1 Cross-Derivatives

A *cross-derivative* of a sufficiently smooth function $f : D \subset \mathbb{R}^N \to \mathbb{R}$ is a partial derivative with only mixed derivatives, i.e. $f_{\mathbf{i}}(x)$ with $\mathbf{i} \in \{0,1\}^N$ in contrast to $\mathbf{i} \in \mathbb{N}_0^N$ for arbitrary partial derivatives. Restricting the multi-index in that way allows simpler addressing schemes on a computer. Use [Gri08] as reference. This technique should lead to a significant performance improvement.

The algebraic structure that is propagated through the computational graph is the quotient

$$\mathbb{R}[\varepsilon_1, \ldots, \varepsilon_N] / \langle \varepsilon_1^2, \ldots, \varepsilon_N^2 \rangle$$

Elements of this algebra have the general form

$$v = \sum_{\mathbf{i} \in \{0,1\}^N} v_{\mathbf{i}} \varepsilon^{\mathbf{i}} = v^0 + \varepsilon_n v^1$$

where v^0 and v^1 only contain the generators $\varepsilon_1, \ldots, \varepsilon_{n-1}$. The general computational rule is that $(\varepsilon_j)^k = 0$ whenever $k > 1$.

The coefficients may now be addressed as a linear array where the coefficient $v_{\mathbf{i}}$ is located at the index $\sum_{k=1}^N \mathbf{i}_k 2^{k-1}$ corresponding to the interpretation of the tupel $\mathbf{i}$ as a binary bit sequence. Thus the coefficient vectors of v^0 and v^1 are the upper and lover halves of the coefficient vector of v.

Now one can always split any operation along the $v = (v^0 | v^1)$ boundary. For instance a multiplication

$$v \cdot w = v^0 \cdot w^0 + \varepsilon_n (v^0 \cdot w^1 + v^1 \cdot w^0)$$

reduces to three multiplication of the half vectors. Nonlinear functions can be represented via their Taylor development. Since all higher powers of ε_n are zero, the Taylor series reduces to the Taylor polynomial of degree one, so one gets:

$$\begin{aligned}
\cos(v) &= \cos(v^0) - \varepsilon_n \sin(v^0) \cdot v^1, \\
\sin(v) &= \sin(v^0) + \varepsilon_n \cos(v^0) \cdot v^1; \\
\exp(v) &= \exp(v^0) \cdot (1 + \varepsilon_n v^1), \\
\log(v) &= \log(v^0) + \varepsilon_n (v^0)^{-1} \cdot v^1 \\
v^{-1} &= (v^0)^{-1} \cdot (1 - \varepsilon_n (v^0)^{-1} \cdot v^1).
\end{aligned}$$

Those formulas suggest recursive implementations of these operations as outlined in [Gri08].

1. Implement a tool to propagate cross-derivatives in the forward mode of AD.
2. Do a thorough speed comparison between your implemented code and exact interpolation. For the exact interpolation you can use the ADOL-C functions hov_forward.
3. Visualize your findings in an appropriate way, e.g. with Gnuplot.
4. Write a concise technical report using LaTeX including introduction, theory, implementation, experiment, summary. The report should be under 10 pages.
5. Give a 10 minutes talk.

2 Differentiation of the Explicit Euler with Adaptive Stepsize Control

Let the solution trajectory $\phi(t,p) \in \mathbb{R}^N$ of an ODE

$$\begin{aligned} \dot{x} &= f(t,x,p) \\ x(0) &= x_0(p) \end{aligned}$$

where $f \in C^{D,1}$ depend on the parameter $p \in \mathbb{R}^{N_p}$. It is desired to obtain the derivatives $\frac{\partial^D}{\partial p^D}\phi(t,p)$. To obtain these derivatives, use univariate Taylor propagation to differentiate the explicit Euler method

$$x_{n+1} = x_n + h_n f(t_n + h_n, x_n, p) \ ,$$

where h_n is the stepsize from x_n to x_{n+1}. I.e. we have

$$t_{n+1} = t_n + h_n$$

The explicit Euler derives directly from Taylor's theorem:

$$x(t_n + h_n) = x(t_n) + h_n f(t_n, x(t_n)) + \frac{h_n^2}{2} f'(\eta, x(\eta), p), \qquad t_n \le \eta \le t_n + h_n$$

by neglecting the remainder term. The local error is

$$\varepsilon_{n+1} := x_{n+1} - x(t_{n+1}) = \frac{h_n^2}{2} f'(\eta, x(\eta), p) \ .$$

