Μέθοδοι Αεροδυναμικής Βελτιστοποίησης

Αυτόματη Διαφόριση (Automatic Differentiation, AD)

Αυτόματη Διαφόριση (Automatic Differentiation)είναι η θεωρία, ο αλγόριθμος και το αντίστοιχο λογισμικό με τα οποία, όταν υπάρχει ένας αλγόριθμος-λογισμικό που (α) λαμβάνει ως εισόδους τις τιμές των N ελεύθερων μεταβλητών $\overrightarrow{b}$, (β) αναλύει ένα πρόβλημα και (γ) υπολογίζει τιμή ή τιμές για M αντικειμενικές συναρτήσεις (βαθμωτή $F(\overrightarrow{b})$, για $M=1$ ή διανυσματική $\overrightarrow{F}(\overrightarrow{b})$, για $M>1$), σύμφωνα με την απεικόνιση $\Re^N \rightarrow \Re^M$), σχηματίζεται ένας νέος αλγόριθμος και ένα νέο λογισμικό που κάνει όλα τα παραπάνω και επιπλέον υπολογίζει τιμές για $N \times M$ παραγώγους όλων των αντικειμενικών συναρτήσεων ως προς όλες τις ελεύθερες μεταβλητές.

Έστω ότι ο μηχανικός που ασχολείται με την αεροδυναμική βελτιστοποίηση και σκοπεύει να εφαρμόσει μια αιτιοκρατική (βασισμένη στην κλίση της αντικειμενικής συνάρτησης) μέθοδο σχεδιασμού–βελτιστοποίησης, διαθέτει έναν κώδικα που (α) διαβάζει τις μεταβλητές σχεδιασμού που καθορίζουν τη μορφή μιας αεροτομής, (β) δημιουργεί το περίγραμμα της αεροτομής, (γ) δημιουργεί το υπολογιστικό πλέγμα, (δ) επιλύει αριθμητικά σε αυτό της εξισώσεις Navier–Stokes και (ε) αποδίδει-τυπώνει την τιμή των συντελεστών άνωσης και οπισθέλκουσας, ως προς τις οποίες γίνεται η βελτιστοποίηση. Έστω, επίσης, ότι τα ανωτέρω πραγματοποιούνται με ενιαίο κώδικα (γιατί, συνήθως, τα παραπάνω πραγματοποιούνται με τη διαδοχική εκτέλεση πολλών κωδίκων - η περίπτωση πολλών κωδίκων δεν είναι πρόβλημα, απλά απαιτεί διαφορετική αντιμετώπιση). Τέλος, έστω ότι ο κώδικας αυτός είναι γραμμένος σε Fortran 77/90 ή ANSI C/C++ και είναι διαθέσιμη η πηγαία μορφή του. Το τελευταίο είναι σημαντικό και ουσιαστικά αποκλείει τη χρήση της Αυτόματης Διαφόρισης σε εμπορικούς κώδικες (οι οποίοι ποτέ δεν είναι διαθέσιμοι σε ανοικτή μορφή - ο μηχανικός έχει πάντα πρόσβαση μόνο στον εκτελέσιμο κώδικα.

Σε μια τέτοια περίπτωση, ο μηχανικός μπορεί να χρησιμοποιήσει λογισμικό Αυτόματης Διαφόρισης, να το τροφοδοτήσει με τον κώδικα ανάλυσης σε πηγαία μορφή και να παράγει τον κώδικα που θα υπολογίζει την κλίση της αντικειμενικής συνάρτησης που θα χρειαστεί στη βελτιστοποίηση. Συνεπώς, το λογισμικό Αυτόματης Διαφόρισης βρίσκεται ουσιαστικά σε υψηλότερο επίπεδο από το λογισμικό ανάλυσης και παράγει κώδικα αντί να παράγει αριθμητικά αποτελέσματα.

