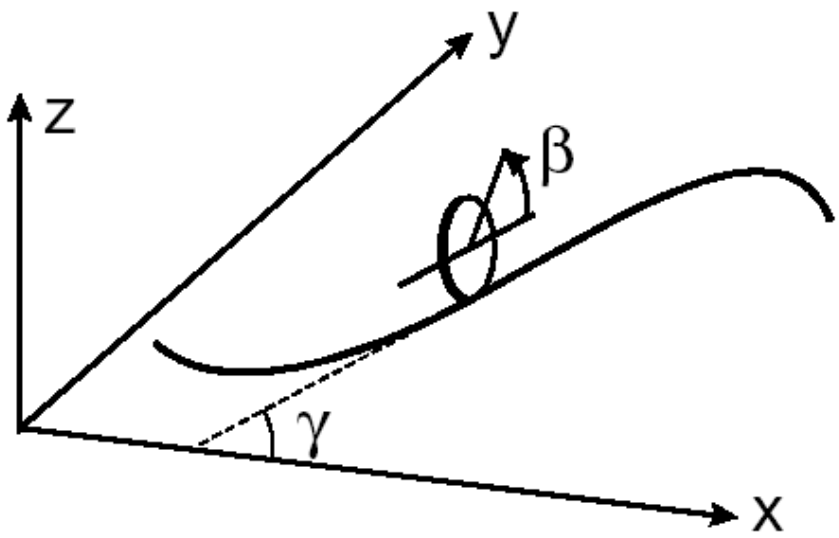
Nonholonomic Mechanic

Calculated with Matlab



NTNU Trondheim Norway

Date: August - September 2005

Author: Jürg Dietiker
Email: dij@gmx.ch
Homepage: http://www.didi80.ch/index_project.htm

Supervisor: Elena Celledoni
Email: elenac@math.ntnu.no

Index

1. SPECIFICATION 3
2. WHAT ARE HOLONOMICS SYSTEMS? 4
2.1 Holonomic system 4
2.2 Nonholonomic system 4
3. THE HOLONOMIC PROBLEM PENDULUM 5
3.1 Matlab function pendulum 5
3.2 Simulink model 6
3.2.1 Legend 6
3.2.2 Explaination of the legend 6
3.2.3 Solution 7
3.2.4 Remarks to the tolerances 8
4.ANALYSE THE ODE SOLVERS 9
4.1 Algorithmus 9
5. VERTICAL ROLLING DISK 10
5.1 with ode45 and ux=1 and uy=1 12
5.2 Change the Radius to R=1 12
5.3 Calculation with a smaller Error 13
5.4 θ2(0)=0 15
5.5 ode45 and ux=0 and uy=0 15
5.6 Comparison with the analytic solution in the case, with ux=0 and uy=0. 16
5.7 Simulink model 18
5.8 Discussion Simulink 19
5.9 Block function 20
6. COMPARE DIFFERENT INTEGRATION METHODS 21
7. THE VARIATIONAL CONTROLLED SYSTEM 24
7.1 ode45 and a=1 and b=1 25
7.2 θ2(0)=0 25
7.3 With function ux=exp(-t+5) and uy=exp(-t+5) 26
7.4 With function ux=exp(-t+5) and uy=exp(-t+5) and θ2(0)=0 26
8. BIBLIOGRAPHY 27

1. Specification

Numerical simulation of nonholonomic mechanical systems

Nonholonomic mechanical systems are of great interest in robot technology applications and control, in particular robotic locomotion and robotic grasping. Roughly speaking a mechanical system with nonholonomic constraints is described by a constrained differential equation (in Lagrange or Hamiltonian form) such that the constrains are involving the velocity of the system and not only the positions. In this project the numerical simulation of a simple noholonomic mechanical systems will be considered, the vertical disk rolling on a plane, Figurgure 2. The aim of the project is understanding the basic theoretical features of nonholonomically constrained systems and discuss which numerical approaches are best suited for such problems. In particular classical Runge-Kutta methods will be applied to the problems. In this work both MATLAB and SIMULINK are used as simulation tools. Some of the relevant features of these two simulation environments will be presented.

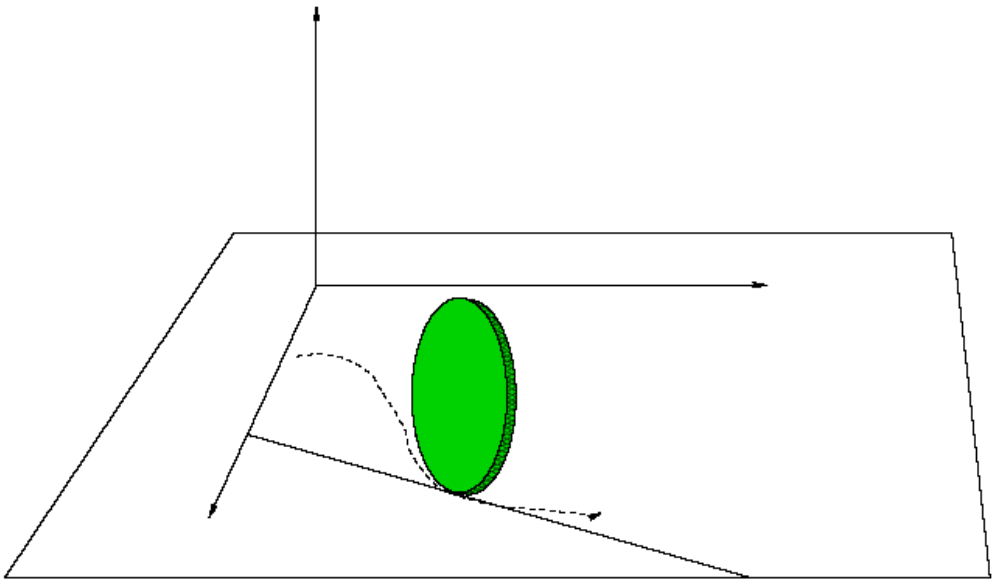


Figure 2: The rolling disk

© Elena Celledoni

2. What are Holonomics Systems?

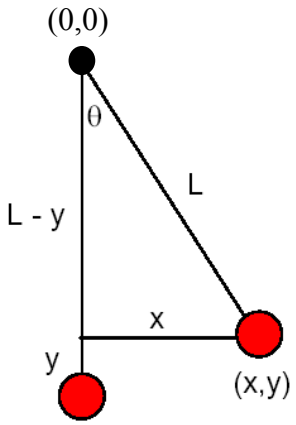
2.1 Holonomic system

In Classical Mechanics a system may be defined as holonomic if all the constraints of the system are holonomic. For a constraint to be holonomic it must be expressible as a function:

$$a_i(q_1, q_2 ... q_n, t) = 0, \quad i = 1,2...,m$$

Examples of holonomic systems are: the simple pendulum with the coinstrants function

$$\sqrt{x^2 + y^2} - L = 0$$



2.2 Nonholonomic system

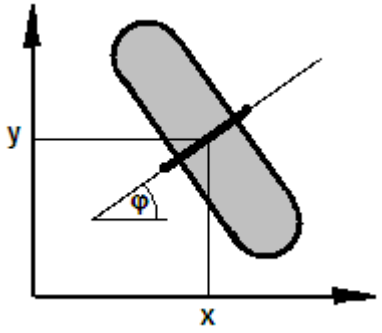
This is a system in which a return to the original internal configuration does not guarantee return to the original system position. In other words, unlike holonomic systems, the outcome of a nonholonomic system is path-dependent.

$$a_i(q_1, q_2, ... q_n, \dot{q}_1, \dot{q}_2 ... \dot{q}_n, t) = 0, \quad i = 1,2,...,m$$

For example, when riding a two-wheeled cart, a return to the original internal (wheel) configuration does not guarantee return to the original system (cart) position.
Cars, bicycles and unicycles are all examples of nonholonomic systems.

constrints for rollingdisk:

$$\dot{x}\cos\varphi + \dot{y}\sin\varphi = 0$$



3. The Holonomic problem Pendulum

We consider the simple pendulum

For this system, Newtons equations are;

$$F = mg\sin(\theta) = ma$$
$$a = g\sin(\theta)$$

Linear acceleration along the (mg sin(θ)) axis can be related to the change in angle θ by the arc length formula

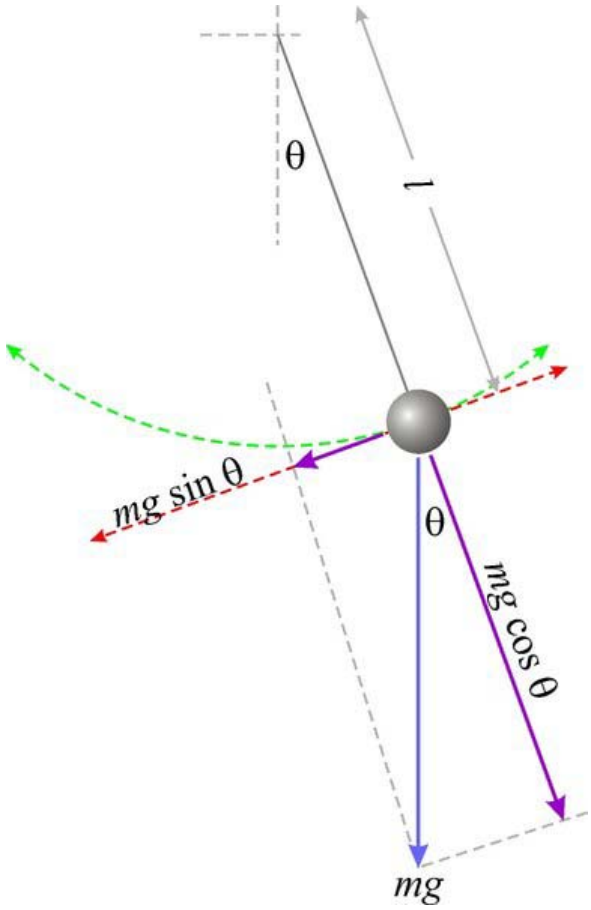
$$s = l\theta$$

The derivitives are:

$$v = \frac{ds}{dt} = l\frac{d\theta}{dt}; \quad a = \frac{d^2s}{dt^2} = l\frac{d^2\theta}{dt^2}$$

