Inverse Problems, Design and Optimization Symposium
Rio de Janeiro, Brazil, 2004

IDENTIFICATION OF REYNOLDS NUMBER USING AUTOMATIC DIFFERENTIATION

Yuya Takahashi
Department of Civil Engineering
Chuo University
Bunkyo-ku, Tokyo, Kasuga
u-ya@kc.chuo-u.ac.jp

Mutsuto Kawahara
Department of Civil Engineering
Chuo University
Bunkyo-ku, Tokyo, Kasuga
kawa@civil.chuo-u.ac.jp

ABSTRACT

The progress of computer and numerical technique in recent years allows us not only complex numerical simulation but also resolution of inverse problems. It is important to pursue higher and higher quality of gradient computation (it is called sensitivity analysis) in inverse problem and this is the most bone-crushing point. In this study, the authors will propose the forward mode automatic differentiation as a new approach to the parameter identification problems. Forward mode automatic differentiation computes the partial derivatives according to the differentiation rules of a composite function whenever basic operation is performed.

INTRODUCTION

The Reynolds number is named after Osborne Reynolds (1842-1912) who conducted an experimental study to see how and when laminar and turbulent flows occur through a pipe. The Reynolds number is a dimensionless parameter and is defined as

$$\mathrm{Re} = \frac{\rho U L}{\mu} \tag{1}$$

where ρ is fluid density, μ is viscosity coefficient, *U* is characteristic velocity and *L* is characteristic length scale. The Reynolds number is important in analyzing any type of flow when there is substantial velocity gradient. It indicates the relative significance of the viscous effect compared to the inertia effect.

In this paper, the authors will show the Reynolds number identification using forward mode automatic differentiation. Derivatives of functions can be computed exactly not only by hand but also by computers. The differentiation rules are defined for each operation in computation. Thus if a function is described by its computer implementation, it can be differentiated exactly and automatically by overloading operators. This technique is called automatic differentiation. Some packages for AD are already available[1], [2].

PARAMETER IDENTIFICATION

Inverse problems such as parameter identification finally arrive at minimization problem of performance function *J* using optimal control theory. The performance function *J* is defined by the square sum of residual between computed value and observed data, which can be written as follows:

$$J(\mathrm{Re}) = \frac{1}{2}\int_T \int_\Omega (\boldsymbol{u} - \boldsymbol{u}^*)^T (\boldsymbol{u} - \boldsymbol{u}^*) d\Omega dt \tag{2}$$

$$\boldsymbol{u} = (u_1, u_2, \cdots, u_n)^T \tag{3}$$

$$\boldsymbol{u}^* = (u_1^*, u_2^*, \cdots, u_n^*)^T \tag{4}$$

where $\boldsymbol{u}$ are computed value and $\boldsymbol{u}^*$ are observed data. To find $\boldsymbol{u}$ so as to minimize performance function *J*, gradient methods are very efficient. The Sakawa-Shindo method [7] is one of the minimization techniques. In this method, the modified performance function *K* is introduced adding penalty term to the performance function. The modified performance function is expressed as follows:

$$K^l = J^l + \frac{1}{2}\int_T \int_\Omega (\mathrm{Re}^{l+1} - \mathrm{Re}^l) W^l (\mathrm{Re}^{l+1} - \mathrm{Re}^l) d\Omega dt \tag{5}$$

where *l* is iteration number for minimization, W^l is the weighting constant, which is defined arbitrary. In this study, weighting constant is 50. ***Re*** is renewed by the following equation:

$$\mathrm{Re}^{l+1} = \mathrm{Re}^{l} - \frac{\partial J}{\partial \mathrm{Re}} / W^{l} \tag{6}$$

which can be obtained from stationary condition of modified performance function $\partial K / \partial \mathrm{Re} = 0$. Here we have to compute partial derivative of performance function *J* with respective to Reynolds number. In this case, performance function is described by finite element equations and analytic computation of partial derivative is a formidable task.

