PS/AR Note 93-23 (Min.)

SAP-07

2 December 1993

Minutes of the Forum of

Symbolic Computing for Accelerator Physics

held on Friday 5 November 1993

Present: B. Autin (Chairman), F. Barbarin, G. Dôme, W. Fischer, J. Gareyte, M. Giovannozzi, S. Hancock, E. Jensen, J. Jowett (Deputy), M. Martini (Secretary), D. Manglunki, F. Méot, H. Mulder, J.C. Schnuriger, G. Shirkov.

---

1 Dynamic aperture visualization: J. Jowett

The graphics facilities of *Mathematica* are used to display 3D dynamic apertures. Input data are provided by tracking results performed with MAD. The technical point which has to be solved concerns the treatment of data which are not evenly distributed on a grid (scatter plot).

A description of the program is given in the appendix.

2 Time dependent perturbation of dynamical systems: B. Autin

The perturbative treatment of dynamical systems has been one of the very first topics treated by symbolic codes, especially in the field of celestial mechanics. The method developed in the package NLDynamics is based on a classical iterative procedure (Picard method) for solving ordinary differential equations. It is sometimes called time dependent perturbations in contrast with frequency domain perturbation. Such a method is the natural extension of Courant and Snyder theory to non-linear fields and can be considered as the symbolic counterpart of numerical tracking programs.

The complete theory involves 2 steps:

- Perturbative form of the differential system at each iteration.
- Integration of the perturbed equations.

The present version of the program deals only with the first step but the formalism is prepared for a further integration. The integration contains indeed multiple integrals which express the correlation of the various field sources and the "time" variable is thus labelled to be recognized by the multiple integrals.

The program can be applied to any system which has to be linearized or expanded to a higher-order. It is then able to provide analytical formulae which can be used for correction systems or for checking numerical simulations. Two examples are given in the attached slides: one concerns the transverse coupled motion in the non-linear field of an accelerator, and the second the equations of a Free-Electron-Laser.

3 Porting a program on MathSource: B. Autin

Many programs have been developed with *Mathematica.* The authors are encouraged to port them on *MathSource*, the electronic forum of *Wolfram Research.* Two conditions must be fulfilled:

1. The program must be thoroughly documented in a Notebook;
2. the functions of the program have to be accessible from outside.

A package is typically:

```
BeginPackage[" ... `"] (* package name *)

Function1::usage=" ... " (* description of functions *)

...

Begin["`private` ... `"] (* private context *)

Function1[param1_, ... ] := ... (* definition of functions *)

...

End[] (* end of private context *)

EndPackage[]
```

See the attached slide.

J.C. Schnuriger has kindly accepted to help the authors who wish to establish a communication with *MathSource.*

The next meeting will be held on:

Thursday 16 December at 16.00 hr in the Adams room - Prévessin, Bldg 864, 2-B14

M. Martini

Distribution list

AT, MT, PS and SL Division Leaders and Deputies.
AT, MT, PS and SL Group Leaders and Associates.
SAP list.

A Dynamic aperture visualization: J. Jowett

A.1 Introduction

This notebook manipulates and displays dynamic aperture data, usually found by tracking with MAD. To be able to use this notebook the choice of initial conditions for the tracked particles must follow the rules given in "Selection of initial conditions" below. The data are normally read in from an external file specified as defined in "Input file format" below. The notebook will provide functions which operate on these data to display various views of the 3-dimensional dynamic aperture. These include various 2-dimensional cuts, including the cuts along the 3 coordinate planes and along the "fully-coupled" emittance plane. All functions, variables and objects defined in this notebook (to become a package?) will have names beginning with "dynap". The remainders of the names will consist of complete words with the initial letter capitalized.

A.2 Input file format

For now at least, the input file is created from the output of several MAD tracking jobs using a *Unix* "awk" program script. (It would have been nice to do it with *Mathematica* but, at least on the PC, there seem to be some bugs in the file searching functions. See the notebook `prfilt2.ma` in this directory for attempts in this direction. This may be something to do with the DOS file system and may not happen on *Unix*, so there is still hope to filter the MAD print jobs using *Mathematica* itself.

The input file has the following structure:

```
<MAD title line>
Qx Qy Qs
Ex Ey sigmae
phiPoints  thetaPoints
SAx11 SAy11 SAt11
SAx12 SAy12 SA12
........
SAx_1phiPoints SAy_1phiPoints SAt_1phiPoints
SAx21 SAy21 SAt21
SAx22 SAy22 SAt22
.........
.........
SAx_thetaPoints_phiPoints SAy_thetaPoints_phiPoints
SAt_thetaPoints_phiPoints
```

The tunes {`Qx,Qy,Qs`} will usually be just the fractional parts. The emittances `Ex` and `Ey` are given in micro-metres. The points {`SAx,SAy,SAt`} define the boundary of the dynamic aperture. Each corresponds to a definite direction in spherical polar coordinates in square-root-of-action or "`squaction`" variable space. The units are square-root metres for `SAx` and `SAy`, fractional momentum deviation for `SAt`.

