec{s}_k$ are obtained by expanding Eqs (8) in a Taylor series up to the first order

$$\begin{aligned} \vec{B}(\vec{s}_{k+1};T) &\approx \vec{B}(\vec{s}_k;T) + \frac{\partial \vec{B}}{\partial \vec{s}_k}\Delta\vec{s}_k = 0, \\ \vec{B}(\vec{s}_k;T) &+ \left[\frac{\partial \vec{x}_k(T;\vec{s}_k)}{\partial \vec{s}_k} - I_{2N}\right]\Delta\vec{s}_k = 0, \end{aligned} \tag{11}$$

where at the k^{th} iteration

$$\vec{B}(\vec{s}_k;T) = \vec{x}_k(T;\vec{s}_k) - \vec{s}_k. \tag{12}$$

The matrix (Jacobian)

$$Z_k(T) = \frac{\partial \vec{x}_k(T;\vec{s}_k)}{\partial \vec{s}_k}, \tag{13}$$

is the *Fundamental Matrix*, which is evaluated by integrating the *variational equations*

$$\dot{\vec{\zeta}}(t) = A(t)\vec{\zeta}(t) \quad (0 \leq t \leq T), \tag{14}$$

where the second derivatives of the Hamiltonian with respect to coordinates and momenta are needed

$$A(t) = J\partial^2 H[\vec{x}(t)]. \tag{15}$$

These equations calculate the linearized part of the difference of two initially neighboring trajectories in time, $\vec{\zeta} = \vec{x}' - \vec{x}$. The Fundamental Matrix is also a solution of the variational equations

$$\dot{Z}(t) = A(t)Z(t). \tag{16}$$

Thus, to perform the k^{th} iteration in the Newton-Raphson method we first integrate for time T the differential equations

$$\begin{aligned} \dot{\vec{x}}_k(t) &= J\nabla H[\vec{x}_k(t)] \\ \dot{Z}_k(t) &= A_k(t)Z_k(t), \end{aligned} \tag{17}$$

with initial conditions

$$\begin{aligned} \vec{x}_k(0) &= \vec{s}_k \\ Z_k(0) &= I_{2N}. \end{aligned} \tag{18}$$

Then, we solve the linear algebraic equations

$$[Z_k(T) - I_{2N}]\Delta\vec{s}_k = -\vec{B}(\vec{s}_k; T), \tag{19}$$

in order to find the initial conditions for the $(k+1)^{th}$ iteration (Eqs (10)).

For a periodic orbit the Fundamental Matrix at the time $t = T$ is called *Monodromy Matrix*, $M = Z(T)$. The eigenvalues of the Monodromy Matrix are used for the stability analysis of the periodic orbit. Details can be found in ref..[16]

A. The multiple shooting method

Let us assume that we divide the period T in $(m-1)$ time intervals by choosing m nodes. The idea of the multiple shooting method is to integrate $m-1$ trajectories and to check if the final values of each trajectory coincide with the initial conditions of the next. To formulate this idea we introduce the scaled time, $\tau = t/T, \quad (0 \leq \tau \leq 1)$,

$$0 = \tau_1 < \tau_2 < \cdots < \tau_{m-1} < \tau_m = 1. \tag{20}$$

Thus, for the simple shooting method $m = 2$.

In this section we drop the index for the iterations k, and we use the index j to denote the nodes in the periodic orbit. If the initial conditions of the trajectory at each node j is $\vec{s}_j$ at time τ_j, and the final value of the trajectory at time τ_{j+1} is denoted by $\vec{x}(\tau_{j+1}; \vec{s}_j)$, then $(m-2)$ *continuity conditions* should be satisfied (for a graphical representation see Fig. 1)

$$\vec{C}_j(\vec{s}_j, \vec{s}_{j+1}; T) = \vec{x}(\tau_{j+1}; \vec{s}_j) - \vec{s}_{j+1} = 0, \quad j = 1, 2, \cdots, m-2, \tag{21}$$

together with the boundary conditions

$$\vec{B}(\vec{s}_{m-1}, \vec{s}_1; T) = \vec{x}(\tau_m; \vec{s}_{m-1}) - \vec{s}_1 = 0. \tag{22}$$

