

# *Approximating Solutions of Linear Ordinary Differential Equations with Periodic Coefficients by Exact Picard Iterates*

**Armando G. M. Neves**

Linear ordinary differential equations (ODEs) with periodic coefficients appear in various interesting applications, such as determining the linear stability regions of systems of vertically driven multiple pendula. Sinha and Butcher [1, 2] have obtained very good approximations to the solutions of such equations by calculating approximate Picard iterates symbolically in the parameters on which the system depends. In this article we show an improvement to the method of Sinha and Butcher. We are able to calculate exact, rather than approximate, Picard iterates of high order. The key point in the programming is the necessity of introducing a user-defined function to carry out the integrations that appear in the definition of the Picard iterates. After introducing the concept of Picard iteration and explaining its fast implementation, we apply the method to determine the stability regions for linearized systems of vertically driven multiple pendula.

## ■ 1. Introduction

Systems of differential equations with periodic coefficients arise in several different applications in physics and engineering. Because of their applications, such systems also deserve attention from pure scientists. One important method for understanding them is linearization around an equilibrium solution. This justifies the attention we devote to systems of linear differential equations with periodic coefficients.

Despite being simpler than nonlinear systems, linearized systems of differential equations in general are not exactly solvable if their coefficients are not constant. It is thus natural to consider approximate solutions for these systems. Besides traditional numerical methods, other kinds of methods, such as perturbation theory [3] and averaging [4], are used. Whereas traditional numerical methods suffer from the problem of being difficult to apply in cases where the equations depend on parameters, perturbation and averaging usually perform well only in some limited regions in the space of parameters.

Picard iteration, described in Section 2, is a well-known method because of its theoretical importance in proving the existence-uniqueness theorem for differential equations [5]. We were delighted and surprised to see it appear in [1] as a practical tool for approximating solutions of differential equations. Furthermore, it can also be used if the equations depend on parameters.

The authors of [1] expand the coefficients matrix of a linear system with periodic coefficients in a series of shifted Chebyshev polynomials. By cleverly using properties of the Chebyshev polynomials, they are able to trade the integrations occurring in Picard iteration by matrix multiplications. As a result, excellent *approximations* for the coefficients of the Chebyshev polynomial series of high-order Picard iterates can be obtained. Their method is thus approximate in a double sense.

First, the infinite sequence of Picard iterates converges to the exact solution, but the Picard iterate at any finite order is only an approximate solution for the differential equations. Second, they do not obtain exact Picard iterates but a truncation of a series converging to them.

While reading [1] we asked ourselves what the difficulty would be if, instead of approximate Picard iterates, we tried to calculate exact ones. Picard iterates are seldom used as an approximate solution method to general differential equations because functions that are impossible to integrate are likely to appear during the iteration. We will see that this is indeed what happens in a simple case, the nonlinear pendulum equation. However, this will not be the case for systems of linear ODEs with periodic coefficients. This assertion will be proved in Section 3. In order to approximately solve these equations, all we then need is a sufficiently fast integration function.

In [6], we showed that it is possible to obtain practical results by calculating *exact* Picard iterates for linear differential equations with periodic coefficients. The number of iterations performed were equal to the ones computed by Sinha and Butcher in [1] and [2]. Although the computers and *Mathematica* versions used are different, the times needed are much shorter with our method. Moreover, exactness in our results leads to important differences in accuracy in the parameter regions considered. In this article we will concentrate on explaining the implementation in detail instead of comparing results.

Our method directly uses definition (2) of the Picard iterate and the specialized integration function we created, which calculates all necessary integrals much faster than by using the built-in function `Integrate`. We provide the package

(described in Section 4) in Additional Material that carefully implements all these ideas.

Section 5 is devoted to the application of the package to the question of *linear* stability of a system of upside-down pendula driven by vertical periodic motion of its suspension point. This kind of system has been considered recently both by pure [7, 8, 9] and applied [1, 2] scientists. Besides being an application of the method devised, it is also a good illustration of the power of *Mathematica*. Not only is it used to approximately solve the differential equations, but also for deducing the equations of motion of the pendula, linearizing them, diagonalizing matrices, and graphing the results.

## ■ 2. Picard Iterates

Let  $t \in \mathbb{R}$  and  $x \equiv (x_1, \dots, x_n) \in \mathbb{R}^n$ . If  $f(t, x)$  is a function with values in  $\mathbb{R}^n$ , then consider the following general initial value problem for a system of  $n$  ordinary differential equations:

$$\begin{cases} \frac{dx}{dt} = f(t, x) \\ x(t_0) = x_0 \end{cases} \quad (1)$$

The Picard iterate of a function  $y(t)$  with values in  $\mathbb{R}^n$  related to the initial condition  $x(t_0) = x_0$  is a new function defined as

$$(T_{x_0} y)(t) = x_0 + \int_{t_0}^t f(s, y(s)) ds. \quad (2)$$

Taking any initial function  $y_0(t)$ , let  $y_1(t) = (T_{x_0} y_0)(t)$ ,  $y_2(t) = (T_{x_0} y_1)(t)$ , .... It can be shown that if  $f$  and  $\frac{\partial f}{\partial x}$  in equation (1) are both continuous functions in a neighborhood of the initial point  $(t_0, x_0)$ , then the sequence of Picard iterates  $y_1(t)$ ,  $y_2(t)$ , ... converges in a neighborhood of  $t_0$  to the solution of the initial value problem (1). This is in fact the usual proof of the existence-uniqueness theorem for differential equations [5]. In general we choose as the initial function  $y_0(t)$  the constant function equal to the initial condition  $x_0$ . This ensures that the very first iterates are good approximations to the solution for  $t$  close enough to  $t_0$ .

Although convergence to the solution of the problem is guaranteed (and exponentially fast in the number of iterates), Picard iteration is seldom used as a method for approximating solutions to differential equations. The problem with it lies in the integration contained in the very definition (2) of the Picard iterate, which may be laborious or even impossible to perform. Take as an example the *pendulum equation*

$$\theta'' + \frac{g}{l} \sin \theta = 0, \quad (3)$$

whose solution cannot be written in terms of elementary functions. We rewrite it as a first-order system of the form (1) by taking  $x(t) = (x_1(t), x_2(t)) = (\theta(t), \theta'(t))$ ,

$f(t, x(t)) = (x_2(t), \frac{-g}{l} \sin x_1(t))$ . A function `onepicard` that performs the operation  $T_{x_0} y$  in equation (2) is defined as follows.

```
In[1]:= onepicard[f_, tzero_, xzero_, y_] :=
      Function[t, xzero + ∫tzerot f[s, y[s]] ds]
```

Both  $f = f(t, x)$ , the function on the right-hand side of the system of differential equations, and  $y = y(t)$ , the function that we want to iterate, should be supplied as pure functions. As we will also be interested in iterating the operator  $T_{x_0}$  over an initial function that is a constant function equal to the initial condition  $x_0$ , we define the function `picard`.

```
In[2]:= picard[f_, tzero_, xzero_, n_Integer] :=
      Nest[onepicard[f, tzero, xzero, #1] &, xzero &, n]
```

Using  $g = l = 1$  to simplify matters, here is the fifth Picard iterate for the pendulum equation with initial condition  $x(0) = (1, 0)$ .

```
In[3]:= picard[{{#2[2]}, -Sin[#2[1]]} &, 0, {1, 0}, 5] [t][[2]]
```

$$Out[3]= \int_0^t -\text{Sin}\left[1 - \text{Cot}[1] + \text{Cos}\left[1 - \frac{1}{2} s^2 \text{Sin}[1]\right] \text{Csc}[1] - s \text{FresnelC}\left[\frac{s}{\sqrt{\pi \text{Csc}[1]}}\right] \sqrt{\pi \text{Sin}[1]} + s \text{Cot}[1] \text{FresnelS}\left[\frac{s}{\sqrt{\pi \text{Csc}[1]}}\right] \sqrt{\pi \text{Sin}[1]}\right] ds$$

Notice that this result is an unevaluated integral! The reader may rework the last input to find that all integrals can be calculated up to the fourth Picard iterate. This shows that high-order Picard iterates may be difficult or impossible to calculate. By comparing the graphs of the fourth Picard iterate and the solution of the same problem by a traditional numerical method, such as the one implemented in `NDSolve`, the reader will also notice that the approximation is not accurate enough in a time interval as small as the period of the pendulum.

