NATIONAL TECHNICAL UNIVERSITY OF ATHENS (NTUA)
SCHOOL OF MECHANICAL ENGINEERING
LAB. OF THERMAL TURBOMACHINES
PARALLEL CFD & OPTIMIZATION UNIT (PCOpt/NTUA)

Μέθοδοι Αιτιοκρατικής Βελτιστοποίησης

Deterministic or Gradient-Based Optimization

Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

8 Νοε. 2023

Μέθοδοι Βελ/σης (Optimization Methods)

Gradient-Based Method (Μέθοδοι Κλίσης F)

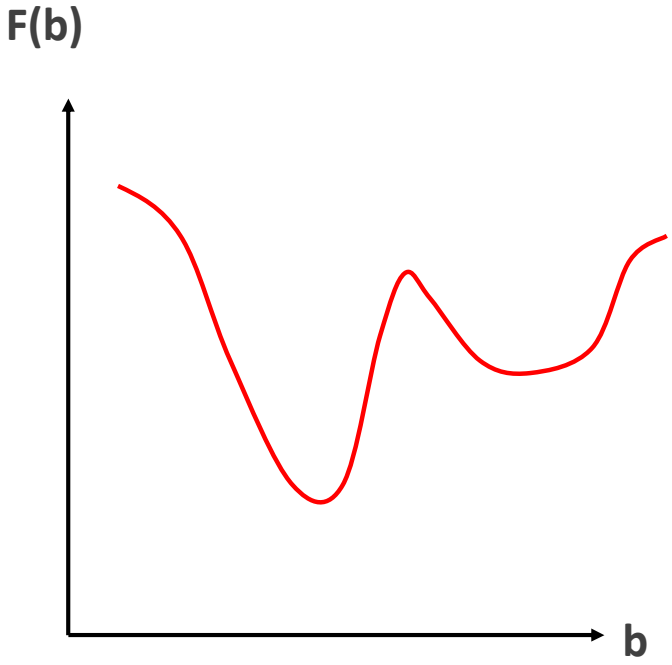
vs.

Stochastic Methods (Στοχαστικές Μέθοδοι)

Individual-based Methods (Ατομικές Μέθοδοι)

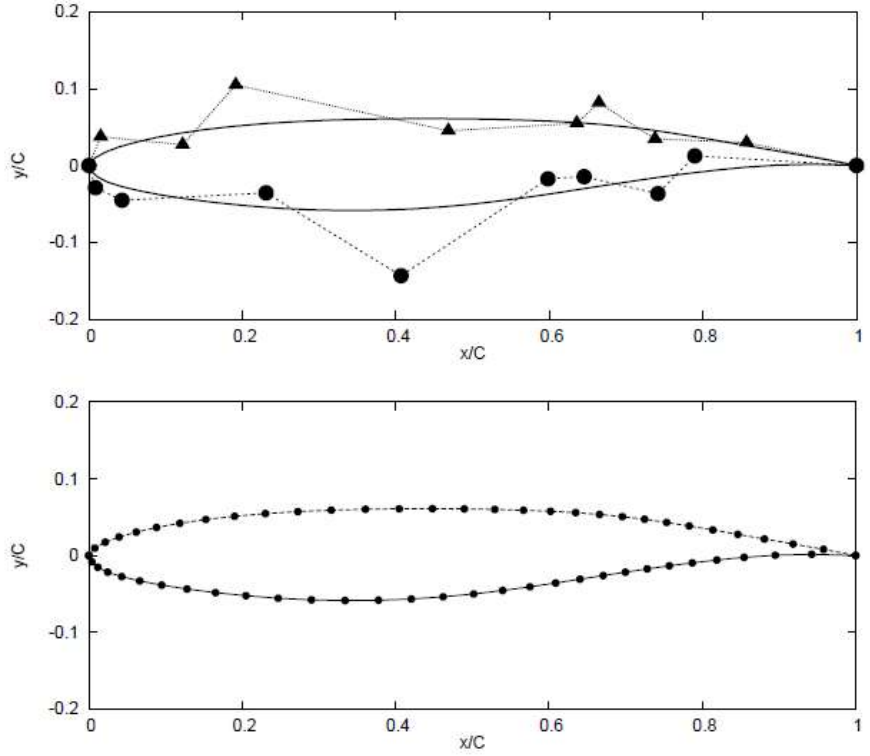
vs.

Population-based Methods (Πληθυσμιακές Μέθοδοι)



Grad(F) Η Κλίση του F (ή του J)

Λχ σε ένα πρόβλημα σχεδιασμού βελτιστοποίησης αεροδυναμικής μορφής (aerodynamic shape design-optimization)



$$\overrightarrow{b} = (b_1, b_2, b_3, \ldots, b_N)^T \in R^N$$

$$\nabla F = (\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \ldots \frac{\partial F}{\partial b_N})$$

- **Accuracy**
- **Computational Cost**
- **Method S/W Development Time**

Αιτιοκρατική Βελ/ση (Gradient-based Methods)

Μέθοδος Απότομης Καθόδου Steepest Descent Method:

$$\overrightarrow{b}^{new} = \overrightarrow{b}^{old} - \eta \nabla F(\overrightarrow{b}^{old})^T$$

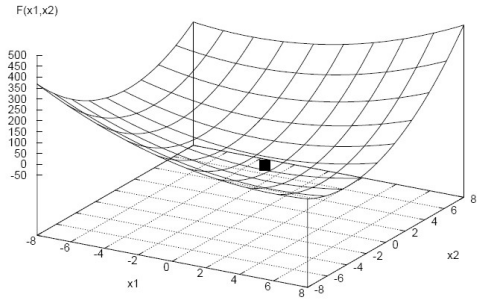
$$b_n^{new} = b_n^{old} - \eta \frac{\delta F}{\delta b_n}, \quad i = 1, \ldots, N$$

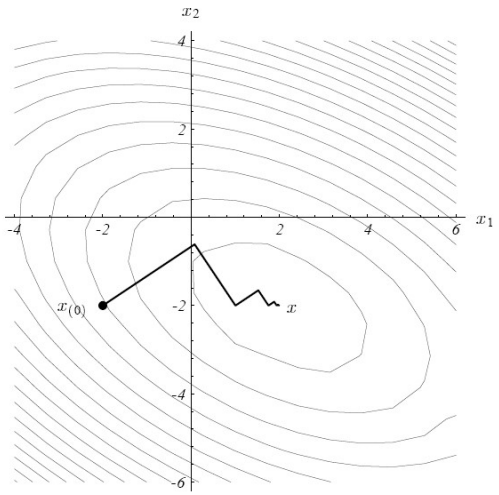
Newton Μέθοδοι (ή, Quasi-Newton) Method):

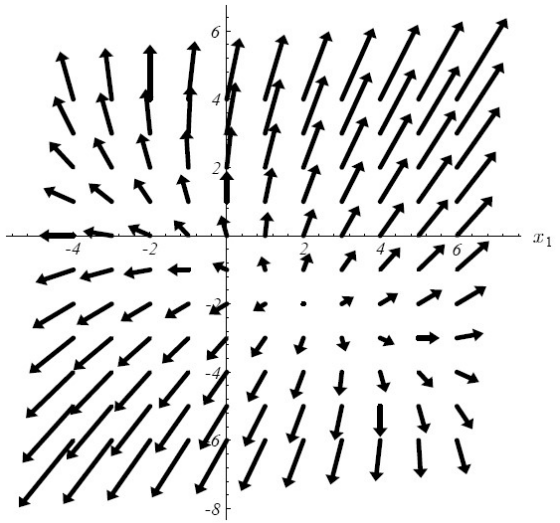
$$\overrightarrow{b}^{new} = \overrightarrow{b}^{old} - \nabla^2 F(\overrightarrow{b}^{old})^{-1} \nabla F(\overrightarrow{b}^{old})^T$$

- **Ακριβής ή προσεγγιστικός υπολογισμός του grad(F). Ακρίβεια vs. Κόστος.**
- **5 διαφορετικοί τρόποι υπολογισμού του grad(F). Κλ. παρακάτω**
- **Το βήμα η στη Μέθοδο της Απότομης Καθόδου. Μονάδες? Τιμή?**
- **Newton or Quasi-Newton: Περί υπολογισμού ή προσέγγισης της Εσσιανής (Hessian)**

Αιτιοκρατική Βελ/ση (Gradient-based Methods)







- **Κίνδυνος εγκλωβισμού σε τοπικά ακρότατα.**
- **Τελική/βέλτιστη λύση εξαρτώμενη από την αρχικοποίηση.**
- **Το «πάν» είναι ο υπολογισμός του Grad(F).**
- **Δευτερεύων (αλλά όχι ασήμαντος) είναι ο τρόπος χρήσης του στη βελ/ση.**

Important

Working with a gradient-based optimization method not only the objective function gradients, but the constraint function gradients should be computed too.

Συγκρίνετέ το με το τι κάνετε στη βελτιστοποίηση με περιορισμούς όταν χρησιμοποιείτε μια στοχαστική μέθοδο (λ.χ. έναν ΕΑ): Μια ποινή στη συνάρτηση-στόχο είναι αρκετή!

Possible Ways to Compute (the exact) Grad(F)

- **Finite Differences (FD) – Πεπερασμένες Διαφορές**
- **Complex Variable (CV) Method – Μέθοδος Συζυγών Μεταβλητών**
- **Direct Differentiation (DD) – Ευθεία Διαφόριση**
- **Automatic or Algorithmic Differentiation (AD) – Αυτόματη ή Αλγοριθμική Διαφόριση**
- **Adjoint Method (AM) – Η Συζυγής Μέθοδος**

The computation of the gradient of the objective and/or constraint functions is the most important and costly part of a gradient-based optimization.

Εκτός από την ακρίβεια υπολογισμού του gradF, το πιο σημαντικό είναι αν το κόστος για να βρείτε το gradF είναι ανάλογο ή ανεξάρτητο του πλήθους των μεταβλητών σχεδιασμού N. Στις συγκρίσεις, ο μηχανικός πάντα θεωρεί ότι θα κριθεί να λύσει ένα πρόβλημα βελ/σης με πολλές μεταβλητές σχεδιασμού, δηλαδή με N>>.

Quasi-Newton Methods (με προσέγγιση του Grad(F))

- **Η Newton είναι πολύ πιο γρήγορη από την απότομη κάθοδο…**
- **Χρειάζεται όμως τον υπολογισμό του Εσσιανού Μητρώου hess(F) της F (συμμετρικό μητρώο NxN δεύτερων παραγώγων). Ακριβό!!!**
- **Με την Quasi-Newton προσεγγίζονται, αντί να υπολογίζονται ακριβώς οι δεύτερες παράγωγοι.**
- **Η προσέγγιση γίνεται με αναδρομικό τύπο, χρησιμοποιώντας την υπολογισμένη κλίση (grad F).**
- **Η συζυγής μέθοδος (ή οτιδήποτε άλλο) χρειάζεται για να βρίσκεται το grad(F) και από αυτό να προσεγγίζεται το hess(F).**
- **Η Quasi-Newton μέθοδος είναι, γενικά, γρήγορη αρκεί να μην είμαστε πολύ μακριά από τη βέλτιστη λύση.**

Quasi-Newton Methods (με προσέγγιση του Grad(F))

Γενική γραφή του διανύσματος διόρθωσης των τιμών των μεταβλητών σχεδιασμού:

$$\overrightarrow{p}^n = -(B^n)^{-1}\nabla F(\overrightarrow{x}^n)$$

$$F(\overrightarrow{x}^n + \overrightarrow{p}^n) \approx F(\overrightarrow{x}^n) + \overrightarrow{p}^{nT}\nabla F(\overrightarrow{x}^n)$$

Επιθυμητός ο μηδενισμός του grad($F(\overrightarrow{x}^n + \overrightarrow{p}^n)$**):**

$$\nabla F(\overrightarrow{x}^n) + \nabla^2 F(\overrightarrow{x}^n)\ \overrightarrow{p}^n = 0$$

$$\overrightarrow{p}^n = -(\nabla^2 F(\overrightarrow{x}^n))^{-1}\ \nabla F(\overrightarrow{x}^n)$$

Quasi-Newton Methods (με προσέγγιση του Grad(F))

$$\nabla^2 F(\overrightarrow{x}^{n+1}) \ (\overrightarrow{x}^{n+1} - \overrightarrow{x}^{n}) \approx \nabla F(\overrightarrow{x}^{n+1}) - \nabla F(\overrightarrow{x}^{n})$$

Μέθοδος SR1 (Symmetric Rank One)

$$B^{n+1} = B^n + \frac{(\overrightarrow{y}^n - B^n \overrightarrow{s}^n) \ (\overrightarrow{y}^n - B^n \overrightarrow{s}^n)^T}{(\overrightarrow{y}^n - B^n \overrightarrow{s}^n)^T \ \overrightarrow{s}^n}$$

Μέθοδος BFGS (Broyden-Fletcher-Goldfarb-Shanno)

$$B^{n+1} = B^n - \frac{B^n \overrightarrow{s}^n \overrightarrow{s}^{nT} B^n}{\overrightarrow{s}^{nT} B^n \overrightarrow{s}^n} + \frac{\overrightarrow{y}^n \overrightarrow{y}^{nT}}{\overrightarrow{y}^{nT} \overrightarrow{s}^n}$$

Quasi-Newton Methods (με προσέγγιση του Grad(F))

Όπου:

$$\overrightarrow{s}^n = \overrightarrow{x}^{n+1} - \overrightarrow{x}^n$$

$$\overrightarrow{y}^n = \nabla F(\overrightarrow{x}^{n+1}) - \nabla F(\overrightarrow{x}^n)$$

<u>Βασικό</u>: ο αναδρομικός τύπος πρέπει να διατηρεί τη <u>συμμετρικότητα</u> του μητρώου!!!!

Βρίσκω το Εσσιανό μητρώο ή τον αντίστροφό του (απευθείας)?

$$H^n = (B^n)^{-1}$$

$$H^{n+1} = (I - \rho^n \overrightarrow{s}^n \overrightarrow{y}^{nT}) H^n (I - \rho^n \overrightarrow{y}^n \overrightarrow{s}^{nT}) + \rho^n \overrightarrow{s}^n \overrightarrow{s}^{nT}$$

$$\rho^n = \frac{1}{\overrightarrow{y}^{nT} \overrightarrow{s}^n}$$

$$\overrightarrow{p}^n = -H^n \nabla F(\overrightarrow{x}^n)$$

… βλ. Βιβλίο μαθήματος

1. Πεπερασμένες Διαφορές – Finite Differences (FD)

$$\frac{\partial F}{\partial b_i} = \frac{F(b_1, b_2, \ldots, b_i + \epsilon, \ldots, b_N) - F(b_1, b_2, \ldots, b_i, \ldots, b_N)}{\epsilon}$$

$$\frac{\partial F}{\partial b_i} = \frac{F(b_1, b_2, \ldots, b_i + \epsilon, \ldots, b_N) - F(b_1, b_2, \ldots, b_i - \epsilon, \ldots, b_N)}{2\epsilon}$$

where J is the objective function. The computational cost for computing the derivatives of the objective function F w.r.t. b_n (n=1,…,N) scales with N. Solution of 2N (2nd order accuracy) or N (1rst order) systems of ODEs for N sensitivity derivatives! Expensive!