Now, we have to solve $(m-1)$ initial value problems, and for that we adopt the Newton-Raphson method

$$\vec{C}_j(\vec{s}_j, \vec{s}_{j+1}; T) + \frac{\partial \vec{C}}{\partial \vec{s}_j}\Delta\vec{s}_j + \frac{\partial \vec{C}}{\partial \vec{s}_{j+1}}\Delta\vec{s}_{j+1} = 0. \tag{23}$$

These equations become

$$\vec{C}_j(\vec{s}_j, \vec{s}_{j+1}; T) + Z_j(\tau_{j+1})\Delta\vec{s}_j - \Delta\vec{s}_{j+1} = 0, \quad 1 \le j \le m-2. \tag{24}$$

Using the boundary conditions (Eqs (22)) we get

$$\vec{B}(\vec{s}_{m-1}, \vec{s}_1; T) + Z_{m-1}(\tau_m)\Delta\vec{s}_{m-1} - \Delta\vec{s}_1 = 0, \tag{25}$$

where,

$$Z_j(\tau_{j+1}) = \frac{\partial\vec{x}(\tau_{j+1}; \vec{s}_j)}{\partial\vec{s}_j}. \tag{26}$$

Eqs (24-25) are written in a matrix form of dimension $2N(m-1) \times 2N(m-1)$

$$\begin{bmatrix} Z_1 & -I_{2N} & 0 & \cdots & 0 & 0 \\ 0 & Z_2 & -I_{2N} & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & Z_{m-2} & -I_{2N} \\ -I_{2N} & 0 & 0 & \cdots & 0 & Z_{m-1} \end{bmatrix} \begin{bmatrix} \Delta\vec{s}_1 \\ \Delta\vec{s}_2 \\ \cdots \\ \Delta\vec{s}_{m-2} \\ \Delta\vec{s}_{m-1} \end{bmatrix} = - \begin{bmatrix} \vec{C}_1 \\ \vec{C}_2 \\ \cdots \\ \vec{C}_{m-2} \\ \vec{B} \end{bmatrix} \tag{27}$$

The above system of linear equations is solved by invoking the so called *condensing algorithm*[10]

$$\Delta\vec{s}_1 = -E^{-1}\vec{u}, \tag{28}$$

$$\Delta\vec{s}_{j+1} = Z_j\Delta\vec{s}_j + \vec{C}_j, \quad j = 1, 2, \cdots, m-2, \tag{29}$$

where

$$\begin{aligned} &E = Z_{m-1}Z_{m-2}\cdots Z_2Z_1 - I_{2N}, \\ &\vec{u} = \vec{B} + Z_{m-1}(\vec{C}_{m-2} + Z_{m-2}(\vec{C}_{m-3} + Z_{m-3}(\vec{C}_{m-4} + \cdots(\cdots + Z_2\vec{C}_1)\cdots))). \end{aligned} \tag{30}$$

B. The damped Newton-Raphson method

Quite often the Newton-Raphson method diverges, although, when it converges it does that quadratically. Problems of divergences are cured by scaling the Newton step with a parameter λ_k

$$\vec{s}_{k+1} = \vec{s}_k + \lambda_k\Delta\vec{s}_k, \tag{31}$$

where $0 \le \lambda_k \le 1$, and $\lambda_k \to 1$ as $\vec{s}_k \to \vec{s}_*$. Newton methods of this type are called *damped or underrelaxed.*

Several schemes for choosing λ_k have been proposed.[3,9] One is to require that the selected step decreases the function

$$f = \frac{1}{2}\vec{B}(\vec{s};T) \cdot \vec{B}(\vec{s};T). \tag{32}$$

It can be shown that the Newton step, $\Delta\vec{s}_k$, is a descent direction for f. Therefore, the strategy is first to test if the Newton step decreases sufficiently the function f. If not, we reduce the Newton step by a factor λ_k. A method to choose the appropriate λ_k is by backtracking along the Newton direction.[3] The method consists of modeling the function

$$g(\lambda_k) = f(\vec{s}_k + \lambda_k \Delta\vec{s}_k), \tag{33}$$

by a quadratic or cubic polynomial of λ_k, and then to find the value of λ_k that minimizes g.