Since η is unknown ε_{n+1} has to be estimated by $\hat{\varepsilon}_{n+1}$, typically up to a certain order of h_n. In practice, one can use the following estimator of the local error

$$\begin{aligned} \hat{\sigma}(h; t_n, x_n) &:= x^1_{n+1} - x^2_{n+1} \\ &= \hat{c}_n h^{p+1} + \mathscr{O}(h^{p+2}) \ , \end{aligned}$$

where x^1_{n+1} and x^2_{n+1} are solutions of two methods Φ^1 and Φ^2 with different consistency order:

$$\begin{aligned} x^1_{n+1} &= x_n + h\Phi^1(h; t_n, x_n) \text{consistency order } p \\ x^2_{n+1} &= x_n + h\Phi^2(h; t_n, x_n) \text{consistency order } p+1 \ . \end{aligned}$$

One then defines a measure of the error

$$E_n(h) := \|\hat{\sigma}(h; t_n, x_n)\|$$

and one accepts if $E_n(h) \le TOL$. Let h_k been accepted in the last step. Use the ansatz $E_n(h_n) = c_n h_n^{p+1}$. Then compute the new step size h_{new} as

$$\begin{aligned} E_{n+1}(h_{\text{new}}) &= c_{n+1} h_{\text{new}} h^{p+1} \le TOL \\ &= \gamma\, TOL \quad \text{when } c_n = c_{n+1} \text{ is assumed.} \end{aligned}$$

The final update formula is therefore

$$\begin{aligned} h_{\text{new}} &= \left(\frac{\gamma\, TOL}{c_n}\right)^{\frac{1}{p+1}} \\ &= \tau \left(\frac{TOL}{E_n(h_n)}\right)^{\frac{1}{p+1}} h_n \ , \end{aligned}$$

where γ and τ are safety factors. They are typically $\gamma = 0.5$ and $\tau = 0.9$. For reference you can have a look at the lecture script by Bock [**?**] and the Wikipedia article [WikiAdapt].

It is an open question what happens if an integrator with adaptive stepsize control is differentiated. See Figure 2.1 for a short discussion and [Lang08] for a more detailed discussion.

