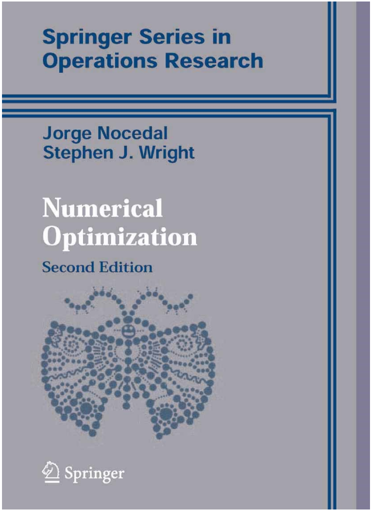
Numerical and Automatic Differentiation

Moritz Wagner

February 09, 2024

Reference


Springer Series in
Operations Research
Jorge Nocedal
Stephen J. Wright
Numerical
Optimization
Second Edition
Springer

Content

- Numerical Differentiation
  - One-sided Differencing
  - Central Differencing
  - Complex Step Method
  - Jacobians and Hessians
- Automatic Differentiation
  - Forward Mode
  - Reverse Mode
- Practical Things
  - Implementation Examples
  - Empirical Comparisons

One-Sided Differencing (1)

Let $f : \mathbb{R}^n \to \mathbb{R}$ differentiable.

$$\frac{\partial}{\partial x_i} f(x) = \lim_{h \to 0} \frac{f(x + he_i) - f(x)}{h}$$

One-Sided Differencing (1)

Let $f : \mathbb{R}^n \to \mathbb{R}$ differentiable.

$$\frac{\partial}{\partial x_i} f(x) = \lim_{h \to 0} \frac{f(x + he_i) - f(x)}{h}$$

$$\frac{\partial}{\partial x_i} f(x) \approx \frac{f(x + \varepsilon e_i) - f(x)}{\varepsilon}$$

One-Sided Differencing (2)

Let $f : \mathbb{R}^n \to \mathbb{R}$ twice differentiable. By Taylor's Theorem

$$f(x+p) = f(x) + \nabla f(x)^\top p + \frac{1}{2} p^\top \nabla^2 f(x+tp)p, \quad t \in (0,1)$$

Central Differencing (1)

Let $f : \mathbb{R}^n \to \mathbb{R}$ differentiable.

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x + \varepsilon e_i) - f(x - \varepsilon e_i)}{2\varepsilon}$$

Central Differencing (2)

Let $f : \mathbb{R}^n \to \mathbb{R}$ twice differentiable. By Taylor's Theorem:

$$f(x+p) = f(x) + \nabla f(x)^\top p + \frac{1}{2} p^\top \nabla f(x) p + O(\|p\|^3)$$

Central Differencing (2)

Let $f : \mathbb{R}^n \to \mathbb{R}$ twice differentiable. By Taylor's Theorem:

$$f(x+p) = f(x) + \nabla f(x)^\top p + \frac{1}{2} p^\top \nabla f(x) p + O(\|p\|^3)$$

Set $p = \varepsilon e_i$ and $p = -\varepsilon e_i$.

$$f(x+\varepsilon e_i) = f(x) + \varepsilon \frac{\partial f}{\partial x_i}(x) + \frac{1}{2}\varepsilon^2 \frac{\partial^2 f}{\partial x_i^2}(x) + O(\varepsilon^3)$$

$$f(x-\varepsilon e_i) = f(x) - \varepsilon \frac{\partial f}{\partial x_i}(x) + \frac{1}{2}\varepsilon^2 \frac{\partial^2 f}{\partial x_i^2}(x) + O(\varepsilon^3).$$

Central Differencing (2)

Let $f : \mathbb{R}^n \to \mathbb{R}$ twice differentiable. By Taylor's Theorem:

$$f(x+p) = f(x) + \nabla f(x)^\top p + \frac{1}{2} p^\top \nabla f(x) p + O(\|p\|^3)$$

