Efficient Higher-Order Reverse Mode Automatic Differentiation

Matthew R. Supernaw

National Oceanic Atmospheric Administration
National Marine Fisheries Service
matthew.supernaw@noaa.gov

Z. Teresa A'mar, PhD

National Oceanic Atmospheric Administration
Office Of Science And Technology
teresa.amar@noaa.gov

Abstract

The method of computing first-order partial derivatives in reverse mode has been well documented in literature. In this article we present an extension to the reverse mode automatic differentiation algorithm which allows us to compute higher-order mixed partial derivatives up to third-order. In addition, we introduce an implementation written in the C++ programing language. These higher-order mixed derivatives are particularly useful for optimization problems where gradients and/or Hessian matrices are needed. To demonstrate the usefulness of these algorithms, we'll show how the second and third-order derivatives are used in various scientific programming routines.

I. Introduction

Automatic, or algorithmic, differentiation (AD) is a chain rule-based technique for evaluating derivatives of functions given as computer programs for their elimination [3].This technique exploits the fact that every computer program, no matter how complicated, executes a sequence of elementary arithmetic operations (addition, subtraction, multiplication, division, etc.) and elementary functions (exp, log, sin, cos, etc.) [**?**]. In this section, we provide a very basic description of automatic differentiation. Compared to finite-difference techniques, AD has proven to be more efficient for accurately computing partial derivatives.

At the heart of AD is the chain rule. Recall from calculus, the chain rule is a method for computing the derivative of two or more functions:

$$y = f(g(x)) \tag{1}$$

with *y=f(g(x))* as the outer function and *g(x)* as the inner function.

Chain Rule:

$$f(g(x))' = g'(x)\, f'(g(x)) \tag{2}$$

or

$$\frac{df}{dx} = \frac{dg}{dx}\frac{df}{dg} \tag{3}$$

with $\frac{df}{dg}$ as the outer derivative and $\frac{dg}{dx}$ as the inner derivative.

```
Why is it useful?
Who uses AD?
```

II. Forward Mode

Forward mode (or the tangent linear method) AD traverses the chain rule from inside to outside. That is, from (3), $\frac{df}{dg}$ is computed before $\frac{dg}{dx}$. To demonstrate how the forward accumulation of the chain rule works, consider the expressions $f(x_1, x_2) = ln(x_1 x_2)$. Here $g(x_1, x_2) = x_1 x_2$ and $f(x_1, x_2) = ln(g(x_1, x_2))$. So, to find the gradient $\nabla f(x_1, x_2)$ we must evaluate $f(x_1, x_2)$ and record these operations to a "Tape" so we can evaluate the partial derivatives later.

Evaluation	**Tape**
$x_1 = 3.1459$	$x_1' = 0.0$
$x_2 = 2.0$	$x_2' = 0.0$
$g(x_1, x_2) = x_1 x_2$	$g(x_1, x_2)' = x_1' x_2 + x_1 x_2'$
$f(g(x_1, x_2)) = ln(g(x_1, x_2))$	$f(g(x_1, x_2))' = \frac{g(x_1, x_2)'}{g(x_1, x_2)}$

Now that we have a record of $f(x_1, x_2)$ on the "Tape", we can apply the forward mode accumulation of the chain rule and compute the gradient:

Let $\nabla f(x_1, x_2) = [x_1', x_2']$

Compute: x_1'
Tape
$x_1' = 1.0$***(seed)***
$x_2' = 0.0$

$g(x_1, x_2)' = x_1' x_2 + x_1 x_2' = 1.0 * 2.0 + 3.1459 * 0 = 2.0$

$f(g(x_1, x_2))' = \frac{g(x_1, x_2)'}{g(x_1, x_2)} = \frac{2.0}{2.0 * 3.1459} = 0.317874$

Compute: x_2'
Tape
$x_1' = 0.0$
$x_2' = 1.0$***(seed)***

$g(x_1, x_2)' = x_1' x_2 + x_1 x_2' = 0.0 * 2.0 + 3.1459 * 1.0 = 3.1459$

$f(g(x_1, x_2))' = \frac{g(x_1, x_2)'}{g(x_1, x_2)} = \frac{3.1459}{2.0 * 3.1459} = 0.5$

Gives:

$\nabla f(x_1, x_2) = [\frac{df}{dx_1}, \frac{df}{dx_2}] = [0.317874, 0.5]$

The above "Tape" evaluation can be generalized algorithmically by:

Algorithm 1 Forward Mode Accumulation

$\hat{w} = [x'_1, x'_2, x'_3 ... x'_m]$
for $i = 1$ to m **do**
  $\hat{w}[i] = 1.0$
  for $j = 1$ to n **do**
    $Tape[j] \rightarrow Evaluate$
  end for
  $\nabla f[i] = Tape[n] \rightarrow Value$
  $\hat{w}[i] = 0.0$
end for

As you can see from Algorithm 1, for a function $f(x_1, x_2 ..., x_m)$ the "Tape" must be evaluated m times to compute the gradient. For highly parameterized functions, it may be desirable to compute the gradient using reverse mode accumulation.