The finally equation is (for solve the Pendulum):

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin\theta = 0$$



3.1 Matlab function pendulum

To integrate his second order differential equation we consider the change of variables:

$$\theta_1 = \theta$$
$$\theta_2 = \dot{\theta} = \dot{\theta}_1$$

Wich gives the following first order system:

$$\dot{\theta}_1 = \theta_2$$
$$\dot{\theta}_2 = -\frac{g}{l}\sin\theta_1$$

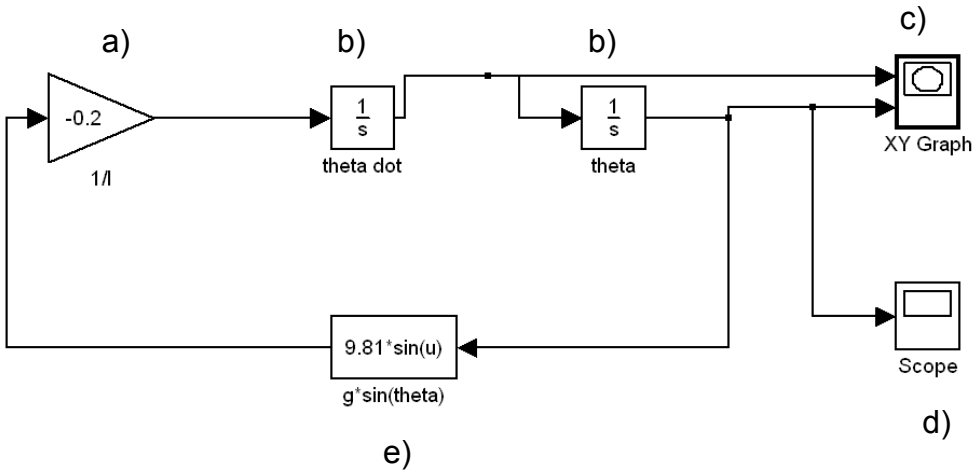
The right hand side of this system is coded in the Matlab function f=pendfunc(t,y). This Matlab function is used for the integration with the built-in Matlab routin ode15s:

```
function f=pendfunc(t,y)
%
l=5;
f(1,1)=y(2,1);
f(2,1)=-1/l*9.81*sin(y(1,1));
```

```
[t,yy]=ode15s('pendfunc',[0,10],[pi/2;0]);
```

3.2 Simulink model

The Simulink in Matlab is a Software to modeling and simulation automatic control systems. The model of the pendulum is.



3.2.1 Legend

a) Gain, Matrix Gain	→Multiply the input by a constant
b) Integrator	→Integrate a signal
c) XY Graph	→Display an X-Y plot of signals
d) Scope, Floating Scope	→Display signals generated during a simulation
e) Function	→Fcn Apply a specified expression to the input

3.2.2 Explaination of the legend

a) The Gain block multiplies the input by a constant value. In this case it's $-\frac{1}{l}$ with length l=5 the input is –0.2.

b) The Integrator block outputs the numerical solution of the different equation at the current time step. Simulink can use a number of different numerical integration methods for this task. In the Simulation parameters one can choose between the different Matlab functions. We must use two blocks because the equations is second order.

Initial condition → $\dot{\theta}(0) = 0$

Initial condition → $\theta(0) = \frac{\pi}{2}$

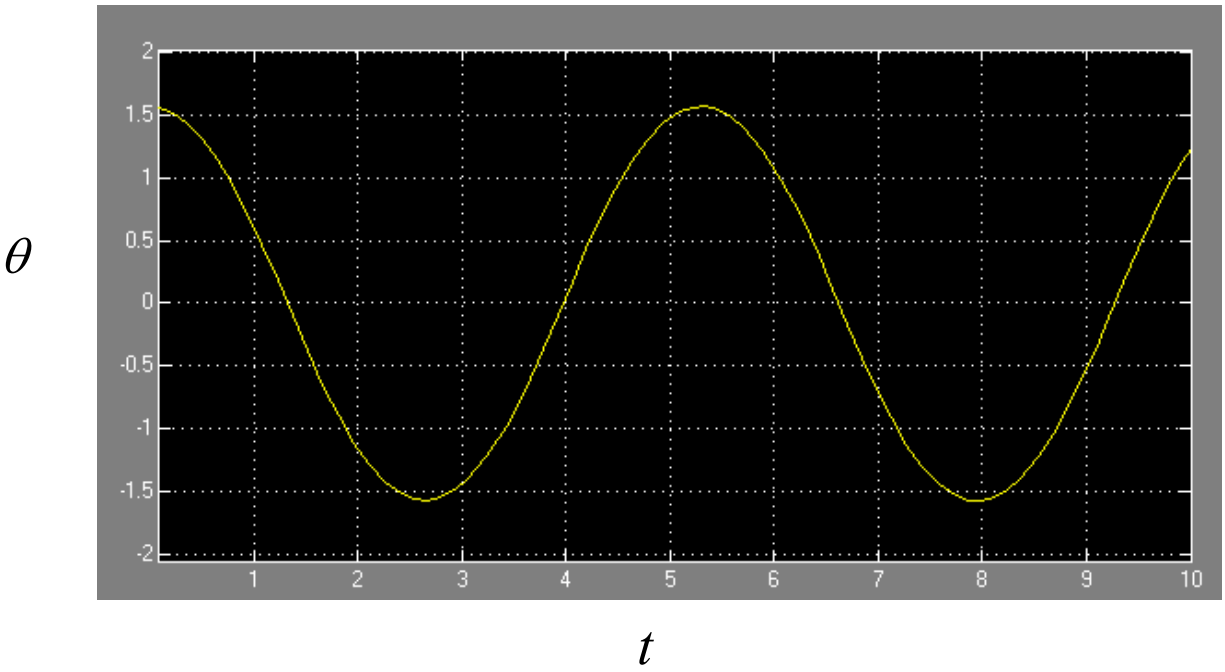
c) Gives a Graph with the relation velocity and position.

d) Gives a Graph with the relation time and position.

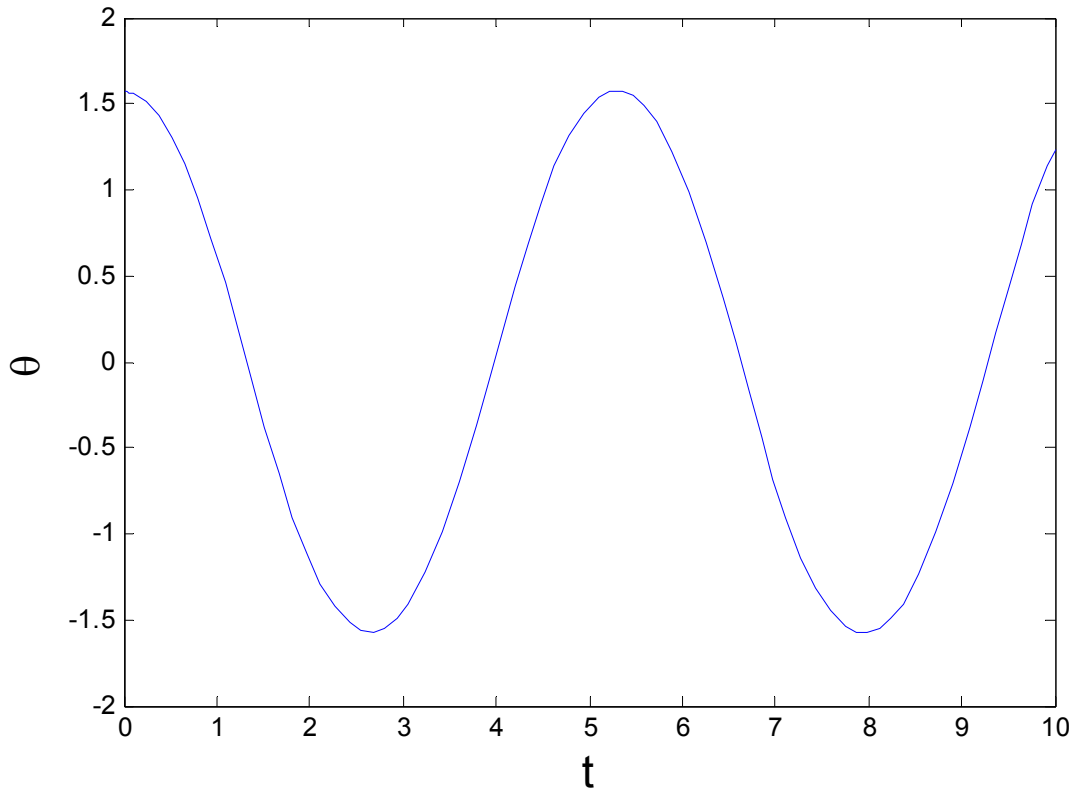
e) This block is a user defined function. The input to the block is u.

