Inverse Heat Conduction Methods in the *CHAR* Code for Aerothermal Flight Data Reconstruction

A. Brandon Oliver,[*] and Adam J. Amar,[*]

NASA Lyndon B. Johnson Space Center

2101 NASA Parkway, Houston, TX, 77058

Reconstruction of flight aerothermal environments often requires the solution of an inverse heat transfer problem, which is an ill-posed problem of specifying boundary conditions from discrete measurements in the interior of the domain. This paper will present the algorithms implemented in the *CHAR* code for use in reconstruction of EFT-1 flight data and future testing activities. Implementation nuances will be discussed, and alternative hybrid-methods that are permitted by the implementation will be described. Results will be presented for a number of one-dimensional and multi-dimensional problems.

I. Background

A note on this abstract: This is still very much a work in progress, however the code is functional and has been used to reconstruct flight data from the EFT-1 flight test. This paper is being presented provide information relevant to multiple papers in the ITAR sessions on the EFT-1 flight data (much of which used this tool) and allow those papers to focus on the data rather than the reconstruction techniques. As there is nothing specifically ITAR about the tool or algorithms, this paper is being submitted to a public session.

I have left some notes in red text throughout to indicate areas we intend to beef up in the final version. I also have not yet taken the time to completely include references and citations, though there are many places where they are needed.

I.A. Motivation

Validation of aerothermal environments developed for atmospheric entry of a spacecraft can be incrementally performed in carefully scaled ground tests; however, only flight can capture all of the relevant physics and their interactions, making flight data extremely valuable to the design and development of a practical spacecraft. A number of terms are of interest to the aerothermodynamicist during reentry as far as validation goes, but perhaps the most fundamental term of interest is the net heat flux from the environment to the surface of the heatshield. One of the primary purposes for predicting aerothermal environments is to size the thermal protection system (TPS) heatshield, and errors in the heat flux predictions can lead to heatshield failure or unnecessarily massive heatshields, which rob the system of valuable payload mass. When flight data is obtained to validate predicted environments, measurement errors can likewise contribute to the liklihood of heatshield failure or over-conserativsm if they falsely ‘validate’ bad predictions or ‘invalidate’ good predictions. Care must be taken to reduce measurement errors to every extent possible.

Heat flux can be a difficult term to measure. It is not an intrinsic property, like temperature or pressure, and it can only be inferred from measurements of other properties. Commonly available sensors that ‘directly’ measure heat flux, such as Schmidt-Boelter and Gardon gages, actually use temperature measurements on either side of a material with known thermal properties, and the heat flux is inferred from the temperature difference. Other sensors, such as slug calorimeters, measure the time rate of change of temperature of a carefully-designed mass with known thermal properties to infer the influx of thermal energy. Still another family of sensors operate by measuring as near as possible the surface temperature, and as with the calorimeters, the heat flux is inferred from the time rate of change of the surface temperature and heatshield thermal properties. This reliance on inference places considerable restrictions

[*]Applied Aeroscience and CFD Branch, Member AIAA.

on the operational conditions of heat flux sensors. There are few sensors that can operate at the conditions seen in atmospheric reentry.

A further complication in the effort to measure heat flux is the behavior of the heatshield on which the environments are being assessed. The windside heatshields for NASA's MSL and Orion capsules, SpaceX's Dragon, and Boeing's Starliner capsules are all charring ablators, so it is expected that the heatshield will char, meaning that the thermal properties will change, and ablate, meaning that the surface will recede and leave anything embedded in the heatshield to protrude into the oncoming flow. A protruding sensor will amplify heating in the vicinity of the measurement and therefore measure heating higher than intended (and then likely fail).

The Apollo Program instrumented a few of the flight test vehicle ablative heatshields with sacrificial calorimeters, but this path was not followed for the Orion capsule EFT-1 flight test, nor the MSL MEDLI program. These vehicles used embedded thermocouples (TCs) inside their respective heatshields with the TCs set deep enough that they would survive through the relevant part of reentry. As with the other types of heat flux sensor, the surface heating must be inferred from the actual sensor measurements by solving a problem known as the inverse heat conduction problem (IHCP).

This paper will describe the inverse heat conduction algorithm implemented in the *INHEAT* module of the *CHAR* ablation response code with intent to be used in reconstruction of aerothermal environments for the Orion EFT-1 flight test. Following a discussion of the IHCP and several common approaches to solving the problem, the specific formulation implemented will be given, and examples demonstrating the various modes of operation will be presented. In the final version of the paper, we will intend to present a reconstruction of one of MSL's MIPS plugs to compare to the work of Mahzari et. al.[1]

I.B. Problem Definition

In an inverse heat conduction problem, the objective is to estimate the boundary conditions given discrete (temporally and spatially) solutions interior to the domain of a heat conduction problem. The problem is an ill-posed problem, so frequently optimization methods are used to obtain the desired boundary condition values. Given sufficiently accurate measurements and sufficient knowledge of the material thermal properties, very accurate estimates (or reconstructions) of the boundary conditions can be obtained; however, the problems ill-conditioning means that the reconstruction will be very sensitive to measurement or modeling errors, and slight measurement noise could render an algorithm unstable. Any algorithm proposed to solve the IHCP must address the manner in which smoothing is applied to handle this tendency.

A class of problems similar to the IHCP are problems involving measured surface temperatures (ie: the solutions are on the boundary). In many cases these problems can be well posed as solved by straight forward techniques such as the Cook-Felderman[?] method; however, limitations on measurement resolution (spatial and temporal) lead to an incompletely-defined boundary condition, which can introduce many problems similar to those introduced by the IHCP. Often, these problems can be solved with IHCP algorithms and so often get lumped into the same class of problem.

Depending on the problem of interest, the unknown boundary conditions can take several forms. They can be temporally varying, spatially varying, or both. In most cases in the literature, IHCP algorithms tend to focus on problems for which the boundary conditions are temporally varying, sometimes referred to as function estimation problems. If the boundary conditions are fixed in time and only vary in space (or are just simply a constant) the problem is generally classified as a parameter estimation problem. The gulf between function and parameter estimation techniques is not very large, but IHCP algorithms are often built to explicitly address issues introduced by temporally varying boundary conditions.

I.C. Common Algorithm Classifications

A large number of approaches have been proposed to solve the IHCP, however a fairly common and well characterized approach is to discretize the boundary condition values in the time domain and compute values that minimize the least-squares difference of the measured and computed temperatures. A number of algorithms use optimization algorithms such as conjugate gradient[?] to estimate the values. Others use more novel approaches with Bayesian inference and neural networks[?] to estimate the parameters of interest. A large number of algorithms[?] solve for the minimum solution by differentiating the least squares residual and setting it equal to zero. This introduces sensitivity coefficients, which will be discussed at length in the next section, but provides a straightforward approach to estimating the unknown boundary condition values.

