Object-oriented Development of Computer Code for Inverse Heat Conduction Problem

Sun Kyoung Kim[1,#]

1 College of Engineering, Seoul National University of Technology, Seoul, South Korea
Corresponding Author / E-mail: sunkkim@snut.ac.kr, TEL: +82-2-970-6394, FAX: +82-2-976-5173

KEYWORDS : Inverse heat conduction problem(IHCP), Object-oriented development (OOP), Unified modeling language (UML)

This paper suggests a method for developing computer code that can solve inverse heat conduction problem. The concept of the object-oriented development is employed to implement the computer code in an efficient and flexible fashion. The software design is conducted based on the unified modeling language. Furthermore, this paper also explains how to implement the deliverable computer code using the existing software development tools.

Manuscript received: September 29, 2005 / Accepted: June 30, 2006

NOMENCLATURE

J = residual
L = Lagrangian
N = number of measurement data
R = regularization term
T_i = calculated temperature at i-th sensor location
Y_i = measured temperature at i-th sensor location
α = regularization parameter
λ = Lgrange multiplier
σ_i = standard deviation of measured temperature

1. Introduction

Inverse heat conduction problem (IHCP) has been thoroughly investigated for a variety of configurations.[1-4] There are many classes of IHCPs according to the specification of the corresponding direct problem. For example, a three-dimensional steady-state IHCP[5] and two-dimensional transient IHCP[2,4] have been solved in the previous studies. Recently, IHCPs in micro- or nano-scale is also studied.[6] The scale of the problem is now considered another measure for classification of IHCP. On the other hand, IHCPs can also be classified by kinds of the unknowns to be determined. Such unknowns can be the material property,[7] boundary conditions[1] and geometry.[8,9]

Many different types of IHCPs can be encountered while analyzing the real-world thermal systems.[2-3] Implementational issues, however, are not much studied. This work will suggest a unified process for implementing the computer software that can solve IHCPs. This work employs the methods of the object-oriented programming to realize the computer software in a rapid and reliable fashion.[10] This work provides a framework for developing IHCP computer codes. This framework makes it possible to build up a new IHCP code simply by assembling existing codes since the concept of object-oriented software development allows easier reusing, safe wrapping and insured interoperability.

More specifically, the unified modeling language (UML), which is the standard object modeling language, is utilized in this development.[11,12] The present paper does not explain the mathematical details of the IHCP.[2,4] The readers who are not familiar with IHCP are recommended to read.[1] This work presents several UML diagrams and explains them. In the process of realizing those diagrams, the software design is completed. UML suggests that the scenario, use case, class and sequence diagrams be included as essential elements in the design. This work shows and explains those four elements sequentially. By doing so, what to realize and how to implement become clear.

2. Delineation of the inverse heat conduction problem

2.1 Scenario

We need to delineate what it is to solve an IHCP. There are many different solution methods for IHCPs including the sequential method,[1] the whole domain method[1-4] and the space marching method.[1,2] In the sequential and space marching methods, the inverse estimation and the direct problem are so strongly coupled that it is very hard to develop a flexible and reusable code.[1] Flexibility and reusability are crucial issues in software developments since those properties allow shortening the development time.[10] Although the sequential method and other methods have many excellent features, they are not considered in this work for the aforementioned reason.

The scenario here means the abstract description of what happens inside the IHCP software while it runs. There are decisions that should be made according to the developmental phases. Some of them are to be decided during the design phase and others during the implementational phase or run time. Let us describe those details.