STATE EQUATIONS

In this study, Navier-Stokes equations of incompressible flow are employed as state equations which are expressed as follows;

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \nabla p - \nu\nabla\cdot(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^{T}) = \boldsymbol{f} \ \text{ in } \Omega \tag{7}$$

$$\nabla\cdot\boldsymbol{u} = 0 \ \text{ in } \Omega \tag{8}$$

where ν is the inverse of Reynolds number. This inverse of Reynolds number will be identified in this research using forward mode automatic differentiation according to optimal control theory. The initial condition is given as follows;

$$\boldsymbol{u}(t_0) = \boldsymbol{u}_0 \tag{9}$$

Boundary condition is divided into Dirichlet and Neumann boundary condition which are expressed as follows:

$$\boldsymbol{u} = \hat{\boldsymbol{u}} \ \text{ on } \Gamma_1 \tag{10}$$

$$-p\boldsymbol{\delta} + \nu\{\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^{T}\}\cdot\boldsymbol{n} = \hat{\boldsymbol{t}} \ \text{ on } \Gamma_2 \tag{11}$$

DISCRETIZATION

Temporal Discretization

Implicit solution, which can make large temporal space and superior in stability, is applied to temporal discretization. A Crank-Nicolson method is applied to momentum equations and continuity equation is treated completely implicit as follows:

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^{n}}{\Delta t} + \overline{\boldsymbol{u}}\cdot\nabla\boldsymbol{u}^{n+\frac{1}{2}} + \nabla p^{n+1} - \nu\nabla\cdot\{\nabla\boldsymbol{u}^{n+\frac{1}{2}} + (\nabla\boldsymbol{u}^{n+\frac{1}{2}})^{T}\} = \boldsymbol{f} \tag{12}$$

$$\nabla\cdot\boldsymbol{u}^{n+1} = 0 \tag{13}$$

where

$$\boldsymbol{u}^{n+\frac{1}{2}} = \frac{\boldsymbol{u}^{n+1} + \boldsymbol{u}^{n}}{2} \tag{14}$$

$$\overline{\boldsymbol{u}} = \frac{1}{2}(3\boldsymbol{u}^{n} - \boldsymbol{u}^{n-1}) \tag{15}$$

Fractional Step Method and Spatial Discretization

The Navier-Stokes equation can be solved by the fractional step method, by which flow and pressure fields are separated by deriving the pressure Poisson equation from the momentum and continuity equations. The pressure Poisson equation is derived introducing an intermediate velocity, which may not satisfy the continuity equation. The Galerkin formulation is used in space.

$$\frac{\tilde{\boldsymbol{u}}^{n+1} - \boldsymbol{u}^{n}}{\Delta t} + \overline{\boldsymbol{u}}\cdot\nabla\tilde{\boldsymbol{u}}^{n+\frac{1}{2}} + \nabla p^{n} - \nu\nabla\cdot\{\nabla\boldsymbol{u}^{n+\frac{1}{2}} + (\nabla\boldsymbol{u}^{n+\frac{1}{2}})^{T}\} = \boldsymbol{f} \tag{16}$$

$$\Delta t\nabla^{2}(p^{n+1} - p^{n}) = \nabla\cdot\tilde{\boldsymbol{u}}^{n+1} \tag{17}$$

$$\frac{\boldsymbol{u}^{n+1} - \tilde{\boldsymbol{u}}^{n+1}}{\Delta t} + \frac{1}{2}\overline{\boldsymbol{u}}\cdot\nabla(\boldsymbol{u}^{n+1} - \tilde{\boldsymbol{u}}^{n+1}) + \nabla(p^{n+1} - p^{n}) - \frac{1}{2}\nu\nabla\cdot\{(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^{T})^{n+1} - (\nabla\tilde{\boldsymbol{u}} + (\nabla\tilde{\boldsymbol{u}})^{T})^{n+1} = 0 \tag{18}$$

The mixed interpolation is applied to the spatial discretization. A bubble element adding a bubble function to a bilinear element is employed as an interpolation for the flow field. A linear element is employed as an interpolation for the pressure field.

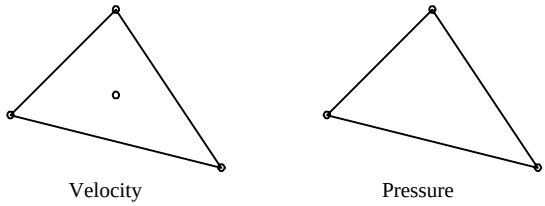

Fig. 1 Mixed interpolation

Stabilized Bubble Function

The stabilized bubble function method is developed by J. Matsumoto et al.[9] In this method, the criteria for the steady problem is used, in which the discretized form derived from the bubble element is equivalent to those from the SUPG in case each shape of element. Thus, the stabilization parameter τ_{eB} of the SUPG method is used. In the bubble element for the steady problem, the magnitude of the streamline stabilization term for a bubble function is expressed by the stabilization parameter τ_{eB} defined as:

$$\tau_e = \frac{\{\int_{\Omega_e} \phi_e d\Omega\}^2}{(\nu + \nu')\int_{\Omega_e} (\nabla \phi_e)^2 d\Omega V_e} \qquad (19)$$

where V_e is the element volume, ϕ_e is the bubble function defined on each element.

