Implementing Rigorous Defect Control

John M. Ernsthausen

Joint work with Ned Nedialkov

McMaster University
Canada

8th Taylor Model Workshop
May 9-11, 2018

Problem statement

Given the initial-value problem (IVP)

$$x'(t) = f\big(t, (t)\big) \in \mathbb{R}^d \quad x(t_0) = x_0 \quad [t_0, t_{\text{end}}] \quad \text{tol} > 0$$

compute a piecewise polynomial approximate solution u

- continuously differentiable $t \in [t_0, t_{\text{end}}] \mapsto u(t) = \big(u_1(t), \ldots, u_d(t)\big) \in \mathbb{R}^d$
- nearly satisfying the initial condition

Compute defect

$$\Delta u(t) = u'(t) - f\big(t, u(t)\big) \quad \Delta u(t_0) \stackrel{\text{def}}{=} u(t_0) - x_0$$

Rigorous bound

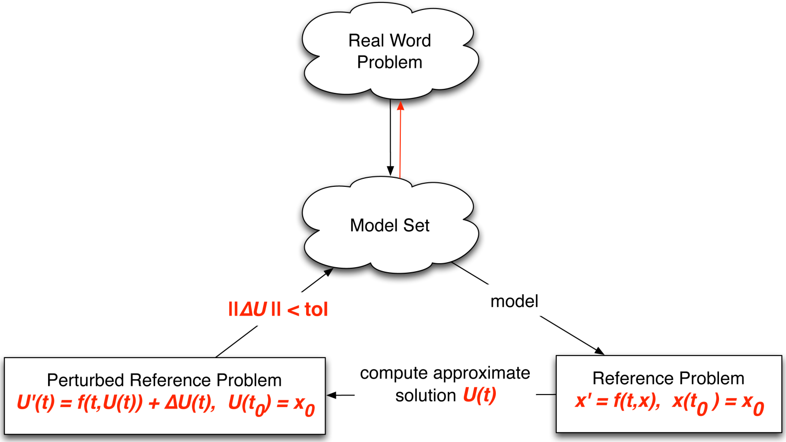
$$\|\Delta u\|_\infty \stackrel{\text{def}}{=} \max_{i,t} |\Delta u_i(t)| \leq \text{tol} \quad t \in [t_0, t_{\text{end}}] \quad i = 1, \ldots, d$$

- Rigorous Polynomial Approximation (RPA)
- Taylor models (TM)
- Interval Arithmetic (IA)

Defect control literature

- Enright advocates asymptotic defect control
  Enright and Coworkers and Students (since 1989)
- Defect control and ODE boundary value problem
  Enright and Muir, Shampine and Muir (1993-2004)
- Corless and Corliss outlined rigorous defect control
  Corless and Corliss (1991)

Residual-based backward error analysis for ODE



Local residual-based backward error analysis for ODE

Given

$$x'(t) = f\big(t, x(t)\big) \quad x(t_n) = x_n \quad \text{tol} > 0$$

compute approximate u on $[t_n, t_{n+1}]$ and compute defect

$$\Delta u(t) \stackrel{\text{def}}{=} u'(t) - f\big(t, u(t)\big) \quad \Delta u(t_n) \stackrel{\text{def}}{=} u(t_n) - x_n$$

Find stepsize so that u satisfies on $[t_n, t_{n+1}]$

$$u'(t) = f\big(t, u(t)\big) + \Delta u(t) \quad u(t_n) = x_n \quad \|\Delta u\|_\infty \le \text{tol}$$

Then u exactly solves modified problem on $[t_0, t_{\text{end}}]$

$$u'(t) = f\big(t, u(t)\big) + \Delta u(t) \quad u(t_0) = x_0 + \Delta u(t_0) \quad \|\Delta u\|_\infty \le \text{tol}$$

Outline

- Why defect control
- Approximate solution
- Our method
- Interval arithmetic evaluation
- ODETS software
- Results
- Conclusions

Backward error vs Forward error

Forward error

- Standard ODE-IVP solvers control local error on each step
  - Local error control can be deceived
  - No guarantee the global error is within some bounds
- Interval methods compute rigorous bounds on solution
  - hard to keep them tight

