Symbolic Sliding Mode Control Package

Ali ASSAF

Summer School 2015

Contents

Contents

a - Goals of the ChaSLIM project

- Design new sliding mode controllers avoiding the chattering phenomena.
- Using **SymPy** language to develop a symbolic software package based on NOLIACPA (IRCCyN software).
- Synthesize controllers and simulate the closed-loop system.

a.1 - NOLIACPA and ChasLIM

- NOLIACPA: a symbolic **NO**n**LI**near **A**nalysis and **C**ontrol **PA**ckage for nonlinear systems using **Maple**.

- Structural Analysis of nonlinear systems
  - Relative degrees
  - Zeros at infinity
  - Nonlinear Interactor (Inversion algorithm)
  - Essential orders

a.1 - NOLIACPA and ChasLIM

- Application to nonlinear systems Control :
  - Decoupling control via Dynamic Compensator,
  - State and input-output Linearization via feedback,
  - Observers design via exact transformation to linear system and input-output injection.

b - Introduction to SymPy

- **SymPy** is a Python library for symbolic computation that aims to become a full-featured computer algebra system.
- **SymPy** is entirely written in Python and does not require any external libraries.
- http://sympy.org/fr/index.html

b.1- Sympy vs Maple vs Matlab: Some examples

- **SymPy** is a cost free open source, while **Maple** *and* **Matlab** are proprietary softwares released under proprietary commercial licenses.
- **Maple** and **Matlab** come with both a GUI and a command line interface.
- **SymPy** has a command line interface. Plotting with **SymPy** needs to install others libraries.

b.1- Sympy vs Maple vs Matlab: Some examples

- In **SymPy**, to raise something to a power, we use **, **Maple** and **Matlab**: ^.
- In **SymPy**, we have to define symbols before using them.
- **Differentiation :** diff(f(x), x)

- **Integration : SymPy** integrate(f(x), x)**,**
  Maple and **Matlab** int(f, x)

b.1- Sympy vs Maple vs Matlab: Some examples

- **Algebraic equations :** solve (f(x),x)
- **simplify()** to arrive at the simplest form of an expression.
- **expand()** to expand polynomial expressions.
- **Linear Algebra** :

  M = Matrix(2, 3, [1, 2, 3, 4, 5, 6])

  M.det() **SymPy**

  det(M) **Matlab**

  determinant(M) **Maple**

c - Sliding Mode Control of Order 1 based on Input-Output Feedback Linearization

- Consider the nonlinear system

$$\sum \begin{cases} \dot{x} = f(x) + g(x)u \\ y = h(x) \end{cases} \qquad (1)$$

- The main goal of the sliding mode control is to force the system dynamics to slide on a desired manifold (Sliding surface) in order to robustly follow a given trajectory.

c - Sliding Mode Control of Order 1 based on Input-Output Feedback Linearization

- A particular choice for $S(x,t)$

$$S(x,t) = \sum_{i=0}^{r-1} \Lambda_i e^i \tag{2}$$

- r is the relative degree of ouput $y(t)$.
- Tracking error $e(t) = y_d(t) - y(t)$
- Λ_i tuning parameters, positives.

- Candidate Lyapunov function

$$V(x) = \frac{s^2(x,t)}{2} \tag{3}$$

c - Sliding Mode Control of Order 1 based on Input-Output Feedback Linearization

- Time derivative of *S(x,t):*

$$\dot{S}(x,u,t) = \sum_{i=0}^{r-1} \Lambda_i e^{i+1} = \frac{\partial S}{\partial t} + \frac{\partial S}{\partial x}(f(x) + g(x)u) \quad (4)$$

$$= S_1(x,t) + S_2(x)u$$

- *Control law:* $u = u_{eq} + \upsilon_n$ (5)
- *Equivalent Control:* $u_{eq} = S_2(x)^{-1}[-S_1(x,t)]$
- *Discontinuous Control:* $\upsilon_n = S_2(x)^{-1} u_n$

