*Proc. Indian natn. Sci. Acad.*, **60**, A, No. 1, 1994, pp. 193-204.


# VARIATIONAL DATA ASSIMILATION WITH AN ADIABATIC VERSION OF THE LMD GCM USING ADJOINT METHOD

H C UPADHYAYA, SANGEETA AGARWAL AND O TALAGRAND*

*Centre for Atmospheric Science, Indian Institute of Technology Hauz Khas, New Delhi-*110 016, *India*

(*Received* 14 *June* 1993; *Accepted* 5 *July* 1993)

Variational four-dimensional data assimilation is performed using an adiabatic version of the Laboratoire de Meterologie Dynamique (LMD) multi-level grid point global model. The low resolution model used here has 8 parallels of 16 points each with 5 sigma levels. The cost function or distance function consists of a square sum of the difference between the model forecast and obervations over a time interval. The derivation of the adjoint of the LMD GCM (General Circulation Model) is presented. This adjoint model is then used to compute the gradient of the distance function with respect to the initial conditions. This is performed by a single backward time integration of the derived adjoint equations. The optimal solution is obtained by using the conjugate gradient method. The test experiment results show that the variational assimilation technique performed well. The main objective of this study, that is, to develop the adjoint code for the adiabatic version of the LMD GCM is achieved and tested successfully.

**Key Words: Adjoint Technique; Conjugate Gradient Method; Cost Function; Distance Function; Descent Methods; Variational Assimilation**

## Introduction

Numercial modelling has now become a major tool for the study of atmospheric and oceanic dynamics such as weather forecasting and its theoretical and fundamental aspects. Numerical models are computer programs which compute the temporal evolution of the atmosphere and oceanic flow from specified initial and boundary conditions. These conditions control the evolution of the solution in space and time.

The advent of powerful computers has allowed the development of very high resolution models, such as those which are used for operational medium-range weather forecasting. The heterogeneous distribution of data in space and time are available from satellite, and other remote sensing devices. There is a need for an improved assimilation system because of these advances in numerical modelling and the observation network.

There are many techinques used in data assimilation. But considerable attention has been focussed on four-dimensional variational data assimilation[1-7]. This powerful technique is also applicable to sensitivity and analysis[8-10] and to

*Laboratoire de Meteorologie Dynamique, Ecole Normale Superieur, 24, rue Lhomond, 75231, cedex 05, Paris (France)

parameter estimation. Another application of adjoint equations is the determination of the perturbations which, when superimposed on given meteorological fields, will imply most rapidly.

The objective of variational data assimilation is to define, as accurately as possible, the optimal solution which requires at each iteration the gradients of the cost-function with respect to all the adjusting parameters (e.g., the initial conditions) of a numerical model by fitting the model dynamics to data over a given period of time, where the optimality is measured by a cost-function that expresses the degree of discrepancy or distance function between the model and the data.

Hoffmann's[3] way of obtaining these gradients was found to be impossible to implement on present-day computers. A feasible, yet still expensive, way of evaluating these gradients is through the adjoint technique.

The variational approach to assimilation, with explicit use of the model adjust equations, seems to have been first suggested by Penenko and Obraztsov[13]. They studied a simple linear, small-dimensional example, which nevertheless contained all the basic ingradients of the method. The equation was studied again by Lewis and Derber[2] and La Dimet and Talagrand[1]. Since then, a fairly large number of works have been performed[6,7,14,15]. The gradient of the distance function is computed by first integrating the forecast model forward and then by integrating the adjoint model equations backward in time. The forcing terms are added to the solutions of the adjoint equations at times when observational data are available. Once the values of the distance function and its gradients are available, different minimization methods can be employed to find the optimal solution. In this paper, results are presented for a variational assimilation system being developed for the low-resolution version of the LMD GCM. In the next section, after a brief description of the LMD model and its discretization in space and time, we define the distance function and derive the adjoint equations for the model. Here we also deal with the verification of the correctness of the adjoint code and the gradient. Results of this study are presented in the third section. A summary and general conclusions are discussed in the last section.

