# A mixed symbolic - numeric software environment

S. L. Campbell*, F. Delebecque† and D. von Wissel †
∗ Dept. of Mathematics, North Carolina State University, Raleigh
NC 27695 - 8205 - USA

† INRIA - Rocquencourt, 78153 Le Chesnay Cedex - France

## 1. Introduction

The objective of this paper[1] is to illustrate the use of a Maple to Scilab interface (Scilab is a free CACSD package developed at INRIA. `ftp://ftp.inria.fr/INRIA/Projects/Meta2/Scilab` `http://www-rocq.inria.fr/scilab`). We show that the interface is a powerful tool for analyzing nonlinear systems and illustrate the interface on examples of the modeling and control of mechanical multibody systems.

Many control applications require the use of both symbolic and numeric computation. An illustrative example is the analysis and controller design of a mechanical system. To begin with, we need to compute the equations of motion. To study local stability we need to compute its linearization. To apply control concepts like feedback linearization we may need to compute the inverse system. All these operations are symbolic computations. To continue, we may want to do a computer simulation of the nonlinear system, study stability of the linearization at different operating points or try a nonlinear controller. These are numerical operations evaluating the symbolic objects obtained before.

In the applications we have in mind, we first use symbolic computation. Then we need to transfer symbolic objects to numerically evaluable objects and finally we work with these new objects in a numerical environment. The opposite direction, that is the transfer of a numerical objects into a symbolic object is often of no interest. Indeed symbolic calculations are essentially based on rational numbers i.e. each number is associated with a pair a two "bignums" representing its numerator and its denominator. This representation allows to perform efficiently exact calculations [5, 4] involving e.g. polynomials and is well suited to study specific problems which depend on a few number of parameters. For instance it is well suited for studying parametric controllability of a linear system. However it is difficult to use this approach if the basic data are floating point numbers. In that case, after conversion to rational numbers, the symbolic algorithm is likely to return a generic result. For instance any floating point matrix converted to rational numbers will be considered as full rank by a symbolic package eventhough it has a arbitrarily small singular values. In other words, the use of symbolic package in that context may lead to non robust results. Our approach is based on the use of symbolic package for generating specific numerical Fortran or C code which is dynamically linked to Scilab.

[1]Research partially supported by INT-9220802, DMS-9423705, ECS-9500589

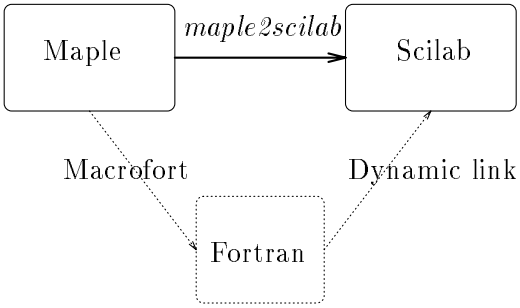
## 2. Software tools

We use Maple for symbolic and Scilab for numerical computation. What we need is an interface that permits these tools to communicate with each other.

One realization of this interface is to incorporate the Maple kernel in Scilab. This would allow one to perform symbolic operations within the Scilab environment. However, as a matter of fact, only a few Maple functions would be available since symbolic computation requires a different data structure and available Maple operations have to be compatible with the existing Scilab data structure.

Another realization of the desired interface is to keep Maple and Scilab separated and to handle the transfer by an external tool. The latter structure allows one to use the original software (thus keeping all of its advantages). There is no need to find a compromise between two essentially incompatible data structures. Furthermore, this approach makes the transfer transparent to the user and allows the use of compiled code (Fortran or C) for the numerical evaluation of Maple objects. Since the applications we have in mind require the use of numerous Maple facilities and execution time for the numerical evaluation of the Maple objects is important (mechanical multibody systems often involve large expressions), we have opted for the latter approach for the interface considered here. The key point of this interface is the facility of dynamic linking available in Scilab. The interface is based on the Maple

expressions into Scilab functions. Using Macrofort[2], we transfer the Maple object into a Fortran (or C) subroutine and dynamically link the code to Scilab. The linked code can be called like any other Scilab function with constant matrix parameters. For the applications that we consider, such as simulation and control of large dynamic systems, it is important to generate code which can be compiled. For instance, for ode simulations we generate a standalone Fortran subroutine with a specific calling sequence compatible with Scilab ode solvers. Another very important aspect is the use of Maple `optimize` facility which allows to find common subexpressions in a symbolic matrix, thus producing efficient code.




