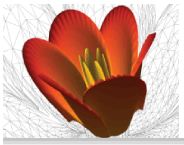


IMS 2008 June 20-24th International Mathemati

# Solving Differential Equations under Uncertainty

*Guaranteed Numerical Integration of Nonlinear Parametric ODEs*

**Yves Papegay**

**Nacim Ramdani**

INRIA - Sophia Antipolis Méditerrannée and LIRMM CNRS Univ. Montpellier 2.
nacim.ramdani@inria.fr, yves.papegay@inria.fr

**In this talk, we present a Mathematica implementation of algorithms which compute reachable space for nonlinear dynamical systems in presence of model uncertainty. These algorithms will be described through an example of a biological process, discussing the quality of the results and the efficiency for getting them.**

# ■ Non-Linear Dynamical Systems

## □ Generic Case

$$\left\{\dot{\mathrm{x}}(t) = f(\mathrm{x}, \mathrm{p}, t),\ \ \mathrm{x}(t_0) \in \mathrm{X}_0 \subseteq \mathrm{D},\ \ \mathrm{p} \in \mathrm{P}\right\} \tag{1}$$

## □ Example

Haldane model of a biotechnological process in a stirred reactor, addressing the existence of one specy on a chemostat with a single substrate

$$\begin{cases} \dot{x} &=& f_x(x, s) &=& \left(\mu_0 \dfrac{s}{s + k_s + s^2 / k_i} - \alpha\, d\right) x \\ \dot{s} &=& f_s(x, s) &=& -k\, \mu_0 \dfrac{s}{s + k_s + s^2 / k_i}\, x + (s_{in} - s)\, d \end{cases} \tag{2}$$

$x$ : biomass density,
$s$ : substrate concentration,
$d$ : dilution rate of the chemostat,
$s_{in}$ : concentration of input substrate given by $s_{in}(t) = s_{in}^0 + 15\cos(1/5\ t)$.

$\mu_0$ and $s_{in}^0$ are assumed uncertain.

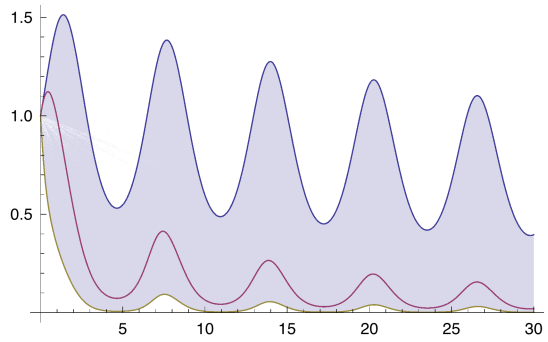
## ■ Reachable Set

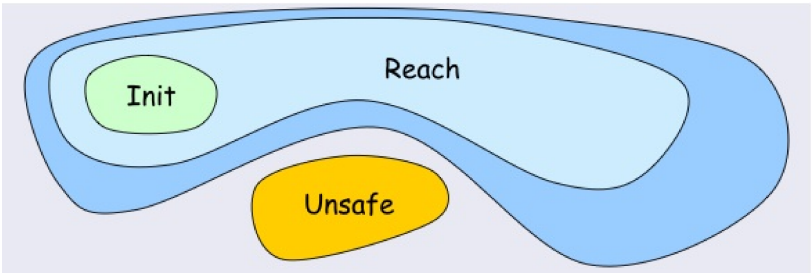
$$\left\{\dot{x}(t) = f(x, p, t),\ \ x(t_0) \in X_0 \subseteq D,\ \ p \in P\right\} \tag{3}$$

The reachable space is the set of all feasible solutions for the differential equation when initial state vector is taken in domain $X_0$ and parameter vector in domain P

$$R([t_1, t_2]; X_0) = \left\{x(\tau),\ t_1 \le \tau \le t_2 \ \middle|\ \left(\dot{x}(\tau) = f(x, p, \tau)\right) \wedge (x(t_0) \in X_0) \wedge (p \in P)\right\} \tag{4}$$