Generally, in an inversion process, we reverse the direction of information flow.[2] By doing so, the causes are found from the observations of the results. In a typical IHCP, the boundary condition

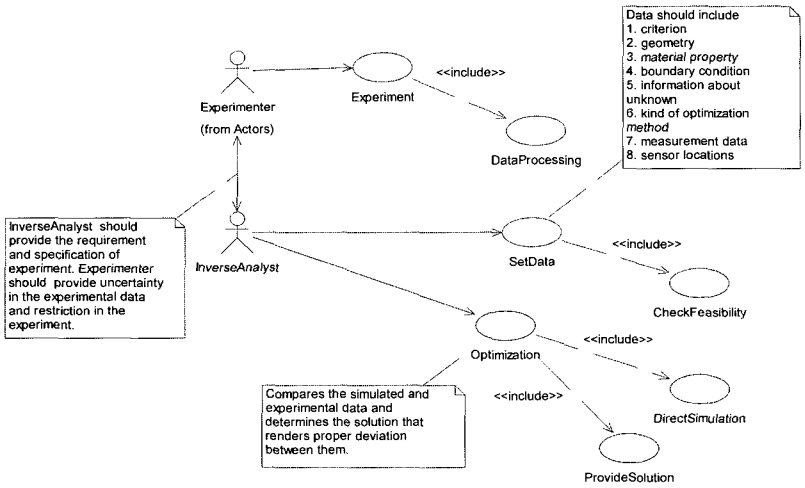

Fig. 1 Use case diagram

of Dirichlet or Neumann type is determined based on the temperature readings elsewhere. The main issue in this sort of inversion is the instability of the solution, which is attributed to the mathematical ill-posedness inherent in the problem.[1-4] To overcome this difficulty, treating the problem as an optimization problem is widely accepted and proven reliable.[2, 4] This study also employs the optimization-based approach.

The optimization problem for IHCP finds the unknowns of interest that render a statistical consistency between the calculation and measurement data. Thus, numerical or analytical calculation should be conducted to compare the results with the experimental data. Usually the experimental data in IHCP are temperature readings. Although thermal strain or heat flux measurements can also be such data under special situations, this study considers temperature measurements only.

2.2 Use case

Use cases mean a collection of scenario which can happen during running the software.[10, 11] The conventional scenarios of numerical tools are usually straightforward. There is no fatal risk that can be incurred by a failure of this kind of software run. Thus in most cases a simple description can explain the entire scenario successfully. The use cases for IHCP can be as follows.

Basic flow:

a. The experimenter obtains a set of experimental data.

b. The analyst gathers all the information required for the analysis including experimental data, statistical criterion, geometry, boundary condition, material property, sensor locations and information about unknowns of interest.

c. The analyst starts the inverse analysis by initiating the optimization routine.

d. The optimization routine repeatedly calls the direct simulation routine to calculate deviation between experimental and numerical data with the provision of test unknowns.

e. The optimization routine eventually finds the unknowns that satisfy the criteria imposed by the analyst and returns the unknowns as a final solution set.

Alternative flow1

c. The input data are inconsistent with each other so that the analysis cannot start.

Alternative flow2

e. The optimization routine cannot find the unknowns that meet the imposed criteria.

The use case diagram can explain the above scenario in a visual fashion, which is shown in Fig. 1. There is no strict rule in drawing this diagram. As long as the diagram can reveal what will happen in the system during run time, any form is good enough. The use case is an overview of the entire system from the outside of the system. It clearly explains the requirements of the system. Although it does not necessarily have to be connected to the classes of the system, it gives a conceptual foundation for building the classes.

