Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

Flipping the Analytical Coin: Closing the Information Flow Loop In High Speed (Real Time) Analysis

K.E. Shahroudi

Software Engineering (SEN)

SEN-R9703 February 28, 1997

Report SEN-R9703
ISSN 1386-369X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Flipping the Analytical Coin: Closing the Information Flow Loop In High Speed (Real Time) Analysis

Kamran Eftekhari Shahroudi
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Email: kash@cwi.nl

ABSTRACT

Analysis modules tend to be set up as one way flow of information, i.e a clear distinction between cause and effect or input and output. However, as the speed of analysis approaches real time (or faster than movie rate), it becomes increasingly difficult for an external user to distinguish between cause and effect because they are simultaneously available for visualization and can be viewed in any desired order. This paper discusses some potential benefits of setting up a Flipping Analytical Coin Tool (FACT) which closes the information flow loop, thereby taking advantage of the apparent equivalence between forward and reverse analysis. A simple and stable non-linear algorithm is derived which uses the principle of dimensional similarity in each independent dimension of the problem to efficiently calculate the reverse path of high speed multidimensional functions or analysis modules. An example application in the field of conceptual thermodynamic design of a turbofan engine demonstrates the usefulness of this approach to interactive design optimization of complex engineering systems. The method given here is applicable to both well and ill-posed problems.

1991 Mathematics Subject Classification: 65K10, 90C29, 90C31, 93B51.
1991 Computing Reviews Classification System: B.5.2, C.4., D.4.7, F.1.2., J.6.
Keywords and Phrases: Inverse Design & Analysis, Modelling & Simulation, Computational Steering, Parameter Estimation, Dimensional Similarity.
Note: To appear in the Proceedings of the Second International Conference on Inverse Methods, Le Croisic, France, June 96. Work carried out under project SEN 1.3 Interactive Visualization Environments.

1. INTRODUCTION

In recent years, there has been increased pressure for developing fast (or real time) analysis modules from several scientific and engineering fields, e.g. High Speed Visualization, Animation, Computational Steering [ref. (4)] and Interactive (or Human-in-the-Loop) Optimization [ref. (6)]. Fast computational models or analysis modules can be very advantageous because they enable:

- interactive (or human in the loop) optimization;
- real time navigation in multi-dimensional parameter or design spaces [ref. (5)];
- the use of computationally intensive global optimization techniques;

- guiding more accurate and sluggish calculation modules to reduce the number of iterations or to ensure convergence.

Despite the above advantages, by far the majority of models and solvers aim for numerical accuracy with little or no consideration for the trade-off between accuracy and computational speed. While the improvement in technology and algorithms result in better computational performance, the gain is not likely to be sufficient to obtain real time for majority of applications. In some particular applications, it might be fruitful to pursue analytical shortcuts at some small expense to accuracy. For example it was possible to obtain a closed form solution (i.e. non-iterative, high speed and 99.9% accurate) for calculation of the steady state off-design performance of gas turbine engines which is traditionally an iterative procedure and several orders of magnitude slower to compute [ref. (6)]. However finding analytical short-cuts is a highly intuitive process and hence becomes more unlikely as the problem becomes more complex. So how can one exploit the above mentioned advantages of real time analytical modules, where in majority of fields, these are traditionally set-up as long iterative procedures? It is the goal of this paper to show that the answer probably lies in the direction of information flow in analysis. Suppose that for the special class of high speed analytical modules, it is possible to construct a high speed reverse path in the flow of information.

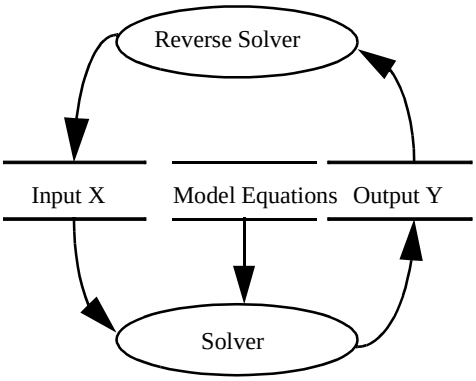

Figure 1. The Flipping Analytical Coin Tool (*FACT*).