Το κεφάλαιο αυτό δεν αποτελεί εγχειρίδιο χρήσης για κάποιο από τα υπάρχοντα λογισμικά Αυτόματης Διαφόρισης. Προσπαθεί να εξηγήσει, με απλό μαθηματικό τρόπο, το πως λειτουργεί ένα τέτοιο λογισμικό. Ως παράδειγμα, όμως, και πριν την παρουσίαση της θεωρίας, ας δούμε τι θα έκανε ένα (ελεύθερο) τέτοιο λογισμικό, το Tapenade (έχει αναπτυχθεί από το Γαλλικό Εθνικό Ερευνητικό Ινστιτούτο INRIA), αν τροφοδοτείτο από το υποπρόγραμμα

```
subroutine ff(b1,b2,b3,F1,F2)
implicit double precision(a-h,o-z)
F1 = b1*b2*b3
F2 = b1+b2+b3
return
```

```
      end
```

Το υποπρόγραμμα αυτό υπολογίζει τις απλές συναρτήσεις $F_1 = b_1 b_2 b_3$ και $F_2 = b_1 + b_2 + b_3$, έχοντας τρείς εισόδους και δύο εξόδους ως ορίσματα. Η επεξεργασία του με το Tapenade (με έναν από τους πιθανούς τρόπους που περιγράφονται στη συνέχεια) θα επέστρεφε τον κώδικα

```
C        Generated by TAPENADE     (INRIA, Tropics team)
C  Version 2.0.12 - (Id: 1.30.4.1 vmp Exp - Wed Nov 10 11:04:16 MET 2004)
C
C  Differentiation of ff in forward (tangent) mode:
C   variations  of output variables: F1 F2
C   with respect to input variables: b1 b2 b3
C
C
      SUBROUTINE FF_D(b1, b1d, b2, b2d, b3, b3d, F1, F1d, F2, F2d)
      IMPLICIT NONE
      DOUBLE PRECISION x1, x1d, x2, x2d, x3, x3d, y, yd, z, zd
C
      F1d = (b1d*b2+b1*b2d)*b3 + b1*b2*b3d
      F1 = b1*b2*b3
      F2d = b1d + b2d + b3d
      F2 = b1 + b2 + b3
C
      RETURN
      END
```

Αρκεί να κληθεί το υποπρόγραμμα που προέκυψε με ορίσματα $(b1d, b2d, b3d)$ τα $(1,0,0)$ ή $(0,1,0)$ ή $(0,0,1)$ και να προκύψουν στα $(F1d, F2d)$ οι τιμές των παραγώγων $(\frac{\partial F_1}{\partial b_1}, \frac{\partial F_2}{\partial b_1})$ ή $(\frac{\partial F_1}{\partial b_2}, \frac{\partial F_2}{\partial b_2})$ ή $(\frac{\partial F_1}{\partial b_3}, \frac{\partial F_2}{\partial b_3})$, αντίστοιχα. Σημειώστε ότι, για να υπολογισθούν οι τιμές όλων των παραγώγων, θα χρειαστούν τόσες κλίσεις όσες και οι ελεύθερες μεταβλητές (τρεις).

Η μέθοδος της Αυτόματης Διαφόρισης είναι ουσιαστικά ένας αυτοματοποιημένος τρόπος να υλοποιείται η συζυγής μέθοδος που επίσης παρουσιάζεται εκτενώς σε αυτό το μάθημα. Σε αυστηρή διατύπωση, η μέθοδος της Αυτόματης Διαφόρισης παράγει το διακριτό συζυγή κώδικα (discrete adjoint code) του κώδικα ανάλυσης. Το λογισμικό Αυτόματης Διαφόρισης χρησιμοποιεί μια σειρά έξυπνων λεκτικών και μαθηματικών κανόνων. Εφαρμόζεται σε κώδικες γραμμένους σε Fortran 77/90 ή ANSI C/C++. Γενικά, υπάρχουν εντολές στον πηγαίο κώδικα που 'δυσκολεύουν' τη διαφόριση, αλλά αυτό αλλάζει άπό λογισμικό σε λογισμικό. Είναι, όμως, πολύ σημαντικό να διατυπωθεί ότι η μέθοδος της Αυτόματης Διαφόρισης οδηγεί σε υπολογισμό της ακριβούς τιμής της παραγώγου.

