Optimal control of fluid forces using second order automatic differentiation

Ayako Kamikawa* and Mutsuto Kawahara†
Department of Civil Engineering, Chuo University, 1-13-27, Kasuga, Bunkyo-ku, Tokyo, JAPAN

In recent years, the progress of computer and numerical computation technique allows not only complex numerical simulations but also resolution of the inverse analysis including optimal control problem. It is important to pursue high quality of the gradient computation in the inverse problem. Sometimes, it is called as the sensitivity analysis. In case of the inverse analysis, the minimization technique of a function is often used. Only first order variation of function is usually applied. But in order to minimize the functional exactly, it is also necessary to introduce the second order variation of the function. However, pursing variation is difficult to formulate and to program. There is a method of computation to pursue the variation of functional automatically, i.e., Automatic Differentiation (AD). The AD is implemented with C++ programs using operator overloading technique. It computes the partial derivatives according to the differentiation rule of a composite function whenever basic operation is performed. Using AD, the first order variation can be obtained. Ordinary AD is called First Order Automatic Differentiation (FOAD). In addition, AD can be extended to the Second Order Automatic Differentiation (SOAD) to obtain second order variation. In this study, both FOAD and SOAD are applied to the optimal control problem of fluid forces. The control problem uses the minimization technique of the function. The purpose of this study is to present the application of SOAD in control problem. The Sakawa-Shindo Method using FOAD and the Steepest descent method using SOAD are compared as the minimization technique. The automatic differentiation is proposed as a new approach to the sensitivity analysis of the optimal control problem.

1. INTRODUCTION
In the inverse analysis of the fluid flow, the minimization principle of the functional is defined, and then it is minimized by taking the variation. The minimization of the functional is carried out using partial derivatives. The

*Corresponding author.
†Email: kawa@civil.chuo-u.ac.jp
derivative of function can be obtained by computers. There is a method of computation to obtain derivatives of function, which is called as Automatic Differentiation (AD).

Automatic Differentiation can be applied to the sensitivity analysis of the optimal control problem, in which the gradient of the performance function $J$ of control values is required. As the minimization technique of the performance function $J$, the gradient method requires the partial derivative of $J$. Furthermore using the Steepest descent method as the minimization technique the second order partial derivative is required. It is called the Hessian. In many cases, the formulation of $J$ is so complex that calculation of the gradient of function $J$ by hand becomes difficult. However the usage of AD enables that not only first order derivatives but also second order derivatives can be obtained exactly and automatically in computation.

In this research, the sensitivity analysis with AD is applied to the optimal control of fluid forces around circular a cylinder, defined in a 2-dimensional space domain. The state equation is the Navier-Stokes equation of the incompressible viscous fluid. The optimal control problem is to reduce the fluid forces around circular a cylinder. The fluid forces are drag and lift forces. The control of fluid forces is implemented by giving the velocity at the control points on the cylinder. The purpose of this research is the investigation of the effectiveness of SOAD applied to the optimal control of fluid forces and the comparison between the Sakawa-Shindo method using FOAD with the Steepest descent method using SOAD.

2. DERIVATIVE COMPUTATION

2.1. Automatic Differentiation

Automatic Differentiation (AD) is the technique of computing the partial derivative of function. If the algorithm to compute value of function is given, the algorithm to compute partial derivative of the function is obtained with AD, which can be realized by the operator overloading function in programing language C++.

Basic operation $\psi$ is the operator (function) set by computational model $
abla \psi : \Omega \rightarrow R^n (\Omega \subset R^m)$, where

$$\psi (u) = [\psi_1 (u_1, \ldots, u_j), \ldots, \psi_m (u_1, \ldots, u_j)]$$

The elementary partial derivative $\psi_j$ of function $\psi$ at point $x$ is $\nabla \psi_j$. Figure 1 shows the relation between basic operator $\psi$, value $x$ and the elementary partial derivative $\nabla \psi_j (x)$. Under normal
In case that a basic operator is given as,

$$A = X \ast Y$$  \hspace{1cm} (1)$$

and the relation of those arguments is clearly expressed as follows.