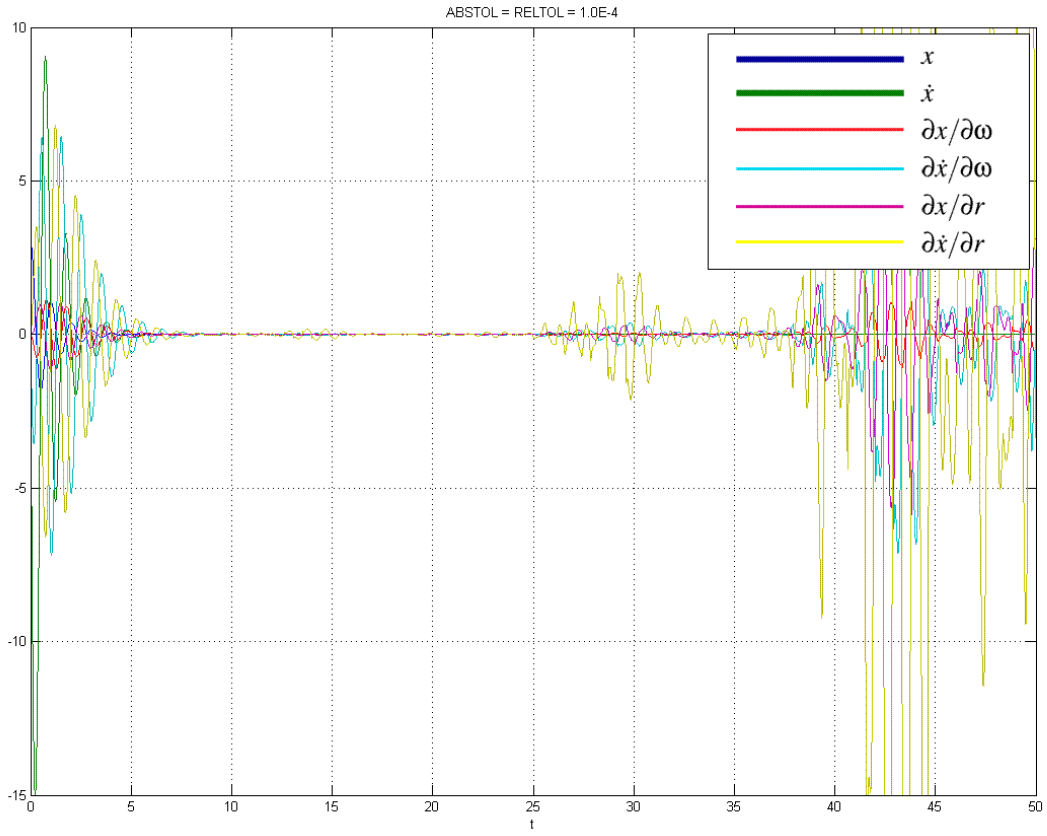


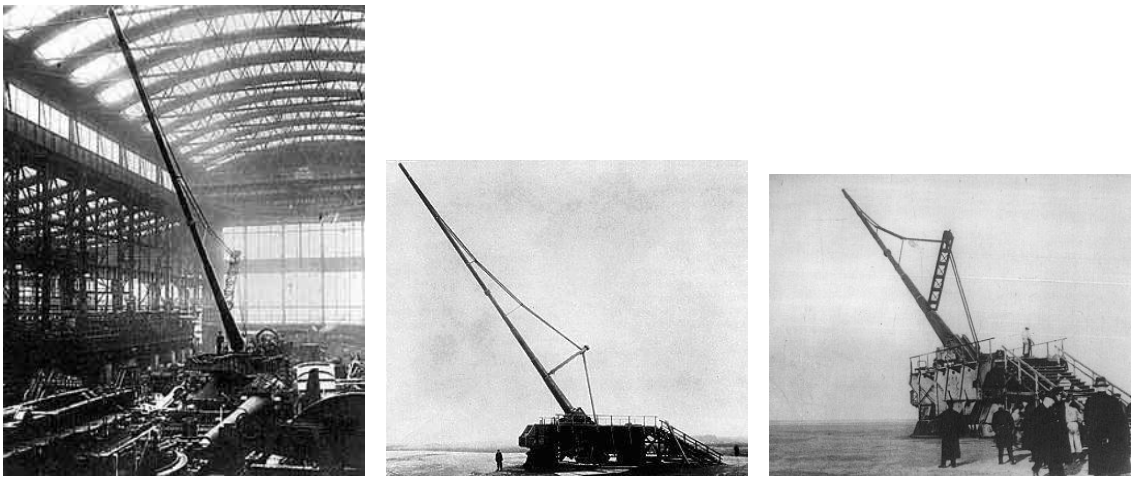
Fig. 2.1: This is solution of the Rungke Kutta DE solver ODE45 available in Matlab which uses a Runge Kutta method of order 4 and 5 to obtain an estimate of the local error. From the known analytical solution one knows that amplitude of the solution and its derivative should decay exponentially. However, at $t = 25$ one can see that the derivatives start to oscillate wildly.