For algorithms that take the latter approach, a choice exists as to how to go about solving for the temporally-discretized values of the boundary condition. In *sequential* algorithms, they are estimated one at a time, in order from the earliest time. By contrast, *whole domain* algorithms simultaneously estimate all of the values in one step. Each method has strengths and weaknesses. Sequential algorithms can be computationally more efficient for non-linear problems as less time is spent computing temperature estimates and sensitivity coefficients at times where the initial guess is poor. On the flip side, whole domain algorithms perform much better when TCs take a substantial amount of time to respond to changes in the boundary condition. Neither is universally better than the other; the optimal choice comes down to the sensitivity coefficients.

In the present work, a hybrid sequential-whole domain algorithm is proposed which allows the user to tailor the problem to the sensitivity coefficients for the problem at hand. Depending on the values of a few parameters, the pure sequential function specification algorithm of Beck et. al.[2], or a whole-domain Gauss-Newton method[?] can be obtained. Furthermore, the concept of multiple smaller whole-domain solutions is introduced to allow sequential specification of the unknown boundary condition values, allowing reduced computational effort than the pure whole domain approach without having to accept the excessive smoothing introduced for slowly responding sensors in the sequential function specification algorithm.

II. Sensitivity Coefficients

As stated before, IHCP algorithms are generally built on a least-squares optimization of the computed and measured temperatures.

$$\text{Minimize wrt } \mathbf{q} : S = \sum \left(\mathbf{Y} - \mathbf{T}(\mathbf{q})\right)^2 \tag{1}$$

The derivative of Equation 1 is taken with respect to the boundary condition values and set equal to zero to find the minimum value. This process introduces the derivatives

$$\frac{\partial \mathbf{T}}{\partial \mathbf{q}^T} = \begin{bmatrix} \frac{\partial T_1}{\partial q_1} & \frac{\partial T_1}{\partial q_2} & \cdots & \frac{\partial T_1}{\partial q_{max}} \\ \frac{\partial T_2}{\partial q_1} & \frac{\partial T_2}{\partial q_2} & \cdots & \frac{\partial T_2}{\partial q_{max}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial T_{max}}{\partial q_1} & \frac{\partial T_{max}}{\partial q_2} & \cdots & \frac{\partial T_{max}}{\partial q_{max}} \end{bmatrix} \tag{2}$$

which are known as sensitivity coefficients. As will be shown later, the sensitivity coefficients drive IHCP algorithms, and depending on how they behave for certain problems, the ideal IHCP approach may change.

Demonstrate that they contain the physics of a problem

Demonstrate how different they can be for different physics

Illustrate how the optimal reconstruction approach could be very different with different classes of sensitivity coeffs

III. Mathematical Formulation

III.A. Derivation

The algorithm implemented in *INHEAT* is based on the trial function specification algorithm of Beck et. al.[2]. It is a least-squares optimization of the modeled temperature to the measured temperatures with a regularization term for stability:

$$S = (\mathbf{Y} - \mathbf{T})^T \mathbf{\Psi}^{-1} (\mathbf{Y} - \mathbf{T}) + \alpha \left[\mathbf{H}(\mathbf{q} - \tilde{\mathbf{q}})\right]^T \mathbf{W} \left[\mathbf{H}(\mathbf{q} - \tilde{\mathbf{q}})\right] \tag{3}$$

where $\mathbf{Y}$ is the vector of temperature measurements (ie: the reconstruction target), $\mathbf{T}$ is the temperature resulting from the numerical model of the system, $\mathbf{\Psi}^{-1}$ is a covariance matrix describing $\mathbf{Y}$, α is the regularization factor, $\mathbf{H}$ the regularization matrix, $\mathbf{W}$ is the regularization weighting matrix, $\mathbf{q}$ is the vector of boundary condition values. The term $\tilde{\mathbf{q}}$ is a vector of expected boundary condition values that can be used to fine-tune regularization if desired. It is assumed to have the form $\tilde{\mathbf{q}} = \mathbf{B}\mathbf{q} + \tilde{\mathbf{q}}^f$ with $\mathbf{B}$ and $\tilde{\mathbf{q}}^f$ specified by the user. Note that multiple regularization terms of this form can be applied if desired, but only one will be included to simplify presentation. The specific structure of all of these matrices will be described later in this section. Depending on how they are specifically defined, sequential function specification or whole domain solutions (or combinations in between those limits) can be obtained.

We seek to minimize the residual S, so we will take the derivative of Eqn. 3 with respect to each of the boundary condition values:

$$\frac{\partial S}{\partial \mathbf{q}} = -2\mathbf{X}^T \Psi^{-1} \left(\mathbf{Y} - \mathbf{T}\right) + 2\alpha \left[\mathbf{H}\left(\mathbf{I} - \mathbf{B}\right)\right]^T \mathbf{W}\mathbf{H}\left(\mathbf{q} - \tilde{\mathbf{q}}\right) \tag{4}$$

where

$$\mathbf{X} = \left(\frac{\partial \mathbf{T}^T}{\partial \mathbf{q}}\right)^T = \frac{\partial \mathbf{T}}{\partial \mathbf{q}^T} = \begin{bmatrix} \frac{\partial T_1}{\partial q_1} & \frac{\partial T_1}{\partial q_2} & \cdots & \frac{\partial T_1}{\partial q_{max}} \\ \frac{\partial T_2}{\partial q_1} & \frac{\partial T_2}{\partial q_2} & \cdots & \frac{\partial T_2}{\partial q_{max}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial T_{max}}{\partial q_1} & \frac{\partial T_{max}}{\partial q_2} & \cdots & \frac{\partial T_{max}}{\partial q_{max}} \end{bmatrix} \tag{5}$$

is the sensitivity coefficient matrix. Introducing a Taylor series expansion about the current solution yields

$$\mathbf{T} = \mathbf{T}^* + \mathbf{X}\Delta\mathbf{q} \tag{6}$$

where $\mathbf{T}^*$ is the temperature evaluated with the flux vector $\mathbf{q}^*$, and $\Delta\mathbf{q} = (\mathbf{q} - \mathbf{q}^*)$ is the update to the estimated solution vector. Setting Eqn. 4 to zero, substituting in Eqn. 6, and rearranging yields the system:

$$\left(\mathbf{X}^T \Psi^{-1}\mathbf{X} + \alpha \left[\mathbf{H}\left(\mathbf{I} - \mathbf{B}\right)\right]^T \mathbf{W}\left[\mathbf{H}\left(\mathbf{I} - \mathbf{B}\right)\right]\right) \Delta\mathbf{q} = \mathbf{X}^T \Psi^{-1} \left(\mathbf{Y} - \mathbf{T}^*\right) - \alpha \left[\mathbf{H}\left(\mathbf{I} - \mathbf{B}\right)\right]^T \mathbf{W}\mathbf{H}\left(\left(\mathbf{I} - \mathbf{B}\right)\mathbf{q}^* - \widetilde{\mathbf{q}^f}\right) \tag{7}$$

The term containing $\mathbf{q}^*$ does not appear in the derivation of Beck et. al.[2] as that derivation makes the assumption that $\mathbf{q}^* = \mathbf{0}$. This term is necessary for the regularization to appropriately smooth the solution $\mathbf{q}$ if $\mathbf{T}^*$ and $\mathbf{X}$ are evaluated at non-zero $\mathbf{q}^*$. If this term is omitted, the regularization smooths the update $\Delta\mathbf{q}$.

