Readme file for Mike Smyth from Rui Wang (wang.810@osu.edu) and Noel Cresssie (ncressie@stat.osu.edu)

March 8, 2012

1 Introduction

This readme file has a brief description of the code and the associated algorithm. For further details, you can refer to a statistics paper written by Cressie and Wang (2012),which is available upon request.

There is one main function and two Matlab functions included in the package. The main function calls the two Matlab functions and stores the results. Each Matlab function performs one of: unit-free Hessian estimation, and the nonlinearity-bias calculation.

The data used as input are in a HDF format, but the file includes more information than we need. The Matlab code will read the file and retrieve the data it needs. Each file corresponds to the retrieval for one location at a particular time point. It includes all the data used and generated in the ACOS retrieval, such as the state vector (prior and retrieved values), radiances (measured and calculated), the prior covariance matrix (of both the state vector and the radiances), the pressure weighting function, the Jacobian matrix, etc. The state vector has n_α elements and the radiance vector has n_ε elements. Currently (version B2.8), $n_\alpha = 112$ and $n_\varepsilon = 2240$.

2 Function 'Hessian': Unit-free Hessian Estimation

2.1 Algorithm

2.1.1 Original Estimate

The Hessian array, $\{H_{ijk} : i = 1, \cdots, n_\varepsilon, j = 1, \cdots, n_\alpha, k = 1, \cdots, n_\alpha\}$, was estimated from a numerical derivative of the Jacobian, that is, by finite differencing. A typical element of the Hessian, H_{ijk}, can be estimated by

$$\widetilde{H}_{ijk} \equiv \frac{K_{ij}(\hat{\mathbf{x}} + \Delta_k \mathbf{e}_k) - K_{ij}(\hat{\mathbf{x}})}{\Delta_k}, \tag{1}$$

where $K_{ij}(\mathbf{x}) \equiv \frac{\partial F_i(\mathbf{x})}{\partial x_j}$ is the $n_\varepsilon \times n_\alpha (= 2240 \times 112$, in our case) Jacobian matrix; $\hat{\mathbf{x}}$ is the retrieved state vector; and $\mathbf{e}_k$ is the unit vector with 1 as the k-th element and 0 everywhere else. By the symmetry of the Hessian array with respect to index j and k, $H_{ijk} = H_{ikj}$. However, if we now take the numerical derivative of $K_{ik}(\mathbf{x})$ with respect to x_j, which results in:

$$\widetilde{H}_{ikj} \equiv \frac{K_{ik}(\hat{\mathbf{x}} + \Delta_j \mathbf{e}_j) - K_{ik}(\hat{\mathbf{x}})}{\Delta_j}, \tag{2}$$

we see that $\widetilde{H}_{ijk} \neq \widetilde{H}_{ikj}$, in general. That is, the estimator of the Hessian based on a direct numerical derivative, does not satisfy the important symmetry property.

2.1.2 Weighted Estimate

To obtain estimates of the Hessian array that *are* symmetric, we put the problem into the framework of statistical estimating equations. The estimating equation that is equivalent to (1) is:

$$\Delta_k H_{ijk} = K_{ij}(\hat{\mathbf{x}} + \Delta_k \mathbf{e}_k) - K_{ij}(\hat{\mathbf{x}}),$$

and the estimating equation that is equivalent to (2) is:

$$\Delta_j H_{ijk} = K_{ik}(\hat{\mathbf{x}} + \Delta_j \mathbf{e}_j) - K_{ik}(\hat{\mathbf{x}}).$$

A combined estimating equation is obtained by adding these two equations:

$$(\Delta_k + \Delta_j) H_{ijk} = K_{ij}(\hat{\mathbf{x}} + \Delta_k \mathbf{e}_k) - K_{ij}(\hat{\mathbf{x}}) + K_{ik}(\hat{\mathbf{x}} + \Delta_j \mathbf{e}_j) - K_{ik}(\hat{\mathbf{x}}).$$

From this estimating equation, we obtain the estimate:

$$\widehat{H}_{ijk} \equiv \widetilde{H}_{ijk} \left(\frac{\Delta_k}{\Delta_j + \Delta_k} \right) + \widetilde{H}_{ikj} \left(\frac{\Delta_j}{\Delta_j + \Delta_k} \right). \tag{3}$$

Notice that now the original estimates are *weighted* according to the numerical-derivative increments $\{\Delta_j : j = 1, \cdots, n_\alpha\}$. It is easy to see that $\widehat{H}_{ijk} = \widehat{H}_{ikj}$, and (3) is the Hessian estimate we shall use.