## Variational Formalism

*Brief Description of the Model*

The model equations are formualted using cylindrical; coordinates $x$ (longitude) and $y$ ($\sin\phi$, where $\phi$ is the latitude). The mapping is area-preserving. Horizontal wind velocity is equivalently described by its natural components in the zonal and meridional directions.

The horizontal grid is chosen regular in the $(x, y)$ plane. The density of grid points on the sphere is therefore uniform; however, anisotrophy becomes quite large in the vicinity of both poles. The actual model has 48 parallels of 64 points each. The vertical coordinate is chosen as the normalized pressure $\sigma = p/p_s$. The layers are evenly spaced in an auxiliary coordinate $z$, the mapping $z \to \sigma$ being an analytic mapping (0,1) onto itself. The number of layers is 11. In $\sigma$ coordinate, the layers are unevenly spaced. They are relatively thinner in the vi-

cinity of the ground and near top. The horizontal grid is Arakawa-C. The basic thermodynamic variable in the model is potential enthalpy rather than temperature. In order to define temperature without ambiguity, we introduce an alternative independent vertical variable

$$s = \sigma^{k} = \pi / \pi_{s}$$

defined at the layer cores. Values of $\sigma$ will be considered at interfaces only; the correspondence between discrete values of $s$ and discrete values of $\sigma$ is given by

$$\overline{\sigma \delta_{z} s^{z}} = ks \delta_{z} \sigma.$$

For more details on the LMD GCM, *see* Sadourny and Laval.

*The Adjoint Method*

For any continuous linear operator **L** defined on $E$, it is possible to define its adjoint **L***, which is also a linear operator on $E$, characterized by the property that, for any tow vectors **X** and **Z** in $E$ the following equality holds between inner products

$$\langle \mathbf{X}, \mathbf{LZ} \rangle = \langle \mathbf{L}^{*}, \mathbf{X}, \mathbf{Z} \rangle.$$

In discretized problems, **X** and **Z** are vectors and **L** is a matrix, with **L***, being its transpose. The objective of variational assimilation is to find optimal solution to a given model that will best fit observed fields distributed over some space and time interval. One possible measure of the fit, the distance or cost function, $J$, consists of weighted squared norm of the difference between the observations at time $t$ and the corresponding components of a field variable.

Let us consider a system whose state at any time $t$ is defined by a state vector $Y(t)$ belonging to some space $\varepsilon$ and whose time evolution is described by the equation

$$\mathrm{d}Y/\mathrm{d}t = H(Y), \qquad \ldots (1)$$

where $H$ is regular function of $\varepsilon$ into itself. Assuming that observations $Y^{0}(t_1)$, $Y^{0}(t_2)$, ....., $Y_{0}(t_n)$ of the state of the system are available at times $t_1 < t_2 < .... < t_n$, we want to find a solution $Y(t)$ of (1) minimizing the functional

$$J[Y(t)] = \sum_{i=1}^{n} \langle Y(t_i) - Y^{0}(t_i), Y(t_i) - Y^{0}(t_i) \rangle, \qquad \ldots (2)$$

where $\langle , \rangle$ denotes an inner product defined over $\varepsilon$. The observations here are not exactly compatible with a solution of (1), therefore, $J$ is not equal to zero.

Any solution $Y)t)$ of (1) is uniquely defined by the specification of the initial condition $Y(t_1)$. The problem is now to find the initial condition $Y(t_1)$ such that the corresponding solution of (1) minimzes the distance function (2).