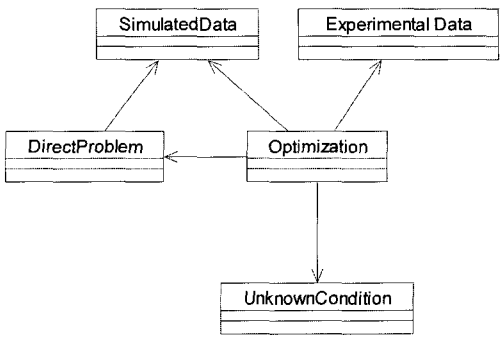

Fig. 2 Basic class diagram

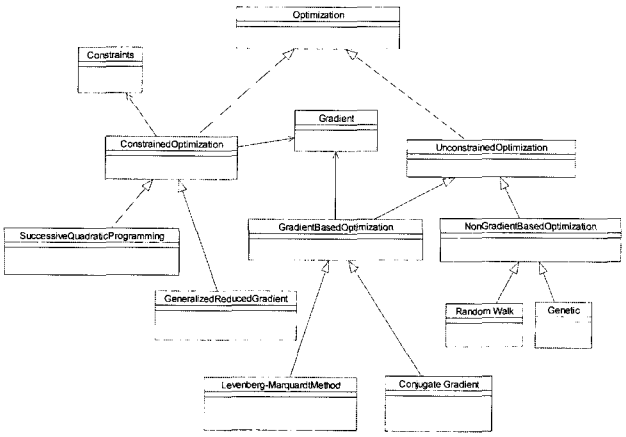

Fig.3 Optimization class diagram

As shown in the figure, there are two actors in the diagram, who are the Experimenter and the Inverse Analyst. The diagram shows what those actors should fulfill. Here, we assume the experimental system and the software are under their control. Especially, the Inverse Analyst's role is crucial. He or she should initiate the analysis and check up the integrity of the entire analysis. What should be done inside the software is now quite clear. Next step is to build up the classes to implement the system.

3. Diagrams

3.1 Basic Class Diagram

The classes are central to the objected-oriented modeling. The classes explain the objects of the system and the static relationship among them. There are three different views in classes. Each of them focuses on concept, specification and implementation respectively. This work basically considers the conceptual class diagram since the implementational details are not the purpose of this paper .

Three important aspects in solving IHCPs are how to conduct the optimization, how to simulate the phenomena numerically, and how to handle the experimental, numerical and user-provided data. The core of the classes is the optimization class where all the data should be processed. Thus, the class diagram in Fig. 2 shows the essence of the IHCP software. The optimization software seeks the desired unknown condition that meets the rule of discrepancy between the simulated and experimental data.

The implementation level class diagram should be extended from the basic diagram shown in Fig. 2. Basically, the optimization class seeks the unknown condition of interest that meets the discrepancy rule of IHCP. Consider an IHCP system with N measurement data. The discrepancy rule is expressed as[1]

$$J = \sum_{i=1}^{N} \frac{(T_i - Y_i)^2}{\sigma_i^2} \le N \tag{1}$$

where T_i, Y_i and σ_i are the simulated, measured temperatures and the standard deviation at the i-th sensor location respectively. Because seeking a solution that renders $J \approx 0$ returns an unstable solution,[2] the optimization problem usually becomes to find the unknown such that $J \approx N$. However, this kind of equality is quite hard to satisfy. Thus, the expression is replaced by an inequality condition like $N - \sqrt{2N} < J < N + \sqrt{2N}$.[1, 2] This discrepancy rule becomes the feasibility condition of the optimization problem. In most cases, we have to employ some sort of regularization to obtain sufficiently smooth and dependable solutions.[1-4] Depending on the way the regularization is handled, the optimization can be either constrained or unconstrained. Basically, when the inversion involves regularization, the optimization problem should be a constrained one.

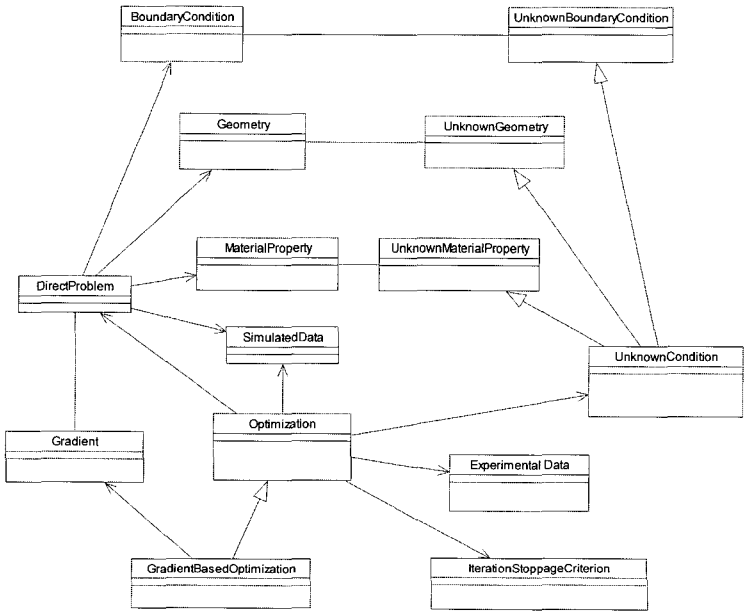

Fig. 4 Whole class diagram

However, introduction of the regularization parameter relieves the burden of the constrained optimization. Consider the regularization term, R, and the Lagrange multiplier, λ. In this case, the optimization problem should minimize R subject to the feasibility condition. The Lagrangian of the optimization in this case is written as