3.2.3 Solution

Graph with Simulink and the integrator ODE15s t=0…10



Graph with Matlab and the integrator ODE15s t=0…10



The differenc of these two numerical solutions gives an error of 1e-13 there we used:
Relative tolerance:1e-3
Absolute tolerance:1e-6

To compare the different of these two solutions, simulink and Matlab, we should change the parameters, otherwise the error is oversized and the step size aren't exact the same.

We set the parameters in Simulink:
→"simulation" → "simulate parameters"→"option outputs"→ "Produce Specified Output Only" with "output times" [0:0.1:10] to obtain the same step size. I get up to the max. Tolerance Rel. and Abs.

Relative tolerance:1.0e-13
Absolute tolerance:1.0e-20

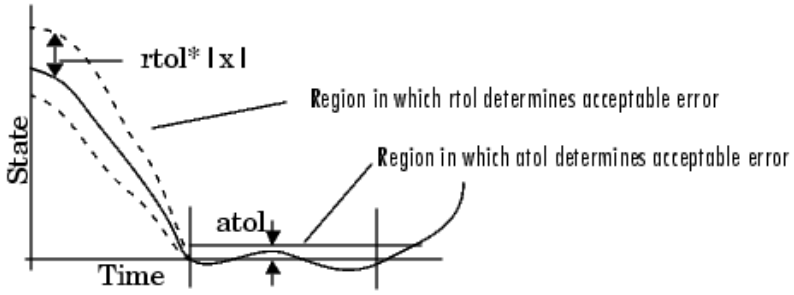
By ode45 compare with Matlab it's an difference 1.0e-13
By ode23tb compare with Matlab it's an difference 1.0e-8
By ode113 compare with Matlab it's an difference 1.0e-12

If the integrators do exactly the same it would be near zero and have a little error. Perhaps, some parameters aren't setting up right. Actually the error is prety little. I couldn't find out the right one, to have the exactly solution. It's strange that I obtain with ode15s very small difference, with normal Tolerances and with the other odes not.

With the higher order methode with ode45 , we obtain the smallest error.

3.2.4 Remarks to the tolerances

Relative tolerance measures the error relative to the size of each state. The relative tolerance represents a percentage of the state's value. The default, 1e-3, means that the computed state is accurate to within 0.1%.
Absolute tolerance is a threshold error value. This tolerance represents the acceptable error as the value of the measured state approaches zero.



The min. value for the Relative tolerance is 1e-13. Setting smaller tolerances one causes Matlab to produce an error message.

4.Analyse the ode solvers

ODE = Ordinary Differential Equation

Typ:ode45, ode23, ode113, ode15s, ode23s, ode23t, ode23tb

As we have seen in the pendulum example, the ODE solvers accept only first-order differential equations. To use the ODE solvers, one must write such equations as an equivalent system of first-order differential equations:

$$y = y_1$$
$$\dot{y} = y_2$$
$$\ddot{y} = \dot{y}_2$$

Exp: for the second order
Rewrite the problem as a system of first-order

$$\ddot{y}_1 = (\sqrt{1 - y_1})\dot{y}_1 - y_1$$

$$\dot{y}_1 = y_2$$

$$\dot{y}_2 = (\sqrt{1 - y_1})y_2 - y_1$$

---

solver	Problem Type	Type Order of Accuracy	When to Use
ode45	Nonstiff	Medium	Most of the time. This should be the first solver you try.
ode23	Nonstiff	Low	If using crude error tolerances or solving moderately stiff problems.
ode113	Nonstiff	Low to high	If using stringent error tolerances or solving a computationally intensive ODE file.
ode15s	Stiff	Low to medium	If ode45 is slow because the problem is stiff.
ode23s	Stiff	Low	If using crude error tolerances to solve stiff systems and the mass matrix is constant.
ode23t	Moderately Stiff	Low	If the problem is only moderately stiff and you need a solution without numerical damping.
ode23tb	Stiff	Low	If using crude error tolerances to solve stiff systems.

We used mostly the ode45 solver with odeset to set the tolerances.
The solver ode15s performs well but it uses more time than ode45.
With odeset, one can change the tolerances. There are big differences between the cases Tol. 1e-6 and 1e-13 in the solution. Proportionally to the size of the integrational interval the errors accumulate.

4.1 Algorithmus

ode45 →is based on an explicit Runge-Kutta (4,5) formula.
ode23 →is an implementation of an explicit Runge-Kutta (2,3).
ode113 →is a variable order Adams-Bashforth-Moulton PECE solver.
ode15s →is a variable order solver based on the numerical differentiation formulas (NDFs).
ode23s →is based on a modified Rosenbrock formula of order 2. Because it is a one-step solver
ode23t →is an implementation of the trapezoidal rule using a "free" interpolant.
ode23tb →is an implementation of TR-BDF2, an implicit Runge-Kutta formula with a first stage that is a trapezoidal rule step and a second stage that is a backward differentiation formula of order two.

5. Vertical Rolling Disk

It's a Nonholonomic Problem with constraints. A disk rolling without slipping on a horizontal plane. The Lagrangian is given by the total kinetic energy of the system:

$$L\{x, y, \varphi, \theta, \dot{x}, \dot{y}, \dot{\varphi}, \dot{\theta}\} = \frac{1}{2} m(\dot{x}^2 + \dot{y}^2) + \frac{1}{2} I\dot{\theta}^2 + \frac{1}{2} J\dot{\varphi}^2$$

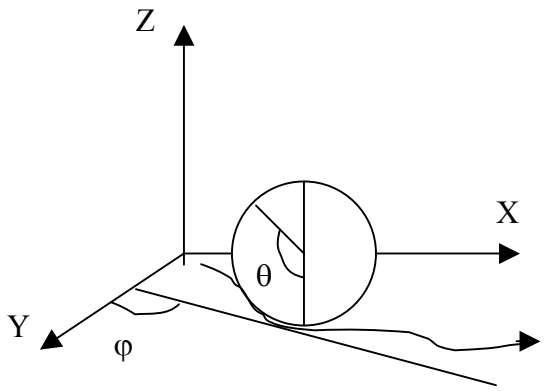
I = is the moment of inertia of the disk:

$$I = \frac{m}{12} * 3 * R^2 + h^2$$

h=thickness
R=radius
M=mass

J = is the moment of inertia about an axis in the plane of the disk:

$$J = \frac{1}{2} * m * R^2$$



There are 4 functions about the example in the book,
[Nonholonomic mechanics and Control, A.M.Block, P.18]:

Dynamic Equations

$$J * \ddot{\varphi} = v_\varphi$$
$$I + (m * R^2) * \ddot{\theta} = v_\phi$$

The constraints

$$\dot{x} = R * (\cos(\varphi)) * \dot{\theta}$$
$$\dot{y} = R * (\sin(\varphi)) * \dot{\theta}$$

We rewrite the second order System into a first order System. We obtain the following equations.

$$\dot{\varphi}_1 = \varphi_2$$
$$\dot{\varphi}_2 = J^{-1} * \upsilon_\varphi$$
$$\dot{\theta}_1 = \theta_2$$
$$\dot{\theta}_2 = (I + m * R^2)^{-1} * \upsilon_\theta$$
$$\dot{x} = R * \cos(\varphi_1) * \theta_2$$
$$\dot{y} = R * \sin(\varphi_1) * \theta_2$$

We write the right hand side of this problem in the Matlab function rolldisk

Program written in Matlab (M-file):

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function f=rolldisk(t,u)
%
R=10;                   %Radius
ux=1                    %control function
uy=1                    %control function
m=5;                    %mass
h=2;                    %broadness disk
J=(1/2)*m*R^2;          %rotation axes
I=(m/12)*(3*R^2+h^2);   %perpendicular axes trough barycentre
```

```
ph=u(1,1)
ph'=u(2,1)
om=u(3,1)
om'=u(4,1)
x=u(5,1)
y=u(6,1)
```

```
f(1,1)=u(2,1);
f(2,1)=(J^-1)*ux;

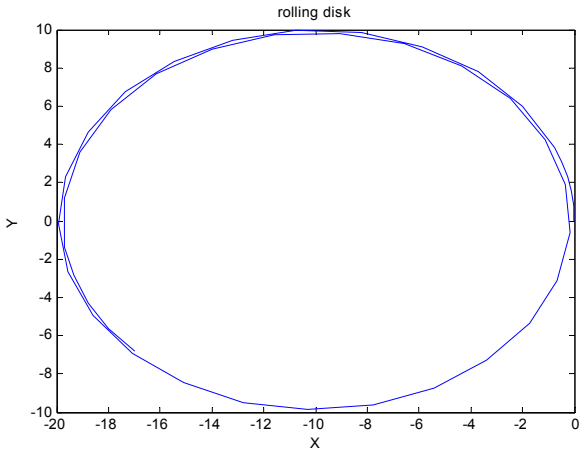
f(3,1)=u(4,1);
f(4,1)=((I+m*(R^2))^-1)*uy;