- **First-order accuracy, N instead of 2N evaluations per gradient computation.**
- **Parallelization!**
- **Accuracy and the role of ε.**

2. Μέθοδος Μιγαδικών Μεταβλητών – Complex Variable Method (CV)

$$F(b+\delta b) = F(b) + \frac{\delta b}{1!}\frac{dF}{db}\Big|_b + \frac{\delta b^2}{2!}\frac{d^2F}{db^2}\Big|_b + O(\delta b^3)$$

$$F(b+i\delta b) = F(b) + \frac{i\delta b}{1!}\frac{dF}{db}\Big|_b + O(\delta b^2)$$

$$\text{real part}: \quad \text{real}\left[F(b+i\delta b)\right] = F(b) + \cdots$$

$$\text{imag part}: \quad \text{imag}\left[F(b+i\delta b)\right] = \delta b \frac{dF}{db}\Big|_b + \cdots$$

$$\frac{\partial F}{\partial b_i} = \lim_{\epsilon \to 0} \frac{imag(F(b_i + i\epsilon))}{\epsilon}$$

- **N evaluations per gradient computation.**
- **Parallelization!**
- **Accuracy and the role of ε.**

2. Μέθοδος Μιγαδικών Μεταβλητών – Complex Variable Method (CV)

Επίδειξη σε ένα απλό παράδειγμα με γνωστή αναλυτική λύση:

$$F(b_1, b_2) = b_1^2 + 3b_2^4 - 4b_1 b_2$$

$$\frac{\partial F}{\partial b_1} = 2b_1 - 4b_2$$

$$\frac{\partial F}{\partial b_2} = 12b_2^3 - 4b_1$$

$$(b_1, b_2) = (1, 2) \begin{cases} \frac{\partial F}{\partial b_1}(1,2) = -6 \\ \frac{\partial F}{\partial b_2}(1,2) = 92 \end{cases}$$

2. Μέθοδος Μιγαδικών Μεταβλητών – Complex Variable Method (CV)

Κώδικας CV:

```
program demo_complex_variables
implicit double precision (a-h,o-z)
double complex F,x,y
F(x,y) = x**2 + 3.d0*y**4 -4.d0*x*y
epsilon = 1.d-30
b1 = 1.d0
b2 = 2.d0
x=b1*(1.d0,0.d0)+epsilon*(0.d0,1.d0)
y=b2*(1.d0,0.d0)
dFdb1 = imag(F(x,y))/epsilon
x=b1*(1.d0,0.d0)
y=b2*(1.d0,0.d0)+epsilon*(0.d0,1.d0)
dFdb2 = imag(F(x,y))/epsilon
write(*,*)' Computed   dF/db1 = ',dFdb1
write(*,*)' Computed   dF/db2 = ',dFdb2
end
```

Σε οποιαδήποτε γλώσσα προγραμματισμού, οι ΔΙΠΛΗΣ ΑΚΡΙΒΕΙΑ ΜΕΤΑΒΛΗΤΕΣ είναι ΑΠΑΡΑΙΤΗΤΕΣ

2. Μέθοδος Μιγαδικών Μεταβλητών – Complex Variable Method (CV)

Ενώ ο κώδικας σε <u>Πεπερασμένες Διαφορές</u> θα ήταν:

```
program demo_central_diff
implicit double precision (a-h,o-z)
F(x,y) = x**2 + 3.d0*y**4 -4.d0*x*y
epsilon = 1.d-30
b1 = 1.d0
b2 = 2.d0
dFdb1 = (F(b1+epsilon,b2)-F(b1-epsilon,b2))*.5d0/epsilon
dFdb2 = (F(b1,b2+epsilon)-F(b1,b2-epsilon))*.5d0/epsilon
write(*,*)' Computed   dF/db1 = ',dFdb1
write(*,*)' Computed   dF/db2 = ',dFdb2
end
```

Σε οποιαδήποτε γλώσσα προγραμματισμού, οι ΔΙΠΛΗΣ ΑΚΡΙΒΕΙΑ ΜΕΤΑΒΛΗΤΕΣ είναι ΑΠΑΡΑΙΤΗΤΕΣ

2. Μέθοδος Μιγαδικών Μεταβλητών – Complex Variable Method (CV)

CV

ϵ	$\partial F/\partial b_1$	$\partial F/\partial b_2$
10^{-30}	-6.000000000000000	92.000000000000000
10^{-20}	-6.000000000000000	92.000000000000000
10^{-15}	-6.000000000000000	91.999999999999986
10^{-10}	-6.000000000000000	92.000000000000000
10^{-5}	-6.000000000000000	91.999999997600000

FD

ϵ	$\partial F/\partial b_1$	$\partial F/\partial b_2$
10^{-30}	0.000000000000000	0.000000000000000
10^{-20}	0.000000000000000	0.000000000000000
10^{-15}	-7.1054273576010 01	92.370555648813010
10^{-10}	-5.999964969305438	92.000007612114132
10^{-5}	-6.000000000128124	92.000000003267232

Παρατηρήστε πόσο αναίσθητες είναι οι παράγωγοι από τη μέθοδο CV, σε αντίθεση με αυτές από τη μέθοδο FD

3. Μέθοδος Ευθείας Διαφόρισης – Direct Differentiation Method (DD)

Θα παρουσιαστεί ταυτόχρονα με τη συζυγή μέθοδο…

Θα είναι όμως και αυτή μια μέθοδος που το κόστος για τον υπολογισμό του GradF είναι ανάλογο του πλήθους των μεταβλητών σχεδιασμού N, όπω; Συνέβαινε και με τις δύο προηγούμενες μεθόδους (FD και CV).

4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Αφετηρία:

Σκοπός:

Μέσω της αυτόματης επεξεργασίας του παραπάνω λογισμικού με το λογισμικό AD να παραχθεί ένας νέος κώδικας, στην ίδια γλώσσα, ο οποίος εκτός από όσα έκανε ο προηγούμενος να δίνει επιπλέον και τις παραγώγους της συνάρτησης F ως προς τις ελεύθερες μεταβλητές.
Καλό, αλλά δουλεύει πάντα; Είναι τόσο αυτόματο όσο λέει το όνομά του;
Που βρίσκεται αυτό το λογισμικό; Έχει κόστος;
Ποια είναι τα μειονεκτήματά του (Δεν μπορεί!! Θα έχει!!)

4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Δύο σημαντικά διαφορετικοί τρόποι να γίνεται η AD:

Ευθεία ΑΔ
Forward AD

ή

Αντίστροφη ΑΔ
Reverse AD

Υβριδική ΑΔ
Hybrid AD

<u>Συσχέτιση</u>: Διακριτή Συζυγής Μέθοδος / Discrete Adjoint Method και DD
(θα το καταλάβετε αργότερα, όταν γνωρίσετε αυτές τις μεθόδους!)

4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Διαδοχή τελέσεων κατά την υλοποίηση της αξιολόγησης:

Ανεξάρτητες Μεταβλητές

Ενδιάμεσες Εξαρτημένες Μεταβλητές

$$
\begin{aligned}
b_{N+1} &= f_{N+1}\left(b_1, \ldots, b_N\right) \\
b_{N+2} &= f_{N+2}\left(b_1, \ldots, b_{N+1}\right) \\
&\vdots \\
b_m &= f_m\left(b_1, \ldots, b_{m-1}\right) \\
F &= b_m
\end{aligned}
$$

4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Ευθεία ΑΔ – Forward AD:

$$
\begin{aligned}
b_{N+1} &= f_{N+1}\left(b_1, \ldots, b_N\right) \\
\nabla b_{N+1} &= \sum_{i=1}^{N} \frac{\partial f_{N+1}}{\partial b_i} \overrightarrow{e}_i \\
b_{N+2} &= f_{N+2}\left(b_1, \ldots, b_{N+1}\right) \\
\nabla b_{N+2} &= \sum_{i=1}^{N} \frac{\partial f_{N+2}}{\partial b_i} \overrightarrow{e}_i + \frac{\partial f_{N+2}}{\partial b_{N+1}} \nabla b_{N+1} \\
&\vdots \\
b_m &= f_m\left(b_1, \ldots, b_{m-1}\right) \\
\nabla b_m &= \sum_{i=1}^{N} \frac{\partial f_m}{\partial b_i} \overrightarrow{e}_i + \sum_{i=N+1}^{m-1} \frac{\partial f_m}{\partial b_i} \nabla b_i \\
F &\equiv b_m \\
\nabla F &\equiv \nabla b_m
\end{aligned}
$$

Με δαπάνη αρκετής μνήμης!!!

4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Ευθεία ΑΔ – Forward AD – ΠΑΡΑΔΕΙΓΜΑ:

$$F = b_1 b_2 b_3 b_4$$

$$\Re^4 \rightarrow \Re$$

Ενδιάμεσες Εξαρτημένες Μεταβλητές:

$$\begin{aligned}
b_5 &= f_5(b_1, b_2, b_3, b_4) = b_1 \\
b_6 &= f_6(b_1, b_2, b_3, b_4, b_5) = b_2 b_5 \\
b_7 &= f_7(b_1, b_2, b_3, b_4, b_5, b_6) = b_3 b_6 \\
b_8 &= f_7(b_1, b_2, b_3, b_4, b_5, b_6, b_7) = b_4 b_7 \\
F &= b_8
\end{aligned}$$

$$\begin{aligned}
b_5 &= b_1 \\
\nabla b_5 &= \overrightarrow{e}_1 \\
b_6 &= b_2 b_5 \\
\nabla b_6 &= b_5 \overrightarrow{e}_2 + b_2 \nabla b_5 \\
b_7 &= b_3 b_6 \\
\nabla b_7 &= b_6 \overrightarrow{e}_3 + b_3 \nabla b_6 \\
b_8 &= b_4 b_7 \\
\nabla b_8 &= b_7 \overrightarrow{e}_4 + b_4 \nabla b_7 \\
F &= b_8 \\
\nabla F &= \nabla b_8
\end{aligned}$$

4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Αντίστροφη ΑΔ – Reverse AD:

Ορίζονται: $$\bar{b}_i = \frac{\partial b_m}{\partial b_i} \qquad (i = N+1, \ldots, m)$$ **Εξ ορισμού:** $$\bar{b}_m \equiv 1$$

Υπολογίζονται:

$$\bar{b}_i = \sum_{j=i+1}^{m} \frac{\partial b_m}{\partial b_j} \frac{\partial f_j}{\partial b_i} = \sum_{j=i+1}^{m} \bar{b}_j \frac{\partial f_j}{\partial b_i} \;, \quad \underbrace{i = N+1, \ldots, m-1}$$

Εξαρτημένες Μεταβλητές

$$\bar{b}_i = \sum_{j=N+1}^{m} \frac{\partial b_m}{\partial b_j} \frac{\partial f_j}{\partial b_i} = \sum_{j=N+1}^{m} \bar{b}_j \frac{\partial f_j}{\partial b_i} \;, \quad \underbrace{i = 1, \ldots, N}$$

Ανεξάρτητες Μεταβλητές

4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Αντίστροφη ΑΔ – Reverse AD:

1. Προς τα Εμπρός (κανονική) Σάρωση:

$$b_i = f_i\left(b_1, \ldots, b_{i-1}\right) \ , \quad i = N+1, \ldots, m$$

$$\bar{b}_i = 0 \ , \quad i = 1, \ldots, m-1$$

$$\bar{b}_m = 1$$

2. Προς τα Πίσω (ανάποδη) Σάρωση:

$$do\ j = m, N+1, -1$$

$$do\ i = 1, j-1$$

$$\bar{b}_i = \bar{b}_i + \bar{b}_j \frac{\partial f_j}{\partial b_i}$$

$$enddo$$

$$enddo$$

4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Αντίστροφη ΑΔ – Reverse AD- Παράδειγμα:

$$F = b_1 b_2 b_3$$

$$\Re^3 \longrightarrow \Re$$

$$\bar{b}_6 = \frac{\partial f_6}{\partial b_6} \equiv 1$$

$$\bar{b}_5 = \frac{\partial f_6}{\partial b_6}\frac{\partial f_6}{\partial b_5} = \bar{b}_6 \frac{\partial f_6}{\partial b_5}$$

$$\bar{b}_4 = \frac{\partial f_6}{\partial b_6}\frac{\partial f_6}{\partial b_4} + \frac{\partial f_6}{\partial b_5}\frac{\partial f_5}{\partial b_4} = \bar{b}_6 \frac{\partial f_6}{\partial b_4} + \bar{b}_5 \frac{\partial f_5}{\partial b_4}$$

$$\bar{b}_3 = \frac{\partial f_6}{\partial b_6}\frac{\partial f_6}{\partial b_3} + \frac{\partial f_6}{\partial b_5}\frac{\partial f_5}{\partial b_3} + \frac{\partial f_6}{\partial b_4}\frac{\partial f_4}{\partial b_3} = \bar{b}_6 \frac{\partial f_6}{\partial b_3} + \bar{b}_5 \frac{\partial f_5}{\partial b_3} + \bar{b}_4 \frac{\partial f_4}{\partial b_3}$$

$$\bar{b}_2 = \frac{\partial f_6}{\partial b_6}\frac{\partial f_6}{\partial b_2} + \frac{\partial f_6}{\partial b_5}\frac{\partial f_5}{\partial b_2} + \frac{\partial f_6}{\partial b_4}\frac{\partial f_4}{\partial b_2} = \bar{b}_6 \frac{\partial f_6}{\partial b_2} + \bar{b}_5 \frac{\partial f_5}{\partial b_2} + \bar{b}_4 \frac{\partial f_4}{\partial b_2}$$

$$\bar{b}_1 = \frac{\partial f_6}{\partial b_6}\frac{\partial f_6}{\partial b_1} + \frac{\partial f_6}{\partial b_5}\frac{\partial f_5}{\partial b_1} + \frac{\partial f_6}{\partial b_4}\frac{\partial f_4}{\partial b_1} = \bar{b}_6 \frac{\partial f_6}{\partial b_1} + \bar{b}_5 \frac{\partial f_5}{\partial b_1} + \bar{b}_4 \frac{\partial f_4}{\partial b_1}$$

Ενδιάμεσες Εξαρτημένες Μεταβλητές:

$$b_4 = f_4(b_1, b_2, b_3) = b_1$$

$$b_5 = f_5(b_1, b_2, b_3, b_4) = b_2 b_4$$

$$b_6 = f_6(b_1, b_2, b_3, b_4, b_5) = b_3 b_5$$

$$F = b_6$$

$$\frac{\partial f_3}{\partial b_1} = \frac{\partial f_3}{\partial b_2} = \frac{\partial f_2}{\partial b_1}$$

4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Αντίστροφη ΑΔ – Reverse AD- Παράδειγμα:

1 $$\bar{b}_6 \equiv 1 \ , \quad \bar{b}_5 = 0 \ , \quad \bar{b}_4 = 0 \ , \quad \bar{b}_3 = 0 \ , \quad \bar{b}_2 = 0 \ , \quad \bar{b}_1 = 0$$