1. Implement a tool to propagate univariate Taylor polynomials in the forward mode of AD. Start with fixed stepsizes $h_n = h = const.$. Then enhance your code to allow variable stepsizes. For that look up a reliable stepsize control in the literature.
2. Derive the analytic solution of the damped harmonic oscillator $\ddot{x} = -2r\dot{x} - \omega^2 x$, $x(0) = x_0$ and $\dot{x}(0) = v_0$ and compare it with the approximate solutions. The parameters are $p = (r, \omega)$.
3. Visualize your findings in an appropriate way, e.g. with Gnuplot.
4. Write a concise technical report using LaTeX including introduction, theory, implementation, experiment, summary. The report should be under 10 pages.
5. Give a 10 minutes talk.

3 Differentiation of the Backward-Euler Method by Internal Numeric Differentiation

Let the solution trajectory $\phi(t, p) \in \mathbb{R}^N$ of an ODE

$$\begin{aligned} \dot{x} &= f(t, x, p) \\ x(0) &= x_0(p) \end{aligned}$$



(a) (b) (c)

Fig. 3.1: The *Paris gun* (aka *Big Bertha*) that was used in World War I to besiege Paris.

depend on the parameter $p \in \mathbb{R}^{N_p}$. It is desired to obtain the derivatives $\frac{\partial^d}{\partial p^d}\phi(t,p)$. To obtain these derivatives, use univariate Taylor propagation to differentiate the backward-Euler method

$$x_{n+1} = x_n + h_n f(t_n + h_n, x_{n+1}) \,,$$

where h_n is the stepsize from iterate x_n to iterate x_{n+1}. To compute the Taylor series of the rhs f in C++ you can use the ADOL-C routine `hov_forward` or PYADOLC if you use Python. You can, of course, use any other tool that supports UTP. The backward Euler method yields a nonlinear implicit system of equations that have to be solved. As references on internal numeric differentiation you can use [Alb05] and [Rue99] for the theory on the differentiation of nonlinear implicit systems.

3.1 Test Case: A Shooting Problem

Assume we want to hit a point TU with a Paris Gun (c.f. 3.1). We want to know at which angle θ we have to shoot. The problem is depicted in Figure 3.2.

The movement of the projectile is defined by a second order, nonlinear ordinary differential equation (ODE). Formally:

$$\begin{cases} \ddot{x} = \begin{pmatrix} 0 \\ -a_G \end{pmatrix} - \frac{1}{2m} c_w A \rho |\dot{x}| \dot{x} \\ x(0) = x_0 \,, x(T) = r \,. \end{cases} \tag{3.1}$$

The constants are described in Table 3.1 and $x(t) \in \mathbb{R}^2$, $\dot{x}(t) = \frac{d}{dt}x(t)$ denotes the position and the velocity of the projectile. Written as first order ODE:

$$\dot{z}(t) = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} y \\ \begin{pmatrix} 0 \\ -a_g \end{pmatrix} - \frac{1}{2m} A \rho c_w |y| y \end{pmatrix} = g(z,t) \,. \tag{3.2}$$

The approximation z_k depends on the initial values z_0. The initial position $(x_0, x_1) = (0,0)$ and the speed of the projectile v_0 are fixed. A variable parameter is the angle θ. In Figure 3.2 one can see the trajectories of projectiles shot with different angles.