The syntax of the function `maple2scilab` is the following:
**maple2scilab**(*fname*, *maplematrix*, *parameters*)

*maplematrix* is the name of the Maple matrix to be transfered and admitting *parameters* as parameters (list of vectors or matrices). *fname* is the name of the generated Scilab function: this function returns the numerical value of the transfered matrix for given values of input parameters. A similar function exists for sparse matrices. *maple2scilab* generate two files: *fname.f* and *fname.sci* which contain Fortran and Scilab code. Compilation of the numerical code can be easily automatized by a short shell script, making the transfer transparent to the user.

## 3. Examples

In this section we illustrate the approach on two examples.

### 3.1. Inverted pendulum

In this well known example, the state variables are the position of the cart $x = q_1$ and the angle $\alpha = q_2$. Below

[2]Macrofort and MacroC are Maple libraries that can be found in the Maple share-directory

$L = \frac{m_1+m_2}{2}\dot{x}^2 + \dot{\alpha}^2(J/2 + l^2m_2^2/2) + lm_2\cos(\alpha)(\dot{x}\dot{\alpha} - g)$

```
L:=qd[1]^2*(m[1]+m[2])/2+qd[2]^2*(J/2+l^2*m[2]/2)+
qd[1]*qd[2]*l*m[2]*cos(q[2])-g*l*m[2]*cos(q[2]);

q:=vector(2);qd:=vector(2);qdd:=vector(2);m:=vector(2);

S1:=grad(L,qd);

S2:=evalm(grad(L,q)-
jacobian(S1,q)&*qd-jacobian(S1,qd)&*qdd);

F:=subs(qdd[1]=0,qdd[2]=0,evalm(S2));
M:=jacobian(S2,qdd);

maple2scilab(‘msci‘,M,[q,l,m,g,J]);
maple2scilab(‘fsci‘,F,[q,qd,l,m,g,J]);
```

This yields the equations of motion

$$M \begin{bmatrix} \ddot{x} \\ \ddot{\alpha} \end{bmatrix} = F$$

where matrix $M$ and vector $F$ are directly available as Scilab functions `msci` and `fsci` which evaluate $M$ and $F$ as function of the input parameters `q,l,m,g,J`. `maple2scilab` generated:

```
subroutine msci(q,l,m,g,J,fmat)
implicit double precision (t)
double precision q(2),l,m(2),g,J,fmat(2,2)
t4 = l*m(2)*cos(q(2))
t5 = l**2
fmat(1,1) = -m(1)-m(2)
fmat(1,2) = -t4
fmat(2,1) = -t4
fmat(2,2) = -J-t5*m(2)**2
end


function var=msci(q,l,m,g,J)
var=fort('msci',q,1,'d',l,2,'d',m,3,'d',...
g,4,'d',J,5,'d','out',[2,2],6,'d')
```

A simulation of the pendulum for a given proportional control $u$ is then easily obtained by the following Scilab script:

```
link('fsci.o','fsci');getf('fsci.sci','c')
link('msci.o','msci');getf('msci.sci','c')
l=1;m=[1;.1]J=.1;g=9.81;
deff('zd=pend(t,z)',..
'q=z(1:2),qd=z(3:4);u=K*(z-zop);..
zd=[qd;inv(msci(q,l,m,g,J))*..
      fsci(q,qd,l,m,g,J)+[u;0]]')
z0=[0;%pi/2;0;0];t=0:0.1:10;
xx=ode(z0,0,t,pend);
```

$$\ddot{\alpha} - g/p \sin(\alpha) = 1/p \cos(\alpha) \qquad (1)$$

is obtained by assuming that we control $\alpha$ directly by the acceleration of the cart $\ddot{x}$.