2 $$f_6 = b_3 b_5 \qquad \frac{\partial f_6}{\partial b_5} = b_3 \ , \quad \frac{\partial f_6}{\partial b_4} = 0 \ , \quad \frac{\partial f_6}{\partial b_3} = b_5 \ , \quad \frac{\partial f_6}{\partial b_2} = 0 \ , \quad \frac{\partial f_6}{\partial b_1} = 0$$

Σ: $$\bar{b}_5 = b_3 \ , \quad \bar{b}_4 = 0 \ , \quad \bar{b}_3 = b_5 \ , \quad \bar{b}_2 = 0 \ , \quad \bar{b}_1 = 0$$

3 $$f_5 = b_2 b_4 \qquad \frac{\partial f_5}{\partial b_4} = b_2 \ , \quad \frac{\partial f_5}{\partial b_3} = 0 \ , \quad \frac{\partial f_5}{\partial b_2} = b_4 \ , \quad \frac{\partial f_5}{\partial b_1} = 0$$

Σ: $$\bar{b}_4 = b_2 b_3 \ , \quad \bar{b}_3 = b_5 \ , \quad \bar{b}_2 = b_3 b_4 \ , \quad \bar{b}_1 = 0$$

4 $$f_4 = b_1 \qquad \frac{\partial f_4}{\partial b_3} = 0 \ , \quad \frac{\partial f_4}{\partial b_2} = 0 \ , \quad \frac{\partial f_4}{\partial b_1} = 1$$

Σ: $$\bar{b}_3 = b_5 \ , \quad \bar{b}_2 = b_3 b_4 \ , \quad \bar{b}_1 = b_2 b_3$$

4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Αντίστροφη ΑΔ – Reverse AD- Παράδειγμα:

Τελική Συνάθροιση Όρων Προς τα Εμπρός:

$$
\begin{aligned}
b_4 &= b_1 \\
\nabla b_4 &= \overrightarrow{e}_1 \\
b_5 &= b_2 b_4 \\
\nabla b_5 &= b_4 \overrightarrow{e}_2 + b_2 \nabla b_4 = b_4 \overrightarrow{e}_2 + b_2 \overrightarrow{e}_1 \\
b_6 &= b_3 b_5 \\
\nabla b_6 &= b_5 \overrightarrow{e}_3 + b_3 \nabla b_5 = b_5 \overrightarrow{e}_3 + b_3 b_4 \overrightarrow{e}_2 + b_3 b_2 \overrightarrow{e}_1
\end{aligned}
$$

4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Software:

ADIFOR (Automatic Differentiation of Fortran)
AMC (Tangent linear and Adjoint Model Compiler}
TAF (Transformation of Algorithms in Fortran)
DAFOR (Differential Algebraic Extension of Fortran)
GRESS (Gradient--Enhanced Software System)
Odyssee
TAPENADE
AD01
ADOL-F (Automatic Differentiation of FORTRAN Codes)
IMAS (Integrated Modeling and Analysis System)
OPTIMA90
ADIC (Automatic Differentiation of C Programs)
ADOL-C (Automatic Differentiation of Algorithms written in C/C++)

4. AD – Using TAPENADE (σε Forward Mode)

```
C      Generated by TAPENADE     (INRIA, Tropics team)
C  Differentiation of ff in forward (tangent) mode:
C
      SUBROUTINE FF_D(x1, x1d, x2, x2d, x3, x3d, f1, f1d, f2, f2d)
      IMPLICIT NONE
      DOUBLE PRECISION x1, x2, x3, x1d, x2d, x3d, f1, f2, f1d, f2d
C
      f1d = x1d + x2d + x3d
      f1 = x1 + x2 + x3
      f2d = (x1d*x2+x1*x2d)*x3 + x1*x2*x3d
      f2 = x1*x2*x3
C
      RETURN
      END
```

```
subroutine ff(x1,x2,x3,f1,f2)
implicit double precision(a-h,o-z)
f1 = x1+x2+x3
f2 = x1*x2*x3
return
end
```

4. AD – Using TAPENADE (σε Forward Mode)

```
C      Generated by TAPENADE     (INRIA, Tropics team)
C  Differentiation of ff in forward (tangent) mode:
C
      SUBROUTINE FF_D(x1, x1d, x2, x2d, x3, x3d, f1, f1d, f2, f2d)
      IMPLICIT NONE
      DOUBLE PRECISION x1, x2, x3, x1d, x2d, x3d, f1, f2, f1d, f2d
C
      f1d = x1d + x2d + x3d
      f1 = x1 + x2 + x3
      f2d = (x1d*x2+x1*x2d)*x3 + x1*x2*x3d
      f2 = x1*x2*x3
C
      RETURN
      END
```

Η ρουτίνα θα κληθεί τόσες φορές όσο το πλήθος των μεταβλητών σχεδιασμού (εδώ 3: x1, x2, x3). Κάθε κλήση γίνεται με: (x1d,x2d,x3d)=(1,0,0),(0,1,0) και (0,0,1) Σε κάθε μια από αυτές τις κλήσεις υπολογίζονται οι παράγωγοι όλων των συναρτήσεων (εδώ f1 και f2) ως προς την αντίστοιχη μεταβλητή που δείχνει η «1»

4. AD – Using TAPENADE (σε Reverse Mode)

```
C      Generated by TAPENADE     (INRIA, Tropics team)
C  Differentiation of ff in reverse(adjoint) mode:
C
      SUBROUTINE FF_B(x1, x1b, x2, x2b, x3, x3b, f1, f1b, f2, f2b)
      IMPLICIT NONE
      DOUBLE PRECISION x1, x2, x3, x1b, x2b, x3b, f1, f2, f1b, f2b
      x1b = f1b + x3*x2*f2b
      x2b = f1b + x3*x1*f2b
      x3b = f1b + x1*x2*f2b
      f1b = 0.D0
      f2b = 0.D0

      RETURN
      END
```

```
subroutine ff(x1,x2,x3,f1,f2)
implicit double precision(a-h,o-z)
f1 = x1+x2+x3
f2 = x1*x2*x3
return
end
```

4. AD – Using TAPENADE (σε Reverse Mode)

```
C      Generated by TAPENADE     (INRIA, Tropics team)
C  Differentiation of ff in reverse(adjoint) mode:
C
      SUBROUTINE FF_B(x1, x1b, x2, x2b, x3, x3b, f1, f1b, f2, f2b)
      IMPLICIT NONE
      DOUBLE PRECISION x1, x2, x3, x1b, x2b, x3b, f1, f2, f1b, f2b
      x1b = f1b + x3*x2*f2b
      x2b = f1b + x3*x1*f2b
      x3b = f1b + x1*x2*f2b
      f1b = 0.D0
      f2b = 0.D0

      RETURN
      END
```

Η ρουτίνα θα κληθεί τόσες φορές όσο το πλήθος των συναρτήσεων που παραγωγίζονται (εδώ 2: f1, f2). Κάθε κλήση γίνεται με: (f1b,f2b)=(1,0) και (0,1)
Σε κάθε μια από αυτές τις κλήσεις υπολογίζονται οι παράγωγοι της συνάρτησης που δείχνει η «1» ως προς όλες τις μεταβλητές (x1,x2,x3).

4. AD – Using TAPENADE (σε Forward Mode) – C Code

```
void ff_d(double x1, double x1d, double x2, double x2d, double x3, double x3d,  double *f1,
double *f1d, double *f2, double *f2d)      //function produced by TAPENADE - forward
(tangent) mode
{
/*       Generated by TAPENADE     (INRIA, Ecuador team)
   Tapenade 3.16 (develop) -  3 Jan 2023 19:02
*/
/*
  Differentiation of ff in forward (tangent) mode:
   variations   of useful results: *f1 *f2
   with respect to varying inputs: x1 x2 x3
   RW status of diff variables: f1:(loc) *f1:out f2:(loc) *f2:out
              x1:in x2:in x3:in    Plus diff mem management of: f1:in f2:in
*/

   *f1d = x1d + x2d + x3d;
   *f1 = x1 + x2 + x3;
   *f2d = x3*(x2*x1d+x1*x2d) + x1*x2*x3d;
   *f2 = x1*x2*x3;
}
```

```
#include <stdio.h>

void ff(double x1, double x2, double x3, double* f1,
double* f2)                //functions to be differentiated
{
   *f1 = x1 + x2 + x3;     //1st function
   *f2 = x1 * x2 * x3;     //2nd function
}
```

Δείτε οδηγίες στα προηγούμενα slides (παράδειγμα Fortran)

4. AD – Using TAPENADE (σε Reverse Mode) – C Code

```
void ff_b(double x1, double *x1b, double x2, double *x2b, double x3, double *x3b, double
*f1, double *f1b, double *f2, double *f2b)    //function produced by TAPENADE - reverse
(adjoint) mode
{
/*
  Differentiation of ff in reverse (adjoint) mode:
   gradient     of useful results: *f1 *f2
   with respect to varying inputs: *f1 *f2 x1 x2 x3
   RW status of diff variables: f1:(loc) *f1:in-out f2:(loc) *f2:in-out
                x1:out x2:out x3:out
   Plus diff mem management of: f1:in f2:in
*/
    *x1b = x2*x3*(*f2b) + *f1b;
    *x2b = x1*x3*(*f2b) + *f1b;
    *x3b = x1*x2*(*f2b) + *f1b;
    *f2b = 0.0;
    *f1b = 0.0;
}
```

Δείτε οδηγίες στα προηγούμενα slides (παράδειγμα Fortran)

```
#include <stdio.h>

void ff(double x1, double x2, double x3, double* f1,
double* f2)                  //functions to be differentiated
{
   *f1 = x1 + x2 + x3;     //1st function
   *f2 = x1 * x2 * x3;     //2nd function
}
```

4. AD – Using TAPENADE (Main) – C Code – Part 1

```
int main()
{
   int i;
   double x1,x2,x3;
   double f1,f2,f1d,f2d,f1b,f2b;
   double x1b,x2b,x3b;

   //point where the derivatives are calculated
   x1 = 2.0;
   x2 = 3.0;
   x3 = 1.0;

   printf("Computation of the derivatives of the functions f1 = x1 + x2 + x3 and f2 =
x1*x2*x3 w.r.t. x1,x2,x3 \n");
   printf("at x1 = %f, x2 = %f, x3 = %f \n\n",x1,x2,x3);

   ff(x1, x2, x3, &f1, &f2);
   printf("f1 = %f \n",f1);
   printf("f2 = %f \n",f2);
```

4. AD – Using TAPENADE (Main) – C Code – Part 2

```
printf("\nAnalytical Differentiation:\n");
printf("f1_x1 = %f, f1_x2 = %f, f1_x3 = %f\n",1.0,1.0,1.0);
printf("f2_x1 = %f, f2_x2 = %f, f2_x3 = %f\n",x2*x3, x1*x3, x1*x2);

printf("\nForward (tangent) AD:\n");
ff_d(x1, 1.0, x2, 0.0, x3, 0.0,  &f1, &f1d, &f2, &f2d);
printf("f1_x1 = %f,\tf2_x1 = %f\n",f1d,f2d);
ff_d(x1, 0.0, x2, 1.0, x3, 0.0,  &f1, &f1d, &f2, &f2d);
printf("f1_x2 = %f,\tf2_x2 = %f\n",f1d,f2d);
ff_d(x1, 0.0, x2, 0.0, x3, 1.0,  &f1, &f1d, &f2, &f2d);
printf("f1_x3 = %f,\tf2_x3 = %f\n",f1d,f2d);
```

4. AD – Using TAPENADE (Main) – C Code – Part 3

```
    printf("\nReverse (adjoint) AD:\n");
    f1b = 1.0;
    f2b = 0.0;
    ff_b(x1, &x1b, x2, &x2b, x3, &x3b, &f1, &f1b, &f2, &f2b);
    printf("f1_x1 = %f,\tf1_x2 = %f,\tf1_x3 = %f\n",x1b,x2b,x3b);
    f1b = 0.0;
    f2b = 1.0;
    ff_b(x1, &x1b, x2, &x2b, x3, &x3b, &f1, &f1b, &f2, &f2b);
    printf("f2_x1 = %f,\tf2_x2 = %f,\tf2_x3 = %f\n",x1b,x2b,x3b);

    return 0;
}
```

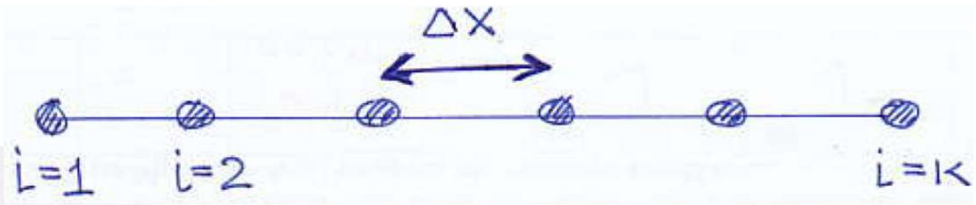
4. Αλγοριθμική ή Αυτόματη Διαφόριση – Automatic Differentiation (AD)

Σύνοψη:

- **Αν θέλετε να χρησιμοποιήσετε AD πρέπει ο κώδικας σας να είναι σε μια από τις γλώσσες προγραμματισμού τις οποίες υποστηρίζει η AD. Μπορεί να απαιτηθεί αναδόμηση σημαντικού τμήματος του λογισμικού σας που δεν είναι «αρεστά» στο λογισμικό AD.**
- **Το προϊόν/αποτέλεσμα της AD αναμένεται να έχει μεγάλες απαιτήσεις μνήμης (για κώδικες λχ που λύνουν φυσικά προβλήματα, επαναληπτικά, λχ ένας επιλύτης εξισώσεων ροής κλπ).**
- **Χρησιμοποιείστε AD σε forward mode αν διαφορίζετε πιο πολλές συναρτήσεις ως πιο λίγες μεταβλητές σχεδιασμού (οι κλήσεις στο λογισμικό είναιόσες οι μεταβλητές σχεδιασμού). Είναι ως να εφαρμόζετε Direct Differentiation (Ευθεία Διαφόριση).**
- **Χρησιμοποιείστε AD σε reverse mode αν διαφορίζετε πιο λίγες συναρτήσεις ως πιο πολλές μεταβλητές σχεδιασμού (κλήσεις όσες οι συναρτήσεις που διαφορίζονται). Είναι ως να εφαρμόζετε τη Συζυγή (Adjoint) Μέθοδο.**
- **Συνιστώμενο δωρεάν λογισμικό για εξάρτηση το TAPENADE από την INRIA (Γαλλία).**

3. Μέθοδος Ευθείας Διαφόρισης – Direct Differentiation Method (DD)

<u>Ένα 1Δ παράδειγμα κατανόησης:</u>



$$\frac{d^2v}{dx^2} + b_1 v + b_2 v^2 - 5 = 0$$

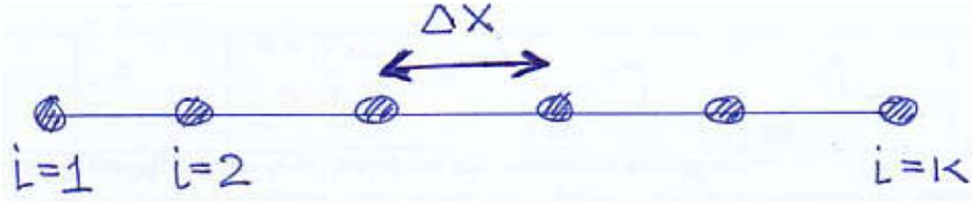
στο $\emptyset \le x \le 1$

οριακές συνθήκες: $v(\emptyset)=1$, $v(1)=3$

- **Είναι πρόβλημα γραμμικό ή μη-γραμμικό;**
- **Γιατί έχει οριακές συνθήκες και στα δύο άκρα;**
- **Η αριθμητική του επίλυση θέλει επαναλήψεις ή μπορεί να λυθεί «με τη μία»;**

3. Μέθοδος Ευθείας Διαφόρισης – Direct Differentiation Method (DD)

<u>Πρωτεύον Πρόβλημα:</u> Διακριτοποίηση του χωρίου (γραμμής) με Ν ισαπέχοντες κόμβους



Διακριτοποίηση της ΣΔΕ με Πεπερασμένες Διαφορές (Γραμμικοποίηση!!!)