Set $p = \varepsilon e_i$ and $p = -\varepsilon e_i$.

$$f(x+\varepsilon e_i) = f(x) + \varepsilon \frac{\partial f}{\partial x_i}(x) + \frac{1}{2}\varepsilon^2 \frac{\partial^2 f}{\partial x_i^2}(x) + O(\varepsilon^3)$$

$$f(x-\varepsilon e_i) = f(x) - \varepsilon \frac{\partial f}{\partial x_i}(x) + \frac{1}{2}\varepsilon^2 \frac{\partial^2 f}{\partial x_i^2}(x) + O(\varepsilon^3).$$

Subtract the second from the first equation and divide by 2ε.

$$\frac{\partial f}{\partial x_i}(x) = \frac{f(x+\varepsilon e_i) - f(x-\varepsilon e_i)}{2\varepsilon} + O(\varepsilon^2).$$

Complex Step Method (1)

Reminder:

Complex Step Method (1)

Reminder:

- Complex numbers are of the form $a + ib$, $a, b \in \mathbb{R}$, $i^2 = -1$.

Complex Step Method (1)

Reminder:

- Complex numbers are of the form $a + ib$, $a, b \in \mathbb{R}$, $i^2 = -1$.
- The set of complex numbers is denoted by $\mathbb{C}$.

Complex Step Method (1)

Reminder:

- Complex numbers are of the form $a + ib$, $a, b \in \mathbb{R}$, $i^2 = -1$.
- The set of complex numbers is denoted by $\mathbb{C}$.
- A complex function $f : \mathbb{C} \supseteq U \to \mathbb{C}$ is complex differentiable at $z_0 \in U$ if the following limit exists

$$f'(z_0) = \lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0}.$$

Complex Step Method (1)

Reminder:

- Complex numbers are of the form $a + ib$, $a, b \in \mathbb{R}$, $i^2 = -1$.
- The set of complex numbers is denoted by $\mathbb{C}$.
- A complex function $f : \mathbb{C} \supseteq U \to \mathbb{C}$ is complex differentiable at $z_0 \in U$ if the following limit exists

$$f'(z_0) = \lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0}.$$

- $f : \mathbb{C} \supseteq U \to \mathbb{C}$ is called complex differentiable (or holomorphic) if it is differentiable at every $z_0 \in U$.

Complex Step Method (1)

Reminder:

- Complex numbers are of the form $a + ib$, $a, b \in \mathbb{R}$, $i^2 = -1$.
- The set of complex numbers is denoted by $\mathbb{C}$.
- A complex function $f : \mathbb{C} \supseteq U \to \mathbb{C}$ is complex differentiable at $z_0 \in U$ if the following limit exists

$$f'(z_0) = \lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0}.$$

- $f : \mathbb{C} \supseteq U \to \mathbb{C}$ is called complex differentiable (or holomorphic) if it is differentiable at every $z_0 \in U$.
- Complex differentiability implies infinite differentiability.

Complex Step Method (2)

Let $f : \mathbb{R} \supseteq D \to \mathbb{R}$ be naturally extendable to a holomorphic function.

Complex Step Method (2)

Let $f : \mathbb{R} \supseteq D \to \mathbb{R}$ be naturally extendable to a holomorphic function.

- Taylor expansion off the real axis ($x \in D$)

$$f(x + i\varepsilon) = f(x) + i\varepsilon f'(x) + \frac{1}{2} i^2 \varepsilon^2 f''(x) + \ldots$$

Complex Step Method (2)

Let $f : \mathbb{R} \supseteq D \to \mathbb{R}$ be naturally extendable to a holomorphic function.

- Taylor expansion off the real axis ($x \in D$)

$$f(x + i\varepsilon) = f(x) + i\varepsilon f'(x) + \frac{1}{2} i^2 \varepsilon^2 f''(x) + \ldots$$

- Take only the imaginary part on both sides

$$\operatorname{Im}(f(x + i\varepsilon)) = \varepsilon f'(x) - \frac{1}{3!} \varepsilon^3 f'''(x) + \ldots$$

Complex Step Method (2)

Let $f : \mathbb{R} \supseteq D \to \mathbb{R}$ be naturally extendable to a holomorphic function.