Υπάρχουν δύο τρόποι Αυτόματης Διαφόρισης, ο λεγόμενος **Ευθύς** (Forward) και ο λεγόμενος **Αντίστροφος** (Reverse). Σήμερα, η έρευνα στρέφεται σε υβριδικές παραλλαγές τους, που συνδυάζουν έξυπνα τα πλεονεκτήματα των δύο τρόπων. Και οι

δύο τρόποι Αυτόματης Διαφόρισης χρησιμοποιούν τον κανόνα της αλυσίδας για την εύρεση των παραγώγων. Για N ανεξάρτητες μεταβλητές (μεταβλητές σχεδιασμού) $(b_1, \ldots, b_N)$ και (έστω) μια βαθμωτή αντικειμενική συνάρτηση F (M=1), η τιμή της F θεωρείται ότι προκύπτει ως αποτέλεσμα m–N διαδοχικών τελέσεων (πράξεων, με άλλα λόγια), καθεμιά από τις οποίες αντιστοιχεί σε μια εντολή του κώδικα. Κατά συνέπεια, η σειρά αυτή των πράξεων εισάγει m–N ενδιάμεσες (εξαρτημένες, δηλαδή) μεταβλητές $b_{N+1}, \ldots, b_m$.

Η **Ευθεία Αυτόματη Διαφόριση** δημιουργείται με τη διαδοχική μεταφορά παραγώγων των ενδιάμεσων μεταβλητών ως προς τις ανεξάρτητες μεταβλητές. Ας θεωρήσουμε την παρακάτω διαδοχή βοηθητικών τελέσεων (που αντιστοιχούν σε εντολές εικονικού κώδικα)

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

οι οποίες καταλήγουν στο να παράγουν την επιθυμητή απόκριση F. Ο ευθύς τρόπος αναγκάζεται να αποθηκεύσει όλες τις κλίσεις των ενδιάμεσων μεταβλητών (αυτές που εδώ συμβολίζονται με ∇b_i) και οι οποίες υπολογίζονται με τον κανόνα της αλυσίδας ως εξής:

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

όπου $\overrightarrow{e}_i$ είναι η i–ιοστή στήλη του $N{\times}N$ μοναδιαίου μητρώου. Στον αντίστοιχο παράγωγο κώδικα, που έχει εντολές αντίστοιχες των τελέσεων που παρουσιάσθηκαν παραπάνω, η επιθυμητή κλίση είναι το αποτέλεσμα της τελευταίας τέλεσης (∇F). Άμεσα φαίνεται το κόστος σε μνήμη υπολογιστή είναι σημαντικό, αφού αποθηκεύονται όλες

οι ενδιάμεσες μεταβλητές και οι παράγωγοί τους.

Ακολουθεί ένα απλό παράδειγμα εφαρμογής του αλγορίθμου της Ευθείας Αυτόματης Διαφόρισης. Έστω ότι έχουμε ένα πρόβλημα με $N=4$ ελεύθερες μεταβλητές, τις b_1, b_2, b_3, b_4 και μια αντικειμενική συνάρτηση, την $F=b_1 b_2 b_3 b_4$. Η απεικόνιση δηλαδή είναι η $\Re^4 \rightarrow \Re$. Εισάγονται οι βοηθητικές (ενδιάμεσες, εξαρτημένες) μεταβλητές

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

είναι δηλαδή $m=8$. Βέβαια, ο παραπάνω τρόπος εκτέλεσης του πολλαπλασιασμού δεν είναι μοναδικός. Ο αναγνώστης μπορεί να τον αντικαταστήσει με ένα διαφορετικό, λ.χ. ορίζοντας ότι $b_5 = b_1 b_2$, κοκ. Η Ευθεία Αυτόματη Διαφόριση υλοποιείται με τα βήματα–εντολές κώδικα που κατά σειρά θα είναι τα εξής:

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

Στην **Αντίστροφη Αυτόματη Διαφόριση** ορίζεται μια βαθμωτή παράγωγος (συμβολίζεται με $\bar{b}_i$) και αντιστοιχίζεται σε κάθε ενδιάμεση μεταβλητή ($i = N+1, \ldots, m$). Οι νέες ποσότητες ορίζονται ως εξής

$$\bar{b}_i = \frac{\partial b_m}{\partial b_i}$$

Εξ ορισμού, $\bar{b}_m \equiv 1$. Ο ορισμός επεκτείνεται και στις ανεξάρτητες μεταβλητές (μεταβλητές σχεδιασμού, δηλαδή για $i=1, \ldots, N$). Σύμφωνα με τον κανόνα της αλυσίδας, όλες πλήν της m–ιοστής ποσότητας $\bar{b}_i$, που αντιστοιχούν στις εξαρτημένες μεταβλητές,

μπορούν να υπολογισθούν από τη σχέση

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

ενώ, για τις ανεξάρτητες μεταβλητές, είναι

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

Από αλγοριθμικής σκοπιάς, η Αντίστροφη Αυτόματη Διαφόριση δημιουργείται με δύο διαδοχικές σαρώσεις, την προς τα εμπρός (forward sweep) και την προς τα πίσω σάρωση (backward sweep). Με τη σάρωση προς τα εμπρός, υπολογίζονται και αποθηκεύονται όλες οι εξαρτημένες μεταβλητές ενώ στις παραγώγους τους δίνονται αρχικές μηδενικές τιμές. Σύμφωνα με τον προηγούμενο συμβολισμό:

$$\begin{aligned} b_i &= f_i\left(b_1, \ldots, b_{i-1}\right) \ , \quad i = N+1, \ldots, m \\ \bar{b}_i &= 0 \ , \quad i = 1, \ldots, m-1 \\ \bar{b}_m &= 1 \end{aligned}$$

Η προς τα πίσω σάρωση, η οποία συγκεντρώνει (συναθροίζει) συμμετοχές στα $\bar{b}_i, i = m, \ldots, N+1$, αποτελείται ουσιαστικά από δύο βρόχους, τον ένα μέσα στον άλλο (nested loops), οι οποίοι γράφονται σε ψευδοκώδικα στη μορφή

$$\begin{array}{l} do\ j = m, N+1, -1 \\ \quad do\ i = 1, j-1 \\ \quad\quad \bar{b}_i = \bar{b}_i + \bar{b}_j \dfrac{\partial f_j}{\partial b_i} \\ \quad enddo \\ enddo \end{array}$$

Ένα αντίστοιχο παράδειγμα με αυτό που παρουσιάσθηκε προηγουμένως για την Ευθεία μέθοδο επαναλαμβάνεται, εδώ, και για την Αντίστροφη Αυτόματη Διαφόριση. Τώρα, επιλέγουμε μόνο $N = 3$ ελεύθερες μεταβλητές, τις b_1, b_2, b_3 και το γινόμενό τους ως αντικειμενική συνάρτηση, είναι δηλαδή $F = b_1 b_2 b_3$. Η νέα απεικόνιση είναι, προφανώς, η $\Re^3 \rightarrow \Re$. Οι ενδιάμεσες (εξαρτημένες) μεταβλητές θα είναι οι

$$\begin{aligned} b_4 &= f_4\left(b_1, b_2, b_3\right) = b_1 \\ b_5 &= f_5\left(b_1, b_2, b_3, b_4\right) = b_2 b_4 \\ b_6 &= f_6\left(b_1, b_2, b_3, b_4, b_5\right) = b_3 b_5 \\ F &= b_6 \end{aligned}$$

είναι δηλαδή $m = 6$. Για τις εξαρτημένες και ανεξάρτητες μεταβλητές ισχύουν κατά

