ADIODES, a Self-Validating ODE Solver

Ole Stauning <ostauning@amexmail.com>

now at

Price Waterhouse
Copenhagen, Denmark

previously at

Department of Mathematical Modelling
Technical University of Denmark

SIAM Annual Meeting,
University of Toronto,
July 15, 1998

IMM

A D I O D E S

Automatic Differentiation Interval Ordinary Differential Equation Solver

The methods used in ADIODES

- Picard Iterations
- The (Extended) Mean Value Method
- Mixed Taylor/Forward Automatic Differentiation

The building blocks of ADIODES

- The Interval Package BIAS/PROFIL
- FADBAD/TADIFF
  - FAD : Forward Automatic Differentiation
  - BAD : Backward Automatic Differentiation
  - TAD : Taylor Automatic Differentiation

The extended mean value enclosure

Consider discrete maps of the type:

$$\varphi(y) = \tilde{\varphi}(y) + \varepsilon(y)$$

where:

$\tilde{\varphi}$ is differentiable function,

ε is a (small) error which can be bounded.

From $y_0 \in \mathbb{R}^n$ we have a sequence $\{y_j\}_{j=0,\dots}$ defined by

$$y_{j+1} = \varphi(y_j) = \tilde{\varphi}(y_j) + \varepsilon(y_j).$$

Using the **extended mean value method**, we obtain two enclosures

Internal representation "rotating rectangle":

$$y_{j+1} \in \widehat{y}_{j+1} + A_{j+1}[\widehat{r}_{j+1}]$$

External representation "interval vector":

$$y_{j+1} \in [y_{j+1}]$$

where $\widehat{y}_{j+1}$ is a real vector, $[y_{j+1}]$ and $[\widehat{r}_{j+1}]$ are interval vectors and A_{j+1} a real orthogonal matrix.

The extended mean value enclosure

The algorithm

Initialize:

$[y_0]$, $\widehat{y}_0 = \mathrm{m}([y_0])$, $[\widehat{r}_0] = [y_0] - \widehat{y}_0$, $A_0 = I$.

Input:

$[y_j]$, $\widehat{y}_j$, $[\widehat{r}_j]$, A_j.

Iteration:

$[z_{j+1}] = \Sigma([y_j])$,

$[\widehat{y}_{j+1}] = \tilde{\varphi}(\widehat{y}_j) + [z_{j+1}]$,

$[S_j] = \tilde{\Phi}'([y_j])$,

$\widehat{y}_{j+1} = \mathrm{m}([\widehat{y}_{j+1}])$,

Choose a regular real matrix A_{j+1},

$[y_{j+1}] = ([S_j]A_j)[\widehat{r}_j] + [\widehat{y}_{j+1}]$,

$[\widehat{r}_{j+1}] = (A_{j+1}^{-1}([S_j]A_j))[\widehat{r}_j] + A_{j+1}^{-1}([\widehat{y}_{j+1}] - \widehat{y}_{j+1})$.

Output:

$[y_{j+1}]$, $\widehat{y}_{j+1}$, $[\widehat{r}_{j+1}]$.

How to compute $\tilde{\varphi}(\hat{y}_j)$ and $\tilde{\Phi}'([y_j])$

Consider the Cos-Sin map:

$$\varphi_{cs}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos(x+ay) \\ \sin(bx+y) \end{pmatrix},$$

The C++ code:

```
INTERVAL X,Y,PX,PY;
X=0.5; Y=0.5;

PX=Cos(X+4*Y);
PY=Sin(4*X+Y);
```

$$\varphi_{cs}\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \in \begin{pmatrix} \mathtt{PX} \\ \mathtt{PY} \end{pmatrix}$$

Evaluates the map in interval arithmetic. And

```
FINTERVAL X,Y,PX,PY;
X=0.5; Y=0.5;
X.diff(0,2);Y.diff(1,2);

PX=Cos(X+4*Y);
PY=Sin(4*X+Y);
```

$$\varphi_{cs}\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \in \begin{pmatrix} \mathtt{PX.x()} \\ \mathtt{PY.x()} \end{pmatrix}, \Phi'_{cs}\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \in \begin{pmatrix} \mathtt{PX.d(0)} & \mathtt{PX.d(1)} \\ \mathtt{PY.d(0)} & \mathtt{PY.d(1)} \end{pmatrix}$$