### ■ 3. Picard Iteration for Systems of Linear ODEs with Periodic Coefficients

In this section we show that for systems of linear differential equations with periodic coefficients we never encounter integrals that are impossible to calculate. But we will also see that evaluating Picard iterates, even if technically possible, may take too long if we use the built-in `Integrate` command.

More exactly, we will be concerned with systems of the form

$$\frac{dx}{dt} = P(t)x(t), \tag{4}$$

where  $P(t)$  is a matrix satisfying

$$P(t + T) = P(t) \tag{5}$$

for all  $t \in \mathbb{R}$ . As a simplification, we restrict ourselves a bit more by considering the case where  $P(t)$  is a finite sum of terms proportional to sines and cosines of period  $T$ , that is,

$$P(t) = P_0 + \sum_{k=1}^N \left[ P_k^c \cos\left(\frac{2\pi k}{T} t\right) + P_k^s \sin\left(\frac{2\pi k}{T} t\right) \right]. \quad (6)$$

In Section 5 of this article we will see some very interesting examples belonging to this class, but any reasonable periodic matrix with period  $T$  can be cast approximately in the previous form by retaining a finite number of terms in its Fourier series.

An important example is the *Mathieu equation*

$$y'' + (a + b \cos t) y = 0, \quad (7)$$

which can be rewritten in the form (4) by making  $x_1 = y$  and  $x_2 = y'$  and taking

$$P(t) = \begin{pmatrix} 0 & 1 \\ -(a + b \cos t) & 0 \end{pmatrix}.$$

Although the Mathieu equation is exactly solvable with *Mathematica*, we will use it as our prototype example for its simplicity and importance.

`In[4]:= DSolve[y'' [t] + (a + b Cos[t]) y [t] == 0, y [t], t]`

`Out[4]= {{y [t] -> C [1] MathieuC [4 a, -2 b, t/2] + C [2] MathieuS [4 a, -2 b, t/2]}}`

We may also compare our approximate solutions with the exact ones. It should be noted that Mathieu functions, that is, the solutions of the Mathieu equation, are notably difficult to implement with a computer. For this reason they are the subject of recent research [10, 11, 12].

Let us evaluate some Picard iterates approximating the solution of the Mathieu equation, so that we can learn something from them. Taking an initial condition such as  $x(0) = (1, -2)$ , here are the first, second, and third iterates.

`In[5]:= Table [Expand [`

`picard [{{0, 1}, {- (a + b Cos [#1]), 0}}.#2 &, 0, {1, -2}, i] [t]], {i, 3}]`

`Out[5]=`  $\left\{ \left\{ 1 - 2t, -2 - at - b \sin[t] \right\}, \left\{ 1 - b - 2t - \frac{a t^2}{2} + b \cos[t], \right. \right.$   
 $\left. -2 - 2b - at + a t^2 + 2b \cos[t] - b \sin[t] + 2bt \sin[t] \right\},$   
 $\left\{ 1 - b - 2t - 2bt - \frac{a t^2}{2} + \frac{a t^3}{3} + b \cos[t] - 2bt \cos[t] + 4b \sin[t], \right.$   
 $\left. -2 - 2b - at + abt - \frac{b^2 t}{2} + a t^2 + \frac{a^2 t^3}{6} + 2b \cos[t] + \right.$   
 $\left. abt \cos[t] - b \sin[t] - 2ab \sin[t] + b^2 \sin[t] + \right.$   
 $\left. 2bt \sin[t] + \frac{1}{2} abt^2 \sin[t] - \frac{1}{2} b^2 \cos[t] \sin[t] \right\}$

The reader can see that the expanded form of all these iterates is a sum of terms that either are constants, positive integer powers of  $t$ , sines or cosines, or assume

one of the forms  $t^r \cos(qt)$  or  $t^r \sin(qt)$ , where  $r$  is a positive integer. Let us define  $\mathbb{F}$  as the family of the finite linear combinations of all such functions and see whether the next iterate still belongs to the family  $\mathbb{F}$ .

If we calculate the next iterate by hand, we first have to multiply matrix  $P$  by the last element in the previous list and then integrate.

`In[6]:= {{0, 1}, {-(a + b Cos[t]), 0}}.%[[3]]`

$$\text{Out[6]} = \left\{ -2 - 2b - at + abt - \frac{b^2 t}{2} + at^2 + \frac{a^2 t^3}{6} + 2b \cos[t] + abt \cos[t] - b \sin[t] - 2ab \sin[t] + b^2 \sin[t] + 2bt \sin[t] + \frac{1}{2} abt^2 \sin[t] - \frac{1}{2} b^2 \cos[t] \sin[t], (-a - b \cos[t]) \left( 1 - b - 2t - 2bt - \frac{at^2}{2} + \frac{at^3}{3} + b \cos[t] - 2bt \cos[t] + 4b \sin[t] \right) \right\}$$

Keeping in mind the possible appearance of difficult integrals, the main difficulty at this point is the appearance of products involving sines and cosines when we expand the last expression. But we may transform such products into sums of trigonometric functions by using `TrigReduce`, which again brings our expression back to the explicit form of a function in  $\mathbb{F}$ . We still need to integrate this expression. That may be done by using the recursive formulas

$$\int t^r \cos qt dt = \frac{t^r}{q} \sin qt - \frac{r}{q} \int t^{r-1} \sin qt dt \tag{8}$$

$$\int t^r \sin qt dt = -\frac{t^r}{q} \cos qt + \frac{r}{q} \int t^{r-1} \cos qt dt, \tag{9}$$

obtained by an easy integration by parts. We thus see that all integrals of functions in  $\mathbb{F}$  always belong to  $\mathbb{F}$ .

Although we are not interested in giving a formal proof here, all the reasoning with the third iterate can be turned into a proof by induction of the following result.

**Theorem:** Any Picard iterate of linear systems in the form of equation (4) with  $P$  in the form of equation (6) is a function in the family  $\mathbb{F}$ .

Although Picard iteration is thus technically possible at any finite order, as the order grows the resulting expressions become increasingly complicated. As a consequence, the time needed to compute them increases quickly.

`In[7]:= Table[`

`Timing[picard[{{0, 1}, {-(a + b Cos[#1]), 0}}.#2 &, 0, {1, -2}, n][t];][`  
`1], {n, 8}]`

`Out[7]= {0. Second, 0.05 Second, 0.11 Second, 0.39 Second,`  
`0.99 Second, 2.41 Second, 8.02 Second, 17.91 Second}`

Although one might assume that eight Picard iterates are enough, we caution the reader that for the applications we have in mind it will be necessary to evaluate the solution to the Mathieu equation at time  $t = 2\pi$ , that is, at the end of the period of the matrix of the system. To see that eight iterations are too few, we

recommend comparing the graphs of the approximate solution obtained by Picard iteration (substituting  $a$  and  $b$  by typical values 0.9 and 0.6, respectively) and the exact solution given by `DSolve`.

## ■ 4. Fast Implementation of Picard Iteration for Systems of Linear ODEs with Periodic Coefficients and the *FastPicard* Package

By constructing an alternative function, we will see that the main reason for the slowness of `picard` is that it uses the built-in function `Integrate` for integration. `Integrate` is of course very good for general purposes, but it is too slow in cases such as ours, in which we have to calculate a very large number of integrals of functions in a limited class. As shown in the previous section, we do not need all the generality of `Integrate` because all integrands belong to the family  $\mathbb{F}$ .

