Acta Appl Math (2020) 170:649–659
https://doi.org/10.1007/s10440-020-00351-9

Dual Numbers and Automatic Differentiation to Efficiently Compute Velocities and Accelerations

F. Peñuñuri[1] · R. Peón[2] · D. González–Sánchez[3] · M.A. Escalante Soberanis[4]

Received: 17 August 2019 / Accepted: 15 July 2020 / Published online: 21 July 2020
© Springer Nature B.V. 2020

Abstract Differentiation is one of the most common subjects of numerical calculations. Gradients and Hessians are used in many problems of the physical and engineering sciences. Automatic differentiation (AD) is usually employed when the accuracy in derivatives calculations is important. When AD is implemented, there are no truncation or cancellation errors. Therefore, the derivatives are calculated with the available machine precision. In this study, the forward mode of AD by using dual numbers is implemented to develop efficient methods for computing velocities and accelerations. It is known that the reverse mode of AD is more efficient than the forward mode of AD to compute gradients and Hessians. Nonetheless, gradients and Hessians are not directly required for the calculation of velocities and accelerations. However, directional derivatives and the action of the Hessian operator on specific vectors are required. Both operations can be efficiently computed through the use of dual numbers.

Keywords Automatic differentiation · Dual numbers · Velocities and accelerations

Mathematics Subject Classification 65D25 · 65Z05

1 Introduction

Automatic Differentiation (AD) [1–3] is an algorithmic way of applying the chain rule. It can be implemented essentially in two ways; the forward mode of AD, also known as

✉ F. Peñuñuri
francisco.pa@correo.uady.mx

1 Departamento de Ingeniería Física, Universidad Autónoma de Yucatán, A.P. 150, Cordemex, Mérida, Yucatán, Mexico

2 Departamento de Ingeniería Mecatrónica, Universidad Autónoma de Yucatán, A.P. 150, Cordemex, Mérida, Yucatán, Mexico

3 CONACYT–Departamento de Matemáticas, Universidad de Sonora, C.P 83000, Hermosillo, Sonora, Mexico

4 Departamento de Ingeniería en Energías Renovables, Universidad Autónoma de Yucatán, A.P. 150, Cordemex, Mérida, Yucatán, Mexico

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

the tangent mode, and the reverse mode of AD, also known as the adjoint mode of AD. These names come from the form in which the chain rule is used for computing derivatives [4]. There is a large number of studies on AD and some of them are reported in selected publications [5–18]. The reader can also consult the books [4, 19], where some works related to AD (including applications and implementations) are presented. However, the literature about the implementation of AD by using dual numbers is not so vast, we can mention [3], where a package for automatic differentiation (first order derivatives) of Fortran codes is included, and [20], where an extension of the dual numbers for calculating second order derivatives is developed. These studies deal with the derivatives of functions of a single variable. The computation of mixed derivatives for functions of several variables is presented in [21], following the ideas presented in [20]. Dual numbers provide a clear and easy way to implement the forward mode of AD. These numbers were introduced by Clifford, who also developed their algebra in the late nineteenth century [22]. Since then, dual numbers have been applied mainly in the mechanical description of rigid bodies [23]. However, a study relating dual numbers to field theory and supersymmetry is presented in [24].

The main disadvantage of the forward mode of AD compared to the reverse mode is its much lower efficiency when computing gradients and Hessians for functions of many variables [25]. Nevertheless, in many applications, it is not the gradient or the Hessian which are needed, but the directional derivative $\langle \nabla f(\mathbf{x}) | \mathbf{v} \rangle$ of a function $f(\mathbf{x})$ along the vector $|\mathbf{v}\rangle$, or the action $\mathbf{H} f(\mathbf{x}) | \mathbf{v} \rangle$ of the Hessian operator on a vector [26, 27]. Furthermore, the calculations of velocities and accelerations involve terms of the form $\langle \nabla f(\mathbf{x}) | \mathbf{v} \rangle$ and $\langle \mathbf{v} | \mathbf{H} f(\mathbf{x}) | \mathbf{v} \rangle$, which can be efficiently computed by using dual numbers, as will be illustrated later. Therefore, in this study, the implementation of AD using dual numbers is used to develop methods for computing velocities and accelerations. As an additional material to this article, the AD implementation in Fortran is available in [28].