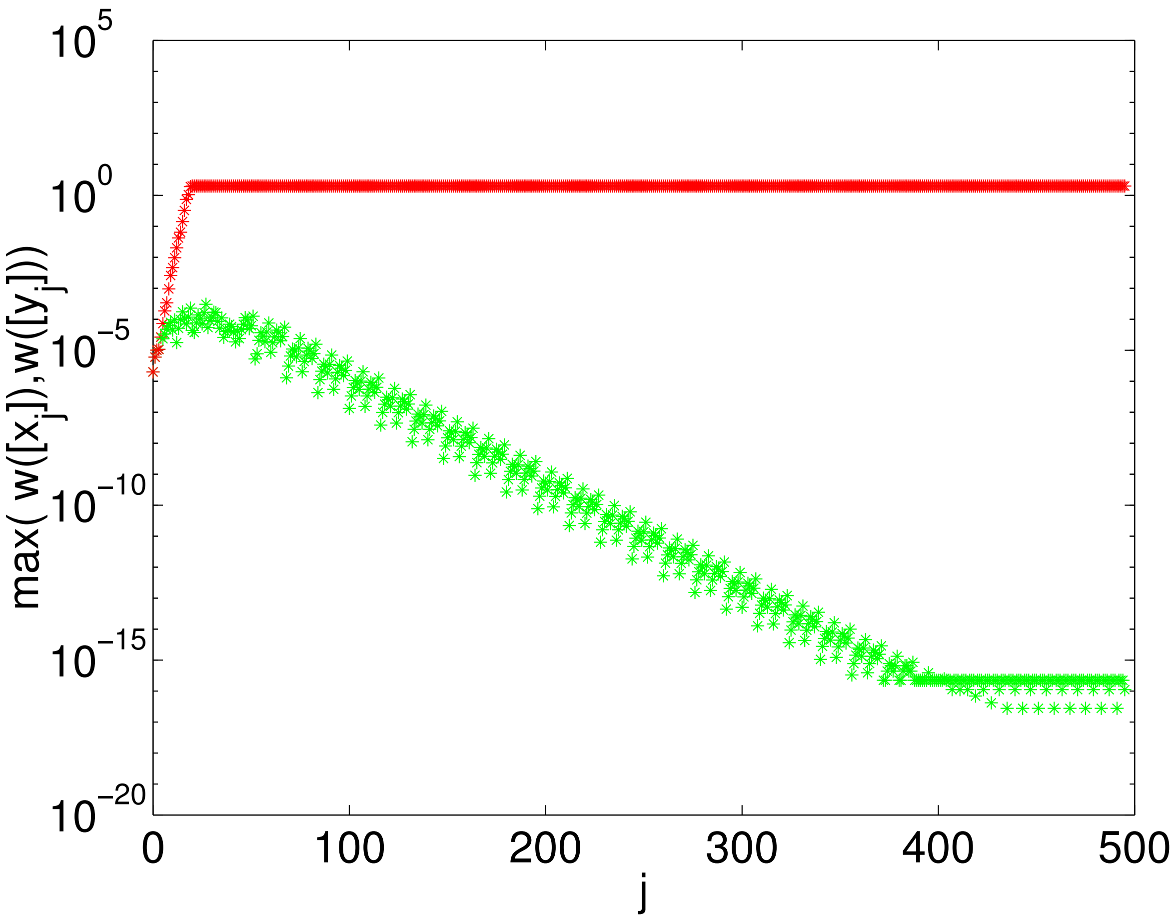
Evaluates the map and derivatives in interval arithmetic.

Using the extended mean value enclosure

Using parameters $a = b = 2$, and initial values

$$([x_0],[y_0]) = ([-10^{-6},10^{-6}],[-10^{-6},10^{-6}])$$

we obtain



The graph shows the width of

'*':the simple interval enclosure,
'*':the extended mean value enclosure.

Discretizing solutions of Ordinary Differential Equations (ODE's)

Consider the ordinary differential equation

$$y' = f(y)$$

with a solution $y : [t_0, t_N] \to \mathbb{R}^n$.

Using $t_0 < t_1 < \cdots < t_N$, and $y_j = y(t_j)$ we have

$$y_{j+1} = \varphi(y_j) = \tilde{\varphi}(y_j) + \varepsilon(y_j)$$

where $\tilde{\varphi}$ is a Taylor polynomial of order k and ε the remainder:

$$\tilde{\varphi}(y_j) = y_j + \sum_{i=1}^{k} (y_j)_i (t_{j+1} - t_j)^i,$$

$$\varepsilon(y_j) = (t_{j+1} - t_j)^{k+1}(k+1) \int_0^1 y^{[k+1]}(\theta t_{j+1} + (1-\theta)t_j)(1-\theta)^k d\theta.$$

Where

$$y^{[k]} = \frac{1}{k!}\frac{d^k y}{dt^k} \quad \text{and} \quad (y_j)_k = y^{[k]}(t_j).$$

Obtaining Taylor coefficients with Automatic Differentiation

From the ODE $y' = f(y)$ we have the recursive relation:

$$(y_j)_{k+1} = \frac{(f)_k}{k+1}$$

By using Taylor arithmetic to compute $(f)_k$ we can obtain $(y_j)_{k+1}$ and so forth.

Taylor arithmetic:

$$\begin{aligned}
(u+v)_k &= (u)_k + (v)_k, \\
(u-v)_k &= (u)_k - (v)_k, \\
(u \cdot v)_k &= \sum_{i=0}^{k} (u)_i (v)_{k-i} = \sum_{i=0}^{k} (u)_{k-i} (v)_i, \\
(u/v)_k &= \frac{1}{(v)_0} \left((u)_k - \sum_{j=1}^{k} (v)_j (u/v)_{k-j} \right) \text{ for } (v)_0 \neq 0, \\
(\cos u)_k &= -\frac{1}{k} \sum_{j=0}^{k-1} (k-j)(\sin u)_j (u)_{k-j} \text{ for } k \geq 1, \\
(\sin u)_k &= \frac{1}{k} \sum_{j=0}^{k-1} (k-j)(\cos u)_j (u)_{k-j} \text{ for } k \geq 1.
\end{aligned}$$

etc...