- Για $2 \le i \le K-1$:

$$\frac{v_{i+1}^{NEW} - 2v_i^{NEW} + v_{i-1}^{NEW}}{\Delta x^2} + b_1 v_i^{NEW} + b_2 v_i^{OLD} v_i^{NEW} - 5 = \emptyset$$

ή

$$\left[\frac{1}{\Delta x^2}\right] v_{i+1} + \left[b_1 + b_2 v_i^{OLD} - \frac{2}{\Delta x^2}\right] v_i + \left[\frac{1}{\Delta x^2}\right] v_{i-1} = 5$$

- $i=1$: $v_1^{NEW} = 1$
- $i=K$: $v_K^{NEW} = 3$

3. Μέθοδος Ευθείας Διαφόρισης – Direct Differentiation Method (DD)

<u>Πρωτεύον Πρόβλημα:</u> Διακριτοποίηση με τη Δέλτα Διατύπωση

$$v_i^{new} = v_i^{old} + \Delta v_i$$

$$\frac{d^2(v^{old}+\Delta v)}{dx^2} + b_1(v^{old}+\Delta v) + b_2(v^{old^2} + 2v^{old}\Delta v^{old}) - 5 = \emptyset$$

$$\frac{d^2\Delta v}{dx^2} + b_1\Delta v + 2b_2 v^{old}\Delta v = 5 - \frac{d^2 v^{old}}{dx^2} - b_1 v^{old} - b_2 v^{old^2}$$

3. Μέθοδος Ευθείας Διαφόρισης – Direct Differentiation Method (DD)

<u>Πρωτεύον Πρόβλημα:</u> Διακριτοποίηση με τη Δέλτα Διατύπωση

$$\left[\frac{1}{\Delta x^2}\right]\Delta v_{i+1} + \left[b_1 + 2b_2 v_i^{old} - \frac{2}{\Delta x^2}\right]\Delta v_i + \left[\frac{1}{\Delta x^2}\right]\Delta v_{i-1} =$$

$$= 5 - \frac{v_{i+1}^{old} - 2v_i^{old} + v_{i-1}^{old}}{\Delta x^2} - b_1 v_i^{old} - b_2 v_i^{old\,2}$$

Συμβολίζοντας:

$$a = 1/\Delta x^2$$

$$\gamma_i = b_1 + 2b_2 v_i^{old} - \frac{2}{\Delta x^2} = b_1 + 2b_2 v_i^{old} - 2a$$

3. Μέθοδος Ευθείας Διαφόρισης – Direct Differentiation Method (DD)

<u>Πρωτεύον Πρόβλημα:</u> Διακριτοποίηση με τη Δέλτα Διατύπωση (αν N=6 κόμβοι)

$$\begin{bmatrix} 1 & & & & & \\ a & \gamma_2 & a & & & \\ & a & \gamma_3 & a & & \\ & & a & \gamma_4 & a & \\ & & & a & \gamma_5 & a \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} \Delta U_1 \\ \Delta U_2 \\ \Delta U_3 \\ \Delta U_4 \\ \Delta U_5 \\ \Delta U_6 \end{bmatrix} = \begin{bmatrix} \emptyset \\ R_2^{old} \\ R_3^{old} \\ R_4^{old} \\ R_5^{old} \\ \emptyset \end{bmatrix}$$

3. Μέθοδος Ευθείας Διαφόρισης – Direct Differentiation Method (DD)

<u>Χρήση:</u> Έστω η (προς ελαχιστοποίηση) συνάρτηση κόστους

$$F = \int_{\varnothing}^{1} v(x)\,dx \cong \sum_{i=1}^{K} v_i \,\Delta x \;(\text{or } \Delta x/2)$$

Θα είναι

$$\frac{\delta F}{\delta b_1} = \int_{\varnothing}^{1} \frac{\delta v}{\delta b_1}\,dx \cong \sum_{i=1}^{K} \frac{\delta v_i}{\delta b_1}\,\Delta x \qquad \frac{\delta F}{\delta b_2} = \int_{\varnothing}^{1} \frac{\delta v}{\delta b_2}\,dx \cong \sum_{i=1}^{K} \frac{\delta v_i}{\delta b_2}\,\Delta x$$

Οπότε χρειάζονται τα πεδία των $\delta v/\delta b_1$ και $\delta v/\delta b_2$.

Βελ/ση με τη μέθοδο της απότομης καθόδου (steepest descent method):

$$b_1^{NEW} = b_1^{OLD} - \eta \frac{\delta F}{\delta b_1}\left(b_1^{OLD}, b_2^{OLD}\right)$$

$$b_2^{NEW} = b_2^{OLD} - \eta \frac{\delta F}{\delta b_2}\left(b_1^{OLD}, b_2^{OLD}\right)$$

3. Μέθοδος Ευθείας Διαφόρισης – Direct Differentiation Method (DD)

Πρώτη ΣΔΕ:

$$\frac{d^2}{dx^2}\left(\frac{\delta v}{\delta b_1}\right) + v + b_1\frac{\delta v}{\delta b_1} + 2b_2 v\frac{\delta v}{\delta b_1} = 0$$

στο $0 \le x \le 1$

οριακές συνθήκες: $\frac{\delta v}{\delta b_1}(0) = 0$, $\frac{\delta v}{\delta b_1}(1) = 0$

Δεύτερη ΣΔΕ:

$$\frac{d^2}{dx^2}\left(\frac{\delta v}{\delta b_2}\right) + b_1\frac{\delta v}{\delta b_2} + v^2 + 2b_2 v\frac{\delta v}{\delta b_2} = 0$$

στο $0 \le x \le 1$

οριακές συνθήκες: $\frac{\delta v}{\delta b_2}(0) = 0$, $\frac{\delta v}{\delta b_2}(1) = 0$

DD στη συνεχή/continuous εκδοχή του:
Πρέπει να λυθούν Ν εξισώσεις DD. Συμφέρει;

5. Η Συζυγής Μέθοδος – Adjoint Method

Continuous Adjoint Method:

1. Differentiate the primal equations (PDEs). Derive adjoint equations in the form of PDEs.

2. Discretize & solve the primal and adjoint equations (PDEs).

Discrete Adjoint Method:

1. Discretize the primal equations (PDEs) and differentiate them in discrete form.

2. Derive the adjoint equations in discrete form (linear system) and solve them.

Adjoint is the art of computing the derivatives of J (or F) w.r.t. b_n (n=1,…,N) without first computing the fields of the derivatives of the primal variables w.r.t. b_n. The adjoint method makes the cost of computing the gradient of J independent of N!

Εισαγωγή στη Διακριτή Συζυγή Μέθοδο – Discrete Adjoint Method

Objective Function: Minimize

$$F = F(\vec{U}, \vec{b})$$

or

$$F = F(\vec{U}(\vec{b}), \vec{b})$$

Constraint: Subject to

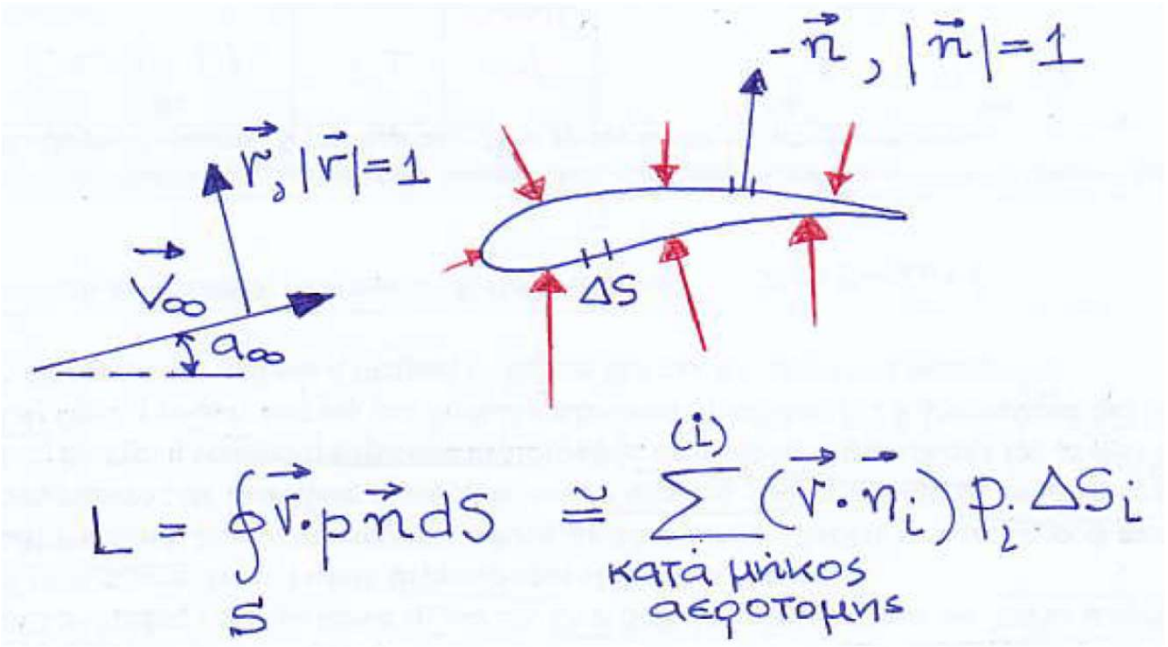
$$\vec{R} = \vec{R}(\vec{U}, \vec{b}) = 0$$

or

$$\vec{R} = \vec{R}(\vec{U}(\vec{b}), \vec{b}) = 0$$

Εξηγήσεις για τις Εξαρτήσεις

$$F = F(\vec{U}(\vec{b}), \vec{b})$$

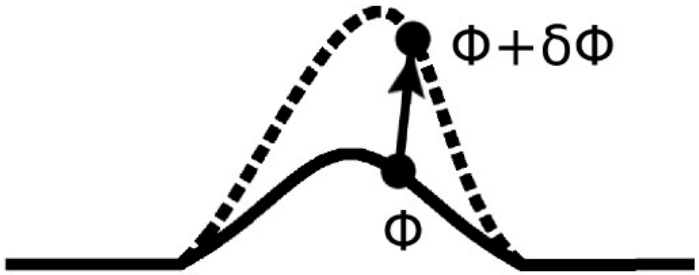


Εισαγωγή στη Διακριτή Συζυγή Μέθοδο – Discrete Adjoint Method

Or, compute $\frac{\delta F}{\delta \vec{b}} = \frac{\partial F}{\partial \vec{b}} + \frac{\partial F}{\partial \vec{U}} \frac{\delta \vec{U}}{\delta \vec{b}}$ **(since** $F = F(\vec{U}(\vec{b}), \vec{b})$ **)**

Subject to $\frac{\delta \vec{R}}{\delta \vec{b}} = \frac{\partial \vec{R}}{\partial \vec{b}} + \frac{\partial \vec{R}}{\partial \vec{U}} \frac{\delta \vec{U}}{\delta \vec{b}} = 0$ **(since** $\vec{R} = \vec{R}(\vec{U}(\vec{b}), \vec{b}) = 0$ **)**

IMPORTANT: Total vs. Partial Derivatives:



$$F = \int_S p\vec{n} \cdot \vec{a}_\infty dS = \sum p_i \vec{n}_i \cdot \vec{a}_\infty \Delta S_i$$

Εισαγωγή στη Διακριτή Συζυγή Μέθοδο – Discrete Adjoint Method

Τι «κρύβει» το:

$$\frac{\delta F}{\delta \vec{b}} = \frac{\partial F}{\partial \vec{b}} + \frac{\partial F}{\partial \vec{U}} \frac{\delta \vec{U}}{\delta \vec{b}}$$

$$\frac{\delta F}{\delta \vec{b}} \quad , \quad \frac{\partial F}{\partial \vec{b}} = \underbrace{\boxed{}\boxed{\cdots}\boxed{}}_{\leftarrow N \rightarrow} \updownarrow 1 \qquad \frac{\partial F}{\partial \vec{U}} = \underbrace{\boxed{}\boxed{}\boxed{}\boxed{\cdots}\boxed{}\boxed{}\boxed{}}_{\leftarrow \quad \kappa \quad \rightarrow} \updownarrow 1$$

$$\frac{\delta \vec{U}}{\delta \vec{b}} = \begin{array}{|c|c|c|} \hline & \cdots & \\ \hline & \cdots & \\ \hline & \cdots & \\ \hline & \vdots & \\ \hline & \cdots & \\ \hline & \cdots & \\ \hline & \cdots & \\ \hline \end{array} \begin{array}{l} \uparrow \\ \\ \\ \kappa \\ \\ \\ \downarrow \end{array}$$
$$\leftarrow N \rightarrow$$

Αν μπούμε στον κόπο (ουσιαστικά στο κόστος) να υπολογίσουμε τις μεταβολές των πρωτευουσών μεταβλητών ως προς τις μεταβλητές σχεδιασμού (δηλ. το αριστερά κxN μητρώο) τότε κάνουμε Ευθεία Διαφόριση (DD) και το υπολογιστικό κόστος είναι ανάλογο του πλήθους N των μεταβλητών σχεδιασμού.

