

Implicit Differentiation

Benoît Legat

☐ Full Width Mode ☐ Present Mode

Table of Contents

Differentiating numerical procedures

Implicit function theorem

Implicit VJP and JVP

Sensitivity of a linear program

Differentiating numerical procedures

Consider a program that numerically converge to solutions:

```
function f(x)
  while abs(error) > tol
    y += # Make some step in the right direction
    error = # update error
  end
  return y
end
```



► How can we differentiate with respect to x ? Forward, reverse, something else ?

Square root example

To illustrate consider [this Jax example](#) rewriting $x^2 = a$ into $2x^2 = x^2 + a$ or $x = (x + a/x)/2$

fixed_point (generic function with 1 method)

```
1 function fixed_point(f, x)
2   fx = f(x)
3   while abs(x - fx) > 1e-6
4     x = fx
5     fx = f(x)
6   end
7   return x
8 end
```

my_sqrt (generic function with 1 method)

```
1 my_sqrt(a) = fixed_point(x -> (x + a / x) / 2, a)
```

1.4142135623746899

```
1 my_sqrt(2)
```

AD through fixed point

0.3535533906116973

```
1 FiniteDifferences.central_fdm(5, 1)(my_sqrt, 2.0)
```

0.35355339061171626

```
1 ForwardDiff.derivative(my_sqrt, 2.0)
```

0.35355339061171626

```
1 DI.derivative(my_sqrt, DI.AutoMooncake(), 2.0)
```

► Can't we do something simpler using the solution of the fixed point and the fixed point equation ?

Implicit function theorem ⇔

In a sense, the implicit function theorem can be thought as the mother theorem, as it can be used to prove envelope theorems, the adjoint state method and the inverse function theorem. Section 11.6 of [The Elements of Differentiable Programming book](#)

Inverse function theorem ⇔

Assume

- $f : \mathcal{W} \rightarrow \mathcal{W}$ is C^2
- $\partial f(w_0)$ is invertible

Then

- f is bijective from a neighborhood of w_0 to a neighborhood of $f(w_0)$
- For ω in a neighborhood of $f(w_0)$, f^{-1} is C^2 and $\partial f^{-1}(\omega) = (\partial f(f^{-1}(\omega)))^{-1}$

► **Proof sketch**

Implicit function theorem (univariate case) ⇔

Assume

- $F : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$
- (w_0, λ_0) such that $F(w_0, \lambda_0) = 0$ and $\partial_1 F(w_0, \lambda_0) \neq 0$
- $F(w, \lambda)$ is C^2 in a neighborhood \mathcal{U} of (w_0, λ_0)

Then there exists a neighborhood $\mathcal{V} \subseteq \mathcal{U}$ there exists $w^*(\lambda)$ such that

$$\begin{aligned} w^*(\lambda_0) &= w_0 \\ F(w^*(\lambda), \lambda) &= 0, & \forall (w^*(\lambda), \lambda) \in \mathcal{V} \\ \partial w^*(\lambda) &= -\frac{\partial_2 F(w^*(\lambda), \lambda)}{\partial_1 F(w^*(\lambda), \lambda)} \end{aligned}$$

Example ⇔

Implicit relation between x and y :

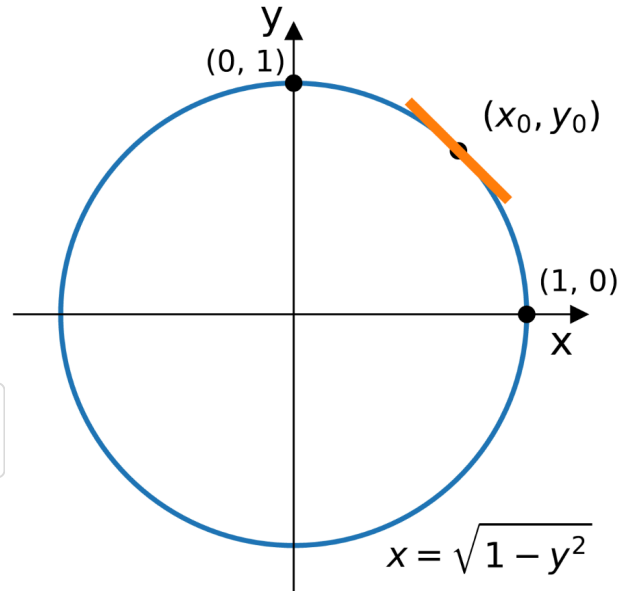
$$x^2 + y^2 = 1$$

Two possible explicit functions:

$$y^+(x) = \sqrt{1 - x^2}$$

$$y^-(x) = -\sqrt{1 - x^2}$$

► Does that contradict the Inverse Function Theorem (IFT) ?