Backward error

- Compute exact solution to a modified problem
  Approximate solution solves exactly $u'(t) = f\big(t, u(t)\big) + \Delta u(t)$
  The model is usually an approximation anyhow
- Monitor and control the maximum magnitude of the defect
  Asymptotically correct defect estimate (Enright)
  Guarantee $\|\Delta u\|_\infty \leq \text{tol}$

Why now is the right time for defect control

Approximate solution is true solution of modified problem

Defect encaptulates all errors

Bounding real valued function

Well-studied problem in interval analysis

Rigorous Polynomial Approximation (Joldes 2011)

TM arithmetic in one independent variable

Rigorous supremum norm of a polynomial

Approximate solution

Good numerical ODE solvers for the initial value problem

$$x'(t) = f\big(t, x(t)\big) \quad x(t_0) = x_0$$

- control local error on each step
- return skeletal solution $\big(t_n, x_n\big)$
- return a continuously differentiable approximation u to x

Defect control (DC) methods

- monitor and control the maximum magnitude of the defect
- Asymptotic DC estimates $\|\Delta u\|_\infty$ by evaluating it at carefully selected points in each integration interval
- Rigorous DC ensures $\|\Delta u(t)\| \le$ tol on $[t_0, t_{\text{end}}]$

Taylor series method

Computation often regarded as expensive

This is not the case

Computing defect inexpensive

Compared to cost of Taylor series method itself

$$u(t) = \sum_{k=0}^{K} (u)_k (t - t_i)^k \quad \text{where} \quad (u)_k = \frac{1}{k} (f)_{k-1}$$

Data management: ApproximateSolution class

Automatic differentiation via operator overloading

From f to its Computational Graph, a DAG

Bendtsen and Stauning [FADBAD++, TADIFF] (1997)

Idea: Taylor arithmetic

- Assume user equations are elementary functions
- Construct an efficient computational graph
- Nodes (basic functions): sin, asin, sqrt, pow, log, exp
- Edges (basic operators): add, sub, mul, div, composition

Interface to TADIFF: TaylorExpansion class

Method to integrate ODE by time stepping

Given initial condition x_n at t_n and stepsize h_n, take a step to $t_{n+1} = t_n + h_n$

Phase I. Compute an approximate polynomial solution
floating-point arithmetic

Phase II. Bound the defect
Taylor models and interval arithmetic

Phase III. Accept/reject step
floating-point arithmetic

Phase I: Compute an approximate polynomial solution

(a) We use Taylor series:

$$u(t) = x_n + (x_n)_1(t - t_n) + \cdots + (x_n)_K(t - t_n)^K,$$

- $(x_n)_k$ are Taylor coefficients at t_n
- Computed using automatic differentiation and FADBAD++ Bendtsen and Stauning

(b) Evaluate $x_{n+1} = u(t_{n+1})$ and interpolate $f(t_{n+1}, x_{n+1})$:

$$U(t) = u(t) + \frac{\Delta u(t_{n+1})}{h_n^K}(t - t_n)^{K+1} - \frac{\Delta u(t_{n+1})}{h_n^{K+1}}(t - t_n)^{K+2}$$

This ensures $\Delta U(t_n) = \Delta U(t_{n+1}) = 0$

Phase II: Bound the defect

We use the SOLLYA package: RPA and sup-norm computation
Chevillard, Joldes, Lauter

(a) Evaluate the code list of $x' - f(t, x)$ with $\big(U, [0, 0]\big)$ in TM arithmetic

- ith component of the result is $\big(p_i, \boldsymbol{r}_i\big)$:

$$\Delta U_i(t) - p_i(t) \in \boldsymbol{r}_i \qquad \text{for all } t \in [t_n, t_{n+1}]$$

(b) Compute a rigorous enclosure $\boldsymbol{b}_i = [\underline{b}_i, \bar{b}_i]$:

$$\underline{b}_i \leq \sup_{t \in [t_n, t_{n+1}]} |p_i(t)| \leq \bar{b}_i$$

Then, on $[t_n, t_{n+1}]$,

$$\|\Delta U_i\|_\infty \leq \delta_i := \bar{b}_i + |\boldsymbol{r}_i|, \quad |\boldsymbol{r}_i| = \max\{|\underline{r_i}|, |\overline{r_i}|\}$$