$$A = \psi(u_1, u_2) \big|_{u_1 = X, u_2 = Y},$$  \hspace{1cm} (2)$$

where the elementary partial derivative of $\psi$ is

$$\frac{\partial \psi}{\partial u_1} \bigg|_{u_1 = X, u_2 = Y} = u_2 \bigg|_{u_1 = X, u_2 = Y} = Y,$$  \hspace{1cm} (3)$$

$$\frac{\partial \psi}{\partial u_2} \bigg|_{u_1 = X, u_2 = Y} = u_1 \bigg|_{u_1 = X, u_2 = Y} = X.$$  \hspace{1cm} (4)$$

Partial derivative of $y$ coming out from the usual programming language can be obtained. For example, $\nu = \psi(u_1, u_2)$ is defined, basic operator and elementary partial derivative is
\[ \nabla \psi = \begin{bmatrix} \frac{\partial \psi}{\partial u_1} \\ \frac{\partial \psi}{\partial u_2} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad (5) \]

\[ \nabla \psi = \begin{bmatrix} \frac{\partial \psi}{\partial u_1} \\ \frac{\partial \psi}{\partial u_2} \end{bmatrix} = \begin{bmatrix} u_2 \\ u_2 \end{bmatrix}, \quad (6) \]

\[ \nabla \psi = \frac{\partial \psi}{\partial u_1} = \cos u_1, \quad (7) \]

and so on.

### 2.2. Forward Mode Automatic Differentiation

Forward Mode Automatic Differentiation can compute the partial derivative automatically without construction of computational graph. Consider the computation of the derivative of \( P \) with respect to \( \{x_i\}_{i=1}^n \). Each \( x_i \) has gradient \( \text{grad} \ x_i \). Before doing computation, one has to define the gradient as an initial value:

\[ \text{grad} \ x_i = \begin{cases} 1 \text{ (independent)} \\ 0 \text{ (otherwise)} \end{cases}, \]

then, the gradient computation is expressed as:

\[ \text{grad} \ x_i = \sum \text{grad} \ S_\alpha \frac{\partial \psi_k}{\partial S_\alpha}, \quad (8) \]

where \( \psi_k \) is basic operation.

### 2.3. Second Order Derivative

The following figure shows the relation between differentiation and programing by algorithm. The function \( f \) approximates \( \bar{f} \) in programed by algorithm. If the value of \( \nabla \bar{f} \), \( \nabla \nabla \bar{f} \) cannot be computed directly, approximation of partial
derivative $\nabla f$ and $\nabla^2 f$ are required. When AD is applied, the programming of function can be obtained, as well as the exact differentiation solution can be calculated automatically. This is the advantage of AD.

3. STATE EQUATION AND DISCRETIZATION

3.1. State Equation

In this study, the incompressible Navier-Stokes equation is employed as a state equation. Let $\Omega$ denote the computational domain with boundary $\Gamma$, and suppose that the incompressible Navier-Stokes flow occupies $\Omega$. The Navier-Stokes equation can be expressed by the non-dimensional form as

$$\dot{u}_i + u_j u_{i,j} + p_j - \nu (u_{i,j} + u_{j,i})_{,j} = 0 \quad \text{in} \quad \Omega, \quad \text{(9)}$$

$$u_{i,i} = 0 \quad \text{in} \quad \Omega, \quad \text{(10)}$$

where, $u$, $p$ and $\nu$ are the velocity, pressure and the kinematic viscosity coefficient ($\nu = 1/Re$) respectively. Consider a typical problem described in figure 2, where a solid body $B$ with the boundary $\Gamma_B$, is laid in the external flow. Suppose that the boundary condition and initial values are given below

$$u_i = (\dot{u}_i, 0) \quad \text{on} \quad \Gamma_1, \quad \text{(11)}$$

$$u_2 = 0, \quad t_1 = 0 \quad \text{on} \quad \Gamma_2, \quad \text{(12)}$$

$$t_i = 0 \quad \text{on} \quad \Gamma_3, \quad \text{(13)}$$

![Figure 2. Flow domain.](image-url)
\[ u_i = 0 \quad \text{on} \quad \Gamma_B, \quad (14) \]