$$L = R + \lambda J \tag{2}$$

In the optimum point, we have $\nabla L = \nabla R + \lambda \nabla J = 0$. Introducing the regularization parameter, α, which is equal to $1/\lambda$, we have $\nabla J + \alpha \nabla R = 0$. We can convert the expression into a positive definite one by multiplying ∇J^T, which turns out

$$\nabla J^T \cdot (\nabla J + \alpha \nabla R) = 0 \tag{3}$$

For a given α, the above expression becomes a nonlinear system of equations. It should be noted that an engineer should determine α that satisfies the feasibility condition. The level of smoothing in the resulted solution varies greatly depending on the selection of the regularization parameter. Therefore, in a sense, the constrained optimization is a process that automatically determines the regularizat ion parameter. However, the computational cost also dramatically increases in the constrained optimization.[13] The IHCP software can run in either way according to the requirement.

3.2 Optimization Class

In order to provide flexibility in the analysis, the computer codes should be adaptable to various optimization methods. Thus, the optimization class should be an interface class to realize such adaptability and utilize many existing optimization codes. We can wrap any existing optimization code with an interface to make it adaptable to the IHCP code. This kind of wrapping is done by employing the adaptor pattern, which is shown in Fig. 3.[14]

A class diagram can be presented in various styles. The class diagram shown in Fig. 3 is not the most general or unique one. It is simply a plausible one. This class diagram shows that a variety of optimization codes can be utilized to implement the optimization class in the IHCP code. The generalization arrow in the diagram is realized by inheritance toward the direction reverse to the arrow. For example, in C++ the Levenberg-Marquardt Method can be declared as[10]

```
Class LevenbergMarquardtMethod :
        public GradientBasedOptimization{   };
```

