# Exercises

P. J. Rayner,[1,5] R. Giering,[2] T. Kaminski,[3] R. Ménard,[4] R. Todling[4] and C. M. Trudinger[5]

This chapter presents a series of exercises related to the tutorial material presented in some of the earlier papers. Its aim is to help students familiarize themselves with that material by the use of some simple examples. Further, some of the examples provide skeleton applications on which students can build. The software for the exercises is supplied on the accompanying CD-ROM.

## 1. Introduction

This chapter contains a set of exercises which accompany the tutorial material in *Prinn* [1999], *Giering* [1999], *Enting* [1999], *Todling* [1999] and *Ménard* [1999]. The exercises attempt to span and unite some of the material presented elsewhere. For example, a simple box model of atmospheric transport is used in three different approaches to similar problems. This reduces the learning overhead for a student interested in the methods and allows students to compare various approaches.

The material treats the three dominant methods presented throughout the book.

1. Adjoint methods. A series of both analytic and computer-based exercises demonstrate the principles of adjoint code construction and its application to sensitivity analysis and optimization.

2. Green's function methods. A two-box model is constructed and the Green's function (or synthesis inversion) method is used to calculate estimates of sources and concentrations as well as their uncertainties.

3. Kalman filtering. A set of exercises explores the various properties of the Kalman filter and smoother. These exercises range from a simple application of the two-box model through to investigating the stability of the Kalman filter. Various more computationally tractable approximations to the Kalman filter are also presented.

Students interested in one of the methods only should be able to approach the relevant section without needing to work through the other sections. This is true even where the same model is used in several sections and we make deliberate comparison of the methods. Students wishing to undertake any of the exercises using the simple box model (Exercises 1–23) should read section 2 first.

The exercises are presented with the simplest cases first. The sections each contain at least one computer-based example to provide hands-on experience. The code and data for the computer-based examples are provided on the accompanying CD-ROM. The `README` file on the CD-ROM gives an index of the directories where each exercise can be found. Instalation instructions for each exercise are found in the `README` file in the relevant directory. Solutions and some discussion are provided at the end of the chapter. The computer exercises have been written in three different languages reflecting the different authors' preferences. Some of the exercises exist in two languages, Fortran and IDL. Some are written in the scripting language of the algebra and graphics package matlab. Some examples require the use of the automatic differentiation tool TAMC described in *Giering* [1999]. TAMC operates on Fortran source code only so these exercises are only available in Fortran.

The outline of this chapter is as follows:

- Introduce the two-box model used subsequently;
- Introduce the automatic differentiation system TAMC;
- Demonstrate the principles of adjoint code generation [*Giering*, 1999] and use TAMC for a series of sensitivity and optimization studies;
- Demonstrate a Bayesian synthesis inversion [*Enting*, 1999] and solve for sources in the two-box model;
- Present a set of examples demonstrating the Kalman filter [*Prinn*, 1999; *Todling*, 1999; *Ménard*, 1999] and suboptimal variants of it in some simple dynamical and chemical systems.

## 2. Box models of transport

Box models generally divide the model domain (the atmosphere for example) into a number of large boxes and describe the exchange of constituents between them. In the simplest case of two boxes representing the two hemispheres we can write very simple equations for the evolution of the concentration of a constituent. In general the rate of change of tracer mass in a hemisphere is the sum of transport (between hemispheres) and sources.

$$\frac{M}{2}\frac{\partial c}{\partial t} = T + S \tag{1}$$

where $M$ is the mass of the atmosphere, $c$ the mass mixing ratio (mass of tracer divided by mass of air), $S$ the source (as an emission rate) within the hemisphere and $T$ the tracer mass-flux (in mass per time) from the other hemisphere. $S$ may contain chemical sources or sinks which may depend on $c$. A common form for $S$ is a simple first-order or linear loss rate

$$S = E - \frac{c}{\lambda} \tag{2}$$

where $E$ is an emission rate and $\lambda$ is the chemical lifetime. For radioactive decay $\ln 2 \times \lambda$ is the half-life.

To produce a simple box model we need to express $T$ in terms of $c$. For a two-box model this is most simply done by assuming a transport proportional to the difference between the average concentrations in the two hemispheres; a diffusive approximation. Thus

$$T = -\kappa \Delta c \tag{3}$$

where $\Delta c$ is the concentration difference between the hemispheres. Substitution yields the usual form for the two-box model, namely

$$\frac{M}{2}\frac{\partial c_1}{\partial t} = s_1 - \kappa(c_1 - c_2) \tag{4a}$$
$$\frac{M}{2}\frac{\partial c_2}{\partial t} = s_2 - \kappa(c_2 - c_1) \tag{4b}$$

where index 1 refers to the northern hemisphere. The expression $\frac{M}{2\kappa}$ represents a time and is conventionally defined as the mixing time or exchange time $\tau$. In the absence of sources and with a fixed concentration in one hemisphere the concentration in the other hemisphere would equilibrate exponentially with decay time $\tau$. Physically, $\tau$ represents the average time for an air parcel in one hemisphere to move to the other hemisphere. It can be regarded as the time in which the air in one hemisphere will be mixed into the other hemisphere. Equivalently $\kappa$ represents the fraction of the air in one hemisphere mixed into the other hemisphere per unit time. $\tau$ is used as a simple parameter to summarize the rate of exchange between hemispheres in atmospheric transport models (see, for example, *Law et al.* [1996] Table 3).

The convention in the literature is to reformulate equation (4) in terms of the average concentration, $c_+ = \frac{c_1+c_2}{2}$, and the difference or gradient between the hemispheres, $c_- = c_2 - c_1$. (The average concentration is proportional to the total tracer mass because both hemispheres have equal masses.) Also we introduce $s_+ = s_1 + s_2$ for the sum of the sources and

$s_- = s_2 - s_1$ for their difference. Adding and subtracting equations (4a) and (4b) yields

$$M\frac{\partial c_+}{\partial t} = s_+ \qquad (5a)$$

$$\frac{M}{2}\frac{\partial c_-}{\partial t} = s_- - 2\kappa c_- \qquad (5b)$$

Frequently we will divide by the atmospheric mass so that, e.g., equation (5b) can be written

$$\frac{1}{2}\frac{\partial c_-}{\partial t} = \frac{s_-}{M} - \frac{c_-}{\tau} \qquad (6)$$

In the absence of sources, $c_-$ will evolve according to

$$c_-(t) = c_-(0)e^{-2t/\tau} \qquad (7)$$

This means the decay time for $c_-$ is $\frac{\tau}{2}$. This decay time is different from the decay time derived above with one fixed concentration. Taking account of the return flow from the other hemisphere halves the decay time.

If $s_-$ is constant then asymptotically, i.e. for large $t$,

$$c_-(t) = \frac{s_-}{M}\tau \qquad (8)$$

Note that various slightly different definitions of $\tau$ are possible usually depending on whether the mass of a hemisphere or the total atmosphere is used. The above equations hold if $\tau$ is constant.

Several exercises in the following sections are based around this two-box model with constant exchange time. We use two datasets. The first is a set of emissions and concentrations for the gas methyl chloroform ($CH_3CCl_3$) which has been previously used to infer interhemispheric exchange time and chemical lifetime for this trace gas by *Prinn et al.* [1995]. We shall replicate this calculation using a simple nonlinear optimization of the two-box model outlined above. We shall also use a Kalman filter version of the model to infer hemispheric sources for this gas under various conditions. The second dataset is a trivial example with constant sources and exchange time. We will use this dataset to compare the Green's function and Kalman filter approaches to deducing sources from concentrations.

## 3. Using TAMC for Automatic Differentiation

### *3.1. Introduction*

*Giering* [1999] has summarized applications for codes which evaluate the derivatives of models. In particular many of the problems encountered throughout this book require the calculation of derivatives in some form. For example, many optimization calculations (such as estimating optimal parameter values) require some knowledge of the derivative of some cost function with respect to the parameters (see section 4.1.2). In the Green's function approaches the derivative of simulated concentrations with respect to source parameters constitutes the Jacobian matrix that is to be inverted (see section 4.2). Recursive algorithms like the Kalman filter may require a linearized version of the model to step either the state or its covariance forward. This linearized model requires the calculation of the derivative of the new state with respect to the current state (see section 4.3). In a nonlinear model this derivative will depend on the actual state. Finally, the derivative may be of direct interest, since it represents the sensitivity of some output (e.g. mean surface temperature) to some inputs (e.g. cloud amount, $CO_2$ concentration). In section 4.1.2 we show an example of the sensitivity of concentrations at the end of a model integration to sources throughout the integration.

In this section we briefly outline the use of the Tangent linear and Adjoint Model Compiler (TAMC) which has been introduced by *Giering* [1999]. While we discuss the generalities of the techniques we focus on a specific example, namely the two-box model introduced in section 2. TAMC applies a technique called automatic differentiation [*Griewank*, 1989] , which has been briefly introduced by *Giering* [1999]. The source-to-source translator is available remotely and various utilities are described to facilitate some of the uses of the code described above. A brief description of both TAMC and the utilities is given in section 3.2. Finally a set of exercises explains the use of TAMC for both sensitivity and optimization calculations with the two-box model. Many of the exercises require actual use of TAMC and hence a computer capable of accessing the TAMC server. For those readers who want to do the exercises without such access, the output of the program is included in figures. More details about the topics introduced in section 3.2 can be found in *Giering* [1997] and *Giering and Kaminski* [1998]. In particular when doing the exercises it is recommended to have these documents available.

### *3.2. TAMC and TAMLINK*

Using the concept of automatic differentiation, the task of constructing adjoint or tangent linear code can be based on simple rules, such as those described in *Giering and Kaminski* [1998]. These simple rules can be applied automatically, e.g. by source to source translation programs. There are a number of these programs, e.g. Odyssée [*Rostaing et al.*, 1993], GRESS [*Horwedel*, 1991], , or TAMC [*Giering*, 1997] for reverse mode and ADIFOR [*Bischof*, 1992] or TAMC for forward mode (see *Bischof* [1999] for more references). In this section the program TAMC is introduced. The focus lies on features that are necessary to understand the examples, see *Giering* [1997] and *Giering and Kaminski* [1998] for more details.

To briefly summarize the material in *Giering* [1999], adjoint and tangent linear codes propagate derivatives. In tangent linear or forward mode, the first set of derivatives calculated are those of the first set of intermediate results with respect to the input variables; then the next set of intermediate results with respect to the first and this process continues to the output variables. In adjoint or reverse mode, the first derivatives calculated are those of the *output* variables with respect to the last set of intermediate results, then of the last intermediate with respect to the second last and so on back to the inputs. In TAMC there is a one to one correspondence of variables in the model code to variables which hold the derivatives. Those variables that either have no influence on the output variables or do not depend on the input variables are called passive variables. For passive variables no derivatives need to be propagated [*Giering and Kaminski*, 1998]. In contrast, active variables are those that influence the output variables and also depend on the input variables. For each active variable, an adjoint (or tangent linear variable) is declared in the adjoint (or tangent linear) code, to hold the corresponding derivative. Tangent linear or adjoint statements are generated for each statement in the code which involves active variables. The derivatives are constructed according to simple rules. Exercises 3 and 4 include identification of active variables.

Many statements in the model code will contain nonlinear operations on active variables. In these cases the derivative statements must also contain model variables and so the value of the derivative will depend on the value of some model variables. Those variables are called required variables. In tangent linear code values for those variables can be easily provided (e.g. by inserting all tangent linear statements in the model code, locating each tangent linear statement before the statement it corresponds to). In adjoint code, providing required variables efficiently is one of the major challenges. Exercises 3 and 4 include identification of required variables.

The TAMC system consists of two parts: a utility that is to be installed on your local computer and the actual software that does automatic differentiation, which is installed on a remote machine. The source code of the utility is provided with the code for the exercises. TAMC is invoked by the utility through a UNIX shell script **tamc**, which uses a UNIX remote shell to communicate with the remote machine. Through command line options, the user defines the function to be differentiated by naming a FORTRAN subroutine, its input and output variables as well as the files containing the code to be differentiated. Section 3.3 gives two examples for invoking TAMC.

To execute the generated derivative code, it has to be embedded in a main program. For different applications of derivative code, a software package (TAMLINK) comprising a number of main programs is included in the utility. The code is linked by a second UNIX shell script named `tamlink`. `tamlink` expects a set of subroutines with particular names and interfaces. Section 3.3 gives a number of examples for invoking `tamlink`.

## *3.3. Sensitivities of Box Model*

In this section we illustrate the use of TAMC to calculate the sensitivity of concentrations to various parameters in the simple two-box model of section 2. The material provides explanation and hints for Exercises 3 and 4. Although the model is extremely simple, from the viewpoint of automatic differentiation it contains most of the important features found in complex transport models.

The code of `Boxmod` is listed in Figure 1. Using prescribed hemispheric estimates of the sources of methyl chloroform, which have been provided by *Prinn et al.* [1992], and an atmospheric lifetime of 4.7 years as calculated by *Houweling et al.* [1998], `Boxmod` simulates the hemispheric concentrations of methyl chloroform. `Boxmod` has a (single) transport parameter, namely the rate of interhemispheric mixing, `mixrate`. One of the optimization exercises is to tune this parameter using sensitivities computed by the adjoint of `Boxmod`. Unlike more sophisticated transport models, the transport in `Boxmod` has neither seasonality nor interannual variations. The change in the simulated concentration depends nonlinearly on the mixing rate and the inverse of the atmospheric lifetime (`invlif`) but linearly on the sources. The sensitivities (or response functions) that quantify these linearizations depend on the values of `mixrate` and `invlif`.

$\boxed{\text{Figure 1}}$

*3.3.1. Adjoint.* Exercise 3 can be solved by applying TAMC to generate code for computation of the sensitivity of the concentration in box 1 at the last time step with respect to the mixing rate, the inverse lifetime,

and the source components. Since the function to be differentiated has one output variable and many input variables, the reverse mode is most efficient [*Giering*, 1999], so we calculate the adjoint of `Boxmod`. To provide all necessary information about the function to be differentiated to TAMC we have constructed a subroutine `model` (see Figure 2), which has in its argument list the number of input variables, the vector of input variables `X`, and the output variable `FC`. (The related header file is listed in Figure 3.) TAMC is called with the options
**-module model -input X -output FC -reverse**.