Obtaining Taylor coefficients using Automatic Differentiation

Example

$$
\begin{aligned}
x' &= f_1(x,y) = A - x(B - xy + 1), \\
y' &= f_2(x,y) = x(B - xy),
\end{aligned}
$$

Introducing functions $\tau_j(x,y)$ defined by

$$
\begin{aligned}
\tau_1 &= xy, \\
\tau_2 &= B - \tau_1, \\
\tau_3 &= \tau_2 + 1, \\
\tau_4 &= x\tau_3.
\end{aligned}
$$

we have

$$
f_1 = A - \tau_4, \quad f_2 = x\tau_2
$$

From the initial value (x_j, y_j, t_j) we can compute

$$
\begin{aligned}
(x)_0 &= x_j, & (\tau_1)_1 &= (x)_0(y)_1 + (x)_1(y)_0, \\
(y)_0 &= y_j, & (\tau_2)_1 &= -(\tau_1)_1, \\
(\tau_1)_0 &= (x)_0(y)_0, & (\tau_3)_1 &= (\tau_2)_1, \\
(\tau_2)_0 &= B - (\tau_1)_0, & (\tau_4)_1 &= (x)_0(\tau_3)_1 + (x)_1(\tau_3)_0, \\
(\tau_3)_0 &= (\tau_2)_0 + 1, & (f_1)_1 &= -(\tau_4)_1, \\
(\tau_4)_0 &= (x)_0(\tau_3)_0, & (f_2)_1 &= (x)_0(\tau_2)_1 + (x)_1(\tau_2)_0, \\
(f_1)_0 &= A - (\tau_4)_0, & (x)_2 &= (f_1)_1/2, \\
(f_2)_0 &= (x)_0(\tau_2)_0, & (y)_2 &= (f_2)_1/2, \\
(x)_1 &= (f_1)_0, & & \cdots\cdots\cdots \\
(y)_1 &= (f_2)_0, & &
\end{aligned}
$$

How to compute $\tilde{\varphi}(\hat{y}_j)$ using TADIFF

Taylor expanding the solution of

$$\begin{aligned} x' &= f_1(x,y) = A - x(B - xy + 1), \\ y' &= f_2(x,y) = x(B - xy), \end{aligned}$$

The C++ code:

```
INTERVAL X0(0.3),Y0(0.4),PX,PY;
TINTERVAL X(X0),Y(Y0),F1,F2;

  F1 = A - X*(B - X*Y + 1);
  F2 = X*(B - X*Y);

  PX=X[0];
  PY=Y[0];
  for(i=0; i<N; i++){
    F1.eval(i);
    F2.eval(i);

    X[i+1] = F1[i]*h/(i+1);
    Y[i+1] = F2[i]*h/(i+1);

    PX+=X[i+1];
    PY+=Y[i+1];
  }
```

$$\tilde{\varphi}\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \in \begin{pmatrix} \mathtt{PX} \\ \mathtt{PY} \end{pmatrix}$$

Evaluates the map in interval arithmetic. And ...

How to compute $\tilde{\varphi}(\hat{y}_j)$ and $\tilde{\Phi}'([y_j])$ using FADBAD and TADIFF

The C++ code:

```
FINTERVAL X0(0.3),Y0(0.4),PX,PY;
X0.diff(0,2); Y0.diff(1,2);
TFINTERVAL X(X0),Y(Y0),F1,F2;

  F1 = A - X*(B - X*Y + 1);
  F2 = X*(B - X*Y);

  PX=X[0];
  PY=Y[0];
  for(i=0; i<N; i++){
    F1.eval(i);
    F2.eval(i);

    X[i+1] = F1[i]*h/(i+1);
    Y[i+1] = F2[i]*h/(i+1);

    PX+=X[i+1];
    PY+=Y[i+1];
  }
```

$$\tilde{\varphi}\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \in \begin{pmatrix} \texttt{PX.x()} \\ \texttt{PY.x()} \end{pmatrix}, \tilde{\Phi}'\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \in \begin{pmatrix} \texttt{PX.d(0)} & \texttt{PX.d(1)} \\ \texttt{PY.d(0)} & \texttt{PY.d(1)} \end{pmatrix}$$

Evaluates the map and derivatives in interval arithmetic.