III. Reverse Mode

.1 Reverse Mode

Reverse mode (or the adjoint method) AD traverses the chain rule from outside to inside. That is, from (3), $\frac{dg}{dx}$ is computed before $\frac{df}{dg}$. It works by accumulating a series of adjoints in the opposite direction of the forward mode method. This can be generalized by the following algorithm:

Algorithm 2 Reverse Mode Accumulation [4]

Input: *Tape*
$\bar{w} = [\bar{x_1}, \bar{x_2}, \bar{x_3} ... \bar{x_{m-1}}] = 0$
$\bar{w}[m] = 1$
for $i = m$ to 1 **do**
  $\bar{w_i} = \bar{w}[i]$
  $\bar{w}[i] = 0$
  for $j = 1$ to i **do**
    $\bar{w}[j] += \frac{\partial \phi_i}{\partial x_i} \bar{w_i}$
  end for
end for
Output: $\nabla f = \bar{w} = [\bar{x_1}, \bar{x_2}, \bar{x_3} ... \bar{x_m}]$

To demonstrate the reverse mode method, lets revisit our example from the forward mode description in the previous section with slight modifications to the "Tape" in order to represent the partial derivative entries, also know as adjoints.

	Evaluation	**Tape**
↓	$x_1 = 3.1459$	$x'_1 = 0.0$
	$x_2 = 2.0$	$x'_2 = 0.0$
	$g(x_1, x_2) = x_1 x_2$	$x'_3 = g(x_1, x_2)' = [[x'_1, \frac{\partial g}{\partial x_1} = 1.0 * 2.0], [x'_2, \frac{\partial g}{\partial x_2} = 1.0 * 3.1459]]$
	$f(g(x_1, x_2)) = ln(g(x_1, x_2))$	$x'_4 = f(g(x_1, x_2))' = [g(x_1, x_2)', \frac{\partial f}{\partial x_3} = (2.0 * 3.1459)^{-1}]$

Now we have a tape that contains entries that each have a list of adjoints to use in the reverse mode calculation. By traversing the tape from the bottom up, we can easily compute the full gradient in one sweep using **Algorithm 2**:

$\nabla f = \bar{w} = [\bar{x_1}, \bar{x_2}, \bar{x_3}, \bar{x_4}]$

Tape	**Trace**
m = 2 // last elementary operation $\bar{w}[m] = 1$**(seed)**	
i = m = 2 $\bar{w}_i = \bar{w}[i]$ $\bar{w}[i] = 0$ $\bar{w}[3] += \bar{w}_i \frac{\partial f}{\partial x_3} = 1 * (2.0 * 3.1459)^{-1} = 0.159$	$\nabla f = \bar{w} = [0, 0, 0.159, 0]$
i = m-1 = 1 $\bar{w}_i = \bar{w}[i]$ $\bar{w}[i] = 0$ $\bar{w}[1] += \bar{w}_i \frac{\partial g}{\partial x_1} = 0.159 * 2.0 = 0.317874$ $\bar{w}[2] += \bar{w}_i \frac{\partial g}{\partial x_2} = 0.159 * 3.1459 = 0.5$	$\nabla f = \bar{w} = [0.317874, 0.5, 0, 0]$
Final Result: $\nabla f = [0.317874, 0.5, 0, 0]$	

Reverse mode requires only one sweep to compute all the partials for a function $f(x_1, x_2..., x_m)$, therefore it is more efficient for functions having $m > 1$ variables.

IV. Higher-Order Reverse Mode

```
Why is it useful?
Why isn't it as simple as first-order reverse mode?
```

In nonlinear optimization problems, in particular those using some variation of the Newton method, one must evaluate either the exact or an approximation of the Hessian matrix at each iteration. The Hessian matrix is a square matrix of all the second-order partial derivatives for a function. It describes the local curvature of a function of many variables [1]. In some statistical computing routines, such as mixed effects problems using the Laplace Transformation method, third-order mixed partials are required. In either case, it is desirable to calculate these derivatives in the most efficient manner possible, as problems involving a high number of parameters can easily become arduous if using a finite differences method or even a forward mode automatic differentiation routine.
At first glance, it may seem the solution is to simply differentiate Algorithm 2, but this will neglect to account for important nonlinear interactions as Gowler, et. al. have demonstrated with their "Edge-Pushing" algorithm [2]. This observation is an important contribution to the research of higher-order reverse mode automatic differentiation and has been tremendously insightful in the development of our algorithm.