Given a solution $Y(t)$ of (1), the first order variation $\delta J$ resulting from a perturbation $\delta Y(t_1)$ of the initial condition is equal to

$$\delta J = 2 \sum_{i=1}^{n} \langle Y(t_i) - Y^{(i)}(t_i), \delta Y(t_i) \rangle , \qquad \ldots (3)$$

where the first-order variation $\delta Y(t_i)$ are themselves governed by the linear tangent equation

$$\mathrm{d}\delta Y / \mathrm{d}t = H'(t)\, \delta Y \qquad \ldots (4)$$

where $H'$ is the derivative of $H$ with respect to $Y$, evaluated at $Y$. System (4) is obtained by linearising the basic evolution equation (1) about the solution $Y(t)$. For any solution of (4), the value $\delta Y(t)$ at a given time $t$ depends linearly on the initial condition $\delta Y(t_1)$. For simple forward time integration scheme as an example, one can have

$$\delta Y_i = (I + \Delta t H'_{i-1}) \delta Y_{i-1} = (I + \Delta t H'_{i-1})(I + \Delta t H'_{i-2}) \ldots (I + \Delta t H'_1)\, \delta Y_1 \qquad \ldots (5)$$

Thus,

$$\delta J = 2 \sum_{i=1}^{n} \langle (I + \Delta t H'^{*}_1) \ldots (I + \Delta t H'^{*}_{i-1})(Y(t_i) - Y^{(i)}(t_i)), \delta Y_1 \rangle .$$

The summation in this expression is the gradient $\nabla_{Y1} J$. The $i$th in this series is obtained by a backward finite-difference integration of the adjoint equation

$$-\mathrm{d}\delta^* Y / \mathrm{d}t = H'^*(t)\, \delta^* Y$$

from time step $i$ to the initial step starting from

$$\delta^* Y = 2(Y_i - Y_i^{(i)}).$$

In summary, the gradient $\nabla J$ corresponding to some initial condition $Y(t_1)$ can be explicitly obtained by performing the following operations:

(1) Starting from $Y(t_1)$, integrate the basic equation (1) from $t_1$ to $t_n$. Store the solution at each time step $Y_i$ ($i = 1, \ldots, n$), which will make up coefficients of the operator $H'^*(t)$.

(2) Starting from $\delta^* Y(t_n) = 2(Y(t_n) - Y_0(t_n))$, integrate the adjoint equations backward in time from $t_n$ to $t_1$, the forcing term $2(Y(t_i) - Y^{(i)}(t_i))$ being added to the currently computed solution $\delta^*(Y(t_i)$ at each observation time $t_i$. The final result at time $t_1$ is the gradient $\nabla J$.

Examples of deriving $H'^*$ have been given by Tâlagrand and Courtier[4]. The adjoint code can be constructed directly from the model code in a simple way, which is also effective to easy error detection in adjoint code. Thus, the derivation of the adjoint model in both forms, the differential or the discrete form, is not required.

*The Adjoint Model*

The GCMs are rather long and complicated codes. The length and complexity of the corresponding adjoint codes is comparable to those of the direct models. One should write adjoints with a few basic principles in mind. The first principle is the principle of locality, i.e., whenever a local modification is made in the direct code, the corresponding modification must also be local in the adjoint code. Locality in turn requires readability, meaning that when a local modification is made in the direct code, it must be easy to locate the place where the corresponding modification is to be made in the adjoint code. This requires transparent notations, between the direct and adjoint codes, as regards subroutine and variable names, instruction labelling, etc.

These requirements for locality and readability clearly imply that the adjoint code must be developed directly from the basic direct code. They also imply that the components of the adjoint code must be in one-to-one correspondence with the components of the direct code: to each subroutine of the direct code, performing a well defined task, will correspond a subroutine of the adjoint code, performing the adjoint task. For example, a typical Fortran statement in the direct code is

$$X = AYZ + BZ^2,$$

where $A$ and $B$ are constants and $X$, $Y$, and $Z$ are variables. The corresponding Fortran statement in the tangent linear code is

$$X = A((aZ)Y + (aY)Z) + 2B(aZ)Z,$$

where $aY$ and $aZ$ are the values of $Y$ and $Z$ stored during the forward integration of the model, and $Y$, and $Z$ now represent the perturbation variables. The corresponding adjoint code will be