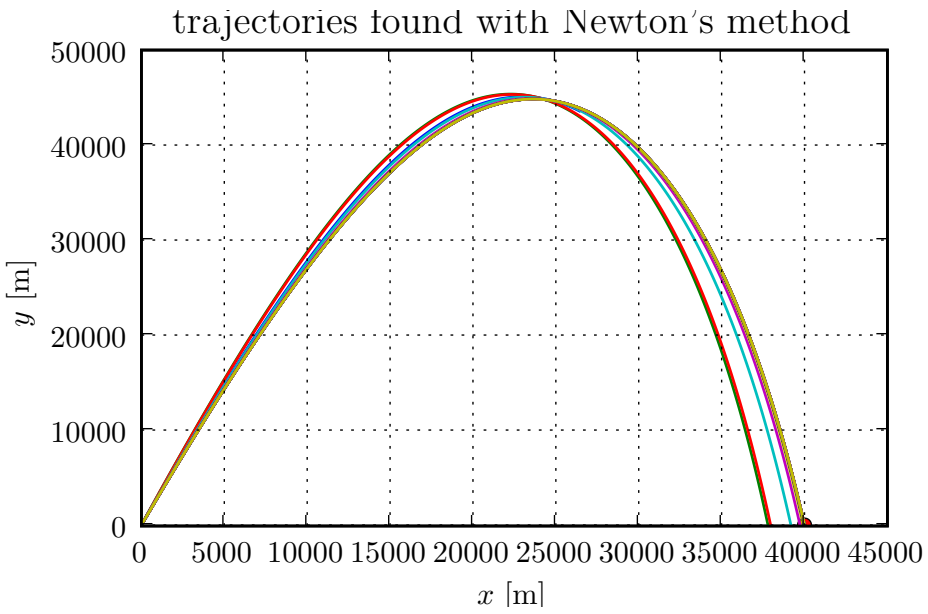


Fig. 3.2: The goal is to hit a point TU with a cannon located at the origin $(0,0)$. The initial velocity is fixed at v_0 and we are only free to vary the angle θ at which the projectile is ejected. In the above plot we used the bisection method to find a good guess for the angle θ.

constant	notation	value	unit
Earth's gravity constant	a_g	9.81	$\frac{m}{s^2}$
initial velocity	v_0	1600	$\frac{m}{s}$
air drag factor	c_w	0.15	
density of air	ρ	1.3	$\frac{kg}{m^3}$
mass of projectile	m	194	kg
area of projectile	A	$\pi(0.105)^2$	m^2

Tab. 3.1: The parameters we are using in our simulation. The true values have been lost during war and only estimates are available. We adapted the values from Wikipedia [Wiki08].

3.2 Finding The Best Angle with Newton's Method

We define our objective function as a least squares problem

$$0 \stackrel{!}{=} f(\theta) := \min_t \underbrace{\| z_\theta(t) - \begin{pmatrix} R \\ 0 \end{pmatrix} \|_2^2}_{=f(t,\theta)} . \tag{3.3}$$

However, this is not a least-square problem we can handle easily because of the $\min_t$. There are several ways to reformulate the problem. One way is to perform a linear transformation on the time such that the end time is always 1 and not the unknown time T. The transformation reads

$$t_{\text{new}} = \frac{1}{T} t_{\text{old}} \Leftrightarrow \frac{d}{dt_{\text{old}}} = \frac{dt_{\text{new}}}{dt_{\text{old}}} \frac{d}{dt_{\text{new}}} = \frac{1}{T} \frac{d}{dt_{\text{new}}} , \tag{3.4}$$

where t_{new} is the time measured by a new clock and t_{old} the time measured by the old clock. Then the ODE can be reformulated in the following way

$$\begin{aligned}
\frac{d}{dt_{\text{old}}} z(t_{\text{old}}; p) &= g(z, t_{\text{old}}) && (3.5)\\
\frac{1}{T}\frac{d}{dt_{\text{new}}} z(T t_{\text{new}}; p) &= g(z, p, T t_{\text{new}}) && (3.6)\\
\frac{d}{dt_{\text{new}}} z_{\text{new}}(t_{\text{new}}; p, T) &= T g(z, p, T t_{\text{new}}) && (3.7)\\
\frac{d}{dt_{\text{new}}} z_{\text{new}}(t_{\text{new}}; p, T) &= g_{\text{new}}(z, p, T, t_{\text{new}})\,. && (3.8)
\end{aligned}$$

We now obmit the subscripts new. Therefore, the reformulated ODE we have to solve reads

$$\begin{cases} \frac{d}{dt} z(t; p) = g(z, p, t) \\ z(0) = z_0\,, z(1) = r \end{cases} \tag{3.9}$$