Introducing the Taylor series approximation (Eqn 6) linearizes the problem about $\mathbf{q}^*$, so iteration will likely be required to converge the solution $\mathbf{q}$ if the problem is non-linear.

III.B. Matrix Definition

The literature found to date almost exclusively makes the assumption that boundary condition values are estimated on the same time scales as measurements are available. This is notationally convenient, however, it can cause problems for practical applications; say, for instance, that measurements are unavailable at a particular time or if the data acquisition rate changes near an expected transient event. In an attempt to make this as general and flexible as possible, the algorithm has been defined to allow measurement and boundary condition estimation times to be completely flexible, and it is up to the user to specify a reasonable problem for which an answer exists. Furthermore, an arbitrary number of sensors could be providing data, and possibly multiple uncertain boundary conditions could be estimated simultaneously. This flexibility makes the notation become much more complex, so to assist in understanding it, the notation will be explained in detail.

Figure 1 presents an illustration of how time is indexed. The top timeline describes the notation for the overall problem, and the red line below shows the nomenclature for a specific local solution (in the event that the overall problem is being solved sequentially). The top row of each timeline denotes that measurements times for which target data is provided to the algorithm, which are globally indexed by the letter m and locally indexed by the letter i. Note that the spacing between these times are not uniform. The bottom row of each line denotes the time *intervals* for which BC values are to be estimated. There is no requirement that these intervals be regularly spaced, and the BC values defined in each interval is assumed to be constant through the interval. While a special symbol is not provided to indicate the max of the global problem, the max measurement time index in a particular local solution is r, and the max BC interval index in a particular local solution is F. Note that for different local solutions, r and F may change based on the length of the future time window (the length of the local solution) and user specification.

Beyond the times described in Figure 1, there may be input from multiple sensors at each measurement time (for instance, if there are several TCs in the system), and there may be multiple distinct BC values estimated on each interval (for instance, heating of different values to different boundaries of the system). Multiple TCs are indexed using the letter j (max J) in the matrix definitions, and multiple BCs on each interval are indexed with the letter n (max N). In the present derivation, it is possible to have only a subset of sensors providing data for a specific measurement time by using a special blanking matrix. However, at this time, it is not possible to define separate intervals for each of the distinct BCs; all distinct BCs are estimated on each BC interval. Given this, there will always be $(N \times F)$ BC values estimated in each local solution, and this dimension is given a special symbol P.

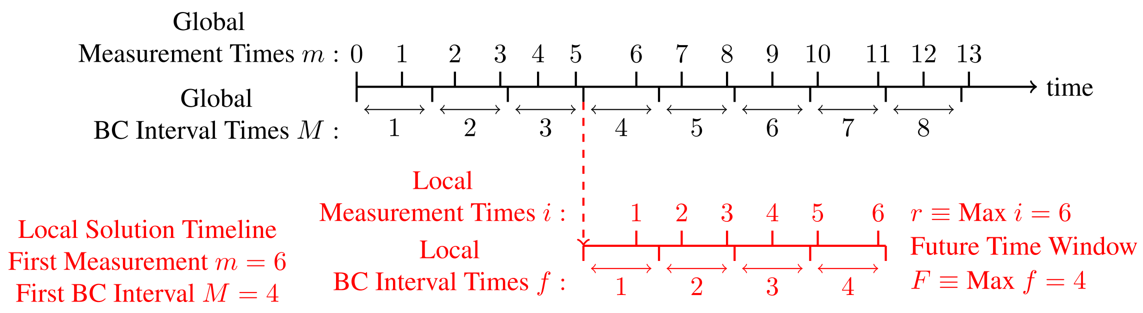


Figure 1. Definition of nomenclature describing the time-scales of the reconstruction.

Global Solution Indices:

j	TC number (max J)
m	Global measurement times
M	Global boundary condition time intervals
n	Distinct boundary conditions (max N)

Local Solution Indices:

i	Measurement times in current local solution (max r)
f	Boundary condition time intervals in current local solution (max F)
k	Boundary condition values in current local solution (max $P = FN$)

Table 1. Summary of indices used in matrix definitions

All of the matrices presented herein are built for a specific local solution with m referring to the first measurement time in a local solution and M the first BC interval in the local solution domain.

The vectors that represent temperature (ether the target data $\mathbf{Y}$ or modeled temperatures $\mathbf{T}$ have the following block form:

$$\mathbf{T} = \begin{array}{c c} \overset{1}{\longleftrightarrow} & \\ \begin{bmatrix} T(m) \\ T(m+1) \\ \vdots \\ T(m+r-1) \end{bmatrix} & \updownarrow Jr \end{array} \tag{8}$$

$$T(i) = \begin{array}{c c} \overset{1}{\longleftrightarrow} & \\ \begin{bmatrix} T_1(i) \\ T_2(i) \\ \vdots \\ T_J(i) \end{bmatrix} & \updownarrow J \end{array} \tag{9}$$

I need to define the blanking matrix used when a particular sensor is not available at a specific time i. These look messy too. Look into changing indexing notation to make things more compact.