  where $u_n = -K\,\mathrm{sgn}(S), \quad K > 0$

- $\dot{V} = S\dot{S} = -K * S * \mathrm{sgn}(S) = -K\|S\| < 0$ (6)

c.1 - Example of Nominal System

```
The considered system is :
dx1/t = u1 + u2*x1(t) + x2(t)
dx2/t = x3(t)
dx3/t = u2 + x1(t) + x2(t)**2
y1 = x1(t)
y2 = x2(t)

the vector of relative degrees :  Matrix(  1 ,  2  )


The sliding variables Sy1(x,t) of output y1 :

-x1(t) + y1ref(t)


The sliding variables Sy2(x,t) of output y2 :


-k20*x2(t) + k20*y2ref(t) - x3(t) + Derivative(y2ref(t), t)]
```

c.1 - Example of Nominal System

Time derivative of S(x,t) = S1N + S2N * U

S1N :

```
Matrix([
[                                                                  -x2(t) + Derivative(y1ref(t), t)],
[-k20*x3(t) + k20*Derivative(y2ref(t), t) - x1(t) - x2(t)**2 + Derivative(y2ref(t), t, t)]])
```

S2N :

```
Matrix([
[-1, -x1(t)],
[ 0,     -1]])
```

First order Sliding Mode Control U = ueq + Vn :

U[1] = -K1*sign(x1(t) - y1ref(t)) + K2*x1(t)*sign(k20*x2(t) - k20*y2ref(t) + x3(t) - Derivative(y2ref(t), t)) + (k20*x3(t) - k20*Derivative(y2ref(t), t) + x1(t) + x2(t)**2 - Derivative(y2ref(t), t, t))*x1(t) - x2(t) + Derivative(y1ref(t), t)

with K1>0

U[2] = -K2*sign(k20*x2(t) - k20*y2ref(t) + x3(t) - Derivative(y2ref(t), t)) - k20*x3(t) + k20*Derivative(y2ref(t), t) - x1(t) - x2(t)**2 + Derivative(y2ref(t), t, t)

with K2>0

c.2 - Case of Unknown Parameters System

- In the real case, the system includes uncertainties in the model to be controlled.

$$\sum \begin{cases} \dot{x} = f(x) + \Delta f(x) + g(x)u + \Delta g(x)u \\ y = h(x) + \Delta h(x) \end{cases} \quad (7)$$

$\Delta f(x)$, $\Delta g(x)$ and $\Delta h(x)$ are bad known but bounded.

c.2 - Case of Unknown Parameters System

- $u = S_{2N}(x)^{-1}[-S_{1N}(x,t) - k * sign(S)]$ **(Nominal System Control)** (8)

- $\dot{S}_{\Delta}(x,t) = S_{1\Delta} + S_{2\Delta} * S_{2N}(x)^{-1}[-S_{1N}(x,t) - k * sign(S)]$ (9)

- In MIMO case, from the Lyapunov function, we obtain p inequations with p variables, which are complicated to be solved.

- The gain k_j *(j=1,…,p)* depends on the uncertainties and initial conditions at the same time.

c.2 - Case of Unknown Parameters System : a proposed method

- Limitation on inputs :

$$|u| \leq U_{max}$$

if S > 0 :

$$u = -U_{max} = u_{eq} - \frac{K_{max}}{S_2} \Rightarrow K_{max} = S_2 * U_{max} + S_2 * u_{eq} \quad (10)$$

if S < 0 :

$$u = +U_{max} = u_{eq} - \frac{K_{max}}{S_2} \Rightarrow K_{max} = S_2 * U_{max} - S_2 * u_{eq} \quad (11)$$