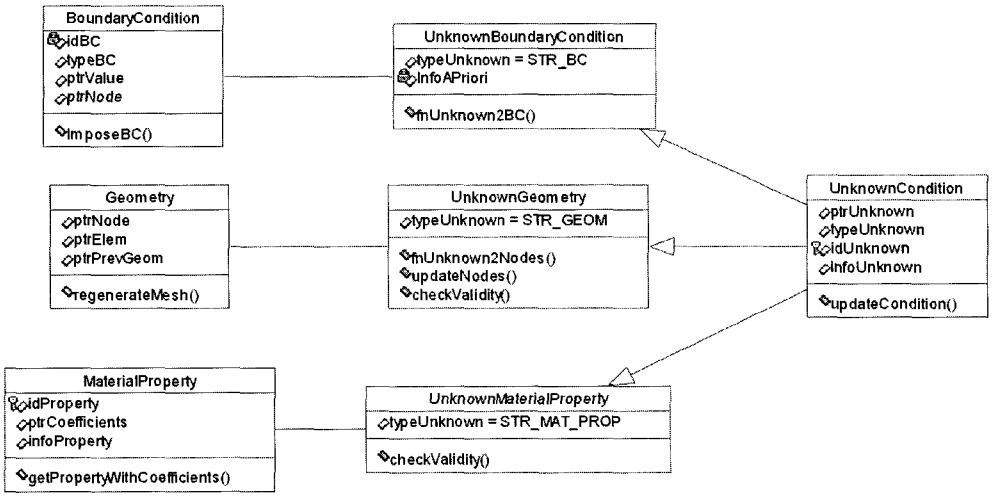

Fig. 5 Unknown condition class

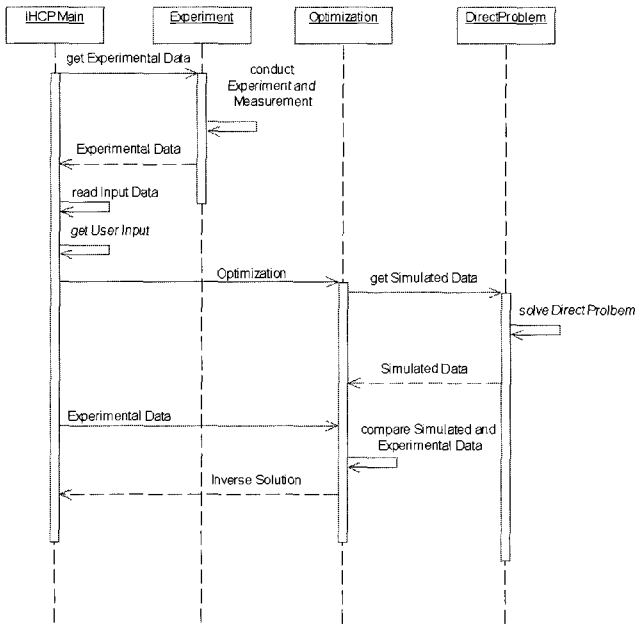

Fig. 6 Sequence diagram

3.3 Whole Class Diagram

The whole class diagram, which neglects the detail around the optimization class, is shown in Fig. 4. This diagram reveals several important natures of the IHCP code. The ultimate purpose of any IHCP code is to determine the unknown of interest. Most of IHCP, as in this diagram, regards geometry, boundary condition and material property as unknowns. There are several other unknowns such as heat source, initial condition and time constant of thermal wave in the hyperbolic heat equation. These are not considered here. The unknown condition is necessary for the renewal of the simulated data. The diagram conceptualizes the polymorphism of the unknown condition using a typical pattern. The realization of the unknown condition greatly depends on the type of the unknown. Thus, we need to unify the access method to the UnknownCondition class. Consider that the unknown is the boundary condition. The difficulty arises here is t hat the unknown condition should be treated as known boundary condition when solving the direct problem.

The conventional way of handling this kind of situation is to allot related variables to a global array. This negatively impacts the flexibility and reusability of the code. The UnknownCondition class should hold the pointer to the data in the BoundaryCondition class instead of holding the boundary condition data values. The UnknownCondition class should update the corresponding boundary condition when it is updated. The BoundaryCondition class does not have to hold any information whether it has any unknown condition or not. In this fashion, the UnknownCondition class and classes directly related to the DirectProblem class are isolated. Three classes derived from an abstract class, UnknownCondtion, are employed to resolve the different nature of three different kinds of unknown conditions. Complete separation of the Geometry, BoundaryConditon and MaterialProperty classes from the UnknownCondtion derived-classes allows great flexibility in designing and implementing the code for the direct problem.

A sample design of the UnknownCondition class is shown in Fig. 5. The class should have some members characterizing the unknown. The one important thing is the class should involve the metadata that explains the data itself in the class.

3.4 Sequences and Collaboration Diagrams

The sequence diagram presents the procedures to be done by the software. Fig. 6 shows the sequences diagram that performs the described inverse analysis. Understanding the diagram is quite straightforward. The diagram clarifies the tasks to be done by the software. The diagram does not have to include every detail of the procedures that the software should implement. As shown in the diagram, it reveals the essence of the process that should be done in an optimization-based IHCP code. The solid arrows mean the actions and dashed arrows mean the returns.

The main routine, IHCPMain, initiates the procedures. Then, the optimization routine repeatedly calls the direct problem and compares the simulated data with the experimental data. By doing so, the optimization routine finds the inverse solution. The idea that was described in the use case diagram is further concretized in the sequence diagram. The role of this diagram is to procedurally declare what is supposed to happen inside the software.

The collaboration diagram is similar to the sequence diagram. It further clarifies the tasks to be executed. Each operation is numbered according to the order of execution. As shown in Fig. 7, this diagram depicts how each class is collaborating with other classes.