I. Nonlinear Interactions

A nonlinear interaction can be defined at the expression level as an interaction of variables in the linear portion of an expression with those that are nonlinear. For example:

$$f(x, y, z) = \frac{x * y}{e^z} \tag{4}$$

Here x and y interact with the nonlinear expression e^z, therefore x and y are said to have a nonlinear interaction with z. These nonlinear interactions must be accounted for in the reverse mode accumulation of order greater than one. To satisfy this requirement, we must update Algorithm 2 in order to accommodate these nonlinear dependencies. This means, whenever x and y are found at the statement level, z will accompany them and vice versa. Furthermore, a variable resulting from the evaluation of $f(x, y, z)$ will also be considered nonlinear as it is the result of a nonlinear expression. This means anytime the resulting dependent variable is used, it will be considered nonlinear, just as the expression e^z is in the above equation.

```
What is a nonlinear interaction?
Why is it important to track them?
```

II. Second-Order Reverse Mode

Given the observation of nonlinear interactions, we are now able to differentiate and expand on Algorithm 2 in order to account for any nonlinear interactions that may have occurred at the statement level. To do this, we introduce the idea of variable sets at each entry in the "Tape" structure. Recall a set is a list of unique entries. These sets not only hold variables that were found in the evaluated statement, but also those that have nonlinear interaction with any of the statement level variables. Before each entry in the "Tape" is evaluated, we prepare the entry by combining the variables from the expression statement with those that have nonlinear interactions with variables found at the statement level. This procedure occurs whenever a "=" operator is found. The result is a concise list of variable information required to properly accumulate the higher-order derivatives for that entry. In addition, after the entry has been evaluated during the reverse mode accumulation, it is also necessary to add the statement level variables into the next entry to be evaluated. This leads us to Algorithm 3.

Algorithm 3 Reverse Mode With Hessian Accumulation

Input: *Tape*

$\nabla f = \bar{w}[m] = 0$
$\nabla^2 f = \bar{h}[m][m] = 0$
$\bar{w}[m] = 1$

for $i = m$ to 1 **do**
    $\bar{w}_i = \bar{w}[i]$
    $\bar{w}[i] = 0$
    $\bar{h}_{ii} = \bar{h}[i][i]$
    $\bar{h}[i][i] = 0$
    n = size(variable set)

    for $j = 1$ to n **do**
        $\bar{H}[i][j] = \bar{h}[i][j]$
        $\bar{h}[i][j] = 0$
    end for

    for $j = 1$ to n **do**
        $\bar{w}[j] += \frac{\partial \phi_i}{\partial x_i} \bar{w}_i$
        for $k = 1$ to n **do**
            $\bar{h}[j][k] += \bar{H}[i][k] \frac{\partial \phi_i}{\partial x_j} + \bar{H}[i][j] \frac{\partial \phi_i}{\partial x_k} + \bar{h}_{ii} \frac{\partial \phi_i}{\partial x_k} \frac{\partial \phi_i}{\partial x_j} + \bar{w}_i \frac{\partial^2 \phi_i}{\partial x_j \partial x_k}$
        end for
    end for
    if i **then** > 1
        Push statement level variables to entry[i-1].
    end if
end for

As you can see from Algorithm 3, we obtain not only the second-order mixed partials, but the gradient as well. To properly evaluate Algorithm 3, we are required to have the computational graph for each expression statement. Having this computational graph gives us all the tools and information we need to build any nonlinear interaction list that may be required. In addition, the computational graph gives us the opportunity to omit any temporary variables that arise from function evaluations in the expression. Furthermore, it allows us to compute local derivatives for any variables that are in the expression and they are our adjoint entries. Figure 1 shows an example of what happens during a second-order statement level evaluation involving nonlinear interactions.