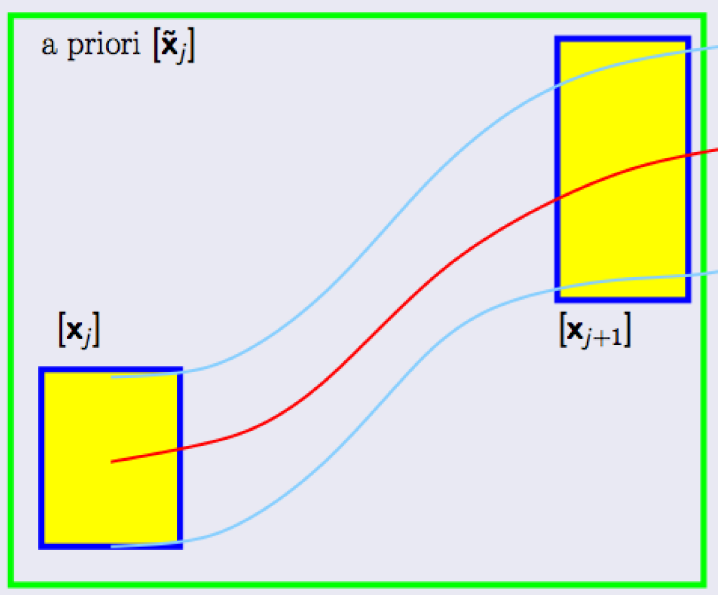
```
s[α_] := NDSolve[
    {y'[x] == α y[x] Cos[x + α^(3) y[x]], y[0] == 1}, y, {x, 0, 30}]
Plot[Evaluate[{y[x] /. s[0.5], y[x] /. s[1.], y[x] /. s[1.5]}],
  {x, 0, 30}, PlotRange → All, Filling → {1 -> {3}}]
```



## ■ Set Integration with Interval Taylor Models

- define a time grid $t_0 < t_1 < t_2 < \ldots < t_{n_T}$
- objective is to compute interval vectors $[x_j]$, $j = 1, \ldots, n_T$, that are *guaranteed* to contain the solution at time $t_j$




1. Fixed point theorem and Picard–Lindelöf operator, compute an *a priori* enclosure $[\tilde{x}_j]$

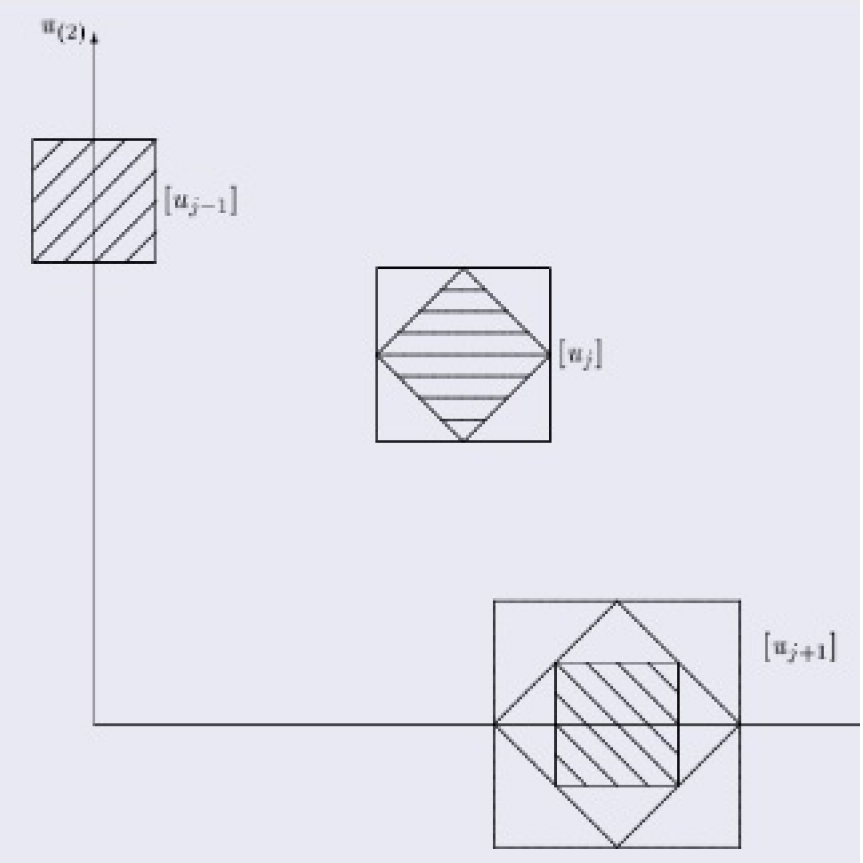
$$\forall\, t \in [t_j,\, t_{j+1}]\; x(t) \in [\tilde{x}_j] \tag{5}$$