The Lorenz equations

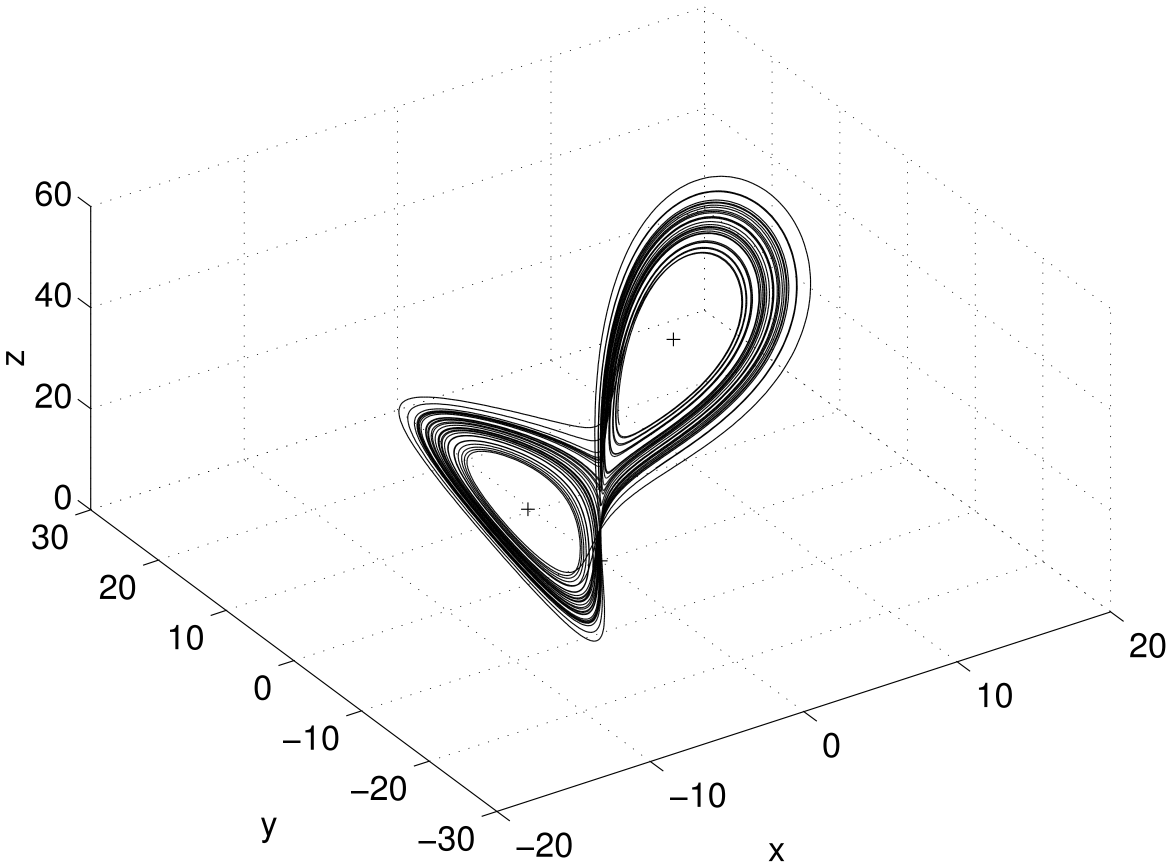
The Lorenz equations are given by

$$\begin{aligned} x' &= \sigma(y-x), \\ y' &= rx - y - xz, \\ z' &= xy - bz. \end{aligned}$$

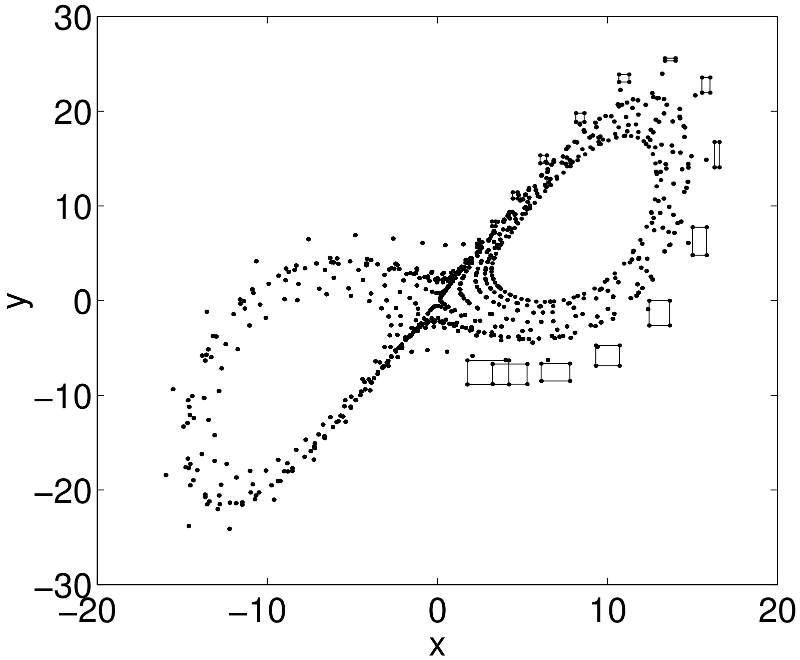
Using the parameters $b = 8/3, r = 28$ and $\sigma = 6$ and the initial values:

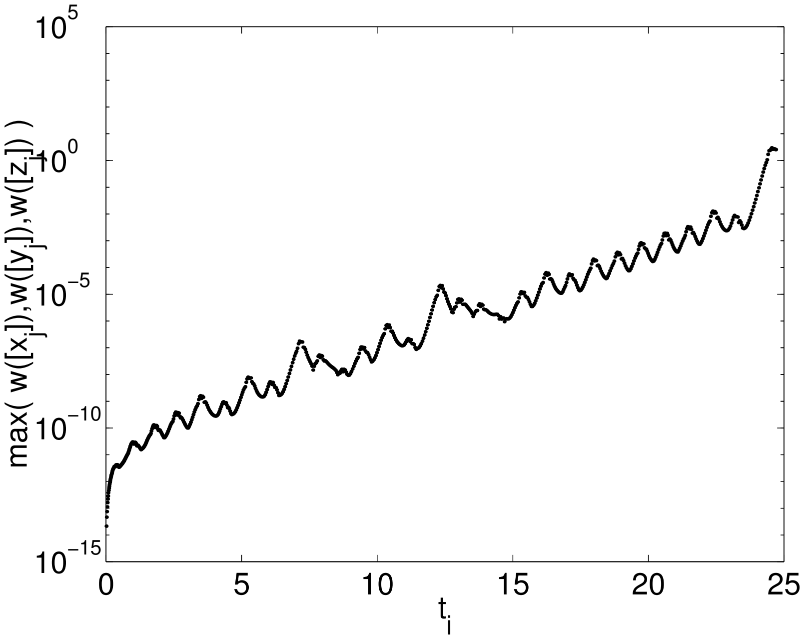
$$x(0) = 4.1879, \;\; y(0) = 6.7601 \text{ and } z(0) = 16.1091$$

Matlab finds the numerical solution (shown i phase space):