f(5,1)=R*cos(u(1,1))*u(4,1);
f(6,1)=R*sin(u(1,1))*u(4,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[t,outf]=ode45('rolldisk',[0,10],[pi;1;0;1;0;0]);     → with the start condition     →
plot(outf(:,5),outf(:,6));
```

5.1 with ode45 and ux=1 and uy=1

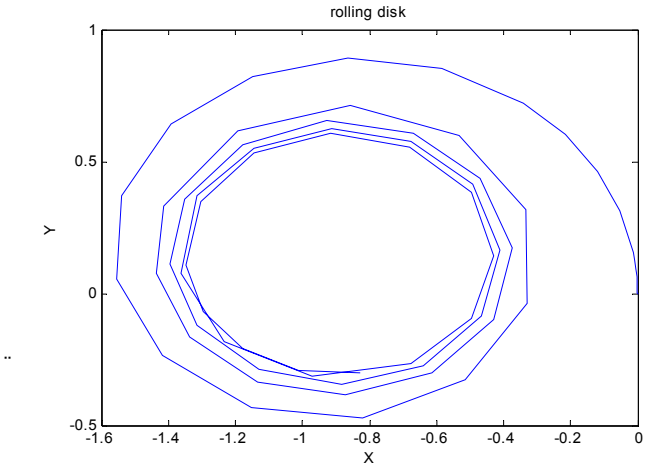
Plot the function in 2D and solve the equation with ode45 and ux=1 and uy=1



t	=0…10
φ1(0)	=Pi
φ2(0)	=1
θ1(0)	=0
θ2(0)	=1
x(0)	=0
y(0)	=0
R=10	
ux=1	
uy=1	
m=5	
h=2	

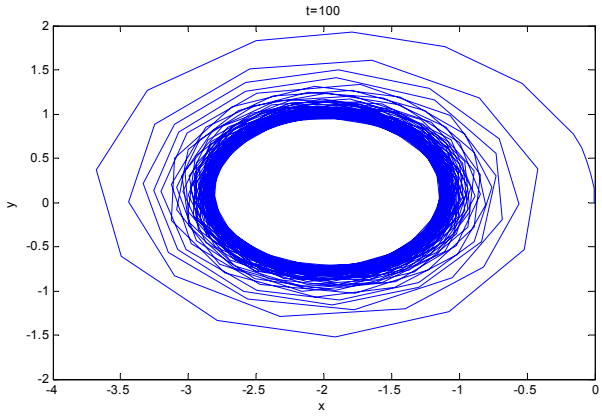
- If there are a cos and sin function, I'll be a circle function. See the functions in [5.5] with ux=0 and uy=0
- φ1 change the start point in X and Y→ experiment in Matlab
- θ2 make the arc/spiral smaller or taller → experiment in Matlab

5.2 Change the Radius to R=1



t	=0…10
φ1(0)	=Pi
φ2(0)	=1
θ1(0)	=0
θ2(0)	=1
x(0)	=0
y(0)	=0
R=1	
ux=1	
uy=1	
m=5	
h=2	

- If the Radius is smaller, the circular path gets smaller



t	=0…100
φ1(0)	=Pi
φ2(0)	=1
θ1(0)	=0
θ2(0)	=1
x(0)	=0
y(0)	=0
R=1	
ux=1	
uy=1	
m=5	
h=2	

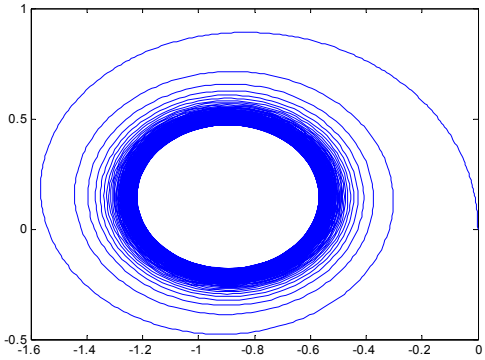
- The coin is rolling towards the centre. After a while the trajectory stabilizes on a minimal circle grading? I think the error get to big in the time 100s

5.3 Calculation with a smaller Error

It gives an auxialary option where you can set the RelTol and AbsTol. I repeat the same calculation from the chapter before to see, what's the different between the Tolerances

```
%%%%%%%%%%%%%%%%%%%%%%%%
options = odeset('RelTol',1e-13,'AbsTol',1e-20);
%
[t,yy]=ode45('rolldisk',[t],[pi/2;1;0;1;0;0],options)
%%%%%%%%%%%%%%%%%%%%%%%%
```

I repeat the calculation in [5.2] with a smaller tolerance now



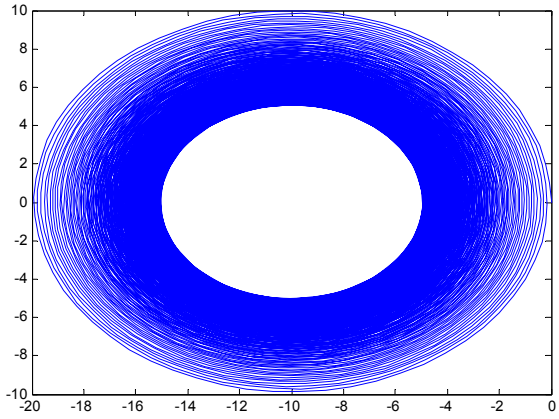
φ1(0) =Pi/2
φ2(0) =1
θ1(0) =0
θ2(0) =1
x(0) =0
y(0) =0

R=1
ux=1
uy=1
m=5
h=2

Step size: t=0:0.01:100

- The solution is the same with a smaller error. The coin stabilizies After a while on a minimal circle.

The same equation with R=10 and time t= 1000

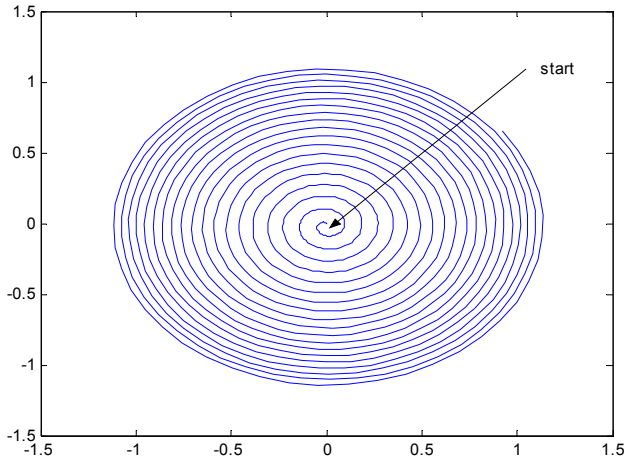


φ1(0) =Pi/2
φ2(0) =1
θ1(0) =0
θ2(0) =1
x(0) =0
y(0) =0

R=10
ux=1
uy=1
m=5
h=2

Step size: t=0:0.1:1000

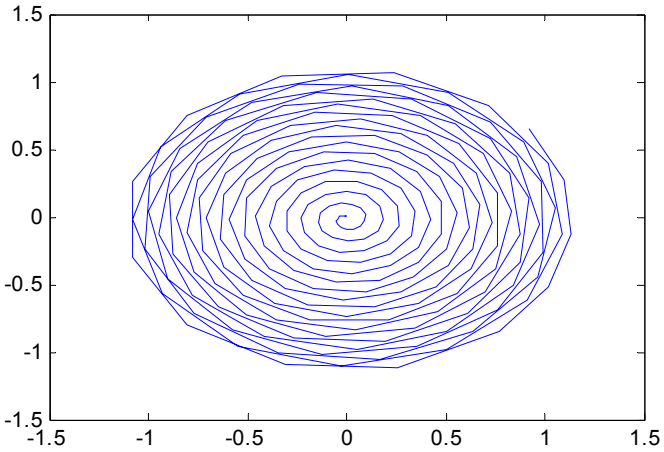
The error is even to big in a long time
I repeat the calculation in 5.3 with a smaller tolerance
θ2(0) =0



t	=0…100
φ1(0)	=Pi/2
φ2(0)	=1
θ1(0)	=0
θ2(0)	=0
x(0)	=0
y(0)	=0
R=10	
ux=1	
uy=1	
m=5	
h=2	
Step size: t=0:0.2:100	

- It's not really clear, why the coin rolling from the centre to outside by the figure before [θ2(0)=0 and θ2(0)=1] it does the opposite?

5.4 θ2(0)=0

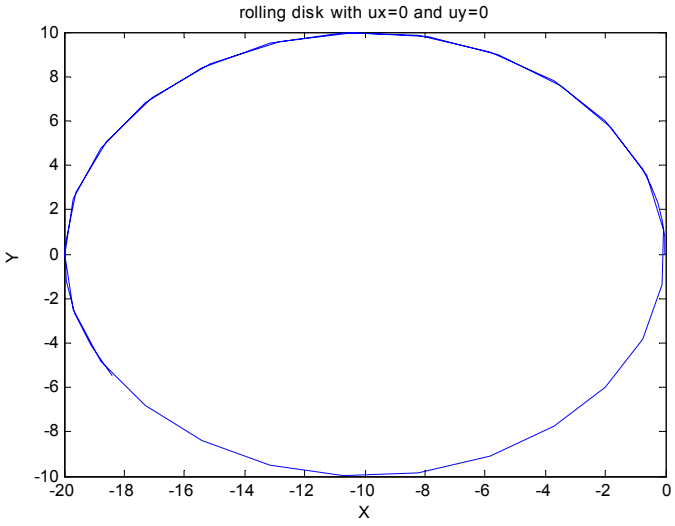


t	=0…100
φ1(0)	=Pi/2
φ2(0)	=1
θ1(0)	=0
θ2(0)	=0
x(0)	=0
y(0)	=0
R=10	
ux=1	
uy=1	
m=5	
h=2	

- If θ2=0 the coin turns in a smaller place round the vertical axes → the initial angulare velocity is zero.
- The coin is rolling from the centrum outward.

5.5 ode45 and ux=0 and uy=0

Plot the function “rolldisk.m” in 2D with the solver ode45 and ux=0 and uy=0



t	=0…10
φ1(0)	=Pi/2
φ2(0)	=1
θ1(0)	=0
θ2(0)	=1
x(0)	=0
y(0)	=0
R=10	
ux=0	
uy=0	
m=5	
h=2	

- If ux=0 and uy=0, the moments of inertia gives no force inside, it means the coin moves in the same circle.
  In this case an analytic solution of the equations can be obtained see section [5.6]

5.6 Comparison with the analytic solution in the case, with ux=0 and uy=0.

One can show that the analytic solution of this problem is:

$$x(t) = \frac{\Omega}{\omega} * R * (\sin(\omega * t + \varphi_0)) + x_0 \qquad y(t) = -\frac{\Omega}{\omega} * R * (\cos(\omega * t + \varphi_0)) + y_0$$

See for example [Nonholonomic mechanics and Control, A.M.Block, P.19]

Here $\dot{\varphi} = \omega$ and $\dot{\theta} = \Omega$ are constants.

Exact solution:

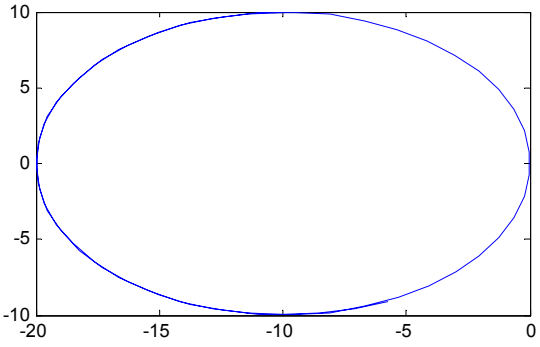
Matlab $t = [0,10],\ x_0 = 0,\ y_0 = 0$