2. a tighter enclosure $[x_{j+1}]$ of the solution is based on Taylor expansion

$$[x_{j+1}] = [x_j] + \sum_{i=1}^{k-1} h_j^i\, f^{[i]}([x_j], [p], t_j) + h_j^k\, f^{[k]}([\tilde{x}_j], [p], [t_j, t_{j+1}]) \tag{6}$$

## ■ Set Integration with Interval Taylor Models (cont'd)

### □ Wrapping Effect




### □ Implementation

- a C++ package which implements Rhim's algorithm (mean value forms, matrices preconditioning and linear transforms)
- Profil/BIAS C++[17] class library for interval computations
- FADBAD++ package is used for computing Taylor coefficients

## ■ Set Integration with Müller's Theorem

### □ Müller's Theorem

- Hypothesis :

$$\begin{aligned} &\forall\, i,\quad D^{\pm}\,\omega_i(t) \le \min_{\underline{\mathsf{T}}_i(t)} f_i(\mathsf{x},\,\mathsf{p},\,t) \\ &\forall\, i,\quad D^{\pm}\,\Omega_i(t) \ge \max_{\overline{\mathsf{T}}_i(t)} f_i(\mathsf{x},\,\mathsf{p},\,t) \end{aligned} \tag{7}$$

$$\begin{aligned} &\underline{\mathsf{T}}_i : \left\{ x_i = \omega_i(t),\ \ \omega_j(t) \le x_j \le \Omega_j(t),\ \ j \ne i,\ \ \underline{\mathsf{p}} \le \mathsf{p} \le \overline{\mathsf{p}} \right\} \\ &\overline{\mathsf{T}}_i : \left\{ x_i = \Omega_i(t),\ \ \omega_j(t) \le x_j \le \Omega_j(t),\ \ j \ne i,\ \ \underline{\mathsf{p}} \le \mathsf{p} \le \overline{\mathsf{p}} \right\} \end{aligned} \tag{8}$$

- Solution of system is inside

$$\mathsf{X} : \begin{cases} t_0 \le t \le t_{n_T} \\ \omega(t) \le \mathsf{x}(t) \le \Omega(t) \end{cases} \tag{9}$$

### □ Using Monotonicity

- Study sign of the partial derivatives $\frac{\partial f_i}{\partial p_k}$ and $\frac{\partial f_i}{\partial x_j}$
- Solve

$$\begin{cases} \dot{\omega}(t) = \underline{f}\left(\omega,\,\Omega,\,\underline{\mathsf{p}},\,\overline{\mathsf{p}},\,t\right),\ \ \omega(t_0) = \underline{\mathsf{x}}_0 \\ \dot{\Omega}(t) = \overline{f}\left(\omega,\,\Omega,\,\underline{\mathsf{p}},\,\overline{\mathsf{p}},\,t\right),\ \ \Omega(t_0) = \overline{\mathsf{x}}_0 \end{cases} \tag{10}$$