The resulting set up can be visualized as a flipping coin (Figure 1) which has X and Y on two faces respectively. To the external user, X and Y co-exist without any particular concern how this co-existence was achieved as long as the coin is flipping fast enough and as long as the external user can interact with X or Y independently. Here interact is synonymous to modify and observe. Once the general purpose reverse solver is available, the challenge of setting up the *FACT* for a specific application involves the definition of X, Y and the Model Equations so that the forward path is high speed *and* X and Y are interesting or useful for the current analytical activity. In general, if the original analytical module used in a particular application is high speed, then X and Y can retain their original definitions, i.e no compromise. Otherwise a new definition of X and Y are necessary that use the Model Equations. The apparent equivalence that exists between input and output (or cause and effect) in this setup provides some freedom in setting up the analysis in a direction that is most convenient. Sluggish analysis modules which use long or complex iterative procedures require some sort of compromise, e.g. by working with intermediate results (Y) since they are available at a much faster rate.

In inverse analysis it is very easy to come across ill-posed problems and solution techniques [ref. (2)]. Current research in inverse methods seems to be weighted towards *regularization* [ref. (1)] and combining existing algorithms with application specific inversion criteria. Consider the 2D robot arm in Figure 2. Here the object of inverse analysis is to find the required

lengths and angles that result in a given change in the end position (P_x,P_y). Nakamura [ref. (3)] views the solution of this ill-posed problem as a constrained optimization problem and uses the kinetic energy as merit function (to minimize for a particular trajectory of end point) and transversality condition as constraint.

Finding the correct constraint and merit functions to solve specific inverse problems is in general not a simple task and it can always be argued whether a given specification gives the best results. The inverse method

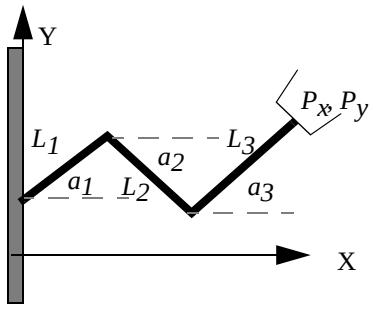

Figure 2. 2D Robot Arm.

of this paper adopts a similar view of inversion as a constrained optimization problem but attempts to make a general purpose specification by making use of the concept of dimensional similarity. A justification for this is that many physical quantities in proper dimensionless form, tend to retain their value to first or second order accuracy within some significant range, a property which is often exploited in engineering. For example the aerodynamic drag coefficient (a dimensionless parameter) of a car remains nearly constant in the normal speed range. Once this drag coefficient is measured at one speed, it can be used to estimate the drag at other speeds. A general specification of the inversion criteria is useful for cases where it is difficult to specify sensible constraint and merit functions. Alternatively it can serve as a benchmark for augmentation or comparison of application dependent specifications.

The following sections first state the 1-d inverse problem as a constrained optimization problem. The derivation of the constraint is based on the assumption that the dimensionless partial derivative (or log derivative) $(\partial lny) / (\partial lnx)$ varies linearly with x. Two solution procedures are then presented for the 1-d problem followed by extension to multiple dimensions with weighted variables. The resulting technique is very efficient and can be used to construct the reverse path of a high speed analytical module in order to set up the *FACT* as shown in Figure 1. The potential of the *FACT* is then shown by way of a example in the following areas:

- Navigation in a hyperspace;
- Parameter Estimation, fitting models to experimental data or fitting less rigorous models to their rigorous counterpart;
- Interactive Optimization (or human in loop) of gas turbine thermodynamic performance.

2. ALGORITHM TO SET UP A HIGH SPEED REVERSE PATH

The strict computational speed requirement of the *FACT*, forces the design of a suitable algorithm towards repeated application of a high speed one dimensional result. Other more obvious requirements are stability and immunity to scaling.