DERIVATIVE COMPUTATION

Forward Mode

Forward mode automatic differentiation can compute partial derivatives automatically without constructing computational graph. Therefore, less computational storage is required comparing with reverse mode automatic differentiation. It computes partial derivatives from input variables to output variables.

Algorithms are made from unary operations (including mathematical functions) and binary operations. Partial derivatives of the basic operations are known.

When we want to know partial derivatives of *f* with respect to $x_i (1 \le i \le n)$. We make temporal variable *S(v)* for each intermediate variable *v*. First of all initialize the temporary variables choosing arbitrary independent variable x_j;

$$S(x_i) = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases} \qquad (20)$$

Thus the gradient of the k-th instruction v_k can be obtained from the process shown below:

For $v_k = \psi_k(v_\alpha, v_\beta)$

$$S(v_k) \leftarrow S(v_\alpha)\frac{\partial \psi_k}{\partial v_\alpha} + S(v_\beta)\frac{\partial \psi_k}{\partial v_\beta} \qquad (21)$$

where v_α and v_β are input variables, intermediate variables or constants, ψ_k is basic operation (in this case, this is binary operation).

Finally, the computation of (21) for the last instruction v_k will provide us with the partial derivative of *f* with respective to x_j.

Implementation

Implementation of forward mode automatic differentiation is available from overloading operators.

$$\psi_{\text{arithmetc}} = \{+, -, *, /, +(\text{unary}), -(\text{unary}), ++, --\} \qquad (22)$$

$$\psi_{\text{relational}} = \{==, !=, <, >, <=, >=\} \qquad (23)$$

$$\psi_{\text{math}} = \{\sin, \cos, \log, \cdots\} \qquad (24)$$

(22) is the set of arithmetic operators, (23) is the set of relational operators, and (24) is the set of mathematical functions defined in C++. These operators will be overloaded partial derivatives of elementary operations. New class for forward mode automatic differentiation will be defined as **list** 1.

List 1: Forward mode automatic differentiation class

```
class fdouble {
    protected:
        double val;
        double *d_val;
        ...
    public:
        ...
}
```

As shown in **list** 1, this class has two data; the first one is the value val, the second one is the vector d_val of partial derivatives. The size of vector will be as same number as independent variables. To access to the value and the i-th partial derivatives, the member function val() and dx(int i) are used respectively. Operator overloading for multiplication is shown in **list** 2.

List 2: Operator overloading for multiplication

```
operator*(const fdouble &x, const fdouble &y) {
//a temporary array is created here
   fdouble tmp(x.size());
   for(int i = 0; i < tmp.size(); i++) //loop
      tmp.dx(i) = x.dx(i) * y.val() + x.val() * y.dx(i);
   tmp.val() = x.val() * y.val();
   return tmp;
}
```

NUMERICAL EXAMPLE

As numerical examples, identification of Reynolds numbers in cavity flow are performed. Observing flow velocity on 4 points of nodes, Reynolds number is identified using forward mode automatic differentiation. As the observed data, computed results of cavity flow are used. Finite element mesh and observation points are shown in **Fig.** 2. The finite element mesh contains 1089 nodes and 2048 elements.

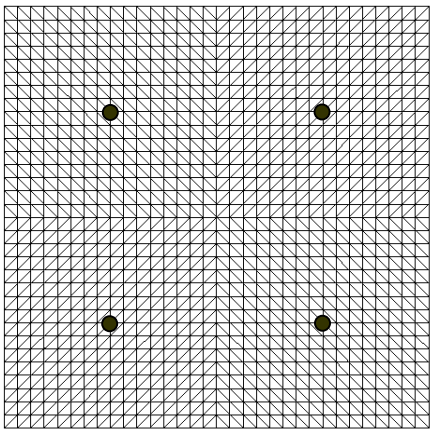

Fig. 2 Finite element mesh

The performance function is written as follows,

$$J(\mathrm{Re}) = \int_T \int_\Omega \frac{1}{2}(\boldsymbol{u} - \boldsymbol{u}^*)^T (\boldsymbol{u} - \boldsymbol{u}^*) d\Omega dt \quad (25)$$

where $\boldsymbol{u}$ is computed value and $\boldsymbol{u}^*$ is observed data. The problem is to find the Reynolds number so as to minimize the performance function **eq.** (25). We employ FADBAD 2.0[1] for automatic differentiation class library, which is a C++ program package for the forward and backward automatic differentiation.