A dual number is a quantity of the form $\hat{r} = a + \epsilon\, b$, with a and b being real numbers and ϵ the dual unit with the property $\epsilon^2 = 0$. These numbers do not form a field because a pure dual number of the form $\hat{r} = \epsilon\, b$ does not have a multiplicative inverse. Nevertheless, dual numbers form a commutative ring whose operations allow the calculation of first order derivatives. For this purpose, all the involved quantities need to be promoted to dual quantities with the independent variable having its dual part equal to one. If an algorithm for computing certain quantity is codified in the context of dual numbers, the information of the real value of such quantity along with its derivative are then automatically calculated. This is an interesting property that permits the calculation of function derivatives not necessarily given in a closed-form expression. As an application of this property, we code Müller's method for finding the roots of equations [29].

The paper is organized as follows. In Sect. 2, we briefly explain the step-size dilemma in numerical derivatives. The relationship (6) between dual numbers and the chain rule for derivatives is presented in Sect. 3. Section 4 deals with second order derivatives; in particular, a multiplication table is introduced and compared to a related implementation. In Sect. 5, we make a runtime comparison for computing directional derivatives between the reverse mode of AD and AD by using dual numbers. A method to compute efficiently velocities and accelerations is proposed in Sect. 6. Three applications are provided in Sect. 7. Finally, some concluding comments are presented in Sect. 8.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

2 Numerical Derivatives

The traditional approach to compute derivatives is by using finite differences. This method computes the derivatives by approximating the limit involved in its definition,

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}. \tag{1}$$

However, Eq. (1) is subject to truncation and subtractive cancellation errors. In principle, the truncation error could be eliminated by choosing a smaller h, but this increases the subtractive cancellation errors. Therefore, in practice, one faces the problem of choosing the right h, the so-called step-size dilemma in [30]. An interesting solution to this problem is to promote the $f(x)$ function in Eq. (1) to be a complex function of complex variable. This originates the so-called Complex-Step derivative approximation method presented in [30], where the first order derivative can be computed as:

$$f'(x_0) = \mathrm{Im}[f(x_0 + i\,h)]/h, \tag{2}$$

being $\mathrm{Im}[z]$ the imaginary part of the complex number z. Notice that there is not subtractive cancellation errors. Moreover, the truncation error is eliminated by choosing h as small as the computer program is able to compute. Generally, the order 10^{-16} is used for double precision numbers in a 64-bit machine. Using a smaller h is pointless since the precision is lost.

Unfortunately, the complex-step approximation method fails in computing higher order derivatives, even for simple functions. For instance, consider $f(x) = \exp(\sin x)$, $h = 1 \times 10^{-8}$, $x_0 = 1.1$ then

$$f''(x_0) = (2f(x_0) - 2\,\mathrm{Re}[f(x_0 + i\,h)])/h^2$$
$$f''(x_0) = 0,$$

where $\mathrm{Re}[z]$ stands for the real part of z. A better approximation can be achieved by choosing a larger h. For instance, taking $h = 1 \times 10^{-5}$, we obtain $f''(x_0) = -1.67119$. As we can see, the step-size dilemma appears again. There are more elaborated formulae for computing second order derivatives using a complex-step $i\,h$. However, the problem of choosing the right h never disappears.

3 Dual Numbers and Derivatives

Despite the above problems of truncation and subtractive cancellation errors, the derivatives calculated by AD have a precision given by the programming language used to calculate them. The implementation of AD using dual numbers is presented as follows.

From the expansion of an analytical function f in a Taylor series, we have

$$f(x + h) = f(x) + f'(x)h + O(h^2), \tag{3}$$

and evaluating at $\hat{x} = x + \epsilon$ we obtain (recall that $\epsilon^2 = 0$)

$$f(x + \epsilon) = f(x) + f'(x)\epsilon. \tag{4}$$

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Then, we obtain the dual function

$$\hat{f}(\hat{x}) = f(x) + f'(x)\epsilon, \tag{5}$$

which has the following feature: by evaluating f at the dual variable $\hat{x}$ we obtain both $f(x)$ and $f'(x)$. Considering f_0 as the real part of $\hat{f}$ and f_1 as the dual part of $\hat{f}$, we have, for a general composition of two functions,

$$\hat{f}(\hat{g}) = f_0(g_0) + f_1(g_0)g_1\,\epsilon. \tag{6}$$

In the Fortran programming language we can use

```
type, public :: dual
real(8) :: f0, f1
end type dual
```

to define this type of number.

