자동미분의 공력최적설계 적용

이 재 훈*1, 김 수 환*1, 안 중 기*1, 권 장 혁*2

Application of the Automatic Differentiation to Aerodynamic Design Optimization

Jaehun Lee, Suwhan Kim, Joongki Ahn, Jang Hyuk Kwon

In gradient based optimization methods, the finite differencing which uses small perturbations in the design variables has been used to calculate the sensitivity. Recently, the automatic differentiation has been widely studied to calculate the function value and the sensitivities simultaneously. In this paper, the applicability of the automatic differentiation in the aerodynamic design optimization is studied. ADIFOR and TAPENADE are used to generate the codes which give the function value and the sensitivities for 2D compressible inviscid flows.

Key Words: 최적설계(Optimum Design), 자동미분(Automatic Differentiation), ADIFOR, TAPENADE

1. 서 론

최근 최적설계 기법이 발달함에 따라 기존의 small perturbation을 주어 sensitivity를 구하는 유한차분(Finite Differencing, FD)이 아닌 자동미분(Automatic Differentiation, AD)을 이용해 sensitivity를 구하는 연구가 광범위하게 진행되고 있다. 이 방법은 수학적인 정확한 미분과 마찬가지의 정확도를 주는 것으로 알려져 있다. 본 연구에서는 이와 같은 자동미분을 공력최적설계에 적용하여 민감도의 정확성을 알아보고 다른 방법과의 계산 성능을 비교해 보고자 하였다.

2. Automatic Differentiation

자동미분의 기본 개념은 이미 1950년대에 제안되었다. 1960년대에는 code list의 개념을 이용한 forward mode의 자동미분이 논의되어졌다[1]. 이는 연속적인 chain rule을 적용하는 것으로, 함수계산에 연이어 미분 계산이 이루어지므로, 실제 적용이 간편한 이점이 있다. 1970년대에는 reverse mode의 개념이 제안되어졌고, Speelpenning[2]에 의해 최초로 적용되어졌다. 이는 adjoint sensitivity와 동일한 원리를 사용하는 것으로 함수의 계산을 한 후 미분 계산을 시작하게 된다.

1980년대에 들어 Griewank와 Bishof와 같은 연구자들의 기여로 인해 자동미분은 부흥기에 접어들었으며 사용하기 편리한 자동미분 프로그램들의 개발에 이르렀다. 공력최적설계와 같이 복잡한 계산 프로그램에도 자동미분은 꾸준히 사용되어져왔다[3,4].

AD는 FD에 비해 수치 오차가 적다는 특징이 있다. FD의 경우 step size가 작아질 경우 round off error 등으로 인해 민감도의 정확도가 떨어지는 단점이 있는데 반해 AD는 수학적 chain rule을 연속적으로 사용해 민감도를 구하기 때문에 step size가 필요없고 machine precision 정도의 정확도를 가진다.

그러나 AD는 보통 함수 계산과 민감도 계산을 동시에 수행하므로 추가적 메모리와 계산량의 증가로 인해 계산시간이 증가하는 경향이 있다.

*1 학생회원, 한국과학기술원 항공우주전공
*2 정회원, 한국과학기술원 항공우주전공
*E-mail : jaehun94@kaist.ac.kr

2.1 AD의 구현 방법

AD를 구현하는 방법은 operator overloading과 source transformation의 두 가지로 나누어진다. operator overloading이란 연산자에 새로운 규칙을 추가하는 것으로 고급의 프로그래밍 언어에서만 지원된다(C++, FORTRAN90 등). 따라서 operator overloading을 이용하는 AD tool의 경우 C나 FORTRAN77로 만들어진 in-house 프로그램에선 사용이 불가능할 수도 있다. source transformation은 기존의 프로그램을 이용해 민감도와 함수값을 계산해 주는 프로그램을 새로 만드는 방법이다.