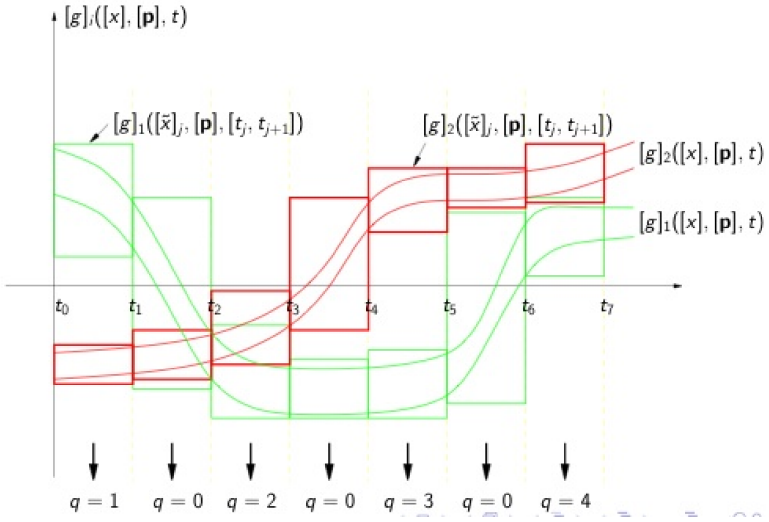
with

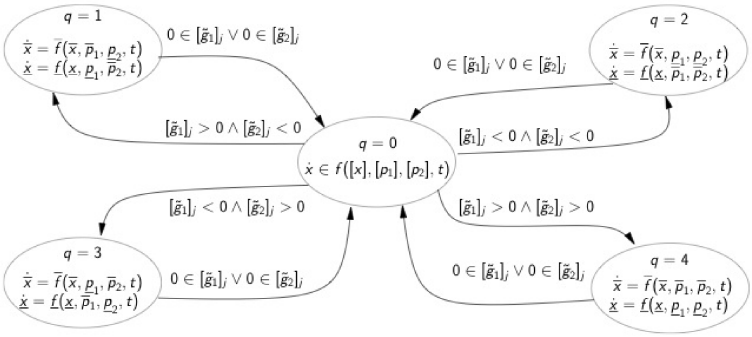
$$\begin{aligned} &\underline{f}_i\left(\omega,\,\Omega,\,\underline{\mathsf{p}},\,\overline{\mathsf{p}},\,t\right) = f_i\left(\underline{\gamma}^i(\mathsf{x}),\,\underline{\delta}^i(\mathsf{p}),\,t\right) \\ &\overline{f}_i\left(\omega,\,\Omega,\,\underline{\mathsf{p}},\,\overline{\mathsf{p}},\,t\right) = f_i\left(\overline{\gamma}^i(\mathsf{x}),\,\overline{\delta}^i(\mathsf{p}),\,t\right) \end{aligned} \tag{11}$$

$$\overline{\gamma}^i\left(x_j\right) = \begin{cases} \Omega_i \text{ if } i = j \\ \Omega_j \text{ if } (i \ne j) \wedge \dfrac{\partial f_i}{\partial x_j} \ge 0 \\ \omega_j \text{ if } (i \ne j) \wedge \dfrac{\partial f_i}{\partial x_j} < 0 \end{cases} \tag{12}$$

## ■ Hybridization

- Use Müller's theorem whenever it is possible, use interval taylor models otherwise




$$g_i(.)=\frac{\partial f}{\partial p_i}(.),\quad [\tilde{g}_i]_j=g_i([\tilde{x}_j],[p_1],[p_2],[t_j,t_{j+1}])$$

## ■ Practical Case

```
Uncert[v_, p_] := Interval[N[{v - p v/100, v + p v/100}]]
si[t] := s0 + p1 Cos[t/5]
equations = {(mu0 s/(s + ks + s^2/ki) - α d) x,
    -k mu0 s/(s + ks + s^2/ki) x + (si[t] - s) d}
variables = {x → Interval[{0.00000001, 1000}],
    s → Interval[{0.00000001, 1000}]};
parameters = {mu0 → Uncert[0.75, 1], s0 → Uncert[65, 1.5]};
constants = {k → 42.14, ks → 9.28, ki → 256, α → 0.5, d → 2., p1 → 15};
```

$$\left\{x\left(\frac{\text{mu0 s}}{\text{ks}+\text{s}+\frac{\text{s}^2}{\text{ki}}}-\text{d}\,\alpha\right),\ -\frac{\text{k mu0 s x}}{\text{ks}+\text{s}+\frac{\text{s}^2}{\text{ki}}}+\text{d}\left(-\text{s}+\text{s0}+\text{p1 Cos}\left[\frac{\text{t}}{5}\right]\right)\right\}$$