4 Second Order Derivatives

Second order derivatives can be computed by extending the common dual numbers to a number of the form:

$$\tilde{r} = a + \epsilon_1 b + \epsilon_2 c \tag{7}$$

where a, b, and c are real numbers, and the definition of ϵ_1 and ϵ_2 satisfies the axioms of a commutative ring. This can be accomplished if ϵ_1 and ϵ_2 satisfy the multiplication table

	1	ϵ_1	ϵ_2
1	1	ϵ_1	ϵ_2
ϵ_1	ϵ_1	$2\epsilon_2$	0
ϵ_2	ϵ_2	0	0

. (8)

By evaluating the Taylor expansion of an analytic function f at the number $\tilde{x} = x + 1\,\epsilon_1 + 0\,\epsilon_2$, we obtain the function

$$\tilde{f}(\tilde{x}) = f(x) + f'(x)\epsilon_1 + f''(x)\epsilon_2. \tag{9}$$

In Eq. (9) we have a number encoding the real value of the function $f(x)$ and its first and second order derivatives. Taking the components of $\tilde{f}$ as f_0, f_1 and f_2, the general composition of functions will be given by

$$\tilde{f}(\tilde{g}) = f_0(g_0) + f_1(g_0)g_1\epsilon_1 + [f_2(g_0)g_1^2 + f_1(g_0)g_2)]\epsilon_2. \tag{10}$$

The dual number related to Eq. (10) can be defined in the Fortran programming language as

```
type, public :: dual2
real(8) :: f0, f1, f2
end type dual2
```


Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Another way to extend the dual numbers to calculate second order derivatives has been presented in [20]. These authors define the multiplication of the so-called hyper-dual numbers $a = a_0 + a_1\epsilon_1 + a_2\epsilon_2 + a_3\epsilon_1\epsilon_2$ and $b = b_0 + b_1\epsilon_1 + b_2\epsilon_2 + b_3\epsilon_1\epsilon_2$ as

$$ab = a_0b_0 + (a_0b_1 + a_1b_0)\epsilon_1 + (a_0b_2 + a_2b_0)\epsilon_2 + (a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0)\epsilon_1\epsilon_2, \quad (11)$$

where $\epsilon_1^2 = 0$, $\epsilon_2^2 = 0$ and $\epsilon_1\epsilon_2 \neq 0$. This representation with *only two* duals ϵ_1 and ϵ_2 can be misleading. Since these new numbers have the symbols 1, ϵ_1 and ϵ_2 as their basics entities, any number of this type must be able to be written as $a_0 + a_1\,\epsilon_1 + a_2\,\epsilon_2$, with $a_i \in \mathbb{R}$ and $i = 0, 1, 2$. Writing $\epsilon_1\epsilon_2$ in the form

$$\epsilon_1\epsilon_2 = a + b\epsilon_1 + c\epsilon_2, \quad (12)$$

and then multiplying both sides by $\epsilon_1\epsilon_2$, we obtain $0 = a\epsilon_1\epsilon_2$, which implies $a = 0$ (otherwise, multiply the latter equality by a^{-1} to get $0 = \epsilon_1\epsilon_2$, which is a contradiction). Thus (12) becomes $\epsilon_1\epsilon_2 = b\epsilon_1 + c\epsilon_2$ and, multiplying by ϵ_2, it follows that $b = 0$. It is similarly proved that $c = 0$ and so $\epsilon_1\epsilon_2 = 0$, which is a contradiction.

An alternative way to avoid ambiguities is to introduce a third symbol ϵ_3 and define the multiplication as follows

	1	ϵ_1	ϵ_2	ϵ_3
1	1	ϵ_1	ϵ_2	ϵ_3
ϵ_1	ϵ_1	0	ϵ_3	0
ϵ_2	ϵ_2	ϵ_3	0	0
ϵ_3	ϵ_3	0	0	0

, (13)

then (11) is recovered with $\epsilon_3 \equiv \epsilon_1\epsilon_2$. Nevertheless, the multiplication table (8) provides a well-defined operation, which is *closed*. Therefore, we do not need an extra symbol, say ϵ_3, to define the product $\epsilon_1\epsilon_2$. Furthermore, the use of table (8) also allows for the computation of derivatives of several variables–see Eq. (20)–without the needed of introducing more symbols, unlike the procedure presented in [21].