3.5 Implementational Issues

Implementation here means realizing the executable computer software based on the abstraction presented in the diagrams. Recently, several development tools that are capable of automatically converting the diagrams into the computer codes are available.[15] Even without such automation tools the diagram greatly helps write codes. An automation tool or a developer creates a skeleton based on the diagrams. Then, the developer should fill out the routines in the skeleton. The developer might be able to utilize many existing codes in order to complete the routines. For example, the developer can reuse any existing code that can solve the direct heat conduction problem. Furthermore, commercial analysis software can be interlinked to the IHCP software via proper interfaces.

This paper does not present any actual computer codes. Such realization is result of the process but not the main issue in this paper. The actual computer code, which is implemented based on the proposed UML design, would greatly vary depending on the coding convention, developing tools, situation and other restrictions. Figure 8 shows a screen view of the implemented IHCP software.

4. Application to One-Dimensional IHCP

4.1 Problem Statement

To demonstrate the usefulness of the proposed framework, a simple one-dimensional IHCP is selected. The domain of interest is a finite one-dimensional slab with length 1. as illustrated in Fig. 9. The temperature of the left side $f(t)$ is unknown and the right side is insulated. Material properties are considered constant over the entire domain. The measured temperature $Y(t)$ is acquired on the insulated surface, $x=1$. The governing equation of this transient conduction problem with unit thermal diffusivity is given as follows.

$$T_t = T_{xx},\ T(0,t) = f(t),\ T_x(1,t) = 0,\ T(x,0) = 0 \tag{4}$$

The inverse problem should determine the $f(t)$ from $Y(t)$.

