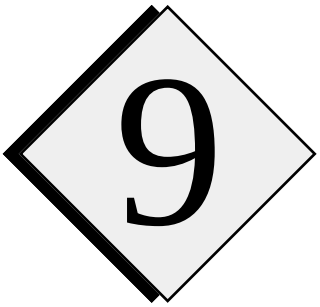


Thermodynamic Recovery and Forward Assimilation

9.1. Introduction

This chapter describes a forward four-dimensional data assimilation package developed for ARPS. A key component of this package is a variational technique based on the Liou/Gal-Chen procedure (Liou *et al.*, 1991), which incorporates the radial wind component from a Doppler radar into a numerical model. The algorithm ingests the radial wind component, adjusts the wind field so that it satisfies the anelastic mass conservation equation while minimizing the difference between the "first guess" or "background" winds and the adjusted winds. The adjusted three-dimensional wind field and its time derivative (based on data interpolated to three successive model time levels) are used to derive the pressure and temperature fields via a thermodynamic recovery procedure. ARPS is then integrated forward in time until radar data are next available, at which time the predicted ARPS fields can be used as background fields for another cycle of insertion, adjustment, thermodynamic recovery and prediction. This forward assimilation package has been extensively tested with analytic flow fields and with pseudo-observations from a simulated supercell storm, and is currently being tested with real radar data.

The package includes options to perform multiple wind adjustments, multiple thermodynamic retrievals, and wind adjustments with or without thermodynamic retrievals (and vice versa). In addition, a simple direct insertion may be performed where the model's radial velocity component is overwritten with the observed radial velocity component and the crossbeam winds remain unchanged. The algorithm does not, at present, make provision for the stretched grid option or terrain.

It should be stressed that the success of this package in any application will likely depend upon the accuracy of the background winds used in the

procedure, since the adjusted winds are forced to a close agreement with them. This algorithm is more an ingest (or assimilation) procedure than a single-Doppler velocity retrieval algorithm. However, single-Doppler velocity retrieval algorithm(s) are being developed which may be included in the background fields.

9.2. Thermodynamic Recovery

The thermodynamic recovery scheme described herein is a slightly modified version of the procedure derived by Gal-Chen (1978) and Hane and Scott (1978). The recovery can be used as a "stand alone" algorithm or as an essential element of the forward assimilation package. In the Gal-Chen/Hane scheme, velocity observations are used to evaluate the advection, turbulence, Coriolis force terms, *etc.*, in the model's equations of motion. The pressure field is then determined up to an arbitrary function of height as a least squares solution of the horizontal equations of motion with the known forcings. Finally, with the pressure known, the temperature is recovered from the vertical equation of motion. This conventional scheme is modified to take into account the presence of pressure in the buoyancy term of the vertical equation of motion.

In ARPS the relevant dynamic equations can be written as:

$$\frac{\partial p'}{\partial x} = -\frac{\partial \bar{\rho} u}{\partial t} - \bar{\rho}\,\vec{V}\cdot\nabla u + \bar{\rho} f v + F_x \tag{9.2.1}$$

$$\frac{\partial p'}{\partial y} = -\frac{\partial \bar{\rho} v}{\partial t} - \bar{\rho}\,\vec{V}\cdot\nabla v - \bar{\rho} f u + F_y \tag{9.2.2}$$

$$\frac{\partial p'}{\partial z} = -\frac{\partial \bar{\rho} w}{\partial t} - \bar{\rho}\vec{V}\cdot\nabla w + \bar{\rho} g\left[\frac{\theta'}{\bar{\theta}} - \frac{p'}{\bar{\gamma p}} - q_c - q_r + 0.608 q_v'\right] + F_z \tag{9.2.3}$$

where primed quantities denote perturbations, barred quantities are base-state variables and other symbols have their conventional meanings. It can be noted that if an arbitrary function of height is added to the perturbation pressure then (9.2.1) and (9.2.2) remain unchanged while (9.2.3) can be satisfied **if the buoyancy allows.** Thus, the best a thermodynamic recovery based on (9.2.1) - (9.2.3) can do is recover pressure and temperature fields up to an arbitrary function of height. In practice this element of non-uniqueness might be circumvented by making use of an updated sounding (if available) or the

model's mean pressure or temperature fields. In the current version of the code, horizontal averages of the "background fields" are used.