5 Fast Calculation of Directional Derivatives and Products of Vectors with the Hessian

Consider the Taylor series expansion of an analytic function $f : \mathbb{R}^m \to \mathbb{R}$

$$f(\mathbf{x} + \mathbf{h}) = f(\mathbf{x}) + \langle \nabla f(\mathbf{x}) | \mathbf{h} \rangle + \frac{1}{2}\langle \mathbf{h} | \mathbf{H} f(\mathbf{x}) | \mathbf{h} \rangle + O(h^3), \quad (14)$$

where $h = |\mathbf{h}|$ is the distance from the origin to the point $\mathbf{h}$. Making the substitution $\mathbf{h} \to \mathbf{v}\,\epsilon_1$, and using the multiplication table given in Eq. (8), we have

$$f(\mathbf{x} + \mathbf{v}\,\epsilon_1) = f(\mathbf{x}) + \langle \nabla f(\mathbf{x}) | \mathbf{v} \rangle\, \epsilon_1 + \langle \mathbf{v} | \mathbf{H} f(\mathbf{x}) | \mathbf{v} \rangle\, \epsilon_2 \quad (15)$$

which is a dual number of the same form given in Eq. (7). Therefore, with a single evaluation of the dual function f in the dual point $\mathbf{x} + \mathbf{v}\,\epsilon_1$, we can compute the directional derivative $\langle \nabla f(\mathbf{x}) | \mathbf{v} \rangle$ and $\langle \mathbf{v} | \mathbf{H} f(\mathbf{x}) | \mathbf{v} \rangle$ by taking respectively, the ϵ_1 and ϵ_2 components of Eq. (15). Consequently, defining the operators Du_1 and Du_2 which extract the ϵ_1 and ϵ_2 components

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

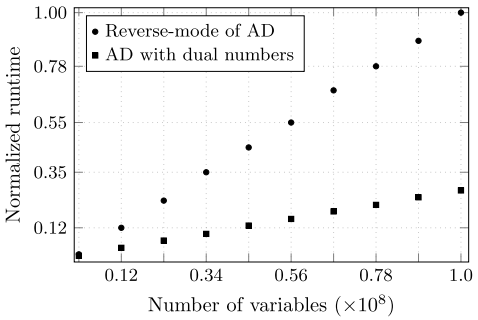


Fig. 1 Runtime for computing $\langle \nabla f(\mathbf{x})|\mathbf{v}\rangle$ as a function of the number of variables

of a dual number (in Fortran, for the type `dual2` previously defined, the ϵ_1 part of f is obtained by `f%f1` and the ϵ_2 part of f by `f%f2`), we have

$$\langle \nabla f(\mathbf{x})|\mathbf{v}\rangle = \mathrm{Du}_1[f(\mathbf{x}+\mathbf{v}\epsilon_1)] \tag{16}$$

$$\langle \mathbf{v}|\mathbf{H}f(\mathbf{x})|\mathbf{v}\rangle = \mathrm{Du}_2[f(\mathbf{x}+\mathbf{v}\epsilon_1)]. \tag{17}$$

Since $|\mathbf{v}\rangle$ is an arbitrary vector, the k-th component of $\nabla f(\mathbf{x})$ can be computed as

$$|\nabla f(\mathbf{x})\rangle_k = \langle \nabla f(\mathbf{x})|\mathbf{e}_k\rangle, \tag{18}$$

being $|\mathbf{e}_k\rangle$ the k-th vector of the standard basis of $\mathbb{R}^p$ (in this case $p = m$). Moreover, considering $\langle \mathbf{v}+\mathbf{e}_k|\mathbf{H}f(\mathbf{x})|\mathbf{v}+\mathbf{e}_k\rangle$, we can directly prove that:

$$|\mathbf{H}f(\mathbf{x})|\mathbf{v}\rangle_k = \frac{1}{2}\left[\langle \mathbf{v}+\mathbf{e}_k|\mathbf{H}f(\mathbf{x})|\mathbf{v}+\mathbf{e}_k\rangle - \langle \mathbf{v}|\mathbf{H}f(\mathbf{x})|\mathbf{v}\rangle - \langle \mathbf{e}_k|\mathbf{H}f(\mathbf{x})|\mathbf{e}_k\rangle\right]. \tag{19}$$