### 3.2. Simplified bicycle model

In this section we consider a more complex constrained mechanical system (which has in addition nonholonomic constraints). In this case, it is much more difficult to obtain the equations of the model. A blind application of the Euler Lagrange equations, introducing Lagrange multipliers for the nonholonmic constraints leads to a much more complex model in implicit form in which all variables are coupled. The main difficulty is to find an appropriate change of coordinates for simplifying the equations of motion. This part requires many symbolic computations which cannot easily be automatically implemented. The model considered here is a simplified bicycle model introduced by N. Getz [6].

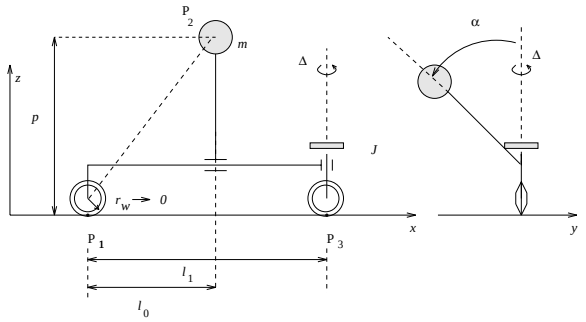



**Figure 1** *Simple bicycle model considered by Getz [6]*

We denote by $x_1, y_1$ (resp. $x_2, y_2$) the plane coordinates of $P_1$ (resp. $P_2$) and $\kappa = \tan(\Delta)/l_1$. The $xy$-coordinates of the velocity of $P_1$ are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} v_1 \\ v_0 \end{bmatrix} \qquad (2)$$

where $v_0$ is the velocity of $P_1$ perpendicular to the contact line $\overline{P_1P_3}$, and $v_1$ the velocity parallel to $\overline{P_1P_3}$ (rolling direcion). The rolling without sliding condition is $v_0 = 0$ and we have from (2) the nonholonomic constrains:

$$\dot{x}_1 = v_1 \cos(\phi), \ \dot{y}_1 = v_1 \sin(\phi)$$

To write the Lagrangian, we need the velocity of the point mass $m$ at $P_2 = (x_2, y_2, z_2)$. We have

$$\begin{aligned} x_2 &= x_1 + \cos(\phi)\, l_0 + \sin(\phi)\sin(\alpha)\, p \\ y_2 &= y_1 + \sin(\phi)\, l_0 - \cos(\phi)\sin(\alpha)\, p \\ z_2 &= p\cos(\alpha) \end{aligned} \qquad (3)$$

is small against the steering velocity $\dot{\Delta}$, we have for the kinetic energy of the steering wheel

$$T_{\text{kin}} = \frac{J}{2}(\dot{\Delta} + \dot{\phi})^2 \approx \frac{J}{2}\dot{\Delta}^2 = \frac{J}{2}\frac{l_1^2\dot{\kappa}^2}{(1+\kappa^2 l_1^2)^2}$$

which yields for the Lagrangian:

$$L = \frac{m}{2}\left(\dot{x}_2^2 + \dot{y}_2^2 + \dot{z}_2^2\right) + \frac{J}{2}\frac{l_1^2\dot{\kappa}^2}{(1+\kappa^2 l_1^2)^2} - g\, m\, z_2 \qquad (4)$$

where $\dot{x}_2, \dot{y}_2$ and $\dot{z}_2$ are computed from (3).