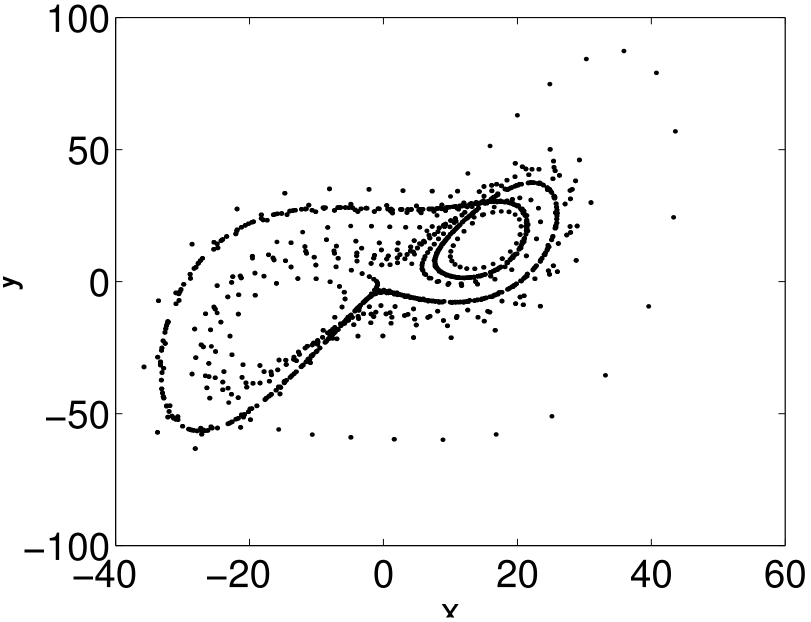
Using ADIODES to solve the Lorenz equations we obtain:

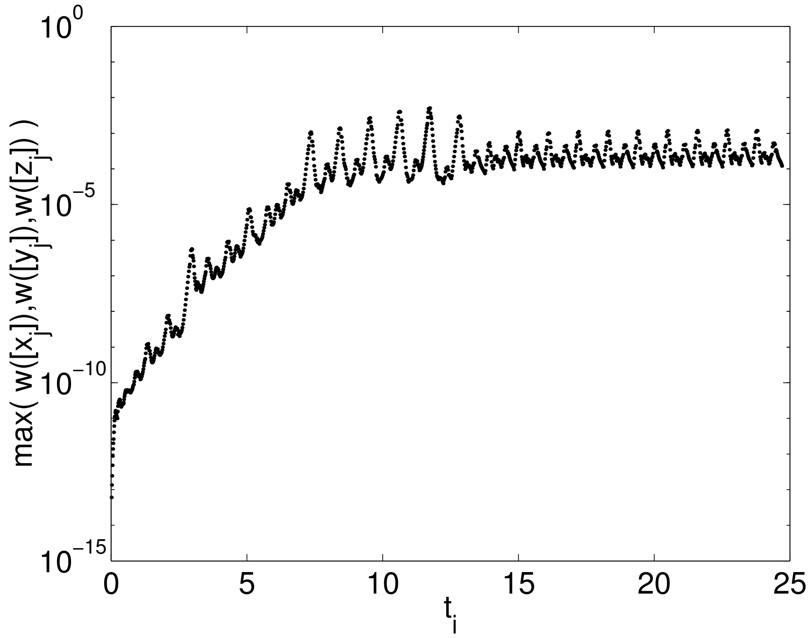




for parameters $b = 8/3, r = 28$ and $\sigma = 6$.

Using ADIODES to solve the Lorenz equations we obtain:





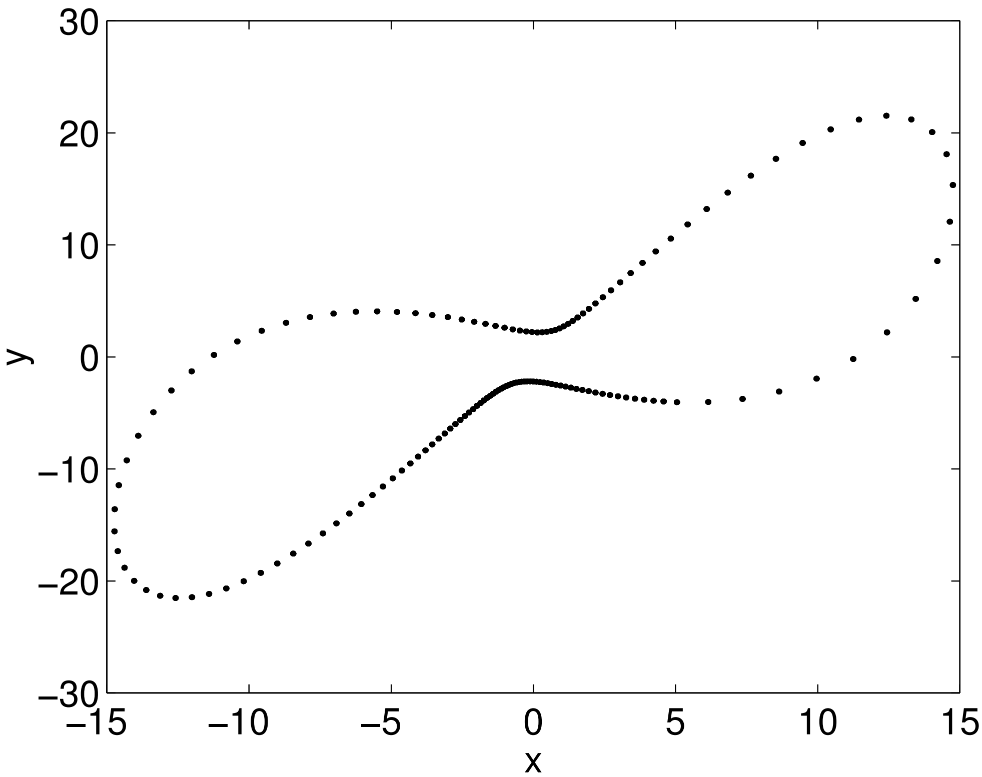
for parameters $b = 8/3, r = 100.5$ and $\sigma = 10$.