- Taylor expansion off the real axis ($x \in D$)

$$f(x + i\varepsilon) = f(x) + i\varepsilon f'(x) + \frac{1}{2} i^2 \varepsilon^2 f''(x) + \ldots$$

- Take only the imaginary part on both sides

$$\mathrm{Im}(f(x + i\varepsilon)) = \varepsilon f'(x) - \frac{1}{3!} \varepsilon^3 f'''(x) + \ldots$$

- Divide by ε

$$f'(x) = \frac{\mathrm{Im}(f(x + i\varepsilon))}{\varepsilon} + O(\varepsilon^2)$$

Comparison

Method	Error	Smallest ε
One-sided differencing	$O(\varepsilon)$	$u^{\frac{1}{2}}$
Central differencing	$O(\varepsilon^2)$	$u^{\frac{1}{3}}$
Complex step method	$O(\varepsilon^2)$	$u^{\frac{1}{2}}$

Jacobians via Finite Differencing

Let $(f_1, \dots, f_m) = f : \mathbb{R}^n \supseteq D \to \mathbb{R}^m$ differentiable. Then

$$J_f(x)p = \begin{pmatrix} \nabla f_1(x)^\top p \\ \vdots \\ \nabla f_m(x)^\top p \end{pmatrix} = \frac{f(x + \varepsilon p) - f(x)}{\varepsilon} + O(\varepsilon)$$

Jacobians via Finite Differencing

Let $(f_1, ..., f_m) = f : \mathbb{R}^n \supseteq D \to \mathbb{R}^m$ differentiable. Then

$$J_f(x)p = \begin{pmatrix} \nabla f_1(x)^\top p \\ \vdots \\ \nabla f_m(x)^\top p \end{pmatrix} = \frac{f(x + \varepsilon p) - f(x)}{\varepsilon} + O(\varepsilon)$$

Sparse Jacobians allow for more efficient computation. Example:

$$f : \mathbb{R}^4 \to \mathbb{R}^2,\ (x_1, x_2, x_3, x_4) \mapsto (x_1 + x_2, x_3 + x_4)$$

Hessians via Finite Differencing

Let $f : \mathbb{R}^n \supseteq D \to \mathbb{R}$ twice differentiable. If we know the gradient, we have

$$\nabla^2 f(x)p = \frac{\nabla f(x + \varepsilon p) - \nabla f(x)}{\varepsilon} + O(\varepsilon).$$

Hessians via Finite Differencing

Let $f : \mathbb{R}^n \supseteq D \to \mathbb{R}$ twice differentiable. If we know the gradient, we have

$$\nabla^2 f(x)p = \frac{\nabla f(x+\varepsilon p) - \nabla f(x)}{\varepsilon} + O(\varepsilon).$$

If we do not know the gradient, we can approximate

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(x) = \frac{f(x+\varepsilon e_i + \varepsilon e_j) - f(x+\varepsilon e_i) - f(x+\varepsilon e_j) + f(x)}{\varepsilon^2} + O(\varepsilon).$$

Chain Rule

Reminder (chain rule):
Let $x \in \mathbb{R}^n$, $h : \mathbb{R}^n \to \mathbb{R}^m$, $g : \mathbb{R}^m \to \mathbb{R}$. Then,

$$\nabla_x g(h(x)) = \sum_{i=1}^{m} \frac{\partial g}{\partial h_i(x)} \nabla_x h_i(x).$$