The vectors that represent the estimated boundary condition values have a similar block form:

$$\mathbf{q} = \begin{array}{c c} \overset{1}{\longleftrightarrow} & \\ \begin{bmatrix} q(M) \\ q(M+1) \\ \vdots \\ q(M+f-1) \end{bmatrix} & \updownarrow P \end{array} \tag{10}$$

$$q(f) = \overset{\xleftrightarrow{1}}{\begin{bmatrix} q_1(f) \\ q_2(f) \\ \vdots \\ q_N(f) \end{bmatrix}} \updownarrow N \tag{11}$$

The sensitivity coefficient matrix has the form:

$$\mathbf{X} = \overset{\xleftrightarrow{\qquad\qquad\qquad\qquad Pr \qquad\qquad\qquad\qquad}}{\begin{bmatrix} \tilde{a}(m) & 0 & 0 & \dots & 0 \\ \tilde{a}(m+1) & \tilde{a}(m) & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \tilde{a}(m+r-1) & \tilde{a}(m+r-2) & \tilde{a}(m+r-3) & \dots & \tilde{a}(m) \end{bmatrix}} \updownarrow Jr \tag{12}$$

where

$$\tilde{a}(i) = \begin{bmatrix} a_{11}(i) & a_{12}(i) & \dots & a_{1F}(i) \\ a_{21}(i) & a_{22}(i) & \dots & a_{2F}(i) \\ \vdots & \vdots & \ddots & \vdots \\ a_{J1}(i) & a_{J2}(i) & \dots & a_{JF}(i) \end{bmatrix} \tag{13}$$

$$a_{jf}(i) = \begin{bmatrix} a(i,f,j,1) & a(i,f,j,2) & \dots & a(i,f,j,N) \end{bmatrix} \tag{14}$$

$$a(i,f,j,n) = \frac{\partial T(\mathbf{x}_j, t_i)}{\partial q_n(t_f)} \tag{15}$$

Note that in Eqn. 15, t_f refers to the time interval and not a specific time. This form of the sensitivity coefficient matrix is particularly convenient for the manner in which f is defined and the sensitivity coefficients are evaluated, as will be seen later. For other problems, it is sometimes easier to evaluate $\mathbf{X}$ with the boundary conditions perturbed on a timescale other than the f intervals. If this is the case, then a matrix must be defined to integrate the coefficients onto the f intervals. this matrix will be provided in the final draft.

The covariance matrix Ψ^{-1} describes the covariance of the various measurements provided to the algorithm. For the present work, it is assumed that the measurement errors are uncorrelated, additive, normally distributed with zero mean, so this matrix reduces to: This needs to be verified

$$\Psi = \text{diag}\begin{bmatrix} \Psi(i) & \Psi(i+1) & \dots & \Psi(r) \end{bmatrix}_{(Jr \times Jr)} \tag{16}$$

with

$$\Psi(i) = \text{diag}\begin{bmatrix} \sigma^2_{Y_1} & \sigma^2_{Y_2} & \dots & \sigma^2_{Y_J} \end{bmatrix}_{(J \times J)} \tag{17}$$

In this form, this matrix effectively scales temperature errors between the measurements and numerical model such that if the error is less than the measurement uncertainty, the least-squares residual contribution is reduced, but it is increased if the error is greater than the measurement uncertainty.

The regularization matrix $\mathbf{H}$ governs how the algorithm attempts to provide stability by smoothing the solution. It is a penalty function, where the penalized condition is defined by $\mathbf{H}$. One common regularization method is to penalize based on the absolute magnitude of the solution, $\alpha_0 q$, which is often referred to as 0^{th}-order regularization. Alternatively, 1^{st}-order regularization penalizes based on the difference of one solution value as compared to the next, $\alpha_1 \cdot (q_{i+1} - q_i)$. For 0^{th}-order regularization, the definition is clear; however, for 1^{st}-order regularization, it can be ambiguous or problem-dependent if the smoothing is applied to multiple BCs in space, or multiple BC values in time. The the present derivation, the overall regularization matrix $\mathbf{H}$ (with dimensions $P \times P$) is the product of two matrices: $\mathbf{H}_t$ which regularizes specific BCs in time, and $\mathbf{H}_x$ which regularizes multiple BC values in space on a given time interval:

$$\mathbf{H} = \mathbf{H}_t \mathbf{H}_x \tag{18}$$

The spatial regularization matrix has the block $(F \times F)$ form:

$$\mathbf{H}_x = \text{diag}\Big[\mathbf{h} \quad \mathbf{h} \quad \ldots \quad \mathbf{h}\Big]_{(P \times P)} \tag{19}$$

with $\mathbf{h}$ being an $(N \times N)$ matrix that maps the appropriate BCs to each other for smoothing. If spacial regularization is to be neglected, $\mathbf{h} = \mathbf{I}_N$, leaving $\mathbf{H}_x = \mathbf{I}_P$.

0^{th}-order temporal regularization is obtained by setting $\mathbf{H}_t = \mathbf{I}_P$. 1^{st}-order regularization is obtained by defining the temporal regularization matrix as the block $(F \times F)$ matrix of the form:

$$\mathbf{H}_t = \begin{bmatrix} -\mathbf{I}_N & \mathbf{I}_N & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_N & \mathbf{I}_N & \mathbf{0} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} & -\mathbf{I}_N & \mathbf{I}_N \\ \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}_{(P \times P)} \tag{20}$$

The regularization weighting matrix $\mathbf{W}$ has taken on a novel form to permit 'focusing' the regularization for sequential whole domain solutions. For pure whole domain solutions, shifting the emphasis of the regularization may not have a large benefit. However, as will be shown later, there is an advantage for certain problems to suppress the regularization at different ends of the local solution time domain (different f indices) with the following:

$$\mathbf{W} = \text{diag}\Big[W(1) \quad W(2) \quad \ldots \quad W(F)\Big]_{(P \times P)} \tag{21}$$

$$W(f) = \text{diag}\Big[\frac{1}{10^{fW_f}} \quad \frac{1}{10^{fW_f}} \quad \ldots \quad \frac{1}{10^{fW_f}}\Big]_{(N \times N)} \tag{22}$$

where W_f is a user input factor. For pure whole domain solutions, W_f can be set to 0 to obtain $\mathbf{W} = \mathbf{I}_P$ as desired. However, a positive value of W_f will result in less regularization at later f, whereas a negative W_f will yield less regularization on earlier f.

At the present time, very little has been done to characterize the behavior of the $\mathbf{B}$ matrix enabling the trial function capability. For all presented results,

$$\mathbf{B} = \mathbf{0}_{(P \times P)} \tag{23}$$

IV. Implementation

IV.A. *CHAR* Description

The IHCP capability described herein has been implemented as a module in the *CHAR* code. *CHAR* is a 1D/2D/3D material thermal response code which solves general heat transfer problems on decomposing charring ablators as well as non-decomposing, non-charring TPS materials, in serial or parallel. A more complete description can be found in Amar et. al.[?] . will be beefed up for the final draft.