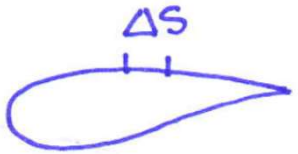
Αυτό θα αποφύγουμε με τη Συζυγή Μέθοδο

Εισαγωγή στη Διακριτή Συζυγή Μέθοδο – Discrete Adjoint Method

Ένα πολύ απλό παράδειγμα για το τι μπορεί να είναι τα:

$$\frac{\delta F}{\delta \vec{b}} \quad , \quad \frac{\partial F}{\partial \vec{b}} = \underbrace{\square\square\cdots\square}_{\leftarrow N \rightarrow} \updownarrow 1 \qquad \frac{\partial F}{\partial \vec{U}} = \underbrace{\square\square\square\cdots\square\square\square}_{\leftarrow \quad \kappa \quad \rightarrow} \updownarrow 1$$

$$F = \oint_{\text{airfoil}} p\,ds = \sum p_i \Delta S_i$$



$$\frac{\partial F}{\partial U_\lambda} \text{ εστω } \frac{\partial F}{\partial p_\lambda} = \Delta S_\lambda$$

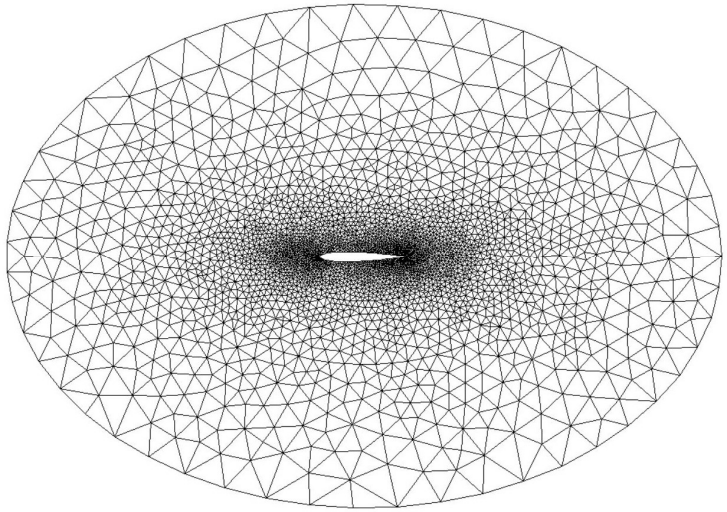
$$\frac{\partial F}{\partial b_n} = \sum p_i \frac{\partial (\Delta S_i)}{\partial b_n}$$

Εισαγωγή στη Διακριτή Συζυγή Μέθοδο – Discrete Adjoint Method

Διευκρινήσεις για το τι «κρύβει» το μητρώο:

$$\frac{\delta \vec{U}}{\delta \vec{b}} = \begin{array}{|c|c|c|} \hline & \cdots & \\ \hline & \cdots & \\ \hline & \cdots & \\ \hline & \vdots & \\ \hline & \cdots & \\ \hline & \cdots & \\ \hline & \cdots & \\ \hline \end{array} \begin{array}{l} \uparrow \\ \\ \\ \kappa \\ \\ \\ \downarrow \end{array}$$

$\leftarrow N \rightarrow$



$$\vec{U} = \begin{array}{|c|} \hline \\ \hline \\ \hline \vdots \\ \hline \\ \hline \\ \hline \end{array} \begin{array}{l} \uparrow \\ \\ \\ \kappa \\ \\ \\ \downarrow \end{array}$$

1

2D Flow Problem,
3469 nodes, 6787 triangles

κ=3469x3 για 2Δ ασυμπίεστη ροή, ή κ=3469x4 για 2Δ συμπιεστή ροή (τι γίνεται αν 3Δ?). Ποιες οι (πρωτεύουσες) μεταβλητές ροής κατά περίπτωση; Τρόπος αποθήκευσης στο U?

Εισαγωγή στη Διακριτή Συζυγή Μέθοδο – Discrete Adjoint Method

Τι «κρύβει» το:

$$\frac{\delta \vec{R}}{\delta \vec{b}} = \frac{\partial \vec{R}}{\partial \vec{b}} + \frac{\partial \vec{R}}{\partial \vec{U}} \frac{\delta \vec{U}}{\delta \vec{b}} = 0$$

$$\frac{\delta \vec{R}}{\delta \vec{b}}, \frac{\partial \vec{R}}{\partial \vec{b}} = \begin{array}{|ccc|} \hline & \cdots & \\ & \cdots & \\ & \cdots & \\ & \vdots & \\ & \cdots & \\ & \cdots & \\ & \cdots & \\ \hline \end{array} \begin{array}{l} \uparrow \\ \\ \\ \kappa \\ \\ \\ \downarrow \end{array}$$

$$\leftarrow N \rightarrow$$

$$\frac{\partial \vec{R}}{\partial \vec{U}} = A = \begin{array}{|ccccccc|} \hline & & & \cdots & & & \\ & & & \cdots & & & \\ & & & \cdots & & & \\ & & & \vdots & & & \\ & & & \cdots & & & \\ & & & \cdots & & & \\ & & & \cdots & & & \\ \hline \end{array} \begin{array}{l} \uparrow \\ \\ \\ \kappa \\ \\ \\ \downarrow \end{array}$$

$$\leftarrow \quad \kappa \quad \rightarrow$$

A= Ιακωβιανή ορίζουσα (Jacobian) των υπολοίπων των πρωτευουσών εξισώσεων

Εισαγωγή στη Διακριτή Συζυγή Μέθοδο – Discrete Adjoint Method

Που έχουμε ξαναδεί την Ιακωβιανή ορίζουσα (A):

Solve the state/primal equations $\vec{R}(\vec{U}) = 0$

through an iterative method (linearization, delta formulation …)

$$\vec{R}(\vec{U}^{n+1}) = 0$$

$$\vec{R}(\vec{U}^{n+1}) = \vec{R}(\vec{U}^{n}) + \left.\frac{\partial \vec{R}}{\partial \vec{U}}\right|_{\vec{U}^n} (\vec{U}^{n+1} - \vec{U}n) = \vec{R}(\vec{U}^{n}) + A|_{\vec{U}^n}\Delta\vec{U}$$

or

$$A|_{\vec{U}^n}\Delta\vec{U} = -\vec{R}(\vec{U}^{n}) \quad \Rightarrow \quad \vec{U}^{n+1} = \vec{U}^{n} + \Delta\vec{U}$$

Numerics **Physics**

Πως θα το λύναμε με το DD?

(1) Επίλυση των N εξισώσεων DD, από την

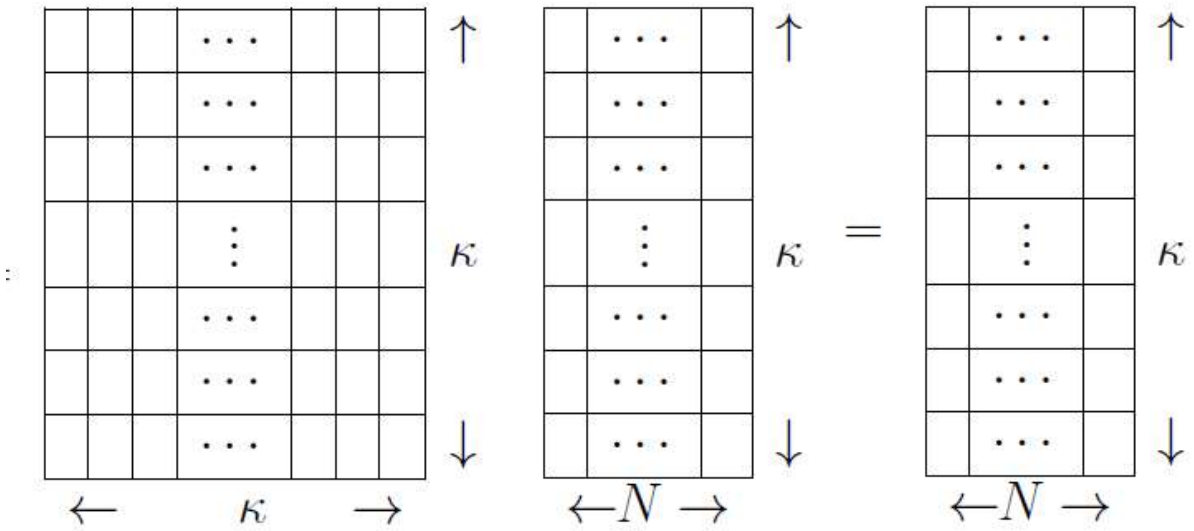
$$\frac{\delta \vec{R}}{\delta \vec{b}} = \frac{\partial \vec{R}}{\partial \vec{b}} + \frac{\partial \vec{R}}{\partial \vec{U}} \frac{\delta \vec{U}}{\delta \vec{b}} = 0$$

δηλαδή των

$$\frac{\partial \vec{R}}{\partial \vec{U}} \frac{\delta \vec{U}}{\delta \vec{b}} = - \frac{\partial \vec{R}}{\partial \vec{b}}$$

(1) Αντικατάσταση:

$$\frac{\delta F}{\delta \vec{b}} = \frac{\partial F}{\partial \vec{b}} + \frac{\partial F}{\partial \vec{U}} \frac{\delta \vec{U}}{\delta \vec{b}}$$



Είναι μια συμπαγής μορφή παρουσίασης N ανεξάρτητων γραμμικών συστημάτων κxκ

Εισαγωγή στη Διακριτή Συζυγή Μέθοδο – Discrete Adjoint Method

Υπενθύμιση Στόχου:

Να αποφύγουμε το κόστος υπολογισμού των μεταβολών των πρωτευουσών μεταβλητών ως προς τις μεταβλητές σχεδιασμού (δηλ. το δεξιά κxN μητρώο), δηλ. να μην κάνουμε Ευθεία Διαφόριση (DD). Αυτό γίνεται με τη Συζυγή Μέθοδο (Adjoint Method), είτε στη διακριτή (discrete adjoint) είτε στη συνεχή (continuous adjoint) εκδοχή της.

$$\frac{\delta \vec{U}}{\delta \vec{b}} = \begin{bmatrix} & \cdots & \\ & \cdots & \\ & \cdots & \\ & \vdots & \\ & \cdots & \\ & \cdots & \\ & \cdots & \end{bmatrix} \begin{matrix} \uparrow \\ \\ \\ \kappa \\ \\ \\ \downarrow \end{matrix}$$

$$\leftarrow N \rightarrow$$

Discrete Adjoint

$$\frac{\delta \vec{R}}{\delta \vec{b}} = \frac{\partial \vec{R}}{\partial \vec{b}} + \frac{\partial \vec{R}}{\partial \vec{U}} \frac{\delta \vec{U}}{\delta \vec{b}} = 0 \Rightarrow \frac{\delta \vec{U}}{\delta \vec{b}} = -\left(\frac{\partial \vec{R}}{\partial \vec{U}}\right)^{-1} \frac{\partial \vec{R}}{\partial \vec{b}}$$

$$\frac{\delta F}{\delta \vec{b}} = \frac{\partial F}{\partial \vec{b}} + \frac{\partial F}{\partial \vec{U}} \frac{\delta \vec{U}}{\delta \vec{b}} = \frac{\partial F}{\partial \vec{b}} - \frac{\partial F}{\partial \vec{U}} \left(\frac{\partial \vec{R}}{\partial \vec{U}}\right)^{-1} \frac{\partial \vec{R}}{\partial \vec{b}}$$

Ορίζουμε: $\frac{\partial F}{\partial \vec{U}} \left(\frac{\partial \vec{R}}{\partial \vec{U}}\right)^{-1} = \vec{\Psi}^T \Rightarrow \left(\frac{\partial \vec{R}}{\partial \vec{U}}\right)^T \vec{\Psi} = \left(\frac{\partial F}{\partial \vec{U}}\right)^T$

Field Adjoint Equations (FAE) in discrete form

Sensitivity Derivatives (SD) $\frac{\delta F}{\delta \vec{b}} = \frac{\partial F}{\partial \vec{b}} - \vec{\Psi}^T \frac{\partial \vec{R}}{\partial \vec{b}}$

Η Διακριτή Συζυγής Μέθοδος – Discrete Adjoint Method

Εναλλακτικά: Η Διακριτή Συζυγής Μέθοδος αν γραφεί σε μορφή μεταβολών (variations) αντί παραγώγων (derivatives)

$$\delta J = \frac{\partial J}{\partial \vec{U}} \delta \vec{U} + \frac{\partial J}{\partial \vec{b}} \delta \vec{b} \qquad \delta R = \frac{\partial \vec{R}}{\partial \vec{U}} \delta \vec{U} + \frac{\partial \vec{R}}{\partial \vec{b}} \delta \vec{b} = 0$$

$$\delta J_{aug} = \left(\frac{\partial J}{\partial \vec{U}} - \vec{\Psi}^T \frac{\partial \vec{R}}{\partial \vec{U}} \right) \delta \vec{U} + \left(\frac{\partial J}{\partial \vec{b}} - \vec{\Psi}^T \frac{\partial \vec{R}}{\partial \vec{b}} \right) \delta \vec{b}$$

$$\left[\frac{\partial \vec{R}}{\partial \vec{U}} \right]^T \vec{\Psi} = \left[\frac{\partial J}{\partial \vec{U}} \right]^T$$

Field Adjoint Equations (FAE) in discrete form

$$\frac{\partial J}{\partial \vec{b}} = -\vec{\Psi}^T \frac{\partial \vec{R}}{\partial \vec{b}} + \frac{\partial F}{\partial \vec{b}}$$

Sensitivity Derivatives (SD)

Discrete Adjoint

Διατύπωση της Διακριτής Συζυγούς Μεθόδου με άλλο τρόπο:

Augmented Objective Function or Lagrangean:

$$F_{aug} = F - \vec{\Psi}^T \vec{R}$$

$$\frac{\delta F_{aug}}{\delta \vec{b}} = \frac{\delta F}{\delta \vec{b}} - \vec{\Psi}^T \frac{\delta \vec{R}}{\delta \vec{b}}$$

$$\frac{\delta F_{aug}}{\delta \vec{b}} = \frac{\partial F}{\partial \vec{b}} + \frac{\partial F}{\partial \vec{U}} \frac{\delta \vec{U}}{\delta \vec{b}} - \vec{\Psi}^T \left(\frac{\partial \vec{R}}{\partial \vec{b}} + \frac{\partial \vec{R}}{\partial \vec{U}} \frac{\delta \vec{U}}{\delta \vec{b}} \right)$$

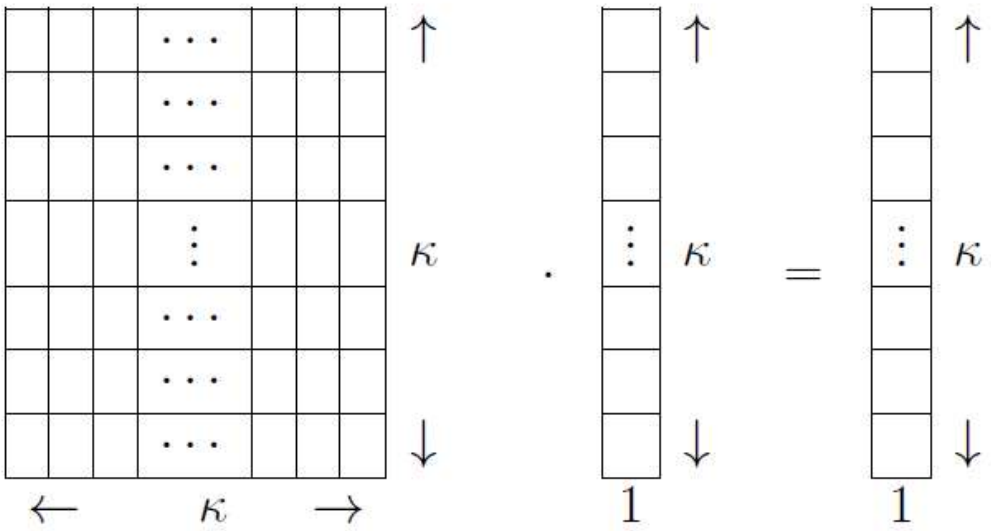
$$\frac{\delta F_{aug}}{\delta \vec{b}} = \frac{\partial F}{\partial \vec{b}} - \vec{\Psi}^T \frac{\partial \vec{R}}{\partial \vec{b}} + \left(\frac{\partial F}{\partial \vec{U}} - \vec{\Psi}^T \frac{\partial \vec{R}}{\partial \vec{U}} \right) \frac{\delta \vec{U}}{\delta \vec{b}}$$

$$\frac{\delta F}{\delta \vec{b}} = \frac{\partial F}{\partial \vec{b}} - \vec{\Psi}^T \frac{\partial \vec{R}}{\partial \vec{b}}$$

$$\left(\frac{\partial \vec{R}}{\partial \vec{U}} \right)^T \vec{\Psi} = \left(\frac{\partial F}{\partial \vec{U}} \right)^T$$