Figure 2

Figure 3

A particularly important feature for adjoint code generation, which is typical for transport models, is that `Boxmod` overwrites the current concentration every time step. For applications like Exercise 3, in which either the mixing rate or the inverse lifetime are active variables, the concentration `c` is one of the required variables. There are further required variables (or parameters) such as `ny, ntpy, kt2pptv, mixrate`, and `invlif`, but those are not overwritten and, thus, easy to provide. By default TAMC generates a loop to recompute the required values of `c`, but TAMC also provides the alternative of storing the required values on a 'tape', i.e. in memory or in a file. In Figure 2 the storing feature is demonstrated, because recomputation would require a second loop within the adjoint of the main loop (see *Giering* [1999]) which, computationally, for a three dimensional model would be prohibitively expensive (see e.g. computational cost of the adjoint of TM2 in *Kaminski et al.* [1999]). Directives are used to make TAMC store and read the required variables by calling special library routines. First an `init` directive is needed to initialize the 'tape', and then a `store` directive is inserted right before the statement whose adjoint uses the required variable (see Figure 2). Before the adjoint code is executed, the required values are recomputed and stored, and during the execution of the adjoint code they are read (see below).

The adjoint of `Boxmod` generated by TAMC is listed in Figures 4–6. Figure 4 contains the declaration of required and adjoint variables and the initialization of the local adjoint variables `adc` and `adcnew` to 0. The global adjoint variable `adsrc` is initialized by the subroutine `adzero`. The global adjoint variable `adfc` has to be initialized to 1 before calling `admodel`. The initialization strategy can be understood as a consequence of the concept of locality. (For details see definition of locality in *Giering and Kaminski* [1998].) The current values of adjoint variables reflect the derivative of `fc` with respect to the corresponding variable in the forward code. 'Cur-

Figures 4–6

rent' refers to the place in the code of `Boxmod` where the statement to which the adjoint statement corresponds is located. Since `admodel` runs in reverse mode, the end of `model` is the location to which the initialization part of `admodel` corresponds. And at the end of `model`, only a change in the variable `fc` could change the function value, all others don't have any impact anymore. Hence their adjoints must be 0. The adjoint computations part in `admodel` (see Figure 5) updates the values of the adjoint variables by executing the adjoints of the statements in `model` in reverse order. At the end of `admodel` the adjoint of `x` holds the derivative of `fc` with respect to `x`, and all other adjoint variables hold zeros.

As discussed above, the correct values of most of the required variables are easy to provide. For instance the values of `mixrate` and `invlif` can be recovered from `x` at the beginning of the adjoint computations part. The computation of `fc` is carried out during computation of the value of the function (which TAMC adds by default before the adjoint computations). `fc` is needed e.g. for use in optimization procedures. (Generation of code for the value of the function can be avoided by using the directive **-pure**). Reading and writing and the necessary bookkeeping are organized by the subroutines `adstore` and `adresto`.

To actually compute the sensitivity needed to answer Exercise 3, `admodel` has to be executed using the correct values of the input variables, i.e. sources, inverse lifetime, and mixing rate. The subroutine can be executed easily after linking the appropriate main program for computing sensitivities by the `tamlink` script. This main program requires that, besides `model`, three subroutines be provided:

- a subroutine `numbmod` defining the number of input variables (see Figure 7),
- a subroutine `initmod` doing all necessary initialization, in particular setting the values of the input variables (see Figure 8),
- and a subroutine `postmod` doing the post processing (see Figure 9).

Figure 7

Figure 8

Figure 9

To link the main program for computation of the sensitivity in reverse mode, `tamlink` should be invoked with the command line option **-adjoint**.

*3.3.2. Tangent linear model.* Exercise 4 can be solved by applying TAMC to generate code for computation of the sensitivity of the concentration in both boxes and all years with respect to the inverse lifetime and the mixing ratio. Since the function to be differentiated has two input variables and many output variables, the forward mode is most efficient [*Giering*, 1999], i.e. we should construct the tangent linear model to `Boxmod`. We should stress again that the choice of forward or reverse mode is a computational one. To provide all necessary information about the function to be differentiated to TAMC

- a subroutine `func` (see Figure 10), which has in its argument list the number of input variables, the vector of input variables `X`, the number of output variables, and the output variable `Y` has been constructed. Note this is different from the reverse mode in which only one output variable was used.

Figure 10

- TAMC is called with the options
  **-module func -input X -output Y -ldg**
  **-jacobian 20 -forward**
  (The **-jacobian** option sets the number of output variables and has a default of 1. The option **-ldg** extends the parameter list of `g_func`).

The tangent linear code of `Boxmod` is listed in Figures 11 and 12. The tangent linear variables are marked by the prefix `g_`. Recall that they hold the derivative of the (active) variable they correspond to with respect to the input variable `x`. Figure 4 contains the declaration of forward code and tangent linear variables. Note that unlike adjoint variables, tangent linear variables do not have to be initialized, because, except for the tangent linear variables corresponding to the input variable `x`, they are not referenced before being set by at least one tangent linear statement in `g_func`. Each statement from the function evaluation is preceded by its tangent linear statement. Thanks to this scheme, correct values of all required variables are naturally provided. It might seem from this that the tangent linear or forward mode must be more efficient than the adjoint or reverse mode. However the extra pass through the function needed by the adjoint model might be a very minor cost compared to the calculation of unnecessary derivatives.

Figures 11 and 12

One advantage of tangent linear over adjoint code is that it is much more readable, with statements occurring in natural rather than reverse order. While the function part of `g_` computes `y`, the derivative part computes `g_y`.

As with the adjoint code, to actually compute the sensitivity needed to answer Exercise 4, `g_func` has to be executed using the correct values of the input variables, i.e. sources, inverse lifetime, and mixing rate. The subroutine can be executed after linking the appropriate main program for computing sensitivities by TAMLINK. This main program requires that, besides `func`, three subroutines be provided:

- a subroutine `setfunc` defining the number of input and output variables (see Figure 13),

Figure 13

- a subroutine `initfunc` doing all necessary initialization, in particular setting the values of the input variables (see Figure 14),

Figure 14

- and a subroutine `postfunc` for post processing (see Figure 15).

Figure 15

Note that unlike the scalar valued function of the previous example, we are now differentiating a vector-valued function. This is the reason for the differences in the argument lists of the subroutines. To link the main program for computation of the sensitivity in forward mode, `tamlink` should be invoked with the command line option **-forward**.

## 4. Exercises

### *4.1. Adjoint Methods Exercises*

This section demonstrates the use of the tangent linear and adjoint compiler (TAMC) for both sensitivity and optimization calculations. The computer-based examples all use the two-box model of atmospheric transport introduced in section 2. The discrete equations are given by

$$c_1(k+1) = c_1(k) + \Delta t \left( s_1(k) - \frac{c_1(k) - c_2(k)}{\tau} - \frac{c_1(k)}{\lambda} \right) \tag{9}$$

$$c_2(k+1) = c_2(k) + \Delta t \left( s_2(k) - \frac{c_2(k) - c_1(k)}{\tau} - \frac{c_2(k)}{\lambda} \right) \tag{10}$$

where the subscripts refer to the box and the $k$ to the time level. $\Delta t$ is the timestep of the model, $\tau$ the interhemispheric exchange time and $\lambda$ the chemical lifetime. Note that numerical stability is only guaranteed for $\Delta t < \tau$ and $\Delta t < \lambda$. This is the well-known CFL criterion (see *Press et al.* [1986, p627]). Also note that the code uses $\kappa = \frac{M}{2\tau}$ and $\frac{1}{\lambda}$ for convenience. The supplied basic model integrates this set of equations forward from some initial condition according to a set of prescribed sources and model parameters.

*4.1.1. Sensitivity.* Code for these exercises can be found in the directory `/adjoint/sens` on the CD-ROM.

1 Imagine you have a very simple model which takes 5 inputs and produces 1 output. In between the input and output, the model produces sets of intermediate results, the first step produces 2, the second 3, the third 3 and finally the fourth produces the 1 output. Using the convention `<number of`

`rows>` $\times$ `<number of columns>`, their respective Jacobians are 2×5, 3×2, 3×3, and 1×3 matrices.

- a Make a drawing of the matrix product and of its evaluation in forward and reverse modes. What physical quantities are represented by the entries in the matrices?
- b What is the largest matrix needed to hold an intermediate result in forward and reverse modes?
- c How many additions and multiplications are needed for the evaluation of the product in forward and reverse modes?

2 a Compile and run `boxmod` (file `boxmod_0.F`) in the `adjoint/sens` directory on the CD-ROM. For the case of zero sources and either no chemical loss or no interhemispheric mixing, verify that the numerical solution approximates the exact solution.

- b Set $\tau$ to 0.1 y. By considering the time evolution of global concentration, compare the chemical lifetime as actually simulated to the input value.
- c Calculate the response of concentrations to a unit source in one hemisphere in the first year. This is the so-called Green's function [*Enting*, 1999, equation (2)]. The same result can be obtained with TAMC as we shall see below.

3 Imagine you perform a unit change of the mixing rate, the inverse lifetime, or any of the source components. Which change, to first order, has the largest impact on the simulated concentration in box 1 at the end of year 10 (this is an application for reverse mode)?
Which are the active variables?
Which are the required variables?
Detailed instructions (for those not familiar with TAMC):

- a Transform the code in `boxmod_0.F` to a Fortran subroutine that computes the concentration at the end of year 10 as a function of all sources, mixing time and inverse lifetime in the form required by TAMC and TAMLINK.
- b Invoke **tamlink -cost** to check whether this modified code runs and computes the correct value for the concentration in year 10.
- c Invoke TAMC to compute the derivative of this function.
- d Check this derivative against finite differences by using **tamlink -check**.
- e Run the adjoint code by using **tamlink -adjoint**.

4 In which box and for which year will the concentration be most strongly affected by a unit change of the inverse lifetime or the mixing rate? This is an application for forward mode.
Which are the active variables?
Which are the required variables?
Why are the active and required variables different from the previous exercise?
Detailed instructions (for those not familiar with TAMC):

   a Transform the code in `boxmod_0.F` to a subroutine that defines the concentration in all years and both boxes as a function of mixing time and inverse lifetime.
   b Invoke TAMC to compute the derivative of this function.
   c Check this derivative against the derivative computed in reverse mode by using the command **tamlink -compare**.

*4.1.2. Optimization by adjoint method.* In these exercises we use a version of the two-box model, data on sources and emissions of the gas methyl chloroform ($CH_3CCl_3$) from *Prinn et al.* [1992], the capabilities of TAMC to provide derivatives, and one of the powerful iterative minimisation algorithms that rely on availability of derivatives. We combine these ingredients to solve for the chemical lifetime and interhemispheric mixing rate of the gas. The forward version of the box model and associated data files is found in the directory `/adjoint/optim` on the CD-ROM. As provided, the code performs the necessary initialization for the adjoint calculations but the model does not compute a cost function.

5 Recast the model to calculate the mean squared mismatch between observed and modelled concentrations as a function of chemical lifetime and mixing time assuming the sources are correct. Now use the optimization routines provided with TAMC to find the optimal values for $\tau$ and $\lambda$. You can do this simply by entering **make optim** in the directory. Plot the observed and modelled concentrations to assess how well such a simple model performs. What are potential causes of any mismatch?

6 Exercise 5 assumes the sources are perfectly well-known. We can include the impact of source uncertainties by expanding the cost or mismatch function to

$$C = \sum (c_{mod} - c_{obs})^2 / e_c^2 + \sum (s_{mod} - s_{obs})^2 / e_s^2 \tag{11}$$

where $c$ and $s$ refer to concentrations and sources, "obs" and "mod" refer to observed and modelled and the $e$ in the denominator refers to errors. Recast the model to include the sources as input parameters and to use this cost function. This example should collapse to Exercise 5 for very small $e_s$, check this. What is the impact on lifetime and mixing time calculations for a 10% standard deviation in sources? What happens to the concentration mismatch from Exercise 5? We can further expand the cost function to include a prior estimate for the lifetime and mixing time.

### *4.2. Green's Function Methods*

These exercises illustrate the methods described in *Enting* [1999] using the two-box model in section 2. The Green's function approach is restricted to problems in which concentrations can be approximated as linear combinations of the sources and sinks. For such problems estimates of uncertainties are more efficiently derived with the Green's function approach than the optimization methods in section 4.1.2. Green's function methods are not well-suited to nonlinear problems like Exercise 5 in which we solved for the chemical loss rate. Such problems require iterative optimization methods. Hence in this exercise we fix the inverse lifetime determined in Exercise 5 and solve only for sources. As we can see from the right-hand side of equation (A2b) of *Enting* [1999] we need to specify both prior estimates and uncertainties for both sources and concentrations.

The following exercises use a trivial case with constant sources so that answers can be checked analytically. This simple case also makes it possible to compare with the Kalman filter case (section 4.3.1.1) in which we are estimating a constant source. The concentrations consist of a uniform trend in both hemispheres and a constant offset between them. The value of $\tau$ in this example is 1.0 and we also divide by atmospheric mass so the value of $M$ in equation (1) is also 1.0. Code and data for the examples can be found in the `Fortran` and `IDL` subdirectories of `/greens` on the CD-ROM.

7 Using the data from the CD-ROM (`trivial.dat`) and equation (5a) and equation (6) calculate the steady-state sources analytically.

8 Check the results with the supplied code. In the code, there is no gradient assumed between the hemispheres before the start of the calculation. How does this affect the sources early in the inversion? Why do the predicted uncertainties of sources change throughout the time series?

9 The inversion returns the estimated uncertainty in the form of a covariance matrix (see *Enting* [1999, equation (B5)]). In the supplied code we have printed the square root of the diagonal elements of the covariance i.e. the uncertainties of the individual sources. We can also consider the variance of groups of sources such as the global source or long-term mean hemispheric source. The variance of a sum of random variables can be calculated as the sum over all elements in their covariance matrix. To compute the variance of the *mean* of the same variables the above sum should be divided by $n^2$ where $n$ is the number of variables. Use this relation to calculate the uncertainty in the *time-averaged* (over the period of the inversion) northern hemisphere, southern hemisphere and total source. How does this uncertainty change with the length of the inversion and why?