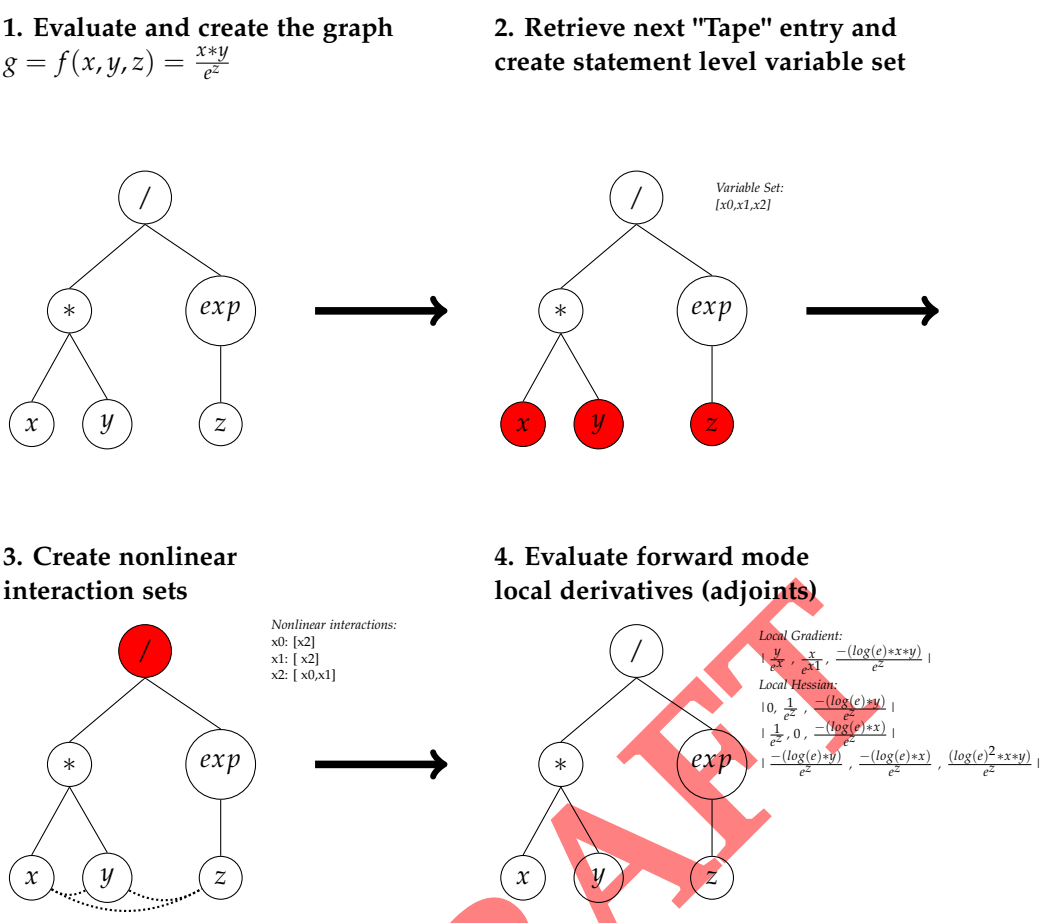

Figure 1: *The evaluation sequence at the statement level resulting in a second-order entry into the derivative structure, or "Tape". If no nonlinear interactions are found, step 3 is simply omitted.*

```
Insert computational graph example showing nonlinear interactions.
```

For a detailed trace of the evaluation and derivative accumulation for the expression in **Figure 1**, see **Appendix 1**.

III. Third-Order Reverse Mode

Continuing the observation of nonlinear interactions and differentiating Algorithm 3 leads us to Algorithm 4 for third-order reverse mode accumulation. The results from the third-order algorithm gives us the gradient, Hessian, and the third-order mixed partial derivatives of our objective function.

Algorithm 4 Third-Order Reverse Mode Accumulation

Input: *Tape*
$\nabla f = \bar{w}[m] = 0$
$\nabla^2 f = \bar{h}[m][m] = 0$
$\nabla^3 f = \bar{d}[m][m][m] = 0$

$\bar{w}[m] = 1$
for $i = m$ to 1 **do**
    $\bar{w}_i = \bar{w}[i]$
    $\bar{w}[i] = 0$
    $\bar{h}_{ii} = \bar{h}[i][i]$
    $\bar{h}[i][i] = 0$
    $\bar{d}_{iii} = \bar{d}[i][i][i]$
    $\bar{d}[i][i][i] = 0$
    n = size(variable set)

    for $j = 1$ to n **do**
        $\bar{H}[i][j] = \bar{h}[i][j]$
        $\bar{h}[i][j] = 0$
        for $k = 1$ to n **do**
            $\bar{D}[i][j][k] = \bar{d}[i][j][k]$
            $\bar{d}[i][j][k] = 0$
        end for
    end for