A.3 Implementation of data structure

A.3.1 Units and rescaling of input data

We usually need to convert the input data to standard units of `squaction`: `10^3 Sqrt[Ax/m]`, `10^3 Sqrt[Ay/m]` and % (percent) momentum deviations. These factors are automatically applied

when the dynamic aperture data is read in using the function **dynapRead** defined below.

```
dynapInputRescaleFactor={10^3,10^3,10^2}
{1000, 1000, 100}
```

A.3.2 Rescaling in squaction space

A simple way to rescale a 3-dimensional point is just to multiply it by another one. The first part of the definition of the following function rescales the point {a,b,c} along each coordinated axis by the factors {fx,fy,ft} to get {fx a, fy b, ft c}; the functions makes sure that the "coordinates" themselves have no deeper structure. The rest of the definition applies the same rescaling to 1- and 2-dimensional lists of points (e.g. an emittance ellipsoid).

```
Clear[dynapStretch]
dynapStretch[{fx_,fy_,ft_},pnt:{ _,  _, _}]:=
             {fx,fy,ft} pnt /;Depth[pnt]==2
dynapStretch[{fx_,fy_,ft_},pntList{{ _,  _, _}..}]:=
        Map[dynapStretch[{fx,fy,ft},#]&,pntList]
dynapStretch[{fx_,fy_,ft_},pntArray:{{{ _,  _, _}..}..}]:=
        Map[dynapStretch[{fx,fy,ft},#]&,pntArray,{2}]
```

The following anticipates the structure of a **dynapData** object which is defined in the functions **dynapRead** and **dynapEmittanceEllipsoid** below. It performs surgery on such objects, rescaling the array of points representing a surface and recording the fact by prefixing the title string with the values of {fx, fy, ft}. In the case of an emittance ellipsoid which is rescaled just once (the most likely usage), this will produce something readable. We do not bother to do much about other cases.

```
dynapStretch[{fx_,fy_,ft_},dynap_dynapData]:=
 ReplacePart[Join[Drop[dynap,-1]
                  ,dynapData[dynapStretch[{fx,fy,ft},Last[dynap]]]
                  ]
             ,StringJoin[ToString[{fx,fy,ft}],Part[dynap,2]]
             ,2]
```

We shall use this kind of pattern-matching approach in the definitions of several other functions (including graphics functions) below. Define a standard set of values for {fx,fy,ft} which will be used in certain standard plots.

```
dynapStandardStayClear={10,10,7}
{10, 10, 7}
```

A.4 Function to read a dynamic aperture from a file

The following function reads a file (whose name is given as argument) and returns a dynamic aperture object. The dynamic aperture object has its own data type, **dynapData** (which might otherwise be a nested list). The structure of this object is implicit in the definition of this function.

The first few elements contain information on the dynamic aperture being studied, e.g. the first element will always be the input stream assigned to the file. The following elements contain other things, whose number may grow. The last element of the dynamic aperture object is the most important and is a matrix of points in 3-dimensional square-root-of-action variable space. Note

that the boundary data points are automatically re-scaled, using the function dynapStretch and the dynapInputRescaleFactor, as they are read in.

```
dynapRead[filename_]:=
       Block[{dynapStream,dynapPhiPoints,dynapThetaPoints},
            dynapData[
             dynapStream=OpenRead[filename],
             Read[dynapStream,String],
             Read[dynapStream,{Number,Number,Number}],
             Read[dynapStream,{Number,Number,Number}],
             dynapPhiPoints=Read[dynapStream,Number],
          dynapThetaPoints=Read[dynapStream,Number],
             dynapStretch[dynapInputRescaleFactor,
                  Partition[ReadList[dynapStream,{Number,Number,Number}]
                            ,dynapPhiPoints
                           ]
                      ]
                  ]
            ]
```

A.5 Functions to extract pieces of a dynamic aperture

The following functions can be used to extract pieces of the dynamic aperture object. It may be necessary to add to this list of definitions when the structure of the dynamic aperture object is extended. dynapFileName[dynap_dynapData]:=First[First[dynap]] The first line in the file can be up to 133 characters long. Generally we don't want the first character (which is for line-printer carriage control) and those towards the end which usually give the MAD version number.

```
dynapTitle[dynap_dynapData]:=
    StringTake[Part[dynap,2],{1,Min[50
                                    ,StringLength[Part[dynap,2]]
                                    ]
                              }
              ]
dynapTunes[dynap_dynapData]:=Part[dynap,3]
dynapEmittances[dynap_dynapData]:=Part[dynap,4]
dynapPhiPoints[dynap_dynapData]:=Part[dynap,5]
dynapThetaPoints[dynap_dynapData]:=Part[dynap,6]
dynapBoundary[dynap_dynapData]:=Last[dynap]
```

A.6 The natural emittance ellipsoid

This function generates a matrix of points on any emittance ellipsoid, defined by emittance values (in micrometer) and a fractional energy deviation (in units of the beam energy). Besides it's obvious use to generate the "1-sigma" ellipsoid, we can easily scale it (using either an overall multiplicative factor or, more usefully, the function dynapStretch) to get the "10-sigma" ellipsoid or whatever. When the argument is a dynamic aperture object it uses the natural emittances specified in the object.