This section first treats the 1-d inverse problem followed by a fast solution technique. For clarity only a minimum of information is included to illustrate the main points. The actual implementation also includes some numerical tricks to make sure that the algorithm is stable. For example the definition of ratios where the denominator is zero, maximum and minimum constraint on the allowable values for dimensionless sensitivities etc.

2.1. Problem Statement

Given a 1-d transfer function $y = f(x)$ (where $x \in IR$), a current point $x_A, y_A = f(x_A)$ and a new output y_B, find x_B which minimizes the merit function or figure of merit (*fom*),

$$fom = \left| y_B - f\left(x_B \right) \right| , \qquad \textbf{(EQ 1)}$$

subject to a constraint based on dimensional similarity, $g(x)$.

2.2. Derivation of Similarity Constraint

The dimensionless sensitivity (or the log derivative) of *y* with respect to *x* is:

$$(\partial y / y) / (\partial x / x) = t(x) \quad . \qquad \textbf{(EQ 2)}$$

Integrating the above relation from A to B:

$$\int_A^B \partial y / y = \int_A^B t(x)\ (\partial x / x) \quad . \qquad \textbf{(EQ 3)}$$

Assuming a linear variation of the log derivative, i.e. assuming:

$$t(x) = t_A + \frac{t_B - t_A}{x_B - x_A}\left(x - x_A \right) \quad , \qquad \textbf{(EQ 4)}$$

and substituting for $t(x)$ and solving the integral yields:

$$ln\left(\frac{y_B}{y_A} \right) / t_A = ln\frac{x_B}{x_A} + \left(\frac{t_B}{t_A} - 1 \right)\left(2 + O_2 \right) ,$$

$$where\ O_2 = \frac{x_A}{x_B - x_A} ln\left(\frac{x_B}{x_A} \right) - 1 or\ O_2 = O_2\left(x_B, x_A \right) \qquad \textbf{(EQ 5)}$$

The above relation assumes that the dimensionless sensitivity (or log derivative) of *y* varies linearly from A to B. Simplifying the above relation by setting $t_B = t_A$ and introducing offset O_1 (to correct for this setting) yields the 1-d *forward* and *reverse* relations that we are seeking:

$$\frac{x_B}{x_A} = exp\left\{ ln\left(\frac{y_B}{y_A} \right) / t_A + O_1 \right\} \ or\ O_1 = ln\left(x_B / x_A \right) - ln\left(\frac{y_B}{y_A} \right) / t_A \quad , \qquad \textbf{(EQ 6)}$$

or expressed in functional form:

$$\boxed{\begin{aligned} x_B &= reverse\_1d(x_A, y_A, y_B, t_A, O_1) \\ O_1 &= forward\_1d(x_A, x_B, y_A, y_B, t_A) \end{aligned}} \quad \text{(EQ 7)}$$

2.3. General Solution

A general solution to the 1-d problem can be obtained by substituting the similarity constraint into the definition of fom. Substituting for y_B from (EQ 6) into (EQ 1) gives:

$$fom = \left| y_A exp\{ t_A \left(ln\left(x_B / x_A \right) + O_1 \right) \} - f(x_B) \right| \quad , \quad \text{(EQ 8)}$$

which transforms the original 1-d constrained optimization problem into an unconstrained optimization problem to determine O_1 *and* x_B, which can be solved by a variety of numerical optimization algorithms.

2.4. Efficient Solution

The general solution above transforms a 1-d problem into a 2-d problem. A more efficient solution is possible in the form of a predictor-corrector method which terminates when the magnitude of the error does not reduce any further. The error is defined by:

$$\varepsilon = \left(\frac{y_P - y_B}{y_B} \right) or\ \varepsilon = epsilon(y_P, y_B) \quad , \quad \text{(EQ 9)}$$

where subscript P denotes a predicted value. The iterative procedure of the 1-d reverse path is shown in Figure 3. The iteration starts with zero offset ($O_1 = 0$) as shown, which roughly locates the solution in the first iteration. Thereafter O_1 is allowed to have a value (i.e. relaxed) to refine the solution. In practice, for functions which are not too erratic, convergence is very fast, i.e. only a few iterations are needed (circa 5) to reach termination.