We created a new function, specialized for definite and indefinite integration of functions in  $\mathbb{F}$ , to be used instead of `Integrate` in the Picard iteration scheme. We called it `NewIntegrate` and gave it the same syntax as `Integrate`. It is implemented in our *FastPicard* package (see Additional Material) and exported by it.

First, `NewIntegrate` recognizes a linear combination of terms and uses the fact that the integral is a linear operator. This is a standard trick used, for example, in the implementation of Laplace transforms [9, 13]. It then readily performs indefinite integrals of constants, powers, sines, and cosines. The more difficult functions in  $\mathbb{F}$  are dealt with by using the recursive formulas of equations (8) and (9). As usual in these cases, we track all results, so that the same integral need not be calculated more than once. This is accomplished by the old trick of using a delayed and an immediate assignment, as explained in Section 2.4.9 of [14].

In calculating definite integrals, a trick for further speed is to use  $\frac{1}{m} - \frac{1}{m} \cos mx$  instead of  $-\frac{1}{m} \cos mx$  as the expression for the indefinite integral of  $\sin mx$ . In most cases, we will be interested in using  $t_0 = 0$  in equation (2). This trick ensures that all definite integrals of functions in  $\mathbb{F}$  with a lower limit equal to zero will coincide with the corresponding indefinite integrals.

The reader is invited to test the accuracy and performance of `NewIntegrate` by producing a large random function in  $\mathbb{F}$  and integrating it with both functions. In our tests, we found that `NewIntegrate` is tens of times faster than the built-in function.

The package also exports the function `FastPicard[matrix, tzero, xzero, n]` that is similar to `picard` but uses `NewIntegrate`. For functions in  $\mathbb{F}$  to be recognized by `NewIntegrate`, it is necessary to transform products of sines and cosines into sums before integration. As we are always dealing with linear systems, the syntax is a bit simplified—instead of providing the function  $f(t, x)$ , the reader should provide matrix  $P(t)$ .

Of course, times are much shorter than the ones obtained by `picard`. This is shown in the following example, which also illustrates the syntax.

```
In[8]:= << FastPicard.m
In[9]:= Table[
  Timing[FastPicard[{{0, 1}, {-a - b Cos[#1], 0}} &, 0, {1, -2}, n][t];],
  {n, 10}]
Out[9]= {{0. Second, Null}, {0. Second, Null},
  {0. Second, Null}, {0. Second, Null}, {0.05 Second, Null},
  {0.06 Second, Null}, {0.11 Second, Null},
  {0.16 Second, Null}, {0.22 Second, Null}, {0.44 Second, Null}}
```

Besides using `NewIntegrate`, *FastPicard* incorporates other improvements we discovered after some experimenting. Here is a list of the differences.

- In order to avoid calculating the same integral more than once, similar terms are gathered before they are acted on by `NewIntegrate`. The gathering is performed by an auxiliary function named `gather`, not exported by the package, and is responsible for part of the speed increase when the number of iterates is large enough to compensate for the extra work of gathering.
- To our surprise, we found that in Version 4.1 the built-in function `TrigReduce` is also slower than a set of transformation rules designed to transform products of sines and cosines into sums. We implemented such rules in *FastPicard*.
- As we are mostly interested in Picard iterates with  $t_0 = 0$ , we made the second argument of `FastPicard` optional with a default value of 0. Instead of calculating definite integrals, in this case we may replace them by indefinite ones that evaluate much faster.

The package also exports the functions `FundamentalMatrix` and `FloquetMatrix`.

A *fundamental matrix solution* to a linear system such as equation (4) is a matrix whose columns are solutions to the system with linear independent initial conditions. The function `FundamentalMatrix[matrix, tzero, n]` calculates  $n$  Picard iterates approximating the particular fundamental matrix solution that takes the columns of the identity matrix as initial conditions at time `tzero`. The second argument `tzero` is optional with a default value of 0. For example, with 20 iterates and initial time 0, an approximate fundamental matrix solution to the system with

$$P(t) = \begin{pmatrix} -1 + a \cos^2 t & 1 - a \sin t \cos t \\ -1 - a \sin t \cos t & -1 + a \sin^2 t \end{pmatrix}$$

is obtained by

```
In[10]:= p[t_] = TrigReduce[{{-1 + a Cos[t]^2, 1 - a Sin[t] Cos[t]},
  {-1 - a Sin[t] Cos[t], -1 + a Sin[t]^2}}];
```

```
In[11]:= fms = FundamentalMatrix[p, 20][t];
```

One advantage of having a fundamental matrix solution calculated at initial time  $t_0$  is that with it we may recover solutions to the system with any desired initial conditions at time  $t_0$ . This is even simpler when we use the particular fundamental matrix solution calculated by `FundamentalMatrix`. For example, this gives the approximate numerical value for the solution of this system with  $a = 2$  at time  $t = 2\pi$  and initial condition  $y(0) = (1, -2)$ .

```
In[12]:= ((fms /. {a -> 2., t -> 2 Pi}).{1, -2})
```

```
Out[12]= {535.492, -0.00373489}
```

These values can be compared with the corresponding exact ones. In fact, the exact fundamental matrix solution for the system is given in [1]. Using it, we obtain

```
In[13]:= N[{{e^(a-1) t Cos[t], e^-t Sin[t]}, {-e^(a-1) t Sin[t], e^-t Cos[t]}} /. {a -> 2, t -> 2 Pi} /. {1, -2} /. {a -> 2, t -> 2 Pi}]
```

```
Out[13]= {535.492, -0.00373489}
```

The *Floquet transition matrix* for a linear system of differential equations with periodic coefficients of period  $T$  is just the special fundamental matrix solution calculated by `FundamentalMatrix` with initial time 0 evaluated at time  $T$ . In the next section we will see an interesting application for this matrix. An approximation to it using  $n$  Picard iterates is calculated by `FloquetMatrix[matrix, per, n]`, where the second argument, the period of the matrix, is optional with a default value of  $2\pi$ .

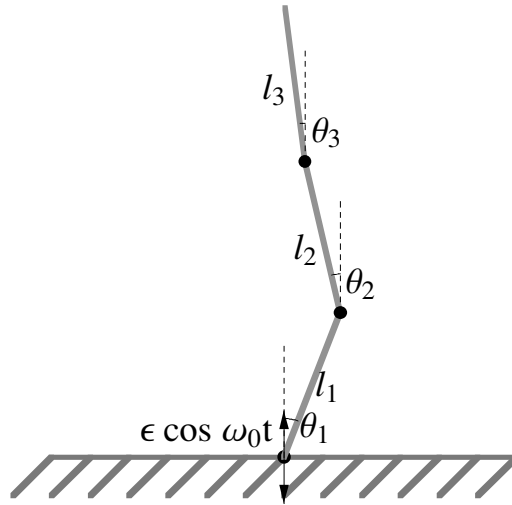
For example, here is an approximation to the Floquet transition matrix for the Mathieu equation with  $a = -0.1$  and  $b = 0.55$ .

```
In[14]:= ftmmathieu = FloquetMatrix[{{0, 1}, {-a - b Cos[#1], 0}} &, 20];
ftmmathieu /. {a -> -0.1, b -> 0.55}
```

```
Out[15]= {{-0.0397796, 18.6577}, {-0.0535123, -0.0397796}}
```

## ■ 5. Application to Linearized Systems of Driven Pendula

Consider a system of  $N$  upside-down physical pendula (homogeneous rods, in fact) such as in the following graphic, where we depict the case  $N = 3$ . It is assumed that each rod may rotate without friction around its bearing and that gravity points down.



It is a fact of everyday experience that the upside-down equilibrium position in which all rods are fixed at the vertical position  $\theta_i = 0, i = 1, 2, \dots, N$  is highly unstable.