2.2 AD의 두 가지 모드

민감도를 계산하는 방법은 크게 두 가지로 나누어지는 데, forward mode와 reverse mode를 들 수 있다. forward모드는 함수값들의 진행과 미분값의 진행방향이 동일한 것으로 tangent mode라고도 불린다. reverse 모드는 함수값들의 계산을 완료한 후 미분값의 계산을 시작하는 것으로, discrete adjoint equation의 생성과 동일한 원리를 사용하기 때문에 adjoint mode라고도 불린다.

reverse mode의 경우엔 함수값의 계산 결과를 보존해야 하므로 메모리의 사용이 큰 단점이 있으나 adjoint equation을 이용하므로 설계변수가 많을 경우 유리하다.

2.3 사용 가능한 AD tool

■ADIFOR: hybrid 형으로 forward와 backward 모드가 동시에 사용된다[5]. SparseLinC library를 통해 Jacobian matrix가 sparse한 경우에 유용하다. ADIFOR2.0은 공개 소프트웨어로 인터넷을 통해 일정 절차를 거쳐 구할 수 있다.

■ADMAT: Matlab에서 사용되는 AD program으로 operator overloading을 사용한다[6].

■TAMC, TAF: TAF는 TAMC를 승계한 프로그램으로 상용이며 f77, f90 및 병렬화 지원 등의 특징이 있다[7]. TAMC는 더 이상 지원되진 않지만 연구목적에 한해 공개되어 있다.

■TAPENADE: forward mode, reverse mode를 지원한다[8]. 인터넷의 application server를 통해 프로그램의 upload, download가 이루어져 별도의 프로그램을 설치할 필요가 없다. 코드의 보안을 위해서 직접 개인의 컴퓨터에 이 프로그램을 설치할 수도 있다.

2.4 예제

AD tool들의 작동 방식에 대해 알아보기 위해 식 (1)과 같은 함수에 대해 ADIFOR, TAPENADE를 이용해 derivative code를 만들어 보았다.

$$y = \Pi x_i^2 \qquad ,i=1,10 \tag{1}$$

$$\frac{dy}{dx_j} = 2x_j \Pi x_i^2 \qquad ,i=[1..j-1],[j+1..10] \tag{2}$$

Table 1에는 식 (1)의 함수를 FORTRAN77로 코딩한 것이고, Table 2,3,4는 ADIFOR, TAPENADE 등을 이용해 만든 derivative code들이다.

Table 2에 나타낸 forward mode의 경우 chain rule을 Table 1의 function code에 적용할 것이라 쉽게 이해할 수 있다. Table 3의 reverse mode는 forward mode와 달리 우선 함수값의 계산을 한 후 미분값을 구하는 것을 볼 수 있다.reverse mode에서는 pushreal4, popreal4와 같은 루틴들이 삽입돼있는데, 이들은 TAPENADE에서 제공하는 루틴들로서

Table 1 Function code for the Equation (1)

```
subroutine func(x,y)
real x(10),y
integer i
y=1.0
do i=1,10
      y=y*x(i)*x(i)
enddo
return
end
```

Table 2 Derivative code generated by TAPENADE with forward mode

```
subroutine func_d(x, xd, y, yd)
implicit none
real x(10), xd(10), y, yd
integer i
y = 1.0
yd = 0.0
do i=1,10
 yd =(yd*x(i)+y*xd(i))*x(i) + y*x(i)*xd(i)
 y = y*x(i)*x(i)
enddo
return
end
```

Table 3 Derivative code generated by TAPENADE with reverse mode

```
      subroutine func_b(x, xb, y, yb)
      implicit none
      real x(10), xb(10), y, yb
      integer ad_to, i, ii1
      y = 1.0
      do i=1,10
        call pushreal4(y)
        y = y*x(i)*x(i)
      enddo
      call pushinteger4(i - 1)
      do ii1=1,10
        xb(ii1) = 0.0
      enddo
      call popinteger4(ad_to)
      do i=ad_to,1,-1
        call popreal4(y)
        xb(i) = xb(i) + y*2*x(i)*yb
        yb = x(i)**2*yb
      enddo
      yb = 0.0
      end
```