2.5. Extension to Multiple Dimensions with Weights

In this section a weights tensor is used to reduce the M-dimensional problem (i.e. input is M-dimensional vector) to a series of separate one-dimensional problems such that the one dimensional reverse path result of the previous section can be used.

2.5.1 Problem Statement

Given a high speed transfer function $Y = F(\underline{X})$, a weights vector

$O_1 = 0$

while ($|\varepsilon| > 0$ and reducing){

$x_B = reverse\_1d\left(x_A, y_A, y_B, t_A, O_1 \right)$

$y_P = f(x_B)$

$O_1 = forward\_1d\left(x_A, x_B, y_A, y_P, t_A \right)$

Figure 3. The 1-d Reverse Path Algorithm.

$\underline{W} = w_1, w_2, \ldots w_M$, a current state $\underline{X}_A, Y_A = f(\underline{X}_A), \underline{K}_A$, a new output state $\underline{Y}_B$ and a dimensionless sensitivity tensor $\underline{K}$, where $\underline{X} = x_1, x_2, \ldots, x_M$ is the *M*-dimensional input vector, Y is the output, $\underline{K}$ is the dimensionless sensitivity vector whose element $K_i = \partial lnY / \partial lnx_i = \partial Y / Y / \partial x_i / x_i \approx \delta Y / Y / \delta x_i / x_i$, find $\underline{X}_B$ which minimizes $fom = \left| Y_B - f(\underline{X}_B) \right|$, subject to a similarity constraint in each dimension.

2.5.2 Decomposition and Solution

The weights vector determines the relative influence of each input dimension in achieving the required output Y_B. The contribution of x_i to *Y* is denoted δY_i which is calculated as follows:

for each $i \in \{1, 2, \ldots, M\}$ {

$X_{B_i} = 1d\ reverse(X_{A_i}, Y_A, Y_A + \delta Y_i, K_{A_i}, f(x))$

}

Figure 4. The multi-dimensional reverse path.

$$\delta Y_i = \left(Y_B - Y_A \right) w_i / \sum_{i=1}^{M} w_i \ . \qquad \textbf{(EQ 10)}$$

The multidimensional minimization problem can now be decomposed into *M* one dimensional optimizations as follows:

For each dimension, minimize $\left| Y_A + \delta Y_i - f(\underline{X}_p) \right|$ *, such that similarity criteria is satisfied, where* $\underline{X}_p = \{x_{A_1}, x_{A_2}, \ldots, x_{B_i}, \ldots, x_{A_M}\}$.

In each minimization, the elements of the input vector $\underline{X}_p$ are the same as the initial input vector $\underline{X}_A$ except the i^{th} element x_{B_i} which is the result of the minimization. The calculation procedure of the multi-dimensional reverse path then makes use of the 1-d reverse path as shown in Figure 4.

2.5.3 Further Extensions

The multi-dimensional inverse problem is in general a *M* by *N* problem. It should be easy to see that the above approach can be extended in a similar way to decompose the *M* by *N* problem into *N*, *M*-dimensional minimizations which requires repetitive application of the above result. In this case the weight and sensitivity tensors become *M* by *N* tensors. Note that the weights and

sensitivity tensors are not inverted so that *M* and *N* are not forced to be related, e.g. *M* can be greater than *N* or vice versa.

Another extension is by not using the $t_A = t_B$ simplification of section 2.2. in which case extra parameters O_2 and t_B are also included in the iterative solution, which can be useful for some particular applications. For example, if the output *Y* represents some error, then making t_B equal zero or a given fraction of t_A pushes or launches the solution towards a minimum error without actually requesting $Y_B = 0$.