```
%%%%%%%%%%%%%%%%%%%%
function f=rolldiskcon(t)
%
R=10;
a=1;
ph=pi;
w=1;
x=-10;
y=0;
%
f(1,1)=(a/w)*R*sin(w*t+ph)+x;
f(1,2)=-(a/w)*R*cos(w*t+ph)+y;
%%%%%%%%%%%%%%%%%%%%
```

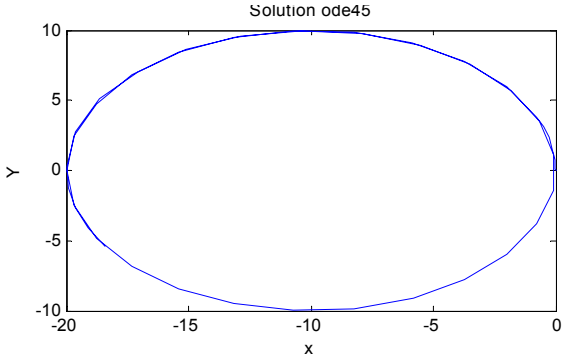
Comparing the exact solution with the numerical approximation from ode45:

```
%%%%%%%%%%%%%%%%%%%%
Function rolldisktime

t=0:0.147:10;
length=69;
z=zeros(length,2);
for j=1:length
   z(j,:)=rolldiskcon(t(j));
end
y1 = z(:,1);
y2 = z(:,2);
%%%%%%%%%%%%%%%%%%%%
```



Comparison ‘rolldisk’ with ‘rolldiscon’:




