Automatic Differentiation in Ordinary Differential Equations

Joseph Gergaud, Toulouse Univ., INP-ENSEEIHT-IRIT, UMR CNRS 5505, 2 rue Camichel, 31071 Toul

November 5, 2025

Abstract

1 Introduction

The objective of the article is to present the goods methods for computing the derivatives of the flow in the final time t_f with respect of the initial condition, a parameter or the initial time t_0. In the second section we present the different possibilities for computing these derivatives and in the third section we make some numerical comparaisons.

2 Mathematical results

Let f a continuous differentiable function from an open Ω of $\mathbf{R} \times \mathbf{R}^n \times \mathbf{R}^p$ into $\mathbf{R}^n$ We note $x(t, t_0, x_0, \lambda)$ the solution at time t of the following initial value problem

$$(IVP) \begin{cases} \dot{x} = f(t, x, \lambda) \\ x(t_0) = x_0(\lambda) \end{cases}$$

which is defined on the intervalle $]\omega_-(t_0, x_0, \lambda), \omega_-(t_0, x_0, \lambda)[$. We denote

$$\mathcal{O} = \{(t, t_0, x_0, \lambda) \in \mathbf{R} \times \Omega, \omega_-(t_0, x_0, \lambda) < t < \omega_+(t_0, x_0, \lambda)\}.$$

which is an open set.

Théorème 2.1. *Let $f : \Omega \subset \mathbf{R} \times \mathbf{R}^n \times \mathbf{R}^p \to \mathbf{R}^n$, Ω open, f continue. We suppose that f have continuous partial derivatives with respect to x and λ*

$$\frac{\partial f}{\partial x} : \Omega \to \mathcal{M}_n(\mathbf{R}) \quad \frac{\partial f}{\partial \lambda} : \Omega \to \mathcal{M}_{(n,p)}(\mathbf{R}).$$

Then the function $x : \mathcal{O} \to \mathbf{R}^n$ is C^1 (it is C^{k+1}, $1 \leq k \leq \infty$, if f is C^k) and

(i) the function $\frac{\partial x}{\partial x_0}(., t_0, x_0, \lambda_0) :]\omega_-(t_0, x_0, \lambda_0), \omega_+(t_0, x_0, \lambda_0)[\to \mathcal{M}_n(\mathbf{R})$ is the solution of the linear ordinary differential system

$$(VAR1)^1 \begin{cases} \dot{\widehat{\delta x}}(t) = A(t)\delta x(t) \\ \delta x(t_0) = I_n, \end{cases}$$

where $A(t) = \frac{\partial f}{\partial x}(t, x(t, t_0, x_0, \lambda_0), \lambda_0) \in \mathcal{M}_n(\mathbf{R})$ and I_n is the identity matrix of order n.

(ii) the function $\frac{\partial x}{\partial t_0}(., t_0, x_0, \lambda_0) :]\omega_-(t_0, x_0, \lambda_0), \omega_+(t_0, x_0, \lambda_0)[\to \mathbf{R}^n$ est is the solution of the linear ordinary diffferential equation

$$(VAR2) \begin{cases} \dot{\widehat{\delta x}}(t) = A(t)\delta x(t) \\ \delta x(t_0) = -f(t_0, x_0, \lambda_0). \end{cases}$$

(iii) the function $\frac{\partial x}{\partial \lambda}(., t_0, x_0, \lambda_0) :]\omega_-(t_0, x_0, \lambda_0), \omega_+(t_0, x_0, \lambda_0)[\to \mathcal{M}_{n,p}(\mathbf{R})$ is the solution of the linear ordinary differential equation

$$(VAR3) \begin{cases} \dot{\widehat{\delta x}}(t) = A(t)\delta x(t) + B(t) \\ \delta x(t_0) = x_0'(\lambda), \end{cases}$$

with $B(t) = \frac{\partial f}{\partial \lambda}(t, x(t; t_0, x_0, \lambda_0), \lambda_0) \in \mathcal{M}_{n,p}(\mathbf{R})$ and $0_{n,p}$ is the 0 matrix of order (n, p).

[1] Variationnal equation.