Curiously, however, if the support point of the first rod is driven by an external force such that it executes a vertical sinusoidal motion of amplitude  $\epsilon$  and angular frequency  $\omega_0$ , and if  $\epsilon$  and  $\omega_0$  are chosen from a suitable range, then the upside-down equilibrium becomes stable. This fact seems to have been first predicted in 1908 by Stephenson [15] for a single rod and subsequently extended by him to the case of two and three rods in 1909 [16]. Interest in such systems was aroused more recently by Acheson, who proved in [7] that if the nonlinear equations of motion for the rods are linearized in the neighborhood of the equilibrium solution, then, for any value of  $N$ , there exists a stability region for this system of *linearized* upside-down rods. In other words, there exists a range in  $\epsilon$  and  $\omega_0$  where the upside-down equilibrium for the linearized equations is stable. If the values of the angles  $\theta_i$  and their derivatives are small, one may hope that the solution of the linearized equations is close to the solution of the nonlinear equations, but a mathematical proof of stability for nonlinear rods is much more difficult [9]. In fact, even an experimental proof of stability for systems of one, two, and three rods has merited a good deal of attention recently [8]. Since then, the number of papers in this field has been very large and even outside of the physics and mathematics communities. Stability of upside-down rods (or pendula) have attracted interest in areas ranging from control engineering to biomechanics. A good reference from the mathematical physics point of view with many further references is [9].

### □ The Lagrangian

In this subsection we use the Lagrangian formalism of classical mechanics [17] to deduce the equations of motion for *undamped* systems of periodically driven

physical pendula and then linearize them around the upside-down equilibrium position. The Euler–Lagrange equations of motion are

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}'_i} - \frac{\partial L}{\partial q_i} = 0, \quad i = 1, 2, \dots, N, \quad (10)$$

where the Lagrangian function  $L$  is the difference between kinetic and potential energies.

Consider a system such as the one in the previous graphic, where the  $i$ th pendulum has length  $l_i$  and mass  $m_i$ . Suppose that the mass of the  $i$ th pendulum is distributed such that its center of mass is located at a distance  $H_i$  from its bottom extremity and its inertia moment through an axis passing through the bottom extremity and orthogonal to the plane of motion is  $\frac{1}{2} m_i R_i^2$ , that is, the gyration radius is  $R_i$ . Suppose also that the support point of the lower pendulum is externally driven so that its position vector is  $\vec{r}_1(t) = (\xi(t), \eta(t))$ . We will denote with  $\vec{v}_i(t)$  the velocity vector of the bottom extremity of the  $i$ th pendulum with respect to the inertial lab frame. The velocity vector of the top of pendulum  $i$  with respect to the noninertial frame of its bottom will be denoted with  $\vec{w}_i(t)$ . Using well-known expressions for  $K_i$  and  $V_i$ , the kinetic and gravitational potential energies of a rigid body, and  $g$  for the acceleration of gravity, we obtain as follows the Lagrangian  $L$  for the system.

$$\begin{aligned} \text{In[16]:= } & \mathbf{v}[1] := \partial_t \{ \xi[t], \eta[t] \}; \\ & \mathbf{w}[i\_ ] := \partial_t \{ l[i] \text{Sin}[\theta[i][t]], l[i] \text{Cos}[\theta[i][t]] \}; \\ & \mathbf{v}[i\_ ] := \mathbf{v}[i-1] + \mathbf{w}[i-1]; \\ & \mathbf{K}[i\_ ] := \\ & \quad \frac{1}{2} m[i] \mathbf{v}[i] \cdot \mathbf{v}[i] + \frac{1}{2} m[i] R[i]^2 (\partial_t \theta[i][t])^2 + \frac{m[i] H[i] \mathbf{v}[i] \cdot \mathbf{w}[i]}{l[i]}; \\ & \mathbf{V}[i\_ ] := m[i] g \left( \sum_{k=1}^{i-1} l[k] \text{Cos}[\theta[k][t]] + H[i] \text{Cos}[\theta[i][t]] \right); \\ & \mathbf{L}[i\_ ] := \sum_{j=1}^i (\mathbf{K}[j] - \mathbf{V}[j]) \end{aligned}$$

$L[n]$  will produce the Lagrangian for a system of  $n$  pendula.

## □ A Single Rod

Here is the Lagrangian for a single upside-down driven rod, for example.

$$\begin{aligned} \text{In[22]:= } & \mathbf{L}[1] \\ \text{Out[22]= } & -g \text{Cos}[\theta[1][t]] H[1] m[1] + \\ & \frac{1}{2} m[1] (\eta'[t]^2 + \xi'[t]^2) + \frac{1}{2} m[1] R[1]^2 \theta[1]'[t]^2 + \\ & \frac{1}{l[1]} (H[1] m[1] (-l[1] \text{Sin}[\theta[1][t]] \eta'[t] \theta[1]'[t] + \\ & \quad \text{Cos}[\theta[1][t]] l[1] \xi'[t] \theta[1]'[t])) \end{aligned}$$

From it we can evaluate the left-hand side of equation (10).

$$\text{In[23]:= lhs[1] = } \partial_t (\partial_{\theta[1][t]} L[1]) - \partial_{\theta[1][t]} L[1];$$

For a rod with uniform mass distribution ( $H_1 = \frac{l}{2}$ ,  $R_1 = \frac{l}{\sqrt{3}}$ ) and sinusoidal (in fact, cosinusoidal) vertical driving, the equation of motion is

$$\text{In[24]:= Simplify[lhs[1] /. {m[1] } \rightarrow m, l[1] \rightarrow l, H[1] \rightarrow \frac{l}{2},$$

$$R[1] \rightarrow \frac{l}{\sqrt{3}}, \xi[t] \rightarrow 0, \xi'[t] \rightarrow 0, \xi''[t] \rightarrow 0, \eta[t] \rightarrow \epsilon \text{Cos}[\omega_0 t],$$

$$\eta'[t] \rightarrow \partial_t (\epsilon \text{Cos}[\omega_0 t]), \eta''[t] \rightarrow \partial_{\{t,2\}} (\epsilon \text{Cos}[\omega_0 t])}]$$

$$\text{Out[24]= } \frac{1}{6} l m (-3 g \text{Sin}[\theta[1][t]] + 3 \epsilon \text{Cos}[t \omega_0] \text{Sin}[\theta[1][t]] \omega_0^2 + 2 l \theta[1]''[t])$$

Here linearization is substituting  $\sin \theta$  by  $\theta$ .

$$\text{In[25]:= Collect[Expand[\frac{3}{m l^2} \theta /. Sin[\theta[1][t]] } \rightarrow \theta[1][t]], \theta[1][t]]$$

$$\text{Out[25]= } \left(-\frac{3 g}{2 l} + \frac{3 \epsilon \text{Cos}[t \omega_0] \omega_0^2}{2 l}\right) \theta[1][t] + \theta[1]''[t]$$

It is possible to put the equation of motion in a simpler form, depending on a smaller number of parameters, if we rescale the time variable defining

$$\tau = \omega_0 t, \tag{11}$$

the new form assumed by the equation is then

$$\theta''(\tau) + \left(-\frac{3 g}{2 \omega_0^2 l} + \frac{3 \epsilon}{2 l} \cos \tau\right) \theta(\tau) = 0, \tag{12}$$

which is the Mathieu equation (7) with

$$a = -\frac{3 g}{2 \omega_0^2 l} \tag{13}$$

and

$$b = \frac{3 \epsilon}{2 l}. \tag{14}$$

So, the question of stability of an upside-down vertically driven single rod amounts to stability of the solutions of the Mathieu equation. This has been studied for a long time [18].

The Floquet theory [18] for linear differential equations with periodic coefficients examines, among other things, the stability of equilibrium solutions. It is proved that the equilibrium solution  $x(t) = 0$  for a linear system with periodic coefficients is stable if and only if all eigenvalues  $\lambda_i$  of the Floquet transition matrix (FTM) are such that  $|\lambda_i| \leq 1$ .