10 Notice in Exercise 9 that the uncertainty for the average *total* source is less than might be expected from summing that from two hemispheres. This suggests compensating errors in source estimates. Confirm this by calculating the off-diagonal term in the covariance matrix of long-term average sources. From the governing equations, why might this occur? In general, for two variables with equal standard deviations, what correlation is required for their sum to have a lower standard deviation than either variable alone?

11 Until now we have used prior sources and concentrations which were consistent. Hence the calculated posterior estimates were the same as the priors. Using the original concentrations and uncertainties, set all prior sources to zero. Noting that the predicted sources are a weighted sum of the prior sources and sources consistent with the data, vary the prior source uncertainty and observe the change in calculated sources.

12 In general we require some consistency between prior and predicted estimates and the uncertainties we specify. If the data mismatch or the changes between prior and predicted sources are much larger than our specified uncertainties we should have more caution about our set-up. This increased caution is implemented by increasing the uncertainties we use on initial sources and data. We quantify the mismatch as the minimum value of the misfit function [*Enting*, 1999, equation (A1)]. See *Tarantola*, [1987, p211] for the derivation of this result. This misfit function follows a $\chi^2$ distribution with degrees of

freedom given by the number of data $n_d$. The misfit is often termed the $\chi^2$ (chi-squared) value for a particular inversion. The value $\chi^2/n_d$ should be near 1. If it is too large (our misfit is too large) then we have been too confident either in our measurements or prior source estimates. We should increase uncertainties accordingly. Alternatively, a small $\chi^2$ value suggests we have been too conservative. A discussion of what constitutes too large or small is beyond our scope here. In the language of statistics the question is rephrased as "significantly different from 1". Most statistical software packages will provide calculations of the significance levels for $\chi^2$ for a given $n_d$. Tables are presented by *Abramowitz and Stegun*, [1970].

Calculate the $\chi^2$ for the case with the original uncertainties (0.5 for sources and data) and zero prior sources. Adjust the uncertainties by scaling source and data uncertainties equally until the value is near 1. How does this impact the uncertainty of the average source? Note that an equal scaling of data and source uncertainties does not change the final estimate, why not? Now adjust the source uncertainties only (like Exercise 11) until $\chi^2$ is near 1.

13 We have considered uncertainties in sources and data so far. The relationship between sources and data (embodied in the transport model) is also uncertain, as demonstrated by *Law et al.* [1996]. *Enting* [1999, section 3] discusses the usual approach to dealing with such uncertainty, namely increasing the uncertainty in the data. As an example of the problem, we can modify the artificial Jacobian. As provided, the Jacobian has no mixing in the first year then complete mixing. Modify this so that $\frac{1}{3}$ of the extra injected tracer from a source is mixed in the first year, then complete mixing occurs after that. We can think of this as roughly changing the mixing time from 1 year to $\frac{2}{3}$ years. To implement this, set the Jacobian in the same hemisphere as the source to $\frac{5}{6}$ and in the other hemisphere to $\frac{1}{6}$. What is the impact on estimated sources. Why does the uncertainty rise?

## *4.3. Kalman Filtering*

This section presents an extensive set of exercises and examples concerning Kalman filtering and related variants. For a derivation of the Kalman filter equations see *Prinn* [1999] or *Todling* [1999, section 4]. The computer-based exercises treat three relatively simple cases

1. The two-box model of transport used in sections 4.1.2 and 4.2.

2. A damped harmonic oscillator.
3. A one-dimensional advection problem.

In the third of these examples some more computationally tractable approximations to the optimal Kalman filter are introduced. The analytical examples cement several of the key concepts in using the Kalman filter and show methods by which important inputs to the filter may be derived, particularly correlation functions.

*4.3.1. Box Model.* This section presents an application of the Kalman filter and smoother to the simple two-box model of transport used in sections 4.1.2 and 4.2. In general we solve for sources given data on concentrations and an assumed interhemispheric exchange time and chemical lifetime. The cases are the same as sections 4.1.2 and 4.2, i.e. a trivial case with constant sources and a more realistic case using the gas methyl chloroform.

4.3.1.1. Trivial case: This case has been set up to compare with the case in section 4.2. The true solution, with constant sources and hence constant growth rate and fixed interhemispheric difference is the same. We start with the simplest case, refining estimates of a constant source. This is similar to the case described in *Prinn* [1999, section 6] except that they solved for the chemical lifetime rather than sources. This case is directly comparable with the Green's function Exercises 7–13. As the final step we include some of the complications normally included in a Kalman filtering method such as evolution error. These lead on to the next exercise in which we treat a more realistic example, although still with a simple model.

There is one important difference between the Kalman filter examples presented in *Prinn* [1999] and in this section. *Prinn* [1999] used the transport model to compute the measurement operator **H** which related the unknowns in the state vector (chemical lifetimes in that case) to the observed concentrations. Here, we use the transport model as part of the evolution operator **M** for the state vector itself. Our state vector contains sources and concentrations for each hemisphere and the Kalman filter solves equation (4) for the evolution of concentrations. We discuss the evolution of the sources in the exercises below.

14 Run the code as supplied in either the `Fortran` or `IDL` subdirectories of the `/kalman/box/trivial` directory and compare the final average source and uncertainty of the source with that for the equivalent Green's function case. Compare the evolution of the uncertainty in the two cases, why is it different?

15 Let us now investigate more closely the similarities and differences between the Kalman filter and Green's function cases. First make them as comparable as possible. We need to make the Green's function code obey the exact solution. Even with correct sources this did not happen in section 4.2 because there was no initial gradient assumed between hemispheres so the inversion needed to spin up. We can trick the code into an exact match by adding 0.5 to all concentrations in `trivial.dat` in the `green` subdirectory. This makes the total tracer mass in year 1 consistent with the prior sources for that year. Now both the Kalman filter and Green's function example should match the exact solution throughout although their uncertainties will be different as we have just seen. Confirm that both codes are behaving correctly. Now modify the Kalman filter code to set the initial state to include a source of 1 for one hemisphere and 0 for the other. Similarly modify the `trivial.src` file in the Green's function directory to set *all* initial sources to 1 for one hemisphere and 0 for the other. Run both cases and compare the evolution of the sources, can you explain the behaviour? How does the *average* source for the 10-year Green's function run compare with the *final* source from the Kalman filter run? Can you explain this? Hint: consider the constraint towards the initial *average* source in the Green's function case.

16 In the Kalman filter example, effectively delete several of the data points by greatly inflating their observational error. For example, for the years 1981–1985, change the observational error for both hemispheres from 0.5 to 10. What happens to both source and concentration uncertainties? Referring to the discussion by [*Prinn*, 1999, section 5], on intuitive concepts of Kalman filter behaviour, can you explain the different behaviour? Starting from the usual incorrect initial state of $(1, 0)$, what is the impact on the final predicted source? Now repeat the experiment setting data uncertainties to $(0.5, 1.5)$ throughout and starting from sources of $(1, 0)$. Students interested in pursuing the comparison with Green's function methods should repeat these data deletion experiments in that case.

17 Now let us consider adding stochastic noise to the problem. In our example we have a state containing sources and concentrations. The concentrations evolve according to the two-box model. There is no explicit evolution of the sources; we assume persis-

tence i.e. the evolution matrix $\mathbf{M} = 1$. The evolution of both concentrations and sources is subject to error and we can include this error in the stochastic noise covariance matrix $\mathbf{Q}$ defined in *Todling* [1999, equation (64)]. With correct initial sources, include stochastic noise with $\sigma = 0.5$ for just the sources (the last two elements in the state). How does the uncertainty compare with the case with $\sigma = 0$? Now try a range of values for the stochastic covariance and a range of observational uncertainties. What general relationship do you notice between the predicted uncertainty and the stochastic covariance, explain?

4.3.1.2. Methyl Chloroform Case: In this exercise we revisit the problem of estimating sources for methyl chloroform ($CH_3CCl_3$). Unlike the Kalman filter example described in *Prinn* [1999] or the optimization Exercise 5 we will estimate sources rather than chemical lifetime and mixing time. We use the same two-box model as section 4.1.2 and 4.2 with mixing time and lifetime taken from the solution of Exercise 5. The concentration data are taken from *Prinn et al.* [1995]. We use the source data from that paper for comparison only.

The code is supplied in the `fortran` or `IDL` subdirectories of the `kalman/box/ch3ccl3` directory on the CD-ROM. As supplied there is no stochastic covariance.

18 Run the supplied code. By considering the evolution of the source uncertainty can you explain the mismatch to the concentrations from the late 1980s onwards?

19 How should you interpret the mismatch with sources in this case?

20 As with the Green's function examples (Exercises 11 and 12) we can quantify the mismatch between our evolution model and observed data in a variable with the $\chi^2$ distribution and expressed as

$$\chi^2 = (\mathbf{Hx} - \mathbf{z})^T(\mathbf{HPH}^T + \mathbf{R})^{-1}(\mathbf{Hx} - \mathbf{z}) \quad (12)$$

This expression is similar to *Enting* [1999, equation (A1)]. An important difference is that in the Green's function case we specify an initial estimate for our sources while in the Kalman filter case we specify a *model* for how these sources will evolve. This can seem a little confusing at the moment since our source evolution model is completely trivial but later examples show cases with more sophisticated evolution models for the state. Turn on the $\chi^2$ calculation in the code which should enable you to quantify the conclusions from Exercise 19.

21 Unlike the trivial case it is clear we need some stochastic noise here to allow the sources to vary. It can play the same role as the prior uncertainties in the Green's function case and we can choose our value so that $\chi^2$ remains reasonable. Add stochastic covariance with $\sigma = 5$ Tg y$^{-1}$. What is the maximum value of $\chi^2$? The average? Does this suggest 5 Tg y$^{-1}$ is a reasonable choice?

22 As mentioned in *Prinn* [1999, section 5, equations (24) and (25)] and *Todling* [1999, section 4] an extension to the Kalman filter, the Kalman smoother, enables *all* data to influence predicted sources. In the filter version only previous and current concentrations impact source estimates. It makes sense to allow future concentrations to help constrain sources. This happens naturally in the Green's function approach but not in the regular Kalman filter, a deficiency addressed by the Kalman smoother. By default the Kalman smoother part of the code is turned off. With no stochastic covariance, activate it. To what do you attribute the apparently wild behaviour? Now add the same $\mathbf{Q}$ as before i.e. $\sigma = 5$ Tg y$^{-1}$ for sources. What is the impact of the smoother on calculated concentrations, sources and uncertainties?

23 Finally we should demonstrate that any inversion is only as good as the model which underlies it. What happens to predicted sources and concentrations if you double the atmospheric lifetime? What about doubling the mixing time?

*4.3.2. Oscillator example.* In this section we consider the application of the Kalman filter to a simple dynamical system. The example is partially based on *Lewis*, [1986, Example 2.5-2]; see also the mass–spring oscillator example in *Wunsch* [1996, section 6.2.1]. Students should also be familiar with the terminology introduced in *Todling* [1999].

Consider the damped harmonic oscillator which has the governing equation

$$\ddot{y}(t) + 2\alpha\dot{y} + \omega^2 = 0 \tag{13}$$

We can write a discrete version as

$$\mathbf{x}_k \begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} = \begin{pmatrix} 1 & T \\ -\omega^2 T & 1 - 2\alpha T \end{pmatrix} \begin{pmatrix} x_1(k-1) \\ x_2(k-1) \end{pmatrix} + \begin{pmatrix} \eta_1(k-1) \\ \eta_2(k-1) \end{pmatrix} \tag{14}$$

where the $\eta$ represents noise. We make observations $z$ on the state $\mathbf{x}$ via some linear measuring process represented as

$$z(k) = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} + \epsilon(k) \qquad (15)$$

i.e. we only observe one of the state variables.

We assume the state $\mathbf{x}$ is distributed according to a bivariate normal distribution so that $\mathbf{x}(k) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$, and that the observational errors are also normally distributed so that $\epsilon(k) \sim \mathcal{N}(0, r/T)$ and both are uncorrelated from each other at all times. We take the covariance $\mathbf{Q}$ to be given by

$$\mathbf{Q} = \begin{pmatrix} 0 & 0 \\ 0 & T \end{pmatrix} \qquad (16)$$

and we assume the initial state to be normally distributed as $\mathcal{N}(0, \mathbf{P}_0^a)$. For the choice of parameters: $\omega = 0$, $\alpha = -0.1$, $r = 0.02$ and $T = 0.02$, address the following questions:

24 Is the dynamical system stable or unstable?

25 Using `matlab`, simulate the stochastic dynamical system from $k = 0$ to $k = 500$, starting from $\mathbf{x}_0 = \begin{pmatrix} 0.1 \\ 0.2 \end{pmatrix}$. Plot the state $\mathbf{x}_k$ against $k$.

26 Using the linear Kalman filter, simulate the evolution of the error covariance matrix, starting from the initial condition $\mathbf{P}_0^a = \mathbf{I}$, where $\mathbf{I}$ is the $2 \times 2$ identity matrix. Plot the analysis error variance, in both variables, for the same time interval as in Exercise 24.

27 Is the filter stable or unstable? Explain.

28 Are your answers to questions 24 and 27 incompatible? Explain.

29 Plot the true state evolution together with the analysis estimate for both variables and for the time interval in Exercise 25.

Hint: Remember that your initial estimate should be sampled from the initial state statistics. When the initial is distributed as $\mathcal{N}(0, \mathbf{P}_0^a)$, a sample can be obtained from

$$\mathbf{x}_0^a = \mathbf{x}_0 + \mathbf{V} * \mathbf{D}^{1/2} * \boxed{\text{randn(:)}} \qquad (17)$$

which is the multivariate version of a sample from a normal distribution. The matrices $\mathbf{V}$ and $\mathbf{D}$ arise from the eigendecomposition of the initial error covariance matrix: written symbolically as

$$[\mathbf{V}, \mathbf{D}] = \boxed{\text{eig}(\mathbf{P}_0^a)} \qquad (18)$$

Let us now study the behavior of two suboptimal filters. Before starting, however, replace the analysis error covariance equation in your `matlab` program by the Joseph formula [*Todling*, 1999, equation (80)] if you are not already using it. We mentioned in *Todling* [1999, section 4] that the Joseph formula is valid for any gain matrix $\tilde{\mathbf{K}}_k$, thus we can use it to evaluate the performance of suboptimal filters.