3. APPLICATIONS

This section first describes a simple model matching application in order to clarify the concept followed by a complex application in the field of conceptual design of gas turbine engines. Figure 5 illustrate the simple example and is a snapshot of the CSE (Computational

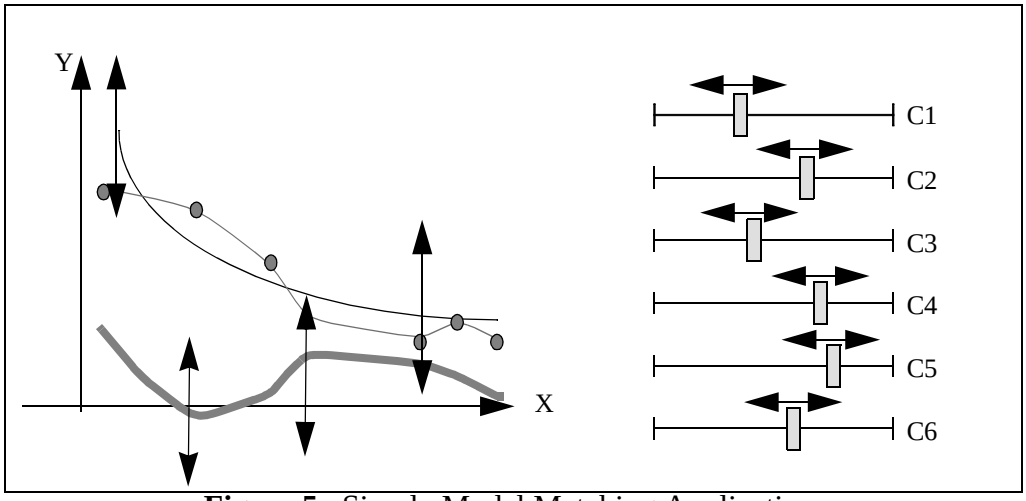

Figure 5. Simple Model Matching Application.

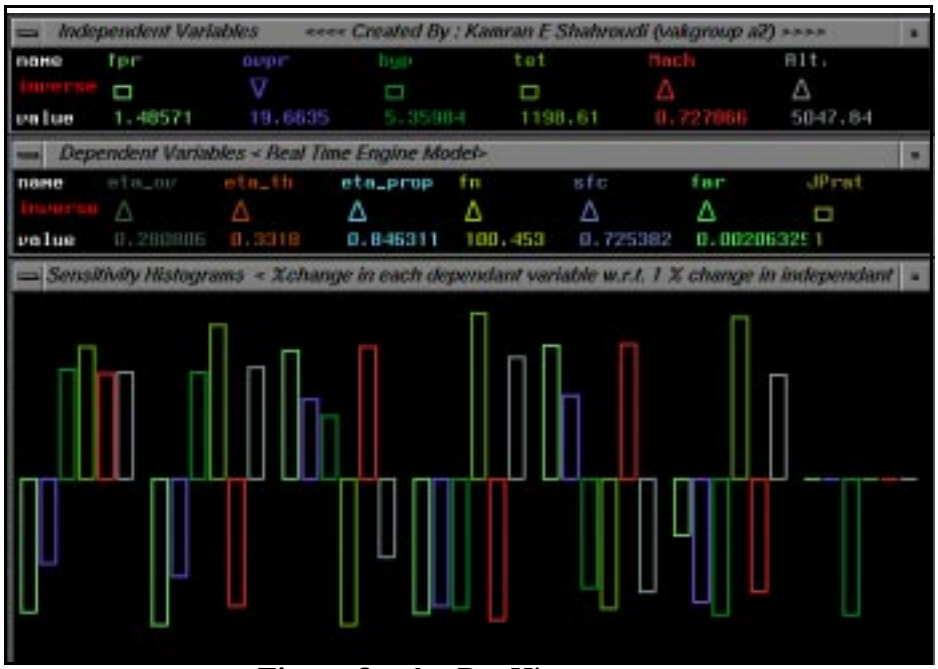

Figure 6. The *DynHist* Program

Steering Environment [ref. (4)]). The CSE allows the user to graphically interact with, control and monitor a simulation. The solid curve in the 2-d plot represents the 7-dimensional model $y = c_1 exp\{c_2[x - c_3]\} + c_4 exp\{c_5[x - c_6]\}$ whose independent variables are mapped onto the 6 sliders on the left (the *X* dimension is included in the curve). The dashed curve is formed by the connection of the experimental points or independent observations of *X* and *Y*. The gray curve represents the error, i.e. the difference between the dashed and solid curves.

