INVERSE PROBLEMS INFINITE ELEMENT METHOD
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

\[ Re = \frac{\rho UL}{\mu} \]  

(1)

where \( \rho \) is fluid density, \( \mu \) 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:

\[ J(Re) = \frac{1}{2} \int \int_\Omega (u - u^*)^T (u - u^*) d\Omega dt \]  

(2)

\[ u = (u_1, u_2, \cdots, u_n)^T \]  

(3)

\[ u^* = (u_1^*, u_2^*, \cdots, u_n^*)^T \]  

(4)

where \( u \) are computed value and \( u^* \) are observed data. To find \( u \) 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:

\[ K' = J' + \frac{1}{2} \int \int_\Omega (Re'^{i+1} - Re'^i W'(Re'^{i+1} - Re'^i) d\Omega dt \]  

(5)

where \( l \) is iteration number for minimization, \( W' \) is the weighting constant, which is defined arbitrary. In this study, weighting constant is 50. \( Re \) is renewed by the following equation:
\[ \text{Re}^{t+1} = \text{Re} - \frac{\partial J}{\partial \text{Re}} / W^t \]  

(6)

which can be obtained from stationary condition of modified performance function \( \partial K / \partial \text{Re} = 0 \). 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;

\[ \partial u / \partial t + u \cdot \nabla u + \nabla p - \nu \nabla \cdot (\nabla u + (\nabla u)^T) = f \text{ in } \Omega \]  

(7)

\[ \nabla \cdot u = 0 \text{ in } \Omega \]  

(8)

where \( \nu \) 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;

\[ u(t_0) = u_i \]  

(9)

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

\[ u = \hat{u} \text{ on } \Gamma_1 \]  

(10)

\[ -p\delta + \nu(\nabla u + (\nabla u)^T) \cdot n = \hat{i} \text{ on } \Gamma_2 \]  

(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:

\[ \frac{u^{*+1} - u^*}{\Delta t} + \tilde{u} \cdot \nabla u^{*+1/2} + \nabla p^{*+1} \]  

\[ -\nu \nabla \cdot \{\nabla u^{*+1/2} + (\nabla u^{*+1/2})^T\} = f \]  

(12)

\[ \nabla \cdot u^{*+1} = 0 \]  

(13)

where

\[ u^{*+1/2} = u^{*+1} + u^* \]  

(14)

\[ \tilde{u} = \frac{1}{2}(3u^* - u^{*+1}) \]  

(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.

\[ \frac{\tilde{u}^{*+1} - u^*}{\Delta t} + \tilde{u} \cdot \nabla u^{*+1/2} + \nabla p^n \]  

\[ -\nu \nabla \cdot \{\nabla u^{*+1/2} + (\nabla u^{*+1/2})^T\} = f \]  

(16)

\[ \Delta \nabla^2 (p^{*+1} - p^*) = \nabla \cdot \tilde{u}^{*+1} \]  

(17)

\[ \frac{u^{*+1} - \tilde{u}^{*+1}}{\Delta t} + 1/2 \tilde{u} \cdot \nabla (u^{*+1} - \tilde{u}^{*+1}) + \nabla (p^{*+1} - p^*) \]  

\[ -\frac{1}{2} \nu \nabla \cdot (\nabla u + (\nabla u)^T)^{*+1} - (\nabla \tilde{u} + (\nabla \tilde{u})^T)^{*+1} = 0 \]  

(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.
Stabilized Bubble Function

The stabilized bubble function method is developed by J. Matsumoto et al. 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 \( \tau_{eb} \) 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 \( \tau_e \) defined as:

\[
\tau_e = \frac{\left\{ \int_{\Omega} \phi_e d\Omega \right\}^2}{(\nu + \nu') \int_{\Omega} (\nabla \phi_e)^2 d\Omega V_e}
\]  

(19)

where \( V_e \) is the element volume, \( \phi_e \) 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 \( x_i \) \((1 \leq i \leq n)\). We make temporal variable \( S(v) \) for each intermediate variable \( v \). First of all initialize the temporary variables choosing arbitrary independent variable \( x_j \):

\[
S(x_j) = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}
\]  

(20)

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

\[
S(v_k) \leftarrow S(v_k) + S(v_a) \frac{\partial \psi_i}{\partial v_a} + S(v_p) \frac{\partial \psi_i}{\partial v_p}
\]  

(21)

where \( v_a \) and \( v_p \) are input variables, intermediate variables or constants, \( \psi_i \) 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.

\[
\psi_{arith} = \{+,-,*,/,(unary),-(unary),++,--\}
\]  

(22)

\[
\psi_{relational} = \{==,!==,\lt,\lt=,\gt,\gt=\}
\]  

(23)

\[
\psi_{arith} = [\sin,\cos,\log,\cdots]
\]  

(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**

```cpp
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.
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.

\[
J(Re) = \int_0^T \int_\Omega \frac{1}{2} (u - u^*)^T (u - u^*) d\Omega dt \tag{25}
\]

where \( u \) is computed value and \( u^* \) is observed data. The problem is to find the Reynolds number so as to minimize the performance function eq. (25). We employ FABBAD 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.

Case 1 (Re=1.0)

Observed velocity is shown in Fig. 4. \( \Delta t \) 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)).

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.

Case 2 (Re=400)
Observed velocity is shown in fig. 9. \( \Delta t \) is set at 0.01 and total time step is 100. Initial guess of Reynolds number is set at 200(1/Re = 0.005).
REFERENCES


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


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


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 seen 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 independent variables. But the hand working time for adjoint approach cannot be ignored. And when the problem is changed, adjoint system has to be rebuilt 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.