We ensure $\delta_i \leq \text{tol}$ for all $i = 1, \ldots, d$

Example

Consider

$$x'(t) = f(t, x(t)) = x(t) - x(t)^2 \qquad x(0) = 0.2$$

and

$$u(t) = 0.2 + 0.16t + 0.048t^2 + 1.0667\times10^{-3}t^3 \qquad [t_0, t_1] = [0, 0.4]$$

First three coefficients exact; last rounded to 4 digits

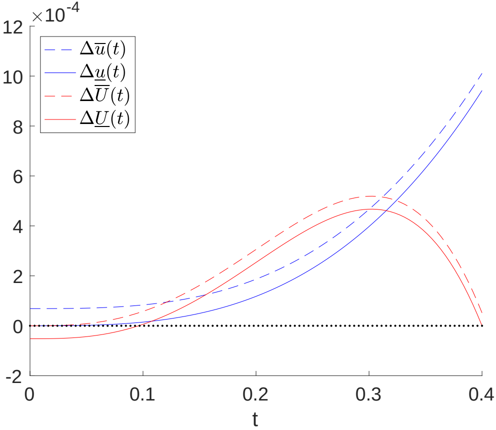
Interpolating $f(t_1, u(t_1))$ (4 digits)

$$U(t) = v(t) + 1.5795\times10^{-2}t^4 - 3.9486\times10^{-2}t^5$$

Evaluating $x' - (x - x^2)$ with $(U, [0, 0])$: $\Delta U(t) - p(t) \in \boldsymbol{r}$ on $[0, 0.4]$

$$\begin{aligned} p(t) &= 1.3878\times10^{-17}t + 10^{-10}t^2 + 7.7898\times10^{-2}t^3 \\ &\qquad - 2.0426\times10^{-1}t^4 + 2.8849\times10^{-2}t^5 \\ \boldsymbol{r} &= [-5.1923\times10^{-5}, 1.8090\times10^{-17}] \end{aligned}$$

Figure 1: Enclosures of $\Delta u(t)$ (blue) and $\Delta U(t)$ (red)



Phase III: Accept/reject step

We use "elementary controller"

(a) $\delta_{\max} = \max_i \delta_i$, $\|\Delta u_i\|_\infty \leq \delta_i$

If $\delta_{\max} \leq \text{tol}$, accept step and predict

$$h_{n+1} = 0.9 h_n \left(\frac{0.5\text{tol}}{\delta_{\max}} \right)^{1/K} \qquad K \text{ order of defect}$$

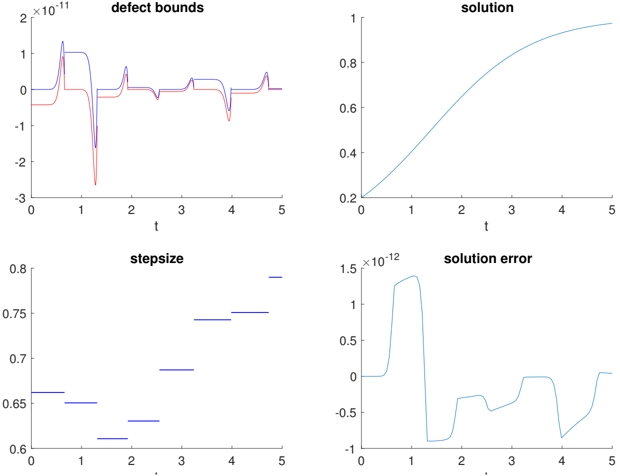
(b) else reject step and recompute $\delta_{\max}$ with

$$h_n \leftarrow h_n \left(\frac{0.25\text{tol}}{\delta_{\max}} \right)^{1/K}$$

- That is, repeat from Phase I(b)
- This involves evaluating $x' - f(t, x)$ in TM arithmetic
- Taylor coefficients are not recomputed

