Assignment-5 (Computing Derivatives)

Vishwa Mohan Tiwari

Wednesday 15th November, 2023

1 5.1 : Finite differences, Complex step and Automatic Differentiation

1.1 Finite difference Method

The most popular methods of computing derivatives are the finite- difference methods. These methods are derived by combing Taylor series expansions, we can obtain the derivatives of any order with certain order of truncation error. The forward difference method (FD) Eq 1 and central difference method (CD) Eq 2 are first and second order estimation of the first derivative . In theory as h approches zero the estimate becomes better. This method was coded by following the Algorithm 6.1 in Engineering design optimization (EDO) book. Every finite difference evaluation yields a column of the Jacobian matrix ($\frac{\partial f}{\partial x_j}$)

$$\frac{\partial f}{\partial x_j} = \frac{f(x + h\hat{e}_j) - f(x)}{h} + O(h) \tag{1}$$

$$\frac{\partial f}{\partial x_j} = \frac{f(x + h\hat{e}_j) - f(x - \hat{e}_j)}{2h} + O(h^2) \tag{2}$$

1.2 Complex step

This method is used to find the derivatives of real function using the complex variables, access to the source code is needed and it can not be applied to a black box method. We derive the derivatives using taylor series expansion with a purely step ih. Unlike finite- differences there is no subtractive cancellation error, the only error associated with this method is truncation error. Equation 3 shows the first order derivative. Algorithm 6.2 from EDO is used for coding this method.

$$\frac{\partial f}{\partial x_j} = \frac{Im(f(x + ih\hat{e}_j))}{h} + O(h^2) \tag{3}$$

1.3 Forward method Automatic differentiation

Also known as computational differentiation, AD is a systematic implementation of the chain rule. The idea behind AD is that even the most complicated codes are made up sequential of operations and each operation can be symbolically differentiated w.r.t the variables in that expressions. We can represent a variable in a code as an explicit function of the *previous* variables and itself Equation 4.

$$v_i = v_i(v_1, v_2, ..., v_{i-1}) \tag{4}$$

Following the sequence of operations, with the loops *unrolled* and applying chain rule we can obtain, AD in forward and reverse mode. The total derivative of a expression v_i with right to v_j is given in Equation 5

$$\frac{dv_i}{dv_j} = \sum_{k=j}^{i-1} \frac{\partial v_i}{\partial v_k}\frac{\partial v_k}{\partial v_j} \tag{5}$$

where i > j. We seed $\frac{dv_j}{dv_j} = 1$. The implementation for forward AD is done using operator overloading. A new user-defined class named `AD` is made in python with two attributes `value` and `derivative`. Then for overloading operators, method such as `__add__`, `__mul__` etc are defined such that they act on the objects which are the instances of the class `AD`. The class structure is shown in the listing 1.

```python
# Defining class for operator overloading
class AD:
    def __init__(self,value, derivative):
        self.value =value
        self.derivative = derivative

    def __add__(self, other):
        if isinstance(other, AD):
            # check if 'other' is an instance of the same class
            value = self.value + other.value
            derivative = self.derivative + other.derivative
            return AD(value, derivative)
        else:
            raise TypeError("Unsupported operand type")

    def __mul__(self, other):
        ...
        ...
        ...

    def __str__(self):
        return f"value= {self.value} and derivative = {self.derivative}"
```

Listing 1: Class structure of the objects used for performing automatic differentiation

`AD` class takes two arguments which variable value and its derivative and the function of interest is evaluated using these data type and by the end function evaluations we also have it's derivative.

1.4 Questions 1(a)

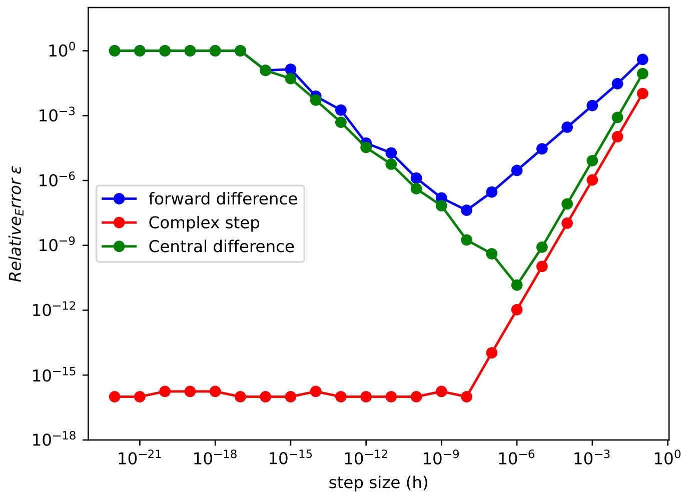

Figure 1: Plot of relative error vs step size for forward and central finite difference and complex step

The complex step can never give zero error when compared with the analytical expression, as there is some truncation error and even if the truncation error is reduced there still will be errors due finite precision arithmetics. The Figure 1 shows that complex step and central difference estimates converge quadratically whereas the forward difference has linear to converges.