Incidentally, the second order derivatives are given by

$$\frac{\partial^2 f(x_1, x_2, \ldots, x_m)}{\partial x_i \partial x_j} = \langle \mathbf{e}_i|\mathbf{H}f(\mathbf{x})|\mathbf{e}_j\rangle. \tag{20}$$

Since the Hessian matrix is symmetric, from Eq. (20) we can see that $m(m+1)/2$ evaluations of the dual vectorial function are required for the Hessian matrix calculation. Similarly, from Eq. (18), m evaluations are required for computing the gradient. Nevertheless, we only need one function evaluation for computing Eq. (15). This will result in an efficient way of computing Eqs. (16), (17)—A Fortran module (`diff_mod.f90`) for computing these equations is available at [28]. For instance, Fig. 1 illustrates the normalized runtime when computing $\langle \nabla f(\mathbf{x})|\mathbf{v}\rangle$ for

$$f(\mathbf{x}) = \sum_{k=1}^{m} \frac{1}{1+\exp(-x_k)},$$

with $\mathbf{x} = (\cos 1, \cos 2, \ldots, \cos m)$, $\mathbf{v} = (\sin x_1, \sin x_2 \ldots, \sin x_m)$, and taking 10 equally spaced values of m in $[1 \times 10^6, 1 \times 10^8]$. The implementation of the reverse mode of AD used was taken from [10]. In an Intel processor (i5-4460) with frequency of 3.4 GHz and 8 GB of memory, the normalization constant was of 10.41 seconds. We have used the GNU Fortran compiler (gfortran –gcc version 7.3.0-27ubuntu1 18.04) running on GNU/Linux (Linux Mint 19.1).

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

6 Velocity and Acceleration in a System with *m* Degrees of Freedom

The position vector of a point particle in a system with m degrees of freedom is a function of the form $\mathbf{r} : \mathbb{R}^m \to \mathbb{R}^n$ with $n = 3$. Nevertheless, we will discuss the case for arbitrary n. Explicitly, the position vector will be a function of the form

$$|\mathbf{r}(x_1, x_2, \ldots, x_m)\rangle = \sum_{k=1}^{n} r_k(x_1, x_2, \ldots, x_m)\, |\mathbf{e}_k\rangle, \tag{21}$$

where r_k is a function $r_k : \mathbb{R}^m \to \mathbb{R}$.

The velocity is given by (to avoid cumbersome notation, the dependency on $(x_1, x_2, \ldots, x_m)$ is not written)

$$|\dot{\mathbf{r}}\rangle = \sum_{k=1}^{n} \sum_{j=1}^{m} \frac{\partial r_k}{\partial x_j} \dot{x}_j |\mathbf{e}_k\rangle, \tag{22}$$

where the over dot denotes the first order derivative with respect to time.

Defining

$$|\dot{\mathbf{x}}\rangle = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_m \end{bmatrix}, \tag{23}$$

the velocity vector can be written as

$$|\dot{\mathbf{r}}\rangle = \sum_{k=1}^{n} \langle \nabla r_k | \dot{\mathbf{x}} \rangle |\mathbf{e}_k\rangle. \tag{24}$$

Differentiating Eq. (22) or Eq. (24), and from the fact that the Hessian matrix is symmetric, the acceleration vector $|\ddot{\mathbf{r}}\rangle$ is given by

$$|\ddot{\mathbf{r}}\rangle = \sum_{k=1}^{n} \left[\langle \dot{\mathbf{x}} | \mathbf{H} r_k | \dot{\mathbf{x}} \rangle + \langle \nabla r_k | \ddot{\mathbf{x}} \rangle \right] |\mathbf{e}_k\rangle. \tag{25}$$