Let $\alpha$ denote the lean angle of the bicycle, set $\mathrm{c}_\alpha = \cos(\alpha)$ and $\mathrm{s}_\alpha = \sin(\alpha)$, and define the partitioned generalized coordinates and velocities of the bicycle as follows:

$$\begin{aligned} r = (\sigma_1, \alpha, \kappa)^T, &\qquad \dot{r} = (\dot{\sigma}_1, \dot{\alpha}, \dot{\kappa})^T \Leftrightarrow \dot{r} = (v_1, \dot{\alpha}, \dot{\kappa})^T \\ s = (\sigma_0, \phi)^T, &\qquad \dot{s} = (\dot{\sigma}_0, \dot{\phi})^T \Leftrightarrow \dot{s} = (v_0, \dot{\phi})^T \end{aligned}$$

where $\sigma_1 = \int_0^t v_1 dt$ is the arc length of the trajectory of the contact point of the rear wheel in the $xy$ plane and $\sigma_0 = \int_0^t v_0 dt$. In these velocity coordinates the nonholonomic constraints become $\kappa v_1 = \dot{\phi}$, $v_0 = 0$, or equivalently

$$\dot{s} + A\dot{r} = 0, \text{ where } A = \begin{bmatrix} 0 & 0 & 0 \\ -\kappa & 0 & 0 \end{bmatrix} \qquad (5)$$

The Lagrangian is computed from (4), substituting $\dot{x}_2, \dot{y}_2$ and $\dot{z}_2$ by (3), and $\dot{x}_1$ and $\dot{y}_1$ by (2)

As shown in [1, 6, 3], the equations of motion can be computed using a "constrained" Lagrangian $L_c$ obtained by substituting the constraints into the Lagrangian.

$$L_c(r^i, \dot{r}^i) = L(r^i, -A_i^j(r)\dot{r}^j)$$

where $\dot{s}^j + A_i^j(r)\dot{r}^j = 0$. We obtain

$$V := \frac{d}{dt}\frac{\partial L_c}{\partial \dot{r}^j} - \frac{\partial L_c}{\partial r^j} + \frac{\partial L}{\partial \dot{s}^i} B_{j,l}^i \dot{r}^l = 0 \qquad (6)$$

where

$$B_{j,l}^i = \frac{\partial A_j^i}{\partial r^l} - \frac{\partial A_l^i}{\partial r^j}$$

which can be written in the following form:

$$M(r)\ddot{r} = F(r, \dot{r})$$

where $M(r) := \partial V/\partial \ddot{r}$ and $F := V - M(r)\ddot{r}$.

For the bicycle model $L_c$ is computed from $L$ setting $v_0 = 0$ and $\dot{\phi} = \kappa v_1$ This yields the simple bicycle model, which is composed out of the dynamical part, containing the reduced order equations of motion, and

straints. Furthermore, we need to a vector of generalized forces $G(r)u$ representing the control inputs. We take a steering torque and a pedalling torque as control inputs.

$$\begin{array}{ll} \text{Dynamical part} & \{M(r)\ddot{r} = F(r,\dot{r}) + G(r)u \\ \text{Kinematic part} & \begin{cases} \dot{\phi} &= v_1\kappa \\ \dot{x}_1 &= v_1\cos(\phi) \\ \dot{y}_1 &= v_1\sin(\phi) \end{cases} \end{array} \qquad (7)$$

Designing a controller for such a model is unnecessarily complex. On the contrary, using the above decoupled model a controller is easily designed.

*Maple program to compute the reduced equations of motion:* We have implemented the procedure described before as a Maple script. The key step in this procedure is the change of coordinates that puts the nonholonomic constraint in the special form (5). In particular it is not easy to determine the partitioning of the variables. There is no systematic procedure for executing this step. The remaining steps can easily be translated into a Maple script. To this end we use the following set of generalized coordinates

| foo | $\phi$ | bar | $\alpha$ | $\kappa$ |
|---|---|---|---|---|
| $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ |
| $v_0$ | $\dot{\phi}$ | $v_1$ | $\dot{\alpha}$ | $\dot{\kappa}$ |
| $qd_1$ | $qd_2$ | $qd_3$ | $qd_4$ | $qd_5$ |
| $\dot{v}_0$ | $\ddot{\phi}$ | $\dot{v}_1$ | $\ddot{\alpha}$ | $\ddot{\kappa}$ |
| $qdd_1$ | $qdd_2$ | $qdd_3$ | $qdd_4$ | $qdd_5$ |