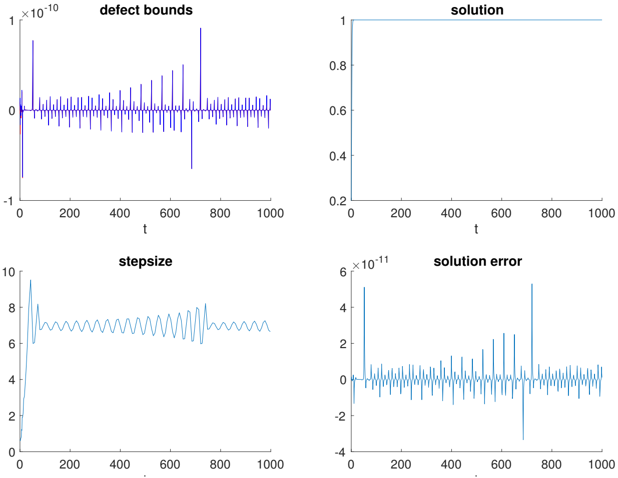
Example: defect controlled $x' = x - x^2$, $x(0) = 0.2$

Figure 2: $t_{\text{end}} = 5$, order 15, tol $= 10^{-10}$



Example: defect controlled $x' = x - x^2$, $x(0) = 0.2$

Figure 3: $t_{\text{end}} = 1000$, order 15, tol $= 10^{-10}$



Why not interval arithmetic (IA) evaluation?

- Because is not very good
- IA evaluation: replace reals by intervals and execute in IA
- IA operations

$$\boldsymbol{a} \bullet \boldsymbol{b} = \{\, a \bullet b \,|\, a \in \boldsymbol{a},\, b \in \boldsymbol{b}, \text{ and } a \bullet b \text{ is defined} \,\}$$

- Evaluating $\Delta u = u' - (u - u^2)$ in IA gives

$$u([0, 0.4]) \in \boldsymbol{u} = [0.2000, 0.2722]$$
$$u'([0, 0.4]) \in \boldsymbol{u}' = [0.1599, 0.2030]$$
$$\Delta u \in \boldsymbol{u}' - (\boldsymbol{u} - \boldsymbol{u}^2) = [-0.0722, 0.0771]$$

- Inexpensive to compute but the width of $[-0.0722, 0.0771]$ is 1.4917×10^{-1}
  Bounds can blow up for complicated f's
- Width of $\boldsymbol{r}$ is 5.1923×10^{-5}
  TM keep bounds small

ODETS: Putting it all together

C++ implementation

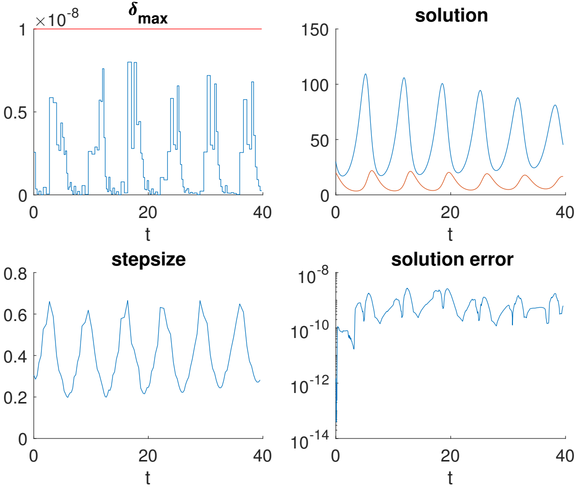
- ODETS class implements the integration scheme
- User provides ODE function, e.g.

```
template <typename T>
void fcn( T t, const T * x, T * xp )
{
  xp[0] =  x[0] -  0.1*x[0]*x[1] +  0.02*t;
  xp[1] = -x[1] + 0.02*x[0]*x[1] + 0.008*t;
}
```

- FADBAD++ uses fcn to generate computational graph
- Taylor coefficients are computed through FADBAD++
- Tmodel class interfaces SOLLYA and overloads arithmetic operators and elementary functions
  fcn is executed with Tmodel objects