    for $j = 1$ to n **do**
        $\bar{w}[j] += \frac{\partial \phi_i}{\partial x_i} \bar{w}[i]$
        for $k = 1$ to n **do**
            $\bar{h}[j][k] += \bar{h}[i][k] \frac{\partial \phi_i}{\partial x_j} + \bar{h}[i][j] \frac{\partial \phi_i}{\partial x_k} + \bar{h}[i][i] \frac{\partial \phi_i}{\partial x_k} \frac{\partial \phi_i}{\partial x_j} + \bar{w}_i \frac{\partial^2 \phi_i}{\partial x_j \partial x_k}$
            for $l = 1$ to n **do**
$$\begin{aligned}\bar{d}[j][k][l] += {} & \frac{\partial \phi_i}{\partial x_j} \bar{D}[i][k][l] + \frac{\partial \phi_i}{\partial x_k} \bar{D}[i][j][l] + \frac{\partial \phi_i}{\partial x_l} \bar{D}[i][j][k] + \\ & \frac{\partial \phi_i}{\partial x_j} \frac{\partial \phi_i}{\partial x_k} \bar{D}[i][i][l] + \frac{\partial \phi_i}{\partial x_j} \frac{\partial \phi_i}{\partial x_l} \bar{D}[i][i][k] + \frac{\partial \phi_i}{\partial x_k} \frac{\partial \phi_i}{\partial x_l} \bar{D}[i][i][j] + \\ & \frac{\partial \phi_i}{\partial x_j} \frac{\partial \phi_i}{\partial x_k} \frac{\partial \phi_i}{\partial x_l} \bar{d}_{iii} + \\ & \frac{\partial \phi_i}{\partial x_j} \frac{\partial \phi_i}{\partial x_k} \bar{H}[i][l] + \frac{\partial \phi_i}{\partial x_j} \frac{\partial \phi_i}{\partial x_l} \bar{H}[i][k] + \frac{\partial \phi_i}{\partial x_k} \frac{\partial \phi_i}{\partial x_l} \bar{H}[i][j] + \\ & [\frac{\partial \phi_i}{\partial x_j} \frac{\partial^2 \phi_i}{\partial x_k \partial x_l} + \frac{\partial^2 \phi_i}{\partial x_k} \frac{\partial^2 \phi_i}{\partial x_j \partial x_l} + \frac{\partial^2 \phi_i}{\partial x_l} \frac{\partial^2 \phi_i}{\partial x_j \partial x_k}] \bar{h}_{ii} + \\ & \bar{w}_i \frac{\partial^3 \phi_i}{\partial x_j \partial x_k \partial x_l}\end{aligned}$$
            end for
        end for
    end for
    if i **then** > 1
        Push statement level variables to entry[i-1].
    end if
end for

```
Third order reverse mode description.
```

V. Implementation in C++

We have chosen to implement our framework for automatic differentiation in the C++ programming language. Being a object oriented language, C++ makes it easy for us to build the computational graphs required at each statement evaluation. C++ also has the added benefit of providing a mechanism for operator overloading, making it easy to implement algorithms just as if they were using native data types like float or double.

```
Why C++?
```

VI. Testing

VII. Benchmarks

VIII. Conclusions

```
Talk about the benefits and limitations of the algorithm.
```

IX. Appendix 1

What follows is an example trace of an objective function evaluation and the accumulation of first and second-order derivatives.

Given:
$x = 3.1459$
$y = 1.5$
$z = 2.4$

Evaluate:
$g = x * y$
$f = \frac{g}{e^z}$

Evaluation	**Tape**
$x = 3.1459$ $y = 1.5$ $z = 2.4$	$x' = 0.0$ $y' = 0.0$ $z' = 0.0$ $\begin{vmatrix} \frac{\partial^2 f}{\partial x \partial x} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y \partial y} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z \partial z} \end{vmatrix} = 0$
$g = x * y$	*Varaible Set[x, y]* *Nonlinear Interactions: None* *Tape Entry = 1* *Local Derivatives:* $\nabla g =$ $[[x', \frac{\partial g}{\partial x} = 1.0 * 1.5.0], [y', \frac{\partial g}{\partial y} = 1.0 * 3.1459]]$ $\nabla^2 g =$ $[[x'', \frac{\partial^2 g}{\partial x^2} = 0], [x'y', \frac{\partial^2 g}{\partial x \partial y} = 1.0]],$ $[[y'x', \frac{\partial^2 g}{\partial y \partial x} = 1.0], [y'y', \frac{\partial^2 g}{\partial y^2} = 0]]$
$f = \frac{g}{e^z}$	*Varaible Set[g , z]* *Nonlinear Interactions: g[z], z[g]* *Tape Entry = 2* *Local Derivatives:* $\nabla f =$ $[[\frac{\partial f}{\partial g} = 1/e^z = 0.09], [\frac{\partial f}{\partial z} = -(log(e) * g)/e^z = -0.43]]$ $\nabla^2 g =$ $[[\frac{\partial^2 f}{\partial g^2} = 0], [\frac{\partial^2 g}{\partial g \partial z} = -log(e)/e^z = -0.09]],$ $[[\frac{\partial^2 f}{\partial z \partial g} = -log(e)/e^z = -0.09], [\frac{\partial^2 f}{\partial z^2} = (log(e)^2 * g)/e^z = 0.43]]$