```
%%%%%%%%%%%%%%%%%%%%
```

y1-outf(:,5) ≈ 0 →OK
y2-outf(:,6) ≈ 0 →OK

```
%%%%%%%%%%%%%%%%%%%%
```

outf→ Vector from the file rolldisk [5]

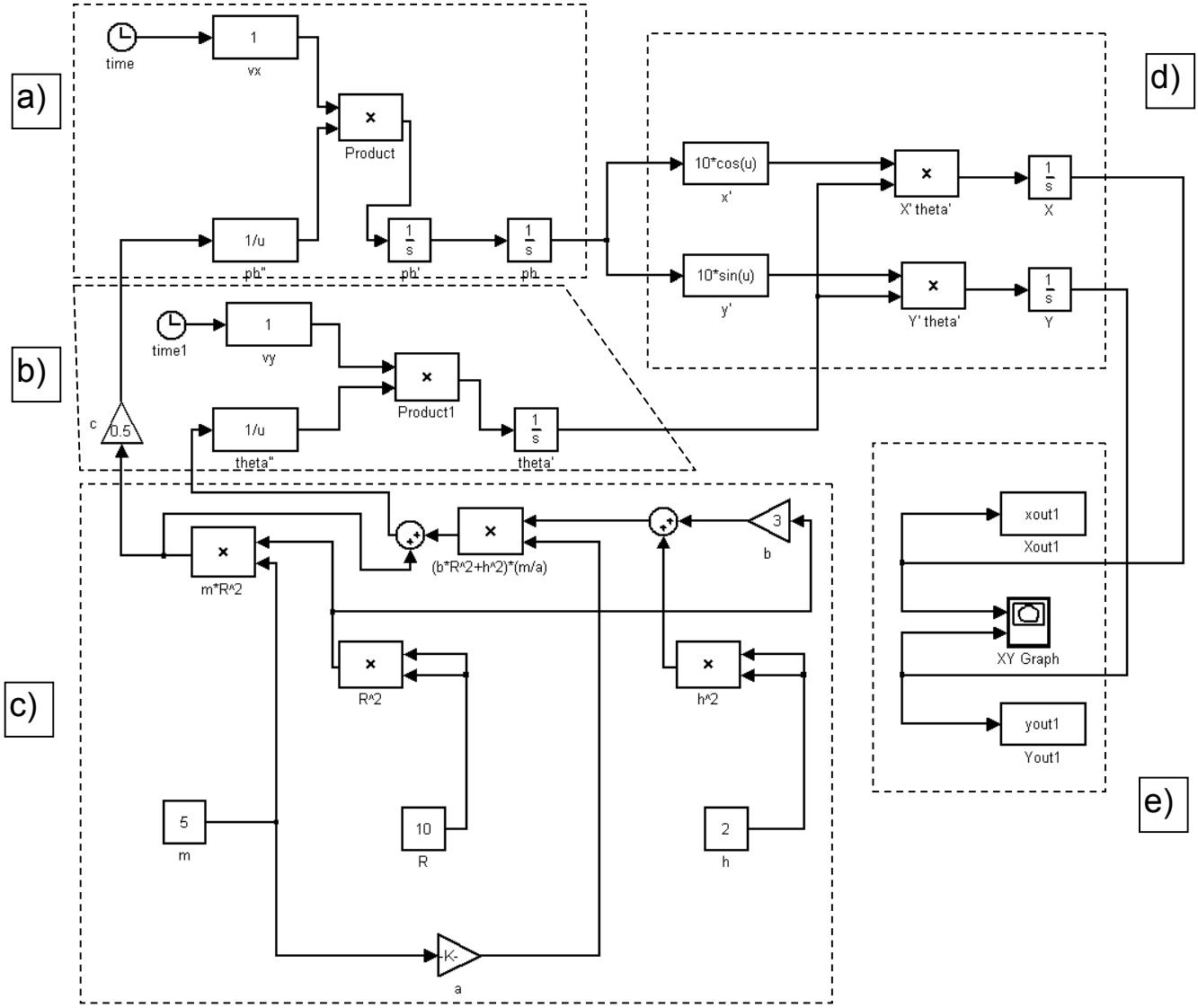
Comparing the analytic and numerical result we get the following errors
Time interval [0,10] , numerical integrations with Ode45

'RelTol'=1e-3,'AbsTol'=1e-6		'RelTol'=1e-13,'AbsTol'=1e-20	
X=y1-outf(:,5)	**Y=y2-outf(:,6)**	**X=y1-outf(:,5)**	**Y=y2-outf(:,6)**
1.0e-003 *	1.0e-003 *	1.0e-011 *	1.0e-011 *
0.00000000000000	-0.00000000000061	0	-0.0001
0.00000000000086	-0.00000000000036	0.0000	-0.0005
-0.00000000000013	-0.00000000000011	-0.0064	0.0151
-0.00000000000118	0.00000000000014	0.0419	-0.0615
-0.00000000000052	0.00000000000039	-0.0516	0.0493
0.00000000000001	-0.00000000000058	0.0790	-0.0519
0.00000000000067	-0.00000000000154	-0.0689	0.0249
-0.00000000000033	-0.00000000000029	0.1984	-0.0361
0.00000000000056	-0.00000000000126	-0.1409	-0.0069
0.00000000000055	-0.00000000000165	-0.0012	-0.0043
0.00000000000074	0.00000000000018	0.0169	0.0032
0.00000000000034	-0.00000000000022	0.2778	0.1972
0.00000000000141	-0.00000000000061	0.0263	0.0205
-0.00000000000100	-0.00000000000045	-0.0615	-0.1125
0.00000000000045	-0.00000000000013	0.0306	0.0712
-0.00000000000066	0.00000000000019	0.0004	-0.0262
-0.00000000000089	0.00000000000038	0.0025	0.0037
0.00000000000264	-0.00000000025489	-0.0124	0.0529
-0.00000000000011	-0.00000000004677	0.0412	-0.0824
-0.00000000000130	0.00000000020314	0.0483	-0.0643
0.00000000000072	-0.00000000000128	-0.1634	0.1375
0.00000003308892	-0.00000079401069	0.2794	-0.1604
0.00000000208063	-0.00000014132823	0.0382	-0.0151
-0.00000003490122	0.00000063483341	0.1327	-0.0179
-0.00000000094605	-0.00000000001177	0.1616	0.0107
0.00051301089542	-0.00242753353930	-0.0759	-0.0270
0.00003330697856	-0.00044482264094	0.0035	-0.0030
-0.00053891430996	0.00191307925768	-0.0512	-0.0485
-0.00001499272162	-0.00000090762065	-0.0313	-0.0473
0.58135703098916	-0.55814083236161	-0.0184	-0.0481
-0.55530570504203	0.34422882479390	0.0025	-0.0110
-0.01642999943918	-0.00662661269324	0.0042	-0.0106
0.76734519766930	0.18101678830362	0.0063	-0.0160
0.14039490854145	-0.03213321281947	0.0160	-0.0390
-0.01972372685088	-0.02401924829432	0.0368	-0.0620
0.24585534200128	0.73688900092339	0.0040	-0.0055
0.08647219647884	0.09416308660493	0.0045	-0.0069
-0.10164846347394	-0.67219377833280	0.0897	-0.0488
-0.00686793715232	-0.03618810545802	0.1800	-0.0533
-0.50365716768042	0.59840945676015	-0.1512	0.0007
-0.04893704064202	0.11702687078685	-0.1117	-0.0252
0.49428799885476	-0.45533558745170	-0.0290	-0.0220
0.01031781588523	-0.03194519302951	-0.2361	-0.1551
-0.79209398653468	-0.10710438991701	0.2149	0.1832
-0.14133817468220	0.01543728198605	-0.0309	-0.0642
0.63379396816732	0.16329663496784	-0.0256	-0.0859
0.01603303017816	-0.01519142512407	0.0245	0.0546
-0.35426763350177	-0.73100636211532	0.0110	-0.0049
-0.10577802913891	-0.11720468034992	0.0139	-0.0211
0.18860829962630	0.61493527660472	0	0.0211
0.00502316406248	-0.00133013869341	0.0092	-0.0036
0.40728636157739	-0.69968386385888		
0.02504956216565	-0.15894860775933		
-0.43196735618389	0.48434585318535		
-0.01258936233661	-0.00310533654613		
0.79239876771275	-0.04193465560398		
0.13086231514414	-0.07141532589827		
-0.65737860322912	-0.10840832166181		
-0.02061167344891	-0.01888490998248		
0.44699901475553	0.63750967395038		
0.11437647267698	0.06491746957238		
-0.28038338053094	-0.61835179319214		
-0.01166809345676	-0.03416119193689		
-0.02623412705205	0.03134681888306		
-0.01124768036576	-0.02174196256988		
0.00667460333759	-0.08489833538405		
-0.01084910134352	-0.03420818564770		

5.7 Simulink model

We modell the problem rollingdisk in Simulink. The results are similar to the analytic solution.

a) Integrator part of the variable φ with the control function ux.
b) Integrator part of the variable θ with the control function uy.
c) Input part of the variables
d) Integrator part of the X and Y function
e) Output in a Graph and in output Vectors (Matlab)



Abstract functions:

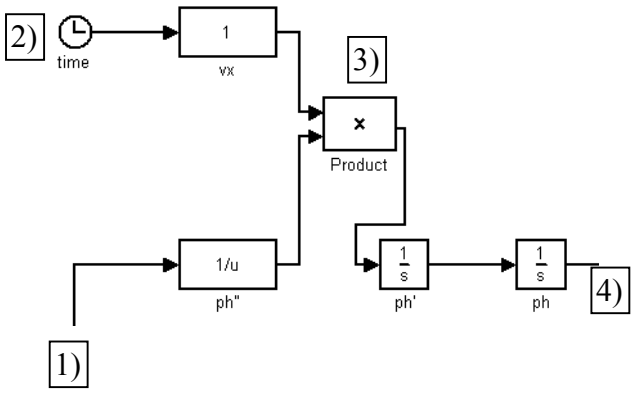
$$\dot{\varphi}_1 = \varphi_2$$
$$\dot{\varphi}_2 = J^{-1} * \upsilon_\varphi$$
$$\dot{\theta}_1 = \theta_2$$
$$\dot{\theta}_2 = (I + m * R^2)^{-1} * \upsilon_\theta$$
$$\dot{x} = R * \cos(\varphi_1) * \theta_2$$
$$\dot{y} = R * \sin(\varphi_1) * \theta_2$$

$$I = \frac{m}{12} * 3 * R^2 + h^2$$
$$J = \frac{1}{2} * m * R^2$$

5.8 Discussion Simulink

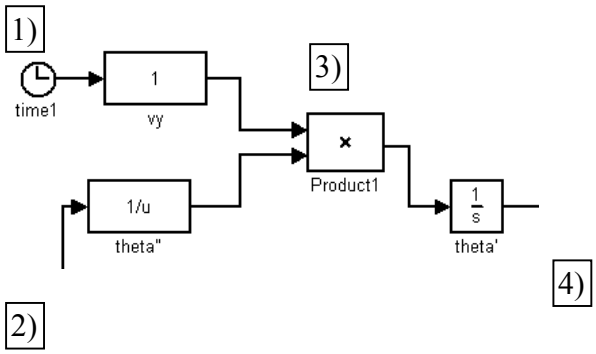
a)

1) The input is the variable u, which is setting with the function J and $\dot{\varphi}_2$.
2) Output the current simulation time and set it in by the control function v_φ.
3) Gives the product of $\ddot{\varphi}$ and v_φ
4) Output of φ after two integrators.
(given the initial conditions for φ and $\dot{\varphi}$)



b)

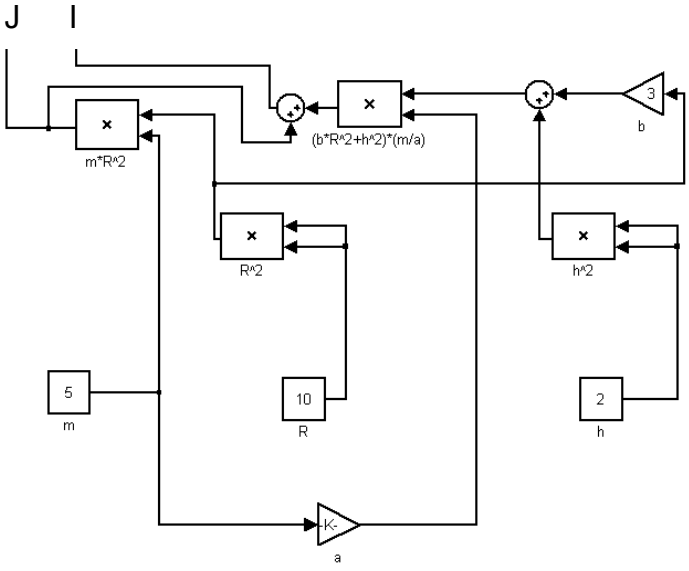
1) The input is the variable u, which is setting with the function I and $\dot{\theta}_2$
2) Output the current simulation time and set it in by the control function v_θ.
3) Gives the product of $\ddot{\theta}$ and v_θ
4) Output of θ after two integrators.
(given the initial condition for $\dot{\theta}$)



c)

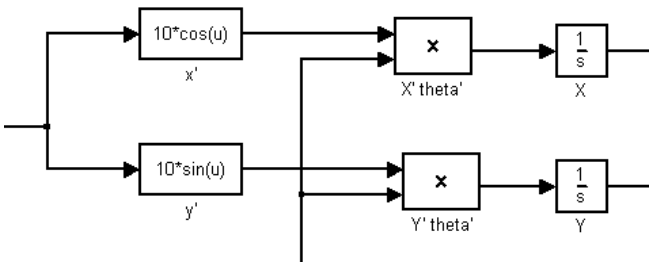
The Variabels m,R,h are setting. With the block Product and Sum build the two functions:

J and I



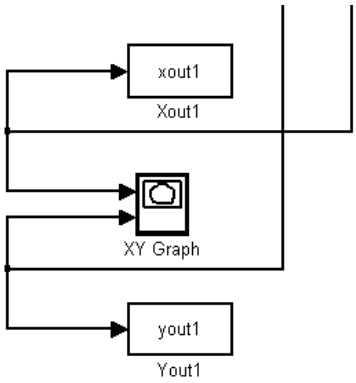
d)

The Function $\dot{x}$ and $\dot{y}$ times the θ_2 then integrate X and Y to the final solution.
(With x(0)=0 and y(0)=0 we could move the Start point in the coordinates.)



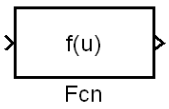
e)

The XY Graph gives an output of the coordinates. With Xout and Yout one obtains vectors to output the solution in Mat lab.



5.9 Block function

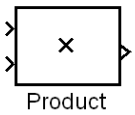
Apply a specified expression to the input. Use u as the input variable name



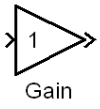
The Sum block performs addition or subtraction on its inputs.



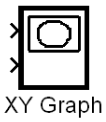
The Product block performs multiplication or division of its inputs.



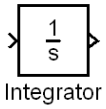
The Gain block multiplies the input by a constant value



The XY Graph block displays an X-Y plot of its inputs in a MATLAB figure



The Integrator block outputs the numerical solution of the diff. equation of its input at the current time step. Use u as the input variable name

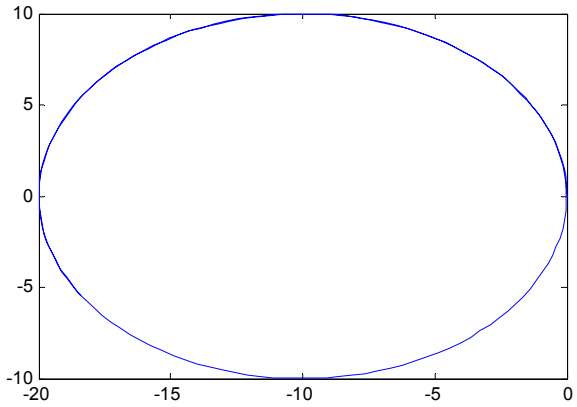


6. Compare different integration methods

I compare the different Ode's integrators with the same parameters. I want to see, if there is a difference between the integrators. The different between the analytic and numerical solution are compared in Matlab. The error is very small and it means the solvers give rather similar results. There are no differents between the pictures.
The error has the similar size witch we see in [3.2.3]

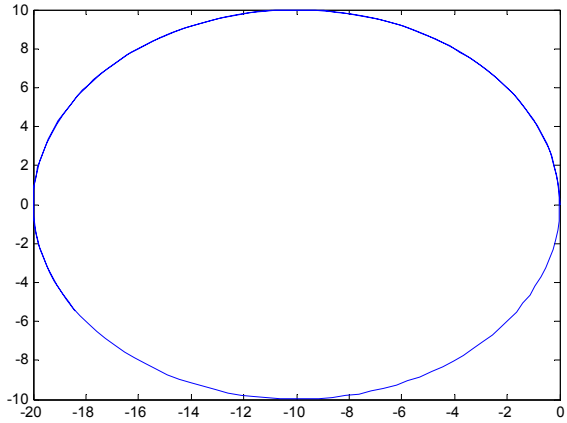
I chose a time interval t=0:0.05:10; → Vector field size 200
See for example [Nonholonomic mechanics and Control, A.M.Block, P.19]

Analytic