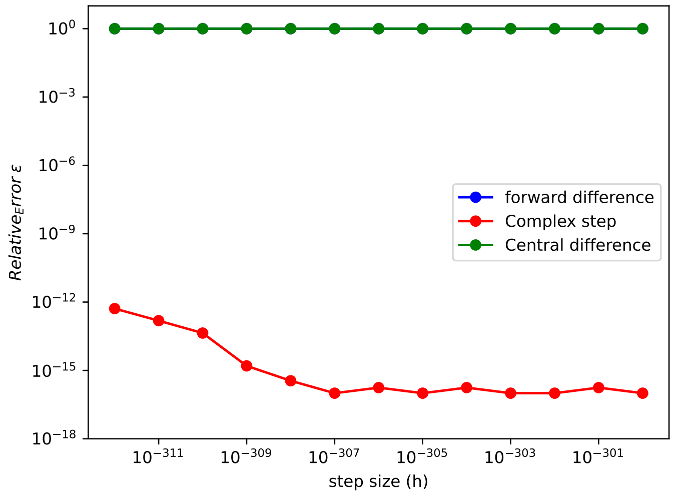

Figure 2: Plot of relative error vs step size for forward and central finite difference and complex step

After a certain step size (10^{-5} for central and 10^{-8} for forward difference) the subtractive cancellations become increasingly dominant and at step size of 10^{-16} the relative error becomes 1.

Whereas the relative error for the complex step keeps decreasing till step of around 10^{-8} where it reaches machine precision.

As seen in Figure 2, the relative error for complex step starts raising around the step size of 10^{-308} that's when underflow happens . Underflow means that the step size becomes so small that the imaginary parts gets impacted. So the results are smaller than the smallest numerical value that can be represented.

To eliminate the truncation error to the precision of the function value a step size value of h = 10^{-200} is good enough.

1.5 Questions 1(b)

The Table 1, shows that values evaluated from AD and complex step match till 15th decimal place.

AD	Complex
4.05342789389862	4.05342789389862

Table 1: Comparision of derivative at x=1.5 for AD and Complex step

2 5.2 Unified Derivative Equation

Let us consider the Equation 6 which is forward form of the UDE. This is derived and given in EDO as Equation 6.65.

$$\frac{\partial r}{\partial u}\frac{du}{dr} = I \tag{6}$$

In the given problem we $\hat{r}$ and $\hat{u}$ taken as shown Equation 7, where we distinguish $\check{M} = 1$ as given input (assumed) from M (the actual variable in the UDE system), f (the variable) from $\check{f} = E - M$ (an explicit function of M and E). Similarly, r is the vector of variables associated with the residual and $\check{r}$ is the residual function itself.

$$\hat{r} = \begin{bmatrix} M - \check{M} \\ r - \check{r} \\ f - \check{f} \end{bmatrix} \hat{u} = \begin{bmatrix} M \\ E \\ f \end{bmatrix} \tag{7}$$

Using forward form of UDE and the definitions above, we get matrix system shown in Equation 8.

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{-\partial \check{r}}{\partial M} & \frac{-\partial \check{r}}{\partial E} & 0 \\ \frac{-\partial \check{f}}{\partial M} & \frac{-\partial \check{f}}{\partial E} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \frac{dE}{dM} & \frac{dE}{\partial r} & 0 \\ \frac{df}{dM} & \frac{df}{dr} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

Substituting the derivatives we get, Equation 9

$$\begin{bmatrix} 1 & 0 & 0 \\ -(-1) & -(1-e\cos(E)) & 0 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \frac{dE}{dM} & \frac{dE}{\partial r} & 0 \\ \frac{df}{dM} & \frac{df}{dr} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

Ignoring all column expect $[1, \frac{dE}{dM}, \frac{df}{dM}]$, we get Equation 10

$$\begin{bmatrix} 1 & 0 & 0 \\ -(-1) & -(1-e\cos(E)) & 0 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \frac{dE}{dM} \\ \frac{df}{dM} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{10}$$

Upon solving the linear system formed by forward UDE in Equation 10 with $e = 1$ amd $\check{M} = 1$, we get.

$$\begin{bmatrix} 1 \\ \frac{dE}{dM} \\ \frac{df}{dM} \end{bmatrix} \begin{bmatrix} 1 \\ 0.73785996 \\ -0.26214004 \end{bmatrix} \tag{11}$$

Hence, using UDE $\frac{df}{dM} = -0.26214004$.