∀ S :

$$K_{max} = S_2 * U_{max} + S_2 * u_{eq} * sgn(S) \quad (12)$$

c.2 - Case of Unknown Parameters System : a proposed method

- MIMO case :

$$\begin{bmatrix} K_{1max} \\ \vdots \\ K_{nmax} \end{bmatrix} = [S_2] * \begin{bmatrix} U_{1max} \\ \vdots \\ U_{nmax} \end{bmatrix} + [S_2] * \begin{bmatrix} u_{eq1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & u_{eqn} \end{bmatrix} * \begin{bmatrix} sgn(S_{y1}) \\ \vdots \\ sgn(S_{yn}) \end{bmatrix} \quad (13)$$

c.2 - Case of Unknown Parameters System : a proposed method

The uncertain system is :
dx1/t = u1*(1.1*x1(t) + 1) + x2(t)
dx2/t = x3(t)
dx3/t = u2 + 0.9*x1(t) + x2(t)**2
y1 = 1.3*x1(t)
y2 = x2(t)

-- Calculation of the vector of relative degrees --
The system is written as diff(y,r)=A0+B0u
with vdeg the vector of relative degrees : Matrix([[1], [2]])
The matrix A0 is : Matrix([[x2(t)], [x1(t) + x2(t)**2]])
The matrice B0 is : Matrix([[x1(t) + 1, 0], [0, 1]])
The sum of relative degrees is : 3
---

c.2 - Case of Unknown Parameters System : a proposed method

The static state feedback control laws which achieve input-output decoupling diff(y,r)=v are :

u1=(v1 - x2(t))/(x1(t) + 1)

u2=v2 - x1(t) - x2(t)**2

First order Sliding Mode Control U = ueq + Vn :

U[1] = (-K1*sign(x1(t) - sin(t)) - x2(t) + cos(t))/(x1(t) + 1)

with K1 = -Umax1*(x1(t) + 1) - (x2(t) - cos(t))*sign(x1(t) - sin(t))

U[2] = -K2*sign(k20*x2(t) - k20*sin(t) + x3(t) - cos(t)) - k20*x3(t) + k20*cos(t) - x1(t) - x2(t)**2 - sin(t)

with K2 = -Umax2 - (k20*x3(t) - k20*cos(t) + x1(t) + x2(t)**2 + sin(t))*sign(k20*x2(t) - k20*sin(t) + x3(t) - cos(t))

---

d - Definition of Second Order Sliding Mode Control (SOSM)

- Using a high order mode control to reduce chattering.
- As a first time, the control law u is calculated as for the first order sliding mode control.

d - Definition of Second Order Sliding Mode Control (SOSM)

- u_{ni} (*i=1..m*), the discontinuous controls signals act on the second time derivate of the sliding variable $\ddot{S}$ to enforce $\dot{S} = 0$ and $S = 0$.

- Candidate Lyapunov function :

$$V(x) = \frac{1}{2} * S(x,t)^2 + \frac{1}{2} * \dot{S}(x,t)^2 \qquad (14)$$

$$\dot{V} = S\dot{S} + \dot{S}\ddot{S} = \dot{S}(S + \ddot{S}) \qquad (15)$$

d.1 - SOSM Sympy example of system SISO with relative degree equals to 1

- Second order Sliding Mode Control :

$$\dot{u}_n = \begin{cases} -\propto_M sgn(S) & when\ \dot{S} * S > 0 \\ -\propto_m sgn(S) & when\ \dot{S} * S < 0 \end{cases} \quad (16)$$

- Control law :

$$u = \begin{cases} -S_{1N}(x,t) * S_{2N}(x)^{-1} - \int \propto_M sgn(S)dt * S_{2N}(x)^{-1}, \dot{S} * S > 0 \\ -S_{1N}(x,t) * S_{2N}(x)^{-1} - \int \cdot \propto_m sgn(S)dt * S_{2N}(x)^{-1}, \dot{S} * S < 0 \end{cases} \quad (17)$$

d.1 - SOSM Sympy example of system SISO with relative degree equals to 1

- Consider the following SISO system with relative degree equal to 1 :

$$\sum \begin{matrix} \dot{x} = x^2 + u + d(t) \\ y = x \end{matrix} \quad (18)$$