3.1. Navigation in a Hyperspace by Modifying Features

The features of the model (e.g. the solid and dashed curves) occupy a multi-dimensional space. The user can explore this space by graphically moving the sliders and observing the resulting change in the shape of these features in real time. As dimensions of the model increase however (i.e. many sliders), it becomes increasingly difficult for the user to imagine or remember how to obtain a desired effect in the model features.

Here the *FACT* set up allows the user to modify the shape of the features by directly dragging the points that define them. The *FACT* algorithm then ensures that the corresponding new positions of the sliders are calculated and displayed in real time. Now since the sliders have moved, the whole shape of the features have changed, i.e. simple local interaction at one point, affects the feature globally, not just locally. In other words, the *FACT* enables the user to simply navigate in a hyperspace (e.g. design, solution or parameter space) by interacting with a graphical feature, as well as direct independent movements.

3.2. Parameter Estimation

The idea here is to match the model (solid curve) to experimental observations (dashed curve). In other words the user is searching for the position of the sliders which give the best match. The starting point (i.e. the initial position of sliders) is the result of a fully automatic matching technique (e.g. the non-linear regression routine RNLIN of IMSL library). In practical model matching, the user may not be satisfied with the usual least square fit criteria, used in such routines. For example some points may be more important than others or the model feature must exhibit certain characteristics near some points. Furthermore, it may not be possible to exactly specify the matching criteria in a mathematical expression.

The *FACT* set up allows the user to combine the following in order to reach a satis*fact*ory match:

- direct and continuous interaction with graphical features of the model (e.g. modifying the shape of the solid or gray curves);
- direct and continuous interaction with the independent parameters of the model (e.g. moving the sliders).

3.3. Interactive or Human-in-the-Loop Optimization

The previous matching example is also a toy example of interactive optimization because the user continuously tries to optimize the match. This section however gives a more involved example in the field of conceptual design optimization of gas turbine engines.

Interactive optimization is a very attractive alternative to fully automatic numerical optimization, when it is difficult to exactly specify a merit function in a mathematical expression. This

situation occurs in a significant number of optimization activities. For example, even the most experienced aircraft designers have difficulty in defining a merit function or a mathematical criteria that describes or leads to the optimum aircraft. This is particularly true in the conceptual design phase where much of the details of the final configuration becomes incrementally available, as the design optimization proceeds.

Figure 6 is a snapshot of the Dynamic Histograms program (*DynHist*) [ref. (5)]. *DynHist* is a module of the Computer Aided General Engine Design system(*CAGED*). Briefly *CAGED* generates high speed performance models for a visually defined engine network followed by interactive navigation and optimization with *DynHist* or *DynCarp* (Dynamic Carpet Plots). *DynHist* comprises of three windows which display design variables(input), dependent variables(output) and dimensionless sensitivities. In the upper two windows, Δ, ∇ *and* $\square$ indicate an increase, decrease and no change in value respectively. The design variables shown are fan pressure ratio (*fpr*), overall pressure ratio (*opr*), bypass ratio (*byp*), and flight Mach (*M*) and altitude (*h*). The dependent variables are overall efficiency (*etaov*), propulsive efficiency (*etaprop*), specific thrust (*fs*) etcetera.

The original version of *DynHist* allowed forward analysis only. Here the user could continuously interact with the design variables (top window) and observe the resulting change in the performance parameters and sensitivities (the lower two windows) in real time. *DynHist* and *DynCarp*, in their original form, showed that including the human designer in the high speed design loop was beneficial for design optimization of complex engineering systems.