30 Assuming the calculation of the forecast error covariance is computationally too costly for the present problem, we want to construct a suboptimal filter that somehow replaces the calculation of $\mathbf{P}_k^f$ by a simpler equation. Let us replace the equation for $\mathbf{P}_k^f$ by the simple expression $\mathbf{P}_k^f = \mathbf{I}$. With this choice of forecast error covariance, it is simple to see that the gain matrix becomes

$$\begin{aligned} \tilde{\mathbf{K}}_k &= \mathbf{H}^T(\mathbf{H}\mathbf{H}^T + r/T)^{-1} \\ &= \tfrac{1}{1+r/T}\mathbf{H}^T \end{aligned} \tag{19}$$

where we used explicitly that $\mathbf{H} = (1, 0)$ for the system under consideration. Keeping the equation for $\mathbf{P}_k^f$, in your `matlab` code as dictated by the Kalman filter, replace the expression for the optimal gain by the one given above. This turns the state estimate into a suboptimal estimate. Also, since you have kept the original expression for the forecast error covariance evolution, and you are using the Joseph formula for the analysis error covariance, these two quantities provide filter performance information about suboptimal choices of gains. With the "approximate" gain matrix above, is the resulting filter stable or unstable? Explain. If this is not a successful choice of gain matrix, can you explain why that is?

31 Let us now build another suboptimal filter that replaces the gain by the asymptotic gain obtained from the optimal run in Exercise 25. To obtain the optimal asymptotic gain, you need to run the experiment in Exercise 25 again, output the gain matrix at the last time step from that run, and use it as a suboptimal choice for the gain matrix in this item. You should actually make sure that the gain has stabilized by looking at its value for a few time steps before the last time step, and verifying that these values are indeed the same. Now rerun Exercise 30 but using the asymptotic gain for the suboptimal gains at all time steps. Is the resulting filter stable or unstable? (Note: This choice of gain corresponds to using the so-called Wiener filter.)

*4.3.3. Linear Advection Equation.* In this section we consider the application of the Kalman filter to a simple transport problem. Consider the one–dimensional advection equation

$$\frac{\partial u}{\partial t} + U\frac{\partial u}{\partial x} = 0 \tag{20}$$

where $U$ = const represents the advection speed, applied to a periodic domain defined by the interval $[-2, 2]$ over the line. Take for the initial condition a "rectangular" wave of the form

$$u(x, t = 0) = \begin{cases} 1, & \text{for } -1 \leq x \leq 0 \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

Using an up–wind finite difference scheme we can write an approximate solution to the advection equation as

$$v_j = Cv_{j-1} + (1 - C)v_j \tag{22}$$

where $v_j$ represents the numeric solution for $u(x = j\Delta x)$ with $\Delta x$ as the spatial interval, and where $C = U\Delta t/\Delta x$ is the Courant number, with time step $\Delta t$ [see *Press et al.*, 1996].

32 *Simulation experiments:* Using the parameters in Table 1, obtain plots of the state evolution at the initial and final times for an integration taken from $T_0 = 0$ to $T_{\text{final}} = 1$ using the following Courant numbers: $C = 1$, $C = 0.95$, and $C = 0.90$. Explain the difference in the results.

$\boxed{\text{Table 1}}$

Let us now slowly build the components for a Kalman filter assimilation experiment. We assume that all error statistics necessary for the filter are normal and uncorrelated. We need to construct error covariance matrices for all stochastic processes involved in the problem. For simplicity, we take the perfect model assumption, so that we do not have to worry about the model error $\boldsymbol{\eta}_k$ and its error covariance $\mathbf{Q}_k$, i.e., $\boldsymbol{\eta}_k = 0$ at all times. Moreover, we assume there are no correlations among observational errors, so that the observation error covariance matrix $\mathbf{R}_k$ is diagonal. Furthermore, this matrix is assumed to be time independent with diagonal elements specified for each experiment. The only error covariance left to specify is that of the initial estimate, $\mathbf{P}_0^a$. Constructing valid spatial error covariance functions is somewhat difficult. Here, we use a `matlab` function $\boxed{\text{gcorr}}$ provided with this exercise to construct an appropriate error correlation field that can be used to generate the covariance matrix. Use the 'help' of this function to see its usage. For three choices of the decorrelation length parameter ($L_d = 0.5, 1.0$ and $1.5$), answer the following questions

33 Is the matrix you constructed an acceptable correlation matrix?

34 Plot the two–point correlation function at two distinct arbitrary locations. Comment on what you see.

35 Make a contour plot of the correlation matrix. What you will see corresponds to the "shape" of a homogeneous and isotropic correlation matrix.

Now we have all the necessary components, write a `matlab` program with the Kalman filter equations for the state estimates $\mathbf{x}_k^f$ and $\mathbf{x}_k^a$, and their corresponding covariances $\mathbf{P}_k^f$ and $\mathbf{P}_k^a$, respectively. We will now use this program in a series of so-called simulated observation experiments in which we take the actual solution of the advection equation at specific places and times then add Gaussian noise to them. This can be done in `matlab` using the random number generator $\boxed{\text{randn}}$ for normally distributed variables. In particular, we investigate the ability of the Kalman filter to reconstruct the true solution starting with an initial guess of zero, i.e., $\mathbf{x}_0^a = \mathbf{0}$. Such experiments are usually termed 'wave generation' in the literature.

In what follows, you are asked to make plots for the true state and its estimate at the final time of the assimilation as well as plots of the time evolution of the domain–averaged forecast and analysis error standard deviation.

The first case we consider is one for which the observations are all located over the left–half of the domain.

36 Following the choice of parameters in Table 2, obtain the output for the true state and its estimate at the final time of the assimilation experiment. $\boxed{\text{Table 2}}$

37 What happens if the observation error level is increased to 0.1?

38 What happens if in Exercise 37, the assimilation period is 5 time units?

39 Comment on the results you just obtained.

40 You can try lots of other combinations and possibilities with this little program. Changing at least one of the parameters in Table 2, here are a couple of other possible scenarios to investigate:

a Take observations at every grid point.

b Change the Courant number to make the dynamics more (numerically) dissipative, i.e. reduce the timestep.

What happens to the filter results in these cases?

*4.3.4. Analytic Exercises.* The following exercises are based on the material in *Ménard* [1999]. Students should familiarize themselves with that material before attempting the exercises. The exercises are of more general application than Kalman filtering but the experience gained in manipulating the statistical and algebraic quantities discussed here are useful in Kalman filtering.

41 Show that the analysis error variance can be obtained as

$$P^a(\mathbf{r}_i, \mathbf{r}_i) = P^f(\mathbf{r}_i, \mathbf{r}_i) - \mathbf{p}_i^T \mathbf{\Gamma}^{-1} \mathbf{p}_i \qquad (23)$$

where

$$\mathbf{p}_i = \left[P^f(\mathbf{r}_1^o, \mathbf{r}_i), P^f(\mathbf{r}_2^o, \mathbf{r}_i), \ldots, P^f(\mathbf{r}_p^o, \mathbf{r}_i)\right]^T \qquad (24)$$

is the forecast error covariance between all observation locations $\{\mathbf{r}_1^o, \ldots, \mathbf{r}_p^o\}$ and the grid point $\mathbf{r}_i$ (see *Ménard* [1999, section 3]).

Assume that the observational error and forecast error are uncorrelated, as usual in Kalman filtering;
(a) First show that the analysis error covariance can be written as,

$$\mathbf{P}^a = (\mathbf{I} - \mathbf{KH})\,\mathbf{P}^f\,(\mathbf{I} - \mathbf{KH})^T + \mathbf{KRK}^T \qquad (25)$$

where $\mathbf{K}$ is the Kalman gain matrix.
(b) Second, using equation (10) from *Ménard* [1999] for the Kalman gain show that the above expression for analysis error covariance reduces to

$$\mathbf{P}^a = (\mathbf{I} - \mathbf{KH})\,\mathbf{P}^f \qquad (26)$$

Then get the desired result.

42 Let $B(\mathbf{r}, \mathbf{r}') = B_0(\|\mathbf{r} - \mathbf{r}'\|)$ and $C(\mathbf{r}, \mathbf{r}') = C_0(\|\mathbf{r} - \mathbf{r}'\|)$ be homogeneous and isotropic correlation functions that are twice differentiable. Defining the correlation length-scale by

$$L_B^2 = \frac{1}{-\left(\frac{d^2 B_0}{d\rho^2}\right)_{\rho=0}} \qquad (27)$$

where $\rho = \|\mathbf{r} - \mathbf{r}'\|$, and similarly for $C$, show that the Hadamard product

$$D(\mathbf{r}, \mathbf{r}') = B(\mathbf{r}, \mathbf{r}')C(\mathbf{r}, \mathbf{r}') \qquad (28)$$

is a homogeneous isotropic correlation function with correlation length scale $L_D^2$ given by

$$\frac{1}{L_D^2} = \frac{1}{L_B^2} + \frac{1}{L_C^2}. \qquad (29)$$

(see *Ménard* [1999, section 4])

43 Show that the gradient of the log-likelihood function $L(\boldsymbol{\alpha})$ (i.e. *Ménard* [1999, equation (58)]) is given by

$$\frac{\partial L}{\partial \alpha_m} = \text{trace}\left[\left(\boldsymbol{\Gamma}^{-1} - \boldsymbol{\Gamma}^{-1}\boldsymbol{\nu}\boldsymbol{\nu}^T\boldsymbol{\Gamma}^{-1}\right)\frac{\partial \boldsymbol{\Gamma}}{\partial \alpha_m}\right] \quad (30)$$

(see *Ménard* [1999, section 5]).

First, use the cofactor expansion of the determinant of a matrix $\mathbf{A} = [a_{ij}]$,

$$\det(\mathbf{A}) = \sum_i a_{ij} A_{ij} \quad (31)$$

where $A_{ij}$ is the $(i,j)$ cofactor. Second, use the following formula for the inverse

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})}\left(\mathbf{A}^c\right)^T, \quad (32)$$

where $\mathbf{A}^c = [A_{ij}]$ is the matrix of cofactors, to evaluate

$$\frac{\partial}{\partial \alpha_m}\log\left(\det\left(\mathbf{A}\right)\right). \quad (33)$$

The differentiation of the inner-product of the log-likelihood function is obtained by using a formula for $\frac{\partial \mathbf{A}^{-1}}{\partial \alpha_m}$ obtained by differentiating $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$.

44 Show that the covariance model $\mathbf{P}$ [*Ménard*, 1999, equation (75)] is anisotropic on a sphere $S^2$, but its corresponding covariance $\mathbf{B}$ in logarithmic space is isotropic on $S^2$ with a variance approximately equal to $\gamma^2$ for $\gamma \ll 1$ (see *Ménard* [1999, section 7]).

## 5. Solutions

1 a Refer to Figure 16. If we consider each step in the model as a function, then the matrices represent the partial derivatives of the output of that step with respect to the input. Multiplying these matrices is really applying the chain rule. In forward mode, if we start evaluating this multiple matrix product from the right, each successive multiplication yields the partial derivatives of some intermediate result (and finally the output variables) with respect to the input variables. In reverse mode, if we start from the left, each multiplication yields the partial derivatives of the output variables with respect to some intermediate model result (and finally the input variables).

Figure 16

b In forward mode it is the $3\times5$ matrix after the first product has been evaluated, and in reverse mode it is the $1\times5$ matrix holding the final result.

c In forward mode, the number of multiplications is:

$$\begin{aligned} n_m &= 3 \times 2 \times 5 \\ &+ 3 \times 3 \times 5 \\ &+ 1 \times 3 \times 5 \\ &= 90, \end{aligned}$$

and the number of additions is:

$$\begin{aligned} n_a &= 3 \times (2-1) \times 5 \\ &+ 3 \times (3-1) \times 5 \\ &+ 1 \times (3-1) \times 5 \\ &= 55. \end{aligned}$$

In reverse mode, the number of multiplications is:

$$\begin{aligned} n_m &= 1 \times 3 \times 3 \\ &+ 1 \times 3 \times 2 \\ &+ 1 \times 2 \times 5 \\ &= 25, \end{aligned}$$

and the number of additions is:

$$\begin{aligned} n_a &= 1 \times (3-1) \times 3 \\ &+ 1 \times (3-1) \times 2 \\ &+ 1 \times (2-1) \times 5 \\ &= 15. \end{aligned}$$

2 a On a UNIX system you can enter **make run**. More generally you can compile and run `boxmod_0.F`.

b With a sufficiently rapid mixing time and no sources we can calculate the chemical lifetime from the time evolution of the global mean concentration. Using the concentration at year 5 suggests a lifetime of 4.67 y, very close to the input. Note that a much more rapid mixing rate, say 0.01 y, will violate the rather crude numerics of this example.

3 The active variables are `x, mixrate, invlif, src, c_new, c`, and `fc`.
The required variables are `c, ny, ntpy, kt2pptv, mixrate`, and `invlif`.

a Refer to Figures 2 and 7–9.

b Enter: ‘make cost’.

c Enter: ‘make boxmod_1_ad.f’.

d Enter: ‘make check’.

e Enter: 'make adjoint' or refer to Figure 17. Figure 17

A unit change of the inverse lifetime has the largest impact on the simulated concentration in box 1 at the end of year 10 ($-424$ pptv $\times$ year). Note that the answer depends on what units we choose for the input and output variables.

4 The active variables are `x, g_p_, mixrate, invlif, c_new, c`, and `y`.
The required variables are `c, g_p_, ny, ntpy, kt2pptv, mixrate`, and `invlif`.
The active and required variables are different from the ones in the previous exercise, because the function that is differentiated is different. Switching from reverse to forward mode does not change active or required variables.

a Refer to Figure 10.

b Enter: 'make forward' or refer to Figure 18. Figure 18

c Enter: 'make jacobian'.