In the particular case of Mathieu equations, one proves that the transition from stability to instability occurs when both eigenvalues are equal to 1 or both equal to  $-1$ . In other words, the curves in the  $a b$  plane where solutions to the Mathieu

equation change from stable to unstable or vice versa are given by the condition that

$$\text{Tr FTM} = \pm 2. \quad (15)$$

Before we use this condition to produce a plot of the stability regions for a single inverted rod, let us make a numerical comparison of our results with exact ones. We will use the approximation `ftmmathieu` to the FTM calculated previously with 20 iterates and choose two values of  $b$ ,  $b = 0.1$  and  $b = 1.5$ , in order to find the values of  $a$  corresponding to the stability boundaries at these values of  $b$ . We do that as follows.

```
In[26]:= With[{trftm = Expand[Tr[ftmmathieu /. b -> .1]]},
  Sort[Select[Join[a /. NSolve[trftm == -2, a],
    a /. NSolve[trftm == 2, a]], Im[#1] == 0 &]]]
```

```
Out[26]:= {-0.00497832, 0.198781, 0.298719, 0.993783, 1.01023, 2.2429}
```

```
In[27]:= With[{trftm = Expand[Tr[ftmmathieu /. b -> 1.5]]},
  Sort[Select[Join[a /. NSolve[trftm == -2, a],
    a /. NSolve[trftm == 2, a]], Im[#1] == 0 &]]]
```

```
Out[27]:= {-0.708598, -0.696345, 0.62976, 0.81923, 1.51241, 2.3711}
```

For the sake of comparison, the exact values corresponding to the approximate ones, are related to the so-called Mathieu characteristic numbers implemented in *Mathematica*.

```
In[28]:= Sort[ $\frac{1}{4}$  Join[Table[MathieuCharacteristicA[r, -2 0.1], {r, 0, 3}],
  Table[MathieuCharacteristicB[r, -2 0.1], {r, 2}]]]
```

```
Out[28]:= {-0.00497832, 0.198781, 0.298719, 0.999167, 1.00414, 2.25059}
```

```
In[29]:= Sort[ $\frac{1}{4}$  Join[Table[MathieuCharacteristicA[r, -2 1.5], {r, 0, 3}],
  Table[MathieuCharacteristicB[r, -2 1.5], {r, 2}]]]
```

```
Out[29]:= {-0.708598, -0.696345, 0.62976, 0.81923, 1.5113, 2.30578}
```

Instead of plotting the curves where equation (15) holds in variables  $a$  and  $b$ , it is interesting to use physical variables related to the pendula. So we define  $b$  (dimensionless amplitude) and  $v$  (dimensionless frequency) as

$$b = \frac{\epsilon}{l} \quad (16)$$

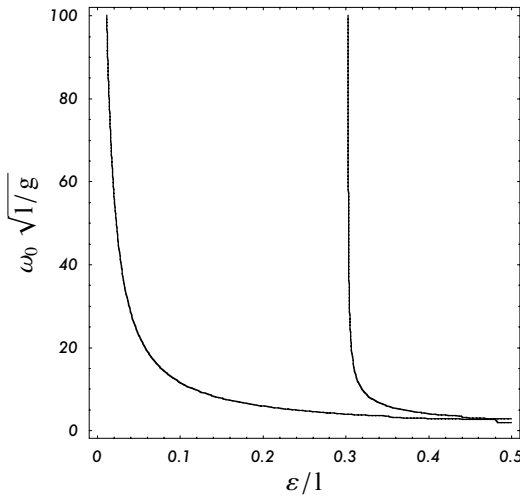
and

$$v = \omega_0 \sqrt{\frac{l}{g}}. \quad (17)$$

The relationship between these variables and  $a$  and  $b$  is  $a = -\frac{3}{2v^2}$ ,  $b = \frac{3}{2}b$ . In order to get a good graphic, we need to use a large value for the option `Plot`:

Points of ContourPlot. It takes a reasonable time to run, but the resulting graph is as follows.

```
In[30]:= Timing[pend1 =
With[{matrix = ftmmathieu /. {a -> - 3
/ (2 v^2), b -> 3 h
/ 2}}, ContourPlot[
Tr[matrix], {h, 0, 0.5}, {v, .1, 100}, Contours -> {-2, 2},
ContourShading -> False, PlotPoints -> 120, FrameLabel ->
{StyleForm["\(\(\epsilon/1\)\)", "TR", FontSize -> 10], StyleForm[
"\(\(\omega_0\)\)\(\(1/g\)\)\)", "TR", FontSize -> 10}]]];
```



```
Out[30]= {118.15 Second, Null}
```

We claim that the inverted rod is stable if the frequency and amplitude of the driving are such that the corresponding point lies in the region between the two curves. To see that, we choose an arbitrary point in this region and check stability by computing the corresponding eigenvalues of the FTM. For example, the point (0.2, 40) in  $(b, v)$  coordinates lies in the mentioned region. The absolute values of the eigenvalues of the FTM are thus.

```
In[31]:= Abs[Eigenvalues[ftmmathieu /. {a -> - 3
/ (2 40^2), b -> 3 0.2
/ 2}]]]
```

```
Out[31]= {1., 1.}
```

This confirms our claim! We see that for reasonably small amplitudes, such as  $\frac{\epsilon}{l} < 0.3$ , the inverted rod is stable (in the linear approximation) if the frequency is high enough. For larger amplitudes, the inverted rod is stable only in a precise range of frequencies.

## □ Two Rods

The left-hand sides of the Euler–Lagrange equations of motion for a system of two pendula, restricted to the simpler case of equal masses, equal lengths, and uniform mass distributions follow.

$$\text{In[32]:= nlefts = Simplify}\left[\begin{aligned} & \left\{ \partial_t (\partial_{\theta[1]'} [t] L[2]) - \partial_{\theta[1]} [t] L[2], \partial_t (\partial_{\theta[2]'} [t] L[2]) - \partial_{\theta[2]} [t] L[2] \right\} / \\ & \left\{ m[i\_]\rightarrow m, l[i\_]\rightarrow l, H[i\_]\rightarrow \frac{1}{2}, R[i\_]\rightarrow \frac{1}{\sqrt{3}} \right\} \end{aligned}\right]$$

$$\text{Out[32]= } \left\{ \begin{aligned} & \frac{1}{6} m (-9 g \sin[\theta[1][t]] + 3 l \sin[\theta[1][t] - \theta[2][t]] \theta[2]'[t]^2 - \\ & 9 \sin[\theta[1][t]] \eta''[t] + 9 \cos[\theta[1][t]] \xi''[t] + \\ & 8 l \theta[1]''[t] + 3 l \cos[\theta[1][t] - \theta[2][t]] \theta[2]''[t]), \\ & \frac{1}{6} m (-3 g \sin[\theta[2][t]] - 3 l \sin[\theta[1][t] - \theta[2][t]] \theta[1]'[t]^2 - \\ & 3 \sin[\theta[2][t]] \eta''[t] + 3 \cos[\theta[2][t]] \xi''[t] + \\ & 3 l \cos[\theta[1][t] - \theta[2][t]] \theta[1]''[t] + 2 l \theta[2]''[t]) \end{aligned} \right\}$$

Notice the presence of several nonlinear terms. Again, the first step in linearizing the equations around the equilibrium  $\theta_1 = 0$ ,  $\theta_2 = 0$  is to replace sines and cosines by their Taylor expansions, retaining terms of at most order 1. But now, this does not eliminate all nonlinearities. In order to finish the linearization procedure, we first create a function to detect the degree of a monomial in a given variable and a Boolean function that tests for monomials of degree not larger than 1.

$$\text{In[33]:= deg}[c_, var_] := 0 /; FreeQ[c, var];$$

$$\text{deg}[c_. var_<sup>n</sup>., var_] := n /; FreeQ[c, var]$$

$$\text{In[35]:= notlargerthan1}[x_, listvars_List] :=$$

$$\text{Plus}@@(\text{deg}[x, \#1] \&) /@ listvars \leq 1$$

Then we use it to select only the terms of degree not larger than 1 in all variables  $\theta_1$ ,  $\theta_2$ ,  $\theta_1'$ ,  $\theta_2'$ ,  $\theta_1''$ ,  $\theta_2''$ .