3 Numerical results on the brusselator example

3.1 Introduction

In all this section we test the methods on the the Brusselator example of The Hairer and al. book [2], page 201. These results with standard options for the numerical integration.

$$(IVP)\begin{cases} \dot{x}_1 = 1 + x_1^2 x_2 - (\lambda+1)x_1 \\ \dot{x}_2 = \lambda x_1 - x_1^2 x_2 \\ x_1(0) = 1.3 \\ x_2(0) = \lambda. \end{cases}$$

3.2 Finite differences

We approximate the dérivatives by finite differences. For example for the derivative with respect to the *jth* component of the parameter $\lambda \in \mathbf{R}^n$ is approximates by

$$\frac{\partial x}{\partial \lambda_j}(t, t_0, x_0, \lambda_0) \approx \frac{1}{\delta\lambda}(x(t, t_0, x_0, \lambda_0 + \delta\lambda e_j) - x(t, t_0, x_0, \lambda_0)),$$

where $(e_1, \ldots, e_p)$ denote the canonical basis of $\mathbf{R}^p$.

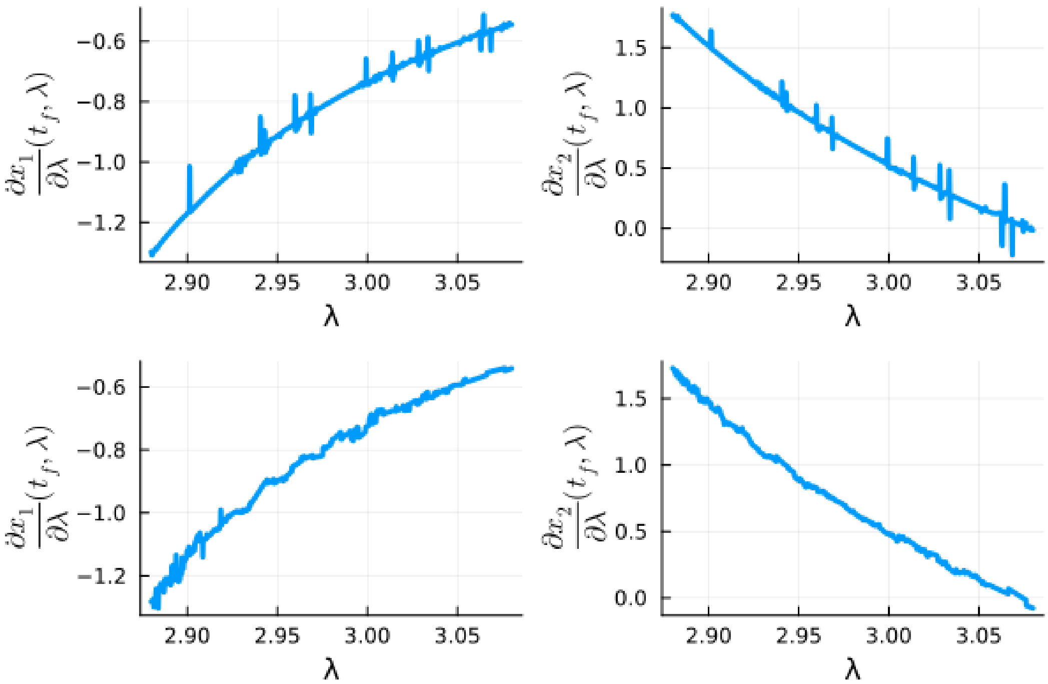

Figure 1: Derivative computing by finite differences. $t_f = 20, \lambda$ ranging from 2.88 to 3.08, $Tol = RelTol = AbsTol = 10^{-4}$. Top graphs is for $\delta\lambda = 4Tol$ and bottom graphs for $\delta\lambda = \sqrt{Tol}$. The numerical integrattion is done with Tsit5().

3.3 Variationnal equation

Here the derivative $\frac{\partial x}{\partial \lambda}(t, t_0, x_0, \lambda_0)$ is the solution of the variational equation

$$(VAR1)\begin{cases} \dot{\widehat{\delta x}}(t) = A(t)\delta x(t) + B(t) \\ \delta x(t_0) = x_0'(\lambda), \end{cases}$$

with $B(t) = \frac{\partial f}{\partial \lambda}(t, x(t; t_0, x_0, \lambda_0), \lambda_0) \in \mathcal{M}_{n,p}(\mathbf{R})$.