2.1.3 Unit-free Hessian Estimate

All our research has pointed towards the importance of doing as many of the calculations in a unit-free environment. In previous presentations, we have defined the unit-free Jacobian:

$$\phi_{ij} \equiv K_{ij} \sigma_j / \sigma_{\varepsilon,i} \,,$$

and the unit-free Hessian:

$$\psi_{ijk} \equiv H_{ijk} \sigma_j \sigma_k / \sigma_{\varepsilon,i} \,,$$

where $\{\sigma_{\varepsilon,i} : i = 1, \cdots, n_\varepsilon\}$ are the standard deviations of the measurement error and $\{\sigma_j : j = 1, \cdots, n_\alpha\}$ are the prior standard deviations. The inverse calculation yields,

$$\begin{aligned} K_{ij} &= \phi_{ij} \sigma_{\varepsilon,i} / \sigma_j \\ H_{ijk} &= \psi_{ijk} \sigma_{\varepsilon,i} / (\sigma_j \sigma_k) \,. \end{aligned}$$

Substituting these into equation (3), we can derive the unit-free Hessian estimate as follows:

$$\begin{aligned} \widehat{\psi}_{ijk} \sigma_{\varepsilon,i} / (\sigma_j \sigma_k) &= \frac{(\sigma_{\varepsilon,i}/\sigma_j)[\phi_{ij}(\hat{\mathbf{x}} + \Delta_k \mathbf{e}_k) - \phi_{ij}(\hat{\mathbf{x}})] + (\sigma_{\varepsilon,i}/\sigma_k)[\phi_{ik}(\hat{\mathbf{x}} + \Delta_j \mathbf{e}_j) - \phi_{ik}(\hat{\mathbf{x}})]}{\Delta_j + \Delta_k} \,, \\ \Rightarrow \quad \widehat{\psi}_{ijk} &= \frac{\sigma_k[\phi_{ij}(\hat{\mathbf{x}} + \Delta_k \mathbf{e}_k) - \phi_{ij}(\hat{\mathbf{x}})] + \sigma_j[\phi_{ik}(\hat{\mathbf{x}} + \Delta_j \mathbf{e}_j) - \phi_{ik}(\hat{\mathbf{x}})]}{\Delta_j + \Delta_k} \,, \\ \Rightarrow \quad \widehat{\psi}_{ijk} &= \frac{\phi_{ij}(\hat{\mathbf{x}} + \Delta_k \mathbf{e}_k) - \phi_{ij}(\hat{\mathbf{x}})}{\Delta_k / \sigma_k} \cdot \frac{\Delta_k}{\Delta_j + \Delta_k} + \frac{\phi_{ik}(\hat{\mathbf{x}} + \Delta_j \mathbf{e}_j) - \phi_{ik}(\hat{\mathbf{x}})}{\Delta_j / \sigma_j} \cdot \frac{\Delta_j}{\Delta_j + \Delta_k} \,. \end{aligned} \tag{4}$$

Based on the unit-free Hessian estimate $\widehat{\psi}_{ijk}$ given above, we then obtain the Hessian estimate,

$$\widehat{H}_{ijk} = \widehat{\psi}_{ijk} \sigma_{\varepsilon,i} / (\sigma_j \sigma_k),$$

which is algebraically equivalent to the weighted estimate given by (3). Our decision to estimate unit-free Hessians first is based on keeping everything on the same scale (and so numerically stable) until the very last step of *any* of our calculations.