The concentration in box 1 after year 10 is affected most strongly by a unit change of the inverse lifetime ($-424$ pptv $\times$ year). The concentrations in both boxes after year 10 are affected equally strongly by a unit change of the mixing rate ($\pm 10.4$ pptv $\times$ year). This time the answers do not depend on the units chosen.

5 The modified code is provided in the `solution` subdirectory. The values for the chemical lifetime and interhemispheric mixing time are 5.03 y and 0.80 y respectively. The sources may not be perfectly correct at all times. The model also assumes transport does not vary from one year to the next. Finally, the chemical lifetime is affected by the chemical state of the atmosphere which may also change over time.

6 The computed values of lifetime and mixing time change very little as we allow sources to be changed. The concentration mismatch improves since the optimization can now adjust sources as well as model parameters. We can see this as we loosen constraints on sources (change `rs` in `boxmod.h`). However we have now increased the number of degrees of freedom in the problem and consequently the uncertainty of any solution will also rise.

7 We see from `trivial.dat` that $\frac{\partial c_+}{\partial t} = 1$. In the supplied code, the mass of a *hemisphere* is 1 so that substituting into equation (5a) yields $s_+ = 2$. We

also see that $c_- = 1$ always. Substituting into equation (6) yields $s_- = 1$. Manipulating the definitions of $s_+$ and $s_-$ yields $s_1 = 1.5$ and $s_2 = 0.5$.

8 See the file `README` for instructions on running either the Fortran or IDL versions. The zero gradient between the boxes before the start of the calculation means the calculation is not in steady-state at the beginning. It takes a few years for the sources and concentrations to come into equilibrium as the initial state is forgotten. The uncertainties increase gradually through the calculation because of the amount of data constraining each source at a given time. A source in a given year is constrained by all data after that time. This implies a stronger constraint for early sources. The increase is small because of the peculiar Green's functions in this simple case which give a response of 1.0 in the same hemisphere and same year and 0.0 for the other hemisphere (i.e mixing) and 0.5 in both hemispheres after that (complete mixing). More physically realistic Green's functions would have a stronger link between current sources and future data. Hence future data would exercise more influence over source estimates.

9 For the 10 timestep case as supplied, the hemispheric standard deviations are 0.08 and the global 0.06. These increase as the time period for the inversion is decreased. The reason is that the average total source $s_+$ is constrained by the time trend in concentration. For a fixed observational error this trend is better constrained by longer records.

10 The covariance between the average source in each hemisphere is -0.01. This negative correlation implies that the sum of the two hemispheric sources (i.e. the global source) is better constrained than either source alone. For two variables with standard deviation $\sigma$ and correlation $x$, the variance of their sum is $2\sigma^2 + 2x\sigma^2 < \sigma^2$ implying $x < -0.5$.

11 As we increase the uncertainty, the predicted values approach the 'correct' value, i.e that consistent with the data.

12 The minimum value of the cost function (which is also the $\chi^2$ value for the inversion) is given by

$$(\mathbf{T}s_0 - d_0)^T \left(\mathbf{T}\mathbf{C}(s_0)\mathbf{T}^T + \mathbf{C}(d_0)\right)^{-1} (\mathbf{T}s_0 - d_0) \tag{34}$$

where $s$ is sources, $d$ is data, $\mathbf{T}$ is the Jacobian matrix, $\mathbf{C}(x)$ is the covariance matrix of $x$ and the subscript zero refers to initial estimates (see *Tarantola* [1987, p211]). For the initial uncertainties and zero prior sources $\chi^2 = 3.91$ i.e the misfit is too large. Doubling the initial uncertainty will reduce this to near 1 at the cost of doubling all predicted uncertainties. This uniform scaling of uncertainties will not change the final estimate since it does not change the relative weighting of prior sources and data. This can be seen by equally scaling the covariance matrices in *Enting* [1999, equation (A2b)]. Doubling only the source uncertainty gives $\chi^2 = 1.1$ and increases the uncertainty on the time-averaged source in one hemisphere from 0.08 to 0.11.

13 The source difference between the hemispheres rises from 1 to 1.16. The uncertainty rises because the Jacobian is now more bland than the original. This is a general property of Green's function methods that the more structure present in the Jacobian the better sources will be resolved.

14 As supplied, the sources for both the Green's function example and the trivial Kalman filter example are consistent with the data. In the case of the Kalman filter the initial state is also consistent whereas in the Green's function case we assumed zero gradient between the hemispheres before the start of the calculation. Thus the sources in the Kalman filter case do not change at all whereas they approach the true asymptotic value in the Green's function case. The uncertainty for the source in the Kalman filter case decreases throughout the period. This is because, with no stochastic covariance, the Kalman filter is merely a recursive estimation of the average source. The uncertainty decreases as more and more data is added at each timestep. The final uncertainty then is the same as the uncertainty for the average source in the Green's function case.

15 From the initial (incorrect) estimate of the sources the two estimates evolve quite differently with the Kalman filter case approaching the true solution and the Green's function case departing from it. This can be explained by the amount of data which influences each source estimate. In the Kalman filter as run in this example, the inclusion of more data refines the estimate of the long-term average source. Thus the final value of the source is most controlled by data and least by the initial state. In the Green's function case the situation is reversed. The posterior source calculated for the first year is influenced by all subsequent data. In the last year, the calculated posterior source is only influenced by the last datapoint hence the prior estimate has greater impact. This points out one important difference

between the calculations. In the Green's function example we are estimating 20 independent sources while in the Kalman filter example only 2. The *average* northern hemisphere source estimated by the Green's function run is 1.469 for this case while the final value for the Kalman filter is 1.495. The difference occurs because the prior estimate acts as a stronger constraint in the Green's function than the Kalman filter case. We saw in Exercise 9 that the uncertainty for an average of equally and normally distributed random variables is $\sigma/\sqrt{n}$ where $\sigma$ is the standard deviation for each variable. This also holds for initial estimates so that while the uncertainty on *each* source is 0.5, the uncertainty of the *average* is $0.5/\sqrt{10} \approx 0.16$. Thus the average source cannot adjust as close to the true solution as might first appear. If we increase the uncertainties by the factor $\sqrt{10}$ in the Green's function example we produce the same average source in that case as the final Kalman filter estimate.

16 With the dramatically increased observational error, concentration errors grow slightly through the period. The source uncertainty continues to fall since the estimates are still being refined by the observations, albeit only slowly. The simulated concentration error is controlled much more by the source error than the observational error. The uncertainty on the final source is only slightly different despite the great deal of essentially absent data from the middle of the run. This is because the average source, here as in the Green's function case, is controlled by the long-term gradients and trends. Similarly, with an initial incorrect state of $(1, 0)$ the final estimate for the northern hemisphere source is 1.495 with the missing data compared with 1.496 with all data. A case with the incorrect initial state and uncertainties of $(0.5, 1.5)$ throughout produces a final estimate of $(1.05 \pm 0.21, 0.49 \pm 0.19)$ i.e. only a small change in the estimate but almost a doubling in the uncertainty. Note that increased uncertainty in one hemisphere affects uncertainty everywhere, an unfortunate and general property of inversions using transport models.

17 In general the uncertainty is greater than the stochastic covariance. The reason for this is somewhat subtle but the underlying cause is that the sources are not directly observed, we infer them from changes in concentration. Hence to estimate sources at time $t$ we require concentrations at time $t + 1$ but these are not available to the Kalman filter until we are

estimating sources at time $t+1$. At each timestep, the new variance for the sources is equal to the old one + the stochastic covariance. In the update step of the Kalman filter, this covariance will be reduced by the presence of data. However because current data does not influence current sources this impact is limited. The source uncertainty will not continue to grow, but nor can we reduce it below the stochastic covariance. At its most general this merely shows that for an unobserved variable we are reliant on the quality of our model. This would not happen if we included some observations of the sources themselves.

18 With no **Q** the source uncertainties drop quickly after the start of the run. By the time there is a change in slope of the concentrations (corresponding to a change in sources) the uncertainty on the source evolution is too small to allow sources to adjust. Recall from Exercise 17 that the 'no **Q**' case corresponded to estimating constant sources, an assumption obviously violated by the methyl chloroform record. Note that the concentration downturn corresponded to the implementation of the Montreal Protocol limiting the use of this gas.

19 The sources here are not used in the calculation at all, unlike the Green's function calculation. We can regard the mismatch with sources as a commentary on the trivial chemical transport model we are using or the veracity of reported sources. Unlike the Green's function calculation the Kalman filter is not inherently a consistency check between estimated sources and concentrations. We could perform such a check by including extra observations corresponding to prior estimates of the sources. This would render the evolution model for the sources irrelevant so while the problem would be still formally a Kalman filter it would not be a very interesting one.

20 The $\chi^2$ value naturally increases greatly around the time of the onset of the Montreal Protocol when the constant source approximation breaks down.

21 The average $\chi^2$ is 1.25 and the maximum 16.7. Note that the sources change so rapidly through the record that a given **Q** cannot hope to capture the behaviour completely. In fact it is our random walk evolution which fails. Given our trivial model for evolution, we must either pick a **Q** so large as to make our uncertainties large or so small as to produce a very spiky mismatch function. Note, though, that $\chi^2$ is a random variable itself. It is properties of

its distribution, including its trend in time, which we should use as guidance. Finally note that **Q** represents a growth-rate of error i.e the rate of departure of our random-walk model. Thus it is dependent on the timestep of our numerical model.

22 With no stochastic covariance for the sources the Kalman filter reduces to a recursive least squares method in which data are used to successively refine an estimate of the average source over the whole period. By the end of the filter pass, all data has been used so there is nothing for the smoother to add. Since the estimate is being refined by successive data, the variance of the sources is quite small by the end. The smoother uses this variance to initialize the variance of the backward pass and ultimately the variances become small enough to cause numerical problems.

As we have seen, a single average source is not a very good representation for methyl chloroform over the study period. The smoother in this problem makes little sense in the absence of stochastic covariance. Once we add **Q** things behave more sensibly. Calculated concentrations are closer to the observations, in particular the levelling and decrease in concentrations is better simulated with the smoother than the filter. Uncertainties are smaller as they must be since the smoother refines the filter pass. Most strikingly the sharp decrease seen in the reported sources is captured with the smoother. The forward pass does not decrease quickly enough since we need future concentrations to estimate current sources.

23 The impact on modelled concentrations of all these changes is small. If transport is slowed (mixing time doubled) northern hemisphere sources will decrease since transport consitutes a loss to the northern hemisphere. The reverse occurs in the southern hemisphere. If lifetime is doubled (the chemical loss reduced) then sources reduce everywhere. Doubling the mixing time actually reduces the $\chi^2$ value from 1.25 to 1.0. This should not be interpreted as suggesting the greater mixing time as a better parameter choice however since the Kalman filter does not take the prior estimates of sources into account.

24 24–31 A `Matlab` program named `RTsolve1` in the directory `kalman/harmonic` walks you through the solution to Exercises 24–31. You can start running this program from the `Matlab` prompt by invoking its name, and hitting the carriage return key on your keyboard after the explanatory pauses.

32 32–40 The solutions to Exercises 32–40 can be seen by running the `Matlab` program `rtsolve2` in the `kalman/advect` directory. Once again, the program will pause from time to time with a small explanation about the solution.

41 Subtracting the true state $\mathbf{x}$ we get from equation (9) in *Ménard* [1999] and using equation (3) in *Ménard* [1999],

$$\tilde{\mathbf{x}}_k^a = (\mathbf{I} - \mathbf{KH})\tilde{\mathbf{x}}_k^f + \mathbf{K}\varepsilon_k^o \tag{35}$$

where $\tilde{\mathbf{x}}_k^a = \mathbf{x}_k - \hat{\mathbf{x}}_k^a$ and $\tilde{\mathbf{x}}_k^f = \mathbf{x}_k - \hat{\mathbf{x}}_k^f$. Taking the outer product of $\tilde{\mathbf{x}}_k^a$, then the conditional expectation, and assuming that $\tilde{\mathbf{x}}_k^f$ is uncorrelated with $\varepsilon_k^o$, we get

$$\mathbf{P}_k^a = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\mathbf{P}_k^f\,(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{36}$$

where $\mathbf{P}_k^a = \langle\tilde{\mathbf{x}}_k^a(\tilde{\mathbf{x}}_k^a)^T|\mathbf{y}_k^o,\ldots,\mathbf{y}_1^o\rangle$, and $\mathbf{P}_k^f = \langle\tilde{\mathbf{x}}_k^f\left(\tilde{\mathbf{x}}_k^f\right)^T|\mathbf{y}_k^o,\ldots,\mathbf{y}_1^o\rangle$. Expanding (36), we get

$$\begin{aligned}\mathbf{P}_k^a &= \mathbf{P}_k^f - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_k^f - \mathbf{P}_k^f\mathbf{H}_k^T\mathbf{K}_k^T \\ &\quad + \mathbf{K}_k\mathbf{H}_k\mathbf{P}_k^f\mathbf{H}_k^T\mathbf{K}_k^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \\ &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^f - \mathbf{P}_k^f\mathbf{H}_k^T\mathbf{K}_k^T \\ &\quad + \mathbf{K}_k(\mathbf{H}_k\mathbf{P}_k^f\mathbf{H}_k^T + \mathbf{R}_k)\mathbf{K}_k^T.\end{aligned} \tag{37}$$

Substituting equation (10) from *Ménard* [1999] into the last expression of (37) we get

$$\mathbf{P}_k^a = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^f - \mathbf{P}_k^f\mathbf{H}_k^T\mathbf{K}_k^T + \mathbf{P}_k^f\mathbf{H}_k^T\mathbf{K}_k^T \tag{38}$$

$$= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^f \tag{39}$$

Using the Kalman gain [*Ménard*, 1999, equation (10)], (39) can be rewritten as

$$\mathbf{P}_k^a = \mathbf{P}_k^f - (\mathbf{H}_k\mathbf{P}_k^f)^T\mathbf{\Gamma}_k^{-1}(\mathbf{H}_k\mathbf{P}_k^f) \tag{40}$$