where

\[ t_i = \{ -p \delta_{ij} + v(u_{i,j} + u_{j,i}) \} n_j, \quad (15) \]

in which \( t_i \) is the traction vector and \( n_j \) is the unit vector of outward normal to \( \Gamma \). The fluid force acting on the body \( B \) is defined as \( F_i \), where \( F_1 \) and \( F_2 \) are the drag and the lift forces, respectively. The fluid force \( F_i \) is obtained by integrating the traction \( t_i \) on the boundary \( \Gamma_B \) as

\[ F_i = -\int_{\Gamma_B} t_i \, d\Gamma. \quad (16) \]

### 3.2. Temporal Discretization

Implicit solution that can make long temporal intervals and superior in stability is applied to temporal discretization. The Crank-Nicolson method is applied to the velocity of momentum equations. Continuity equation and the pressure of momentum equation are treated completely implicit as

\[ \frac{u_i^{n+1} - u_i^n}{\Delta t} + u_j^n u_{i,j} + \frac{1}{2} p_i^{n+1} - v(u_{i,j}^{n+\frac{1}{2}} + u_{j,i}^{n+\frac{1}{2}}) = 0, \quad (17) \]

\[ u_{i,i}^{n+1} = 0, \quad (18) \]

where

\[ u_i^{n+\frac{1}{2}} = \frac{u_i^{n+1} + u_i^n}{2} \quad (19) \]

### 3.3. Spatial Discretization

For the spatial discretization, the Galerkin method is applied and the mixed interpolation is used. Then, the bubble function element for the velocity and the linear element for the kinematic pressure are applied and expressed as follows:

for bubble function element,
\[ u_i = \Phi_1 u_{i1} + \Phi_2 u_{i2} + \Phi_3 u_{i3} + \Phi_4 \tilde{u}_{i4}, \quad \tilde{u}_{i4} = u_{i4} - \frac{1}{3}(u_{i1} + u_{i2} + u_{i3}), \quad (20) \]

\[ \Phi_1 = L_1, \quad \Phi_2 = L_2, \quad \Phi_3 = L_3, \quad \Phi_4 = 27L_1L_2L_3, \quad (21) \]

and, for linear element,

\[ p = \Psi_1 p_1 + \Psi_2 p_2 + \Psi_3 p_3, \quad (22) \]

\[ \Psi_1 = L_1, \quad \Psi_2 = L_2, \quad \Psi_3 = L_3, \quad (23) \]

where \( \Phi_{\alpha} (\alpha = 1, 2, 3, 4) \) is the bubble function element for the velocity, \( \Psi_{\lambda} (\lambda = 1, 2, 3) \) is linear element for the kinematic pressure, in which \( L_i \) is the linear interpolation function.

The criteria for the steady problem is used, in which the discretized form discretization of the bubble function element is equivalent to those of SUPG (Streamline-Upwind/Petrov-Galerkin) method. In the bubble function element for the steady problem, the stabilized parameter \( \tau_{eB} \) determines the magnitude of the stream-line stabilized term. The stabilized parameter \( \tau_{eB} \) is expressed as

\[ \tau_{eB} = \frac{< \phi_e, 1 >_{\Omega_c}^2}{\nu \| \phi_e \|_{\Omega_c}^2 A_e}, \quad (24) \]

where \( \Omega_c \) is element domain and

\[ <u, v>_{\Omega_c} = \int_{\Omega_c} u v d\Omega_c, \quad \| u \|_{\Omega_c}^2 = \int_{\Omega_c} u^2 d\Omega_c, \quad A_e = \int_{\Omega_c} d\Omega_c, \quad (25) \]

the integral of bubble function is expressed as follows:

\[ <\phi_e, 1>_{\Omega_c} = \frac{A_e}{6}, \quad \| \phi_e \|_{\Omega_c}^2 = 2A_e g, \quad g = \sum_{i=1}^{2} |\Psi_{\alpha,i}|^2, \quad (26) \]

from the criteria for the stabilized parameter in SUPG, an optimal parameter \( \tau_{eS} \) can be given as follows:
where $h_e$ is an element size. In general, the stabilized parameter (eq.(24)) does not equal to the optimal parameter (eq.(27)). Thus, the bubble function which gives the optimal viscosity $\nu'$ satisfies.