Periodic solutions of the Lorenz equations

The following periodic solution were proved to exist for parameters

$$b = 8/3, r = 28 \text{ and } \sigma = 6,$$

by using ADIODES and the Interval Newton method:



Solution 1:1, proved to exist for initial values:

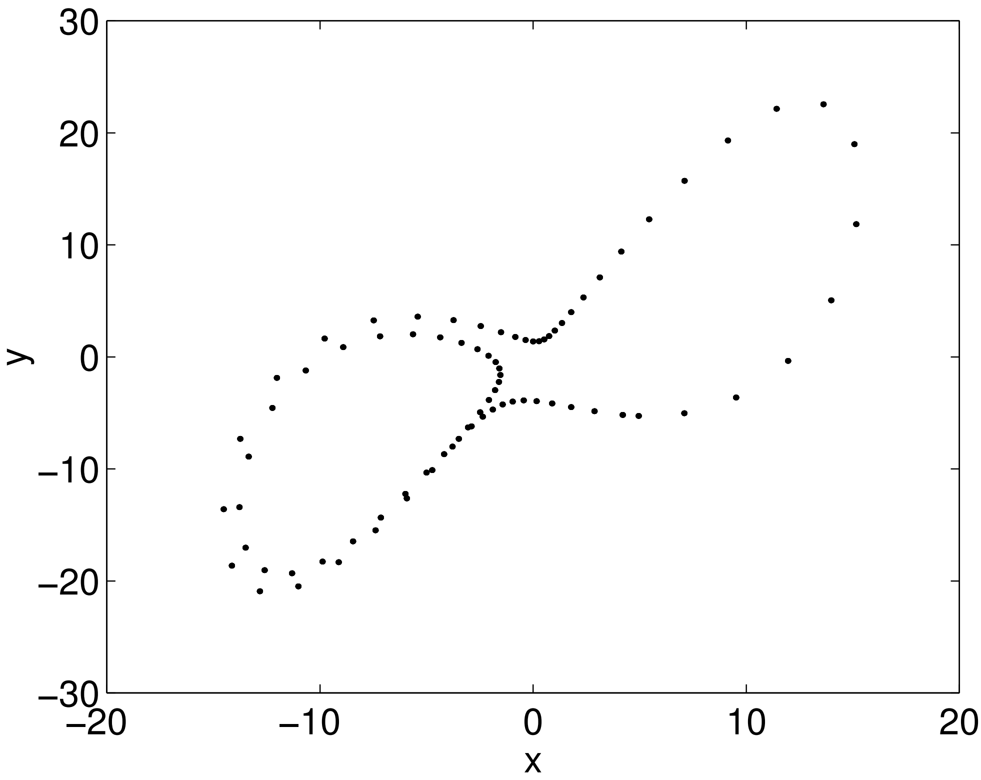
$$\begin{aligned} T &\in [2.594277^{7}_{6}], \\ x(0) &\in [4.194260^{4}_{3}], \\ y(0) &\in [-5.173485^{7}_{8}], \\ z(0) &= 27. \end{aligned}$$

Periodic solutions of the Lorenz equations

The following periodic solution were proved to exist for parameters

$$b = 8/3, r = 28 \text{ and } \sigma = 6,$$

by using ADIODES and the Interval Newton method:



Solution 2:1, proved to exist for initial values:

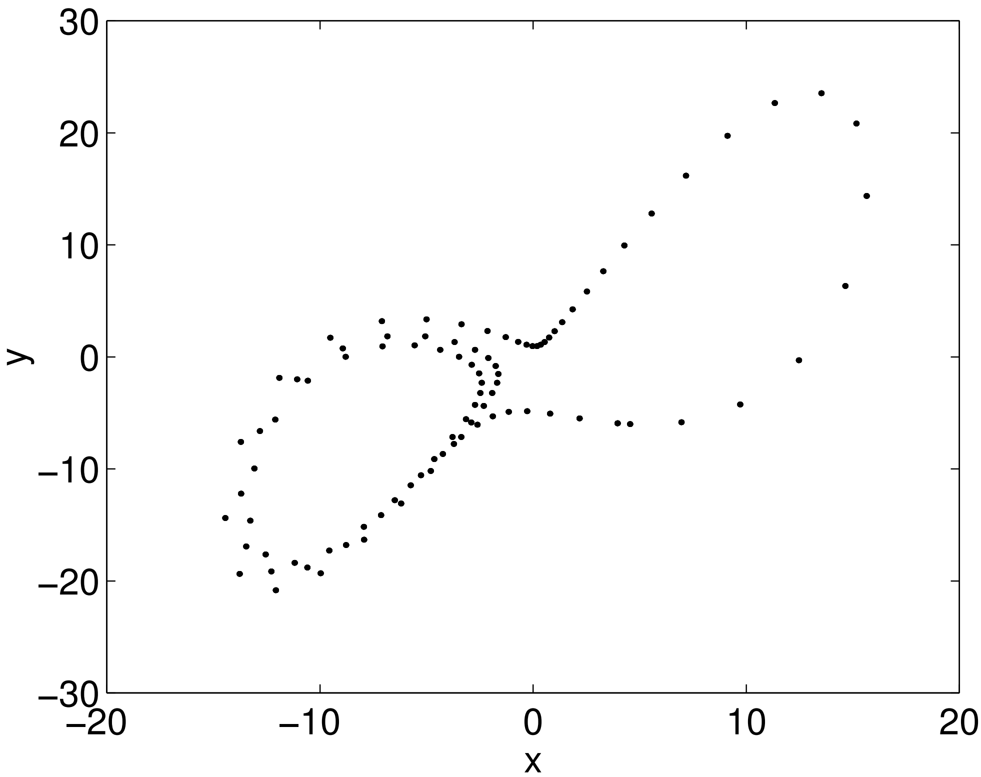
$$\begin{aligned} T &\in [2.594277^{7}_{6}], \\ x(0) &\in [4.194260^{4}_{3}], \\ y(0) &\in [-5.173485^{7}_{8}], \\ z(0) &= 27. \end{aligned}$$