Below is the resulting Maple session ($l_1 = b, l_0 = c$).

```
L:=-m*G*p*cos(alpha) +
1/2*J*(b*kappa[d]/(1+b^2*kappa^2))^2+
1/2*m*((v[1]+p*sin(alpha)*phi[d])^2+
(v[o]-p*alpha[d]*cos(alpha)+c*phi[d])^2+
(p*alpha[d]*sin(alpha))^2);
```

$$L = -mgp\cos(\alpha) + \frac{Jb^2{\kappa_d}^2}{2\,(1+b^2\kappa^2)^2} + m/2\left(v_1 + p\sin(\alpha)\phi_d\right)^2 + \left(v_o - p\alpha_d\cos(\alpha) + c\phi_d\right)^2 + p^2{\alpha_d}^2\left(\sin(\alpha)\right)^2$$

```
q:=vector(5);qd:=vector(5);qdd:=vector(5);
sg:=v[o]=qd[1],phi[d]=qd[2],v[1]=qd[3],
alpha[d]=qd[4],kappa[d]=qd[5],foo=q[1],
phi=q[2],bar=q[3],alpha=q[4],kappa=q[5];

L0:=subs(sg,L);#unconstrained lagrangian
LG2:=collect(
expand(subs(cos(alpha)^2=1-sin(alpha)^2,
expand(subs(phi[d]=kappa*v[1],v[o]=0,L)))),v[o]):
L_c:=subs(sg,LG2);#constrained lagrangian
```

$$+m/2]\,qd_3{}^2 - mpqd_4\cos(q_4)cq_5qd_3 - mgp\cos(q_4) + Jb^2qd_5{}^2/(2\,\left(1+b^2q_5{}^2\right)^2) + mp^2qd_4{}^2/2$$

```
r:=subs(sg,vector([foo,alpha,kappa]));
rd:=subs(sg,vector([v[1],alpha[d],kappa[d]]));
s:=subs(sg,vector([foo,phi]));
sd:=subs(sg,vector([v[0],phi[d]]));
A:=matrix(2,3,[0,0,0,-q[5],0,0]);

V:=proc(al)
d_dt(diff(L_c,rd[al]))-diff(L_c,r[al])+
sum('A[a,al]*diff(L_c,s[a])','a'=1..2)+
sum('sum('diff(L0,sd[b])*B(b,al,be)*rd[be]',
    'be'=1..3)','b'=1..2);
end;

B:=proc(b,al,be)
diff(A[b,al],r[be])-diff(A[b,be],r[al]);
end;

d_dt:=proc(f)
#The function d_dt differentiates f(q) and replaces
#the q' by dq, qd' by qdd respectively.
eval(innerprod(grad(f,q),qd)+
     innerprod(grad(f,qd),qdd));end;

V1:=subs(m=1,vector(3,V));
M:=jacobian(convert(V1,vector),
   vector([qdd[3],qdd[4],qdd[5]]));
```

$$M := \begin{bmatrix} M_{11} & M_{12} & 0 \\ M_{12} & p^2 & 0 \\ 0 & 0 & M_{33} \end{bmatrix}$$

where

$$\begin{aligned} M_{11} &= 1 + 2\,p\sin(q_4)q_5 + p^2\left(\sin(q_4)\right)^2{q_5}^2 + c^2{q_5}^2 \\ M_{12} &= -p\cos(q_4)cq_5 \\ M_{33} &= \frac{Jb^2}{\left(1+b^2{q_5}^2\right)^2} \end{aligned}$$

```
F:=convert(map(collect,map(expand,
subs(qd[2]=q[5]*qd[3],qd[1]=0,qdd[1]=0,qdd[2]=0,qdd[3]=0,
qdd[4]=0,qdd[5]=0,evalm(V1))),{qd[3],qd[4],qd[5]}),matrix);
```