We can verify the total derivative found using UDE, with that found using Finite difference (forward difference) with a newton solver. For this we would perturb M, re-solve the governing equations to obtain E, and then compute f which would account for both dependencies. The results are shown in table 2 . We see that the derivative from FD matches with that of UDE till 3rd decimal.

Finite Difference	UDE
-0.26243028	-0.26214004

Table 2: Comparision between total derivatives found with UDE and Finite Difference for e = 1 and M =1

3 5.3

The given ten bar system, our constraint is the absolute value of the stress $|\sigma_i| < \sigma_{y_i}$. Further upon assembling the system we $r := Kd - q$ and $\sigma = Sd$

3.1 Finite Difference(FD)

For finding the Jacobian using FD, we use the FD code we developed previously (h = 10^{-8}), the function values (σ for all 10 bars) comes from `truss.py`. We take the absolute value of the stress constraint and perform one evaluation of FD to get the column of the Jacobian $\frac{\partial \sigma}{\partial x_i}$. After 10 evaluation of FD we get the entire Jacobian matrix.

3.2 Complex step (CS)

Since we are dealing with absolute value while using CS we redefine it as:

$$|x+iy| = \begin{bmatrix} if & x<0 & -x-iy \\ if & x>=0 & x+iy \end{bmatrix} \tag{12}$$

We use the algorithm developed previously. And get the complete Jacobian over loops.

3.3 Direct Method (DI)

The total derivative equation is give below:

$$\frac{d|\sigma|}{dx} = \frac{\partial|\sigma|}{\partial x} - \frac{\partial|\sigma|}{\partial u}\frac{\partial r^{-1}}{\partial u}\frac{\partial r}{\partial x} \tag{13}$$

Here, $\frac{\partial|\sigma|}{\partial x} = 0$ as stress is not directly depended of the area of the trusses. We know that the direct method solves linear system where is $\frac{\partial r}{\partial x}$ is on the right hand side.

For the given problem, $r = Ku - q$, and x_i denotes the cross-sectional aera of i^{th} truss.

$$\frac{\partial r}{\partial x_i} = \frac{\partial K}{\partial x_i}u, \frac{\partial r}{\partial x_i} = \frac{K(x+h\hat{e}_i)u - K(x)u}{h} \tag{14}$$

$$\frac{\partial r}{\partial u} = K, \frac{\partial|\sigma|}{\partial u} = sign(S.u)S \tag{15}$$

For this direct method linear system to be solved for ϕ_i and the derivative $\frac{d\sigma}{dx_i}$ is givne in the equation below:

$$K\phi_i = \frac{\partial}{\partial x_i}(Ku),, \frac{d\sigma}{dx_i} = -sign(S.u)S\phi_i \tag{16}$$

3.4 Adjoint (AJ)

We know that for the adjoint method we solve the linear system where $\frac{\partial f^T}{\partial u}$ is on the right side. And the total derivative equation becomes as follows:

$$\frac{d|\sigma|}{dx} = \frac{\partial|\sigma|}{\partial x} - \Psi^T\frac{\partial r}{\partial x} \tag{17}$$

The linear system to be solved for Ψ is given below, where j corresponds to each truss member:

$$K\Psi_j = sign(S.u)S^T_{j,*} \tag{18}$$

3.5 KS aggregation

We currently have 10 constraint value, 1 corresponding to each truss. We can reduce the number of ad-joint solutions by aggregating constraints.

We will do aggregation using `max()` function and our constraint is $|\sigma|$. Which means that for the entire truss system we will only use a single value as the constraint. So, the modified Kreisselmeier and Steinhauser aggregation function formed is shown in Equation 19

$$\bar{g}_{KS}(\rho, g) = maxg_j + \frac{1}{\rho} ln(\sum_{j=1}^{n_g} \exp(\rho(g_j - maxg_j))) \tag{19}$$

When using the KS aggregated with DT or AJ, we replace the $\frac{\partial|\sigma|}{\partial u}$ $\frac{\partial g(|\sigma|,\rho)}{\partial u}$, where $\rho = 1000$.