Implicit function theorem (multivariate case) ⇔

Assume

- $F : \mathcal{W} \times \Lambda \rightarrow \mathcal{W}$
- (w_0, λ_0) such that $F(w_0, \lambda_0) = 0$ and $\partial_1 F(w_0, \lambda_0)$ is invertible
- $F(w, \lambda)$ is C^2 in a neighborhood \mathcal{U} of (w_0, λ_0)

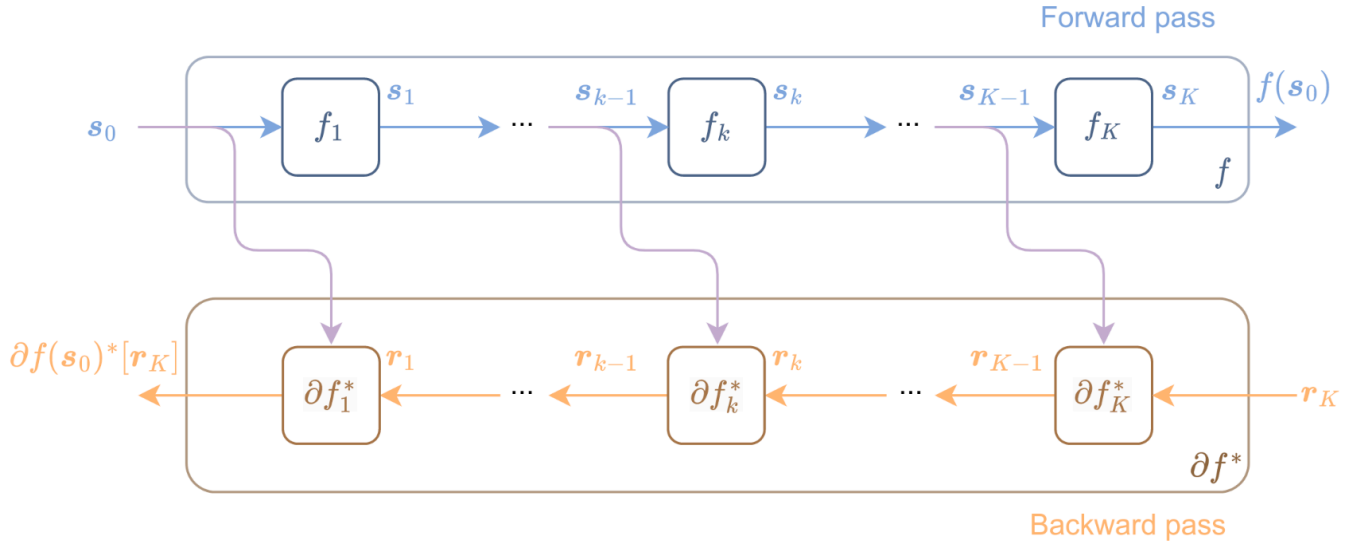
Then there exists a neighborhood $\mathcal{V} \subseteq \mathcal{U}$ there exists $w^*(\lambda)$ such that

$$\begin{aligned} w^*(\lambda_0) &= w_0 \\ F(w^*(\lambda), \lambda) &= 0, & \forall (w^*(\lambda), \lambda) \in \mathcal{V} \\ \partial w^*(\lambda) &= -\partial_1 F(w^*(\lambda), \lambda)^{-1} \partial_2 F(w^*(\lambda), \lambda) \end{aligned}$$

► Proof

Implicit VJP and JVP \Leftrightarrow

How can we integrate implicit function in a chain of functions? Say, $\mathbf{s}_{k-1} = \lambda$ and $\mathbf{f}_k(\lambda) = \mathbf{w}^*(\lambda)$ such that $\mathbf{F}(\mathbf{w}^*(\lambda), \lambda) = 0$.



Pushforward operator (JVP) \Leftrightarrow

Let $\mathbf{A} = -\partial_1 \mathbf{F}(\mathbf{w}^*(\lambda), \lambda)$ and $\mathbf{B} = \partial_2 \mathbf{F}(\mathbf{w}^*(\lambda), \lambda)$. Assuming the condition of the IFT holds, we have

$$\mathbf{A} \partial \mathbf{w}^*(\lambda) / \partial \mathbf{s}_0 = \mathbf{B} \partial \lambda / \partial \mathbf{s}_0$$

Given a forward tangent $\mathbf{t}_{k-1} = \partial \lambda / \partial \mathbf{s}_0$, the forward tangent $\mathbf{t}_k = \partial \mathbf{w}^*(\lambda) / \partial \mathbf{s}_0$ can be obtained by solving the linear system:

$$\begin{aligned} \mathbf{A} \mathbf{t}_k &= \mathbf{B} \mathbf{t}_{k-1} \\ \mathbf{t}_k &= \mathbf{A}^{-1} \mathbf{B} \mathbf{t}_{k-1} \end{aligned}$$