\[
\frac{<\phi,1>^2_{\Omega_e}}{(v + \nu')||\phi_{.j}||_{\Omega_e}^2} = \tau_v B_e. \tag{28}
\]

It is shown that in eq.(28) that only the stabilized operator control term only at the center point of the gravity has been added to the equation of motion,

\[
\sum_{e=1}^{N_e} \nu' ||\phi_{e,j}||_{\Omega_e}^2 b_e, \tag{29}
\]

where $N_e$ and $B_e$ are the total number of elements and center point of the gravity, respectively.

**4. CONTROL PROBLEM**

**4.1. Problem**

Inverse problem such as control problem finally arrives at the minimization problem of the performance function $J$ using optimal control theory. The performance function $J$ is defined by the squaring the residual between computed and objective fluid forces which can be written, i.e.

\[
J = \frac{1}{2} \int_{\Omega} (F - F^*)^T (F - F^*) d\Omega, \tag{30}
\]

\[
F = (f_1, f_2^{*}, \cdots, f_n)^T, \tag{31}
\]

\[
F^* = (f_1^*, f_2^{*}, \cdots, f_n^*)^T, \tag{32}
\]
where $F$ is the computed value and $F^*$ is the objective value. The problem can be written as to find control force so as well as to minimize the performance function $J$.

### 4.2. Gradient Method

The gradient methods require the first order derivative of performance function $J$ to obtain the new control value. The Sakawa-Shindo method is one of the minimization techniques. The modified performance function $K$ is defined as

$$
K = J + \frac{1}{2} \int_{\Omega} (U_{ct}^{(l+1)} - U_{ct}^{(l)})^T W^{(l)} (U_{ct}^{(l+1)} - U_{ct}^{(l)}) \, d\Omega,
$$

(33)

where $W^{(l)}$ is the weighting constants, $l$ is the iteration number. The condition that first order variation of modified performance function is equal to zero is required. Control value $U_{ct}$ is renewed by the following equation as

$$
U_{ct}^{(l+1)} = U_{ct}^{(l)} - \frac{\partial J}{\partial U_{ct}} / W.
$$

(34)

Here the partial derivative of performance function $J$ with respect to control value $U_{ct}^{(l)}$ should be computed. In many cases the formal description of $J$ is so complex that an analytic computation of $\frac{\partial J}{\partial U_{ct}^{(l)}}$ by hand is a formidable task.

### 4.3. Steepest Descent Method

The Steepest descent method is one of the minimization algorithm in unconstrained optimization. In steepest descent method, rapid convergence of solution can be obtained using the second order derivative. If the value $\phi^{(l)}$ is given, the point $\phi^{(l+1)}$ is obtained by

$$
\phi^{(l+1)} = \phi^{(l)} - \alpha g,
$$

(35)

$$
\alpha = \frac{g^T g}{g^T H g},
$$

(36)

In the Steepest descent method, $\phi^{(l+1)}$ is renewed using direction $g$, which gradient $\nabla J(\phi)$, $\alpha$ is step size and $H$ is Hessian matrix $\nabla^2 J(\phi)$. In order to determine $\alpha$, first order derivative $g$ and second order derivative $H$ are required.
5. NUMERICAL STUDY
5.1. Computational Condition
In this study, optimal control of fluid forces around circular a cylinder is analyzed using FOAD and SOAD. Considering the computational model and the finite element mesh applied are shown in figure 3 and figure 4. Control of the drag and lift forces reduction problem of a cylinder located in the viscous flow is analyzed. Flow with constant velocity is imposed on the upwind boundary condition $\Gamma_1$. The total number of nodes and elements are 1636 and 3116, respectively. The Reynolds number applied 250. Figure 6 shows the positions of five control velocity points. Total number of control points are 5 as shown in figure 6. Figure 5 illustrates the time history of drag and lift forces. To reduce drag and lift forces, control velocities are given from direction of $x$ and $y$. For
Figure 5. Time history of fluid forces without control.