Understanding the Discrete Adjoint Equation

$$\left(\frac{\partial \vec{R}}{\partial \vec{U}}\right)^T \vec{\Psi} = \left(\frac{\partial F}{\partial \vec{U}}\right)^T$$



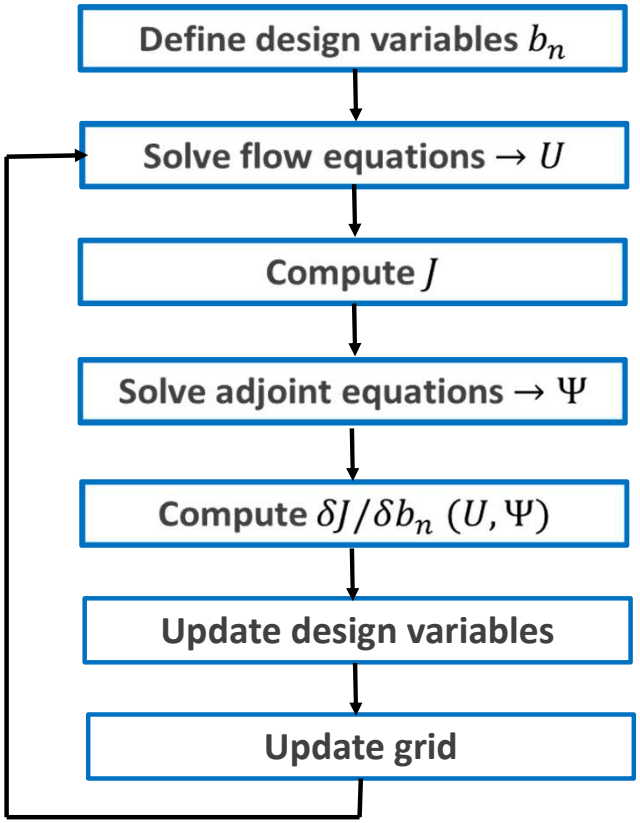
Understanding the Discrete Adjoint Equation

$$\frac{\delta F}{\delta \vec{b}} = \frac{\partial F}{\partial \vec{b}} - \vec{\Psi}^T \frac{\partial \vec{R}}{\partial \vec{b}}$$

$$\delta F = \left(-\vec{\Psi}^T \frac{\partial \vec{R}}{\partial \vec{b}} + \frac{\partial F}{\partial \vec{b}} \right) \delta \vec{b}$$

$$\delta F = \left[\underbrace{\square\square\square\cdots\square\square\square}_{\leftarrow\ \kappa\ \rightarrow} \updownarrow 1 \cdot \begin{array}{c} \square\cdots\square \uparrow \\ \square\cdots\square \\ \square\cdots\square \\ \square\ \vdots\ \square \quad \kappa \\ \square\cdots\square \\ \square\cdots\square \\ \square\cdots\square \downarrow \end{array} \underbrace{\qquad}_{\leftarrow N \rightarrow} + \underbrace{\square\cdots\square}_{\leftarrow N \rightarrow} \updownarrow 1 \right] \begin{array}{c} \square \uparrow \\ \vdots \quad N \\ \square \downarrow \end{array} \underbrace{\qquad}_{1}$$

The Adjoint-Based Optimization Loop



Κόστος: Μια Μονάδα Χρόνου (Time Unit)

Κόστος: (περίπου άλλη) Μια Μονάδα Χρόνου

Με τη Συζυγή Μέθοδο, το υπολογιστικό κόστος ανά κύκλο βελτιστοποίησης (optimization cycle) είναι 2 μονάδες χρόνου, ΑΝΕΞΑΡΤΗΤΩΣ ΤΟΥ ΠΛΗΘΟΥΣ Ν ΤΩΝ ΜΕΤΑΒΛΗΤΩΝ ΣΧΕΔΙΑΣΜΟΥ

Διακριτή Συζυγής Μέθοδος– Discrete Adjoint

Στο ίδιο 1Δ παράδειγμα κατανόησης με το DD:

$$\frac{d^2v}{dx^2} + b_1 v + b_2 v^2 - 5 = 0$$

στο $\varnothing \le x \le 1$

οριακές συνθήκες: $v(\varnothing)=1$, $v(1)=3$

Ακολουθεί παράδειγμα με 6 κόμβους (Δx=0.2 αν L=1)

$$R_i = \frac{v_{i+1} - 2v_i + v_{i-1}}{\Delta x^2} + b_1 v_i + b_2 v_i^2 - 5 = \varnothing \quad , \; 2 \le i \le 5$$

$$R_1 = v_1 - 1 = \varnothing$$

$$R_6 = v_6 - 3 = \varnothing$$

$$R_i = K v_{i-1} + b_1 v_i + b_2 v_i^2 - 2K v_i + K v_{i+1} - 5 \neq \phi$$

where $K = \frac{1}{\Delta x^2}$

Διακριτή Συζυγής Μέθοδος– Discrete Adjoint

Στο ίδιο 1Δ παράδειγμα κατανόησης με το DD:

Συνάρτηση στόχος σε διακριτή μορφή:

OBJECTIVE FUNCTION:

$$F = \int_0^1 v\,dx = \frac{\Delta x}{2} v_1 + \sum_{i=2}^{5} v_i \Delta x + \frac{\Delta x}{2} v_6$$

$$\frac{\partial F}{\partial v_i} = \Delta x \quad \text{for } 2 \le i \le 5$$

$$\frac{\partial F}{\partial v_1} = \frac{\partial F}{\partial v_6} = \frac{\Delta x}{2}.$$

$$\frac{\partial F}{\partial \vec{b}} = \begin{bmatrix} \emptyset & \emptyset \end{bmatrix} \quad \text{since} \quad \frac{\partial F}{\partial b_1} = \frac{\partial F}{\partial b_2} = \emptyset$$

Διακριτή Συζυγής Μέθοδος– Discrete Adjoint

Στο ίδιο 1Δ παράδειγμα κατανόησης με το DD:

$$R_i = K U_{i-1} + b_1 U_i + b_2 U_i^2 - 2K U_i + K U_{i+1} - 5 \neq \phi$$

where $K = \frac{1}{\Delta x^2}$

$$\frac{\partial R_i}{\partial U_{i-1}} = K$$

$$\frac{\partial R_i}{\partial U_i} = b_1 + 2 b_2 U_i - 2K$$

$$\frac{\partial R_i}{\partial U_{i+1}} = K$$

STATE PROBLEM: → Matrix A

$$\begin{bmatrix} 1 & & & & & \\ K & \mu_2 & K & & & \\ & K & \mu_3 & K & & \\ & & K & \mu_4 & K & \\ & & & K & \mu_5 & K \\ & & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} \Delta U_1 \\ \Delta U_2 \\ \Delta U_3 \\ \Delta U_4 \\ \Delta U_5 \\ \Delta U_6 \end{bmatrix} = \begin{bmatrix} \phi \\ -R_2^{old} \\ -R_3^{old} \\ -R_4^{old} \\ -R_5^{old} \\ \phi \end{bmatrix}$$

Adjoint Equation:

$$\left(\frac{\partial \vec{R}}{\partial \vec{U}}\right)^T \vec{\Psi} = \left(\frac{\partial F}{\partial \vec{U}}\right)^T$$

or $A^T \vec{\Psi} = \left(\frac{\partial F}{\partial \vec{U}}\right)^T$

$$A^T \cdot \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \\ \Psi_4 \\ \Psi_5 \\ \Psi_6 \end{bmatrix} = \begin{bmatrix} \Delta x/2 \\ \Delta x \\ \Delta x \\ \Delta x \\ \Delta x \\ \Delta x/2 \end{bmatrix}$$

Διακριτή Συζυγής Μέθοδος– Discrete Adjoint

Στο ίδιο 1Δ παράδειγμα κατανόησης με το DD:

$$R_i = \kappa U_{i-1} + b_1 U_i + b_2 U_i^2 - 2\kappa U_i + \kappa U_{i+1} - 5 \neq \phi$$

where $\kappa = \frac{1}{\Delta x^2}$

$$\frac{\partial R_1}{\partial b_1} = \frac{\partial R_1}{\partial b_2} = \phi$$

$$\frac{\partial R_6}{\partial b_1} = \frac{\partial R_6}{\partial b_2} = \phi$$

$$\frac{\partial R_i}{\partial b_1} = U_i \qquad \frac{\partial R_i}{\partial b_2} = U_i^2$$

$2 \leq i \leq 5$

Sensitivity Derivatives (SD):
Παράγωγοι Ευαισθησίας

$$\frac{\delta F}{\delta \vec{b}} = \frac{\partial F}{\partial \vec{b}} - \vec{\Psi}^T \frac{\partial \vec{R}}{\partial \vec{b}}$$

$$\left[\frac{\delta F}{\delta b_1} \quad \frac{\delta F}{\delta b_2}\right]^T = \left[\phi \quad \phi\right]^T - \left[\Psi_1 \Psi_2 \Psi_3 \Psi_4 \Psi_5 \Psi_6\right] \cdot \begin{bmatrix} \phi & \phi \\ U_2 & U_2^2 \\ U_3 & U_3^2 \\ U_4 & U_4^2 \\ U_5 & U_5^2 \\ \phi & \phi \end{bmatrix}$$

Συνεχής Συζυγής Μέθοδος– Continuous Adjoint

Στο ίδιο 1Δ παράδειγμα κατανόησης με το DD:

$$F_{AUG} = F + \int_{\varnothing}^{1} \psi \left(\frac{d^2U}{dx^2} + b_1 U + b_2 U^2 - 5 \right) dx \Rightarrow$$

$$\Rightarrow \delta F_{AUG} = \delta \int_{\varnothing}^{1} U dx + \int_{\varnothing}^{1} \psi \, \delta \left(\frac{d^2U}{dx^2} + b_1 U + b_2 U^2 - 5 \right) dx \Rightarrow$$

$$\Rightarrow \delta F_{AUG} = \int_{\varnothing}^{1} \delta U dx + \underbrace{\int_{\varnothing}^{1} \psi \frac{d^2 \delta U}{dx^2} dx}_{T1} + \underbrace{\int_{\varnothing}^{1} \psi \delta (b_1 U) dx}_{T2} + \underbrace{\int_{\varnothing}^{1} \psi \delta (b_2 U^2) dx}_{T3}$$

Συνεχής Συζυγής Μέθοδος– Continuous Adjoint

Στο ίδιο 1Δ παράδειγμα κατανόησης με το DD:

$$(T_1): \int_0^1 \psi \frac{d^2(\delta v)}{dx^2} dx = \int_0^1 \frac{d}{dx}\left(\psi \frac{d\delta v}{dx}\right) dx - \int_0^1 \frac{d\psi}{dx}\frac{d\delta v}{dx} dx =$$

$$= \left[\psi \frac{d\delta v}{dx}\right]_0^1 - \int_0^1 \frac{d}{dx}\left(\delta v \frac{d\psi}{dx}\right) dx + \int_0^1 \frac{d^2\psi}{dx^2}\delta v\, dx =$$

$$= \left[\psi\, \delta\left(\frac{dv}{dx}\right)\right]_0^1 - \left[\delta v \frac{d\psi}{dx}\right]_0^1 + \int_0^1 \delta v \frac{d^2\psi}{dx^2} dx.$$

$$(T_2): \int_0^1 \psi \delta(b_1 v)\, dx = b_1 \int_0^1 \psi \delta v\, dx + \int_0^1 \psi v \delta b_1\, dx.$$

$$(T_3): \int_0^1 \psi \delta(b_2 v^2)\, dx = 2b_2 \int_0^1 \psi v \delta v\, dx + \int_0^1 \psi v^2 \delta b_2\, dx$$

Συνεχής Συζυγής Μέθοδος– Continuous Adjoint

Στο ίδιο 1Δ παράδειγμα κατανόησης με το DD:

$$\Rightarrow \delta F_{AVG} = \int_{\varnothing}^{1} \delta U \left[1 + \frac{d^2\psi}{dx^2} + b_1\psi + 2b_2 U\psi \right] dx +$$

$$+ \left[\psi\, \delta\left(\frac{dU}{dx}\right) \right]_{\varnothing}^{1} + \int_{\varnothing}^{1} \psi U \delta b_1 dx + \int_{\varnothing}^{1} \psi U^2 \delta b_2 dx.$$

Field Adjoint Equation (FAE):
Πεδιακή Συζυγής Εξίσωση

$$\frac{d^2\psi}{dx^2} + b_1\psi + 2b_2 U\psi + 1 = 0$$

Adjoint Boundary Conditions (ABC):
Συζυγείς Οριακές Συνθήκες

$$\psi(x=\varnothing) = \psi(x=1) = \varnothing$$

Συνεχής Συζυγής Μέθοδος– Continuous Adjoint

Στο ίδιο 1Δ παράδειγμα κατανόησης με το DD:

$$\Rightarrow \delta F_{AUG} = \int_0^1 \delta U \left[1 + \frac{d^2\psi}{dx^2} + b_1\psi + 2b_2 U\psi\right] dx +$$

$$+ \left[\psi\,\delta\left(\frac{dU}{dx}\right)\right]_0^1 + \int_0^1 \psi U \delta b_1 dx + \int_0^1 \psi U^2 \delta b_2 dx.$$

Sensitivity Derivatives (SD):
Παράγωγοι Ευαισθησίας

$$\frac{\delta F}{\delta b_1} = \int_0^1 \psi U dx$$

$$\frac{\delta F}{\delta b_2} = \int_0^1 \psi U^2 dx$$

Discrete Adjoint for other than Optimization purposes

Compute $F = \vec{g}^T \vec{U}, \qquad \vec{g}, \vec{U} \in R^N$

where $A\vec{U} = \vec{q}, \qquad \vec{q} \in R^N, \quad A \in R^{N \times N}$

Direct Computation

Step 1: $A\vec{U} = \vec{q} \longrightarrow \vec{U}$

Step 2: $F = \vec{g}^T \vec{U} \longrightarrow F$

Adjoint Computation

Step 1: $A^T \vec{\Psi} = \vec{g} \longrightarrow \vec{\Psi}$

Step 2: $F = \vec{\Psi}^T \vec{q} \longrightarrow F$

Their Equivalence:

$$\vec{\Psi}^T \vec{q} = \vec{\Psi}^T \left(A\vec{U} \right) = \vec{\Psi}^T A \vec{U} = \left(A^T \vec{\Psi} \right)^T \vec{U} = \vec{g}^T \vec{U}$$

Benefits:

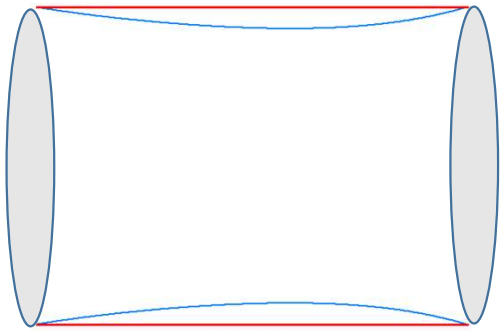
Assume we have m vectors $\vec{q}$ and n vectors $\vec{g}$.