As mentioned before, the velocity involves terms of the form $\langle \nabla r_k | \mathbf{v} \rangle$, while the acceleration involves also terms of the form $\langle \mathbf{v} | \mathbf{H} r_k | \mathbf{v} \rangle$. The sum over such terms can be computed by evaluating Eq. (21) at $\mathbf{x} + \mathbf{v}\epsilon_1$. By doing this, we have

$$\sum_{k=1}^{n} \langle \nabla r_k | \mathbf{v} \rangle |\mathbf{e}_k\rangle = \mathrm{Du}_1[|\mathbf{r}(\mathbf{x} + \mathbf{v}\epsilon_1)\rangle] \tag{26}$$

$$\sum_{k=1}^{n} \langle \mathbf{v} | \mathbf{H} r_k | \mathbf{v} \rangle |\mathbf{e}_k\rangle = \mathrm{Du}_2[|\mathbf{r}(\mathbf{x} + \mathbf{v}\epsilon_1)\rangle]. \tag{27}$$

Notice that the left-hand side of Eq. (26) can also be written in matrix form as

$$\sum_{k=1}^{n} \langle \nabla r_k | \mathbf{v} \rangle |\mathbf{e}_k\rangle = (\mathbf{J}\mathbf{r}) |\mathbf{v}\rangle, \tag{28}$$

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 1 Comparison between finite differences and dual numbers in computing $\langle \mathbf{v}|\mathbf{H}f(\mathbf{x})|\mathbf{v}\rangle$

Finite differences	Dual number approach	Error (%)
$m = 10^7, h = 10^{-10}$ 0.00000000	−18415811.3	100
$m = 10^3, h = 10^{-10}$ 5684341.88	−1836.90127	309553
$m = 10^7, h = 10^{-8}$ −27939677.2	−18415811.3	51.71
$m = 10^3, h = 10^{-8}$ −1705.30257	−1836.90127	7.164
$m = 10^7, h = 0.1$ −18414831.9	−18415811.3	0.005
$m = 10^3, h = 0.1$ −1827.05720	−1836.90127	0.535

being ($\mathbf{Jr}$) the Jacobian of Eq. (21). Furthermore, since only one evaluation—$|\mathbf{r}(\mathbf{x}+\mathbf{v}\epsilon_1)\rangle$—is needed for computing Eqs. (26) and (27), we have an efficient method for computing velocities and accelerations. The aforementioned Fortran module `diff_mod.f90` contains the subroutine `vHFvJFv` for computing these equations.

7 Applications

7.1 Fast Computation of the Product of Vectors with the Hessian

We are interested in computing $\langle \mathbf{v}|\mathbf{H}f(\mathbf{x})|\mathbf{v}\rangle$ for the inverted cosine wave function

$$f(\mathbf{x}) = -\sum_{i=1}^{m-1} \exp[-(x_i^2 + x_{i+1}^2 + 0.5\, x_i\, x_{i+1})/8] \times \cos\left(4\sqrt{x_i^2 + x_{i+1}^2 + 0.5\, x_i\, x_{i+1}}\right), \quad (29)$$

with $m = 10^7$, $\mathbf{x} = [\cos 1, \cos 2, \ldots, \cos m]$ and $\mathbf{v} = [\sin 1, \sin 2, \ldots, \sin m]$. A standard computation of this quantity involves a matrix with 10^{14} elements. However, this computation only takes about 2 seconds using Eq. (17), for the aforementioned processing characteristics, and using the optimized option -O3 for gfortran. This example is available at [28].

The comparison between the proposed derivative approach and the finite differences method is a topic of interest. Table 1 reports this comparison for different values of m (the dimension of $\mathbf{x}$) and h (the step size for finite differences). The results evidence the difficulty of selecting h and the inaccuracy level that the finite differences method may have.