$$\text{In[36]:= lhs} = \frac{1}{m l^2}$$

$$\left( (\text{Select}[\#1, \text{notlargerthan1}[\#1, \{\theta[1][t], \theta[2][t], \theta[1]'[t], \theta[2]'[t], \theta[1]''[t], \theta[2]''[t]\} \&]) \&) /@ \right.$$

$$\left. \text{Expand}[nlefts /. \{\sin[x_] \rightarrow x, \cos[x_] \rightarrow 1\}] \right)$$

$$\text{Out[36]= } \left\{ \begin{aligned} & \frac{1}{l^2 m} \left( -\frac{3}{2} g l m \theta[1][t] - \frac{3}{2} l m \theta[1][t] \eta''[t] + \right. \\ & \left. \frac{3}{2} l m \xi''[t] + \frac{4}{3} l^2 m \theta[1]''[t] + \frac{1}{2} l^2 m \theta[2]''[t] \right), \\ & \frac{1}{l^2 m} \left( -\frac{1}{2} g l m \theta[2][t] - \frac{1}{2} l m \theta[2][t] \eta''[t] + \frac{1}{2} l m \xi''[t] + \right. \\ & \left. \frac{1}{2} l^2 m \theta[1]''[t] + \frac{1}{3} l^2 m \theta[2]''[t] \right) \end{aligned} \right\}$$

We would like to write the previous linear expression in matrix form. In order to obtain the matrices, we use the command `LinearExpressionToMatrix` in the `Developer`` context (or, equivalently, transform the expression into a system of equations and then use the `LinearEquationstoMatrices` command in the standard package `LinearAlgebra`MatrixManipulation``).

`In[37]:= << Developer``

The matrix that serves as the coefficient for the vector  $\theta''(t) = (\theta_1''(t), \theta_2''(t))$  is obtained as

`In[38]:= A = LinearExpressionToMatrix[lhs, {θ[1]'' [t], θ[2]'' [t]}][[1]]`

`Out[38]=`  $\left\{ \left\{ \frac{4}{3}, \frac{1}{2} \right\}, \left\{ \frac{1}{2}, \frac{1}{3} \right\} \right\}$

The coefficient of the vector  $\theta(t) = (\theta_1(t), \theta_2(t))$  is

`In[39]:= LinearExpressionToMatrix[lhs, {θ[1] [t], θ[2] [t]}][[1]]`

`Out[39]=`  $\left\{ \left\{ -\frac{3g}{2l} - \frac{3\eta'' [t]}{2l}, 0 \right\}, \left\{ 0, -\frac{g}{2l} - \frac{\eta'' [t]}{2l} \right\} \right\}$

and the remaining terms are

`In[40]:= remainder = Simplify[lhs - (A . {θ[1]'' [t], θ[2]'' [t]} + % . {θ[1] [t], θ[2] [t]})]`

`Out[40]=`  $\left\{ \frac{3\xi'' [t]}{2l}, \frac{\xi'' [t]}{2l} \right\}$

Notice that the remainder vanishes in the case of vertical driving, that is,  $\xi(t) = 0$ . Using the cosine vertical driving  $\eta(t) = \epsilon \cos \omega_0 t$ , the term proportional to  $\theta(t)$  becomes

`In[41]:= %% /. η'' [t] → ∂t,2 (ε Cos[ω0 t])`

`Out[41]=`  $\left\{ \left\{ -\frac{3g}{2l} + \frac{3\epsilon \text{Cos}[t \omega_0] \omega_0^2}{2l}, 0 \right\}, \left\{ 0, -\frac{g}{2l} + \frac{\epsilon \text{Cos}[t \omega_0] \omega_0^2}{2l} \right\} \right\}$

and again rescaling the time as in equation (11), we arrive finally at

$$A\theta''(\tau) + \left(-\frac{g}{\omega_0^2 l} + \frac{\epsilon}{l} \cos \tau\right)B\theta(\tau) = 0, \tag{18}$$

where  $A$  was previously obtained as

$$A = \begin{pmatrix} 4/3 & 1/2 \\ 1/2 & 1/3 \end{pmatrix} \tag{19}$$

and  $B$  is the diagonal matrix

$$B = \begin{pmatrix} 3/2 & 0 \\ 0 & 1/2 \end{pmatrix}. \tag{20}$$

$$\text{In[42]:= B = Simplify}\left[\frac{\%}{-\frac{g}{1} + \frac{\epsilon \text{Cos}[\omega_0 \tau] \omega_0^2}{1}}\right]$$

$$\text{Out[42]= } \left\{\left\{\frac{3}{2}, 0\right\}, \left\{0, \frac{1}{2}\right\}\right\}$$

Multiplying both sides of equation (18) on the left by  $A^{-1}$  and realizing that

$$\text{In[43]:= Inverse[A] . B}$$

$$\text{Out[43]= } \left\{\left\{\frac{18}{7}, -\frac{9}{7}\right\}, \left\{-\frac{27}{7}, \frac{24}{7}\right\}\right\}$$

we can rewrite it into the form of equation (4) by making  $x_1 = \theta_1$ ,  $x_2 = \theta_2$ ,  $x_3 = \theta'_1$ ,  $x_4 = \theta'_2$ , with the matrix  $P(\tau)$  in equation (4) being

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{18}{7} \left(-\frac{g}{\omega_0^2 l} + \frac{\epsilon}{l} \cos \tau\right) & \frac{9}{7} \left(-\frac{g}{\omega_0^2 l} + \frac{\epsilon}{l} \cos \tau\right) & 0 & 0 \\ \frac{27}{7} \left(-\frac{g}{\omega_0^2 l} + \frac{\epsilon}{l} \cos \tau\right) & -\frac{24}{7} \left(-\frac{g}{\omega_0^2 l} + \frac{\epsilon}{l} \cos \tau\right) & 0 & 0 \end{pmatrix}$$

and obeying equation (5) with  $T = 2\pi$ .

At this point, we can calculate the twentieth approximation to the FTM as

$$\text{In[44]:= ftmrod2 = FloquetMatrix[} \\ \{ \{0, 0, 1, 0\}, \{0, 0, 0, 1\}, \{-18(a + b \text{Cos}[\#1]), 9(a + b \text{Cos}[\#1]), \\ 0, 0\}, \{27(a + b \text{Cos}[\#1]), -24(a + b \text{Cos}[\#1]), 0, 0\} \} \&, 20];$$

where we should have in mind that

$$a = -\frac{g}{7\omega_0^2 l} \equiv -\frac{1}{7v^2}, \\ b = \frac{\epsilon}{7l} \equiv \frac{h}{7},$$

and the dimensionless variables  $h$  and  $v$  are the same as defined in equations (16) and (17).

$$\text{In[45]:= ftmrod2 = ftmrod2 /. \{a \to -\frac{1}{7v^2}, b \to \frac{h}{7}\};$$

In order to produce a picture of the stability boundaries for the double rod in the dimensionless variables, in analogy with what was done in the case of one rod, we must locate the points where (in absolute value) the maximum eigenvalue of the Floquet matrix changes from greater than one to equal to one. In the case of the single rod, this was conveniently accomplished by the trace condition of equation (15), but that does not hold anymore.

As we do not know any condition similar to equation (15) to use instead, the best we can do is locate some points in the stability boundaries. We may do that by using an idea similar to the *bisection method* for solving equations. Whenever the

largest eigenvalue (in absolute value) of the FTM is greater than one at some point  $P$  and equal to one at another point  $Q$ , then there must exist a point between  $P$  and  $Q$  through which a stability boundary passes. We may approximate this point by taking the point  $R$  halfway between  $P$  and  $Q$  (hence the name bisection) and calculating at  $R$  the maximum eigenvalue of the FTM in absolute value. If this number is equal to one, then the stability boundary must pass between  $P$  and  $R$ , otherwise it must pass between  $R$  and  $Q$ . By repeating the bisection procedure a sufficient number of times we may approximate a point in the stability boundary between  $P$  and  $Q$  with an error smaller than any prescribed tolerance.