Table 4 Derivative code generated by ADIFOR

```
      subroutine g_func(g_p_,x,g_x,ldg_x,y,g_y,ldg_y)
      real x(10), y
      integer i
      integer g_pmax_
      parameter (g_pmax_ = 10)
      integer g_i_, g_p_, ldg_y, ldg_x
      real r4_b, r3_b, r3_v, g_y(ldg_y), g_x(ldg_x, 10)
      integer g_ehfid
      data g_ehfid /0/
C
      call ehsfid(g_ehfid, 'func','g_func2.f')
C
      if (g_p_ .gt. g_pmax_) then
        print*,'Parameter g_p_ is greater than g_pmax_'
        stop
      endif
      do g_i_ = 1, g_p_
        g_y(g_i_) = 0.0
      enddo
      y = 1.0
C--------
      do i = 1, 10
          r3_v = y * x(i)
          r4_b = x(i) * x(i)
          r3_b = r3_v + x(i) * y
          do g_i_ = 1, g_p_
           g_y(g_i_) = r4_b*g_y(g_i_)+r3_b*g_x(g_i_,i)
          enddo
          y = r3_v * x(i)
C--------
        enddo
        return
      end
```

변수 y의 중간 계산값들을 저장하고 다시 불러들이는 역할을 한다. 이와 같이 reverse mode에서는 forward mode와 달리 추가적으로 중간 계산 결과를 저장해야 하므로 더 많은 메모리를 필요로 한다. forward mode와 reverse mode의 미분값의 결과도 서로 다른 변수에 저장되는 데, forward mode의 경우에는 yd에, reverse mode의 경우에는 xb에 저장된다. Table 4의 ADIFOR에서는 reverse mode와 비슷하지만 계산의 중간값들을 모두 기록하는 것이 아니라 임시 변수를 이용해 저장했다가 사용하는 것을 알 수 있다. Table 4에서 g_x는 행렬로서 identity 행렬일 경우에는 모든 변수들에 대한 미분값들을 구할 수 있다. 만약 대각항의 k번째 항이 1일 때는 k번째 변수에 대한 gradient만을 준다. 따라서 ADIFOR은 gradient는 물론, Jacobian을 계산하는 데 편리하다.

3. 공력최적설계

AD의 성능 검증을 위해 ADIFOR, TAPENADE등을 이용해 이차원 비점성 유동에 대한 공력최적설계를 수행하였다.

3.1 Flow solver 개요

사용된 유동 지배방정식은 Euler equation이고, 격자 중심의 유한체적법을 이용해 공차 차분을 하였다. 해의 정밀도 개선을 위해 2nd order upwind TVD와 minmod limiter가 사용되었다[9]. 시간전진은 내재적 기법의 일종인 대각화된 ADI를 이용하였다. 또한 Multigrid를 통해 수렴성 증대를 꾀하였다. 격자는 O타입으로 사이즈는 129×33이다. 유동 조건은 마하수 0.73, 받음각 2.78도로 설정하였다.

3.2 2D 에어포일 형상 최적화 개요

최적화 알고리즘은 BFGS 사용하였다. 목적함수는 양력 계수에 대한 구속조건을 포함하는 벌칙함수 형태로서 다음과 같이 구성하였다.

$$\min f = \frac{1}{2}(C_L - C_{L0})^2 + \frac{10}{2} C_D^2 \qquad (3)$$

식 (3)에서 C_{L0}는 RAE2822의 초기 양력계수로 최적화 초기의 함수 계산을 통해 구해지고, 이 값은 0.8911이다.

형상함수는 Hicks-Henne을 사용하였다. 설계변수

가 n개일 때의 일반적인 Hicks-Henne 함수는 아래와 같다.

$$f_1(x)=x^{0.25}(1-x)e^{-15x} \quad ,0\le x\le 1 \tag{4}$$

$$f_k(x)=(\sin\pi x^{c_k})^3 \quad ,0\le x\le 1 \quad k=2,..,n$$

$$c_k=\frac{\log 0.5}{\log\left(\frac{k-1}{n}\right)}$$

이들을 이용한 날개 형상의 변형은 다음과 같다.

$$y=y_0+\sum_{i=1}^{nd} d_i f_i \tag{5}$$

설계변수는 이들 Hicks-Henne 함수에 대한 weight인 d_i 로서 총 5개의 설계변수를 사용하였다.

격자변형을 위해서는 spring analogy 사용하였다. 그리고 RAE2822를 기본형상으로 해서, 윗면의 형상만을 최적화 대상으로 하였다.