Use the Direct Computation if m<n.

Use the Adjoint Computation if m>n.

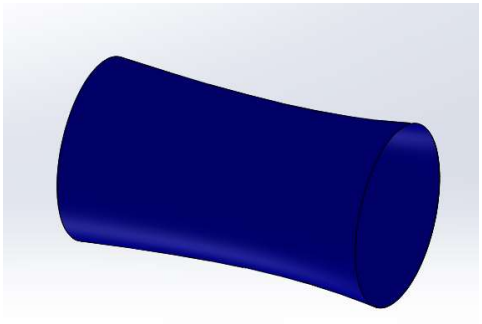
Inverse Design of a Quasi-1D Duct – Το Πρόβλημα

Cross Section

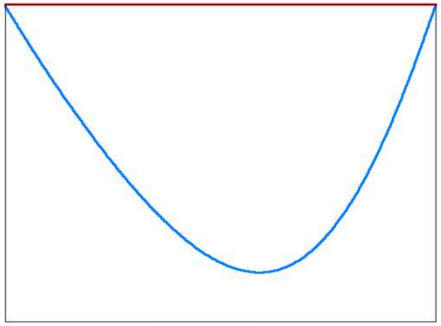


Target Duct

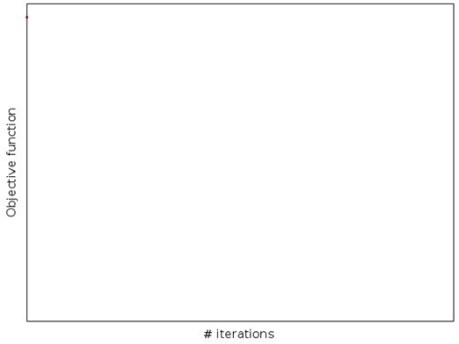
$$F = \frac{1}{2}\int_0^1 \left(p(x) - \underbrace{p_{tar}(x)}_{given}\right)^2 dx$$



Pressure



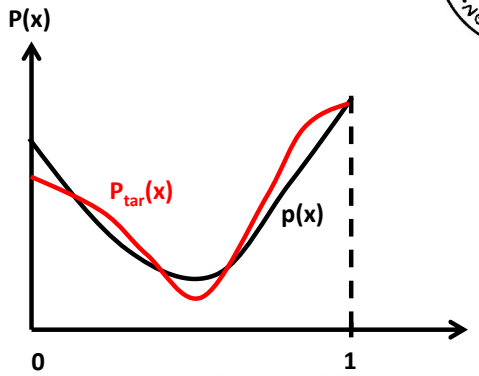
Convergence History



Inverse Design of a Quasi-1D Duct – Incompressible Flow

Objective Function (min.):

$$J = \frac{1}{2}\int_0^1 \left(p(x) - p_{tar}(x)\right)^2 dx$$



Shape (Cross-Sectional Area) Parameterization:

$$S(x, b_1, b_2, b_3, b_4) = b_1\left(-x^3+3x^2-3x+1\right)+b_2\left(3x^3-6x^2+3x\right)+b_3\left(-3x^3+3x^2\right)+b_4x^3$$

Design/Optimization Variables:

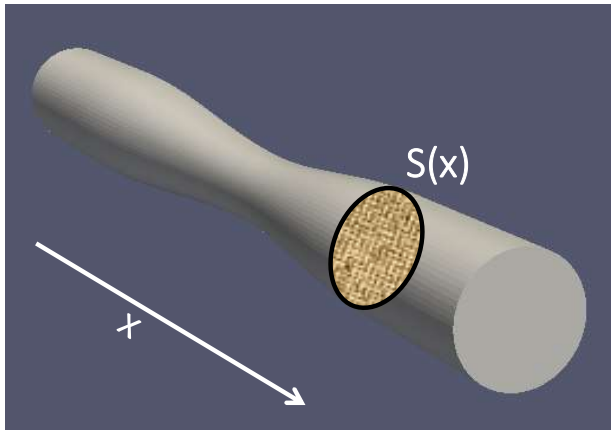
$$b_n,\ n=1,2,3,4$$

State/Primal/Flow Equations (ODEs):

$$\frac{d(vS)}{dx} = 0 \qquad v\frac{dv}{dx} + \frac{dp}{dx} = 0$$

State/Primal/Flow Boundary Conditions:

$$v|_{x=0} = v_0 \qquad p|_{x=1} = p_0$$



Inverse Design of a Quasi-1D Duct – Incompressible Flow

Differentiation of the objective function:

$$\frac{\delta J}{\delta b_n} = \int_0^1 \left(p(x) - p_{tar}(x)\right) \frac{\delta p}{\delta b_n} dx, \;\; n = 1, \ldots, N$$

Differentiation of the flow (primal or state) equations (non-conservative):

$$\left.\begin{aligned} S\frac{d}{dx}\left(\frac{\delta v}{\delta b_n}\right) + \frac{dS}{dx}\frac{\delta v}{\delta b_n} + \frac{dv}{dx}\frac{\delta S}{\delta b_n} + v\frac{d}{dx}\left(\frac{\delta S}{\delta b_n}\right) = 0 \\ \frac{\delta v}{\delta b_n}\frac{dv}{dx} + v\frac{d}{dx}\left(\frac{\delta v}{\delta b_n}\right) + \frac{d}{dx}\left(\frac{\delta p}{\delta b_n}\right) = 0 \end{aligned}\right\} \; n = 1, \ldots, N$$

Differentiation of the flow (primal or state) boundary conditions:

$$\left.\frac{\delta v}{\delta b_n}\right|_{x=0} = 0, \quad \left.\frac{\delta p}{\delta b_n}\right|_{x=1} = 0$$

DD: Direct Differentiation

Adjoint is the art of computing $\frac{\delta J}{\delta b_n}$ without first computing $\frac{\delta v}{\delta b_n}$ and $\frac{\delta p}{\delta b_n}$.

Inverse Design of a Quasi-1D Duct – Incompressible Flow

Define & differentiate the augmented objective function or Lagrangian of J:

$$L = J + \int_0^1 q \frac{d(vS)}{dx} dx + \int_0^1 u \left[v \frac{dv}{dx} + \frac{dp}{dx} \right] dx$$

$$\frac{\delta L}{\delta b_n} = \frac{\delta J}{\delta b_n} + \int_0^1 q \frac{d}{dx} \left[\frac{\delta (vS)}{\delta b_n} \right] dx + \int_0^1 u \left[\frac{\delta v}{\delta b_n} \frac{dv}{dx} + v \frac{d}{dx} \left(\frac{\delta v}{\delta b_n} \right) + \frac{d}{dx} \left(\frac{\delta p}{\delta b_n} \right) \right] dx$$

where q and u are the adjoint pressure and velocity (1D) fields.
Integrate by parts:

$$\frac{\delta L}{\delta b_n} = -\int_0^1 \left[-u \frac{dv}{dx} + \frac{d(vu)}{dx} + S \frac{dq}{dx} \right] \frac{\delta v}{\delta b_n} dx + \int_0^1 \left(-\frac{du}{dx} + p - p_{tar} \right) \frac{\delta p}{\delta b_n} dx$$
$$- \int_0^1 v \frac{dq}{dx} \frac{\delta S}{\delta b_n} dx + \left[(vu + qS) \frac{\delta v}{\delta b_n} \right]_{x=0}^{x=1} + \left[u \frac{\delta p}{\delta b_n} \right]_{x=0}^{x=1} + \left[vq \frac{\delta S}{\delta b_n} \right]_{x=0}^{x=1}$$

Inverse Design of a Quasi-1D Duct – Incompressible Flow

Adjoint field equations:

$$\frac{du}{dx} = p - p_{tar} \qquad u\frac{dv}{dx} - \frac{d(vu)}{dx} - S\frac{dq}{dx} = 0$$

Adjoint boundary conditions:

$$u|_{x=0} = 0, \quad q|_{x=1} = -\left.\frac{vu}{S}\right|_{x=1}$$

Compare with the primal problem equations & boundary conditions:

$$\frac{d(vS)}{dx} = 0 \qquad v\frac{dv}{dx} + \frac{dp}{dx} = 0$$

$$v|_{x=0} = v_0 \qquad p|_{x=1} = p_0$$

Inverse Design of a Quasi-1D Duct – Incompressible Flow

Sensitivity derivatives:

$$\frac{\delta L}{\delta b_n}=\frac{\delta J}{\delta b_n}=-\int_0^1 v\frac{dq}{dx}\frac{\delta S}{\delta b_n}dx+\left.vq\frac{\delta S}{\delta b_n}\right|_{x=1}-\left.vq\frac{\delta S}{\delta b_n}\right|_{x=0},\quad n=1,\ldots,4$$

or:

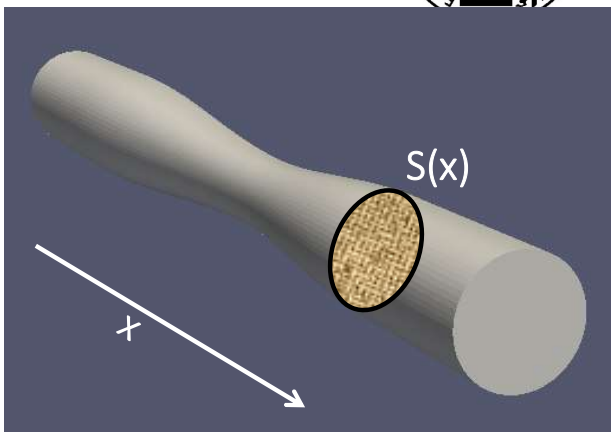
$$\frac{\delta J}{\delta b_1}=-\int_0^1 v\frac{dq}{dx}\left(-x^3+3x^2-3x+1\right)dx-\left.vq\right|_{x=0}$$

$$\frac{\delta J}{\delta b_2}=-\int_0^1 v\frac{dq}{dx}\left(3x^3-6x^2+3x\right)dx$$

$$\frac{\delta J}{\delta b_3}=-\int_0^1 v\frac{dq}{dx}\left(-3x^3+3x^2\right)dx$$

$$\frac{\delta J}{\delta b_4}=-\int_0^1 v\frac{dq}{dx}x^3dx+\left.vq\right|_{x=1}$$

Inverse Design of a Quasi-1D Duct – Compressible Flow



State/Primal/Flow Equations (ODEs):

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{f}}{\partial x} = \vec{q} \qquad \frac{\partial \vec{U}}{\partial t} + A\frac{\partial \vec{U}}{\partial x} = \vec{q}$$

where:

$$\vec{U} = [\rho \ \rho u \ \rho E]^T$$

$$\vec{f} = [\rho u \ \ \rho u^2 + p \ \ u(\rho E + p)]^T$$

$$\vec{q} = -\frac{1}{S}\frac{dS}{dx}[\rho u \ \rho u^2 \ u(\rho E + p)]^T$$

$$A = \frac{\partial \vec{f}}{\partial \vec{U}}$$

Inverse Design of a Quasi-1D Duct – Compressible Flow

Variation in the objective function J:

$$\delta J(\vec{U},\vec{b}) = \int_L (p - p_{tar}) \frac{\partial p}{\partial \vec{U}} \delta \vec{U} \, dx \ , \qquad \frac{\partial p}{\partial \vec{U}} = (\gamma - 1)\left[1/2u^2 \ \ -u \ \ 1\right]$$

Direct Differentiation (DD) of the flow (primal or state) equations & boundary conditions

$$\frac{\partial(\delta \vec{U})}{\partial t} + \frac{\partial(\delta \vec{f})}{\partial x} = \delta(\vec{q})$$

at a cost that scales with the number of design variables N. Solution of N systems of ODEs (see next slide)! Expensive!

Adjoint is the art of computing the derivatives of J w.r.t. b_n (n=1,…,N) without first computing the fields of the derivatives of the primal variables w.r.t. b_n. The adjoint method makes the cost of computing the gradient of J independent of N!

Inverse Design of a Quasi-1D Duct – Compressible Flow

The Direct Differentiation (DD) systems of ODEs, in full expansion:
(by omitting the pseudo-time derivative term)

$$\frac{\partial}{\partial x}(u\delta\rho + \rho\delta u) = \delta(-\frac{1}{S}\frac{dS}{dx})(u\delta\rho + \rho\delta u)$$

$$\frac{\partial}{\partial x}(u^2\delta\rho + 2\rho u\delta u + \delta p) = \delta(-\frac{1}{S}\frac{dS}{dx})(u^2\delta\rho + 2\rho u\delta u)$$

$$\frac{\partial}{\partial x}([\rho E+p]\,\delta u + u[E\delta\rho + \rho(C_v\delta T + u\delta u) + \delta p]) =$$
$$\delta(-\frac{1}{S}\frac{dS}{dx})([\rho E+p]\,\delta u + u[E\delta\rho + \rho(C_v\delta T + u\delta u) + \delta p])$$

Inverse Design of a Quasi-1D Duct – Compressible Flow

Define & differentiate the augmented objective function or Lagrangian of J:

$$\delta J_{aug} = \delta J - \int_L \vec{\Psi}^T \left(\frac{\partial(\delta \vec{f})}{\partial x} - \delta \vec{q} \right) dx = \delta J - \int_L \vec{\Psi}^T \left(\frac{\partial(\delta \vec{f})}{\partial x} - T_U \delta \vec{U} - T_b \delta \vec{b} \right) dx$$

where $\vec{\Psi}$ is the array of the adjoint variable (1D) fields and

$$T_b = \frac{\partial \vec{q}}{\partial \vec{b}} \quad , \quad T_U = \frac{\partial \vec{q}}{\partial \vec{U}}$$

Inverse Design of a Quasi-1D Duct – Compressible Flow

Integrate by parts:

$$\delta J_{aug} = \delta J - \int_L \vec{\Psi}^T \left(\frac{\partial(\delta \vec{f})}{\partial x} - \delta \vec{q} \right) dx = \delta J - \int_L \vec{\Psi}^T \left(\frac{\partial(\delta \vec{f})}{\partial x} - T_U \delta \vec{U} - T_b \delta \vec{b} \right) dx$$

$$\delta J_{aug} = \underbrace{\int_L \delta \vec{U}^T \left[(p - p_{tar}) (\frac{\partial p}{\partial \vec{U}})^T + A^T \frac{\partial \vec{\Psi}}{\partial x} + T_U^T \vec{\Psi} \right] dx}_{FAE}$$

$$- \underbrace{\left[\vec{\Psi}^T A \delta \vec{U} \right]_{out} + \left[\vec{\Psi}^T A \delta \vec{U} \right]_{in}}_{ABC} + 1/2 \underbrace{\int_L \vec{\Psi}^T T_b \, \delta \vec{b} dx}_{SD}$$