The matrix $\mathbf{H}_k\mathbf{P}_k^f$ is $p \times n$, and it represents the forecast error covariance between the $p$ observation locations and the $n$ grid points. Let $\mathbf{p}_i$ denote the column vector of forecast error covariances between all $p$ observations and a single grid point $\mathbf{r}_i$,

$$\mathbf{p}_i = \left[P^f(\mathbf{r}_1^o, \mathbf{r}_i), P^f(\mathbf{r}_2^o, \mathbf{r}_i), \ldots, P^f(\mathbf{r}_p^o, \mathbf{r}_i)\right]^T. \tag{41}$$

We verify that

$$\lfloor\left(\mathbf{HP}^f\right)^T\mathbf{\Gamma}^{-1}\left(\mathbf{HP}^f\right)\rfloor_{i,j} = \mathbf{p}_i^T\mathbf{\Gamma}^{-1}\mathbf{p}_j \tag{42}$$

which gives the result

$$P^a(\mathbf{r}_i, \mathbf{r}_i) = P^f(\mathbf{r}_i, \mathbf{r}_i) - \mathbf{p}_i^T\mathbf{\Gamma}^{-1}\mathbf{p}_i \tag{43}$$

42 Since $B$ and $C$ are homogeneous and isotropic functions, $D$ can only be a function of $\rho = \|\mathbf{r} - \mathbf{r}'\|$. Moreover, at $\rho = 0$, $D_0(0) = 1$, thus $D$ is a homogeneous isotropic correlation function. From

$$D_0(\rho) = B_0(\rho)C_0(\rho), \tag{44}$$

we get

$$\frac{\partial^2 D_0}{\partial \rho^2} = \frac{\partial^2 B_0}{\partial \rho^2} + 2\frac{\partial B_0}{\partial \rho}\frac{\partial C_0}{\partial \rho} + B_0\frac{\partial^2 C_0}{\partial \rho^2}. \tag{45}$$

At $\rho = 0, B = C = 1$, and $\frac{\partial B}{\partial \rho} = \frac{\partial B}{\partial \rho} = 0$, and thus we get

$$\left(\frac{\partial^2 D_0}{\partial \rho^2}\right)_{\rho=0} = \left(\frac{\partial^2 B_0}{\partial \rho^2}\right)_{\rho=0} + \left(\frac{\partial^2 C_0}{\partial \rho^2}\right)_{\rho=0}, \tag{46}$$

from which the result on correlation length-scales follows.

43 Let us first write

$$\frac{\partial}{\partial \alpha_m}\det(\mathbf{A}) = \sum_{ij} \frac{\partial \det(\mathbf{A})}{\partial a_{ij}} \frac{\partial a_{ij}}{\partial \alpha_m}. \tag{47}$$

Differentiating the expansion in cofactors we get

$$\frac{\partial \det(\mathbf{A})}{\partial a_{ij}} = A_{ij}, \tag{48}$$

since in the cofactor expansion all the $A_{ij}$ do not contain $a_{ij}$. It then follows

$$\begin{aligned}
\frac{\partial}{\partial \alpha_m}\det(\mathbf{A}) &= \sum_{ij} A_{ij}\frac{\partial a_{ij}}{\partial \alpha_m} \\
&= \sum_i \left[\sum_j A_{ij}\frac{\partial a_{ij}}{\partial \alpha_m}\right] \\
&= \sum_i \left[(\mathbf{A}^c)^T \frac{\partial \mathbf{A}}{\partial \alpha_m}\right]_{ii} \\
&= \text{trace}\left[(\mathbf{A}^c)^T \frac{\partial \mathbf{A}}{\partial \alpha_m}\right].
\end{aligned} \tag{49}$$

Using the formula for the inverse we get,

$$\begin{aligned}
\text{trace}\left[(\mathbf{A}^c)^T \frac{\partial \mathbf{A}}{\partial \alpha_m}\right] &= \text{trace}\left[\det(\mathbf{A})\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \alpha_m}\right] \\
&= \det(\mathbf{A})\text{trace}\left[\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \alpha_m}\right] \\
&= \frac{\partial}{\partial \alpha_m}\det(\mathbf{A}),
\end{aligned} \tag{50}$$

from which we obtain

$$\frac{\partial}{\partial \alpha_m} \log(\det(\mathbf{A})) = \mathrm{trace}\left[\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \alpha_m}\right]. \qquad (51)$$

Differentiating $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$ we get

$$\frac{\partial \mathbf{A}^{-1}}{\partial \alpha_m} = -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \alpha_m}\mathbf{A}^{-1}, \qquad (52)$$

from which the inner product can be readily evaluated as

$$\begin{aligned} \frac{\partial}{\partial \alpha_m}\boldsymbol{\nu}^T\mathbf{A}^{-1}\boldsymbol{\nu} &= -\boldsymbol{\nu}^T\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \alpha_m}\mathbf{A}^{-1}\boldsymbol{\nu} \\ &= \mathrm{trace}\left[-\boldsymbol{\nu}^T\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \alpha_m}\mathbf{A}^{-1}\boldsymbol{\nu}\right] \\ &= \mathrm{trace}\left[-\mathbf{A}^{-1}\boldsymbol{\nu}^T\boldsymbol{\nu}\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \alpha_m}\right]. (53) \end{aligned}$$

44 The anisotropy of the covariance model

$$P(\mathbf{r}, \mathbf{r}') = \gamma^2\hat{\chi}(\mathbf{r})\hat{\chi}(\mathbf{r}')C(\mathbf{r}, \mathbf{r}'), \qquad (54)$$

arises from the fact that for any orthogonal transformation $g(\bullet)$,

$$\hat{\chi}(g(\mathbf{r}))\hat{\chi}(g(\mathbf{r}')) \neq \hat{\chi}(\mathbf{r})\hat{\chi}(\mathbf{r}'), \qquad (55)$$

except when the mixing ratio field is uniform. According to equation (29) from *Ménard* [1999] we obtain

$$\begin{aligned} B(\mathbf{r}, \mathbf{r}') &= \log\left[1 + \frac{P(\mathbf{r}, \mathbf{r}')}{\hat{\chi}(\mathbf{r})\hat{\chi}(\mathbf{r}')}\right] \\ &= \log\left[1 + \gamma^2 C(\mathbf{r}, \mathbf{r}')\right] \\ &\approx \gamma^2 C(\mathbf{r}, \mathbf{r}'), \qquad (56) \end{aligned}$$

and thus the covariance function $B$ is isotropic.

*Acknowledgments.* The authors thank M. Heimann, N. Mahowald, R. Prinn and A. Winguth for helpful comments on the manuscript. P. Rayner acknowledges the support of the Australian Government through its Cooperative Research Centres Programme.

## Appendix: System and Software Requirements

Note that this information is also included in the `README` file on the CD-ROM.

Exercise 1 does not require any software. Exercise 2 requires a Fortran compiler, either Fortran-77 or Fortran-90. To run the supplied code, compile then execute the file `boxmod_0.f` in the `adjoint/sens` directory.

Exercises 3–6 require that the TAMC system be installed. Ideally this should use a Unix system connected to the internet. If internet connection is not available, the output from the TAMC programme is provided for each exercise. This output should be linked with the relevant TAMC libraries. UNIX commands to install these libraries are described in the `INSTALL` file in the `adjoint/tamc` directory on the CD-ROM.

Exercises 7–13 can be run in two different ways. The `green/idl` directory contains `IDL` scripts to execute the examples. The supplied code may be executed by typing `@rungreen` at the `IDL>` prompt. The `green/fortran` directory contains Fortran-90 code to execute the examples. It can be run by compiling and linking the two files `green.f90` and `bayesinv.f90`. No external libraries are required. On a system supporting the `make` utility, a `Makefile` is included to compile, link and execute the example with the single command `make green`.

Exercises 14–17 can be run in two different ways. The `kalman/box/trivial/idl` directory contains `idl` scripts to execute the examples. The supplied code can be run by typing `@runkf1` at the `IDL>` prompt. Equivalent Fortran-90 code is contained in `kalman/box/trivial/fortran`. It can be run by compiling and executing the file `kf.f90`. No external libraries are required. On a system supporting the `make` utility, a `Makefile` is included to compile, link and execute the example with the single command `make kf`.

Exercises 18–23 can be run in two different ways. The `kalman/box/ch3ccl3/idl` directory contains `idl` scripts to execute the examples. The supplied code can be run by typing `@runkf2` at the `IDL>` prompt. The equivalent Fortran-90 code is contained in `kalman/box/ch3ccl3/fortran`. It can be run by compiling and executing the file `kf.f90`. No external libraries are required. On a system supporting the `make` utility, a `Makefile` is included to compile, link and execute the example with the single command `make kf`. The `idl` version is preferable since it contains graphical display of the results which will make interpretation easier. Note that on systems with 8 character file-names + 3 character extension (such as MS-DOS), you may need to change the names of the `c_CH3CCl3.d` and `src_CH3CCl3.d` files.

Exercises 24–40 require the MATLAB program. The code for Exercises 24–31 is found in the `kalman/dynamic/oscill` directory. It can be started by running the program `lew252m` from within MATLAB. The code for Exercises 32–40 is found in the `kalman/dynamic/advect` directory. It can be started by running the program `rtsolve2` from within MATLAB.

Exercises 41–44 are analytical and do not require any software.

## REFERENCES

Abramowitz, M. and I. A. Stegun, *Handbook of mathematical functions*, Dover, New York, 1970.

Bischof, C., *A collection of automatic differentiation tools*, `http://www.mcs.anl.gov/Projects/autodiff/AD_Tools/index.html`, 1999.

Bischof, C., A. Carle, G. Corlies, A. Griewank, and P. Hovland, ADIFOR: Generating derivative codes from FORTRAN programs, *Scientific Programming*, *1(1)*, 11–29, 1992.

Enting, I. G., Green's function methods of tracer inversion, this volume, American Geophysical Union, 1999.

Giering, R., *Tangent linear and adjoint model compiler, Users manual*, 1997, unpublished, available from `http://puddle.mit.edu/~ralf/tamc`.

Giering, R., Tangent linear and adjoint models, this volume, American Geophysical Union, 1999.

Giering, R., and T. Kaminski, Recipes for adjoint code construction, in press, ACM Trans. Math. Software, 1998.

Griewank, A., On automatic differentiation, in *Mathematical programming: recent developments and applications*, edited by M. Iri, and K. Tanabe, 83–108, Kluwer Academic Publishers, 1989.

Horwedel, J. E., GRESS: A preprocessor for sensitivity studies on Fortran programs, in *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, edited by A. Griewank, and G. F. Corliss, 243–250, SIAM, Philadelphia, Penn., 1991.

Houweling, S., F. Dentener, and J. Lelieveld, The impact of nonmethane hydrocarbon compounds on tropospheric photochemistry, *J. Geophys. Res.*, *D9*, 10673–10696, 1998.

Kaminski, T., M. Heimann, and R. Giering, A coarse grid three dimensional global inverse model of the atmospheric transport 1. Adjoint model and Jacobian matrix, *J. Geophys. Res.*, *104*, 18535–18553, 1999.

Law, R. M., P. J. Rayner, A. S. Denning, D. Erickson, I. Y. Fung, M. Heimann, S. C. Piper, M. Ramonet, S. Taguchi, J. A. Taylor, C. M. Trudinger, and I. G. Watterson, Variations in modelled atmospheric transport of carbon dioxide and the consequences for $CO_2$ inversions. *Glob. Biogeochem. Cyc.*, *10*, 483–496, 1996.

Lewis, F.L., *Optimal Estimation with an Introduction to stochastic control theory*, 376 pp., John Wiley & Sons, 1986.

Ménard, R., Tracer assimilation, this volume, American Geophysical Union, 1999.

Press, W. H., B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical recipes: The art of scientific computing*, Cambridge University Press, New York, 1986.

Prinn, R. G., Measurement equation for trace chemicals in fluids and solution of its inverse, this volume, American Geophysical Union, 1999.

Prinn, R., D. Cunnold, P. Simmonds, F. Alyea, R. Boldi, A. Crawford, P. Fraser, D. Gutzler, D. Hartley, R. Rosen and R. Rasmussen, Global average concentration and trend for hydroxyl radicals deduced from ALE/GAGE trichloroethane (methyl chloroform) data for 1978–1990, *J. Geophys. Res.*, *97*, 2445–2461, 1992.

Prinn, R. G., R. F. Weiss, B. R. Miller, J. Huang, F. N. Alyea, D. M. Cunnold, P. J. Fraser, D. E. Hartley, and P. G. Simmonds, Atmospheric trends and lifetime of

$CH_3CCl_3$ and global OH concentrations. *Science*, *269*, 187–192, 1995.

Rostaing, N., S. Dalmas, and A. Galligo, Automatic differentiation in Odyssée, *Tellus*, *45A*, 558–568, 1993.

Tarantola., A., *Inverse problem theory: methods for data fitting and parameter estimation*, Elsevier, Amsterdam, 1987.

Todling, R., Estimation theory and atmospheric data assimilation, this volume, American Geophysical Union, 1999.

Wunsch, C., *The ocean circulation inverse problem*, 442 pp., Cambridge University Press, 1996.

P.J. Rayner and C.M. Trudinger, CSIRO-AR PMB 1, Aspendale, Vic. 3195, Australia, (e-mail: peter.rayner@dar.csiro.au; cathy.trudinger@dar.csiro.au)

R. Giering, Jet Propulsion Laboratory, MS 300-323, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109, U.S.A., (e-mail: Ralf.Giering@jpl.nasa.gov)

T. Kaminski, Max Planck Institut für Meteorologie, Bundesstr. 55, D-20146 Hamburg (e-mail: kaminski@dkrz.de)

R. Ménard and R. Todling, NASA Goddard Space Flight Center, Code 910.3, Greenbelt, Maryland, USA. (e-mail: rtodling@dao.gsfc.nasa.gov; menard@dao.gsfc.nasa.gov)

[1]Meteorology CRC, Monash University, Clayton, 3168, Australia

[2]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, U.S.A.

[3]Max Planck Institut für Meteorologie, Hamburg, Germany

[4]NASA Goddard Space Flight Center, Greenbelt, Maryland, U.S.A.