IV.B. Implementation details

Sensitivity coefficients are calculated by evaluating temperatures with appropriately perturbed boundary conditions and using those to evaluate finite difference approximations. By default, central differences are computed (with a positive and a negative perturbation about the current guess), however one-sided differences may be used if desired. much more detail here in the final draft

IV.C. Algorithm Modes

The algorithm presented in the previous section has a number of parameters which can be specified by the user, and significantly alter the behavior of the algorithm. Several of the key parameters are:

- **Length of the future time window**. This determines the length of time covered in a specific local solution. It affects how many measurements will inform this reconstruction step, and what period of time over which sensitivity coefficients must be calculated.
- **Discretization of BC interval times**. Determines how many boundary condition values must be estimated, and how many measurements will inform each boundary condition interval. Note that having multiple measurements inside a boundary condition interval implicitly adds stabilization to the algorithm in the same manner as the future time algorithm.
- **Parameter** F. If the user specifies F such that there are more intervals f that are covered by the future time window, then it will ignore the intervals $f > F$. The $f = F$ interval will be assumed to apply all the way to the end of the future time window.
- **How many estimated BC intervals are 'retained' after a local solution**. This algorithm allows the user to 'retain' only a subset of the estimates calculated in a given local solution. In this way, solutions earlier in the future time window, which have sensitivity coefficients that have had time to develop, are retained as they are likely more accurate, and estimates at later times go towards improving the initial guess.
- **Regularization parameters,** α_0**,** α_1**, and** W_f. If $F > 1$, these options control solution stabilization.

IV.C.1. Example Case Definition

The effects of these parameters are perhaps best described by illustration. A benchmark case is used to demonstrate the performance of each mode. This case is a semi-infinite slab of material with the constant properties:

$$\rho = 280\,\mathrm{kg/m^3}$$
$$k = 0.4\,\mathrm{W/m\cdot K}$$
$$C_p = 1000\,\mathrm{J/kg\cdot K}$$

A single thermocouple is placed at a depth of $3.8\,\mathrm{mm}$. This combination yields a sensitivity coefficient that peaks at approximately $5\,\mathrm{s}$, as seen in Figure 2. A heating profile as defined in Figure 3 will be applied to generate artificial data for reconstruction. Each algorithm will attempt to reconstruct this temperature profile, as well as a profile with very agressive white noise added, also shown in Figure 3. This is a relatively simple linear problem, and therefore does not necessarily provide a measure of efficiency, but it does demonstrate the general characteristics of the converged result of a more realistic non-linear problem. This case will probably be refined for the final draft.

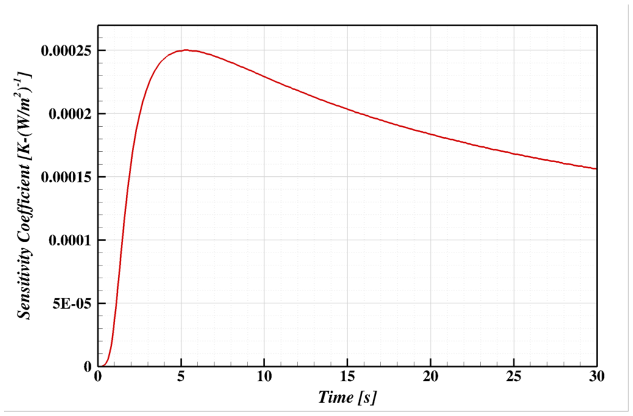


Figure 2. Sensitivity coefficient curve for example problem

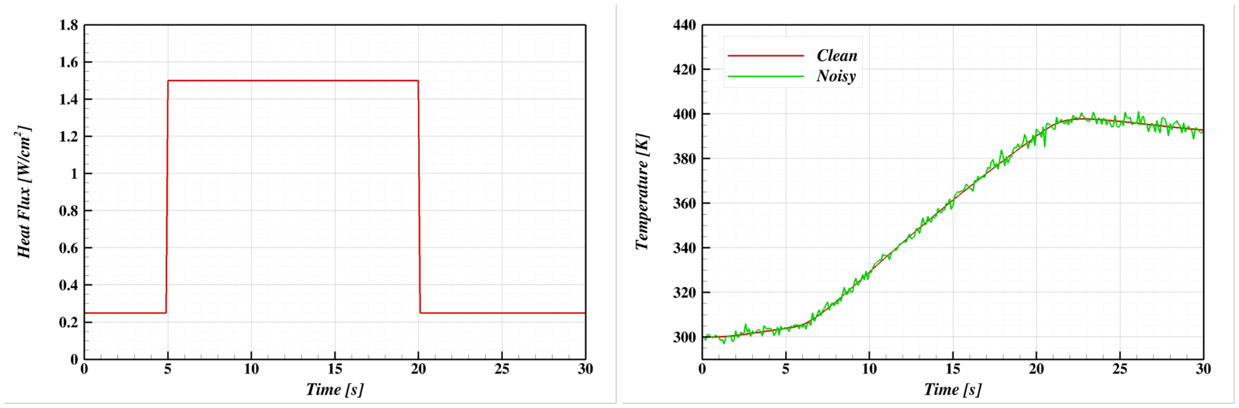
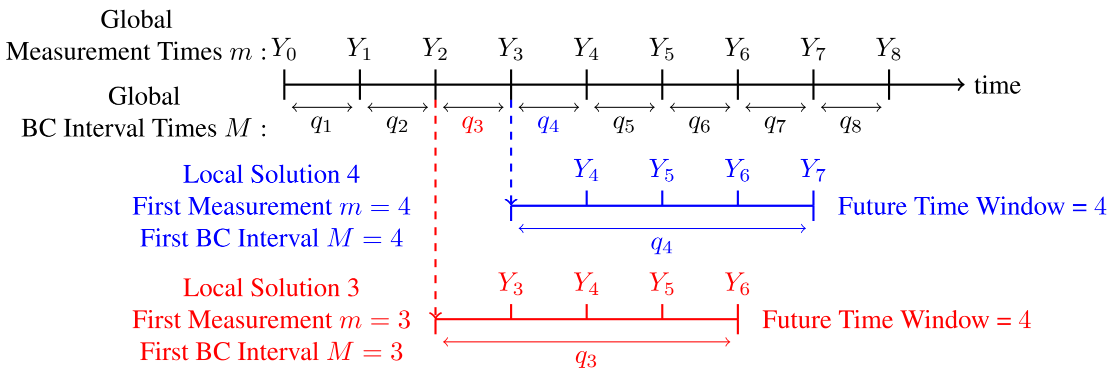


Figure 3. Truth heating profile and target temperature data for example problem

IV.C.2. Beck's Future Time Algorithm

The future time algorithm of Beck et. al.[2] can be obtained by setting $F = 1$, discretizing the boundary condition intervals to match the measurement times, and setting the length of the future time window to the appropriate number of 'future times' in Beck's algorithm. A couple steps of this algorithm are illustrated in Figure 4. Reconstruction results using this algorithm are shown in Figure 5. Increasing the future time increases the stability of the solution when noise exists in the data, but increasingly smears out the results near a discontinuity in the true boundary condition. This also tends to 'lead' a heating profile that varies smoothly in time.