Periodic solutions of the Lorenz equations

The following periodic solution were proved to exist for parameters

$$b = 8/3, r = 28 \text{ and } \sigma = 6,$$

by using ADIODES and the Interval Newton method:



Solution 3:1, proved to exist for initial values:

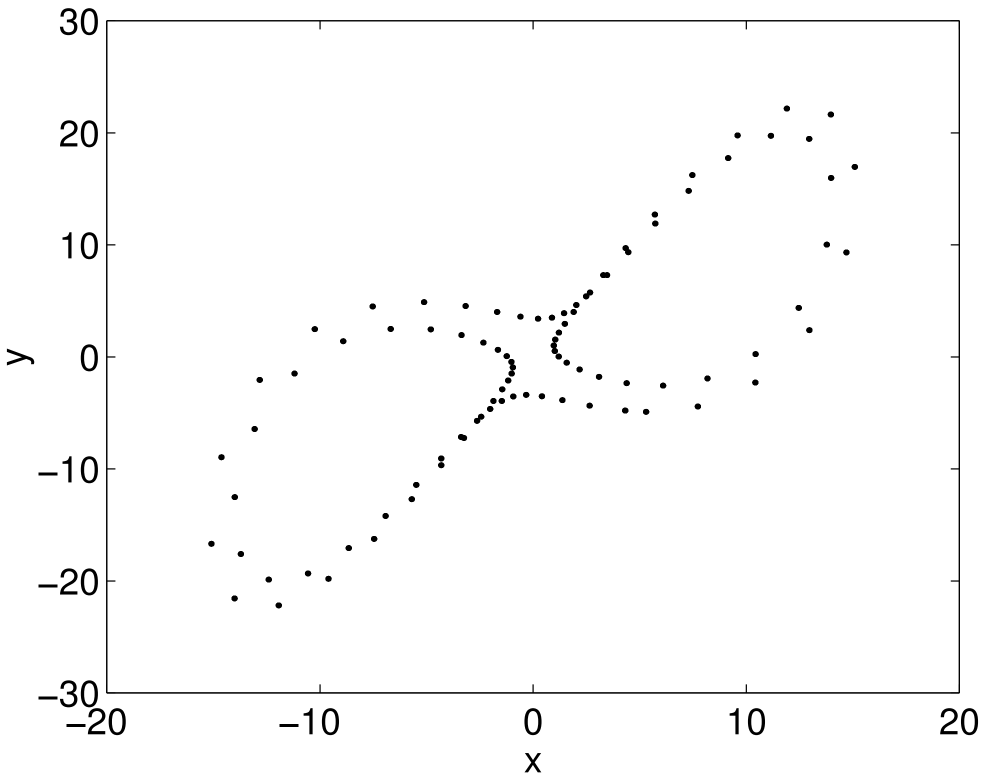
$$\begin{aligned} T &\in [3.405937^{8}_{7}], \\ x(0) &\in [3.952332^{3}_{2}], \\ y(0) &\in [-5.928351^{4}_{5}], \\ z(0) &= 27. \end{aligned}$$

Periodic solutions of the Lorenz equations

The following periodic solution were proved to exist for parameters

$$b = 8/3, r = 28 \text{ and } \sigma = 6,$$

by using ADIODES and the Interval Newton method:



Solution 2:2, proved to exist for initial values:

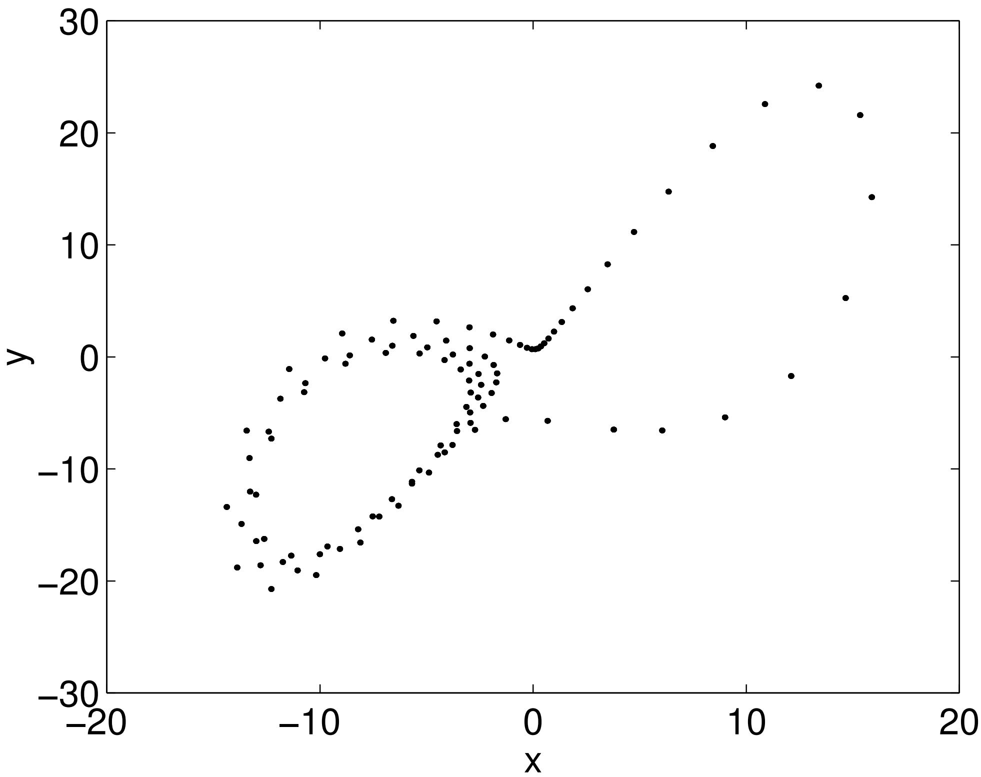
$$\begin{aligned} T &\in [3.469322_0^2], \\ x(0) &\in [4.312692_0^6], \\ y(0) &\in [-4.80296_{83}^{77}], \\ z(0) &= 27. \end{aligned}$$