The linear system of Eqs (19), $\Delta\vec{s}_k$, is solved by LU-Decomposition or Singular Value Decomposition methods. The convergence criteria that we use in the Newton-Raphson iteration scheme are

$$||\vec{B}(\vec{s}_k)|| = max\left(|B_i(\vec{s}_k)|\right) < d_1, \quad \text{Level 1}, \tag{34}$$

and

$$||\Delta\vec{s}_k|| = max\left(|\Delta s_{i(k)}|\right) < d_2, \quad \text{Level 2}. \tag{35}$$

d_1 and d_2 are preselected accuracy criteria.

C. The secant method

Another method that we have applied for solving Eqs (8), is a quasi Newton-Raphson type procedure for the approximate evaluation of the Fundamental Matrix. Instead of an exact calculation of Z at each iteration k, we can use Broyden updating formula[3]

$$P_{k+1} = P_k + \frac{(\Delta\vec{B}_k - P_k \cdot \Delta\vec{s}_k) \otimes \Delta\vec{s}_k}{\Delta\vec{s}_k^{+} \cdot \Delta\vec{s}_k}, \tag{36}$$

where $P_k = Z_k - I_{2N}$, $\Delta\vec{B}_k = \vec{B}_{k+1} - \vec{B}_k$ and $\Delta\vec{s}_k = \vec{s}_{k+1} - \vec{s}_k$. $\otimes$ denotes a direct product.

For the first value of P_0 it is advisable to start with an accurate evaluation of the Fundamental Matrix. In the algorithm described in ref.,[3] the *secant Broyden* *method* is combined with the backtracking scheme for global convergence. Then, a QR decomposition method is used to solve the linear system of equations at each iteration.

D. Implementation

The $(m-1)$ Fundamental Matrices required in the multiple shooting method may be evaluated either from numerically obtained derivatives or analytically. The first require the integration of $2N(m-1)$ neighboring trajectories, and the derivatives are then computed by finite differences. In the case that the analytic second derivatives of the Hamiltonian are available, we integrate Hamilton and the variational equations, Eqs. (17-18), simultaneously.

A stability analysis of the periodic orbit demands the diagonalization of the Monodromy Matrix.[16] This means, that we must integrate the variational equations of the Periodic Orbit for a full period T. However, an estimate of the Monodromy Matrix can be obtained from the product of Fundamental Matrices as are evaluated in the multiple shooting procedure

$$M = Z_{m-1} Z_{m-2} \cdots Z_2 Z_1. \tag{37}$$

From our experience we deduce that this does not always give good results.

Sometimes it is desirable to bring all periodic orbits on a common Poincaré surface of section. The Henon method[17] is not suitable for highly unstable systems. Then, it is more convenient to increase the boundary conditions by fixing one coordinate (momentum), i.e.,

$$x_l - \xi = 0, \qquad 1 \leq l \leq 2N, \tag{38}$$

and to consider, that the period of the periodic orbit satisfies the trivial differential equation[12]

$$\dot{T} = 0. \tag{39}$$

Hence, a $(2N+1)$ dimensional boundary value problem must be solved.

To obtain the whole family of periodic orbits at several energies (*continuation* of the family), we found it useful to use the period as the varying parameter. Continuation is carried out by predictor-corrector techniques with a trivial

$$\vec{x}_{new} = \vec{x}_{old}, \tag{40}$$

or a secant predictor[18]

$$\vec{x}_{new} = \vec{x}_1 + \mu(\vec{x}_1 - \vec{x}_0). \tag{41}$$

μ is a scaling factor, and $\vec{x}_{old}, \vec{x}_0$, and $\vec{x}_1$ are the initial conditions of the previous converged POs.