[5]CSIRO Atmospheric Research, Aspendale, Australia

```
      program boxmod
      implicit none
c parameters
      integer ny !  number of years
      integer ntpy !  number of time steps per year
      parameter(ny=10,ntpy=50)
      real src(2,ny) !  sources
      real mixrate, invlif !  1/mixing rate; 1/life time
      real kt2pptv !  conversion source to concentration
      integer i,l !  loop counters
      real c(2),cnew !  concentrations
      integer y !  year of source
      real src_n,src_s,tot !  fraction in nh and sh, total in kt
c initialize sources
c read three comment lines plus the 1977 record,
c i.e.  start with the 1978 sources
      open(unit=1,file='src_CH3CCl3.d',status='old')
      read(1,'(////)')
      do i=1,ny
         read(1,'(8x,i4,2(f6.3),f6.1)') y,src_n,src_s,tot
         src(1,i) =src_n*tot
         src(2,i) =src_s*tot
      end do
      close(1)
c initialize transport and sink
      mixrate = 1.  !  mixing rate in 1/year
      invlif = 1./4.7 !  inverse lifetime in 1/year
           !  sander houweling found 4.7 years
c conversion for kt to pptv within a hemisphere
      kt2pptv = 0.471 * 2 * 12/133.5
c initialize concentration with values for 1978
      c(1) = 84.  !  northern box
      c(2) = 60.  !  southern box
c output
      write(*,'(a50)') 'The simulated concentration is :  '
      write(*,'(a10,2(a20))'),'year','box1','box2'
      write(*,'(i10,2(12x,f8.4))')
     .      0,c(1),c(2)
c calculate concentrations with forward differencing box model
c and add contribution to misfit function every year
c (cnew stores the new value of c(1) because the old is
c needed for computation of c(2) )
      do i=1,ny
         do l=1,ntpy
         cnew = c(1) + 1./ntpy *
     .           ( kt2pptv*src(1,i) - (c(1)-c(2))
     .           * mixrate - invlif*c(1) )
         c(2) = c(2) + 1./ntpy *
     .           ( kt2pptv*src(2,i) - (c(2)-c(1))
     .           * mixrate - invlif*c(2) )
         c(1) = cnew
         enddo
c output
         write(*,'(i10,2(12x,f8.4))')
     .         i,c(1),c(2)
      enddo
      end
```

**Figure 1.** The code of Boxmod.

**Figure 1.** The code of Boxmod.

```
C==============================================
C The subroutine "MODEL" is called by the optimization
C procedure.  It has to calculate the cost function "FC"
C depending on the control vector "X(N)".
C==============================================
      SUBROUTINE MODEL( N, X, FC )
      implicit none
      INTEGER N
      REAL X(N), FC

#include "boxmod.h"
      integer i,j,l                 !  loop counters
      real c(2),cnew                !  concentrations

c tamc directive to initialize a tape for the trajectory
c cadj init tape1 = 'trajectory'
c alternatively memory can be used
cadj init tape1 = MEMORY

c copy control variables
      mixrate = x(1)
      invlif = x(2)
      do i=1,ny
         do j=1,2
           src(j,i)=x(2+i+(j-1)*ny)
         enddo
      end do
c initialize concentration with values for 1978
      c(1) = 84.                    !  northern box
      c(2) = 60.                    !  southern box

c calculate concentrations with forward differencing box model
c and add contribution to misfit function every year
c (cnew stores the new value of c(1) because the old is
c needed for computation of c(2) )
      do i=1,ny
         do l=1,ntpy
cadj store c = tape1
           cnew = c(1) + 1./ntpy *
     .            ( kt2pptv*src(1,i) - (c(1)-c(2))
     .            * mixrate - invlif*c(1) )
           c(2) = c(2) + 1./ntpy *
     .            ( kt2pptv*src(2,i) - (c(2)-c(1))
     .            * mixrate - invlif*c(2) )
           c(1) = cnew
         enddo
      enddo
c set output variable
      fc=c(1)
      END
```

**Figure 2.** Subroutine `model` defining the function to be differentiated in Exercise 3. The subroutine has been constructed by rearranging the code of `Boxmod`. The header file `boxmod.h` is listed in Figure 3.

**Figure 2.** Subroutine `model` defining the function to be differentiated in Exercise 3. The subroutine has been constructed by rearranging the code of `Boxmod`. The header file `boxmod.h` is listed in Figure 3.

```
c header file for boxmod
c parameters
      integer ny !  number of years
      integer ntpy !  number of time steps per year
      parameter(ny=10,ntpy=50)
c variables
      real src(2,ny) !  sources
      real mixrate, invlif !  1/mixing rate; 1/life time
      real kt2pptv !  conversion source to concentration
      common /vars / mixrate, invlif, src, kt2pptv
c
c values of control variables:
      real src0(2,ny) !  sources
      real mixrate0, invlif0 !  1/mixing rate; 1/life time
      common /cvars / mixrate0, invlif0, src0
```

**Figure 3.** Header file `boxmod.h` for subroutines `numbmod, model, initmod` and `postmod`.

**Figure 3.** Header file `boxmod.h` for subroutines `numbmod, model, initmod` and `postmod`.

```
      subroutine admodel( n, x, fc, adx, adfc )
C**************************************************************
C**************************************************************
C** This routine was generated by the **
C** Tangent linear and Adjoint Model Compiler, TAMC 4.97 **
C**************************************************************
C**************************************************************
      implicit none
C==============================================
C define parameters
C==============================================
      integer ntpy
      parameter ( ntpy = 50 )
      integer ny
      parameter ( ny = 10 )
C==============================================
C define common blocks
C==============================================
      common /advars/ admixrate, adinvlif, adsrc
      real adinvlif, admixrate, adsrc(2,ny)
      common /vars/ mixrate, invlif, src, kt2pptv
      real invlif, kt2pptv, mixrate, src(2,ny)
C==============================================
C define arguments
C==============================================
      integer n
      real adfc, adx(n), fc, x(n)
C==============================================
C define local variables
C==============================================
      real adc(2), adcnew, c(2), cnew
      integer i, ip1, j, l
C----------------------------------------------
C RESET GLOBAL ADJOINT VARIABLES
C----------------------------------------------
      call adzero
C----------------------------------------------
C RESET LOCAL ADJOINT VARIABLES
C----------------------------------------------
      do ip1 = 1, 2
         adc(ip1) = 0.
      end do
      adcnew = 0.
```

**Figure 4.** The adjoint and modified forward codes of `Boxmod`, declaration and initialization of adjoint variables. TAMC output has been slightly edited to fit better in the figure.

**Figure 4.** The adjoint and modified forward codes of `Boxmod`, declaration and initialization of adjoint variables. TAMC output has been slightly edited to fit better in the figure.

```
C----------------------------------------------
C ROUTINE BODY
C----------------------------------------------
C----------------------------------------------
C FUNCTION AND TAPE COMPUTATIONS
C----------------------------------------------
      mixrate = x(1)
      invlif = x(2)
      do i = 1, ny
         do j = 1, 2
           src(j,i) = x(2+i+(j-1)*ny)
         end do
      end do
      c(1) = 84.
      c(2) = 60.
      do i = 1, ny
         do l = 1, ntpy
           call adstore( 'memory_1_model_c',16,c,8,2,1+((-1)+i)
     .               *ntpy+l-1)
           cnew = c(1)+1./ntpy*(kt2pptv*src(1,i)-(c(1)-c(2))
     .               *mixrate-invlif*c(1))
           c(2) = c(2)+1./ntpy*(kt2pptv*src(2,i)-(c(2)-c(1))
     .               *mixrate-invlif*c(2))
           c(1) = cnew
         end do
      end do
      fc = c(1)
C----------------------------------------------
C ADJOINT COMPUTATIONS
C----------------------------------------------
      mixrate = x(1)
      invlif = x(2)
      adc(1) = adc(1)+adfc
      adfc = 0.
      do i = ny, 1, -1
         do l = ntpy, 1, -1
           call adresto( 'memory_1_model_c',16,c,8,2,1+((-1)+i)
     .               *ntpy+l-1 )
           adcnew = adcnew+adc(1)
           adc(1) = 0.
           adinvlif = adinvlif-adc(2)*1./float(ntpy)*c(2)
           admixrate = admixrate-adc(2)*1./float(ntpy)
     .               *(c(2)-c(1))
           adsrc(2,i) = adsrc(2,i)+adc(2)*1./float(ntpy)*kt2pptv
           adc(1) = adc(1)+adc(2)*1./float(ntpy)*mixrate
           adc(2) = adc(2)*(1-1./float(ntpy)*(mixrate+invlif))
           adc(2) = adc(2)+adcnew*1./float(ntpy)*mixrate
           adc(1) = adc(1)+adcnew*(1-1./float(ntpy)
     .               *(mixrate+invlif))
           adinvlif = adinvlif-adcnew*1./float(ntpy)*c(1)
           admixrate = admixrate-adcnew*1./float(ntpy)
     .               *(c(1)-c(2))
           adsrc(1,i) = adsrc(1,i)+adcnew*1./float(ntpy)*kt2pptv
           adcnew = 0.
         end do
      end do
      adc(2) = 0.
      adc(1) = 0.
      do i = 1, ny
         do j = 1, 2
           adx(2+i+(j-1)*ny) = adx(2+i+(j-1)*ny)+adsrc(j,i)
           adsrc(j,i) = 0.
         end do
      end do
      adx(2) = adx(2)+adinvlif
      adinvlif = 0.
      adx(1) = adx(1)+admixrate
      admixrate = 0.
      end
```

**Figure 5.** The adjoint and modified forward codes of `Boxmod`, function computation including computation and storing of `c` as well as adjoint computations including reading of `c`. TAMC output has been slightly edited to fit better in the figure.

**Figure 5.** The adjoint and modified forward codes of `Boxmod`, function computation including computation and storing of `c` as well as adjoint computations including reading of `c`. TAMC output has been slightly edited to fit better in the figure.

```
      subroutine adzero
C*************************************************************
C*************************************************************
C** This routine was generated by the **
C** Tangent linear and Adjoint Model Compiler, TAMC 4.97 **
C*************************************************************
C*************************************************************
      implicit none
C==============================================
C define parameters
C==============================================
      integer ny
      parameter ( ny = 10 )
C==============================================
C define common blocks
C==============================================
      common /advars/ admixrate, adinvlif, adsrc
      real adinvlif, admixrate, adsrc(2,ny)
C==============================================
C define local variables
C==============================================
      integer ip1, ip2
      admixrate = 0.
      adinvlif = 0.
      do ip2 = 1, ny
         do ip1 = 1, 2
           adsrc(ip1,ip2) = 0.
         end do
      end do
      end
```

**Figure 6.** The adjoint and modified forward codes of `Boxmod`, subroutine `adzero` for initialization of global adjoint variables. TAMC output has been slightly edited to fit better in the figure.

**Figure 6.** The adjoint and modified forward codes of `Boxmod`, subroutine `adzero` for initialization of global adjoint variables. TAMC output has been slightly edited to fit better in the figure.

```
C==============================================
C This subroutine sets the number
C of control variables
C==============================================
      SUBROUTINE NUMBMOD( N )
      implicit none
#include "boxmod.h"
      INTEGER N
      n = 2+2*ny

      END
```

**Figure 7.** Code of Subroutine `numbmod`. `numbmod` is one of the subroutines needed by `TAMLINK` to link `admodel` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

**Figure 7.** Code of Subroutine `numbmod`. `numbmod` is one of the subroutines needed by `TAMLINK` to link `admodel` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

```
C==============================================
C The subroutine "INITMOD" is called before the
C optimization.  It must set a first guess
C of the parameter vector.
C It may also contain the initialization of
C the model.
C==============================================
      SUBROUTINE INITMOD( N, X )
      implicit none

      INTEGER N
      REAL X(N)
#include "boxmod.h"

      integer i,j                   !  loop counter
      integer y                     !  year of source
      real src_n,src_s,tot          !  fraction in nh and sh,
                                    !  total in kt
c initialize sources
c read three comment lines plus the 1977 record,
c i.e.  start with the 1978 sources
      open(unit=1,file='src_CH3CCl3.d',status='old')
      read(1,'(////)')
      do i=1,ny
         read(1,'(8x,i4,2(f6.3),f6.1)') y,src_n,src_s,tot
         src0(1,i) =src_n*tot
         src0(2,i) =src_s*tot
      end do
      close(1)

c initialize transport and sink
      mixrate0 = 1.                 !  mixing rate in 1/year
      invlif0 = 1./4.7              !  inverse lifetime in 1/yr
           !  sander houweling found 4.7 years

c conversion for kt to pptv within a hemisphere
c we use 0.471 to transform from
c gigatons of carbon in co2 to ppmv in the entire atmosphere
c the ratio of the molecular weights of carbon and CH3CCl3
c is approximately 12/133.5
c replacing Gt by kt and ppmv by pptv cancels out
      kt2pptv = 0.471 * 2 * 12/133.5

c set first guess of (potential) control vars
      x(1) = mixrate0
      x(2) = invlif0
      do i=1,ny
         do j=1,2
           x(2+i+(j-1)*ny)=src0(j,i)
         enddo
      end do
      END
```

**Figure 8.** Code of Subroutine `initmod`. `initmod` is one of the subroutines needed by `TAMLINK` to link `admodel` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

**Figure 8.** Code of Subroutine `initmod`. `initmod` is one of the subroutines needed by `TAMLINK` to link `admodel` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

```
C==============================================
C The subroutine "POSTMOD" is called after
C the optimization.
C It should contain the output of the results.
C==============================================
      SUBROUTINE POSTMOD( N, X, FC, ADX )
      implicit none

      INTEGER N
      REAL X(N), FC, ADX(N)
      INTEGER I

#include "boxmod.h"
      write(*,'(a25,3(2x,f13.4))')
     .     'The value of fc is :  ',fc
      write(*,'(a25)') 'Its derivatives are :  '
      write(*,'(a55,1(2x,f13.4))')
     .     'with resp.  to mixing rate :  ', adx(1)
      write(*,'(a55,1(2x,f13.4))')
     .     'with resp.  to inv.  lifetime :  ', adx(2)
      write(*,'(a55)') 'with resp.  to sources :  '
      write(*,'(a10,2(a30))'),'year','box1','box2'
      do i=1,ny
         write(*,'(i10,2(22x,f8.4))')
     .         i,adx(2+i),adx(2+i+ny)
      end do
      END
```