$$Z' = Z' + [2B(aZ) + A(aY)]X'$$

$$Y' = Y' + A(aZ)X$$

and $$X' = 0,$$

where $X'$, $Y'$ and $Z'$ represent the gradient of $J$ with respect to the direct code variables $X$, $Y$, and $Z$, respectively (*see* Appendix also). The above form of statements clearly shows that all adjoint variables must be set to zero in the adjoint code before they are used for the first time. The general rules which have just been demonstrated can be very automatically implemented on a Fortran code. It is seen that the tangent linear code is a necessary intermediary for developing the adjoint code. The linear tangent code is also extremely useful for checking the adjoint code. The only basic choice to make, when developing the adjoint code or the linear tangent code, is whether the quantities appearing in nonlinear terms in the direct code will have to be stored at the time of forward time integration, or will have to recalculated when needed. For the present study only the prognostic variables are stored during the forward run.

*Verification of the Adjoint Code*

The safest way to check the correctness of the adjoint code is to check the adjointness equality $\langle v, Lu\rangle = \langle L^*v, u\rangle$ between scalar products. The best way to check the tangent linear code is to verify that it leads to an exact first-order approximation for the output perturbation, i.e., to verify that the difference $\Delta v - \delta v$, where $\Delta v$ is the exact, finite difference, output perturbation computed by the tangent linear code, is of order $0(\delta u)$. The same method can be used for checking the exactness of a gradient produced by an adjoint code, i.e.

$$J(u+\alpha\delta u) - J(u) - \langle \alpha\delta u, \nabla J(u)\rangle = 0(\alpha^2), \qquad \ldots (6)$$

For values of $\alpha$ that are small but not too close to the machine zero, one should expect to obtain the required order (Table I) for the above equation (6). All routines of the adjoint code of the LMD GCM were tested both ways.

**Table I**

*Verification of relation* (6)

| Values of $\alpha$ | Order $(\alpha^2)$ |
|---|---|
| $10^{-1}$ | 0.206908445641E+05 |
| $10^{-2}$ | 0.206908835595E+03 |
| $10^{-3}$ | 0.206908874577E+01 |
| $10^{-4}$ | 0.206908878452E−01 |
| $10^{-5}$ | 0.206908878762E−03 |
| $10^{-6}$ | 0.206908877412E−05 |
| $10^{-7}$ | 0.206908863578E−07 |
| $10^{-8}$ | 0.206908975217E−09 |
| $10^{-9}$ | 0.206907448229E−11 |
| $10^{-10}$ | 0.206908363666E−13 |
| $10^{-11}$ | 0.207009968835E−15 |
| $10^{-12}$ | 0.207777719943E−17 |

*Distance Function*

For the test experiment described in this paper the distance function is defined as

$$J = \sum_{t=1}^{n} \sum_{ijl} \{ W_q (q_{ijlt} - q^0_{ijlt})^2 + \sum_{t=1}^{n} \sum_{ij} W_{ps} (p_{sijt} - p^0_{sijt})^2,$$

where $\sum$ and $\sum$ are the summations over all observation points, $\sum$ is the summation over time, and $W$'s are weights, and superscript 0 denotes observation, and $q = (u, v, h)$, here $h$ denotes potential enthalpy; $p_s$ is the area-weighted surface pressure, and $u$ and $v$ are covariant vector components of the horizontal wind. The variables, $i$ and $j$ are the horizontal indices, and $l$ is the vertical index