2.1.4 Choice of $\boldsymbol{\Delta_j}$

Currently, $\{\Delta_j : j = 1, \cdots, n_\alpha\}$ are chosen based on intuition. It is instructive to write the unit-free equivalent of $\{\Delta_j : j = 1, \cdots, n_\alpha\}$. Define

$$\delta_j \equiv \Delta_j / \sigma_j \,,$$

which is unit free, and hence we can write

$$\Delta_j = \delta_j \sigma_j \,.$$

When the Hessian is computed numerically, Δ_j is prespecified. In Table 1 below, we list those values of Δ_j and δ_j. These values were prespecified by Chris O'Dell when he calculated the numerical derivative, and are subject to modification.

Element Names	**Element Index**	Δ_j	δ_j
CO2 concentration level $1, \cdots, 20$	$1, \cdots, 20$	1e-7	0.0021-0.0696
H2O scale factor	21	0.01	0.02
Surface Pressure in Pa	22	10.0	0.025
Temperature Offset in K	23	0.1	0.02
Ice Cloud logarithmic Concentration	24-43	0.1	0.0434
Kahn Type 2B logarithmic Concentration	44-63	0.1	0.0434
Kahn Type 3B logarithmic Concentration	64-83	0.1	0.0434
Water Cloud logarithmic Concentration	84-103	0.1	0.0434
Mean Albedo Bands 1-3	104,106,108	0.001	0.001
Albedo slope Bands 1-3	105,107,109	1e-6	0.002
Dispersion Offset Bands 1-3	110-112	5e-4	0.001

Table 1: Prespecified perturbations $\{\Delta_j : j = 1, \cdots, n_\alpha\}$

2.1.5 Discussion

The computation of the values $K_{ij}(\widehat{\mathbf{x}} + \Delta_k \mathbf{e}_k)$ were carried out by Chris O'Dell. It may be better to calculate $K_{ij}(\widehat{\mathbf{x}} + 0.5\Delta_k \mathbf{e}_k)$ and $K_{ij}(\widehat{\mathbf{x}} - 0.5\Delta_k \mathbf{e}_k)$ and take a symmetric derivative, or there may be more sophisticated software available that will perform numerical differentiation? Suggestions here would be appreciated. Whatever is used, the Hessian estimate $\{\widehat{H}_{ijk}\}$ must satisfy $\widehat{H}_{ijk} = \widehat{H}_{ikj}$, and we can be involved to ensure this.

2.2 Code

The Matlab function "Hessian" calculates the unit-free Hessian. The Jacobian matrix is first read from the data file, and since

$$\phi_{ij} \equiv K_{ij} \sigma_j / \sigma_{\varepsilon,i} \,; \;\; i = 1, \cdots, 2240, \, j = 1, \cdots, 112 \,,$$

the unit-free Jacob can be calculated by $\widehat{\boldsymbol{\Phi}} = \boldsymbol{\Sigma}_\varepsilon^{-1} \widehat{\mathbf{K}} \boldsymbol{\Sigma}_\alpha$, where the diagonal matrix $\boldsymbol{\Sigma}_\alpha = \text{diag}(\sigma_1, \cdots, \sigma_{n_\alpha})$ and the diagonal matrix $\boldsymbol{\Sigma}_\varepsilon = \text{diag}(\sigma_{\varepsilon,1}, \cdots, \sigma_{\varepsilon,n_\varepsilon})$. Note that $\boldsymbol{\Sigma}_\alpha$ and $\boldsymbol{\Sigma}_\varepsilon$ have standard deviations (not variances) on the diagonal.

After the unit-free Jacobian matrices are calculated, the term $\dfrac{\phi_{ij}(\hat{\mathbf{x}} + \Delta_k \mathbf{e}_k) - \phi_{ij}(\hat{\mathbf{x}})}{\Delta_k / \sigma_k}$ in equation (4) is calculated for each (i, j, k). Then, in the final step of the code, the estimated unit-free Hessian is calculated based on equation (4).