Gal-Chen (1978) noted that if real data is used to evaluate the forcing terms in (9.2.1) and (9.2.2), the y-derivative of $\partial p/\partial x$ may not be equal to the x-derivative of $\partial p/\partial y$. Because of this incompatibility, an exact solution does not exist and the best one can hope for is an approximate solution. The Gal-Chen scheme seeks a least squares error solution, that is, the solution that minimizes the cost (error) functional J defined by:

$$J = \int_{x_l, y_l}^{x_r, y_r} \int \left[\left(\frac{\partial p'}{\partial x} - A \right)^2 + \left(\frac{\partial p'}{\partial y} - B \right)^2 \right] dy\, dx$$

Here A and B denote the right-hand sides of (1) and (2), respectively,

$$A = - \frac{\partial \bar{\rho} u}{\partial t} - \bar{\rho} \vec{V} \cdot \nabla u + \bar{\rho} f v + F_x$$

$$B = - \frac{\partial \bar{\rho} v}{\partial t} - \bar{\rho} \vec{V} \cdot \nabla v - \bar{\rho} f u + F_y$$

The integration domain is the entire horizontal flow domain from the left hand boundaries of the model, x_l, y_l to the right hand boundaries of the model, x_r, y_r.

Setting the variation of J to zero following Mathews and Walker, (1970) results in:

$$\delta J = 2 \int_{x_l, y_l}^{x_r, y_r} \int \left[\left(\frac{\partial p'}{\partial x} - A \right) \frac{\partial (\delta p')}{\partial x} + \left(\frac{\partial p'}{\partial y} - B \right) \frac{\partial (\delta p')}{\partial y} \right] dy\, dx = 0$$

After integrating by parts we obtain:

$$\int_{x_l,y_l}^{x_r,y_r}\int \left[\left(\frac{\partial^2 p'}{\partial x^2} + \frac{\partial^2 p'}{\partial y^2} - \frac{\partial A}{\partial x} - \frac{\partial B}{\partial y}\right)\right] \delta p'\, dy\, dx - \int_{y_l}^{y_r}\left[\frac{\partial p'}{\partial x} - A\right]_{x_l}^{x_r} \delta p'\, dy$$

$$- \int_{x_l}^{x_r}\left[\frac{\partial p'}{\partial y} - B\right]_{y_l}^{y_r} \delta p'\, dx = 0 .$$

Since the variation $\delta p'$ *i*s arbitrary, the integrands must be identically zero and we obtain a Poisson equation for p':

$$\frac{\partial^2 p'}{\partial x^2} + \frac{\partial^2 p'}{\partial y^2} = \frac{\partial A}{\partial x} + \frac{\partial B}{\partial y} \tag{9.2.4}$$

subject to the Neumann conditions:

$$\text{On } x\text{-boundaries: } \frac{\partial p'}{\partial x} = A. \qquad \text{On } y\text{-boundaries: } \frac{\partial p'}{\partial y} = B. \tag{9.2.5}$$

Since solutions of Poisson's equation with Neumann boundary conditions are not unique, the solution for p' on each horizontal surface is determined only up to an arbitrary constant. This is equivalent to saying that p' is determined on the whole flow domain up to an arbitrary function of height, $f(z)$ (as we anticipated). Changing notation, we write the computed (non-unique) solution of the Poisson equation as pc' (rather than the p') and express the final perturbation pressure p' as:

$$p' = pc' + f(z) \tag{9.2.6}$$

To estimate the $f(z)$ function, substitute (9.2.6) into (9.2.3) and take horizontal averages (denoted by brackets ‹ ›) to get:

$$\frac{df}{dz} + \frac{\bar{\rho} g}{\gamma \bar{p}} f + \left[\frac{d\langle pc'\rangle}{dz} + \frac{\bar{\rho} g}{\gamma \bar{p}}\langle pc'\rangle - \langle C\rangle - \bar{\rho} g\left\langle \frac{\theta'}{\bar{\theta}} - q_c - q_r + 0.608 q_v' \right\rangle\right] = 0 \tag{9.2.7}$$

Here C includes the known velocity-dependent forcing terms in the vertical equation of motion:

$$C = -\frac{\partial \bar{\rho} w}{\partial t} - \bar{\rho}\,\vec{V}\cdot\nabla w + F_z$$

It should be noted that since we don't know the horizontal average perturbation potential temperature, we can't legally obtain *f(z)*. Our approach is to replace the (unknown) average perturbation potential temperature by the average perturbation potential temperature from a previous ARPS run or from background fields (which may or may not know about the ARPS fields). Another approach is to make use of an updated sounding. Either case has its drawbacks but, in both cases, the extra information allows us to compute all the coefficients in the equation and we can proceed with the solution for *f(z)*. Writing (9.2.7) as

$$\frac{df}{dz} + a(z)\,f + b(z) = 0,$$

where

$$a \equiv \frac{\bar{\rho} g}{\gamma \bar{p}}, \qquad b \equiv \frac{d\langle pc' \rangle}{dz} + \frac{\bar{\rho} g}{\gamma \bar{p}}\langle pc' \rangle - \frac{\bar{\rho} g}{\bar{\theta}}\langle \theta' \rangle - \langle C \rangle + \bar{\rho} g \langle q_c + q_r \rangle,$$

we see that the equation is a first order linear ordinary differential equation, the solution of which is standard (*cf.* Braun, 1975):

$$f(z) = \exp\left(-\int_{z_0}^{z} a(z')dz' \right) \left[f(z_0) - \int_{z_0}^{z} b(z') \exp\left(\int_{z_0}^{z'} a(z'')dz'' \right) dz' \right] \qquad (9.2.8)$$