### □ Derivatives and signs

```
(jac = D[equations, {First /@ Join[variables, parameters]}]) //
  MatrixForm
```

$$\begin{pmatrix} \frac{\text{mu0 s}}{\text{ks}+\text{s}+\frac{\text{s}^2}{\text{ki}}}-\text{d}\,\alpha & \left(-\frac{\text{mu0 s}\left(1+\frac{2\,\text{s}}{\text{ki}}\right)}{\left(\text{ks}+\text{s}+\frac{\text{s}^2}{\text{ki}}\right)^2}+\frac{\text{mu0}}{\text{ks}+\text{s}+\frac{\text{s}^2}{\text{ki}}}\right)\text{x} & \frac{\text{s x}}{\text{ks}+\text{s}+\frac{\text{s}^2}{\text{ki}}} & 0 \\ -\frac{\text{k mu0 s}}{\text{ks}+\text{s}+\frac{\text{s}^2}{\text{ki}}} & -\text{d}+\frac{\text{k mu0 s}\left(1+\frac{2\,\text{s}}{\text{ki}}\right)\text{x}}{\left(\text{ks}+\text{s}+\frac{\text{s}^2}{\text{ki}}\right)^2}-\frac{\text{k mu0 x}}{\text{ks}+\text{s}+\frac{\text{s}^2}{\text{ki}}} & -\frac{\text{k s x}}{\text{ks}+\text{s}+\frac{\text{s}^2}{\text{ki}}} & \text{d} \end{pmatrix}$$

```
(sigjac = Fold[ReplacePart[#1, 1, {#2, #2}] &,
      Map[Sign, jac /. Join[{t → Interval[{0, 30}]}, constants] /.
        parameters /. variables, {2}],
      Range[Length[jac]]]) // MatrixForm
```

$$\begin{pmatrix} 1 & \text{Interval}[\{-1,\ 1\}] & 1 & 0 \\ -1 & 1 & -1 & 1 \end{pmatrix}$$

### □ Code

***Building equations for each mode***

Given an equation, the corresponding line of the sign matrix, and uncertain variables, **BuildEquation** returns the corresponding bounding equations. Minima and maxima of variable *x* are denoted by new variables xMin and xMax.

```
BuildEquation[eq_, l_, v_] :=
  eq /. Transpose[Map[{#[[1]] → #[[2, 1]], #[[1]] → #[[2, 2]]} &,
        Inner[{#2, SetMm[#1][#2]} &, l, v, List]]]

SetMm[1] = {min[#], max[#]} &;
SetMm[-1] = {max[#], min[#]} &;
SetMm[0] = {#, #} &;

min[s_Symbol] := ToExpression[ToString[s] <> "Min"]
max[s_Symbol] := ToExpression[ToString[s] <> "Max"]
```

For a given mode criterion, and the corresponding sign matrix, **BuildEquations** applies **BuildEquation** for generating all the equations of the mode.

```
BuildEquations[{mc_, sig_}, var_, param_, eqs_] :=
  {mc, MapIndexed[BuildEquation[#1, sig[[#2[[1]]]]],
        First /@ Join[var, param]] &, eqs]}
```

### *Building modes*

**BuildMode** is the low level function computing all the modes and the corresponding equations.

```
BuildMode[jac_, p_, sig_, var_, param_, eqs_] :=
  Map[BuildEquations[#, var, param, eqs] &,
    Transpose[{With[{l = Map[{# > 0, # < 0} &, Extract[jac, p]]},
        Flatten[Outer[List, Apply[Sequence, l]], Length[l] - 1]],
      Map[BuildSignJac[#, sig] &,
        With[{l = Map[{toto[1, #], toto[-1, #]} &, p]}, Flatten[
          Outer[List, Apply[Sequence, l]], Length[l] - 1]]]}]]

BuildSignJac[{x__toto}, sig_] :=
  Fold[ReplacePart[#1, #2[[1]], #2[[2]]] &, sig, {x}]
```