Inverse Design of a Quasi-1D Duct – Compressible Flow

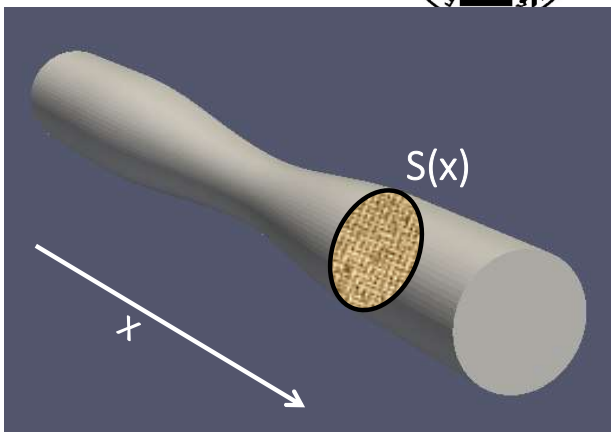
Field Adjoint Equations:

$$\frac{\partial \vec{\Psi}}{\partial t} - \left(A^T \frac{\partial \vec{\Psi}}{\partial x} + T_U^T \vec{\Psi} + (p - p_{tar})(\frac{\partial p}{\partial \vec{U}})^T \right) = 0$$

Compare with the Field Primal Equations:

$$\frac{\partial \vec{U}}{\partial t} + A \frac{\partial \vec{U}}{\partial x} = \vec{q}$$

Inverse Design of a Quasi-1D Duct – Compressible Flow



Adjoint Boundary Conditions (ABC):

$$\delta\vec{U}^T A^T \vec{\Psi} = 0$$

or:

$$\left[\tfrac{\gamma-3}{2}u^2\Psi_2 + (-\gamma u E + (\gamma-1)u^3)\,\Psi_3\right]\delta\rho +$$
$$\left[\Psi_1 + (3-\gamma)u\Psi_2 + (\gamma E - \tfrac{\gamma-1}{2}3u^2)\,\Psi_3\right]\delta(\rho u) +$$
$$\left[(\gamma-1)\Psi_2 + \gamma u\Psi_3\right]\delta(\rho E) = 0$$

<u>Inlet</u> ABC:

$$\rho\left[1-\frac{u^2}{c^2}\right]\Psi_1 + \rho u\left[1-\frac{u^2}{c^2}\right]\Psi_2 + \rho u^2\left[-\frac{u^2}{2c^2} + \frac{\gamma-3}{2(\gamma-1)} + \frac{1}{\gamma-1}\frac{c^2}{u^2}\right]\Psi_3 = 0$$

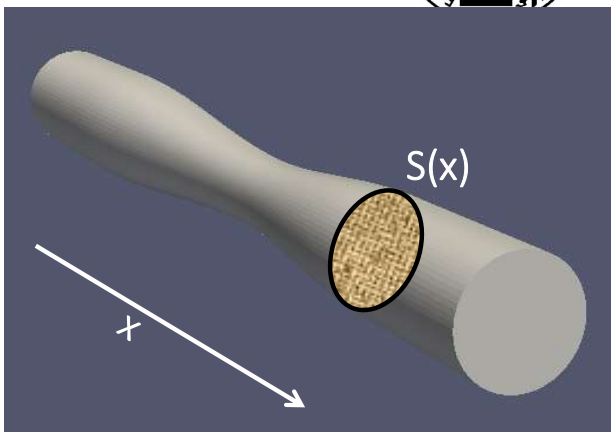
Outlet ABC:

$$-u^2\Psi_2 + \left(-\gamma u E + \frac{\gamma-2}{2}u^3\right)\Psi_3 = 0, \quad \Psi_1 + 2u\Psi_2 + \left(\gamma E - \frac{\gamma-3}{2}u^2\right)\Psi_3 = 0$$

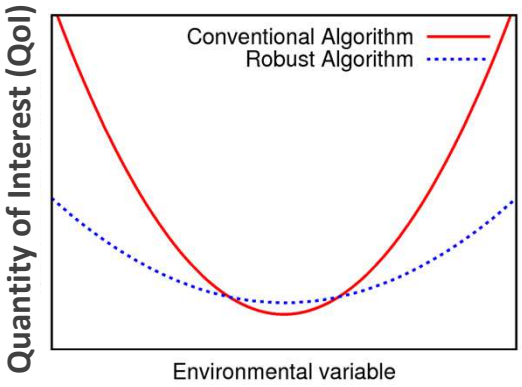
Inverse Design of a Quasi-1D Duct – Compressible Flow

Sensitivity Derivatives (SD):

$$\frac{\delta J}{\delta b_i} = \int_L \vec{\Psi}^T \vec{T}_b dx$$



Στιβαρός Σχεδιασμός – Robust Design (Design under Uncertainties)



For N design (b_i) & M environmental (c_i) variables, minimize

$$\widehat{F} = \hat{\mu_F} + k\hat{\sigma_F}$$

where the mean and the standard deviation of F (QoI) are:

$$\hat{\mu_F} = F_{\mathrm{D}} + \frac{1}{2}\left[\frac{d^2F}{dc_i^2}\right]_{\mathrm{D}}\sigma_i^2$$

$$\hat{\sigma_F} = \sqrt{\left[\frac{dF}{dc_i}\right]_{\mathrm{D}}^2\sigma_i^2 + \frac{1}{2}\left[\frac{d^2F}{dc_idc_j}\right]_{\mathrm{D}}^2\sigma_i^2\sigma_j^2}$$

(Second-Order, Second-Moment, SOSM, approach)

If Steepest Descent:

$$\frac{d\widehat{F}}{db_l} = \underbrace{\frac{dF}{db_l}}_{term1} + \underbrace{\frac{1}{2}\frac{d^3F}{dc_i^2db_l}\sigma_i^2}_{term2} + k\frac{2\frac{dF}{dc_i}\frac{d^2F}{dc_idb_l}\sigma_i^2 + \frac{d^2F}{dc_idc_j}\frac{d^3F}{dc_idc_jdb_l}\sigma_i^2\sigma_j^2}{2\sqrt{\left[\frac{dF}{dc_i}\right]^2\sigma_i^2 + \frac{1}{2}\left[\frac{d^2F}{dc_idc_j}\right]^2\sigma_i^2\sigma_j^2}}$$

Needs the Hessian of F w.r.t. to c and one more differentiation w.r.t. b. The recommended approach, if M<<N, is : DD_c-DD_c-AV_b.

Στιβαρός Σχεδιασμός – Robust Design (Design under Uncertainties)

Uncertainty Quantification (UQ) refers to the computation of the statistical moments (mean value, standard deviation) of a Quantity of Interest, QoI (drag, lift, losses).

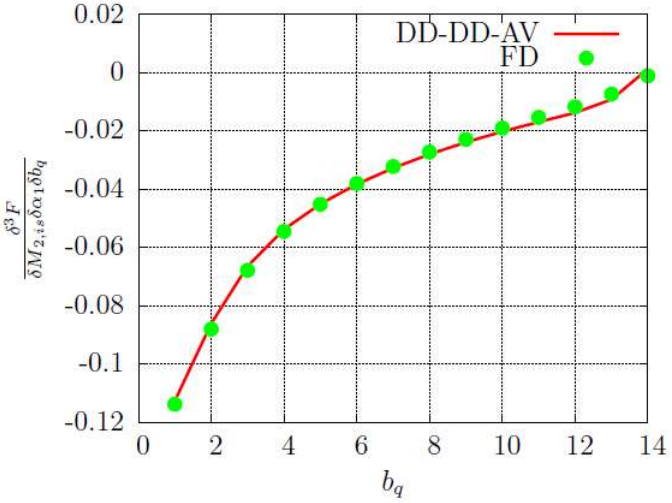
The computation of the Hessian of the QoI is needed even if optimization is not to be performed. A gradient-based optimization under uncertainties (in a Second-Order, Second-Moment, SOSM, approach) leads to the need of computing third-order mixed derivatives.

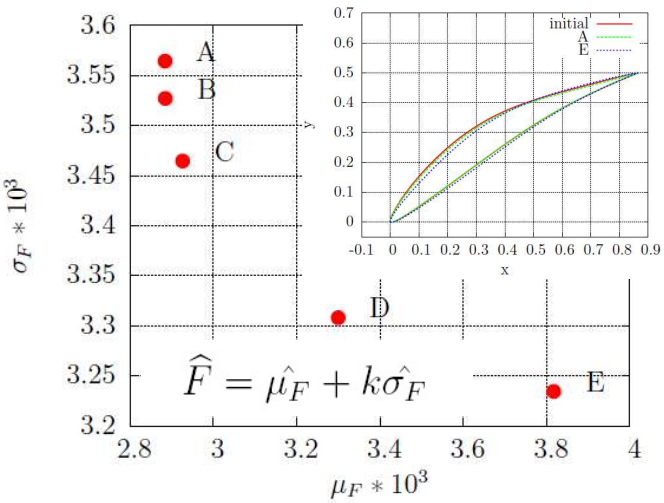
Working with the Newton method means that fourth-order derivatives are needed.

An interesting case: Some of the design and uncertain (environmental) variables may coincide.

Στιβαρός Σχεδιασμός – Robust Design (Design under Uncertainties)

Παράδειγμα Εφαρμογής:





Robust design of a compressor cascade: Two environmental variables: $M_{2,is}$ and α_1.

E.M. PAPOUTSIS-KIACHAGIAS, D.I. PAPADIMITRIOU, K.C. GIANNAKOGLOU: 'Robust Design in Aerodynamics using 3rd-Order Sensitivity Analysis based on Discrete Adjoint. Application to Quasi-1D Flows', International Journal for Numerical Methods in Fluids, Vol. 69, No. 3, pp. 691-709, 2012.

E.M. PAPOUTSIS-KIACHAGIAS, D.I. PAPADIMITRIOU, K.C. GIANNAKOGLOU: Discrete and Continuous Adjoint Methods in Aerodynamic Robust Design problems, CFD and Optimization 2011, ECCOMAS Thematic Conference, Antalya, Turkey, May 23-25, 2011.

Άλλες Χρήσεις της Hessian(F) ή Hess(F)

In (exact) Newton's Method:

$$\overrightarrow{b}^{new} = \overrightarrow{b}^{old} - \nabla^2 F(\overrightarrow{b}^{old})^{-1} \nabla F(\overrightarrow{b}^{old})^T$$

The most efficient approach is: DD-AV.
(DD to compute the gradient, AV=adjoint to compute second derivatives).

Μπορείτε να δείξετε ότι η διττή χρήση της συζυγούς μεθόδους (AV-AV) είναι πιο αργή από τη DD-AV. Ο λόγος: Χρειαζόμαστε το grad(F) πριν βρούμε το Hess(F). (Δείτε παρακάτω).

Computation of the Hessian(F) or Hess(F)

Hessian Matrix Computation using the DD-DD method:
(Think "Discrete" , program Continuous Adjoint)

$$\frac{dF}{db_i} = \frac{\partial F}{\partial b_i} + \frac{\partial F}{\partial U_k}\frac{dU_k}{db_i}$$

k=1,…,N design variables

$$\frac{dR_m}{db_i} = \frac{\partial R_m}{\partial b_i} + \frac{\partial R_m}{\partial U_k}\frac{dU_k}{db_i} = 0$$

$$\frac{d^2F}{db_i db_j} = \frac{\partial^2 F}{\partial b_i \partial b_j} + \frac{\partial^2 F}{\partial b_i \partial U_k}\frac{dU_k}{db_j} + \frac{\partial^2 F}{\partial U_k \partial b_j}\frac{dU_k}{db_i} + \frac{\partial^2 F}{\partial U_k \partial U_m}\frac{dU_k}{db_i}\frac{dU_m}{db_j} + \frac{\partial F}{\partial U_k}\frac{d^2U_k}{db_i db_j}$$

$$\frac{d^2R_n}{db_i db_j} = \frac{\partial^2 R_n}{\partial b_i \partial b_j} + \frac{\partial^2 R_n}{\partial b_i \partial U_k}\frac{dU_k}{db_j} + \frac{\partial^2 R_n}{\partial U_k \partial b_j}\frac{dU_k}{db_i} + \frac{\partial^2 R_n}{\partial U_k \partial U_m}\frac{dU_k}{db_i}\frac{dU_m}{db_j} + \frac{\partial R_n}{\partial U_k}\frac{d^2U_k}{db_i db_j} = 0$$

The cost for computing the Hessian via the DD-DD approach scales with N^2.

Computation of the Hessian(F) or Hess(F)

$$\frac{dR_m}{db_i} = \frac{\partial R_m}{\partial b_i} + \frac{\partial R_m}{\partial U_k}\frac{dU_k}{db_i} = 0 \quad \Rightarrow \quad \boxed{\frac{dU_k}{db_i}} \qquad \boxed{N}$$

System solutions (EFS)

$$\frac{\partial F}{\partial U_k} + \hat{\Psi}_n \frac{\partial R_n}{\partial U_k} = 0 \quad \Rightarrow \quad \boxed{\hat{\Psi}_n} \qquad \boxed{1}$$

EFS

The Adjoint equation is the same with that used to compute the Gradient !!!

$$\begin{aligned}\frac{d^2\hat{F}}{db_i db_j} = {} & \frac{\partial^2 F}{\partial b_i \partial b_j} + \hat{\Psi}_n \frac{\partial^2 R_n}{\partial b_i \partial b_j} + \frac{\partial^2 F}{\partial U_k \partial U_m}\frac{dU_k}{db_i}\frac{dU_m}{db_j} + \hat{\Psi}_n \frac{\partial^2 R_n}{\partial U_k \partial U_m}\frac{dU_k}{db_i}\frac{dU_m}{db_j} \\ & + \frac{\partial^2 F}{\partial b_i \partial U_k}\frac{dU_k}{db_j} + \hat{\Psi}_n \frac{\partial^2 R_n}{\partial b_i \partial U_k}\frac{dU_k}{db_j} + \frac{\partial^2 F}{\partial U_k \partial b_j}\frac{dU_k}{db_i} + \hat{\Psi}_n \frac{\partial^2 R_n}{\partial U_k \partial b_j}\frac{dU_k}{db_i} \\ & + \left(\frac{\partial F}{\partial U_k} + \hat{\Psi}_n \frac{\partial R_n}{\partial U_k}\right)\frac{d^2 U_k}{db_i db_j}\end{aligned}$$

$$\frac{d^2 F^\lambda}{db_i db_j} db_j^\lambda = -\frac{dF^\lambda}{db_i}$$

- The cost per Newton cycle is N+1+1=N+2 EFS! Scales with N, not N2.
- DD-AV is the most efficient approach (among DD-DD, AV-DD, AV-AV)!

Computation of the Hessian(F) or Hess(F)

Ότι παρουσιάστηκε για τη Διακριτή Συζυγή Μέθοδο, εφαρμόζεται εξίσου και με τη Συνεχή Συζυγή Μέθοδο.

Απλώς, προτιμήθηκε η παρουσίαση να γίνει με τη Διακριτή Μέθοδο, γιατί είναι περισσότερο σύντομη και εποπτική!

Ακολουθεί παράδειγμα (στον πίνακα) με εφαρμογή της συζυγούς συζυγούς μεθόδου για τον υπολογισμό του Hess(F)…..