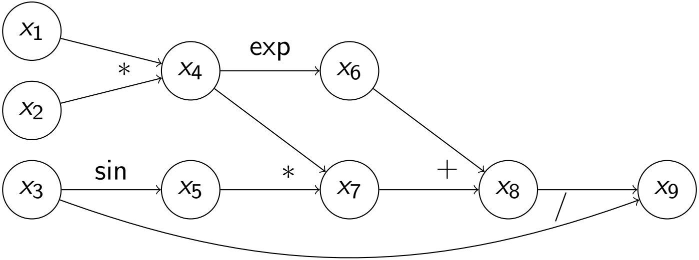
Computation Graph

$$f(x) = \frac{x_1 x_2 \sin(x_3) + e^{x_1 x_2}}{x_3}$$

Computation Graph

$$f(x) = \frac{x_1 x_2 \sin(x_3) + e^{x_1 x_2}}{x_3}$$

Introduce intermediate variables after every elementary operation:



Forward Mode AD (1)

- Choose a seed vector p for which we want to compute the directional derivative $\nabla f(x)^\top p$.

Forward Mode AD (1)

- Choose a seed vector p for which we want to compute the directional derivative $\nabla f(x)^\top p$.
- Simultaneously compute value x_i and directional derivative $D_p x_i := (\nabla x_i)^\top p$ of the intermediate variables. (Chain rule)

Forward Mode AD (1)

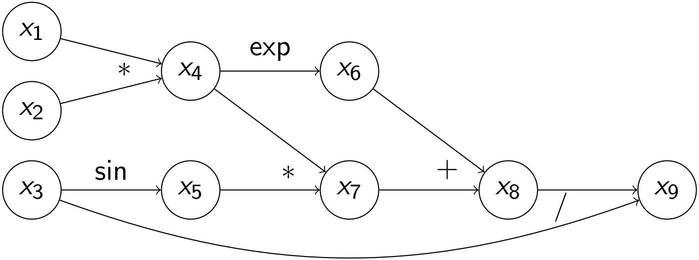
- Choose a seed vector p for which we want to compute the directional derivative $\nabla f(x)^\top p$.
- Simultaneously compute value x_i and directional derivative $D_p x_i := (\nabla x_i)^\top p$ of the intermediate variables. (Chain rule)
- When the directional derivative of the last intermediate variable $x_l = f(x)$ is computed, we have $D_p x_l = \nabla f(x)^\top p$.

Forward Mode AD (1)

- Choose a seed vector p for which we want to compute the directional derivative $\nabla f(x)^\top p$.
- Simultaneously compute value x_i and directional derivative $D_p x_i := (\nabla x_i)^\top p$ of the intermediate variables. (Chain rule)
- When the directional derivative of the last intermediate variable $x_l = f(x)$ is computed, we have $D_p x_l = \nabla f(x)^\top p$.
- To obtain the gradient $\nabla f(x)$, repeat for seed vectors $p = e_1, ..., e_n$.

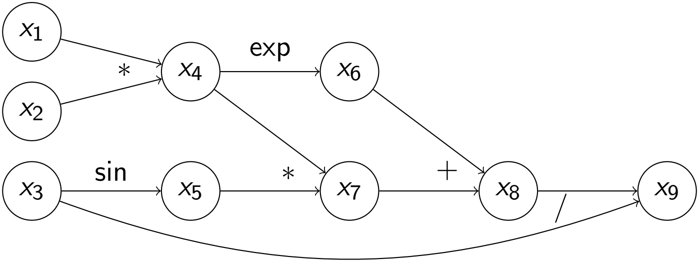
Forward Mode AD (2)

$$f(x) = \frac{x_1 x_2 \sin(x_3) + e^{x_1 x_2}}{x_3} \qquad D_p g(h(x)) = \sum_{i=1}^{m} \frac{\partial g}{\partial h_i(x)} D_p h_i(x)$$



Reverse Mode AD (1)

$$f(x) = \frac{x_1 x_2 \sin(x_3) + e^{x_1 x_2}}{x_3}$$



Reverse Mode AD (2)

- Evaluate all intermediate variables x_i, store the local derivatives $p(i, j) = \partial x_i / \partial x_j$, and construct a computation graph.