**GenMode** is the high level function computing all the modes and the corresponding equations.

```
GenMode[var_, param_, eq_, val_] :=
  With[{jac = D[eq, {First /@ Join[var, param]}]},
    With[{sigjac = Fold[ReplacePart[#1, 1, {#2, #2}] &, Map[Sign,
        jac /. val /. param /. var, {2}], Range[Length[jac]]]},
      With[{pos = Position[sigjac, _Interval]},
        BuildMode[jac, pos, sigjac, var, param, eq]]]]
```

```
GenJacMode[var_, param_, eq_, val_] := Map[FullSimplify,
    With[{jac = D[eq, {First /@ Join[var, param]}]},
      With[{sigjac = Fold[ReplacePart[#1, 1, {#2, #2}] &,
            Map[Sign, jac /. val /. param /. var, {2}],
            Range[Length[jac]]]}, With[
        {pos = Position[sigjac, _Interval]}, Extract[jac, pos]]]]]
```

Result is a list of mode. Each mode is a {<criterion>, <equations>} list. <criterion> is a list of inequations. <equations> is a list of couples of {min, max} equations.

## □ Mode Computation

```
res = GenMode[variables, parameters,
    equations, Join[{t → Interval[{0, 30}]}], constants]]
```

$$\left\{\left\{\left\{-\frac{\text{mu0 s}\left(1+\frac{2\,s}{ki}\right)}{\left(ks+s+\frac{s^2}{ki}\right)^2}+\frac{mu0}{ks+s+\frac{s^2}{ki}}\right\}\;x>0\right\},\right.$$

$$\left\{\left\{xMin\left[\frac{mu0Min\;sMin}{ks+sMin+\frac{sMin^2}{ki}}-d\,\alpha\right],\;xMax\left[\frac{mu0Max\;sMax}{ks+sMax+\frac{sMax^2}{ki}}-d\,\alpha\right]\right\},\right.$$

$$\left\{-\frac{k\;mu0Max\;sMin\;xMax}{ks+sMin+\frac{sMin^2}{ki}}+d\left(s0Min-sMin+p1\,Cos\left[\frac{t}{5}\right]\right),\right.$$

$$\left.\left.\left.-\frac{k\;mu0Min\;sMax\;xMin}{ks+sMax+\frac{sMax^2}{ki}}+d\left(s0Max-sMax+p1\,Cos\left[\frac{t}{5}\right]\right)\right\}\right\}\right\},$$

$$\left\{\left\{-\frac{\text{mu0 s}\left(1+\frac{2\,s}{ki}\right)}{\left(ks+s+\frac{s^2}{ki}\right)^2}+\frac{mu0}{ks+s+\frac{s^2}{ki}}\right\}\;x<0\right\},$$

$$\left\{\left\{xMin\left[\frac{mu0Min\;sMax}{ks+sMax+\frac{sMax^2}{ki}}-d\,\alpha\right],\;xMax\left[\frac{mu0Max\;sMin}{ks+sMin+\frac{sMin^2}{ki}}-d\,\alpha\right]\right\},\right.$$

$$\left\{-\frac{k\;mu0Max\;sMin\;xMax}{ks+sMin+\frac{sMin^2}{ki}}+d\left(s0Min-sMin+p1\,Cos\left[\frac{t}{5}\right]\right),\right.$$

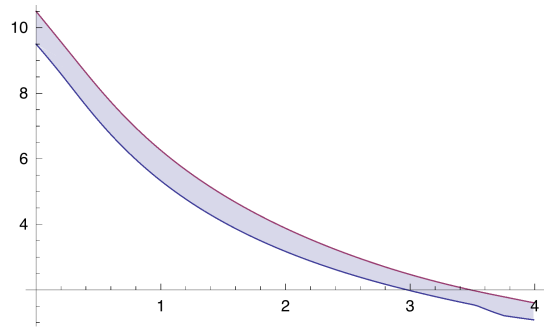
$$\left.\left.\left.-\frac{k\;mu0Min\;sMax\;xMin}{ks+sMax+\frac{sMax^2}{ki}}+d\left(s0Max-sMax+p1\,Cos\left[\frac{t}{5}\right]\right)\right\}\right\}\right\}$$

## □ C code linking

### *Code generation*

## □ Results

**PlotFrontier[l1, 1]**




**PlotFrontier[l1, 2]**