III. Program structure

The subroutines are organized in six directories: *main, root, anals, integr, lineq*, and *dat_tcl*. Figure 2 presents the structure of the program POMULT.

A. Directory *main*

The directory *main* contains the MAIN program that reads the input data, calls the root finding routines, calls for the analysis of the PO, and decides for the continuation scheme. In the same directory the subroutines (DEEVAL, VARSEC, JAC, JACOBN) that interface the integrators of the differential equations with the *user supplied routines* (POTEN, SECDER) are also stored. DEEVAL calls the routine with Hamilton equations (POTEN), and VARSEC, JAC, and JACOBN call the routine with the derivatives of Hamilton equations (SECDER), if the second derivatives of the potential are available. Otherwise, SECDER should be a dummy routine.

The subroutines which calculate the Fundamental (or Monodromy) Matrix (MONOD, MONODB), the boundary conditions (BRC), and the continuation scheme (CONTIN) are also stored here. MONODB gives an approximate Monodromy Matrix according to Eq. (37).

The routine START computes the initial conditions for the $m-2$ nodes given a starting point in phase space. The routine CH2D checks analytic with numeric second derivatives.

B. Directory *root*

ROOT is the routine which decides about the type of solver in the root finding process. GNEWT is the dumped Newton-Raphson solver, and SECANT is the secant-Broyden one. LNSRCH is the routine used for backtracking along the Newton direction.

FMIN calculates the function f (Eq. (32)), FMULTI computes the functions $\vec{B}$ by using the continuity and boundary conditions, Eqs (24-25). FDMULTI computes the Fundamental Matrices. Finally, SMULTI computes the right hand side of Eqs. (27) and solves the linear system of equations, Eqs (28-29). OTHER is a dummy routine which can be used to interface this program with user's supplied root finders.

C. Directory *anals*

In this directory we store all the subroutines needed for the analysis of the PO. DIAGON diagonalizes the Monodromy Matrix for the stability analysis of the PO. ACTION computes the action integral along the PO. PLOT stores points for plotting the PO. POINC computes Poincaré Surfaces of Section with the Henon method.[17]

Finally, FOURIER carries out a FFT of a stored trajectory, and LYAPUNOV estimates the Maximum Lyapunov Exponent. CCORF is a set of routines which calculate the classical autocorrelation function starting with Gaussian distributions for the initial conditions of the trajectories.

D. Directory *integr*

In this directory we store a number of integration routines for the first order differential equation, ranged from Runge-Kutta to Adams predictor-corrector algorithms. Routines for stiff equations are also included.

E. Directory *lineq*

In this directory we store routines for solving linear systems of equations based on LU, QR, and SVD (Singular Value Decomposition) methods. Routines for diagonalizing a matrix are found here.

F. Directory *dat_tcl*

The directory *dat_tcl* keeps the tcl-tk scripts[19,20] for constructing windows to interactively manipulate the input/output data of the POMULT program (PO_T.DATTCL), see Fig. 3. A window file-manager (BOSS) is also provided.

G. Include files

There are two include files; the PARAMETER.INC and CONSTANTS.INC. In the first include file, the maximum dimensions of the array variables are stored. Here, the dimension of phase space, *ndiml*, is declared.

In the second include file the values of some constants or unit conversion factors are defined.