Reverse Mode AD (2)

- Evaluate all intermediate variables x_i, store the local derivatives $p(i,j) = \partial x_i / \partial x_j$, and construct a computation graph.
- Initialize adjoint variables $\bar{x}_l = 1$ for the last intermediate variable $x_l = f(x)$, and $\bar{x}_i = 0$ for all other intermediate variables x_i.

Reverse Mode AD (2)

- Evaluate all intermediate variables x_i, store the local derivatives $p(i,j) = \partial x_i / \partial x_j$, and construct a computation graph.
- Initialize adjoint variables $\bar{x}_l = 1$ for the last intermediate variable $x_l = f(x)$, and $\bar{x}_i = 0$ for all other intermediate variables x_i.
- Starting from a parent x_i of x_l, compute

$$\bar{x}_i \leftarrow \bar{x}_i + \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

as soon as $\partial f / \partial x_j$ becomes known. (x_j child of x_i)

Reverse Mode AD (2)

- Evaluate all intermediate variables x_i, store the local derivatives $p(i,j) = \partial x_i / \partial x_j$, and construct a computation graph.
- Initialize adjoint variables $\bar{x}_l = 1$ for the last intermediate variable $x_l = f(x)$, and $\bar{x}_i = 0$ for all other intermediate variables x_i.
- Starting from a parent x_i of x_l, compute

$$\bar{x}_i \leftarrow \bar{x}_i + \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

  as soon as $\partial f / \partial x_j$ becomes known. (x_j child of x_i)
- Once every child of x_i has contributed to the adjoint $\bar{x}_i$, we have $\bar{x}_i = \partial f / \partial x_i$. x_i is now finished.

Reverse Mode AD (2)

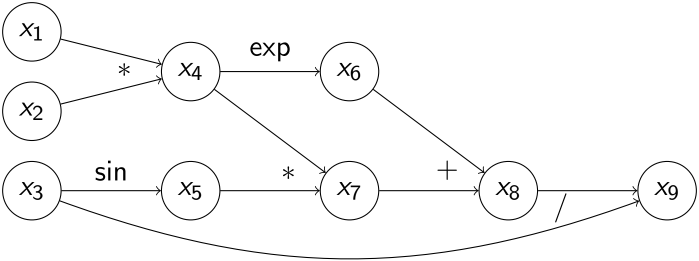
- Evaluate all intermediate variables x_i, store the local derivatives $p(i,j) = \partial x_i / \partial x_j$, and construct a computation graph.
- Initialize adjoint variables $\bar{x}_l = 1$ for the last intermediate variable $x_l = f(x)$, and $\bar{x}_i = 0$ for all other intermediate variables x_i.
- Starting from a parent x_i of x_l, compute

$$\bar{x}_i \leftarrow \bar{x}_i + \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial x_i}$$

  as soon as $\partial f / \partial x_j$ becomes known. (x_j child of x_i)
- Once every child of x_i has contributed to the adjoint $\bar{x}_i$, we have $\bar{x}_i = \partial f / \partial x_i$. x_i is now finished.
- Once all independent variables are finished, we can construct the gradient from their adjoints.

Reverse Mode AD (3)

$$f(x) = \frac{x_1 x_2 \sin(x_3) + e^{x_1 x_2}}{x_3} \qquad \bar{x}_i \leftarrow \bar{x}_i + \frac{\partial f}{\partial x_j}\frac{\partial x_j}{\partial x_i},\ x_j \text{ child of } x_i$$



Number of Evaluations

Let $(f_1, ..., f_m) = f : \mathbb{R}^n \to \mathbb{R}^m$ differentiable.