Reverse Accumulation

$$\nabla f = w = |\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}, \frac{\partial f}{\partial g}, \frac{\partial f}{\partial f}| = 0$$

$$\nabla^2 f = h = \begin{vmatrix} \frac{\partial^2 f}{\partial x \partial x} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} & \frac{\partial^2 f}{\partial x \partial g} & \frac{\partial^2 f}{\partial x \partial f} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y \partial y} & \frac{\partial^2 f}{\partial y \partial z} & \frac{\partial^2 f}{\partial y \partial g} & \frac{\partial^2 f}{\partial y \partial f} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z \partial z} & \frac{\partial^2 f}{\partial z \partial g} & \frac{\partial^2 f}{\partial z \partial f} \\ \frac{\partial^2 f}{\partial g \partial x} & \frac{\partial^2 f}{\partial g \partial y} & \frac{\partial^2 f}{\partial g \partial z} & \frac{\partial^2 f}{\partial g \partial g} & \frac{\partial^2 f}{\partial g \partial f} \\ \frac{\partial^2 f}{\partial f \partial x} & \frac{\partial^2 f}{\partial f \partial y} & \frac{\partial^2 f}{\partial f \partial z} & \frac{\partial^2 f}{\partial f \partial g} & \frac{\partial^2 f}{\partial f \partial f} \end{vmatrix} = 0$$

Tape

m = 2 // last elementary operation

$\bar{w}[m] = 1$**(seed)**

---

i = m = 2

Variables$[g, z]$

$w = \bar{w}[f]$

$\bar{w}[f] = 0$

$\bar{H}[f][f] = \bar{h}[f][f]$

$\bar{h}[f][f] = 0$

$\bar{H}[f][g] = \bar{h}[f][g]$

$\bar{H}[f][g] = 0$

$\bar{H}[f][z] = \bar{h}[f][z]$

$\bar{H}[f][z] = 0$

$\bar{w}[g] += w\frac{\partial f}{\partial g} = \bar{w}[f] * 0.09 = 0.09$

$\bar{h}[g][g] += \bar{H}[f][g]\frac{\partial f}{\partial g} + \bar{H}[f][g]\frac{\partial f}{\partial g} + \bar{H}[f][f]\frac{\partial f}{\partial g}\frac{\partial f}{\partial g} + w\frac{\partial^2 f}{\partial g \partial g}$

$\bar{h}[g][g] += 0 * 0.09 + 0 * 0.09 + 0 * 0.09 * 0.09 + 1.0 * 0 = 0.0$

$\bar{h}[g][z] += \bar{H}[f][z]\frac{\partial f}{\partial g} + \bar{H}[f][g]\frac{\partial f}{\partial z} + \bar{H}[f][f]\frac{\partial f}{\partial z}\frac{\partial f}{\partial g} + w\frac{\partial^2 f}{\partial g \partial z}$

$\bar{h}[g][z] += 0 * 0.09 + 0 * -0.43 + 0 * 0.09 * -0.43 + 1.0 * -0.09 = -0.09$

$\bar{w}[z] += w\frac{\partial f}{\partial z} = \bar{w}[f] * -0.43 = -0.43$

$\bar{h}[z][g] += \bar{H}[f][g]\frac{\partial f}{\partial z} + \bar{H}[f][z]\frac{\partial f}{\partial g} + \bar{H}[f][f]\frac{\partial f}{\partial g}\frac{\partial f}{\partial z} + w\frac{\partial^2 f}{\partial z \partial g}$