- The sliding variables $S_y(x,t)$ of output y :

$$S_y = y_{ref}(t) - y = \sin(t) - x(t) \quad (19)$$

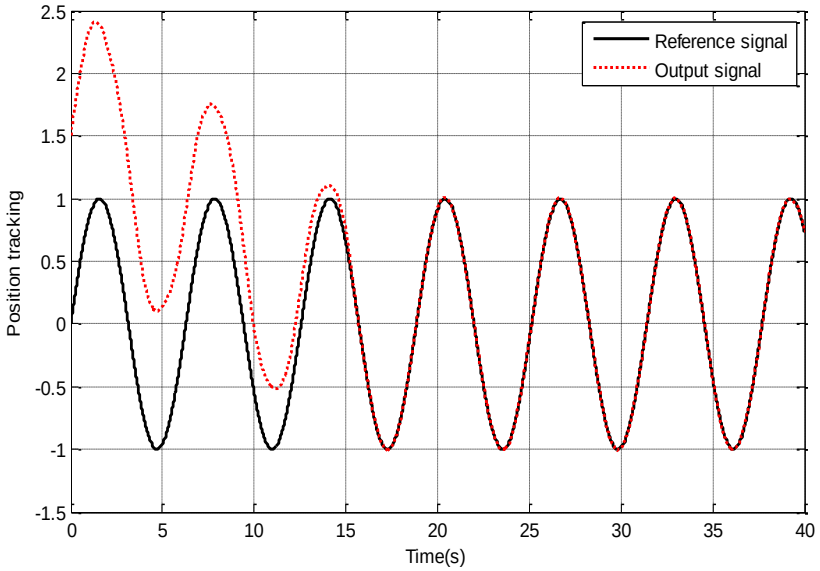
- The time derivative of $S_y(x,t)$:

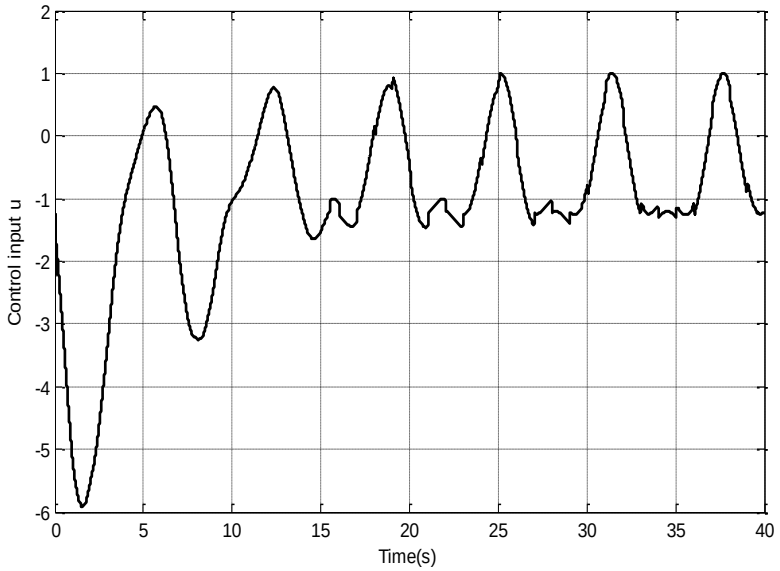
$$\dot{S}_y = \dot{y}_{ref}(t) - x^2 - u = \cos(t) - x^2 - u \quad (20)$$

d.1 - SOSM Sympy example of system SISO with relative degree equals to 1

- Control law :

$$u = \begin{cases} \cos(t) - x^2 + \propto_M * \int sgn(\sin(t) - x(t))dt, & \dot{S} * S > 0 \\ \cos(t) - x^2 + \propto_m * \int sgn(\sin(t) - x(t))dt, & \dot{S} * S < 0 \end{cases} \quad (21)$$





d.2 - SOSM Sympy example of system SISO with the relative degree equals to 2

- Consider the following SISO system with relative degree equal to 2 :

$$\sum \quad \begin{aligned} \dot{x}_1 &= x_2 + d(t) \\ \dot{x}_2 &= x_2^2 + u \\ y &= x_1 \end{aligned} \tag{22}$$