**Table II**

*Convergence of the distance function and the gradient with number of iterations*

| No. of iterations | normalized distance function ($J/J_0$) | normalized gradient |
|---|---|---|
| 0 | 1 | 1 |
| 1 | $1.43*10^{-2}$ | $1.2*10^{-1}$ |
| 2 | $1.15*10^{-5}$ | $3.3*10^{-3}$ |
| 3 | $3.55*10^{-7}$ | $5.9*10^{-4}$ |
| 4 | $5.41*10^{-9}$ | $7.1*10^{-5}$ |
| 5 | $1.86*10^{-10}$ | $1.2*10^{-5}$ |
| 6 | $1.73*10^{-11}$ | $3.6*10^{-6}$ |
| 7 | $9.23*10^{-12}$ | $3.0*10^{-6}$ |
| 8 | $6.12*10^{-13}$ | $7.0*10^{-7}$ |
| 9 | $5.41*10^{-14}$ | $2.1*10^{-7}$ |
| 10 | $3.67*10^{-15}$ | $5.4*10^{-8}$ |
| 11 | $4.28*10^{-16}$ | $2.2*10^{-8}$ |
| 12 | $2.58*10^{-17}$ | $4.7*10^{-9}$ |

The observation is available at all grid points and at each time step of the model. The distance function $J$ is non-dimensional.

*Descent Algorithms*

Once a numerical algorithm is available for computing $\nabla J$, a descent algorithms can be used in order to determine the value $Y_{\min}$ which minimizes $J$. Successive estimates $Y_n$ of $Y_{\min}$ are obtained through descent steps of the form

$$Y_{n+1} = Y_n - \rho_n D_n,$$

where, for each $n$, $D_n$ is a descent direction determined from the successive gradient $\nabla J_n$, $\nabla J_{n-1}$, ....., and $\rho_n$ is an appropriate scalar. If $D_n$ and $\rho_n$ are properly chosen, the sequence $Y_n$ will tend to $Y_{\min}$. Three classical descent algorithm are the steepest descent algorithms, the conjugate gradient algorithm and the quasi-Newton, or variable metric algorithm. These algorithms are described, and their properties compared, in Gill *et al.*

## Test Experiment and Result

As mentioned earlier that all routines of the adjoint code have been tested separately using linear tangent code and by computing the gradients. Besides this the whole system (the direct code, the adjoint code and the minimization procedure) was tested. The forecast model was run for only 5 steps from a specified initial condition and at each time step the field variables (only prognostic ones) were stored as observations. Starting from zero initial conditions for the adjoint model while the $\nabla J$ were inserted whenever an analysis time was reached and this recovered the original initial field through the iterative procedure: that is, the distance function as well as the gradient should reduce to zero.

The test run was conducted with FGGE data with low resolution. The run generates the observation data at each time step. The adjoint integration was

started from final step to initial time. The conjugate gradient method was employed to achieve optimal solution. Table II shows the convergence of the distance function and the gradient with number of iterations. The distance function and the gradient have reduced to several orders of magnitude in only 12 iterations.

## Summary

We have completed writing the adjoint of the dynamics part of the LMD GCM. The correctness of the adjoint model has been tested by the methods discussed in this paper. The same weight is used for each variable at each level. Model generated data is used as observation for the present study. Conjugate gradient method is used to attain the optimal solution of the test experiment. The test run with realistic initial conditions seems to be satisfactory and encouraging. A reasonable reduction in the distance function was achieved, and the reproduction of initial field is quite satisfactory (the root mean square of the field variables is the order of $10^{-6}$ units). Future work involves the addition of 'physics' in the adjoint code and perform some sensitivity experiments using a moderate resolution GCM.

## Acknowledgements

The first author is grateful to Professor R Sadourny, Director LMD for providing an opportunity to work at his Laboratory, and to the Department of Science and Technology, Government of India. This work is undertaken under the Indo-French Centre for Promotion of Advanced Research/Centre Franco-Indian Pour la Promotion de Recherche Avancée (Project No. 711-1).