**Figure 9.** Code of Subroutine `postmod`. `postmod` is one of the subroutines needed by `TAMLINK` to link `admodel` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

**Figure 9.** Code of Subroutine `postmod`. `postmod` is one of the subroutines needed by `TAMLINK` to link `admodel` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

```
C==============================================
C This is the top level routine,
C it has to calculate the dependent variables Y(M)
C out of the independent variables X(N)
C==============================================
      SUBROUTINE FUNC( N, X, M, Y )
      INTEGER N, M
      REAL X(N), Y(M)

#include "boxmod.h"
      integer i,j,l                ! loop counters
      real c(2),cnew               ! concentrations

c tamc directive to initialize a tape for the trajectory
c (adjoint code is generated to check tangent linear code)
c cadj init tape1 = 'trajectory'
c alternatively memory can be used
cadj init tape1 = MEMORY
c copy control variables
      mixrate = x(1)
      invlif = x(2)
c initialize concentration with values for 1978
      c(1) = 84.                   ! northern box
      c(2) = 60.                   ! southern box

c calculate concentrations with forward differencing box
c model and add contribution to misfit function every year
c (cnew stores the new value of c(1) because the old is
c needed for computation of c(2) )
      do i=1,ny
         do l=1,ntpy
cadj store c = tape1
           cnew = c(1) + 1./ntpy *
     .            ( kt2pptv*src(1,i) - (c(1)-c(2))
     .            * mixrate - invlif*c(1) )
           c(2) = c(2) + 1./ntpy *
     .            ( kt2pptv*src(2,i) - (c(2)-c(1))
     .            * mixrate - invlif*c(2) )
           c(1) = cnew
         enddo
c save output
         do j=1,2
           y(j+(i-1)*2) = c(j)
         enddo
      enddo
      END
```

**Figure 10.** Subroutine `func` defining the function to be differentiated in Exercise 4. The subroutine has been constructed by rearranging the code of `Boxmod`. The header file `boxmod.h` is listed in Figure 3.

**Figure 10.** Subroutine `func` defining the function to be differentiated in Exercise 4. The subroutine has been constructed by rearranging the code of `Boxmod`. The header file `boxmod.h` is listed in Figure 3.

```
      subroutine g_func( n, x, m, y, g_p_, g_x, ldx, g_y, ldy )
C**********************************************************
C**********************************************************
C** This routine was generated by the **
C** Tangent linear and Adjoint Model Compiler, TAMC 4.97 **
C**********************************************************
C**********************************************************
      implicit none
C==============================================
C define parameters
C==============================================
      integer g_pmax
      parameter ( g_pmax = 20 )
      integer ntpy
      parameter ( ntpy = 50 )
      integer ny
      parameter ( ny = 10 )
C==============================================
C define common blocks
C==============================================
      common /g_vars/ g_mixrate, g_invlif
      real g_invlif(g_pmax)
      real g_mixrate(g_pmax)
      common /vars/ mixrate, invlif, src, kt2pptv
      real invlif
      real kt2pptv
      real mixrate
      real src(2,ny)
C==============================================
C define arguments
C==============================================
      integer g_p_
      integer ldx
      integer ldy
      integer m
      integer n
      real g_x(ldx,n)
      real g_y(ldy,m)
      real x(n)
      real y(m)
C==============================================
C define local variables
C==============================================
      real c(2)
      real cnew
      real g_c(g_pmax,2)
      real g_cnew(g_pmax)
      integer g_i_
      integer i
      integer j
      integer l
C----------------------------------------------
C CHECK PACT LOWER EQUAL PMAX
C----------------------------------------------
      if (g_p_ .gt.  g_pmax) then
        stop 'error :  pact is greater than pmax'
      endif
```

**Figure 11.** The tangent linear and forward code of `Boxmod`, declaration and initialization of tangent linear variables. TAMC output has been slightly edited to fit better in the figure.

**Figure 11.** The tangent linear and forward code of `Boxmod`, declaration and initialization of tangent linear variables. TAMC output has been slightly edited to fit better in the figure.

```
C----------------------------------------------
C TANGENT LINEAR AND FUNCTION STATEMENTS
C----------------------------------------------
     do g_i_ = 1, g_p_
       g_mixrate(g_i_) = g_x(g_i_,1)
     end do
     mixrate = x(1)
     do g_i_ = 1, g_p_
       g_invlif(g_i_) = g_x(g_i_,2)
     end do
     invlif = x(2)
     do g_i_ = 1, g_p_
       g_c(g_i_,1) = 0.
     end do
     c(1) = 84.
     do g_i_ = 1, g_p_
       g_c(g_i_,2) = 0.
     end do
     c(2) = 60.
     do i = 1, ny
       do l = 1, ntpy
         do g_i_ = 1, g_p_
           g_cnew(g_i_) = g_c(g_i_,2)*1./float(ntpy)*mixrate
    $+g_c(g_i_,1)*(1-1./float(ntpy)*(mixrate+invlif))
    $-g_invlif(g_i_)*1./float(ntpy)*c(1)-g_mixrate(g_i_)
    $*1./float(ntpy)*(c(1)-c(2))
         end do
         cnew = c(1)+1./ntpy*(kt2pptv*src(1,i)-(c(1)-c(2))
    $*mixrate-invlif*c(1))
         do g_i_ = 1, g_p_
           g_c(g_i_,2) = g_c(g_i_,2)*(1-1./float(ntpy)
    $*(mixrate+invlif))+g_c(g_i_,1)*1./float(ntpy)*mixrate
    $-g_invlif(g_i_)*1./float(ntpy)
    $*c(2)-g_mixrate(g_i_)*1./float(ntpy)*(c(2)-c(1))
         end do
         c(2) = c(2)+1./ntpy*(kt2pptv*src(2,i)-(c(2)-c(1))
    $*mixrate-invlif*c(2))
         do g_i_ = 1, g_p_
           g_c(g_i_,1) = g_cnew(g_i_)
         end do
         c(1) = cnew
       end do
       do j = 1, 2
         do g_i_ = 1, g_p_
           g_y(g_i_,j+(i-1)*2) = g_c(g_i_,j)
         end do
         y(j+(i-1)*2) = c(j)
       end do
     end do
     end
```

**Figure 12.** The tangent linear and forward code of `Boxmod`, function and tangent linear computations. TAMC output has been slightly edited to fit better in the figure.

**Figure 12.** The tangent linear and forward code of `Boxmod`, function and tangent linear computations. TAMC output has been slightly edited to fit better in the figure.

```
C==============================================
C This subroutine sets the number
C of independent and dependent variables
C==============================================
      SUBROUTINE SETFUNC( N, M )
      implicit none
#include "boxmod.h"
      INTEGER N, M
      n = 2
      m = 2*ny
      END
```

**Figure 13.** Code of Subroutine `setfunc`. `setfunc` is one of the subroutines needed by `TAMLINK` to link `g_func` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

**Figure 13.** Code of Subroutine `setfunc`. `setfunc` is one of the subroutines needed by `TAMLINK` to link `g_func` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

```
C==============================================
C The subroutine INITFUNC
C must set the independent variables
C==============================================
      SUBROUTINE INITFUNC( N, X )
      implicit none

      INTEGER N
      REAL X(N)
#include "boxmod.h"

      integer i                      !  loop counter
      integer y                      !  year of source
      real src_n,src_s,tot           !  fraction in nh and sh
                                     !  total in kt
c initialize sources
c read three comment lines plus the 1977 record,
c i.e.  start with the 1978 sources
      open(unit=1,file='src_CH3CCl3.d',status='old')
      read(1,'(////)')
      do i=1,ny
         read(1,'(8x,i4,2(f6.3),f6.1)') y,src_n,src_s,tot
         src(1,i) =src_n*tot
         src(2,i) =src_s*tot
      end do
      close(1)

c initialize transport and sink
      mixrate0 = 1.                  !  mixing rate in 1/year
      invlif0 = 1./4.7               !  inverse lifetime in 1/yr
           !  sander houweling found 4.7 years

c conversion for kt to pptv within a hemisphere
c we use 0.471 to transform from
c gigatons of carbon in co2 to ppmv in the entire atmosphere
c the ratio of the molecular weights of carbon and CH3CCl3
c should be something like 12/133.5
c replacing Gt by kt and ppmv by pptv cancels out
      kt2pptv = 0.471 * 2 * 12/133.5

c set first guess of control vars
      x(1) = mixrate0
      x(2) = invlif0
      END
```

**Figure 14.** Code of Subroutine `initfunc`. `initfunc` is one of the subroutines needed by `TAMLINK` to link `g_func` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

**Figure 14.** Code of Subroutine `initfunc`. `initfunc` is one of the subroutines needed by `TAMLINK` to link `g_func` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

```
C================================================
C The subroutine "POSTFUNC" is called at last
C It should contain the output of the results.
C================================================
      SUBROUTINE POSTFUNC( N, X, M, Y, GDX, LDX )
      implicit none

      INTEGER N, M, LDX
      REAL X(N), Y(M), GDX(LDX,M)

#include "boxmod.h"
      integer i                    !  loop counters

      write(*,'(a25)') 'The sensitivities are :  '
      write(*,'(a10,2(a20))'),'year','c box1','c box2'
      write(*,'(a10,4(a10))'),' ',
     .     'mixrate','1/lifet','mixrate','1/lifet'
      do i=1,ny
         write(*,'(i10,4(2x,f8.2))')
     .         i,gdx(1,1+(i-1)*2),gdx(2,1+(i-1)*2),
     .         gdx(1,2+(i-1)*2),gdx(2,2+(i-1)*2)
      end do

      END
```

**Figure 15.** Code of Subroutine `postfunc`. `postfunc` is one of the subroutines needed by `TAMLINK` to link `g_func` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

**Figure 15.** Code of Subroutine `postfunc`. `postfunc` is one of the subroutines needed by `TAMLINK` to link `g_func` to a main program for computation of the sensitivity. The header file `boxmod.h` is listed in Figure 3.

Forward mode

$$\begin{aligned}
& \begin{bmatrix} x & x & x \end{bmatrix} \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \begin{bmatrix} x & x \\ x & x \\ x & x \end{bmatrix} \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \\
= \; & \begin{bmatrix} x & x & x \end{bmatrix} \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \\
= \; & \begin{bmatrix} x & x & x \end{bmatrix} \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \\
= \; & \begin{bmatrix} x & x & x & x & x \end{bmatrix}
\end{aligned}$$

Reverse mode

$$\begin{aligned}
& \begin{bmatrix} x & x & x \end{bmatrix} \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \begin{bmatrix} x & x \\ x & x \\ x & x \end{bmatrix} \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \\
= \; & \begin{bmatrix} x & x & x \end{bmatrix} \begin{bmatrix} x & x \\ x & x \\ x & x \end{bmatrix} \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \\
= \; & \begin{bmatrix} x & x \end{bmatrix} \begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \end{bmatrix} \\
= \; & \begin{bmatrix} x & x & x & x & x \end{bmatrix}
\end{aligned}$$

**Figure 16.** This diagram illustrates the differences in the storage requirements and number of operations. The same matrix product, whose result has 1 row and 5 columns, is evaluated in forward mode, i.e. from right to left (top), and in reverse mode, i.e. from left to right (bottom). In forward mode the matrices holding the intermediate results have 5 columns, while in reverse mode they have 1 row.

**Figure 16.** This diagram illustrates the differences in the storage requirements and number of operations. The same matrix product, whose result has 1 row and 5 columns, is evaluated in forward mode, i.e. from right to left (top), and in reverse mode, i.e. from left to right (bottom). In forward mode the matrices holding the intermediate results have 5 columns, while in reverse mode they have 1 row.

```
==================================
  COMPUTATION OF FUNCTION AND
  DERIVATIVES IN REVERSE MODE
==================================
  The value of fc is :      125.8274
  Its derivatives are :
      with resp. to mixing rate :      -10.3982
      with resp. to inv. lifetime :    -424.1904
      with resp. to sources :
      year             box1             box2
         1           0.0056           0.0056
         2           0.0069           0.0069
         3           0.0086           0.0086
         4           0.0106           0.0106
         5           0.0132           0.0132
         6           0.0163           0.0163
         7           0.0202           0.0201
         8           0.0251           0.0248
         9           0.0327           0.0291
        10           0.0554           0.0211
```

**Figure 17.** Sensitivities computed by adjoint model to Boxmod.

**Figure 17.** Sensitivities computed by adjoint model to Boxmod.

```
==============================
  COMPUTATION OF FUNCTION AND
  JACOBIAN IN FORWARD MODE
==============================
  The sensitivities are :
      year            c box1            c box2
             mixrate  1/lifet   mixrate  1/lifet
         1     –8.06    –71.67     8.06    –63.61
         2     –8.64   –131.99     8.64   –123.34
         3     –8.77   –185.58     8.77   –176.81
         4     –8.49   –232.22     8.49   –223.73
         5     –8.54   –272.68     8.54   –264.15
         6     –9.12   –308.90     9.12   –299.78
         7     –9.44   –341.67     9.44   –332.23
         8     –9.53   –371.27     9.53   –361.74
         9     –9.81   –398.41     9.81   –388.60
        10    –10.40   –424.19    10.40   –413.79
```

**Figure 18.** Sensitivities computed by tangent linear model to Boxmod.

**Figure 18.** Sensitivities computed by tangent linear model to Boxmod.

Table 1. Parameters for the finite–difference.

| | |
|---|---|
| Domain | for $-2 \leq x \leq 2$ |
| Mesh size | $\Delta x = 0.2$ |
| Time step | $\Delta t = 0.05$ |

Table 2. Observational error standard deviations.

| | |
|---|---|
| Assimilation time period | 1 time unit |
| Courant number | 0.95 |
| Obs frequency | $4 \times \Delta t$ |
| Obs error std dev | 0.02 |
| Obs sparsity | left–half of grid points |

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.

EXERCISES

RAYNER ET AL.