```
R=10;
a=1;
ph=pi/2;
w=1;
x=-10;
y=0;
y1=(a/w)*R*sin(w*t+ph)+x;
y2=-(a/w)*R*cos(w*t+ph)+y;
plot(y1,y2)
```

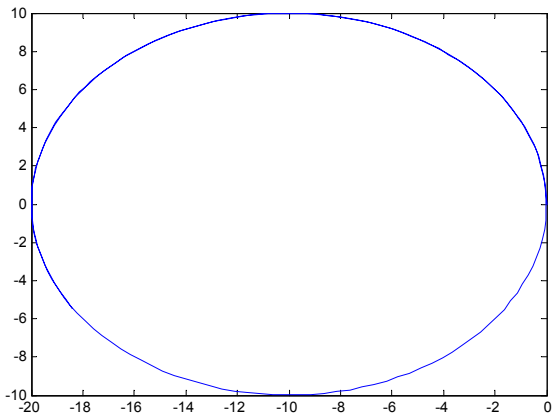
ode45 →'RelTol'=1e-13,'AbsTol'=1e-20



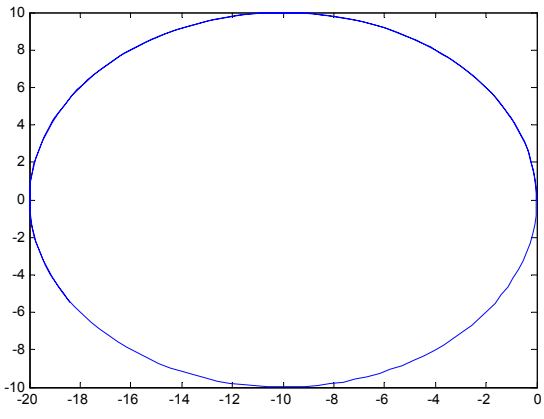
T	=0:0.05:10
φ1(0)	=Pi/2
φ2(0)	=1
θ1(0)	=0
θ2(0)	=1
x(0)	=0
y(0)	=0

R=10
ux=0
uy=0
m=5
h=2

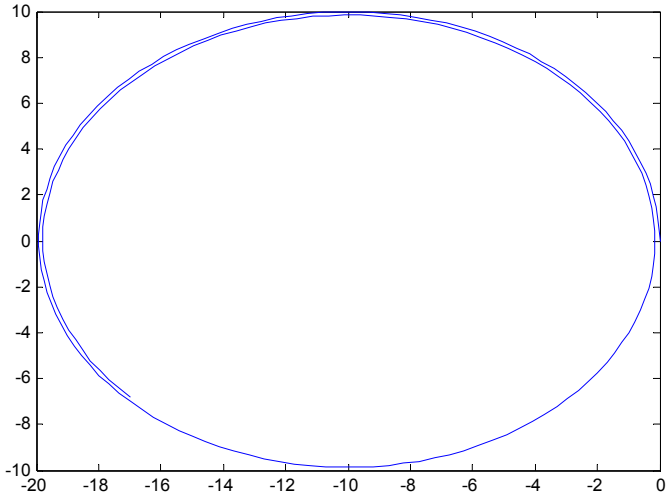
ode15s
→'RelTol'=1e-13,'AbsTol'=1e-20



ode23tb
→'RelTol'=1e-13,'AbsTol'=1e-20

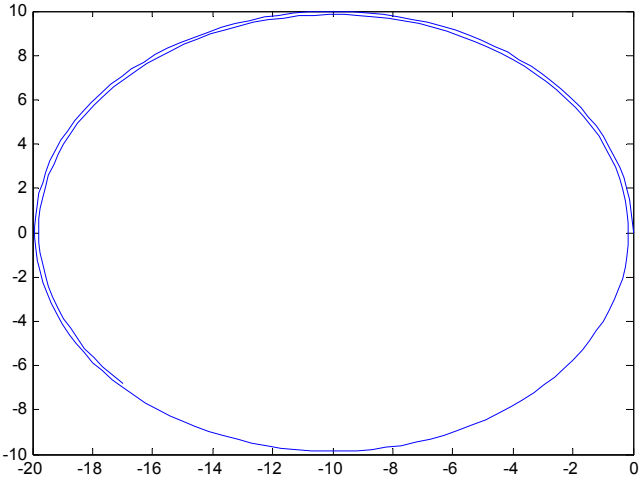


Plot the functions with ux=1 and uy=1
ode45 →'RelTol'=1e-13,'AbsTol'=1e-20

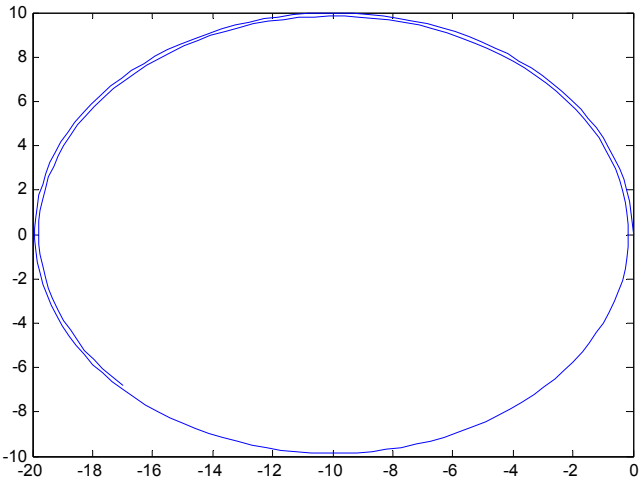


T	=0:0.05:10
φ1(0)	=Pi/2
φ2(0)	=1
θ1(0)	=0
θ2(0)	=1
x(0)	=0
y(0)	=0
R=10	
ux=1	
uy=1	
m=5	
h=2	

ode15s →'RelTol'=1e-13,'AbsTol'=1e-20

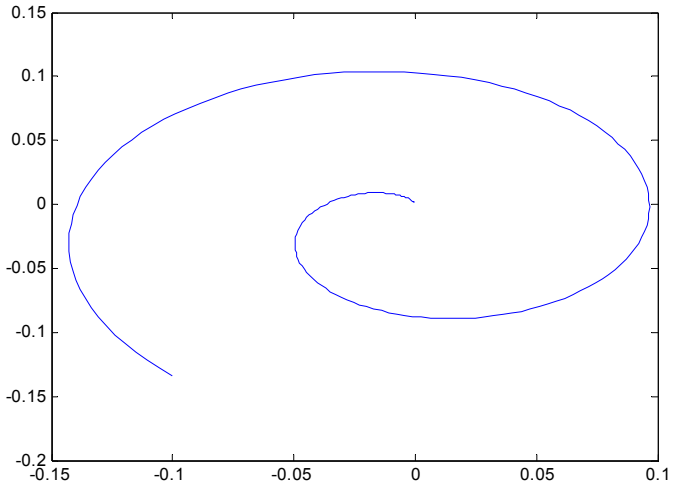


ode23tb →'RelTol'=1e-13,'AbsTol'=1e-20



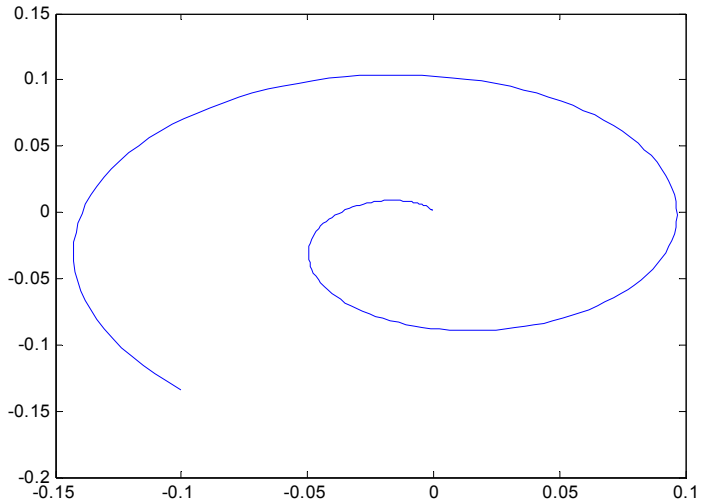
Plot the functions with $\theta 2(0)$ =0

ode45 →'RelTol'=1e-13,'AbsTol'=1e-20

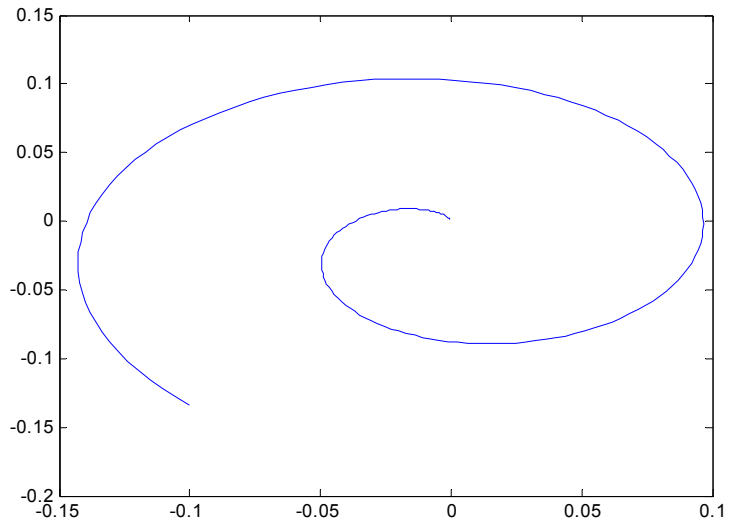


T	=0:0.05:10
$\varphi 1(0)$	=Pi/2
$\varphi 2(0)$	=1
$\theta 1(0)$	=0
$\theta 2(0)$	=0
x(0)	=0
y(0)	=0
R	=10
ux	=1
uy	=1
m	=5
h	=2

ode15s →'RelTol'=1e-13,'AbsTol'=1e-20



ode23tb →'RelTol'=1e-13,'AbsTol'=1e-20



7. The variational controlled System

These equations define the dynamics. The Euler-Lagrange equations are including external forces. See for example [Nonholonomic mechanics and Control, A.M.Block, P.20]

Dynamic Equations

$$J * \ddot{\varphi} = \upsilon_{\varphi} + R * \dot{\theta}(a * \sin(\varphi) - b * \cos(\varphi))$$

$$(I + m * R^2) * \ddot{\theta} = \upsilon_{\theta} + R * \dot{\varphi}(-a * \sin(\varphi) + b * \cos(\varphi))$$

The constraints

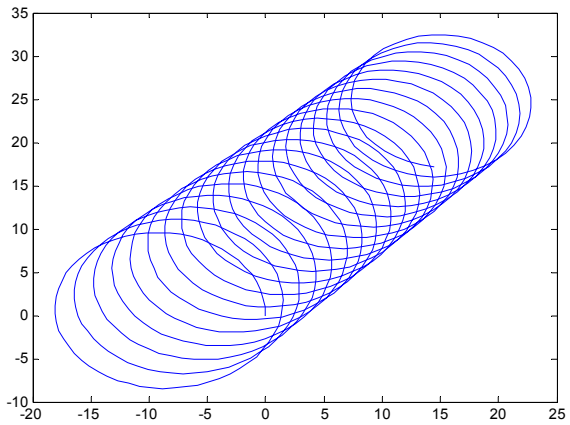
$$\dot{x} = R * (\cos(\varphi)) * \dot{\theta}$$

$$\dot{y} = R * (\sin(\varphi)) * \dot{\theta}$$

Where a and b are integration constants and they are not determined by the constraints or initial data. If a and b are zero, we obtain the same equation witch we have in [5.5]

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function f=rolldiskkin(t,u)
%
R=10;                   %Radius
ux=1;                   %start Function Phi
uy=1;                   %start function Omega
m=5;                    %mass
h=2;                    %broadness disk
a=1;
b=1;
J=(1/2)*m*(R^2);            %rotation axes