Global
Measurement Times m :
Y_0 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7 Y_8
time
Global
BC Interval Times M :
q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8
Local Solution 4
First Measurement $m = 4$
First BC Interval $M = 4$
Y_4 Y_5 Y_6 Y_7
Future Time Window = 4
q_4
Local Solution 3
First Measurement $m = 3$
First BC Interval $M = 3$
Y_3 Y_4 Y_5 Y_6
Future Time Window = 4
q_3

Figure 4. Timeline illustrating two steps of the Future Time Algorithm of Beck et. al.[2]

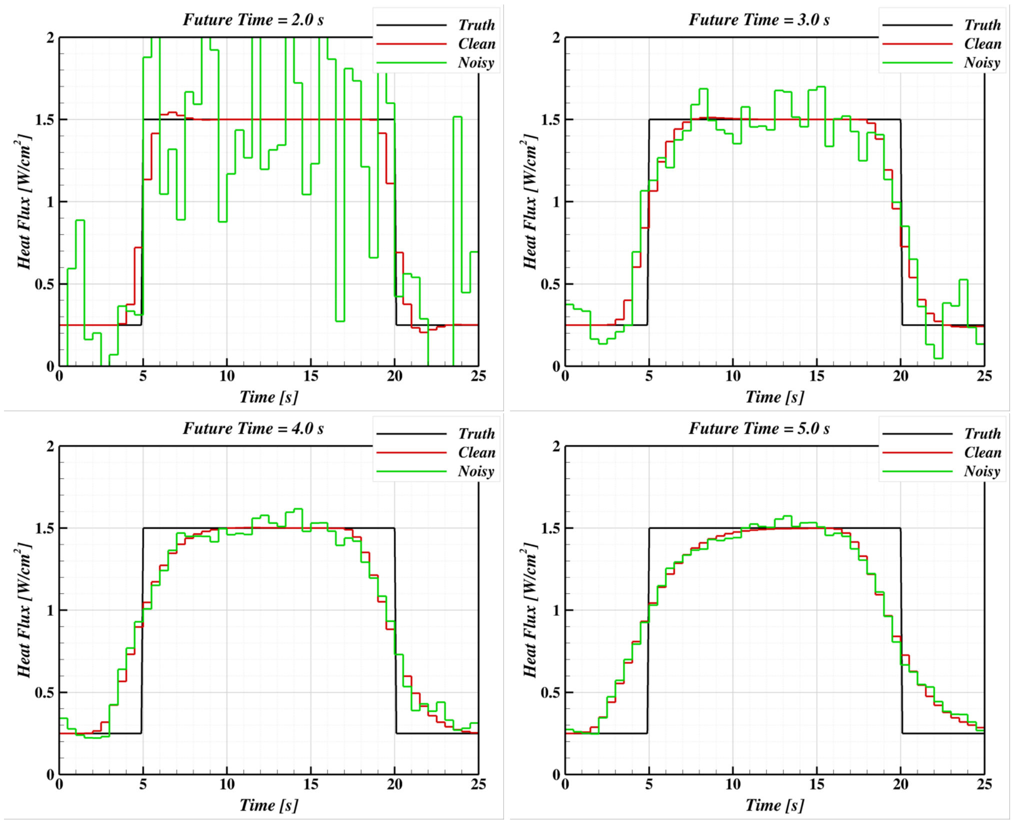
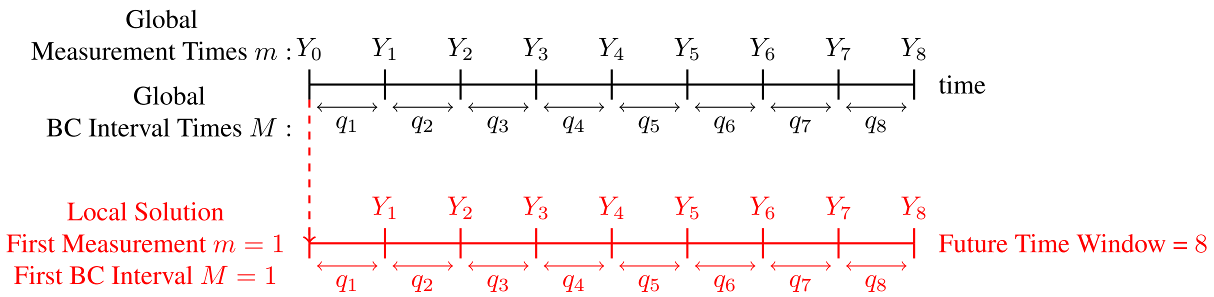


Figure 5. Reconstruction results using the Future Time Algorithm for several future time window lengths

IV.C.3. Whole Domain Algorithm

On the other end of the spectrum, a pure whole domain solution can be obtained by setting $F = f_{max}$ to include all intervals, discretizing the boundary condition intervals to match the measurement times, and setting the length of the future time window to the whole problem domain. Regularization will need to be specified to stabilize the solution, and all solution values must be retained. This is illustrated in Figure 6. Reconstruction results using this algorithm are shown in Figure 7. Increasing the regularization factor (first order, in this case) increases stability in the presence of noise as expected, but significantly smooths out discontinuities in the solution, as with the Future Time Algorithm.



Global Measurement Times m :
Y_0 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7 Y_8
time
Global BC Interval Times M :
q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8
Local Solution First Measurement $m = 1$ First BC Interval $M = 1$
Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7 Y_8
Future Time Window = 8
q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8