The minimization iteration is computed shown in **fig. 3**

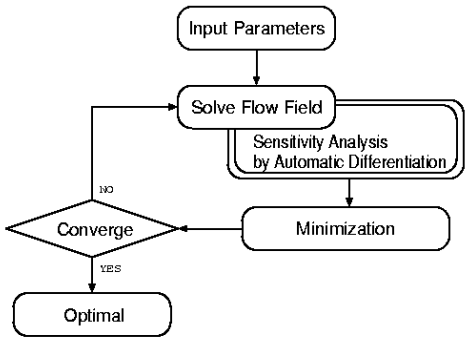

Fig. 3 Minimization algorithm

Case 1 (Re=1.0)

Observed velocity is shown in **fig.** 4. Δt is set at 0.01 and total time step is 5. Initial guesses of inverse of Re number are set at 0.8 (case 1(1)) and 1.2 (case 1(2)).

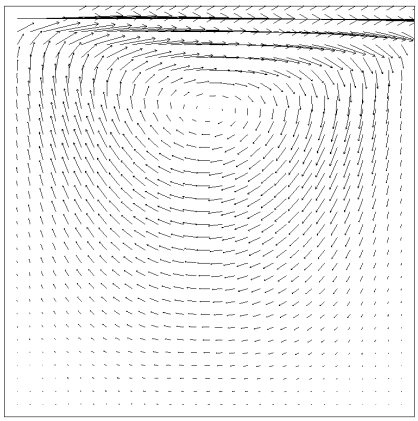

Fig. 4 Observed velocity (Re=1.0)

In both cases 1(1) and 1(2), performance function is converged to zero and target Reynolds number can be obtained as shown in **figs.** 5-8. In **fig.** 6, inverse of Reynolds number is converged from initial guess, 0.8, to 1.0 and also in **fig.** 8, it

is converged from initial guess, 1.2. It can be said that it is independent from initial guesses.

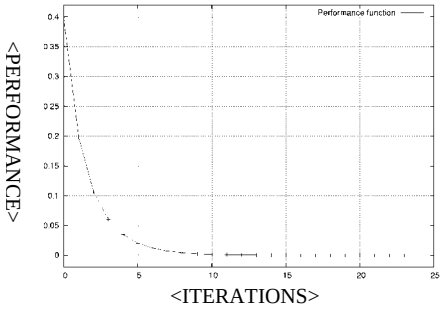

Fig. 5 Performance function (case 1(1))

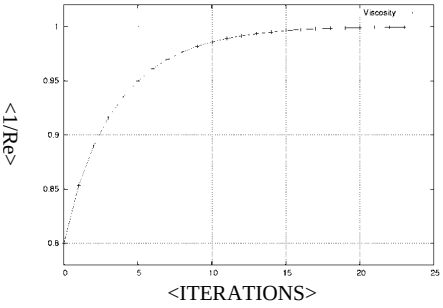

Fig. 6 Convergence of 1/Re (case 1(1))

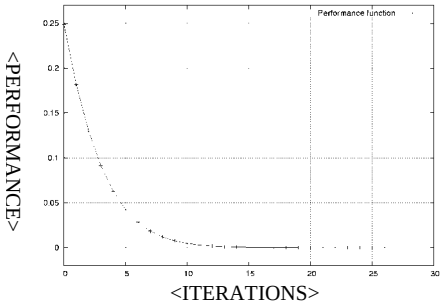

Fig. 7 Performance function (case 1(2))

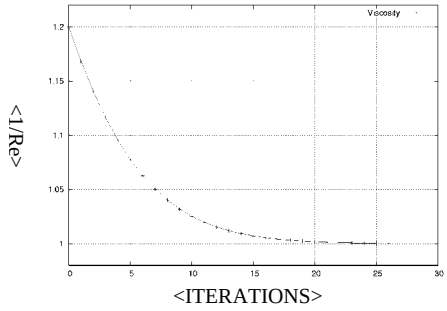

Fig. 8 Convergence of 1/Re (case 1(2))

Case 2 (Re=400)

Observed velocity is shown in **fig.** 9. Δt is set at 0.01 and total time step is 100. Initial guess of Reynolds number is set at 200(1/Re = 0.005).