Remarque 3.1. In the variationnal equations we have the initial flow, so for numerically computing the solution of the variational equation, we must add the equation of the initial value problem.

Remarque 3.2. We can also approximate $A(t)$ and $B(t)$ by finite difference. But know, as we can use automatic differentiation for computing them, we don't test this possibility.

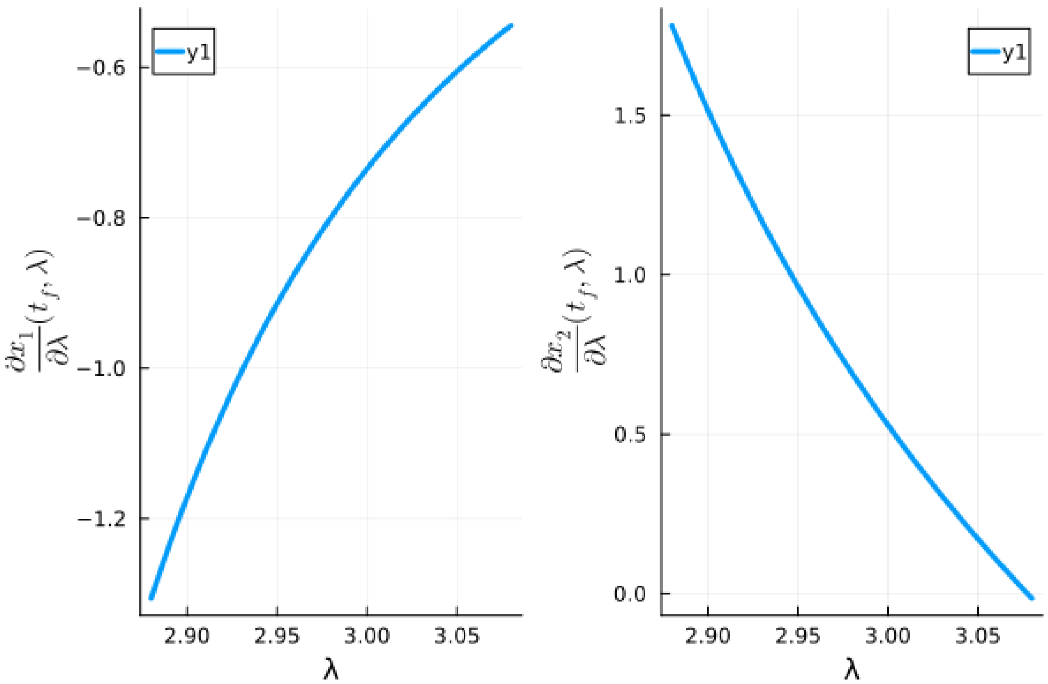

Figure 2: Derivative computing with the variational equation. $t_f = 20, \lambda$ ranging from 2.88 to 3.08, $Tol = RelTol = AbsTol = 10^{-4}$. The numerical integrattion is done with Tsit5().

3.4 Automatic differentiation of the flow

Here, we use automatic differentiation on the function $\varphi(\lambda) = x(t_f, t_0, x_0(\lambda), \lambda)$. This is historically also known as the Internal Numerical Differentiation of Bock [1]