3.3 최적설계 적용을 위한 AD 코드의 수정

실제 공력최적설계에 적용하기 위해서는 AD가 만든 코드에 약간의 손질이 필요하다. AD가 만든 코드에서 프로그램의 종결 조건은, 원래 프로그램을 따르므로 보통 flow variable의 수렴성 여부에 의해 결정된다. 따라서 이를 민감도의 수렴성에 따라 프로그램을 종결하도록 수정할 필요가 있다.

그리고 AD가 만든 코드는 실제 계산 부하가 크므로 되도록 민감도의 계산 횟수를 적게 하는 것이 계산시간을 단축시키게 해준다. 따라서 우선 flow variable이 어느 정도 수렴할 때까지는 원래 프로그램을 사용하고 수렴한 후엔 AD가 만든 코드를 사용해 민감도를 구해 계산 시간의 단축이 가능하다 [10,11].

3.4 2D 에어포일 형상 최적화 결과

최적화에는 ADIFOR, TAPENADE 등이 사용되었다. 여러 가지 다른 모드들에 대한 성능 비교를 위해, scalar forward mode, vector forward mode, 그리고 민감도의 계산이 delay된 vector forward mode 등에 대해 최적화를 수행하였다. scalar mode는 단일 변수에 대해서만 민감도를 주는 것이고, vector mode는 여러 변수에 대해 민감도를 주는 것이다. FD는 5개의 설계변수의 경우, 6번의 solver 계산이 필요하다. AD는 scalar gradient의 경우, 5개 설계변수에 대해 5번의 민감도 계산이 필요하고, vector gradient의 경우, 설계변수의 개수와 무관하게 1번의 계산으로 모든 민감도를 구할 수 있다.

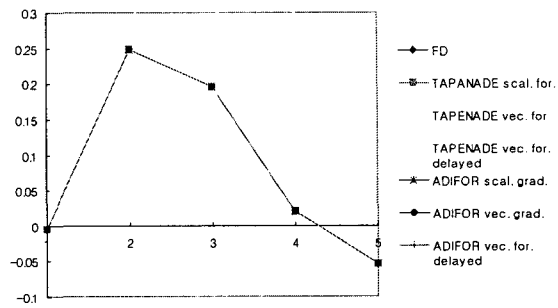

Fig. 1 Comparisons of sensitivities after the 1st iteration

우선 AD에서 구한 민감도의 정확도를 알아보기 위해 최적화 도중 1번째 iteration 후의 민감도를 비교하여 보았다. Fig. 1에서 보듯이 FD와 AD의 민감도는 서로 잘 일치함을 확인할 수 있다.

최적화 결과는 Table 5, 6에 표시하였다. 목적함수나 C_l, C_d는 서로 비슷한 값을 가지는 것을 알 수 있다. 그러나 계산 시간 면에서 AD가 더 오래 걸리는 것을 볼 수 있다. ADIFOR에 비해서는 TAPENADE의 계산 시간이 더 적게 걸린다. 이는 ADIFOR이 순수하게 forward mode만을 쓰는 것이 아니라 부분적으로 reverse mode를 채용하기 때문에 계산 시간과 결과값들에서 차이가 발생하는 것으로 보인다. scalar gradient보다는 vector gradient가 더 적은 계산 시간이 소요된다. 각 테이블에서 마지막 열은 각 AD tool들이 만든 코드의 성능 개선을 한 경우의 결과이다. 즉 3.3절에서 전술한 바와 같이 민감도의 계산을 flow variable의 수렴 전까지 지연 시킨 것으로, TAPENADE의 경우에선 FD와 거의 대등한 수준의 성능을 보인다. 그리고 Fig. 2에는 최적화 초기 형상인 RAE2822와 각 방법들을 사용했을 때의 압력계수를 나타내었다. ADIFOR과 TAPENADE는 scalar forward mode에 대해서만 나타내었다. 그림에서 보듯이 FD와 AD의 최적화 결과는 서로 잘 일치하며 모두 충격파를 없애고 있음을 알 수 있다.