3 Function 'Bias': The Nonlinearity-Bias Calculation

3.1 Algorithm

3.1.1 Original Algorithm

From the statistical research done by Cressie and Wang (2012), the bias of state vector can be estimated by

$$\begin{aligned}\widetilde{\mathbf{bias}}(\mathbf{x}_\alpha) &\equiv (1/2)\begin{pmatrix}\left(\mathrm{vec}\left(\dfrac{\partial \mathbf{A}(\mathbf{x}_\alpha)_{\text{1st-row}}}{\partial \mathbf{x}_\alpha}\right)\right)' \\ \left(\mathrm{vec}\left(\dfrac{\partial \mathbf{A}(\mathbf{x}_\alpha)_{\text{2nd-row}}}{\partial \mathbf{x}_\alpha}\right)\right)' \\ \vdots \\ \left(\mathrm{vec}\left(\dfrac{\partial \mathbf{A}(\mathbf{x}_\alpha)_{n_\alpha\text{th-row}}}{\partial \mathbf{x}_\alpha}\right)\right)'\end{pmatrix} \cdot \mathrm{vec}\left(\widetilde{\mathrm{MSPE}}(\mathbf{x}_\alpha) + 2\mathbf{S}_\alpha \mathbf{A}(\mathbf{x}_\alpha)'\right) \\ &- (1/2)\begin{pmatrix}\left(\mathrm{vec}\left(\dfrac{\partial \mathbf{G}(\mathbf{x}_\alpha)_{\text{1st-row}}}{\partial \mathbf{x}_\alpha}\right)\right)' \\ \left(\mathrm{vec}\left(\dfrac{\partial \mathbf{G}(\mathbf{x}_\alpha)_{\text{2nd-row}}}{\partial \mathbf{x}_\alpha}\right)\right)' \\ \vdots \\ \left(\mathrm{vec}\left(\dfrac{\partial \mathbf{G}(\mathbf{x}_\alpha)_{n_\alpha\text{th-row}}}{\partial \mathbf{x}_\alpha}\right)\right)'\end{pmatrix} \cdot \mathrm{vec}\left(\mathbf{K}(\mathbf{x}_\alpha)\cdot \widetilde{\mathrm{MSPE}}(\mathbf{x}_\alpha) - 2\mathbf{S}_\varepsilon \mathbf{G}(\mathbf{x}_\alpha)'\right).\end{aligned}$$

Or, we have an algebraically equivalent form:

$$\begin{aligned}\widetilde{\mathrm{bias}}(\mathbf{x}_\alpha)_j &= \frac{1}{2}\sum_{k=1}^{n_\alpha}\left[\frac{\partial \mathbf{A}(\mathbf{x}_\alpha)}{\partial x_{k,\alpha}}\left(\widetilde{\mathrm{MSPE}}(\mathbf{x}_\alpha) + 2\mathbf{S}_\alpha \mathbf{A}(\mathbf{x}_\alpha)'\right)\right]_{jk} \\ &- \frac{1}{2}\sum_{k=1}^{n_\alpha}\left[\frac{\partial \mathbf{G}(\mathbf{x}_\alpha)}{\partial x_{k,\alpha}}\left(\mathbf{K}(\mathbf{x}_\alpha)\cdot \widetilde{\mathrm{MSPE}}(\mathbf{x}_\alpha) - 2\mathbf{S}_\varepsilon \mathbf{G}(\mathbf{x}_\alpha)'\right)\right]_{jk}. \qquad (5)\end{aligned}$$

Since we have the relation

$$\begin{aligned}\mathbf{G}(\mathbf{x}) &\equiv \{\mathbf{S}_\alpha^{-1} + \mathbf{K}(\mathbf{x})'\mathbf{S}_\varepsilon^{-1}\mathbf{K}(\mathbf{x})\}^{-1}\mathbf{K}(\mathbf{x})'\mathbf{S}_\varepsilon^{-1} \\ \mathbf{A}(\mathbf{x}) &\equiv \mathbf{G}(\mathbf{x})\mathbf{K}(\mathbf{x}),\end{aligned}$$

$\dfrac{\partial \mathbf{A}(\mathbf{x}_\alpha)}{\partial x_{k,\alpha}}$ and $\dfrac{\partial \mathbf{G}(\mathbf{x}_\alpha)}{\partial x_{k,\alpha}}$ in equation (5) are both functions of $\dfrac{\partial \mathbf{K}(\mathbf{x}_\alpha)}{\partial x_{k,\alpha}} = \mathbf{H}(:,:,k)$.