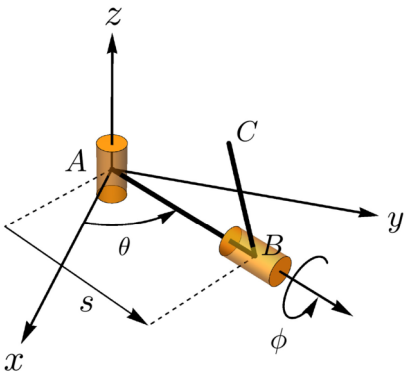
7.2 Velocity and Acceleration for the RC Robot Manipulator

We are interested in computing the velocity and acceleration of the RC (revolute and cylindrical joints) robot manipulator with three degrees of freedom, illustrated in Fig. 2, and for the values

$$\theta_0 = \pi/2,\ \phi_0 = 0,\ s_0 = 2,\ \overline{BC} = 3$$

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Fig. 2 RC manipulator



$$\dot{\theta} = 1,\ \dot{\phi} = 5,\ \dot{s} = 1$$
$$\ddot{\theta} = 1,\ \ddot{\phi} = 0,\ \ddot{s} = 2.$$

Using the homogeneous transformation method [31], the position vector of the point C is constructed with the first three elements of the 4-th column of the matrix ${}^0\mathbf{T}_3$ defined as

$$ {}^0\mathbf{T}_3 = {}^0\mathbf{T}_1\, {}^1\mathbf{T}_2\, {}^2\mathbf{T}_3, \tag{30}$$

where

$${}^0\mathbf{T}_1 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, {}^1\mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 0 & s \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, {}^2\mathbf{T}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & BC \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Once Eq. (30) is implemented in the context of the dual numbers, it is straightforward to compute the velocity and acceleration.

7.3 Dual Version of Müller's Method

As a further application, suppose that we want to calculate $u'(x_0 = -0.1)$ and $u''(x_0 = -0.1)$ for $u(x)$ defined implicitly by

$$u \sin\left(e^{-u\,x^2}\right) - \cos\left(u^2\, x^3\right) + \frac{u}{x} = 0. \tag{31}$$

The problem is not trivial since we cannot obtain a closed-form expression for $u(x)$. However we can use a numerical method to solve this equation and to obtain $u(x_0)$. Nevertheless, the problem of finding $u'(x_0)$ and $u''(x_0)$ still remains. A solution for this problem is to code a numerical method in the context of the dual numbers for solving equations. For instance, applying the Müller's method [29] in the context of dual numbers, we can obtain $u(x_0)$ as well as the derivatives $u'(x_0)$ and $u''(x_0)$. All the presented examples are available at [28].

8 Conclusions

The dual numbers yield an easy implementation of the AD forward mode. Such implementation can be used for efficiently compute directional derivatives and the multiplication of