In actual design exercises the number of design variables may be as high as 50 or more. It soon became interesting therefore to also interact in the middle window with the performance variables, i.e. inverse design. Here it was very difficult to find a meaningful engine specific constrained optimization specification which enables the calculation of the reverse path. The *FACT* proved to be very useful for this purpose as it already contains a general specification based on dimensional similarity. Using the reverse algorithms of this paper, the user can now directly increase say the overall efficiency of the engine (left most Δ in the middle window of Figure 6), and observe the resulting motion in the design variables (top window) and sensitivities (bottom window) in real time.

4. DISCUSSION and CONCLUSIONS

A Flipping Analytical Coin Tool (Figure 1) has been proposed which closes the information flow loop in high speed analysis, thereby taking advantage of the apparent equivalence between cause and effect or input and output in high speed (or real time systems).

A non-linear reverse algorithm has been derived (section 2.) which allows the calculation of the reverse path of a high speed forward analysis module. The algorithm treats each independent dimension of the problem as a constrained optimization problem.

The specification of the constraints is general (i.e. not application dependent) and is derived in section 2.2. based on the assumption of dimensional similarity of sensitivities in each independent dimension of the problem. This is a purely local assumption so that it involves the tenden-

cies only at a single point. Hence the technique is fundamentally different to non-linear regression analysis which requires a multitude of points in some range.

The formulation as a constrained optimization problem ensures that there is always a best possible solution regardless of the degree of redundancy of the problem. The algorithm is very efficient and fast as it combines non-linearity with decomposition of independent dimensions. The efficient solution scheme of section 2.4. assumes zero offset (i.e. $O_1 = 0$) to start the calculation which roughly locates the solution in the first iteration. This is then relaxed in the following iterations, i.e. O_1 is allowed to take a value to refine the solution. In practise a maximum of 5 iterations was found to be sufficient for functions which do not behave too erratically.

Two examples in the domain of interactive systems have been included which demonstrate the benefits of the *FACT* in multi-dimensional navigation, parameter estimation and human-in-the-loop optimization. The second example describes the application to interactive optimization of a turbofan engine in the conceptual design phase, i.e. a complex engineering system. In this phase it is very difficult to define a good engine specific constraint or strategy which enables the calculation of the reverse path. The *FACT* however can calculate the reverse path as it uses a general purpose constraint based on dimensional similarity. Using this it is possible for the user to directly interact with the performance of the engine (e.g. directly increasing the overall efficiency or the specific thrust) and observe the result in the design variables (e.g. fan pressure ratio or the bypass ratio). The *FACT* is therefore suitable as a general purpose interactive inverse design tool or as an augmentation or aid to existing specific inverse design tools.

5. Acknowledgments

This work was supported by the Netherlands Computer Science Research Foundation (SION) with financial support of the Netherlands Organization for Scientific Research (NWO).

6. References

1 Engl, H. W.,
"Regularization methods for the stable solution of inverse problems", Surv. Math. Ind., 3, No. 2 (1993).

2 Morozov, V. A.,
"Methods for Solving Incorrectly Posed Problems", Springer, New York, 1984.

3 Nakamura, Y.,
"Advanced Robotics - Redundancy and Optimization", Addison Wesley Publishing Company, ISBN 0-201-15198-7, 1991.

4 Wijk, J.J. van, van Liere, R.,
"An Environment for Computational Steering", Presented at Dagstuhl Seminar on Scientific Visualization, 23-27 May 1994, Germany, Proceedings to be published.

5 Shahroudi, K.E.,
"A Modern Approach to Conceptual/Preliminary Design of Gas Turbine Based Propulsion Systems", ASME paper 94-GT-458, Presented at the International Gas Turbine and Aeroengine Congress and Exposition, The Hague, Netherlands - June 13-16, 1994.

6 Shahroudi, K.E.,
"Development and Validation of a Computer Assisted Design Methodology for Gas-Turbine Based Aircraft Engines", Ph.D. Thesis, Delft University Press, ISBN 90-407-1070-8, December 1994.