3.1.2 Unit-free Calculation

To avoid unexpected round-off error, we base as much of our Matlabe code on algebraically equivalent unit-free calculations. Recall the definition of the diagonal matrix $\mathbf{\Sigma}_\alpha = \mathrm{diag}(\sigma_1, \cdots, \sigma_{n_\alpha})$ and the diagonal matrix $\mathbf{\Sigma}_\varepsilon = \mathrm{diag}(\sigma_{\varepsilon,1}, \cdots, \sigma_{\varepsilon,n_\varepsilon})$; hence, we can write the unit-free Jacobian as:

$$\mathbf{\Phi}(\mathbf{x}) = \mathbf{\Sigma}_\varepsilon^{-1}\mathbf{K}(\mathbf{x})\mathbf{\Sigma}_\alpha, \qquad (6)$$

and the unit-free Hessian as:

$$\mathbf{\Psi}_{(:,:,k)}(\mathbf{x}) = \mathbf{\Sigma}_\varepsilon^{-1}\mathbf{H}_{(:,:,k)}(\mathbf{x})\mathbf{\Sigma}_\alpha \sigma_k, \qquad (7)$$

where $\mathbf{\Psi}_{(:,:,k)}(\mathbf{x})$ and $\mathbf{H}_{(:,:,k)}(\mathbf{x})$ are $n_\varepsilon \times n_\alpha$ matrices that are obtained by fixing the third dimension of their corresponding three-dimensional arrays at level k.

Since we have the relation,

$$\begin{aligned} \mathbf{G}(\mathbf{x}) &\equiv \{\mathbf{S}_\alpha^{-1} + \mathbf{K}(\mathbf{x})'\mathbf{S}_\varepsilon^{-1}\mathbf{K}(\mathbf{x})\}^{-1}\mathbf{K}(\mathbf{x})'\mathbf{S}_\varepsilon^{-1} \\ \mathbf{A}(\mathbf{x}) &\equiv \mathbf{G}(\mathbf{x})\mathbf{K}(\mathbf{x}), \end{aligned}$$

recall that $\dfrac{\partial \mathbf{A}(\mathbf{x}_\alpha)}{\partial x_{k,\alpha}}$ and $\dfrac{\partial \mathbf{G}(\mathbf{x}_\alpha)}{\partial x_{k,\alpha}}$ in equation (5) can both be written as functions of $\dfrac{\partial \mathbf{K}(\mathbf{x}_\alpha)}{\partial x_{k,\alpha}} = \mathbf{H}_{(:,:,k)}(\mathbf{x}_\alpha)$, $\mathbf{K}(\mathbf{x}_\alpha)$, $\mathbf{S}_\alpha$ and $\mathbf{S}_\varepsilon$. Equation (5) also includes the term $\widetilde{\mathrm{MSPE}}(\mathbf{x}_\alpha)$, for which we have the formula,

$$\widetilde{\mathrm{MSPE}}(\mathbf{x}_\alpha) \equiv (\mathbf{A}(\mathbf{x}_\alpha) - \mathbf{I})\mathbf{S}_\alpha(\mathbf{A}(\mathbf{x}_\alpha) - \mathbf{I})' + \mathbf{G}(\mathbf{x}_\alpha)\mathbf{S}_\varepsilon\mathbf{G}(\mathbf{x}_\alpha)'.$$