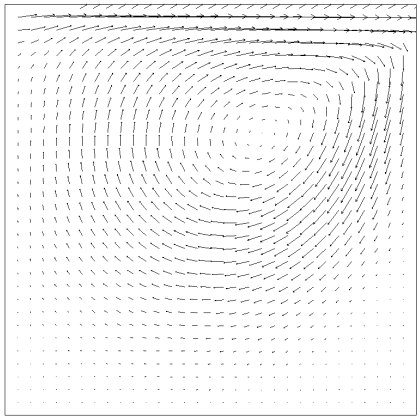

Fig. 9 Observed velocity (Re=400)

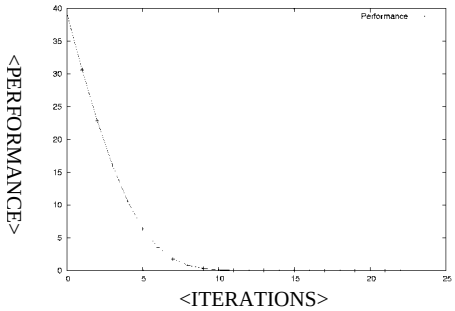

Fig. 10 Performance function (case 2)

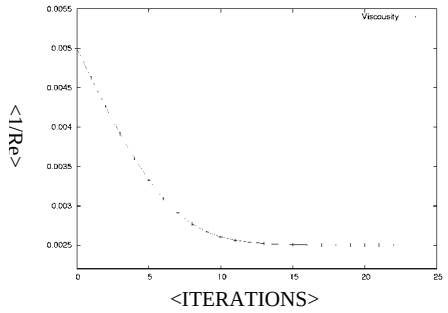

Fig. 11 Convergence of 1/Re (case 2)

The performance function is converged to zero as shown in **fig.** 10, which means that the residual between computed velocity and observed data is zero and the Reynolds number is identified as plotted in **fig.** 11.

As the results, the Reynolds number converged at correct values in both cases.

CONCLUSIONS

The main purpose of this study is to investigate efficiency of automatic differentiation for identification problems, using Navier-Stokes equations as the flow model. It can be said from the numerical example that sensitivity analysis is carried out without any implementation or discretization because sensitivity is automatically obtained from the AD. Thus development of applications for inverse problems can be significantly simplified.

In addition to that the sensitivity of the implicit system is also computed correctly using the AD. The advantages of automatic differentiation are that we need not to discretize adjoint systems, which will be very complex, and that we can concentrate on solving flow fields.

However the CPU time for the AD is much longer than for the adjoint approach because the AD requires N times loop on every operation as shown in **list** 2 if there are N indepen dent variables. But the hand working time for adjoint approach cannot be ignored. And when the problem is changed, adjoint system has to be re-built and implemented. On the other hand, required sensitivity can be easily obtained using the AD by just changing the declaration for independent variables in program.

REFERENCES

1. Claus Bendtsen, Ole Stauning, *FADBAD, a flexible C++ package for automatic differentiation*, TECHNICAL REPORT IMM-REP-1996-17, J. No. 1996-x5-94 August 15, 1996 OS

2. Andreas Griewank, David Juedes, Hristo Mitev, Jean Utke, Olaf Vogel and Andrea Walther, *ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++*, ACMTOMS, vol. 22(2), June 1996, pp. 131-167, Algor. 755.

3. Masao IRI, *Automatic Differentiation in Sensitivity Analysis and Optimization – Computational complexity, guaranteed interval of variation and the role of adjoint systems*, The 44th Nat. Cong. Of Theoretical & Applied Mechanics, 1995

4. Olivier Pironneau, Nicolas Dicesare, *CONSISTENT APPROXIMATIONS, AUTOMATIC DIFFERENTIATION AND DOMAIN DECOMPOSITION FOR OPTIMAL SHAPE DESIGHN*, GAKUTO International Series, Mathematical Sciences and Applications Vol. 16 (2001), Computational Methods for Control Application, pp. 167-178

5. Pierre Aubert, Nicolas Di Césaré, Olivier Pironneau, *Automatic Differentiation in C++ using Expression templates and Application to a Flow Problem*, Computing and Visualization in Sciences, 2000

6. Maruoka A., Kawahara M., Anju A., *A Fractional Step Finite Element Analysis of Incompressible Navier-Stokes Equation*, Proc. Of the 5th Int. Symp. on Computational Fluid Dynamics, Vol I pp. 19-26, Sendai 1993

7. Y. Sakawa Y. Shindo, *On global comvergence of an algorithm for optimal control*, Transactions on automatic control, IEEE, AC-25(6), pp.1149-1153, 1980

8. A. Maruoka, M. Marin and M. Kawahara, *Optimal control in Navier Stokes equations*, IJCFD, 9, pp. 313-322

9. J. Matsumoto and M. Kawahara, *Shape Identification for Fluid-Structure Interaction Problem Using Improved Bubble Element*, IJCFD, vol. 15, pp. 33-45, 2001