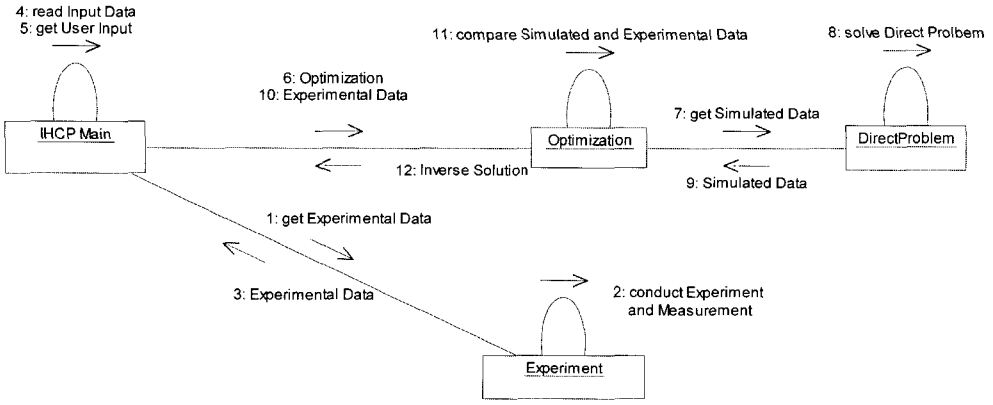

Fig. 7 Collaboration diagram

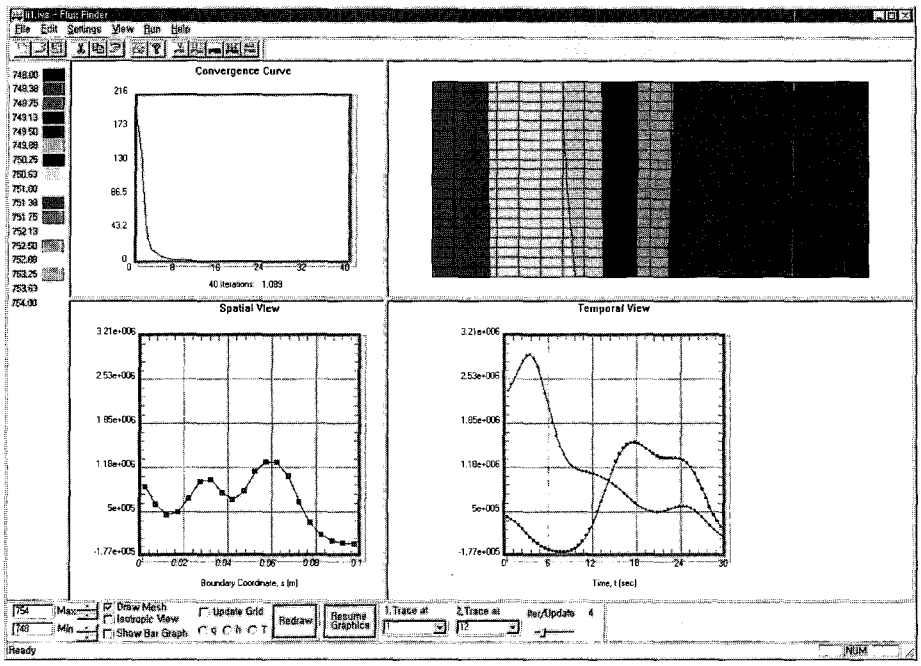

Fig. 8 Integrated computer software for IHCP

4.2 Solution Method

Consider Eqs. (1) and (2). Here, R is set as $\sum_{i=1}^{N} f'(t_i)^2$. The constrained optimization problem defined by Eq. (2) has been solved. That is, the objective is to minimize R subject to Eq. (1). The successive quadratic programming (SQP) in Fig. 3 is selected to solve this optimization problem since the problem is a nonlinear constrained problem. An existing SQP named CFSQP has been reused with a proper interfacing.[16] Equation (4) has been solved using the method of line, which is implemented in an IMSL routine, MOLCH.[17]

4.3 Results

In this investigation, the standard deviation is set as 0.01 and the time step is 0.05. Fig. 10 compares the result by the conventional conjugate gradient method, unconstrained Newton method and the proposed method. In all methods, the gradients are obtained by solving the adjoint problem of Eq. (4).[2,4] All necessary Hessians are numerically evaluated. The result in the figure shows that the constrained optimization using the SQP gives more resolving power.

Within this framework, many optimization methods, including the constrained and unconstrained optimizations, can be tested. Sometimes the regularization using the constrained optimization will be waste of computational resources especially when the problem is rather easy (bigger time step) and the guess of the regularization parameter is not difficult. We can easily develop an IHCP code using the existing codes on demand. As described, existing codes and libraries are reused to realize the IHCP code for this test. A flexible framework proposed in this work allows easier adoption of many existing methods and developed computer codes.

5. Conclusion

This paper presents a method to develop a computer code for the inverse heat conduction problem (IHCP) in an object-oriented fashion. The development process includes the scenario description, use case, class and sequence diagrams. Through completing the diagrams the concept and idea inherent in the original problem become concretized getting close to an implementational level.

There are meaningful implications in introducing object-oriented method to numerical tool developers. First, while realizing the UML diagrams, the developer becomes aware of every important feature to be implemented as well as underlying concepts. Second, revising and reusing the software becomes convenient and efficient.

This study is not simply about developing an IHCP code using the object-oriented programming technique. This study intends to realize the following two objectives. First, this work tried to suggest a