A.7 The 3D dynamic aperture as points

A function which makes a scatter plot in 3D of all the points in a dynamic aperture list. To do this it needs to suppress one level of structure in the matrix of boundary points (so obtaining a list of points) and then feed the result to the package function `ScatterPlot3D`. It turns out to be convenient to define this function for two different patterns of argument. When the argument is a dynamic aperture object, the function calls itself recursively on the dynamic aperture boundary. When the argument has the form of a dynamic aperture boundary (checked by pattern matching) then the work actually gets done.

A.8 The 3D dynamic aperture as a surface

Essentially, we want a function which makes a surface plot in 3D of all the points in a dynamic aperture list. Here the order of the points and the matrix structure of the dynamic aperture boundary are important. The basic idea is similar to the function `ListSurfacePlot3D` in standard package `Graphics`Graphics3D`. It turns out to be convenient to define our own function in a few steps. The "innermost" one does pattern matching to make sure we really have the boundary point matrix. Other patterns include unwrapping the boundary from inside a `dynapData` object. I have partly added information, axes etc. to these plots using the mechanism of `dynapOptions3D`. It seems we can't easily plot two surfaces with `ListSurfacePlot3D`, so looking inside to see how it works, we copy the following definition from the `Graphics`Graphics3D`` package. Then we can construct a surface plot of two or more surfaces quite easily. This led me to further overload the definition of `dynapSurfacePlot3D` to make it accept a list of surfaces.

A.9 Two-dimensional cuts of dynamic aperture

Projection operators which will project points, lists of points, arrays of points or dynamic aperture data objects onto any of the three coordinate planes. Selections of points close to particular surfaces in action space provide 2D cross-sections of the dynamic aperture.

Create transportable package.

Applications.

Reading files into data structures.

Views of dynamic aperture data to compare LEP2 optics.

Perturbation of a differential system

The mechanism of the perturbation technique will first be explained for a single differential equation then generalized to a set of differential equations. Let us consider the first order equation which describes the evolution of the action J of a dynamical system with respect to a variable s:

$$dJ / ds = f(J)$$

In its ground state, the system is submitted to a certain field of forces F such that the action is a constant of the motion and has the value J_0. The right hand side of the equation is then zero. Under the effect of new fields assumed to be small with respect to F, the action J_0 is modified and can be expanded in the form

$$J = J_0 + J_1 + J_2 + \ldots$$

where J_1, J_2, ... are the first, second, ... order perturbations of J. The iterative solution of the differential equation at the order n is obtained by substituting to J at the right hand side its expression at the order n-1. J_1 is thus obtained by integrating

$$dJ_1 / ds = f\left(J_0\right)$$

and the integral form is

$$J_1(L) = \int_0^L J_0(s)\, ds$$

At the second order, the differential equation becomes

$$d\left(J_1 + J_2\right) / ds = f\left(J_0 + J_1\right)$$

which is reduced to

$$dJ_2 / ds = \left(df / dJ\right)_0 J_1(s)$$

The right hand side is known since the derivative of f with respect to J is taken for J_0 and J_1 is the result of the previous iteration. The solution to the equation is thus

$$J_2(L) = \int_0^L \left(dF / dJ\right)_0 J_1(s)\, ds$$

There is at this point a confusion to be avoided in the notations. J_1 is an integral and the previous result is therefore a double integral whose two variables need be distinguished. The detailed form of the solution will therefore be written

$$J_2(L) = \int_0^L \left(dF / dJ\right)_0 ds_1 \int_0^{s_1} J_1(s_0)\, ds_0$$

At the third order, the need for a truncation of the right hand side appears so that the full expansion

$$d\left(J_1 + J_2 + J_3\right) / ds = f\left(J_0 + J_1 + J_2\right) =$$
$$f\left(J_0\right) + \left(df / dJ\right)_0 \left(J_1 + J_2\right) + \left(1 / 2\right)\left(d^2 f / dJ^2\right)_0 \left(J_1 + J_2\right)^2$$

is reduced to

$$d\left(J_3\right) / ds = \left(df / dJ\right)_0 J_2 + \left(1 / 2\right)\left(d^2 f / dJ^2\right)_0 J_1^2$$

since the terms in $J_1 J_2$ and $J_2{}^2$ are of third and fourth order respectively and the integration gives:

$$J_3(L) = \int_0^L \left(\left(dF / dJ\right)_0 J_2(s) + \left(1 / 2\right)\left(d^2 F / dJ^2\right)_0 J_1(s)^2 \right) ds$$