By experience we know that the stability boundary lies in the region  $0 \leq b \leq 0.10$ ,  $0 < v \leq 100$ . We then produce a grid of points in that region and calculate the largest eigenvalue of the FTM in absolute value at each point in the grid.

```
In[46]:= pointgrid = Map[{{#1[[1]], #1[[2]]},
    Max[Abs[Eigenvalues[ftmrod2 /. {h -> #1[[1]], v -> #1[[2]]}]]]} &,
    Table[{x, y}, {x, 0, 0.10, .01}, {y, 0.10, 100, 10}], {2}];
```

In order to select those pairs of points in the grid between which we know that the stability boundary passes, we create the following Boolean function and use it.

```
In[47]:= locstabchange[{{{_, _}, m1_}, {{{_, _}, m2_}}] :=
    Chop[m1 - 1] == 0 && Chop[m2 - 1] > 0 || Chop[m2 - 1] == 0 && Chop[m1 - 1] > 0
```

```
In[48]:= Short[verticalpairs = Flatten[(Select[#1, locstabchange] &) /@
    (Partition[#1, 2, 1] &) /@ pointgrid, 1], 5]
```

```
Out[48]//Short= {{{{0.02, 80.1}, 1.01665}, {{0.02, 90.1}, 1.}},
    {{{0.03, 50.1}, 1.04569}, {{0.03, 60.1}, 1.}},
    << 6 >>, {{{0.09, 20.1}, 1.}, {{0.09, 30.1}, 1.53615}}}
```

Notice that because of the particular way `pointgrid` was produced, we have selected pairs of points in the grid with the same horizontal coordinate, that is, points on vertical lines. Later we will repeat the procedure for selecting points lying on horizontal lines.

We can now create the function `bisect` that takes each pair of points in the output of the previous line and calculates the midpoint of the pair, the largest eigenvalue in absolute value of the FTM at the midpoint, and then selects from among the three points the two between which the stability boundary must lie.

```
In[49]:= bisect[{{fir : {{x1_, y1_}, m1_}, sec : {{x2_, y2_}, m2_}}] :=
    Module[{mid =  $\frac{1}{2}$  (fir[[1]] + sec[[1]]), eigenmid},
    eigenmid = Max[Abs[Eigenvalues[ftmrod2 /. {h -> mid[[1]], v -> mid[[2]]}]]];
    Select[{{fir, {mid, eigenmid}}, {mid, eigenmid}, sec}],
    locstabchange][[1]]
```

If we want to locate the point in the stability boundary between each pair of selected points, it is necessary to iterate `bisect` the right number of times. If the tolerance `tol` is specified, that can be done by the following function.

```

In[50]:= findbybisection[{fir_, sec_}, tol_] :=
  With[{bis = Nest[bisect, {fir, sec},
    Max[1, Ceiling[Log[2,  $\frac{\text{Max}[\text{Abs}[\text{fir}[[1]] - \text{sec}[[1]]]}{\text{tol}}$ ]]]]}],
     $\frac{1}{2} (\text{bis}[[1, 1]] + \text{bis}[[2, 1]])$ 

```

So, this finds with tolerance 0.1 (in the vertical direction) some points in the stability boundaries.

```

In[51]:= stabpoints1 = (findbybisection[#1, .1] &) /@ verticalpairs;

```

We may now repeat the procedure using `Transpose[pointgrid]` instead of `pointgrid`. That will produce pairs of points lying on horizontal lines between which some point in the stability boundaries must exist. Of course, as the scale of the horizontal axes in the picture is smaller, we must accordingly reduce the tolerance `tol`.

```

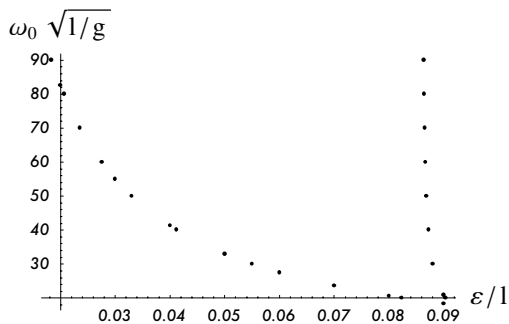
In[52]:= horizontalpairs = Flatten[(Select[#1, locstabchange] &) /@
  (Partition[#1, 2, 1] &) /@ Transpose[pointgrid], 1];
stabpoints2 = (findbybisection[#1, .0001] &) /@ horizontalpairs;

```

```

In[54]:= stabpointspicture =
  ListPlot[Join[stabpoints1, stabpoints2], PlotStyle -> PointSize[.01],
  AxesLabel -> {StyleForm["\(\(\epsilon/1\)\)", "TR", FontSize -> 10],
  StyleForm["\(\(\omega_0\)\)\(\@(\(1/g\)\)\)", "TR", FontSize -> 10]}]

```



An alternative exists for generating this graph: we may use our Mathieu equation results together with the introduction of *normal coordinates* that decouple equation (18) into two separate Mathieu equations. These normal coordinates, which we now briefly explain, were used by Acheson in the proof of his pendulum theorem [7].

Notice that as the matrix  $A^{-1} B$  has two distinct real eigenvalues, then it must be diagonalizable over the real numbers. Let  $U$  be the orthogonal matrix that diagonalizes  $A^{-1} B$ . We define the *normal coordinates*  $\psi$  as  $\psi = U^{-1} \theta$ . Substituting  $\theta$  by  $U \psi$  in equation (18) and then left multiplying it by  $U^{-1} A^{-1}$ , we obtain

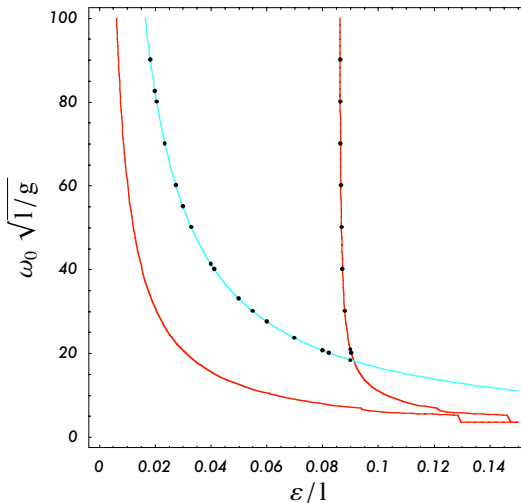
$$\psi''(\tau) + U^{-1} A^{-1} B U \left( -\frac{g}{\omega_0^2 l} + \frac{\epsilon}{l} \cos \tau \right) \psi(\tau) = 0,$$

which, because  $U^{-1} A^{-1} B U$  is diagonal, decouples into

$$\psi_i''(\tau) + \mu_i \left( -\frac{g}{\omega_0^2 l} + \frac{\epsilon}{l} \cos \tau \right) \psi_i(\tau) = 0, \tag{21}$$

where  $i = 1, 2$  and  $\mu_i$  are the eigenvalues of  $A^{-1} B$ . These are two Mathieu equations (7) with  $a = -\frac{g}{\omega_0^2 l} \mu_i \equiv -\frac{\mu_i}{\omega^2}$  and  $b = \frac{\epsilon}{l} \mu_i \equiv \mu_i b$ .

As the original variables  $\theta_1, \theta_2$  are linear combinations of the normal variables  $\psi_1, \psi_2$ , then the equilibrium solution of equation (18) will be stable if and only if the equilibrium solution for both equations (21),  $i = 1, 2$  is stable. We may then determine the stability region for the linearized double rod in equation (18) by intersecting the two stability regions for Mathieu equations. This can be done by using the already calculated approximate FTM for the Mathieu equation. In the following result, the red curve is the stability picture for the mode  $i = 1$ , the cyan curve corresponds to the mode  $i = 2$ , and the points over the curves are the ones calculated previously. Notice that both ways of producing the picture lead to indistinguishable results!