Periodic solutions of the Lorenz equations

The following periodic solution were proved to exist for parameters

$$b = 8/3, r = 28 \text{ and } \sigma = 6,$$

by using ADIODES and the Interval Newton method:



Solution 4:1, proved to exist for initial values:

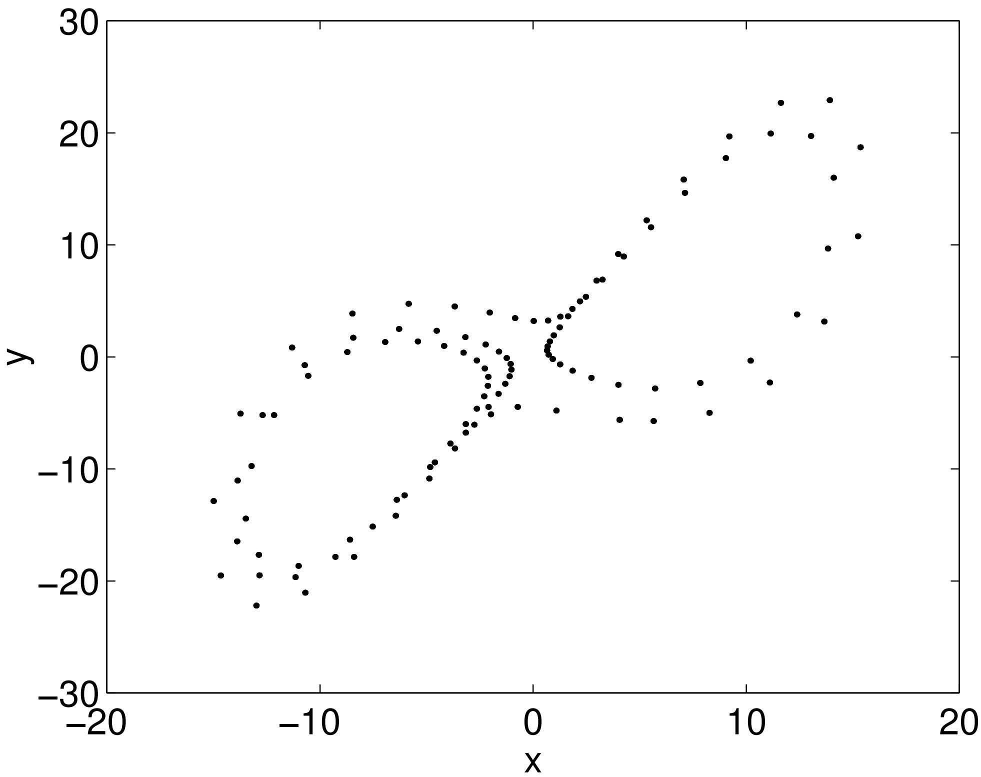
$$\begin{aligned} T &\in [4.2012773^{90}_{89}], \\ x(0) &\in [3.7720541^{62}_{59}], \\ y(0) &\in [-6.48667584^{3}_{7}], \\ z(0) &= 27. \end{aligned}$$

Periodic solutions of the Lorenz equations

The following periodic solution were proved to exist for parameters

$$b = 8/3, r = 28 \text{ and } \sigma = 6,$$

by using ADIODES and the Interval Newton method:



Solution 3:2, proved to exist for initial values:

$$\begin{aligned} T &\in [4.300104^{60}_{59}], \\ x(0) &\in [4.0501795^{8}_{6}], \\ y(0) &\in [-5.6242059^{0}_{3}], \\ z(0) &= 27. \end{aligned}$$

Conclusion

Download FADBAD/TADIFF and ADIODES from:

<http://www.imm.dtu.dk/fadbad.html>

POSITIVE:

- The Extended Mean Value Form also works on discrete mappings.
- ADIODES uses automatic differentiation so that the user only have to provide C++ code implementing the right hand side of the ODE.
- Using ADIODES we can prove properties of ODE's which are hard to prove by hand.

NEGATIVE:

- The Picard itarations takes small stepsizes.
- We need new (implicit) methods for stiff problems.
- Overloading in FADBAD/TADIFF is expensive because of memory management.
- FADBAD/TADIFF and ADIODES are not ported to Visual C++.