$\bar{h}[z][g] += 0 * -0.43 + 0 * 0.09 + 0 * -0.43 * 0.09 + 1.0 * -0.09 = -0.09$

$\bar{h}[z][z] += \bar{H}[f][z]\frac{\partial f}{\partial z} + \bar{H}[f][z]\frac{\partial f}{\partial z} + \bar{H}[f][f]\frac{\partial f}{\partial z}\frac{\partial f}{\partial z} + w\frac{\partial^2 f}{\partial z \partial z}$

$\bar{h}[z][z] += 0 * -0.43 + 0 * -0.43 + 0 * -0.43 * -0.43 + 1.0 * 0.43 = 0.43$

Insert g and z to the i -1 entry

Trace

$\nabla f = \bar{w} = [0, 0, -0.43, 0.09, 0]$

$$\nabla^2 f = h = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.43 & -0.09 & 0 \\ 0 & 0 & -0.09 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

i = m-1 = 1

Variables$[x, y|z]$ // g is omitted since it is the dependent variable for this entry

$w = \bar{w}[g] = .09$

$\bar{w}[g] = 0$

$\bar{H}[g][g] = \bar{h}[g][g]$

$\bar{h}[g][g] = 0$

$\bar{H}[g][x] = \bar{h}[g][x]$

$\bar{H}[g][x] = 0$

$\bar{H}[g][y] = \bar{h}[g][y]$

$\bar{H}[g][y] = 0$

$\bar{H}[g][z] = \bar{h}[g][z]$

$\bar{H}[g][z] = 0$

$$\nabla f = \bar{w} = [0.135, 0.28, -0.43, 0.09, 0]$$

$$\nabla^2 f = h = \begin{vmatrix} 0 & .09 & -0.135 & 0 & 0 \\ 0.09 & .09 & -0.28 & 0 & 0 \\ -0.135 & -0.28 & 0.43 & -0.09 & 0 \\ 0 & 0 & -0.09 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

$\bar{w}[x] += w\frac{\partial f}{\partial x} = \bar{w}[g] * 1.5 = 0.135$

$\bar{h}[x][x] += \bar{H}[g][x]\frac{\partial g}{\partial x} + \bar{H}[g][x]\frac{\partial g}{\partial x} + \bar{H}[g][g]\frac{\partial f}{\partial x}\frac{\partial f}{\partial x} + w\frac{\partial^2 g}{\partial x \partial x}$

$\bar{h}[x][x] += 0 * 1.5 + 0 * 1.5 + 0 * 1.5 * 1.5 + 0.09 * 0 = 0.0$

$\bar{h}[x][y] += \bar{H}[g][y]\frac{\partial g}{\partial x} + \bar{H}[g][x]\frac{\partial g}{\partial y} + \bar{H}[g][g]\frac{\partial f}{\partial y}\frac{\partial f}{\partial x} + w\frac{\partial^2 g}{\partial x \partial y}$

$\bar{h}[x][y] += 0 * 1.5 + 0 * 3.1459 + 0 * 1.5 * 3.1459 + 0.09 * 1 = .09$

$\bar{h}[x][z] += \bar{H}[g][z]\frac{\partial g}{\partial x} + \bar{H}[g][x]\frac{\partial g}{\partial z} + \bar{H}[g][g]\frac{\partial f}{\partial z}\frac{\partial f}{\partial x} + w\frac{\partial^2 g}{\partial x \partial z}$

$\bar{h}[x][z] += (-0.09 * 1.5) + 0 * 0 + 0 * 1.5 * 0 + 1.0 * 0 = -0.135$

$\bar{w}[y] += w\frac{\partial f}{\partial y} = \bar{w}[g] * 3.1459 = 0.28$

$\bar{h}[y][x] += \bar{H}[g][x]\frac{\partial g}{\partial y} + \bar{H}[g][y]\frac{\partial g}{\partial x} + \bar{H}[g][g]\frac{\partial f}{\partial x}\frac{\partial f}{\partial y} + w\frac{\partial^2 g}{\partial y \partial x}$

$\bar{h}[y][x] += 0 * 3.1459 + 0 * 1.5 + 0 * 3.1459 * 1.5 + 0.09 * 1 = 0.09$

$\bar{h}[y][y] += \bar{H}[g][y]\frac{\partial g}{\partial y} + \bar{H}[g][y]\frac{\partial g}{\partial y} + \bar{H}[g][g]\frac{\partial f}{\partial y}\frac{\partial f}{\partial y} + w\frac{\partial^2 g}{\partial y \partial y}$