$$F := \begin{bmatrix} F_1 \\ F_2 - g\,p\sin(q_4) \\ F_3 \end{bmatrix}$$

$$F_1 = \left(\left(2\,p^2\sin(q_4){q_5}^2 + 2\,p\,q_5\right)\cos(q_4)qd_4 + \left(c^2q_5 + p\sin(q_4) + p^2\left(\sin(q_4)\right)^2q_5\right)qd_5qd_3\right)$$

$$
\begin{aligned}
F_2 &= \left(-p\cos(q_4)q_5 - p^2\sin(q_4){q_5}^2\cos(q_4)\right)q{d_3}^2 - \\
&\quad p\cos(q_4)c\,q{d_3}q{d_5} \\
F_3 &= -\frac{2\,Jb^4q{d_5}^2q_5}{\left(1+b^2{q_5}^2\right)^3} = -M_{33}\frac{2\,b^2q{d_5}^2q_5}{\left(1+b^2{q_5}^2\right)}
\end{aligned}
$$

For the vector of generalized forces we have

$$
G(r)u := \begin{bmatrix} u_2/m \\ 0 \\ u_1/m \end{bmatrix}
$$

This bicycle model can be controlled using various control strategies. We can use system inversion for tracking the lean angle of the bicycle to a desired trajectory. This is possible because the bicycle is with this choice of output a minimum phase system [6]. First, we consider linear feedback control for stabilizing the bicycle dynamics. For this we need to compute the linearization of the bicycle model. This is easily done in Maple. The linearization is of the form

$$
M(r)\delta\ddot{r} = \frac{\partial}{\partial r}(F(r,\dot{r}) - M(r)\ddot{r})\delta r + \frac{\partial}{\partial \dot{r}}F(r,\dot{r})\delta\dot{r}
$$

which needs to avaluated at $r = r_{op}$, $\dot{r} = \dot{r}_{op}$ and $\ddot{r} = \ddot{r}_{op}$. In the case of the bicycle we can assume that $\ddot{r}_{op} = 0$. To compute the linear system in $r_{op}$ it suffices to compute the Jacobian of $F$ with respect to $(\alpha, \kappa, v_1, \dot{\alpha}, \dot{\kappa})$:

```
ff_x:=jacobian(convert(F,vector),
     vector([q[4],q[5],qd[3],qd[4],qd[5]]));
```

The simulation function in Scilab are `msci()` and `ff()`. They are simular to those of the inverted pendulum and will not be reproduced here. For computing a linear feedback matrix $K$ we proceed as follows: the linearization is given by $\dot{x} = Ax + Bu$ where $x = (\delta\alpha, \delta\kappa, \delta v_1, \delta\dot{\alpha}, \delta\dot{\kappa})^T$. The following scilab script computes $A$ and $B$.

```
F_x=[0 0 0 1 0;
   0 0 0 0 1;
   ff_x(r_op,rd_op,par)];
E=[eye(5,2),[zeros(2,3);msci(q0,qd0,par)]];
iE=inv(E);
A=iE*F_x;
B=iE*[zeros(2,2);0 1;0 0;1 0]
```

In this paragraph we use system inversion ideas to find a trajectory tracking controller for the lean angle. Considering the lean angle and the velocity as output the resulting system is minimum phase.

