

14<sup>th</sup> European Workshop on AD

ADJOINTS OF FIXED-POINT ITERATIONS

Ala Taftaf

INRIA, SOPHIA  
ANTIPOLIS, FRANCE

The Inria logo is a stylized, cursive script in a red-to-orange gradient, positioned at the bottom right of the slide.

# Adjoint Algorithms



- **Adjoint algorithms : the most efficient way to obtain gradient of a numerical simulation**
- **Computing the gradients has a cost (time, memory)**
- **One way around : take advantage of the structures of the given program (parallel loops, **fixed point methods**,..)**

# Fixed Point Iterations

- Many **implicit functions**  $F(\mathbf{z}, \mathbf{x}) = \mathbf{0}$  are computed with convergent iterations:

Initial guess  $\mathbf{z}_0(\mathbf{x})$

Iterate  $\mathbf{z}_{k+1}(\mathbf{x}) = \varphi(\mathbf{z}_k(\mathbf{x}), \mathbf{x})$  

To fixed point  $\mathbf{z}_*(\mathbf{x}) = \varphi(\mathbf{z}_*(\mathbf{x}), \mathbf{x})$

- Followed by an **objective function**  $y = f(\mathbf{z}_*, \mathbf{x})$

# Adjoint of Fixed Point Iterations

- **First iterations of a fixed-point search :**

$$z_0(x)$$

$$z_1(x) = \varphi(z_0(x), x)$$

.....

**operate on a meaningless state vector =>No adjoint needed**

- **Adjoin only the (few) last iterations**
- **Save trajectory storage**

**At least two authors have studied mathematically fixed point iterations with the goal of defining an efficient adjoint : Christianson and Griewank**

# Christianson's strategy (BC)

$$z_{k+1} = \varphi(z_k(x), x)$$

$z_*$

*Forward Sweep*

$$y = f(z_*, x)$$

$$\bar{y} = 1$$

$$\bar{z} = \bar{y} \cdot f_z(z_*, x)$$

$$\bar{x} = \bar{y} \cdot f_x(z_*, x)$$

$$\bar{w}_0 = \bar{z} \downarrow$$

$$\bar{w}_{k+1} = \bar{w}_k \cdot \varphi_z(z_*, x) + \bar{z}$$

$\bar{w}_*$

*Backward Sweep*

$$\bar{x} = \bar{w}_* \cdot \varphi_x(z_*, x) + \bar{x}$$

# Griewank's strategy (AG) : Piggyback

$$w_k = F(z_k, x)$$

$$z_{k+1} = z_k - P_k \cdot w_k$$

$z_*$

$$y = f(z_*, x)$$

**Original Program**

$$w_k = F(z_k, x)$$

$$\bar{z}_k = \bar{w}_k \cdot F_z(z_k, x) + \bar{y} \cdot f_z(z_k, x)$$

$$z_{k+1} = z_k - P_k \cdot w_k$$

$$\bar{w}_{k+1} = \bar{w}_k - \bar{z}_k \cdot P_k$$

$z_*, \bar{w}_*$

$$\bar{x}_* = \bar{w}_* \cdot F_x(z_*, x) + \bar{y} \cdot f_x(z_*, x)$$

**Adjoint Program**

## AG and BC: Similarities

- **Both manage to avoid naïve inversion of the original sequence of iterations => Save trajectory storage**
- **Stopping criterion of the adjoint fixed point is distinct from the original test**
- **Convergence rate is similar for the derivative computation and the original computation**

# AG and BC: Differences

|                                | AG                                                               | BC                                            |
|--------------------------------|------------------------------------------------------------------|-----------------------------------------------|
| <b>Shape of iteration step</b> | <b>Additional Assumptions</b><br>$z_{k+1} = z_k - P_k \cdot w_k$ | <b>General</b><br>$z_{k+1} = \varphi(z_k, x)$ |
| <b>Start of adjoining</b>      | <b>From the Beginning</b><br>(more iterations)                   | <b>After a total convergence</b>              |
| <b>Sequel of F.P. loop</b>     | <b>Adjoin repeatedly</b>                                         | <b>Adjoin once</b>                            |



## Reasons for our choice

Mostly for implementation reasons, we want :

- No assumption for iteration shape
- no multiple differentiation of the sequel
- differentiate only the (few) last iterations
- to preserve the 2 sweeps structure of the adjoint code

## Identifying the F.P. structure

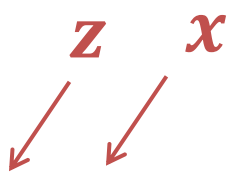
In the computation of the adjoint ( $\bar{w}_{k+1} = \bar{w}_k \cdot \varphi_z(z_*, x) + \bar{z}$ ) we need to determine  $(\varphi, z, x)$  in the original program =>

Adding new directives

```

$AD Begin FP loop
  REPEAT UNTIL ( z stationary)
    $AD Begin FP iteration
    z =  $\varphi(z, x)$ 
    $AD END FP iteration
  END REPEAT
$AD END FP loop
y =  $f(z, x)$ 

```



The diagram shows two red arrows pointing from the variables  $z$  and  $x$  in the function  $\varphi(z, x)$  of the adjoint equation above to the corresponding variables in the code block below. The  $z$  arrow points to the  $z$  in the loop condition and the assignment, while the  $x$  arrow points to the  $x$  in the assignment.

# Discussion

- Is a differentiation strategy for the iterative computations really needed in practice, as yet more focused strategies exist ?
- Have you samples of code where this strategy could be effective ?

# Discussion

- Is a differentiation strategy for the iterative computations really needed in practice, as yet more focused strategies exist ?
- Have you samples of code where this strategy could be effective ?

**Thank you for your attention!**