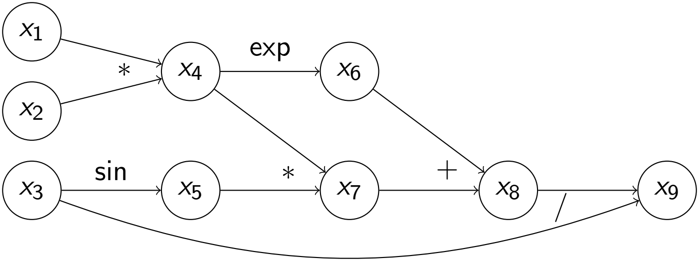
- Forward mode: compute directional derivative for n seed vectors $e_1, ..., e_n$.
- Reverse mode: perform reverse sweep for all m components $f_1, ..., f_m$ of f.

FD Implementation Example

```julia
function forwardGrad(f, x, y)

    eps = sqrt(eps(typeof(x)))

    dx = (f(x + eps, y) - f(x, y)) / eps
    dy = (f(x, y + eps) - f(x, y)) / eps
    return dx, dy
end
```

Reverse Mode AD Implementation Example (1)

$$f(x) = \frac{x_1 x_2 \sin(x_3) + e^{x_1 x_2}}{x_3} \qquad \bar{x}_i \leftarrow \bar{x}_i + \frac{\partial f}{\partial x_j}\frac{\partial x_j}{\partial x_i},\ x_j \text{ child of } x_i$$



Reverse Mode AD Implementation Example (2)

```julia
mutable struct Variable{T} <: Number
    value::T
    adjoint::T
    pjis::Vector{T}
    parents::Vector{Variable{T}}
    numchildren::Integer
    contributionfrom::Integer

    function Variable(value)
        x = new{typeof(value)}()
        x.value = value
        x.adjoint = zero(value)
        x.pjis = typeof(value)[]
        x.parents = typeof(x)[]
        x.numchildren = 0
        x.contributionfrom = 0
        return x
    end
```

Reverse Mode AD Implementation Example (3)

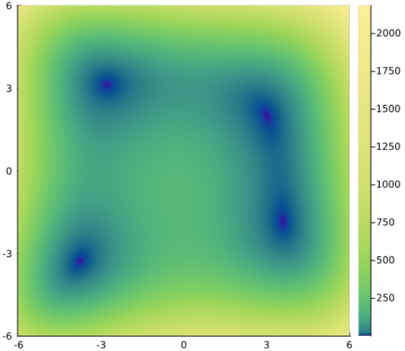
```
function /(x::Variable, y::Variable)
    x.numchildren += 1
    y.numchildren += 1
    z = Variable(x.value / y.value)
    z.pjis = [1 / y.value, -x.value / (y.value ^ 2)]
    z.parents = [x, y]
    return z
end
```

Reverse Mode AD Implementation Example (4)

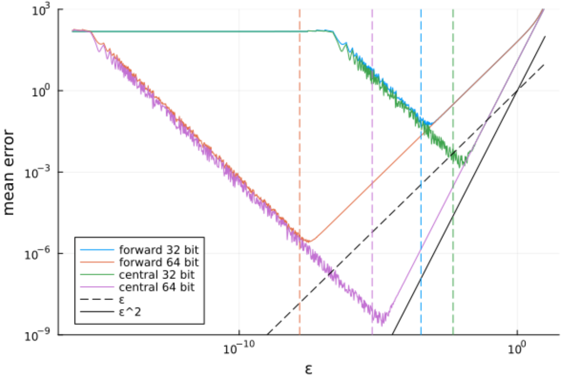
```
function backward!(x::Variable)
    x.adjoint = one(x.adjoint)
    zero_adjoints!(x)
    recursive_backward!(x)
end

function recursive_backward!(x::Variable)
    for i in 1:length(x.parents)
        x.parents[i].adjoint += x.pjis[i] * x.adjoint
        x.parents[i].contributionfrom += 1
        if x.parents[i].numchildren == x.parents[i].contributio
            recursive_backward!(x.parents[i])
        end
    end
    return
end
```

Himmelblau's Function

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$



Empirical Comparison (1)



Empirical Comparison (2)