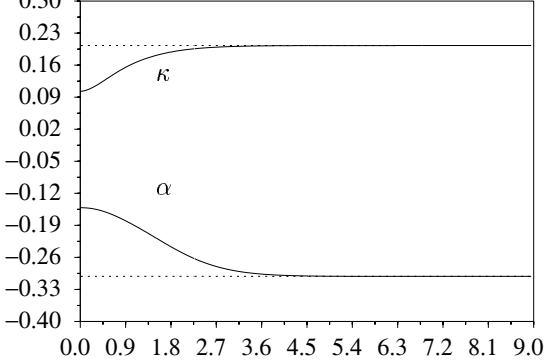



**Figure 2** *Linear stabilizing control*

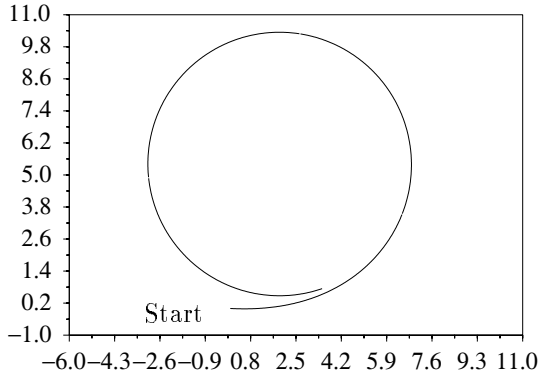



**Figure 3** *The resulting trajectory in the xy-plane.*

The input transformation

$$
\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} M_{33}\left(\ddot{\kappa} + \frac{2\,b^2\dot{\kappa}^2\kappa}{(1+b^2\kappa^2)}\right)m \\ (M_{11}\dot{v}_1 + M_{12}\ddot{\alpha} - F_1)m \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \quad (8)
$$

puts the model into the form

$$
\begin{cases} \ddot{\kappa} = w_1 \\ \dot{v}_1 = w_2 \\ p^2\ddot{\alpha} - g\,p\,\sin(\alpha) = \cos(\alpha)\xi(\kappa,\dot{\kappa},v_1,w_2,\alpha) \end{cases}
$$
$$
\begin{cases} \dot{\phi} = v_1\,\kappa \\ \dot{x}_1 = v_1\cos(\phi) \\ \dot{y}_1 = v_1\sin(\phi) \end{cases} \quad (9)
$$

$$
\xi(\kappa,\dot{\kappa},v_1,w_2,\alpha) = \kappa\,w_2\,p\,l_0 + (\kappa^2p^2\sin(\alpha) + \kappa\,p){v_1}^2 + v1\,\dot{\kappa}\,p\,l_0 \quad (10)
$$

To compute a tracking controller for the output $y = (\alpha, v)$ using the system inversion approach we need to differentiat the output untill we have an expression which can explicitly be solved for $w$. Noting that $\kappa$ is decoupled, we may assume that $\alpha$ is controlled by $\dot{v}_1$ and $\dot{\kappa}$. This simplifies considerably the design procedure. Roughly speaking this amounts considering $\xi$ as

is similar to the simplified equations of motion of the pendulum (1). Here again, a blind application of system inversion is possible using Maple, but this lead to an unnecessarily complex expression for the controller.

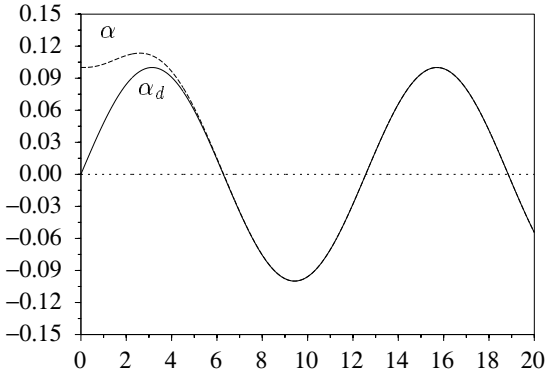



**Figure 4** *The lean angle $\alpha$ converges to the desired trajectory $\alpha_d$.*

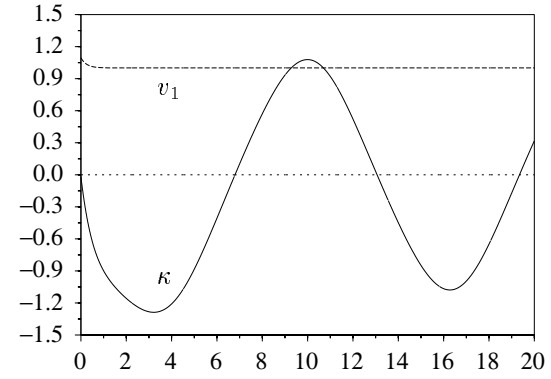



**Figure 5** *The velocity is held constant and $\kappa = \tan(\Delta)/l_1$ follows a kind of sine-function.*

After perfoming two differentiations of $\alpha$ and one differentiation of $v_1$ we get expressions which can be explicitly solved for $\dot{\kappa}$ and $w_2$. We note that this does not require any symbolic differentiation. Defining the error equation as