Interlude: Repeated linear system solve \Leftrightarrow

We need to solve the system $\mathbf{A} \mathbf{t}_k = \mathbf{B} \mathbf{t}_{k-1}$ once by forward tangent, so once by entry of \mathbf{s}_0 in case of forward-mode AD. We may also pre-compute $\mathbf{A}^{-1} \mathbf{B}$ by solving one linear system by column of \mathbf{B} . In both case, we need to solve several linear system with the **same** \mathbf{A} matrix.

```
A = 2x2 Matrix{Int64}:  
  1  2  
  3  4
```

```
1 A = [1 2; 3 4]
```

```
b = ▶ [5, 6]
```

```
1 b = [5, 6]
```

Classical Gaussian elimination finds the solution for one vector only, even though the same row operations would be applied for different vectors.

```
2x3 Matrix{Float64}:  
 1.0  0.0 -4.0  
 0.0  1.0  4.5
```

```
1 rref([A b])
```

Solution: precompute the LU decomposition ⇔

LU decomposition remember the row operations to make U triangular in the L matrix.

```
F = LU{Float64, Matrix{Float64}, Vector{Int64}}  
L factor:  
 2x2 Matrix{Float64}:  
  1.0      0.0  
 0.333333  1.0  
U factor:  
 2x2 Matrix{Float64}:  
  3.0  4.0  
  0.0  0.666667
```

As L and U are triangular, solving the linear systems $LUx = b$ can now be done by solving two simple linear systems (and permutation in case of pivoting)

1. $Lc = b$
2. $Ux = c$

► Can we also solve the linear system with only access to the matrix-vector product of the linear map $\partial_1 F(w^*(\lambda), \lambda)$? (aka matrix-free inversion)

Interlude: Adjoint of inverse \Leftrightarrow

Consider a linear map $A : \mathcal{X} \rightarrow \mathcal{Y}$ between linear spaces of the same dimension. Assume A is invertible, what is the adjoint of A^{-1} ?

► Is the adjoint A^* always invertible ?

► Is the adjoint of the inverse equal to the inverse of the adjoint ?

Pullback operator (VJP) \Leftrightarrow

The pullback operation is the adjoint of the pushforward operator:

$$\langle r, A^{-1}Bt \rangle = \langle (A^{-1})^*r, Bt \rangle = \langle (A^*)^{-1}r, Bt \rangle = \langle B^*(A^*)^{-1}r, t \rangle$$

So the pullback operator maps r to $B^*(A^*)^{-1}r$.

This means that it first solves the linear system $A^*v = r$ (possibly in a matrix-free way) and then returns Bv .

Sensitivity of a linear program \Leftrightarrow

Consider the primal-dual pair of programs

$$\begin{array}{ll} \min c^\top x & \max b^\top y \\ Ax = b & A^\top y \leq c \\ x \geq 0 & \end{array}$$

The Lagrangian function is

$$\mathcal{L}(x, y) = c^\top x - y^\top (Ax - b) = y^\top b - (A^\top y - c)^\top x$$

The KKT condition give

$$\begin{array}{ll} \frac{\partial \mathcal{L}}{\partial y} = Ax - b = 0 & \Leftrightarrow Ax = b \\ (A^\top y - c)^\top x \geq 0 & \Leftrightarrow \text{Diag}(x)(A^\top y - c) = 0 \end{array}$$

IFT for linear programs \Leftrightarrow

The system of equation to consider for the IFT is therefore



$$F((x, y), (A, b, c)) = (Ax - b, \text{Diag}(x)(A^\top y - c))$$

We have

$$\partial_1 F = \begin{bmatrix} A & 0 \\ \text{Diag}(A^\top y - c) & \text{Diag}(x)A^\top \end{bmatrix}$$

See [OptNet: Differentiable optimization as a layer in neural networks](#) for a generalization to quadratic objective.

Acknowledgements and further readings \Leftrightarrow

- Example from : [Implicit function differentiation of iterative implementations](#)
- Chapter 11 of [The Elements of Differentiable Programming book](#)
- See [DiffOpt.jl](#) for implicit differentiation of  optimization problems.

Utils

=====

```
1 using PlutoUI, PlutoUI.ExperimentalLayout, HypertextLiteral, PlutoTeachingTools
```

```
1 import ForwardDiff, FiniteDifferences, Mooncake
```

```
1 import DifferentiationInterface as DI
```

```
1 using LinearAlgebra, RowEchelon
```

img (generic function with 3 methods)

qa (generic function with 2 methods)

```
1 begin
2 function qa(question, answer)
3     return @html("<details><summary>$question</summary>$answer</details>")
4 end
5 function _inline_html(m::Markdown.Paragraph)
6     return sprint(Markdown.htmlinline, m.content)
7 end
8 function qa(question::Markdown.MD, answer)
9     # 'html(question)' will create '<p>' if 'question.content[]' is
'Markdown.Paragraph'
10    # This will print the question on a new line and we don't want that:
11    h = HTML(_inline_html(question.content[]))
12    return qa(h, answer)
13 end
14 end
```