In order to check the numerical accuracy of one of the points in the picture, we may use the built-in Mathieu characteristic numbers together with the normal coordinates method.

```
In[55]:= stabpoints1[[5]]
```

```
Out[55]:= {0.06, 27.5609}
```

Given that the first coordinate is 0.06, here is the exact value for the second coordinate of the chosen point.

```
In[56]:= With[{eigenval = Min[Eigenvalues[Inverse[A].B]}],
```

$$\sqrt{\left(-\text{eigenval} / \left(\frac{1}{4} \text{MathieuCharacteristicA}[0, -2 \text{eigenval} \%[[1]]\right)\right)}$$

```
Out[56]:= 27.5568
```

This is in perfect agreement with the value obtained by our method. Remember that the second coordinate of that point was calculated by the bisection method with a tolerance of 0.1.

It can be shown that the method of normal coordinates works for any number of rods, not only for two. The procedure for producing the previous graph for three or more rods is the same and we do not reproduce it here, leaving it as an exercise for the reader.

## ■ 6. Conclusions

We explained our implementation of a method for producing very good approximations to solutions of linear differential equations with periodic coefficients. The method is based on calculating exact high-order Picard iterates by using a fast integration function and also works for equations or initial conditions depending on parameters. As a first accuracy test, we compared the numerical values for the first Mathieu characteristic numbers as predicted by our method (with 20 iterates) and the ones calculated by built-in *Mathematica* functions. The agreement was very good for the parameter values considered. Of course, the built-in functions are much faster, but we cannot forget that the proposed method is very general and the Mathieu equation is just an interesting example.

We also visually compared the stability chart for the vertically driven upside-down double pendulum obtained by two different methods, both based on calculating Picard iterates. The first method was to directly calculate the Picard iterates for a system of four equations and the second was to decouple the system of four equations into two Mathieu equations, for which we had already calculated the necessary Picard iterates. Again, the results with both methods were visually indistinguishable, providing an indirect check of the accuracy of the Picard iterates. We also checked directly that one of the points in that stability chart agreed numerically with the exact calculated value within the tolerance of the bisection method used.

By running the *FastPicard* package with *Mathematica* 4.1 on a 1.4 GHz Pentium 4 with 256 MB of RAM, we were able to calculate up to 50 iterates for the Mathieu equation in less than one hour. Results were reported in [6].

We should also note that there exist mathematically rigorous upper bounds on the difference between the exact solution of an equation and the  $n$ th Picard iterate approximating it. These bounds decrease exponentially fast in  $n$ . In principle, they can be used with symbolic Picard iterates to give computer-assisted proofs of properties of solutions for differential equations depending on parameters.

We used the phrase “in principle” in the previous paragraph for the following reason. The reader should have noticed from the example with the Mathieu equation that the errors committed in approximating solutions by a fixed number of Picard iterates increase with time. In our stability examples, it was necessary to approximate the solution of the differential equations up to time  $2\pi$ . This is indeed quite a long time interval for the number of iterates we are able to calculate. As a consequence, the error bounds are not very interesting. In applications where the time interval is shorter, it is possible that the error bounds could produce interesting proofs.

There also exist rigorous bounds on the errors committed by most traditional numerical methods. But these methods are difficult to implement in the case of equations depending on parameters and, even when the equations do not depend on parameters, the calculations in these methods are subject to roundoff errors that are difficult to overcome.

Therefore we propose exact Picard iteration as an alternative method for producing approximate solutions to linear differential equations with periodic coefficients depending on parameters. For problems in which the time intervals are short, we may even provide interesting upper bounds for the errors due to calculating only a finite number of iterates.

## ■ Acknowledgments

This work was partially supported by FAPEMIG, the Research Funding Foundation of the State of Minas Gerais, Brazil.

## ■ References

- [1] S. C. Sinha and E. A. Butcher, “Symbolic Computation of Fundamental Solution Matrices for Linear Time-Periodic Dynamical Systems,” *Journal of Sound and Vibration*, **206**(1), 1997 pp. 61–85.
- [2] E. A. Butcher and S. C. Sinha, “Symbolic Computation of Local Stability and Bifurcation Surfaces for Nonlinear Time-Periodic Systems,” *Nonlinear Dynamics*, **17**(1), 1998 pp. 1–21.
- [3] A. H. Nayfeh, *Perturbation Methods*, New York: John Wiley & Sons, 1973.
- [4] J. A. Sanders and F. Verhulst, *Averaging Methods in Nonlinear Dynamical Systems*, New York: Springer-Verlag, 1985.

- [5] M. Braun, *Differential Equations and Their Applications*, 2nd ed., New York: Springer-Verlag, 1978.
- [6] A. G. M. Neves, "Symbolic Computation of High-Order Exact Picard Iterates for Systems of Linear Differential Equations with Time-Periodic Coefficients," in *Computational Science—ICCS 2003, International Conference, Melbourne, Australia and St. Petersburg, Russia, June 2003, Proceedings, Part 1*; Lecture Notes in Computer Science, Vol. 2657, pp. 838–847, Heidelberg: Springer-Verlag, 2003.
- [7] D. J. Acheson, "A Pendulum Theorem," *Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences*, **443**, 1993 pp. 239–245.
- [8] D. J. Acheson and T. Mullin, "Upside-down Pendulums," *Nature*, **366**, 1993 pp. 215–216.
- [9] M. V. Bartuccelli, G. Gentile, and K. V. Georgiou, "On the Dynamics of a Vertically Driven Damped Planar Pendulum," *Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences*, **457**, 2001 pp. 3007–3022.
- [10] F. A. Alhargan, "Algorithms for the Computation of All Mathieu Functions of Integer Orders," *ACM Transactions on Mathematical Software (TOMS)*, **26**(3), 2000 pp. 390–407.
- [11] D. Frenkel and R. Portugal, "Algebraic Methods to Compute Mathieu Functions," *Journal of Physics A, Mathematical and General*, **34**, 2001 pp. 3541–3551.
- [12] A. G. M. Neves, "Upper and Lower Bounds on Mathieu Characteristic Numbers of Integer Orders," *Communications on Pure and Applied Analysis*, **3**(3), 2004 pp. 447–464.
- [13] R. E. Maeder, *The Mathematica Programmer*, Boston: AP Professional, 1994.
- [14] S. Wolfram, *The Mathematica Book*, 4th edition, Champaign, Oxford: Wolfram Media/Cambridge University Press, 1999.
- [15] A. Stephenson, "On a New Type of Dynamical Stability," *Memoirs and Proceedings of the Manchester Literary and Philosophical Society*, **52**(8), 1908 pp. 1–10.
- [16] A. Stephenson, "On Induced Stability," *Philosophical Magazine*, **15**, 1908 pp. 233–236.
- [17] H. Goldstein, *Classical Mechanics*, 2nd ed., Reading, MA: Addison-Wesley, 1980.
- [18] D. W. Jordan and P. Smith, *Nonlinear Ordinary Differential Equations*, 2nd ed., Oxford: Clarendon Press, 1987.

## ■ Additional Material

FastPicard.m

Available at [www.mathematica-journal.com/issue/v10i1/download](http://www.mathematica-journal.com/issue/v10i1/download).

## About the Author

Armando G. M. Neves majored in physics in 1986 at the Federal University of Minas Gerais (UFMG) in Brazil. After receiving a master's degree, he obtained his doctoral degree in physics at Rome University "La Sapienza" in Italy in 1993. Although his degrees are in physics, he has always searched for mathematical rigor. Since 1992 he has held a position in the Mathematics Department of UFMG.

His general research interest is mathematical physics. His principal interests are in statistical mechanics, quantum field theory, and, more recently, dynamical systems. Although his first academic works were noncomputational, after using *Mathematica* he realized that symbolic computation opened up entirely new fields of problems and provided new ways of attacking older problems.

### **Armando G. M. Neves**

*UFMG - Departamento de Matemática*  
*Av. Antônio Carlos, 6627 - Caixa Postal 702*  
*30123-970 Belo Horizonte - MG*  
*BRAZIL*  
*aneves@mat.ufmg.br*  
*www.mat.ufmg.br/~aneves*