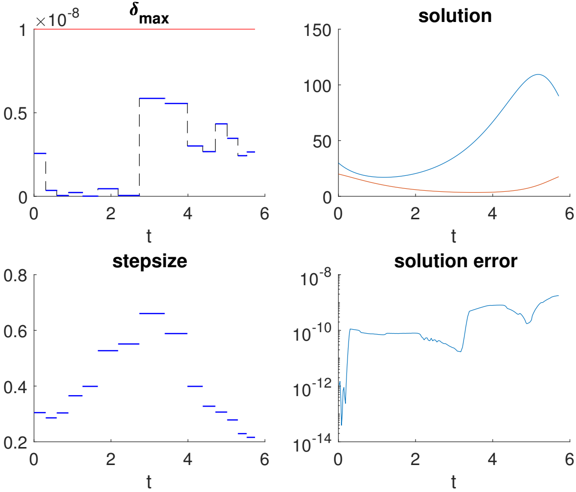
Defect controlled Predator-Prey

Figure 5: Order 14, tol $= 10^{-8}$



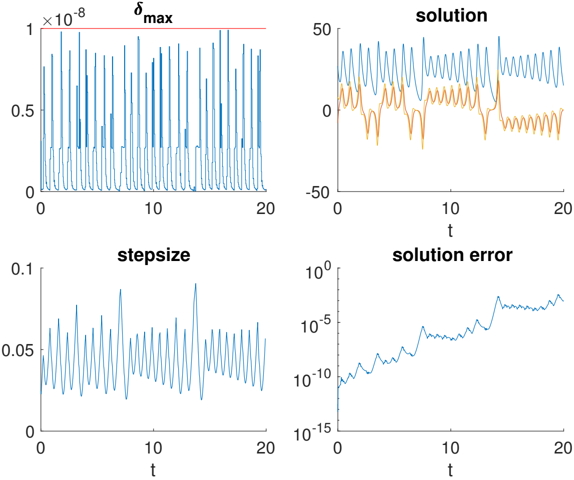
Defect controlled predator-prey, zoom in

Figure 6: Order 14, tol $= 10^{-8}$



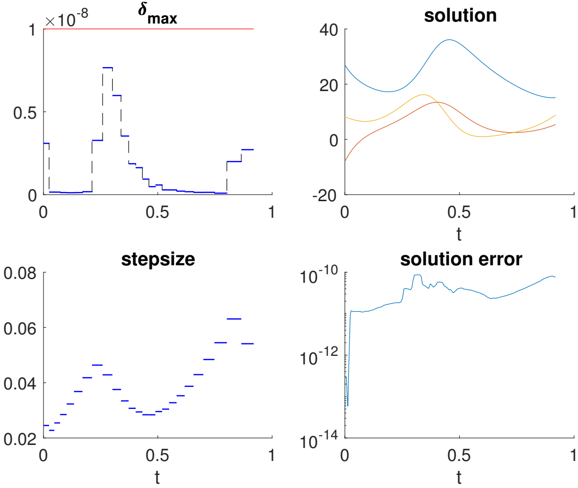
Defect controlled Lorenz

Figure 7: Order 14, tol $= 10^{-8}$



Defect controlled Lorenz, zoom in

Figure 8: Order 14, tol $= 10^{-8}$



Accepted/rejected steps

- We can keep $\delta_{\max}$ below tol
- We need to keep it closer to tol
- Generally, we can have too many stepsize rejections

		Lorenz $t_{\text{end}} = 20$		pred. prey $t_{\text{end}} = 40$	
order	tol	acc	rej	acc	rej
15	10^{-6}	356	79	80	23
	10^{-8}	465	65	103	20
	10^{-10}	612	25	135	15
	10^{-12}	814	2	179	15
20	10^{-6}	266	70	62	19
	10^{-8}	325	80	76	22
	10^{-10}	399	81	92	25
	10^{-12}	508	57	114	28

Conclusions

- Defect encapusulates all errors, well studied problem in IA
- RPA provides better bounds than IA for residual-based backward error analysis
- Given Taylor model arithmetic and RPA for sup-norm of polynomial, get sup-norm for real-valued function
- We can bound the defect rigorously and guarantee it is within tolerance
- We need to understand stepsize control better, construct a better one