The objective function is

$$\begin{aligned}
p^* = (\theta^*, T^*) &= \text{argnull}_{\theta,T} f(\theta, T) = \text{argnull}_{\theta,T} z(1; (\theta, T)) - r && (3.10)\\
&\text{s.t.} \begin{cases} \frac{d}{dt} z(t; p) = g(z, p, t) \\ z(0) = z_0\,, z(1) = r \end{cases}, && (3.11)
\end{aligned}$$

where argnull are the arguments that make the expression zero. To solve this constrained optimization problem, we use Newton's method without globalization routines, i.e.,

$$\underbrace{\nabla f(p_k)}_{:=A} \underbrace{(p_{k+1} - p_k)}_{x} = \underbrace{-f(p_k)}_{:=b},$$

i.e., we have to solve the linear system $Ax = b$. For the method to converge, you need an initial guess that is close enough the exact solution to guarantee the convergence of Newton's method. If you want you can also use a standard solver with guaranteed global convergence.

3.3 Your Job

1. Implement a tool to propagate univariate Taylor polynomials in the forward mode of AD. Use fixed stepsizes, i.e. $h_n = h = const$.
2. Use the tool to solve the shooting problem. I.e. find the parameters p such that a cannon ball hits a predefined target.
3. Compare this to the optimization done by finite differences.
4. Visualize your findings in an appropriate way, e.g. with Gnuplot.
5. Write a concise technical report using LaTeX including introduction, theory, implementation, experiment, summary. The report should be under 10 pages.
6. Give a 10 minutes talk.

4 Numerical Quadrature

Similar to the AD book, second edition, p. 365. Prof. Griewank will explain the details in the exercise class.

5 Edge and Vertex Elimination

1. See the papers by U. Naumann [Nau04, Nau08] and the book (Griewank & Walther, Ch. 9,10) for details on elimination rules.
2. Implement a tool which stores a linearized computational graph for a given function at the current evaluation point and performs Edge and Vertex eliminations accorting to the relative greedy Markowitz strategy and the optimal pathlength reduction strategy.
3. Compare the complexity of the two heuristics with that of the Forward and Reverse Mode. Analyse the dependence of Round-off error propagation on the order of accumulations.
4. Visualize your findings in an appropriate way, e.g. with Gnuplot.
5. Write a concise technical report using LaTeX including introduction, theory, implementation, experiment, summary. The report should be under 10 pages.
6. Give a 10 minutes talk.

References

[Gil08] Giles, M.B : **Collected Matrix Derivative Results for Forward and Reverse Mode Algorithmic Differentiation**, Advances in Automatic Differentiation, Lecture Notes in Computational Science and Engineering (2008)

[Gri08] Griewank, A.: **Cross Derivatives** , unpublished

[Gay09] Gay, David M. **Semiautomatic Differentiation for Efficient Gradient Computations**

[Neu09] Neumaier's Homepage **http://www.mat.univie.ac.at/ neum/papers.html**

[NQ] **http://wj32.wordpress.com/2008/02/03/using-the-taylor-series-to-find-the-indefinite-integral-or-antiderivative-of-xx/**

[Nau04] U. Naumann, Optimal accumulation of Jacobian matrices by elimination methods on the dual computational graph. *Math. Programm.,* Ser. A **99** (399–421), 2004.

[Nau08] U. Naumann, Optimal Jacobian accumulation is NP-complete. *Math. Programm.,* Ser. A **112** (427–441), 2008.

[Alb05] Albersmeyer, J. Effiziente Ableitungserzeugung in einem adaptiven BDF-Verfahren, Diplomarbeit, 2005

[Rue99] Rücker, G. **Automatisches Differenzieren mit Anwendung in der Optimierung bei chemischen Reaktionssystemen**, Diplomarbeit, 1999

[Wiki08] http://en.wikipedia.org/wiki/Paris_gun,

[Lang08] Lang, **Technical Report on Adpative Stepsize Control of ODE Integrators**, included in the repository

[WikiAdapt] **http://en.wikipedia.org/wiki/Adaptive_stepsize**

[] Bock, H. G.**Lecture Script: Numerik 1**