The first contribution is a triple integral and the second one a double integral whose detailed forms are

$$J_3(L) = \int_0^L \left(dF / dJ\right)_0 ds_2 \int_0^{s_2} \left(dF / dJ\right)_0 ds_1 \int_0^{s_1} J_1(s_0)\, ds_0 +$$
$$\left(1 / 2\right) \int_0^L \left(d^2 F / dJ^2\right)_0 ds_2 \left(\int_0^{s_2} J_1(s_0)\, ds_0 \right)^2$$

The multiple integrals are the signature of the correlations between the perturbing fields for which a detailed discussion at the second order is given in [2]. Once the perturbation has been illustrated for a single differential equation, its extension to a system of differential equations is done by replacing the scalar J by a vector **J**.

Symbolic treatment of the perturbation

As soon as the calculation deals with a high order of iteration, the method becomes untractable by hand and one has to resort to a symbolic technique which is composed of two parts: the transformation of the exact but unsolvable system into an approximate solvable system at any order and the integration of the approximate systems starting at the lowest order. The integration is outside the scope of this paper but the perturbation calculation is organized to be suitable for a further integration.

As it has been seen in the previous section , the initial system is split into as many sub-systems as there are iterations, say n. The program thus produces a list of the right hand sides of the sub-systems. If the initial set contains a single equation, the list is a n-vector, if it is made of m equations, the output list is a (n*m)-matrix. The integration of these lists provide J_1, J_2, ... or $\mathbf{J_1}, \mathbf{J_2}, \cdots$

The code is written with *Mathematica* and sets the perturbation in a recursive form as it has been presented so that the expression of the n-th iteration contains the non evaluated results of the n-1 previous iterations. Each approximation of J is denoted J[i] where i is the iteration number. The evaluation is made by means of substitution rules collected in a single list:

$$\{J[1] -> rhs1, J[2] -> rhs2, \cdots\}$$

The right hand sides result from a Taylor expansion of the initial right hand side to the order n of the iteration and the expansion is truncated to eliminate spurious higher order terms. The recognition of the order of a special term uses the argument of J and the power of the term in the series expansion; if the sum of the argument and of the exponent exceeds n, the term is rejected.

At the iteration n+1, the name s of the variable is updated to sn so that s is converted to the sequence of values (s0, s1, s2, ...) in the same way as J is split into (J[1], J[2], ...). The purpose of this re-labeling of the variable is to obtain expressions ready for multiple integrations.

Description of the code

The program of perturbation of differential equations is part of a package of non linear dynamics called NLDynamics. Its name is ODEPerturbation and it is called with the command

ODEPerturbation[{equations},iteration]

The "equations" are logical equalities of the type

$$J'[s] == f[s, J]$$

where "'" means the derivative of **J** with respect to s. "iteration" is an integer denoting the wanted iteration. This syntax is remarkably simple as compared with the *Mathematica* command for differential equations

DSolve[{equations},{functions},variable]

It automatically detects the variable and the functions by deciphering the left hand sides. The perturbation of a single differential equation is thus given by

$$\mathbf{ODEPerturbation}\left[\{j'[s] = f[s,j]\},2\right] = \{f(s0,j0), j(1) f^{(0,1)}(s1,j0)\}$$

The result is a list of two elements which are the result j(1) and j(2) of the two iterations

$$f(s0, j0)$$

and

$$j(1)\, f^{(0,1)}(s1,j0)$$

As f acts on two variables s and j, $f^{(0,1)}$ is the *Mathematica* notation for the partial derivative of f with respect to the second variable j. The variable s is labled s0 at the first iteration so that, after substitution of j(1), there is no ambiguity to integrate j(2) first with respect first to s1 from 0 to s0 then to s0 from 0 to some fixed limit say L.

Application to a dynamical system

A dynamical system is in the most general case described in the so called *phase-space* by three actions J_x, J_y, J_z and three angles ϕ_x, ϕ_y, ϕ_z corresponding to the coordinates (x, y, z) of the real space. The system of differential equations is derived from the potential V of the non-linear field through the Hamilton's equations [3]:

$$dJ / ds = -\partial V / \partial \phi$$
$$d\phi / ds = \partial V / \partial J$$