I=(m/12)*(3*(R^2)+(h^2));   %perpendicular axes trough barycenter
%
f(1,1)=u(2,1);
f(2,1)=(J^-1)*(ux+R*u(4,1)*(a*sin(u(1,1))-b*cos(u(1,1))));
%
f(3,1)=u(4,1);
f(4,1)=((I+m*(R^2))^-1)*(uy+R*u(2,1)*(-a*sin(u(1,1))+b*cos(u(1,1))));
%
f(5,1)=R*cos(u(1,1))*u(4,1);
f(6,1)=R*sin(u(1,1))*u(4,1);
%
%ph=u(1,1)
%ph'=u(2,1)
%th=u(3,1)
%th'=u(4,1)
%x=u(5,1)
%y=u(6,1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

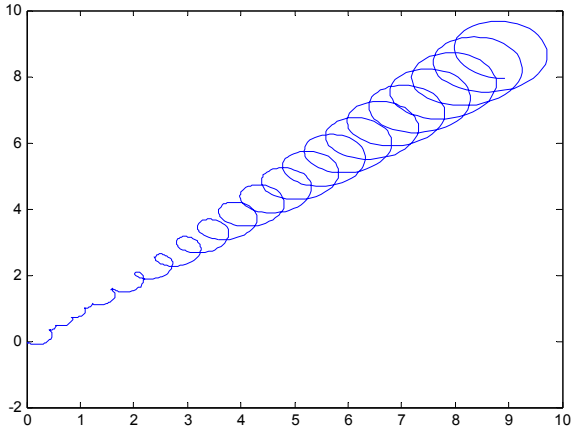
7.1 ode45 and a=1 and b=1



t	=0…100
φ1(0)	=Pi
φ2(0)	=1
θ1(0)	=0
θ2(0)	=1
x(0)	=0
y(0)	=0
R=10	
ux=1	
uy=1	
a=1	
b=1	
m=5	
h=2	

- The coin starts in a circle move and goes up in the x and y axes
- The Spiral has a constant size, it means a and b are constant only the position move

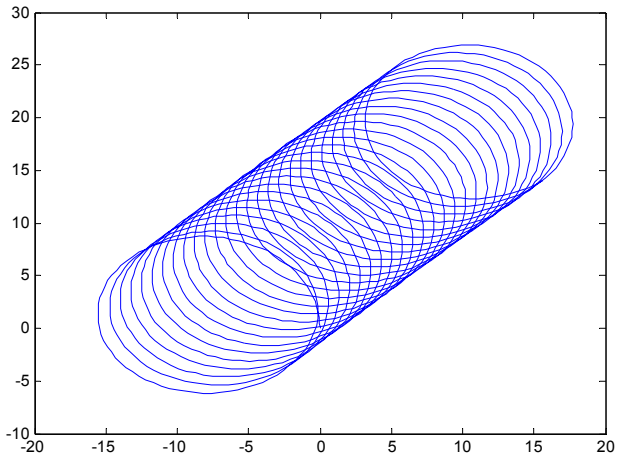
7.2 θ2(0)=0



t	=0…100
φ1(0)	=Pi
φ2(0)	=1
θ1(0)	=0
θ2(0)	=0
x(0)	=0
y(0)	=0
R=10	
ux=1	
uy=1	
a=1	
b=1	
m=5	
h=2	

- If it's the θ2(0) the coin start with a little move and the way isn't a round spiral
- In a while it goes round and make spiral moves and the Radius goes Bigger

7.3 With function ux=exp(-t+5) and uy=exp(-t+5)

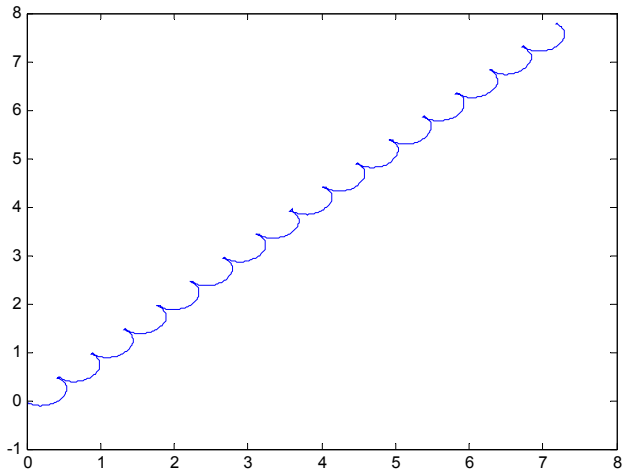


t	=0…100
φ1(0)	=Pi/2
φ2(0)	=1
θ1(0)	=0
θ2(0)	=1
x(0)	=0
y(0)	=0

R=10
ux= exp(-t+5)
uy= exp(-t+5)
a=1
b=1
m=5
h=2

- With set a function in ux and uy the spiral take in his turns

7.4 With function ux=exp(-t+5) and uy=exp(-t+5) and θ2(0)=0



t	=0…100
φ1(0)	=Pi/2
φ2(0)	=1
θ1(0)	=0
θ2(0)	=0
x(0)	=0
y(0)	=0

R=10
ux= exp(-t+5)
uy= exp(-t+5)
a=1
b=1
m=5
h=2

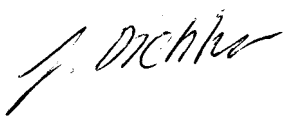
- If there are setting the varable θ2(0)=0 too, the spiral doesn't move in a round circle.

8. Bibliography

Documentation:	Microsoft Office Word Adobe PDF
Math. Solver:	Matlab Ver. 6.5
Books:	Nonholonomic Mechanics and Control; A.M.Bloch ISBN: 0-387-95535-6
	Classical Mechanics; Goldstein Poole& Safko ISBN: 0321-188977
	Modeling and Simulation for Automatic Control; Olav Egeland and Jan Tommy Gravdahl ISBN: 82-92356-01-0
	Flexible Multibody Dynamics; Michel Geradin Alberto Cardona ISBN:0-471-48990-5

Internet links:

http://www.ma.hw.ac.uk/~simonm/conf/tony.pdf
http://www.cds.caltech.edu/mechanics_and_control
http://www.laas.fr/~jpl/book-toc.html
http://www.physics.gatech.edu/people/faculty/flannery/publications/AJP73_March2005_265-272.pdf

Signatur: 

Date: 9/16/2005 Trondheim