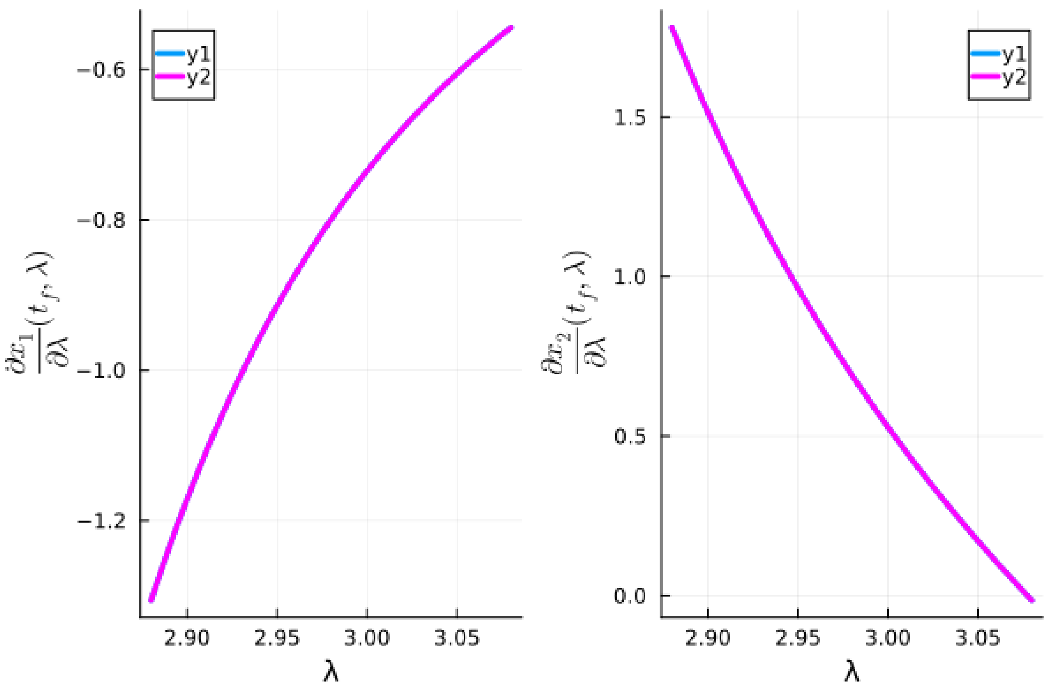

Figure 3: Derivative computing by automatic differentiation of the flow. $t_f = 20, \lambda$ ranging from 2.88 to 3.08, $Tol = RelTol = AbsTol = 10^{-4}$. The numerical integrattion is done with Tsit5(), the automatic differentiation is ForwardDiff.

Remarque 3.3. Historically, automatic differentiation of the flot is known as the Interne Numerical Differentiation

[1].

4 Tests on the time steps

4.1 Example

Here we use the following example with $\lambda = (1, 2)$

$$(IVP) \begin{cases} \dot{x}_1 = \lambda_1 x_1 \\ \dot{x}_2 = \lambda_2 x_2 \\ \dot{x}_3 = (\lambda_1 - \lambda_2) x_3 \\ x_1(0) = \lambda_2 \\ x_2(0) = 1 \\ x_3(0) = 1. \end{cases}$$

So the flow is

$$x(t, 0, x_0(\lambda), \lambda) = \begin{pmatrix} \exp(\lambda_1 t) & 0 & 0 \\ 0 & \exp(\lambda_2 t) & 0 \\ 0 & 0 & \exp((\lambda_1 - \lambda_2)t) \end{pmatrix} x_0(\lambda) = \begin{pmatrix} \lambda_2 \exp(\lambda_1 t) \\ \exp(\lambda_2 t) \\ \exp((\lambda_1 - \lambda_2)t) \end{pmatrix}.$$

And the dérivatives are

$$\frac{\partial x}{\partial x_0}(t, t_0, x_0(\lambda), \lambda) = \begin{pmatrix} \exp(\lambda_1 t) & 0 & 0 \\ 0 & \exp(\lambda_2 t) & 0 \\ 0 & 0 & \exp((\lambda_1 - \lambda_2)t) \end{pmatrix},$$

and

$$\frac{\partial x}{\partial \lambda}(t, t_0, x_0(\lambda), \lambda) = \begin{pmatrix} \lambda_2 t \exp(\lambda_1 t) & \exp(\lambda_1 t) \\ 0 & t \exp(\lambda_2 t) \\ t \exp((\lambda_1 - \lambda_2)t) & -t \exp((\lambda_1 - \lambda_2)t) \end{pmatrix}.$$

4.2 Step grids with variationnal equations

The numerical results are listed in the table 1.