In particle accelerators, the functions of the right hand sides can be decomposed into a product of three functions which depend either on the action or on the angles or even on the variable only. One of the most crucial points is the determination of the transverse motion to determine as a basic parameter as the aperture of the vacuum chamber. The system to solve is then made of four coupled equations

$$dJ_x / ds = f_1(s)\, g_1\left(s, J_x, J_y\right) h_1\left(s, \phi_x, \phi_y\right)$$
$$dJ_y / ds = f_2(s)\, g_2\left(s, J_x, J_y\right) h_2\left(s, \phi_x, \phi_y\right)$$
$$d\phi_x / ds = g_3\left(s, J_x, J_y\right) h_3\left(s, \phi_x, \phi_y\right)$$
$$d\phi_y / ds = g_4\left(s, J_x, J_y\right) h_4\left(s, \phi_x, \phi_y\right)$$

If the entire system is called *sys*, its perturbative solution at the second order is given by

ODEPerturbation[sys,2]

which returns:

$$
\begin{aligned}
&\{\{f1(s0)\,g1(s0,j_{x0},j_{y0})\,h1(s0,\phi_{x0},\phi_{y0}),\\
&\quad f1(s1)\,h1(s1,\phi_{x0},\phi_{y0})\left(j_y(1)\,g1^{(0,0,1)}(s1,j_{x0},j_{y0})+j_x(1)\,g1^{(0,1,0)}(s1,j_{x0},j_{y0}\right)+\\
&\quad f1(s1)\,g1(s1,j_{x0},j_{y0})\left(\phi_y(1)\,h1^{(0,0,1)}(s1,\phi_{x0},\phi_{y0})+\phi_x(1)\,h1^{(0,1,0)}(s1,\phi_{x0},\phi_{y0})\right)\},\\
&\{f2(s0)\,g2(s0,j_{x0},j_{y0})\,h2(s0,\phi_{x0},\phi_{y0}),\\
&\quad f2(s1)\,h2(s1,\phi_{x0},\phi_{y0})\left(j_y(1)\,g2^{(0,0,1)}(s1,j_{x0},j_{y0})+j_x(1)\,g2^{(0,1,0)}(s1,j_{x0},j_{y0})\right)+\\
&\quad f2(s1)\,g2(s1,x0,j_{y0})\left(\phi_y(1)\,h2^{(0,0,1)}(s1,\phi_{x0},\phi_{y0})+\phi_x(1)\,h2^{(0,1,0)}(s1,\phi_{x0},\phi_{y0})\right)\},\\
&\{g3(s0,j_{x0},j_{y0})\,h3(s0,\phi_{x0},\phi_{y0}),\\
&\quad h3(s1,\phi_{x0},\phi_{y0})\left(j_y(1)\,g3^{(0,0,1)}(s1,x0,j_{y0})+j_x(1)\,g3^{(0,1,0)}(s1,j_{x0},j_{y0})\right)+\\
&\quad g3(s1,j_{x0},j_{y0})\left(\phi_y(1)\,h3^{(0,0,1)}(s1,\phi_{x0},\phi_{y0})+\phi_x(1)\,h3^{(0,1,0)}(s1,\phi_{x0},\phi_{y0})\right)\},\\
&\{g4(s0,j_{x0},j_{y0})\,h4(s0,\phi_{x0},\phi_{y0}),\\
&\quad h4(s1,\phi_{x0},\phi_{y0})\left(j_y(1)\,g4^{(0,0,1)}(s1,j_{x0},j_{y0})+j_x(1)\,g4^{(0,1,0)}(s1,x0,j_{y0})\right)+\\
&\quad g4(s1,j_{x0},j_{y0})\left(\phi_y(1)\,h4^{(0,0,1)}(s1,\phi_{x0},\phi_{y0})+\phi_x(1)\,h4^{(0,1,0)}(s1,\phi_{x0},\phi_{y0})\right)\}\}
\end{aligned}
$$

At this level, it is easy to check the result at a glance but, as soon as the order increases, the number of terms grows in an an inflationary way and the interest of a symbolic code becomes obvious.

Acknowledgments

Fabrice Ajalbert from the University of Clermont-Ferrand made a major contribution to the development of the symbolic code.

References

[1] *Nonlinear Dynamics Aspects of Particle Accelerators*, Edited by J.M. Jowett, M. Month and S. Turner, Lecture Notes in Physics, 247, Springer-Verlag (1985)

[2] B. Autin, *Non-linear Betatron Oscillations*, CERN 90-04 (1990)

[3] H. Goldstein, *Classical Mechanics*. Addison Wesley (1980)

FEL Equations

```
eqn={gamma'[z]==(a1/gamma)f[e,theta]-e,
      theta'[z]==a2-a3(1-a4/(gamma^2))^(-1/2)}
```

```
ODEPerturbation[eqn,2]
```

$$\{\{(\gamma_1)'(z) = -e + \frac{a1 f(e,\theta_0)}{\gamma_0},$$

$$(\theta_1)'(z) = a2 - \frac{a3\gamma_0^2\sqrt{\frac{-a4+\gamma_0^2}{\gamma_0^2}}}{-a4+\gamma_0^2}\},$$

$$\{(\gamma_2)'(z) = -\frac{a1 f(e,\theta_0)\gamma_1}{\gamma_0^2} + \frac{a1\theta_1 f^{(0,1)}(e,\theta_0)}{\gamma_0},$$

$$(\theta_2)'(z) = \frac{a3\,a4\,\gamma_0\sqrt{\frac{-a4+\gamma_0^2}{\gamma_0^2}}\gamma_1}{\left(-a4+\gamma_0^2\right)^2}\}\}$$