3.6 Jacobian and Comparisions

The jacobian of stress constraint obtain by various method (non KS methods) show similar sparsity and distributions. This can be seen in the Figure 3 below:

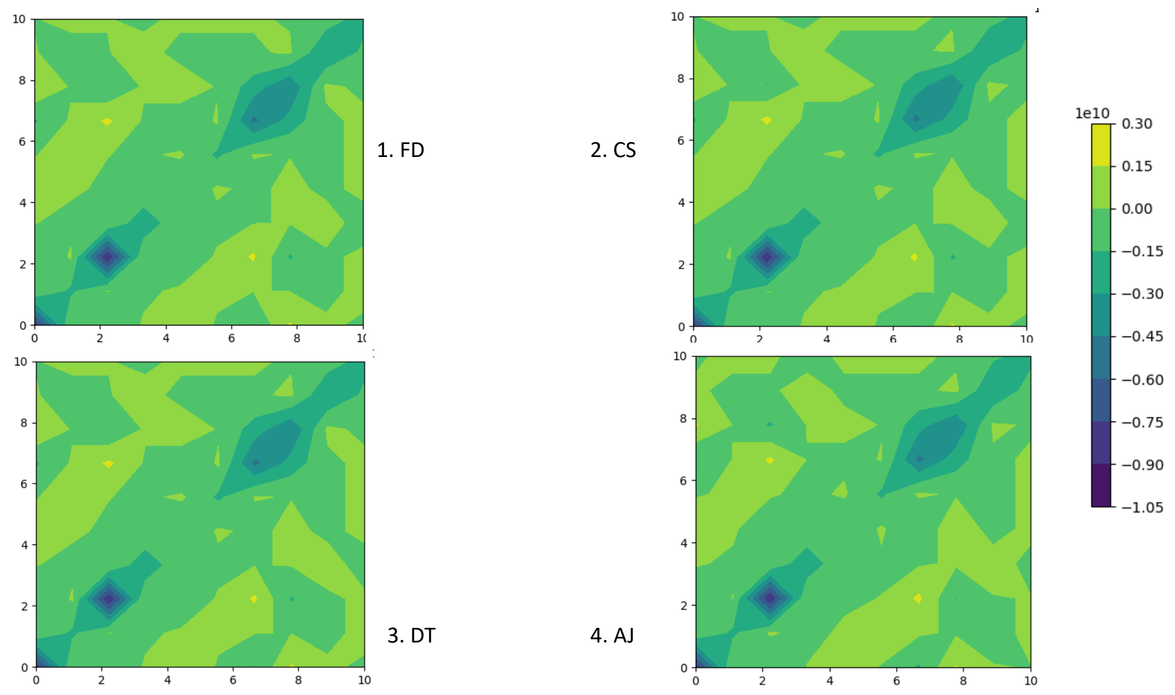

Figure 3: Contours of the Jacobians computed with different methods

We observe a higher sensitivity of stresses of truss 2 or 3 and 7 or 8 as changes are made in the cross-sectional area of the truss 2or 3 and 7 or 8 (purple spots in the jacobian contours).

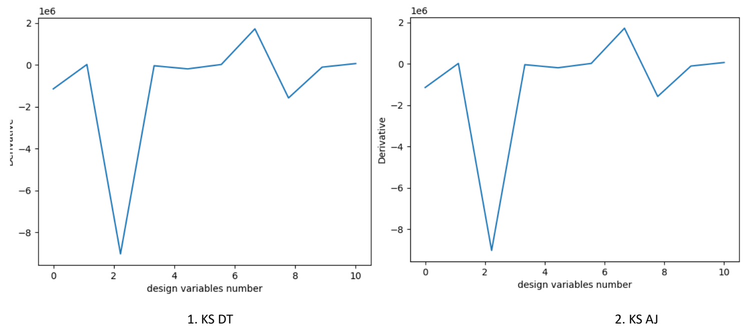

Figure 4: KS gradient of the max(—σ—)

For the derivative obtained by aggregating the constraints it observed that the truss 2, truss 7 and truss 8 have significant impact of the stress constraints. This is seen as peaks and valleys in the Figure 4 near the design variable number 2, 7 and 8. Consider the Implicit Analytical methods to be exact, then the maximum relative error of FD and CS methods is shown in the table below:

	DT as Exact	AJ as Exact
Relative error in FD	1.21E-06	4.896654589
Relative Error in CS	6.02E-13	4.896654589

Table 3: Comparison between implicit analytical method with FD and CS

The high relative error incase of AJ method with FD and CS can to be due to the scales involved in this problem. If scaling is done the error could be brought down. Also the computational cost in calculating the derivative of each method is shown below:

Method	Time (s)
FD	0.018954277
CS	0.019023895
DT	0.150999308
AJ	0.151484728
KS DT	0.143031597
KS AJ	0.140961885

Table 4: Caption

It is observed that finite different different takes the least time of all methods. The cpu time progressively increases till AJ and for KS DT and KS AJ, we see a decrease in the cpu time which is expected as the number of direct and adjoint operations are reduced.