Plugging in the relations (6) and (7) into equation (5), the matrix $\dfrac{\partial \mathbf{A}(\mathbf{x}_\alpha)}{\partial x_{k,\alpha}}\left(\widetilde{\mathrm{MSPE}}(\mathbf{x}_\alpha) + 2\mathbf{S}_\alpha\mathbf{A}(\mathbf{x}_\alpha)'\right)$ in the first row can be written as:

$$\begin{aligned} &\mathbf{\Sigma}_\alpha \mathbf{\Upsilon}(\mathbf{x}_\alpha)\left[\mathbf{\Psi}_{(:,:,k)}(\mathbf{x}_\alpha)'\mathbf{\Phi}(\mathbf{x}_\alpha) + \mathbf{\Phi}(\mathbf{x}_\alpha)'\mathbf{\Psi}_{(:,:,k)}(\mathbf{x}_\alpha)\right] \cdot \left[\mathbf{I} - \mathbf{\Upsilon}(\mathbf{x}_\alpha)\mathbf{\Phi}(\mathbf{x}_\alpha)'\mathbf{\Phi}(\mathbf{x}_\alpha)\right] \\ &\quad \cdot \left[\mathbf{I} + 2\mathbf{\Sigma}_\alpha^{-1}\mathbf{S}_\alpha\mathbf{\Sigma}_\alpha^{-1}\mathbf{\Phi}(\mathbf{x}_\alpha)'\mathbf{\Phi}(\mathbf{x}_\alpha)\right]\mathbf{\Upsilon}(\mathbf{x}_\alpha)\mathbf{\Sigma}_\alpha/\sigma_k, \end{aligned} \tag{8}$$

where

$$\mathbf{\Upsilon}(\mathbf{x}_\alpha) \equiv \left[\left(\mathbf{\Sigma}_\alpha^{-1}\mathbf{S}_\alpha\mathbf{\Sigma}_\alpha^{-1}\right)^{-1} + \mathbf{\Phi}(\mathbf{x}_\alpha)'\mathbf{\Phi}(\mathbf{x}_\alpha)\right]^{-1}$$

is a unit-free $n_\alpha \times n_\alpha$ (here, 112×112) matrix. Finally, the matrix $\dfrac{\partial \mathbf{G}(\mathbf{x}_\alpha)}{\partial x_{k,\alpha}}\left(\mathbf{K}(\mathbf{x}_\alpha) \cdot \widetilde{\mathrm{MSPE}}(\mathbf{x}_\alpha) - 2\mathbf{S}_\varepsilon\mathbf{G}(\mathbf{x}_\alpha)'\right)$ in the second row of equation (5) can be written as:

$$\begin{aligned} &-\mathbf{\Sigma}_\alpha\mathbf{\Upsilon}(\mathbf{x}_\alpha)\left[\left\{\mathbf{\Psi}_{(:,:,k)}(\mathbf{x}_\alpha)'\mathbf{\Phi}(\mathbf{x}_\alpha) + \mathbf{\Phi}(\mathbf{x}_\alpha)'\mathbf{\Psi}_{(:,:,k)}(\mathbf{x}_\alpha)\right\}\mathbf{\Upsilon}(\mathbf{x}_\alpha)\mathbf{\Phi}(\mathbf{x}_\alpha)' + \mathbf{\Psi}_{(:,:,k)}(\mathbf{x}_\alpha)'\right] \cdot \\ &\quad \mathbf{\Phi}(\mathbf{x}_\alpha)\mathbf{\Upsilon}(\mathbf{x}_\alpha)\mathbf{\Sigma}_\alpha/\sigma_k. \end{aligned} \tag{9}$$

We then have a unit-free calculation of bias by plugging equations (8) and (9) into equation (5).

3.2 Code

The Matlab function "Bias" calculates the estimated nonlinearity bias given by equation (5). After the necessary quantities are read from the data file, the expressions (8) and (9) are evaluated. Then, plugging in these values into (5), we get the bias estimate.

4 Reference

Cressie, N. and Wang, R. (2012). Statistical properties of the state obtained by solving a nonlinear multivariate inverse problem, submitted (and available on request).