Integration of first order equations:

```
sol=DSolve[ODEPerturbation[eqn,2][[1]],{gamma[1],theta[1]},z]
{gamma[1][x],theta[1][x]}/.sol
```

$$\{-(ex) + C(2) + \frac{a1 x f(e,\theta_0)}{\gamma_0},$$

$$C(1) - \frac{a2\,a4\,x}{-a4+\gamma_0^2} + \frac{a2x\gamma_0^2}{-a4+\gamma_0^2} - \frac{a3x\gamma_0^2\sqrt{\frac{-a4+\gamma_0^2}{\gamma_0^2}}}{-a4+\gamma_0^2}\}$$

Complete equations for 2 electrons

$$(\gamma(1)_1)'(z) = \frac{-\left(a5\,\sin(\theta(1)_0-\theta(2)_0)\right)}{2} - \frac{a1\,e_0\,\sin(\phi_0+\theta(1)_0)}{\left(1-\frac{a4}{\gamma(1)_0^{\,2}}\right)\gamma(1)_0}$$

$$(\gamma(2)_1)'(z) = \frac{a5\,\sin(\theta(1)_0-\theta(2)_0)}{2} - \frac{a1\,e_0\,\sin(\phi_0+\theta(2)_0)}{\left(1-\frac{a4}{\gamma(2)_0^{\,2}}\right)\gamma(2)_0}$$

$$(\theta(1)_1)'(z) = a2 - \frac{a3\,\gamma(1)_0^{\,2}\sqrt{\frac{-a4+\gamma(1)_0^{\,2}}{\gamma(1)_0^{\,2}}}}{-a4+\gamma(1)_0^{\,2}}$$

$$(\theta(2)_1)'(z) = a2 - \frac{a3\,\gamma(2)_0^{\,2}\sqrt{\frac{-a4+\gamma(2)_0^{\,2}}{\gamma(2)_0^{\,2}}}}{-a4+\gamma(2)_0^{\,2}}$$

$$(e_1)'(z) = \frac{a6\,\sin(\phi_0+\theta(1)_0)\gamma(1)_0\sqrt{\frac{-a4+\gamma(1)_0^{\,2}}{\gamma(1)_0^{\,2}}}}{2\left(-a4+\gamma(1)_0^{\,2}\right)} + \frac{a6\,\sin(\phi_0+\theta(2)_0)\gamma(2)_0\sqrt{\frac{-a4+\gamma(2)_0^{\,2}}{\gamma(2)_0^{\,2}}}}{2\left(-a4+\gamma(2)_0^{\,2}\right)}$$

$$(\phi_1)'(z) = \frac{a6\,\cos(\phi_0+\theta(1)_0)\gamma(1)_0\sqrt{\frac{-a4+\gamma(1)_0^{\,2}}{\gamma(1)_0^{\,2}}}}{2e_0\left(-a4+\gamma(1)_0^{\,2}\right)} + \frac{a6\,\cos(\phi_0+\theta(2)_0)\gamma(2)_0\sqrt{\frac{-a4+\gamma(2)_0^{\,2}}{\gamma(2)_0^{\,2}}}}{2e_0\left(-a4+\gamma(2)_0^{\,2}\right)}\Bigg\}$$

$$(\gamma(1)_2)'(z) =$$

$$-\frac{a1\cos(\phi_0+\theta(1)_0)e_0\phi_1}{\left(1-\frac{a4}{\gamma(1)_0^2}\right)\gamma(1)_0}-\frac{a1\,e_1\sin(\phi_0+\theta(1)_0)}{\left(1-\frac{a4}{\gamma(1)_0^2}\right)\gamma(1)_0}+$$

$$\frac{2a1a4e_0\sin(\phi_0+\theta(1)_0)\gamma(1)_1}{\left(1-\frac{a4}{\gamma(1)_0^2}\right)^2\gamma(1)_0^4}+\frac{a1\,e_0\sin(\phi_0+\theta(1)_0)\gamma(1)_1}{\left(1-\frac{a4}{\gamma(1)_0^2}\right)\gamma(1)_0^2}-$$

$$\frac{a5\cos(\theta(1)_0-\theta(2)_0)\theta(1)_1}{2}-\frac{a1\cos(\phi_0+\theta(1)_0)e_0\theta(1)_1}{\left(1-\frac{a4}{\gamma(1)_0^2}\right)\gamma(1)_0}+\frac{a5\cos(\theta(1)_0-\theta(2)_0)\theta(2)_1}{2}$$