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Jacobians and Hessians on vectors. For instance, the running time for computing the directional derivative of a function with 10^8 variables composed of a sum of sigmoid functions is of the order of seconds (see Sect. 5). Since the computation of velocities and accelerations involve the action of Jacobians and Hessians on vectors, the use of dual numbers yield efficient computation of such quantities. Moreover, the derivatives of a given quantity can be directly computed using dual numbers. This feature is exploited for computing the derivatives of a function defined implicitly by an equation through the application of the dual version of Müller's method.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Griewank, A.: On automatic differentiation. In: Iri, M., Tanabe, K. (eds.) Mathematical Programming: Recent Developments and Applications. Kluwer, Dordrecht (1998)
2. Neidinger, R.D.: Introduction to automatic differentiation and MATLAB object-oriented programming. SIAM Rev. **52**(3), 545–563 (2010)
3. Yu, W., Blair, M.: DNAD, a simple tool for automatic differentiation of Fortran codes using dual numbers. Comput. Phys. Commun. **184**, 1446–1452 (2013)
4. Griewank, A.: Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation. Frontiers in Applied Mathematics, vol. 19. SIAM, Portland (2008)
5. Phipps, E., Pawlowski, R.: In: Efficient Expression Templates for Operator Overloading-Based Automatic Differentiation, pp. 309–319. Springer, Berlin (2012)
6. Pham-Quang, P., Delinchant, B.: In: Java Automatic Differentiation Tool Using Virtual Operator Overloading, pp. 241–250. Springer, Berlin (2012)
7. Werner, J., Hillenbrand, M., Hoffmann, A., Sinzinger, S.: Automatic differentiation in the optimization of imaging optical systems. Schedae Inform. **21**, 169–175 (2012)
8. Hascoet, L., Pascual, V.: The tapenade automatic differentiation tool: Principles, model, and specification. ACM Trans. Math. Softw. **39**(3), 20, 43 pp. (2013)
9. Walter, S.F., Lehmann, L.: Algorithmic differentiation in python with algopy. J. Comput. Sci. **4**(5), 334–344 (2013)
10. Hogan, R.J.: Fast reverse-mode automatic differentiation using expression templates in C++. ACM Trans. Math. Softw. **40**(4), 26, 16 pp. (2014)
11. Moré, J.J., Wild, S.M.: Do you trust derivatives or differences? J. Comput. Phys. **273**, 268–277 (2014)
12. Li, X., Zhang, D.: A backward automatic differentiation framework for reservoir simulation. Comput. Geosci. **18**(6), 1009–1022 (2014)
13. Khan, K.A., Barton, P.I.: A vector forward mode of automatic differentiation for generalized derivative evaluation. Optim. Methods Softw. **30**(6), 1185–1212 (2015)
14. Naumann, U., Lotz, J., Leppkes, K., Towara, M.: Algorithmic differentiation of numerical methods: Tangent and adjoint solvers for parameterized systems of nonlinear equations. ACM Trans. Math. Softw. **41**(4), 26, 21 pp. (2015)
15. Zubov, V.I.: Application of fast automatic differentiation for solving the inverse coefficient problem for the heat equation. Comput. Math. Math. Phys. **56**(10), 1743–1757 (2016)
16. Weinstein, M.J., Rao, A.V.: A source transformation via operator overloading method for the automatic differentiation of mathematical functions in Matlab. ACM Trans. Math. Softw. **42**(2), 11, 44 pp. (2016)
17. Sluşanschi, E.I., Dumitrel, V.: Adijac – automatic differentiation of Java classfiles. ACM Trans. Math. Softw. **43**(2), 9, 33 pp. (2016)
18. Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in machine learning: A survey. J. Mach. Learn. Res. **18**(153), 1–43 (2018)
19. Bücker, M., Corliss, G., Hovland, P., Naumann, U. Norris, B. (eds.): Automatic Differentiation: Applications, Theory, and Implementations Springer, New York (2006)
20. Fike, J., Alonso, J.: The development of hyper-dual numbers for exact second-derivative calculations. AIAA Paper 2011-886. https://doi.org/10.2514/6.2011-886
21. Tanaka, M., Sasagawa, T., Omote, R., Fujikawa, M., Balzani, D., Schröder, J.: A highly accurate 1st- and 2nd-order differentiation scheme for hyperelastic material models based on hyper-dual numbers. Comput. Methods Appl. Mech. Eng. **283**, 22–45 (2015)

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

22. Clifford, W.: Preliminary sketch of biquaternions. Proc. Lond. Math. Soc. **1**(1–4), 381–395 (1873)
23. Pennestrì, E., Stefanelli, R.: Linear algebra and numerical algorithms using dual numbers. Multibody Syst. Dyn. **18**, 323–344 (2007)
24. Frydryszak, A.M.: Dual numbers and supersymmetric mechanics. Czechoslov. J. Phys. **55**(11), 1409–1414 (2005)
25. Gremse, F., Höfter, A., Razik, L., Kiessling, F., Naumann, U.: Gpu-accelerated adjoint algorithmic differentiation. Comput. Phys. Commun. **200**, 300–311 (2016)
26. Møller, M.: Exact calculation of the product of the Hessian matrix of feed-forward network error functions and a vector in 0(n) time. DAIMI Report Series 22 (432)
27. Pearlmutter, B.A.: Fast exact multiplication by the Hessian. Neural Comput. **6**(1), 147–160 (1994)
28. Peñuñuri, F.: Additional material to article: Dual numbers and automatic differentiation to efficiently compute velocities and accelerations. https://doi.org/10.17632/zwxpjbjz44.1
29. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in Fortran 77. The Art of Scientific Computing, vol. 1, 2nd edn. Cambridge University Press, Cambridge (1995)
30. Martins, J.R.R.A., Sturdza, P., Alonso, J.J.: The complex-step derivative approximation. ACM Trans. Math. Softw. **29**(3), 245–262 (2003)
31. Stejskal, V., Valasek, M.: Kinematics and Dynamics of Machinery (Mechanical Engineering). CRC Press, Boca Raton (1996)

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH ("Springer Nature").

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users ("Users"), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use ("Terms"). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com