IV. Input-output description

A. Open files

The following files are opened in the program:

Subroutine MAIN

channel	filename	description
1	po_t.dat	input data
2	po_t.out	output data
3	po_t.int	initial values for the nodes of the trajectory
		(to be read if iread -see bellow- not equal to 0)
9	po_t.nodes	store initial values and nodes of the periodic
		orbit for continuation
		if iread not equal to zero then
		copy this file to po_t.int
		if jprint -see below- greater than 0 store the
		initial values and nodes of all computed POs
10	po_t.res	store several output data for the POs to be used in
		plotting the continuation/bifurcation diagrams

11	po_t.wro	store the wroskian of the final PO

Subroutine POINC

channel	filename	description
4	po_t.pss	Poincare Surface of section data

Subroutine PLOT

channel	filename	description
8	po_t.plo	plotting data

Subroutine FOURIER

channel	filename	description
20	po_t.fourie-out	store frequency and intensity only if the intensity is greater than a tolerance value -toler-
30	po_t.fourie	store all computed frequencies and intensities for plotting the spectrum

Subroutine LYAPUNOV

channel	filename	description
40	po_t.lyapunov	store the rate of exponential divergence as a function of time

Subroutine CCORF

channel	filename	description
80	po_t.ccf	output from classical correlation analysis
81	po_t.cor	store the autocorrelation function of the selected variables
83	po_t.spc	store the FFT results of the autocorrelation function

B. Input Data

All input data are given in free format. They are grouped in categories and the group name is that which appears in the menu widget (Fig. 3). Here is their definition:

	DS_TITLE	
title	:	title for the studied dynamical system
	VAR_INIT	
xin(1:ndiml)	:	starting point in phase space; give coordinates
	:	first and conjugate momenta second
idiab	:	locate periodic orbits in a reduced dimension space
jdiab(1:idiab)	:	the coordinates that will be excluded
	VAR_EQUI	
xequ(1:ndiml)	:	equilibrium values of the coordinates and conjugate
	:	momenta, usually used for the definition of the
	:	variable that will be kept fixed or to define
	:	the Poincare surface of section
iang	:	the number of angle coordinates
jang(1:iang)	:	put the position numbers of the angle coordinates
	:	in the coordinate list; i.e. 3, if the third coordinate
	:	is the angle
	CONST_INT	
mnodes	:	the number of nodes along the periodic orbit
	:	for single shooting, mnodes = 2
	:	for multiple shooting, mnodes = mm
iread	:	decide to read or not the initial point and the nodes
	:	of the trajectory from the file "po_t.int"
	:	0 no read
	:	1 yes read
ivfix	:	decide to keep or not one variable fixed at the value
	:	"xequ(ivar)"
	:	0 no fixed value
	:	+1 fixed value "xc(ivar) = xequ(ivar)" ($1\leq$ ivar $\leq$ ndiml)

	:	-1 not implemented yet
jacob	:	decide to keep the Jacobian constant or not in the
	:	Newton-Raphson iterations
	:	0 keep the Jacobian constant after the first iteration
	:	n vary the Jacobian at each iteration step (n>0)
nmax	:	the total number of periodic orbits to be computed
	:	0 no location of POs, but only analysis of the current
	:	trajectory
nter	:	the maximum number of iterations in the
	:	Newton-Raphson procedure
ivarin	:	the variable which defines the constant
	:	plane of section
idire	:	decide for forward or backward continuation
	:	of the family of POs
	:	+n forward
	:	-n backward
idirt	:	decide for forward or backward integration in time
	:	+n forward
	:	-n backward
jprint	:	decide about the amount of output to be directed
	:	in the "@stdout"
	:	-1 no printing at all
	:	0 initial data - iterative values of level functions -
	:	solution data (or final data, respectively)
	:	+1 more data
	CONST_FLO	
deltat	:	time increment for the period of PO used in the
	:	continuation of the family of POs
tstep	:	integration time step
period	:	the period of PO
tolf	:	convergence criterion for minimizing the function (Level1)