$$(\gamma(2)_2)'(z) =$$

$$-\frac{a1\cos(\phi_0+\theta(2)_0)e_0\phi_1}{\left(1-\frac{a4}{\gamma(2)_0^2}\right)\gamma(2)_0}-\frac{a1\,e_1\sin(\phi_0+\theta(2)_0)}{\left(1-\frac{a4}{\gamma(2)_0^2}\right)\gamma(2)_0}+$$

$$\frac{2a1a4e_0\sin(\phi_0+\theta(2)_0)\gamma(2)_1}{\left(1-\frac{a4}{\gamma(2)_0^2}\right)^2\gamma(2)_0^4}+\frac{a1\,e_0\sin(\phi_0+\theta(2)_0)\gamma(2)_1}{\left(1-\frac{a4}{\gamma(2)_0^2}\right)\gamma(2)_0^2}+$$

$$\frac{a5\cos(\theta(1)_0-\theta(2)_0)\theta(1)_1}{2}-\frac{a5\cos(\theta(1)_0-\theta(2)_0)\theta(2)_1}{2}-\frac{a1\cos(\phi_0+\theta(2)_0)e_0\theta(2)_1}{\left(1-\frac{a4}{\gamma(2)_0^2}\right)\gamma(2)_0}$$

$$(\theta(1)_2)'(z) = \frac{a3a4\gamma(1)_0\sqrt{\frac{-a4+\gamma(1)_0^2}{\gamma(1)_0^2}}\gamma(1)_1}{\left(-a4+\gamma(1)_0^2\right)^2}$$

$$(\theta(2)_2)'(z) = \frac{a3a4\gamma(2)_0\sqrt{\frac{-a4+\gamma(2)_0^2}{\gamma(2)_0^2}}\gamma(2)_1}{\left(-a4+\gamma(2)_0^2\right)^2}$$

$$(e_2)'(z) =$$

$$\frac{a6\cos(\phi_0+\theta(1)_0)\phi_1\gamma(1)_0\sqrt{\frac{-a4+\gamma(1)_0^2}{\gamma(1)_0^2}}}{2\left(-a4+\gamma(1)_0^2\right)}-\frac{a4\,a6\sin(\phi_0+\theta(1)_0)\sqrt{\frac{-a4+\gamma(1)_0^2}{\gamma(1)_0^2}}\gamma(1)_1}{2\left(-a4+\gamma(1)_0^2\right)^2}-$$

$$\frac{a6\sin(\phi_0+\theta(1)_0)\sqrt{\frac{-a4+\gamma(1)_0^2}{\gamma(1)_0^2}}\gamma(1)_1}{2\left(-a4+\gamma(1)_0^2\right)}+\frac{a6\cos(\phi_0+\theta(2)_0)\phi_1\gamma(2)_0\sqrt{\frac{-a4+\gamma(2)_0^2}{\gamma(2)_0^2}}}{2\left(-a4+\gamma(2)_0^2\right)}-$$

$$\frac{a4\,a6\sin(\phi_0+\theta(2)_0)\sqrt{\frac{-a4+\gamma(2)_0^2}{\gamma(2)_0^2}}\gamma(2)_1}{2\left(-a4+\gamma(2)_0^2\right)^2}-\frac{a6\sin(\phi_0+\theta(2)_0)\sqrt{\frac{-a4+\gamma(2)_0^2}{\gamma(2)_0^2}}\gamma(2)_1}{2\left(-a4+\gamma(2)_0^2\right)}+$$

$$\frac{a6\cos(\phi_0+\theta(1)_0)\gamma(1)_0\sqrt{\frac{-a4+\gamma(1)_0^2}{\gamma(1)_0^2}}\theta(1)_1}{2\left(-a4+\gamma(1)_0^2\right)}+\frac{a6\cos(\phi_0+\theta(2)_0)\gamma(2)_0\sqrt{\frac{-a4+\gamma(2)_0^2}{\gamma(2)_0^2}}\theta(2)_1}{2\left(-a4+\gamma(2)_0^2\right)}$$

$$(\phi_2)'(z) =$$

$$-\frac{\left(a6\cos(\phi_0+\theta(1)_0)e_1\gamma(1)_0\sqrt{\frac{-a4+\gamma(1)_0^2}{\gamma(1)_0^2}}\right)}{2e_0^2\left(-a4+\gamma(1)_0^2\right)}-\frac{a6\phi_1\sin(\phi_0+\theta(1)_0)\gamma(1)_0\sqrt{\frac{-a4+\gamma(1)_0^2}{\gamma(1)_0^2}}}{2e_0\left(-a4+\gamma(1)_0^2\right)}-$$