$\bar{h}[y][y] += 0 * 3.1459 + 0 * 3.1459 + 0 * 3.1459 * 3.1459 + 1.0 * 0 = .09$

$\bar{h}[y][z] += \bar{H}[g][z]\frac{\partial g}{\partial y} + \bar{H}[g][y]\frac{\partial g}{\partial z} + \bar{H}[g][g]\frac{\partial f}{\partial z}\frac{\partial f}{\partial y} + w\frac{\partial^2 g}{\partial y \partial z}$

$\bar{h}[y][z] += (-0.09 * 3.1459) + 0 * 0 + 0 * 3.1459 * 0 + 1.0 * 0 = -0.28$

$\bar{w}[z] += w\frac{\partial f}{\partial z} = \bar{w}[g] * 0 = -0.43$

$\bar{h}[z][x] += \bar{H}[g][x]\frac{\partial g}{\partial z} + \bar{H}[g][z]\frac{\partial g}{\partial x} + \bar{H}[g][g]\frac{\partial f}{\partial x}\frac{\partial f}{\partial z} + w\frac{\partial^2 g}{\partial z \partial x}$

$\bar{h}[z][x] += 0 * 0 + (-0.09 * 1.5) + 0 * 0 * 1.5 + 1.0 * 0 = -0.135$

$\bar{h}[z][y] += \bar{H}[g][y]\frac{\partial g}{\partial z} + \bar{H}[g][z]\frac{\partial g}{\partial y} + \bar{H}[g][g]\frac{\partial f}{\partial y}\frac{\partial f}{\partial z} + w\frac{\partial^2 g}{\partial z \partial y}$

$\bar{h}[z][y] += 0 * 0 + (-0.09 * 3.1459) + 0 * 0 * 3.1459 + 1.0 * 0 = -0.28$

$\bar{h}[z][z] += \bar{H}[g][z]\frac{\partial g}{\partial z} + \bar{H}[g][z]\frac{\partial g}{\partial z} + \bar{H}[g][g]\frac{\partial f}{\partial z}\frac{\partial f}{\partial z} + w\frac{\partial^2 g}{\partial z \partial z}$

$\bar{h}[z][z] += 0 * 0 + 0 * 0 + 0 * 0 * 0 + 1.0 * 0 = 0.43$

Resulting Derivatives:

$$\nabla f = \bar{w} = \left| \begin{array}{ccccc} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} & \frac{\partial f}{\partial g} & \frac{\partial f}{\partial f} \end{array} \right| = \left| \begin{array}{ccccc} 0.135 & 0.28 & -0.43 & 0.09 & 0 \end{array} \right|$$

$$\nabla^2 f = h = \left| \begin{array}{ccccc} \frac{\partial^2 f}{\partial x \partial x} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} & \frac{\partial^2 f}{\partial x \partial g} & \frac{\partial^2 f}{\partial x \partial f} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y \partial y} & \frac{\partial^2 f}{\partial y \partial z} & \frac{\partial^2 f}{\partial y \partial g} & \frac{\partial^2 f}{\partial y \partial f} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z \partial z} & \frac{\partial^2 f}{\partial z \partial g} & \frac{\partial^2 f}{\partial z \partial f} \\ \frac{\partial^2 f}{\partial g \partial x} & \frac{\partial^2 f}{\partial g \partial y} & \frac{\partial^2 f}{\partial g \partial z} & \frac{\partial^2 f}{\partial g \partial g} & \frac{\partial^2 f}{\partial g \partial f} \\ \frac{\partial^2 f}{\partial f \partial x} & \frac{\partial^2 f}{\partial f \partial y} & \frac{\partial^2 f}{\partial f \partial z} & \frac{\partial^2 f}{\partial f \partial g} & \frac{\partial^2 f}{\partial f \partial f} \end{array} \right| = \left| \begin{array}{ccccc} 0 & 0.09 & -0.135 & 0 & 0 \\ 0.09 & 0.09 & -0.28 & 0 & 0 \\ -0.135 & -0.28 & 0.43 & -0.09 & 0 \\ 0 & 0 & -0.09 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right|$$

References

[1] Wikipedia contributors *Hessian Matrix*. Wikipedia, The Free Encyclopedia, 2016.

[2] Robert Mansel Gower, Margarida P. Mello *Hessian Matrices via Automatic Differentiation*. Institute of Mathematics, Statistics and Scientific Computing,State University of Campinas, September 29, 2010.

[3] Andreas Griewank, Andrea Walther [*Introduction to Automatic Differentiation*]. PAMM Âů Proc. Appl. Math. Mech. 2, 45?49 (2003).

[4] Andreas Griewank [*On Automatic Differentiation*]. Center for Research on Parallel Computation, 1989.