tolx	:	convergence criterion for the solution of the linear
	:	system in Newton-Raphson method (Level2)
tol	:	criterion for characterizing a stationary point on the
	:	potential, used in the gnewt and secant subroutines
	:	
	:	it is also used in the SVD routines for reducing the
	:	rank of the matrix in solving the linear system of
	:	equations
eta	:	increment for the numerical calculation of the Jacobian
fcmin	:	minimum permitted value for the relaxation factor used
	:	in the damped Newton-Raphson algorithms
sigma	:	constant coefficient that determines the step with
	:	which the function f decreases in the damped
	:	Newton-Raphson algorithms
err1	:	relative error in the integration algorithms
err2	:	absolute error in the integration algorithms
	METH_INT	
meth0_int	:	choose integration algorithm
	:	1 de – Adams-Bashforth method
	:	2 rkf7 – Runge-Kutta 7th order method
	:	depac package[21]
	:	3 dderkf – Runge-Kutta-Fehlberg (4,5) method
	:	4 ddeabm – Adams-Bashforth method
	:	5 ddebdf – for stiff equations, backward differentiation
	:	formulae
	:	Numerical Recipes subroutines[3]
	:	6 rkqs – Runge-Kutta 5th order
	:	7 bsstep – Bulirsch-Store method
	:	8 stifbs – Bulirsch-Store method for stiff equations

METH_ROOT

meth0_root	:	choose the zero finding method
	:	1 call other, a user supplied algorithm
	:	2 call gnewt, damped Newton-Raphson algorithm
	:	3 call secant, a secant-Broyden algorithm
	METH_LIN	
meth0_lin	:	choose a solver for the linear algebraic system
	:	1 dlsvrr, singular value decomposition method
	:	2 dludcmp, lu-decomposition method
	:	3 dqrdcmp, qr-decomposition method
	METH_MON	
meth0_mon	:	meth0_mon is the same as the parameter -mshoot-
	:	(see data ANALYSIS)
	:	1 numerically
	:	2 analytically
	:	3 use odessa package[22] (not implemented)
	:	parameters in odessa package
itask	:	1 for normal computation of output values of y at
	:	t = tout
itol	:	an indicator for the type of error control
	:	1 for scalar relative and absolute errors
mf	:	method flag
iopt(1)	:	0 to indicate no optional inputs for integration
iopt(2)	:	1 to indicate sensitivity analysis, = 0, to indicate
	:	no sensitivity analysis
iopt(3)	:	0 no subroutine for inhomogeneity matrix df/dp
	ANALYSIS	
iplot	:	decide the rate of plotting the POs
	:	n every n^{th} periodic orbits
nppoints	:	the number of points stored to plot a PO
npoinc	:	decide to compute Poincare surfaces of sections
	:	-n just bring the periodic orbit on the

	:	Poincare surface of section
	:	+n the number of intersections of the trajectory
	:	with the Poincare surface of section
meig	:	decide the rate of computing the eigenvalues
	:	of the Monodromy Matrix of the located POs
	:	0 no eigenvalues
	:	+n every nth periodic orbits
ivect	:	decide if you want to compute eigenvectors
	:	0 no eigenvectors
	:	n yes eigenvectors
mshoot	:	the same as meth0_mon
	:	choose how to calculate the Fundamental/Monodromy Matrix
	:	-1 use Monodromy Matrix as coming from the multiple shooting
	:	algorithm
	:	1 use numerically calculated Jacobian
	:	2 integrate Hamilton and Variational equations
	:	simultaneously with analytical second derivatives and
	:	the current integrator
	:	3 use ODESSA package for computing the Monodromy Matrix
ifourie	:	compute power spectra with FFT transforms
	:	0 no FFT
	:	+n every nth trajectory
ilyap	:	compute the local maximum lyapunov exponent
	:	0 no MLE
	:	+n every nth trajectory
mcont	:	choose variable for continuation, $1 \leq$ mcont $\leq$ ndiml+1
	:	ndiml+1 variable is the period
msvar	:	decide for constant or variable parameter continuation step
	:	0 constant step
	:	n variable step with n the optimum iteration number for
	:	convergence. scale the parameter step by n/(No iterations)