σειρά ότι:

$$\begin{aligned}
\bar{b}_6 &= \frac{\partial f_6}{\partial b_6} \equiv 1 \\
\bar{b}_5 &= \frac{\partial f_6}{\partial b_6}\frac{\partial f_6}{\partial b_5} = \bar{b}_6\frac{\partial f_6}{\partial b_5} \\
\bar{b}_4 &= \frac{\partial f_6}{\partial b_6}\frac{\partial f_6}{\partial b_4} + \frac{\partial f_6}{\partial b_5}\frac{\partial f_5}{\partial b_4} = \bar{b}_6\frac{\partial f_6}{\partial b_4} + \bar{b}_5\frac{\partial f_5}{\partial b_4} \\
\bar{b}_3 &= \frac{\partial f_6}{\partial b_6}\frac{\partial f_6}{\partial b_3} + \frac{\partial f_6}{\partial b_5}\frac{\partial f_5}{\partial b_3} + \frac{\partial f_6}{\partial b_4}\frac{\partial f_4}{\partial b_3} = \bar{b}_6\frac{\partial f_6}{\partial b_3} + \bar{b}_5\frac{\partial f_5}{\partial b_3} + \bar{b}_4\frac{\partial f_4}{\partial b_3} \\
\bar{b}_2 &= \frac{\partial f_6}{\partial b_6}\frac{\partial f_6}{\partial b_2} + \frac{\partial f_6}{\partial b_5}\frac{\partial f_5}{\partial b_2} + \frac{\partial f_6}{\partial b_4}\frac{\partial f_4}{\partial b_2} = \bar{b}_6\frac{\partial f_6}{\partial b_2} + \bar{b}_5\frac{\partial f_5}{\partial b_2} + \bar{b}_4\frac{\partial f_4}{\partial b_2} \\
\bar{b}_1 &= \frac{\partial f_6}{\partial b_6}\frac{\partial f_6}{\partial b_1} + \frac{\partial f_6}{\partial b_5}\frac{\partial f_5}{\partial b_1} + \frac{\partial f_6}{\partial b_4}\frac{\partial f_4}{\partial b_1} = \bar{b}_6\frac{\partial f_6}{\partial b_1} + \bar{b}_5\frac{\partial f_5}{\partial b_1} + \bar{b}_4\frac{\partial f_4}{\partial b_1}
\end{aligned}$$

όπου, ειδικά για τις δύο τελευταίες σχέσεις έχει ληφθεί υπόψη ότι

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

αφού τα b_1, b_2, b_3 είναι ανεξάρτητες μεταβλητές.

Αρχικά πραγματοποιείται η διαδοχική συνάθροιση όρων προς τα πίσω (αντίστροφη σάρωση), κατά την οποία ο αλγόριθμος της Αντίστροφης Αυτόματης Διαφόρισης λειτουργεί με τα εξής βήματα:

- Βήμα 0: Αρχικοποιούνται τα:

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

- Βήμα 1: Ορίζοντας $f_6 = b_3 b_5$, προκύπτουν οι παράγωγοι:

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

  και συναθροίζονται συμμετοχές στα μεγέθη b_i ως εξής:

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

- Βήμα 2: Ορίζοντας $f_5 = b_2 b_4$, προκύπτουν οι παράγωγοι:

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

και συναθροίζονται συμμετοχές στα μεγέθη b_i ως εξής:

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

- Βήμα 3: Ορίζοντας, τέλος, $f_4 = b_1$, προκύπτουν οι παράγωγοι:

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

και συναθροίζονται συμμετοχές στα μεγέθη b_i ως εξής:

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

Στο τέλος, πραγματοποιείται η διαδοχική συνάθροιση όρων προς τα εμπρός, όπου

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

που είναι και η ζητούμενη τιμή της κλίσης του F.

Μεταξύ των διαφόρων τύπων λογισμικού Αυτόματης Διαφόρισης που υπάρχουν (κάποια είναι δωρεάν και βρίσκονται στο Διαδίκτυο) για κώδικες γραμμένους σε Fortran 77/90 ή σε ANSI C/C++, αναφέρουμε τα ADIFOR (Automatic Differentiation of Fortran), TAMC (Tangent linear and Adjoint Model Compiler), TAF (Transformation of Algorithms in Fortran) DAFOR (Differential Algebraic Extension of Fortran), GRESS (Gradient–Enhanced Software System), Odyssée, 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++).