- The sliding variables $S_y(x,t)$ of output y :

$$S = \dot{y}_{ref}(t) - \dot{y}(t) + \Lambda_0 \left(y_{ref} - y\right) = \cos(t) - x_2(t) + \Lambda_0 \left(\sin(t) - x_1\right) \tag{23}$$

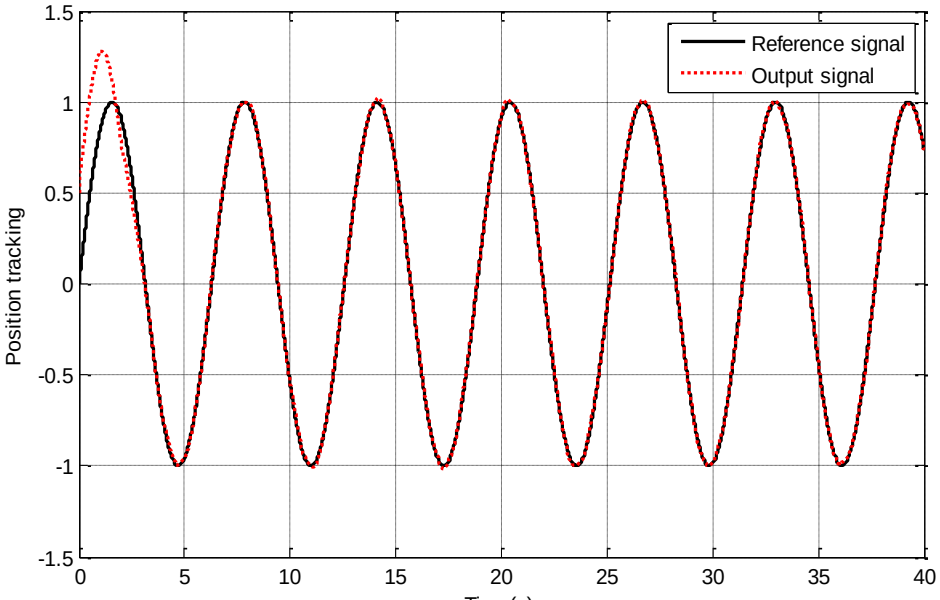
- The time derivative of $S_y(x,t)$:

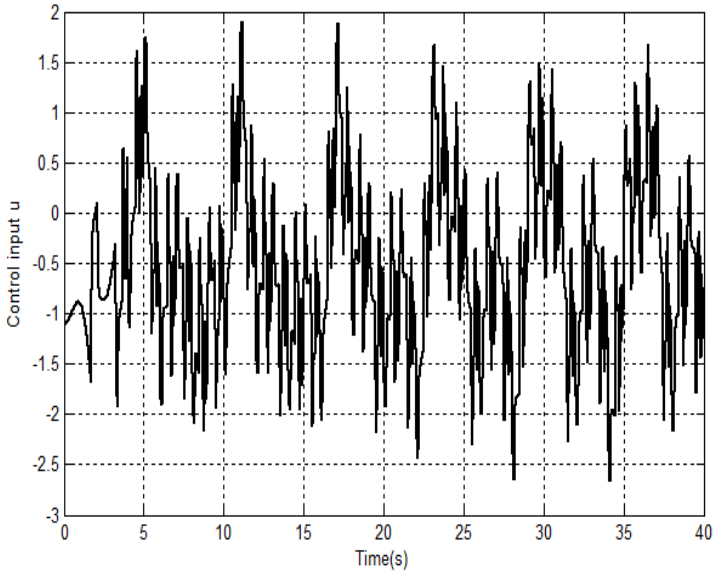
$$\dot{S} = \ddot{y}_{ref} - \dot{x}_2 + \Lambda_0(\dot{y}_{ref} - \dot{y}) = -\sin(t) - x_2^2 - u + \Lambda_0 \left(\cos(t) - x_2\right) \tag{24}$$

d.2 - SOSM Sympy example of system SISO with relative degree equals to 2

- Control law :