$$\frac{a4\,a6\cos(\phi_0+\theta(1)_0)\sqrt{\frac{-a4+\gamma(1)_0^2}{\gamma(1)_0^2}}\gamma(1)_1}{2e_0\left(-a4+\gamma(1)_0^2\right)^2}-\frac{a6\cos(\phi_0+\theta(1)_0)\sqrt{\frac{-a4+\gamma(1)_0^2}{\gamma(1)_0^2}}\gamma(1)_1}{2e_0\left(-a4+\gamma(1)_0^2\right)}-$$

$$\frac{a6\cos(\phi_0+\theta(2)_0)e_1\gamma(2)_0\sqrt{\frac{-a4+\gamma(2)_0^2}{\gamma(2)_0^2}}}{2e_0^2\left(-a4+\gamma(2)_0^2\right)}-\frac{a6\phi_1\sin(\phi_0+\theta(2)_0)\gamma(2)_0\sqrt{\frac{-a4+\gamma(2)_0^2}{\gamma(2)_0^2}}}{2e_0\left(-a4+\gamma(2)_0^2\right)}-$$

$$\frac{a4\,a6\cos(\phi_0+\theta(2)_0)\sqrt{\frac{-a4+\gamma(2)_0^2}{\gamma(2)_0^2}}\gamma(2)_1}{2e_0\left(-a4+\gamma(2)_0^2\right)^2}-\frac{a6\cos(\phi_0+\theta(2)_0)\sqrt{\frac{-a4+\gamma(2)_0^2}{\gamma(2)_0^2}}\gamma(2)_1}{2e_0\left(-a4+\gamma(2)_0^2\right)}-$$

$$\frac{a6\sin(\phi_0+\theta(1)_0)\gamma(1)_0\sqrt{\frac{-a4+\gamma(1)_0^2}{\gamma(1)_0^2}}\theta(1)_1}{2e_0\left(-a4+\gamma(1)_0^2\right)}-\frac{a6\sin(\phi_0+\theta(2)_0)\gamma(2)_0\sqrt{\frac{-a4+\gamma(2)_0^2}{\gamma(2)_0^2}}\theta(2)_1}{2e_0\left(-a4+\gamma(2)_0^2\right)}\Bigg\}$$

```
BeginPackage["NLDynamics`"]

Needs["Utilities`FilterOptions`"]

NLDynamics::usage="fonctions permettant de tabuler les resonnnances d'un
                   multipole (couples {mx,my}) , de tracer les
                   diagrammes  de resonance Qx,Qy a un ordre donne ,de
                   calculer le potentiel scalaire d'un multipole et
                   debuter la resolution
                   d'un systeme d'equations differentielles .
                      Contient:Couple,Resonance, multiOrdre,machineDiagr
                      Potential,ODEPerturbation"

Couple::usage="Couple[m,ordre] tabule les couples caracteristiques de la
               resonance d'un multipole m a un ordre donne ,
               ordre 1 par defaut"

Resonance::usage="Resonance[m,ordre,opts] trace les resonances
                    d'un multipole m a un ordre donne ,
                  les options sont {NLColor->Red,NLWindow->{0,1,0,1},NLDisplay->
                  on peut aussi
                  ajouter ou remplacer des options graphiques "

Potential::usage="Potential[m,bx[s],by[s],mux[s],muy[s],jx,jy,phix,phiy]
donne le potentiel scalaire d'un multipole  m."

ODEPerturbation::usage="ODEPertubation[eq_List,iter] resout par perturbation l
                        eq a l'ordre iter ,eq doit avoir la forme
                        {a'[s]==f1[s,a,b,..],b'[s]==f2[s,a,b..],..} "

NLColor::usage=" option de Options[Resonance] "

NLWindow::usage=" option de Options[Resonance] "

NLDisplay::usage="selection"

Single::usage="selection"

Multiple::usage="selection"

Cumulated::usage="selection"

Black::usage="selection"
Blue::usage="selection"
Green::usage="selection"
Cyan::usage="selection"
Red::usage="selection"
Magenta::usage="selection"
Yellow::usage="selection"

Options[Resonance]=  { NLColor -> Red ,NLWindow -> {0,1,0,1},
                       NLDisplay -> Single}

Begin["`Afabrice`"]


 mCouple[m_]:=        (* catalogue des resonances *)
    Block[{i,j,k,l1,l2,list},
       (* couple +  *)
    l1=Table[{m-2(i+j),2(i-k)},{i,0,Floor[m/2]},{j,0,Floor[m/2]-i},{k,0,i}]//
       Partition[Flatten[#],2]&//Union//
       Complement[#,Cases[#,{0,0}]]&    ;
       (* couples - *)
    l2=If[EvenQ[m],
    (Table[{m-2(i+j),-2(i-k)},{i,0,Floor[m/2]},{j,0,Floor[m/2]-i-1},{k,0,i-1}]
     //Partition[Flatten[#],2]&//Union ),




   End[]
 EndPackage[]
```