Figure 6. Timeline illustrating a pure whole domain solution

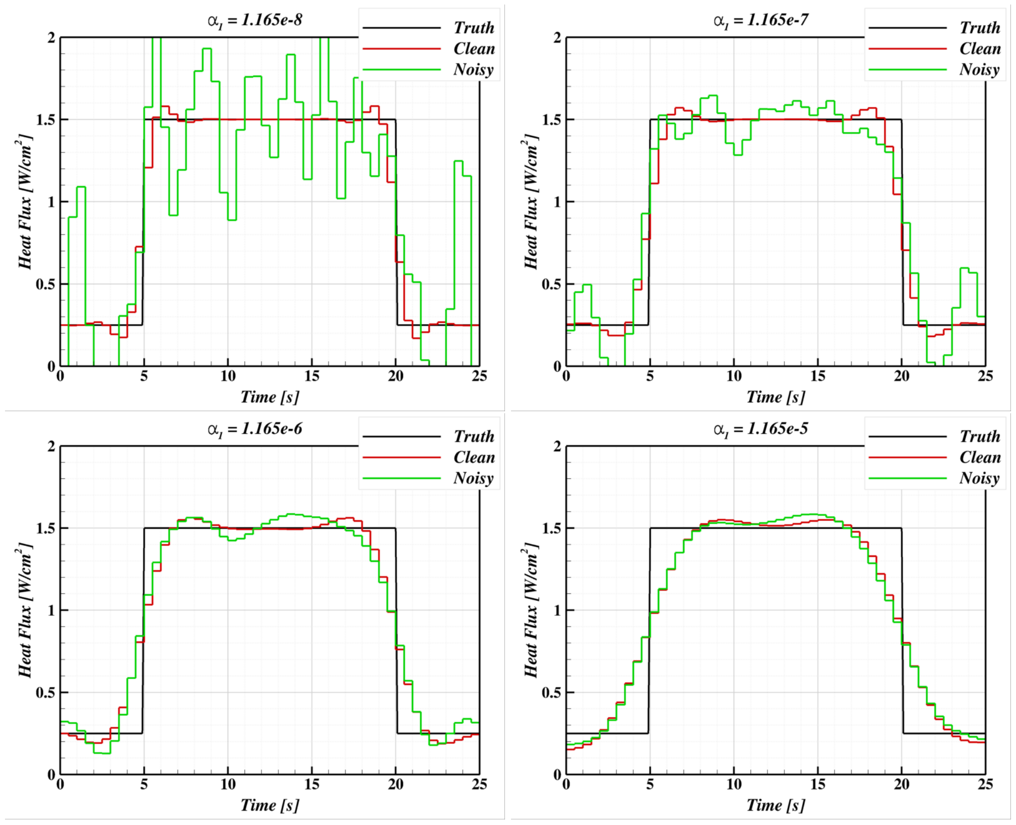


Figure 7. Reconstruction results using the Whole Domain Algorithm for several values of the first-order regularization factor α_1

IV.C.4. Hybrid Algorithms

An example illustrating the general hybrid algorithm, which we will refer to as the *Local Whole Domain Algorithm*, is obtained by setting $F = 4$, setting the length of the future time window to cover the F boundary condition intervals, and setting the terms retained to a value of 2. In Figure 8, The values of q_3 through q_6 are estimated in Local Solution 2, but only q_3 and q_4 are retained. The initial guess for q_5 and q_6 are updated based on the results of Local Solution 2, but they are re-estimated in Local Solution 3. Any value $< F$ can be used, but the value of 1 is a special case used enough to warrant a name, the *Sequential Whole Domain Algorithm*. I need to pull in discussion with sensitivity coefficients to explain why this is smart.

Reconstructions using this approach are shown in Figures 9 and 10. There is a lot of data to process (This will be optimized for the final draft), but three general observations can be made. First of all, it is interesting to note that with a future time window of 3 s (before the peak in sensitivity coefficient), larger regularization will drive the hybrid method to the future time algorithm (which makes sense considering it is essentially driving all of the flux values in a local solution to the same value). This is not quite as significant for the future time window of 5 s as there is more 'physics' for the algorithm to work with in the sensitivity coefficients. Second, in the clean reconstructions, reducing the regularization yields a solution that approaches the pure whole domain reconstruction. For the noisy reconstructions, the hybrid method seems to be more sensitive than the pure whole domain method for the same regularization parameter. Finally, the hybrid method appears to do the best when the future time window is long enough to capture the sensitivity coefficient peak.

V. Applications

- MSL flight data reconstruction
- Wind tunnel backface TC reconstruction (maybe)
- Wind tunnel protuberance example illustrating spatially-varying BC reconstruction

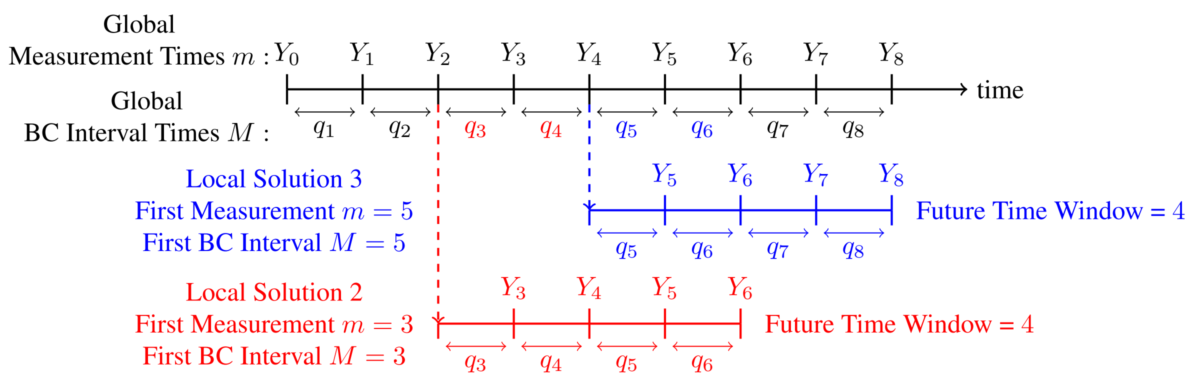


Figure 8. Timeline illustrating a Local Whole Domain solution

VI. Conclusions and Future Work

VII. Acknowledgements

We would like to gratefully acknowledge that this work has relied heavily on input and material provided by Ben Blackwell.

References

[1] Milad Mahzari and Robert D Braun. Time-Dependent Mars Entry Aeroheating Estimation from Simulated In-Depth Heat Shield Temperature Measurements. *Journal of Thermophysics and Heat Transfer*, 27(3):435–446, 2013.

[2] J. V. Beck, B. F. Blackwell, and C. R. St. Clair. *Inverse Heat Conduction Problems*. Wiley-Interscience, New York, NY, 1985.