$$u = \begin{cases} -\sin(t) - x_2^2 - u + \Lambda_0 \left(\cos(t) - x_2\right) + \propto_M * \int sgn(S)dt, \dot{S} * S > 0 \\ -\sin(t) - x_2^2 - u + \Lambda_0 \left(\cos(t) - x_2\right) + \propto_m * \int sgn(S)dt, \dot{S} * S < 0 \end{cases} \quad (25)$$





e - System Dynamics Simulation using Python

- **SciPy** is a package of tools for science and engineering for Python. It includes modules for ODE solvers, statistics, optimization, integration, linear algebra, Fourier transforms, signal and image processing, and more…
- **matplotlib** is a python 2D ploting library.
- **NumPy** is a fundamental package for scientific computing with Python.
- other libraries …

e.1 - Fourth Order Runge-Kutta Method in Python

- Runge-Kutta method is a mathematical algorithm used to solve systems of ordinary differential equations.

ode(sys).set_integrator('dopri5',atol=1.e-6,rtol=1.e-3)

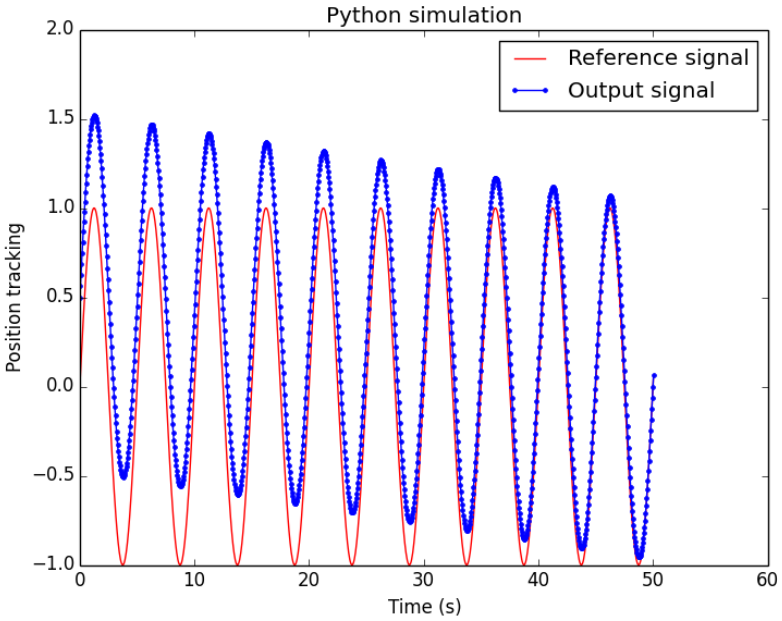
ode(sys).set_integrator(‘vode',atol=1.e-6,rtol=1.e-3)

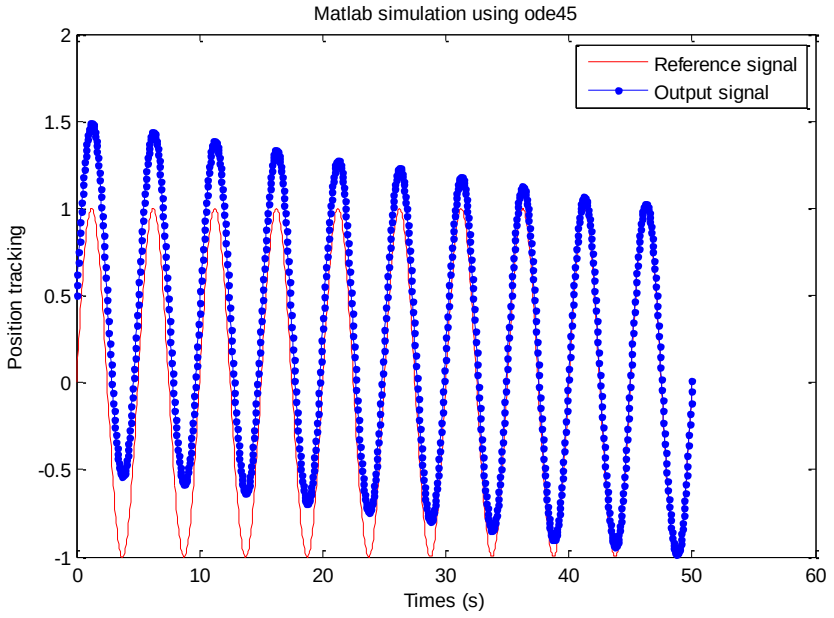
odeint(sys, x0, t)