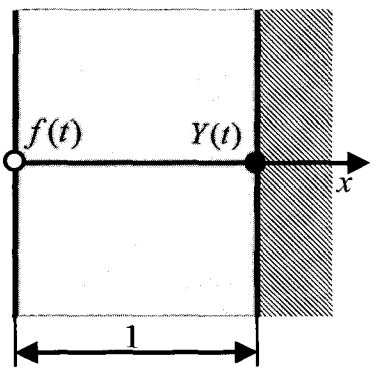

Fig. 9 Schematic of the test problem

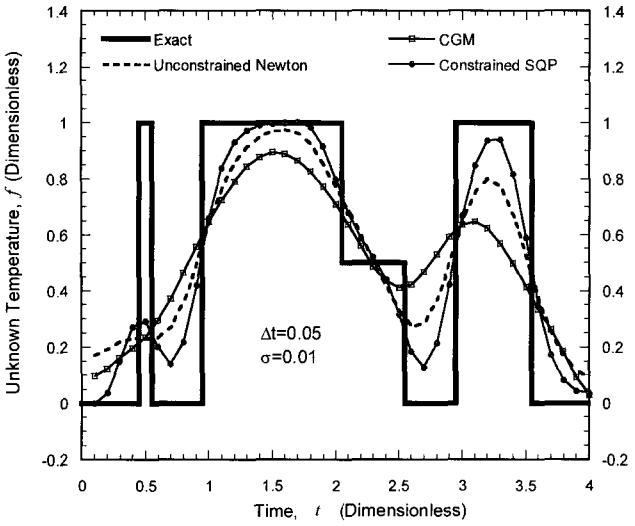

Fig. 10 Computational results for the test problem

versatile developmental environment that allows containing a variety of IHCP solution codes in a single framework. Second, we thought that an IHCP computer program should be capable of incorporating many existing codes, including the commercial CAE packages, as direct solution tools. To build such a computer program, a flexible as well as reusable foundation is essential. In this work, we proposed to use the UML to present and develop this foundation.

The current study has shown how to develop an involving numerical code that solves IHCP. The considered IHCP is solved in the optimization-based method for the flexibility and generality of the code. The completed diagrams clearly and systematically explain what should be done by the IHCP code and how to realize it.

REFERENCES

1. Beck, J. V., Blackwell, B. and St. Clair Jr., C. R., "Inverse Heat Conduction," Wiley, 1985.
2. Alifanov, M., "Inverse Heat Transfer Problem," Springer-Verlag, 1994.
3. Kurpisz, K. and Nowak, A. J., "Inverse Thermal Problems," Computational Mechanics Publications, 1995.
4. Ozisik, M. N. and Orlande, H. R. B., "Inverse Heat Transfer: Fundamentals and Applications," Taylor & Francis, 2000.
5. Kim, S. K. and Lee, W. I., "Inverse estimation of steady-state surface temperature on a three-dimensional body," International Journal of Numerical Methods for Heat & Fluid Flow, Vol. 12, No. 8, pp. 1032-1050, 2002.
6. Kim, S. K. and Daniel, I. M., "Solution to inverse heat conduction problem in nanoscale using sequential method," Numerical Heat Transfer, Part B, Vol. 44, pp. 439-456, 2003.
7. Flach, G. P. and Özişik, M. N.,"Inverse heat conduction problem of simultaneously estimating spatially varying thermal conductivity and heat capacity per unit volume," Numerical Heat Transfer, Part A, Vol. 16, pp. 249-266, 1989.
8. Hsieh, C. K. and Kassab, A. J., "A general mothod for the solution of inverse heat conduction problems with partially unkown geometries," International Journal of. Heat Mass Transfer, Vol. 29, No.1, pp. 47-58, 1986.
9. Kim, S. K. and Lee, W. I., "Implementation of inverse method for estimating undetermined boundary in a two-dimensional slab based on temperature measurement," Numerical Heat Transfer, Part A, Vol. 46, pp. 515-523, 2004.
10. Coad, P. and Nicola, J., "Object-Oriented Programming," Pearson Education, 1993.
11. Folwer, M., "UML Distilled: A Brief Guide to the Standard Object Modeling Language," 3rd Ed.,. Addison-Wesley, 2003.
12. Booch, G, Rumbaugh, J. and Jacobson, I., "The Unified Modeling Language User Guide," 1st Ed., Addison-Wesley, 1998.
13. Arora, J. S., "Introduction to Optimum Design," McGraw-Hill, 1989.
14. Gamma, E., Helm, R., Johnson, R. and Vlissides, J., "Design Patterns," 1st Ed., Addison-Wesley, 1995.
15. Quatrani, T., "Visual Modeling with Rational Rose 2002 and UML," Addison-Wesley, 2002.
16. Lawrence, C., Zhou, Z. L. and Tits, A. L., "User's Guide for CFSQP Version 2.5," University of Maryland, 1997.
17. Visual Numerics, "FORTRAN subroutines for mathematical applications," Houston, Texas, Visual Numerics, 1994.