## References

1 F X Le Dimet, O Talagrand *Tellus* **38A** (1986) 97-110
2 J M Lewis and J C Derber *Tellus* **39A** (1985) 309-322
3 R N Hoffmann *Mon Wea Rev* **114** (1980) 388-397
4 O Talagrand and P Courtier *Q J R met Soc* **113** (1987) 1311-1328
5 P Courtier and O Talagrand *Q J R met Soc* **113** (1987) 1329-1347
6 Winston C Chao and Chan Lang-Ping *Mon Wea Rev* **120** (1992) 1661-1673
7 I M Navon, X Zou, J Derber and J Sela *Mon Wea Rev* **120** (1992) 1433-1446
8 D G Cacuci *J math Phys* **35A** (1981) 2794-2802
9 M C G Hall and D G Cacuci *J atmos Sci* **40** (1983) 2537-2546
10 M C G Hall, D G Cacuci and M E Schlesinger *J atmos Sci* **30** (1982) 2038-2050
11 B Urban *Maximum Error Growth in Simple Meteorological Models Tech* Rep Ecole Normale de la Meteorologie, Toulouse (1985) (French)
12 J F Lacarra and O Talagrand *Tellus* **40A** (1988) 81-95
13 V V Penenko and N N Obraztsov *Sov met Hydrol* **11** (1976) (*Trans. Russian*)
14 A C Lorenc *Q J R met Soc* **114** (1988) 205-240
15 P Courtier and O Talagrand *Tellus* **42A** (1990) 531-549
16 R Sadourny and K Laval *New Perspectives in Climate Modelling* (*Eds.* N Berger and C Nicolis) No 16 In *Elsevier Sci* (1984) 173-197
17 P E Gill, W Murray and M H Wright *Practical Optimizatioin* Academic Press (1981) 401 pp

*Appendix*

We present here one of the several routines used in this study. In the DIRECT CODE, pbarx, for example, is a variable. The same is perturbation in the LINEAR TANGENT CODE, and in the ADJOINT CODE it is a gradient. The variable apbarx is the value of pbarx stored during the forward integration of the model.

---

***DIRECT CODE or FORWARD CODE***

---

```
SUBROUTINE flumass (pbarx, pbary, vcont, ucont, pbaru, pbarv)
*CALL paramet
DIMENSION pbarx (iplijmpl), pbary (ipljm), vcont (ipljm, max),
*ucont (ipljmpl, max), pbaru (ipljmpl, max), pbarv (ipljm, max)
*CALL const
COMMON/scratch/apbarun (iipl), apbarus (iipl), aire4n (iipl),
*aire4s (iipl)
EXTERNAL SSUM
DO 5l = l, max
DO 1 jij = iip2, ipljm
pbaru (ij, l) = pbarx (ij)*ucont(ij, l)
1 CONTINUE
DO 3 ij = l, ipljm
pbarv (ij, l) = pbary (ij)*vcont (ij, l)
3 CONTINUE
5 CONTINUE
sairen = SSUM (iim, aire (l), l)
saireun = SSUM (iim, aireu (l), l)
saires = SSUM (iim, aire (ipljm + 1), l)
saireus = SSUM (iim, aireu (ipljm + 1), l)
DO 20 l = 1, max
ctn = SSUM (iim, pbarv (1,l), 1)/sairen
cts = SSUM (iim, pbarv (ipljmil + 1, l), 1)/saires
pbaru (1, l) = pbarv (1, l)-ctn*aire(1)
pbaru (ipljm + 1, l) = − pbarv (ipljmil + 1, l) + cts*aire (ipljm + 1)
DO 11 i = 2, iim
pbaru (i, l) = pbaru (i − 1, l) +
pbarv (i, l) − ctn*aire(i)
pbaru (i + ipljm, l) = pbaru (i + ipljm − 1, l)
pbarv (i + ipljmil, 1) + cts*aire (i + ipljm)
11 CONTINUE
DO 12i = 1, iim
apbarun (i) = aireu (i)*pbaru (i, l)
apbarus (i) = aireu (i + ipljm)*pbaru (i + ipljm, l)
12 CONTINUE
ctn0 = − SSUM (iim, apbarun, 1)/saireum
cts0 = -SSUM (iim.apbarus, 1)/saireus
DO 14i = 1, iim
pbaru (i, 1) = 2, *(pbaru(i, l) + ctn0)
pbaru (i + ipljm, l) = 2, *(pbaru (i + ipljm, l) + cts0)
```

```
14 CONTINUE
pbaru (iipl, 1) = pbaru (1, 1)
pbaru (ipljmpl, 1) = pbaru (ipljm + 1, 1)'
20 CONTINUE
RETURN
END
```