e.2 – Some Examples

- Matlab & Scipy Comparison

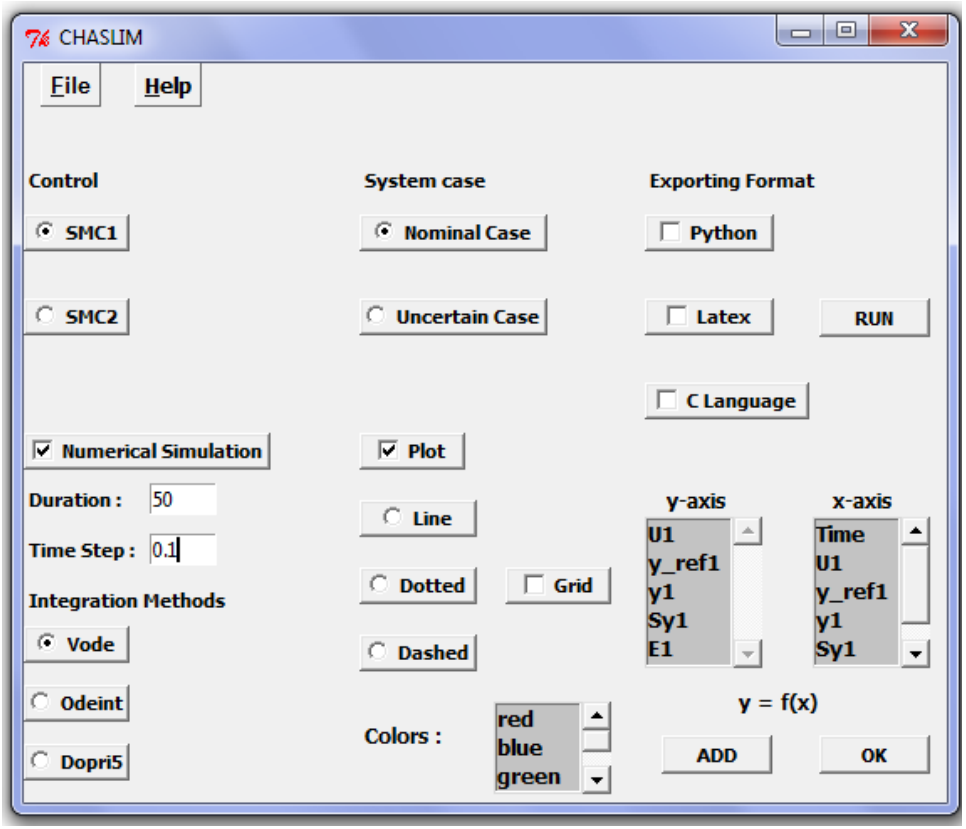
$$\sum \begin{matrix} \dot{x} = x + u \\ y = x \end{matrix} \quad \text{and} \quad y_{ref} = \sin(wt)$$





f- ChaSlim Graphical User Interface

- **cx_Freeze**, **py2app, py2exe** Convert Python scripts into executable programs



g - Conclusion

- This package needs to be extended to include new methodologies of High Order Sliding Mode (HOSM) controllers, in order to reduce the chattering and improve the disturbance rejection and which can also be applied to non-smooth systems.

- The Symbolic Sliding Mode Control Package provides also the simulation results of the closed loop system. This is convenient for the user to help him for the gain k tuning (with respect to the limitation of the input, ...).

- Graphical User Interface to facilitate the use of ChaSlim package.