Table 5 Optimization results for FD, ADIFOR

	FD	AD-ADIFOR		
		scal. for.	vec. for.	vec. for. delay.
Obj	6.0993e-4	6.0933e-4	6.0940e-4	6.0940e-4
C_l	0.8897	0.8900	0.8897	0.8897
C_d	0.0110	0.0110	0.0110	0.0110
Time(s)	411	1726	1415	910
Fun call	31	34	33	33
Grad call	8	9	8	8

Table 6 Optimization results for TAPENADE

	AD-TAPENADE		
	scal. for.	vec. for.	vec. for. delay.
Obj	6.0933e-4	6.0934e-4	6.0933e-4
C_l	0.8900	0.8902	0.8902
C_d	0.0110	0.0110	0.0110
Time(s)	1289	793	584
Fun call	34	37	37
Grad call	9	9	9

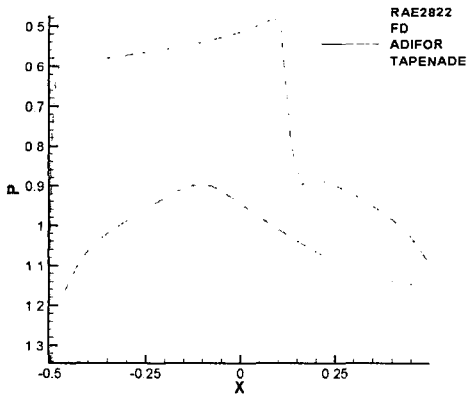

Fig. 2 Comparisons of pressure coefficients on the surface

4. 결 론

2D 공력최적설계에 AD tool을 적용하여 보았으며, 비교를 통해 FD와 일치하는 민감도를 제공하는 것을 알 수 있었다. 최적화에 대한 성능 비교에서는, 결과값은 FD와 큰 차이가 없지만, 계산 시간이 오래 걸리는 것을 알 수 있었다. AD tool에 따라 다르지만 많게는 최대 3배까지 계산 시간이 더 소요되었다. 그러나 본 연구에서와 같이 flow solver의 수렴 후 민감도의 계산을 시작하는 등의 수정을 통해 AD tool들이 만든 코드의 성능을 향상시킬 수 있음을 확인하였다.

참고문헌

[1] R. E. Wengert, "A Simple Automatic Derivative Evaluation Program", *Communications of the ACM*, Vol. 7, (1964), p.463-464.

[2] B. Speelpenning, Compiling Fast Partial Derivatives of Functions Given by Algorithms, PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, (1980).

[3] L.L. Green, P.A. Newman, and K.J. Haigler, " Sensitivity Derivatives for Advanced CFD Algorithm and Viscous Modeling Parameters via Automatic Differentiation", Jouranl of Computational Physics, Vol. 125, (1996), p.313-324.

[4] A. Carle, M. Fagan, and L.L. Green, " Preliminary Results from the Application of Automatic Adjoint Code Generation to CFL3D", *Proceedings of 7th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA-98-4807.

[5] C. Bishof et al., ADIFOR 2.0 User's Guide, Technical Report CRPC-95516-S, Rice University, (1998).

[6] T.C. Coleman, A. Verma, "ADMIT-1: automatic differentiation and MATLAB interface toolbox", *ACM Transactions on*

Mathematical Software, Vol. 26, (2000), p.150-175.

[7] R. Giering, T. Kaminski, "Recipes for Adjoint Code Construction", *ACM Transactions on Mathematical Software*, Vol. 24, (1998), p.437-474.

[8] http://www-sop.inria.fr/tropics/tapenade.html

[9] 박수형, 성춘호, 권장혁, "Upwind TVD 기법을 이용한 효율적인 다중격자 DADI기법", 항공우주학회, 추계학술대회발표 논문집, (1998), p.71-74.

[10] C. Bishof, G. Corliss, L. Green, A. Griewank, K. Haigler, P. Newman, "Autoamtic differentiation of advanced CFD codes for multidisciplinary design", *Journal of Computing Systems in Engineering*, Vol 3, (1992), p.625-638.

[11] H. Bücker et al., "Delayed Propagation of Derivatives in a Two-dimensional Aircraft Design Optimization Problem", *Proceedings of High Performance Computing Systems and Applications*, (2003).