mextr	:	decide for trivial or secant predictor in the continuation
	:	scheme
	:	0 trivial predictor, $x_{new} = x_{old}$
	:	1 secant predictor
	CCF	
ntime	:	decide to calculate the classical autocorrelation function
	:	0 no ccf
	:	+n yes ccf
ntrj	:	the number of trajectories used to compute ccf
nv	:	the number of coordinates for which the ccf is computed
iseed	:	seed for the random number generator
alpha0(1:ndiml)	:	the widths of the gaussians for initial value sampling
enlim0	:	upper bound for the energies of the selected trajectories
	WHAT	
	:	selecting *What* in the menu widget a window
	:	opens which allows you to load the
	:	file po_t.what that describes the data. You can highlight
	:	specific data and by clicking with button-3 an entry
	:	opens that allows you to change the values of these data
	:	
	:	this window allows also to load any file and to select any data
	:	from it which can be written in a file with the name po_t.selection

C. Error Diagnostics

The variable *kprint* notifies possible errors in failing to converge to the appropriate PO.

For the damped Newton-Raphson algorithm *kprint* means:

kprint	: iter converged after -iter- iterations
	: -1 converged to a stationary or turning point

: -2 NO convergence after nter iterations

: -3 NO convergence, stationary point in FMIN

: -4 Possibly converged; need further tests

For the secant algorithm *kprint* means:

kprint : iter converged after -iter- iterations

: -1 converged to a stationary or turning point

: -2 NO convergence after nter iterations

: -3 NO convergence, stationary point in FMIN

: -4 NO convergence, singular matrix

: -5 NO convergence after trying reinitialization

: of the Jacobian

V. Test run description

A Makefile creates the executable program. As a test case we use a relatively simple three dimensional system with the Hamiltonian[23]

$$H = \frac{1}{2}(p_x^2 + p_y^2 + p_z^2) + \frac{1}{2}(\omega_x^2 x^2 + \omega_y^2 y^2 + \omega_z^2 z^2) - \epsilon x^2 y - \eta x^2 z. \tag{42}$$

The values of the parameters are, $\omega_x^2 = 0.9$, $\omega_y^2 = 1.6$, $\omega_z^2 = 0.4$, $\epsilon = 0.08$, and $\eta = 0.01$. The harmonic frequencies satisfy the following resonance conditions; $\omega_x : \omega_y : \omega_z = 3 : 4 : 2$.

The subroutines POTEN and SECDER are stored in the directory *applications*. Here is the content of the output file po_t.out for one PO;

****

****

PO-Search-for-X2YZ

****

****

No of equations = 6

No of nodes = 4

method of integration = 1

method of root search = 2

method of solving l.eq. = 2

read initial vector = 0

fixing one variable = 0

const. or var. jacobian = 1

Equilibrium point

.00000000 .00000000 .00000000 .00000000 .00000000 .00000000

Angular variables : 0

Adiabatic variables : 0

integr. time step = .10000000E+00

initial period = .66253800E+01

variabl. for PSS (cont.)= 2

max. iter. in newton = 50

No of located p.o. = 1

increment step for p.o. = .10000000E-02

errf, errx = .10000000E-07 .10000000E-07

tolerance for ill-cond. = .10000000E-07

odessa v. itask,itol,mf = 1 1 11

iopt = 0 1 0

differentiation const. = .10000000E-06

minimum relaxation fac. = .10000000E-01

sigma = .10000000E+00

abserr, relerr in int. = .10000000E-11 .10000000E-11

direction in time = 1

direction in integr. = 1

iplot = 0

nppoints = 500

npoinc = 0

meig = 1

ivect = 0

mshoot = 2

ifourie = 0

ilyap = 0

mcont = 7

msvar = 0

mextr = 0

ntime = 0

ntrj = 1

nv = 1

iseed = 123456789

****

widths of gaussians

8.00000000 8.00000000 8.00000000 8.00000000 8.00000000 8.00000000 .00000

****

Starting point in phase space

.00000000 .01000000 .00336900 .46408150 .00000000 .00000000

6.62538000

****

analytic jacobian seems to be o.k.