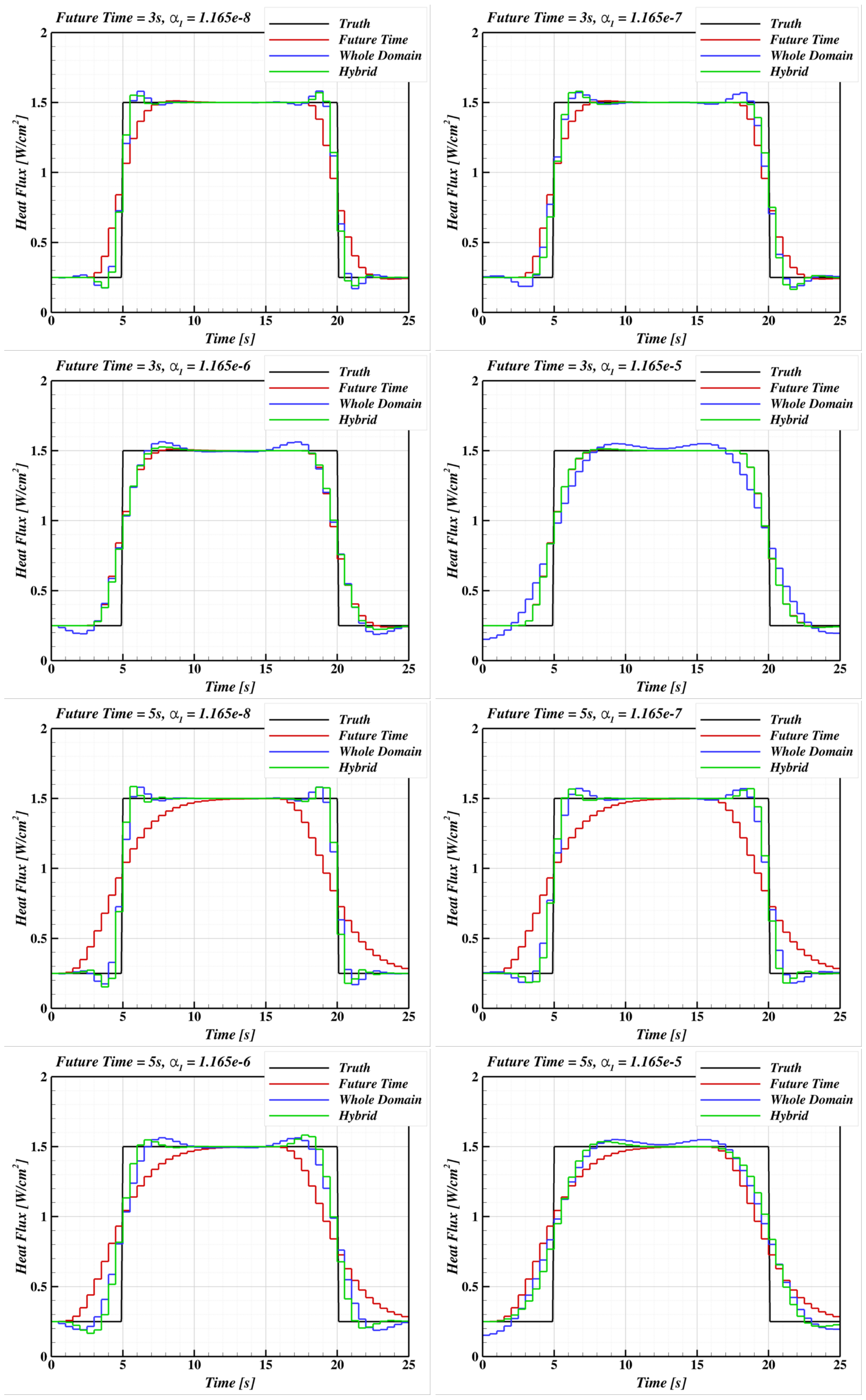


Figure 9. Reconstruction results of 'clean' TC data using the Sequential Whole Domain Algorithm for several values of the first-order regularization factor

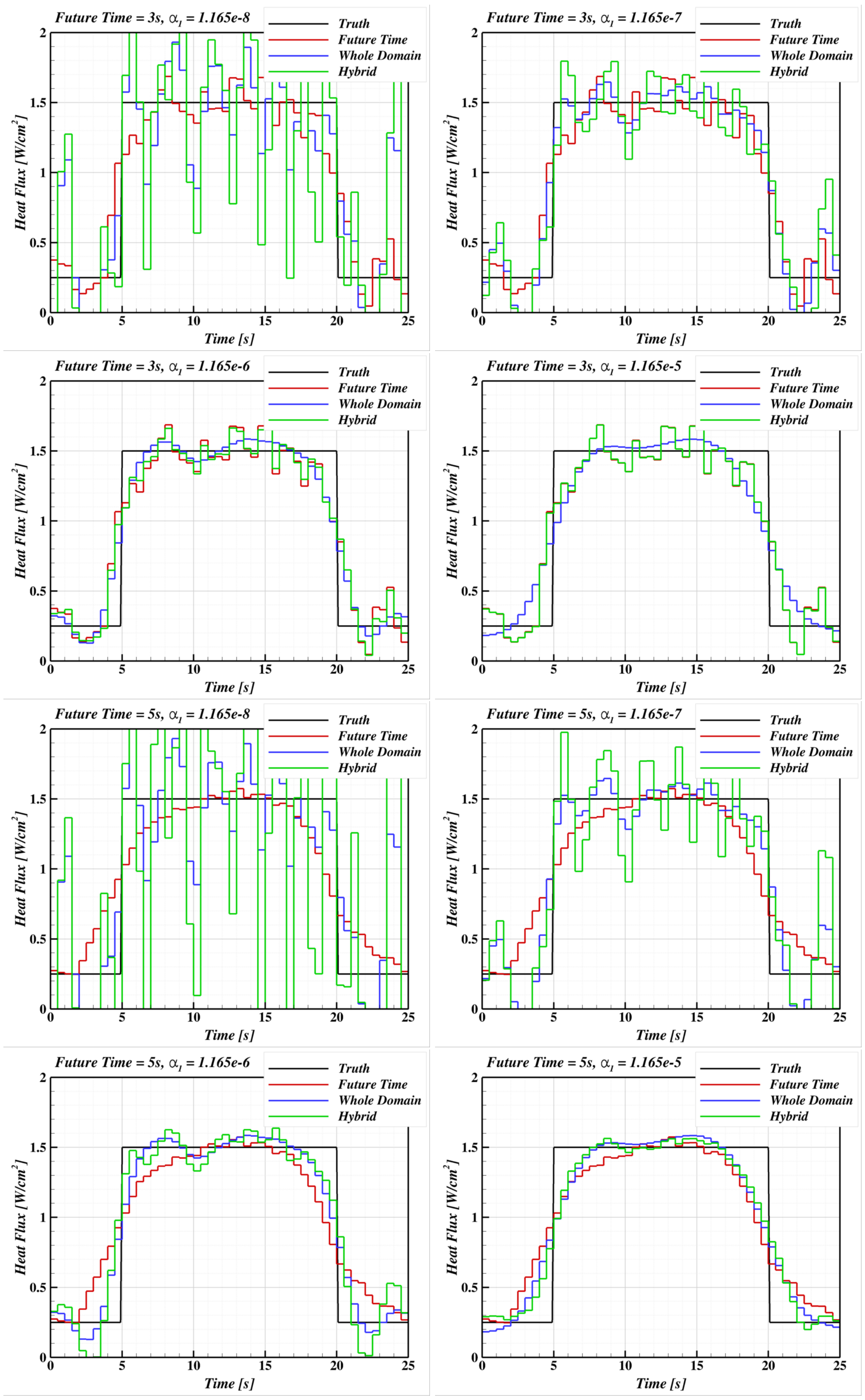


Figure 10. Reconstruction results of 'noisy' TC data using the Sequential Whole Domain Algorithm for several values of the first-order regularization factor