---

***LINEAR TANGENT CODE***

---

```
SUBROUTINE tflumass (pbarx, pbary, vcont, pbaru, pbarv,
*apbarx, apbary, avcont, aucont, apbaru, apbarv)
*CALL paramet
DIMENSION pbarx (ipljmpl), pbary (ipljm), vcont (ipljm, max),
*ucont (ipljmpl, max), pbaru (ipljmpl, max), pbarv (ipljm, max)
DIMENSION apbarx (ipljmpl), apbary (ipljm), avcont (ipljm, max),
*aucont (ipljmpl, max), apbaru (ipljmpl, max) apbarv (ipljm, max)
*CALL const
COMMON/safluma/apbarun (iipl), apbarus (iipl), aire4n (iipn),
*aire4s (iipl)
EXTERNAL SSUM
DO 5l = 1, max
DO 1 ij = iip2, ipljm
pbaru (ij, l) = pbarx (ij) *aucont(ij, l) + apbarx (ij) *ucont (ij, l)
1 CONTINUE
DO 3 ij = 1, ipljm
pbarv (ij, l) = pbary (ij) *avcont (ij, l) + apbary (ij) *vcont (ij, l)
3 CONTINUE
5 CONTINUE
sairen = SSUM (iim, aire (1), l)
saireun = SSUM (iim, aireu (1), l)
saires = SSUM (iim, aire (ipljm + 1), l)
saireus = SSUM (iim, aireu (ipljm + 1), l)
DO 20l = 1, max
ctn = SSUM (iim, pbarv (1, l), l)/ sairen
cts = SSUM (iim, pbarv (ipljmil − 1, l), l)/saires
pbaru (1, l) = pbarv (1, l) − ctn*aire (1)
pbaru (ipljm + 1, l) = − pbarv (ipljmil + 1, l) + cts *aire (ipljm + 1)
DO 11i = 2, iim
pbaru (i, l) = pbaru (i − 1, l) +
pbarv (i, l) − ctn *aire (i)
pbaru (i + ipljm, l) = pbaru (i + ipljm-1, l),
pbarv (i + ipljmil, l) + cts *aire (i + ipljm)
11 CONTINUE
DO 12i = 1, iim
abparun (i) = aireu (I) *pbaru (i, l)
apbarus (i) = aireu (i + ipljm) *pbaru (i + ipljm, l)
12 CONTINUE
ctn0 = − SSUM (iim, apbarun, l)/saireun
cts0 = − SSUM (iim, apbarus, l)/saireus
```

```
DO 14i = 1, iim
pbaru (i, l) = 2, *(pbaru (i, l) + ctn0)
pbaru (i + ipljm, l) = 2, *(pbaru (i + ipljm, l) + cts0)
14 CONTINUE
pbaru (iipl, l) = pbaru (1, l)
pbaru (ipljmpl, l) = pbaru (ipljm + 1, l)
20 CONTINUE
RETURN
END
```