For the step size selection [2], the error is

$$err = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_{1i} - \widehat{x}_{1i}}{sci} \right)^2},$$

where $sci = abstol_i + \max(|x_{0i}|, |x_{1i}|) reltol_i$. So if we take the value for the $abstol_i$ and $reltol_i$ which correspond to the components of the variational equation to Inf and we replace the other value by $reltol_i/sqrtp + 1$ and $abstil_i/sqrtp + 1$ (because there are $n(p+1)$ equation in the variational equation), then the step control for the initial flow and the variational equation are the same.

- $reltol = abstol = 1.e-4$
- $$RelTol = AbsTol = [reltol * ones(n, 1) my\_Inf * ones(n, 2)]/sqrt(p + 1),$$

  where

  - n is the number of the equation for the initial value problem, 3 here,
  - p in the number of parameter, 2 here,
  - my_Inf = prevfloat(typemax(Float64)) is the biggest float number. We take this in place of Inf because $0 * Inf = NaN$.
- :ivp : initial flow;
- :var_reltol : abstol=abstol and reltol = reltol;
- :var_reltol_internalnorm : abstol=abstol, reltol = reltol and internalnorm = $(u, t) -> norm(u)$;
- var_RelTol : abstol=AbsTol and reltol = RelTol;

```
Dict{Symbol, Vector{Float64}} with 4 entries:
  :var_RelTol              => [0.0, 0.0603328, 0.172903, 0.30947, 0.475902, 0.666691, 0.881442, 1.0]
  :var_reltol              => [0.0, 0.0537226, 0.147041, 0.257638, 0.393494, 0.549002, 0.725585, 0.9208
  :ivp                     => [0.0, 0.0603328, 0.172903, 0.30947, 0.475902, 0.666691, 0.881442, 1.0]
  :var_reltol_internalnorm => [0.0, 0.0431253, 0.117812, 0.206115, 0.314575, 0.438265, 0.578163, 0.7319
```

Table 1: *Results with variationnal equations.*

4.3 Step grids with automatic differentiation of the flow

The numerical results are listed in the table 2.

- :ivp : initial flow;
- :diff_auto_Zygote : automatic differentiation with Zygote;
- :diff_auto_ForwardDiff : automatic differentiation with ForwardDiff;
- :diff_auto_ForwardDiff_internalnorm : automatic differentiation with ForwardDiff and internalnorm $= (u, t) - > norm(u)$.

```
Dict{Symbol, Vector{Float64}} with 4 entries:
  :diff_auto_Zygote                    => [0.0, 0.0603328, 0.172903, 0.30947, 0.475902, 0.666691, 0.8814
  :diff_auto_ForwardDiff_internalnorm => [0.0, 0.0540557, 0.154756, 0.276759, 0.425512, 0.59589, 0.7875
  :diff_auto_ForwardDiff              => [0.0, 0.0611775, 0.167513, 0.293407, 0.447446, 0.622839, 0.820
  :ivp                                => [0.0, 0.0603328, 0.172903, 0.30947, 0.475902, 0.666691, 0.8814
```

Table 2: it Results with automatic differentiation.

4.4 Comparaison of the solutions for the methods with the same step grid

The solution at tf of the jacobian are

- exact solution

```
5.43656   2.71828
0.0       7.38906
0.367879  0.367879
```

- with the variational equation

```
5.43656    2.71828
0.0        7.38905
0.367879  -0.367879
```

- with the automatic differentiation with `ForwardDiff`

```
5.43656    2.71828
0.0        7.38905
0.367879  -0.367879
```

- with the automatic differentiation with `Zygote`

```
5.43656   2.71828
5.43656  10.1073
5.80444   9.73945
```

4.5 conclusion

- The use of the option internalnorm as mentioned by Chris don't give the good results for the steps grid !
- The use of Zygote with the numerical integration is not good !

5 Conclusion and perspectives

References

[1] H.G. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In K.H. Hebert, P. Deuflhard, and W. Jäger, editors, *Modelling of chemical reaction systems*, volume 18 of *Springer series in Chem. Phys.*, pages 102–125, 1981.

[2] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I, Nonstiff Problems*, volume 8 of *Springer Serie in Computational Mathematics*. Springer-Verlag, second edition, 1993.