$$0 = \begin{bmatrix} b_0(v_1 - v_d) + (w_2 - \dot{v}_d) \\ a_0(\alpha - \alpha_d) + a_1(\dot{\alpha} - \dot{\alpha}_d) + (\ddot{\alpha} - \ddot{\alpha}_d) \end{bmatrix} \quad (11)$$

where

$$\ddot{\alpha} = \cos(\alpha)\xi(\kappa, \dot{\kappa}, v_1, w_2, \alpha) + gp\sin(\alpha)$$

and solving (11) for $\dot{\kappa}$ and $w_2$ we get an assymptotic tracking controller which tracks the output $y = (v_1, \alpha)$ to the desired ouput trajectory $y_d = (v_d, \alpha_d)$. The error tracking dynamics are determined by the coefficents $b_i$'s and $a_i$'s. Clearly, before we apply this control to the model we need to design a controller for $\kappa$ (we might

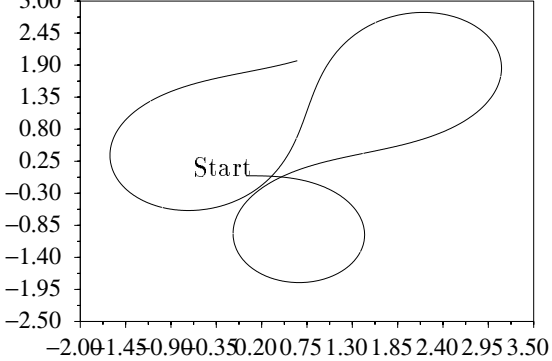



**Figure 6** *The resulting trajectory in the xy-plane.*

take the linear control $w_1 = k_1(\dot{\kappa} - \dot{\kappa}_d)$, where $\kappa_d$ is the solution of (11)) and we need to apply the input transformation (8).

### 3.3. Conclusion

This paper has presented two nonlinear examples of mechanical systems. The first example is a simple inverted pendulum, the second example is a simplified bicycle model. In Maple, using the Euler Lagrange equations, we compute the equations of motion. We have shown how to use the interface to transfer the equations to Scilab for simulation or control. We have shown that for dealing with complex models, we need the full power of Maple to perform change of variables which cannot be automatized since they require a deep a priori knowledge of the model.

### References

[1] A. M. Bloch and P. S. Krishnaprasad, Nonholonomic Mechanical Systems with Symmetry, California Institute of Technology, 1995,CDS Technical Report No. 94-013.

[2] F. Delebecque and R. Nikoukhah, A mixed symbolic-numeric software environment and its application to control system engineering, *Recent advances in computer aided control*, ed. H. Herget and M. Jamshidi, pp. 221-245 (1992).

[3] D. von Wissel, DAE control of dynamical systems: example of a riderless bicycle, Ecole des Mines de Paris (France), PhD thesis (1996).

[4] A. B. Ogunye and A. Penlidis, State Space Analysis using MapleV, Proc. IEEE/IFAC Joint Symp. on CACSD, March 1994.

[5] N. Munro, P. Tsapekis, Some recent results using Symbolic Algebra, Proc. IEEE/IFAC Joint Symposium on CACSD, March 1994, Tucson.

[6] N. H. Getz, *Control of balance for a nonlinear nonholonomic nonminimum phase model of a bicycle*, Proc. of the ACC, Baltimore, 1994.