Figure 6. Position of control velocity.

Figure 7. Velocity vector (without control).
Figure 8. Pressure distribution (without control).

Figure 9. Velocity vector (with control).

Figure 10. Pressure distribution (with control).
Figure 11. Time history of the drag force.

Figure 12. Time history of the lift force.
the control velocity, only on $x$ direction is given at the center of the control point, where $D, L$ are the values of computed drag and lift forces, $q_1$ and $q_2$ are the values of weight caused by $D$ and $L$, respectively. In this research, $q_1$ and $q_2$ have been defined to be both 1.0. The performance function is expressed as
5.2. Numerical Results

Numerical results are shown in figure 7–14. Figure 7 and 8 show the velocity vector and the pressure distribution without control, respectively. The time history of drag and lift forces without control is illustrated in Figure 5. To reduce drag and lift forces, control is started in non-dimensional time 40 when forces become in periodic cycle. Figure 9 and 10 represent the controlled velocity vector and the controlled pressure distribution, respectively. figure 11 is the time history of drag force with control. Figure 12 shows the time history of lift force with control. Figure 13 illustrates the history of the performance function by the Sakawa-Shindo method using FOAD and the Steepest descent method using SOAD. The control velocity vector around circular cylinder is pointed in figure 14.

6 CONCLUSION

In this study, the Sakawa-Shindo method using FOAD and the Steepest descent method using SOAD have been applied to the optimal control of fluid forces on the circular cylinder using the finite element method. The drag force has reduced by 50%. It can be said that in order to control the fluid forces, by giving the control velocity from the circular cylinder is one of the effective method. Computational time of the Sakawa-Shindo method using FOAD is around 80 hours, and that of the steepest descent method using SOAD is about 17 hours. From this results, as a minimization technique, the performance function can be reduced more quickly using the steepest descent method by SOAD than by the Sakawa-Shindo method as the gradient method by FOAD. It is shown that the second order partial derivative of the performance function can be computed exactly using SOAD. Therefore, it can be concluded that SOAD is more effective than FOAD on the optimal control problem of this type of problems. In case that the numerical value of the second order derivative can be obtained exactly using the Automatic Differentiation, the Steepest descent method is shown to be much more effective computational technique.

\[
J = \frac{1}{2} \int (q_1 D^2 + q_2 L^2) dt. 
\]  
(37)

|                | FOAD | SOAD |
|----------------|------|------|
| Computational time | 80h  | 17h  |
REFERENCES

[1] J. Matsumoto, T. Uetsu and M. Kawahara, *Incompressible Viscous Flow Analysis and Adaptive Finite Element Method Using linear Bubble Function*, Journal of Applied Mechanics, vol. 2. 223–232 (1999).

[2] O. Pironneau, *Consist Approximations, Automatic Differentiation and Domain Decomposition For Optimal Shape Design*, International Series, Mathematical Sciences and Applications vol.16 (2001), Computational Methods for Control Application, pp. 167–178.

[3] J. Matsumoto, T. Uetsu and M. Kawahara, *Stable Shape Identification for Fluid-Structure Interaction Problem using MINI Element Method*, Journal of Applied Mechanics, vol. 3. 263–274.

[4] Shao-Liang ZHANG and Senji FUJINO, *A Class of Product-type Methods Based on Lanczos Process*, Journal of Applied Mathematics and science, vol. 5, No.4, 1995, pp. 343–360.

[5] Claus Bendtsen FADBAD, *A flexible C++ package for automatic differentiation*, 1 No 1996–x5-94 August 15, 1996 OS.