****

****

TRAJ. No = 1

****

X(1) = .000078421587 X(2) = .010786284687 X(3) = .003369013419

X(4) = .464085471270 X(5) = -.000002915863 X(6) = -.000000227704

Period = 6.625380000000

Energy = .107783010288

Accur. = .92E-08

Kprint = 1

Action = .113633177667

Eigenvalues

real	imaginary	norm	phase (in degrees)
-.499361993394D+00	.867405522180D+00	.100087698563D+01	.119928867251D+03
-.499361993394D+00	-.867405522180D+00	.100087698563D+01	-.119928867251D+03
-.498487277651D+00	.865886116868D+00	.999123782804D+00	.119928867251D+03
-.498487277651D+00	-.865886116868D+00	.999123782804D+00	-.119928867251D+03
.999999999917D+00	.128911023608D-04	.100000000000D+01	.738605758566D-03
.999999999917D+00	-.128911023608D-04	.100000000000D+01	-.738605758566D-03

Determinant = .1000000000E+01

Denper = .00000000E+00

*** Job ended with kprint = 1

References

[1]S. C. Farantos, Int. Rev. Phys. Chem **15**, 345–374 (1996).

[2]S. C. Farantos, Chem. Phys. **159**, 329 (1992).

[3]W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling. *Numerical Recipes.* Cambridge Univ. Press, (1986).

[4]U. Feudel and W. Jansen, Int. J. Bifurc. and Chaos **2**, 773 (1992).

[5]M. A. M. Aguiar, C. P. Malta, M. Baranger and K. T. R. Davies, Annals Phys. **180**, 167 (1987).

[6]M. Baranger, K. T. R. Davies and J. H. Mahoney, Annals Phys. **186**, 110 (1988).

[7]K. T. R. Davies, T. E. Huston and M. Baranger, Chaos **2**, 215 (1992).

[8]P. Deuflhard, Numer. Math. **33**, 115 (1979).

[9]P. Deuflhard, Numer. Math. **22**, 189 (1974).

[10]H. B. Keller. *Numerical Solution of Two Point Boundary Value Problems*, volume 24. Society for Industrial and Applied Mathematics, (1976).

[11]J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* New York, Springer, (1980).

[12]R. Seydel. *From Equilibrium to Chaos: Practical bifurcation and stability analysis.* Elsevier, (1988).

[13]E. Reithmeier. *Periodic Solutions of Nonlinear Dynamical Systems.* Lecture Notes in Mathematics, Springer-Verlag, (1991).

[14]A. J. Lichtenberg and M. A. Libermann. *Regular and Stochastic Motion.* Springer, New York, (1981).

[15]J. M. G. Llorente and E. Pollak, Ann. Rev. Phys. Chem. **43**, 91 (1992).

[16]S. C. Farantos. In *Time Dependent Quantum Mechanics: Experiments and Theory*, edited by J. Broeckhove. Plenum Co. Ltd, (1992).

[17]M. Henon, Physica D **5**, 412 (1982).

[18]R. Seydel, Int. J. Bifurc. and Chaos **1**, 3 (1991).

[19]J. K. Ousterhout. *Tcl and the TK Toolkit.* Addison-Werley, (1994).

[20]B. B. Welch. *Practical Programming in Tcl and Tk.* Prentice Hall PTR, (1995).

[21]L. F. Shampine and M. K. Gordon. *Computer Solution of Ordinary Diffrential Equations: The initial value problem.* W. H. Freeman and Co., San Francisco, (1975).

[22]J. R. Leis and M. A. Kramer, ACM Trans. Math. Software (1985).

[23]G. Contopoulos, S. C. Farantos, H. Papadaki and C. Polymilis, Phys. Rev. E **50**, 4399 (1994).

Figures

1. A schematical representation of the multiple shooting procedure.

2. A diagram which shows the subroutines of the program POMULT and their interconnectivity. The oval type graphs describe directories, that contain routines for integrating Hamilton and variational equations, (*integr*), for solving linear algebraic equations, (*lineq*), and tcl-tk scripts for GUI, (*dat_tcl*).

3. The interface window for the definition of input data in the program POMULT.