---

***ADJOINT CODE***

---

```
SUBROUTINE aflumass (pbary, vcont, ucont, pbaru, pbarv,
*apbarx, apbary, avcont, aucont, apbaru, apbarv)
*CALL paramet
DIMENSION pbarx (ipljmpl), pbary (ipljm), vcont (ipljm, max),
*ucont (ipljmpl, max), pbaru (ipljmpl, max), pbarv (ipljm. max)
DIMENSION apbarx (ipljmpl), apbary (ipljm), avcont (ipljm, max),
*aucont (ipljmpl max), apbaru (ipljmpl, max), apbarv (ipljm, max)
*CALL const
COMMON/safluma/apbarun (iipl), apbarus (iipl), aire4n (iipl),
*airc4s (iipl)
EXTERNAL SSUM
DO 45i = 1, iipl
apbarun (i) = 0.
45 apbarus (i) = 0.
sairen = SSUM (iim, aire (1), l)
saireun = SSUM (iim, aireu (1), l)
saires = SSUM (iim, aire (ipljm + 1), l)
saireus = SSUM (iim, aireu (ipljm + 1), l)
cts0 = 0.
ctn0 = 0.
DO 20l = max, 1, − 1
pbaru (1, l) = pbaru (iipl, l)
pbaru (ipljm + 1, l) = pbaru (ipljmpl, l)
DO 14i = iim, 1, − 1
cts0 = cts0 + 2, *pbaru (i + ipljm, l)
ctn0 = ctn0 + 2, *pbaru (i, l)
pbaru (i + ipljm, l) = 2, *pbaru (i + ipljm, l)
pbaru (i, l) = 2, *pbaru (i, l)
14 CONTINUE
DO 46i = 1, iim
apbarus (i) = apbarus (i)-cts0/saireus
apbarun (i) = apbarun (i)-.ctn0/sairen
46 CONTINUE
cts0 = 0.
ctn0 = 0.
DO 12i = iim, 1, −1
pbaru (i + ipljm, l) = pbaru (i + ipljm, l) + aireu (i + ipljm) *apbarus (i)
```

```
pbaru (i, l) = pbaru (i, l) + aireu(i) *apbarun (i)
apbarun (i) = 0.
apbarus (i) = 0.
12 CONTINUE
cts = 0.
cts = 0.
DO 11i = iim, 2, - 1
cts = cts + aire (i + ipljm) *pbaru (i + ipljm, l)
pbarv (i + ipljmil, l) = pbarv (i + ipljmil, l) - pbaru (i + ipljm, l)
pbaru (i + ipljm-1, l) = pbaru (i + ipljm-1, l) + pbaru (i + ipljm, l)
pbaru (i + ipljm, l) = 0.
ctn = ctn-aire (i)*pbaru (i, l)
pbarv (i, l) = pbarv (i, l) + pbaru (i, l)
pbaru (i-1, l) = pbaru (i-1, l)+ pbaru (i, l)
pbaru (i, l) = 0.
11 CONTINUE
cts = cts + aire (ipljm + 1) *pbaru(ipljm + 1, l)
pbarv (ipljml + 1, l) = pbarv(ipljmil + 1, l)-pbaru (ipljm + 1, l)
pbaru (ipljm + 1, l) = 0
ctn = ctn-aire (1) *pbaru (1, l)
pbarv (1, l) = pbarv (1, l) + pbaru (1, l)
pbaru (1, l) = 0.
DO 21i = iim, 1, - 1
pbarv (ipljmil + i, l) = pbarv (ipljmil + i, l) + cts/saires
pbarv (i, l) = pbarv (i, l) + ctn/sairen
21 CONTINUE
cts = 0.
ctn = 0.
20 CONTINUE
DO51 = max, 1, - 1
DO 3 ij = 1, ipljm
vcont (ij, l) = vcont (ij, l) + apbary (ij)*pbarv (ij, l)
pbary (ij) = pbary (ij) + avcont (ij, l) *pbarv (ij, l)
pbarv (ij, l) = 0.
3 CONTINUE
DO 1 ij = iip 2, ipljm
ucont (ij, l) = ucont (ij, l) + apbarx (ij) *pbaru (ij, l)
pbarx (ij) = pbarx (ij) + aucont (ij, l) *pbaru (ij, l)
pbaru (ij, l) = 0.
1 CONTINUE
5 CONTINUE
RETURN
END
```