Here $z=z_0$ is the lowest level for which we have computed *pc'*. Now we need to pin down the constant of integration, $f(z_0)$, appearing in (9.2.8). Taking the horizontal average of (9.2.6) at $z=z_0$ shows that $f(z_0) = \langle p'(z_0) \rangle - \langle pc'(z_0) \rangle$ so that if we know the average perturbation pressure at $z=z_0$, we can get $f(z_0)$. In practice, we will approximate this unknown average perturbation pressure at $z=z_0$ by the average background perturbation pressure at $z=z_0$ or an observation (if available). So, now that we have *f(z)*, we can get p' from (9.2.6) and θ' follows immediately (as a residual) from (9.2.3).

9.3. Numerical Solution of the Poisson Equation

The Poisson equation (9.2.4) is solved with successive over-relaxation (SOR) and an alternating direction implicit (ADI) technique based on the Thomas algorithm (Roache, 1982). Readers who are not interested in the details of the solution technique can skip the rest of this section.

We discretize the Poisson equation on ARPS grid as:

$$\frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{(dx)^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{(dy)^2} = \phi_{i,j} \tag{9.3.1}$$

where

$$\phi_{i,j} = \frac{A_{i+1,j} - A_{i,j}}{dx} + \frac{B_{i,j+1} - B_{i,j}}{dy}$$

Now rearrange (9.3.1) so that only p terms evaluated on the j line are kept on the left hand side. The resulting equation is of the form:

$$a\, p_{i+1,j} + b\, p_{i,j} + c\, p_{i-1,j} = d_{i,j} \tag{9.3.2}$$

where

$$a = c = \frac{1}{(dx)^2}, \quad b = -2\left(\frac{1}{(dx)^2} + \frac{1}{(dy)^2}\right), \quad d_{i,j} = \phi_{i,j} - \frac{\left(p_{i,j+1} + p_{i,j-1}\right)}{(dy)^2}\ .$$

We can also rearrange (9.3.1) so that all the p terms evaluated on the i line are kept on the left hand side whereas all other terms get thrown over onto the right hand side. In this case we get:

$$a\, p_{i,j+1} + b\, p_{i,j} + c\, p_{i,j-1} = d_{i,j} \tag{9.3.3}$$

where

$$a = c = \frac{1}{(dy)^2}, \qquad b = -2\left(\frac{1}{(dx)^2} + \frac{1}{(dy)^2}\right),$$

$$d_{i,j} = \phi_{i,j} - \frac{\left(p_{i+1,j} + p_{i-1,j}\right)}{\left(dx\right)^2}.$$

The tridiagonal matrix equations (9.3.2) and (9.3.3) are solved with the Thomas algorithm. Since the right hand sides of (9.3.2) and (9.3.3) each contain unknown terms, we iterate between the solutions of the tridiagonal equations using SOR.

We now describe the solution of (9.3.2) with the Thomas algorithm. First postulate the existence of a recursive solution of (9.3.2) of the form:

$$p_{i-1} = \alpha_i p_i + \beta_i \tag{9.3.4}$$

where, for convenience, the *j*-index has been suppressed. Eliminating p_{i-1} between (9.3.2) and (9.3.4) and rearranging results in:

$$p_i = -\left(\frac{a}{b + c\alpha_i}\right) p_{i+1} + \frac{d_i - c\beta_i}{b + c\alpha_i}$$

Or, after reducing each of the indices in this equation by one:

$$p_{i-1} = -\left(\frac{a}{b + c\alpha_{i-1}}\right) p_i + \frac{d_{i-1} - c\beta_{i-1}}{b + c\alpha_{i-1}}$$

Comparing this new expression for p_{i-1} with (9.3.4) gives recursion formulae for α and β:

$$\alpha_i = -\frac{a}{b + c\alpha_{i-1}}, \qquad \beta_i = \frac{d_{i-1} - c\beta_{i-1}}{b + c\alpha_{i-1}}. \tag{9.3.5}$$

Boundary conditions are required to apply the algorithm. In our case we impose the Neumann conditions (9.2.5) on the top and bottom. The lower condition, $(p_2 - p_1)/dx = A_2$, can be written as $p_1 = p_2 - A_2 dx$. Evaluating (9.3.4) at $i=2$ yields $p_1 = \alpha_2 p_2 + \beta_2$. Comparing these two expressions for p_1 shows that:

$$\alpha_2 = 1, \qquad \beta_2 = -A_2\, dx \tag{9.3.6}$$

Similarly, the upper condition of (9.2.5) can be written as $p_{nx-2} = p_{nx-1} - A_{nx-1}\, dx$ and compared with (9.3.4) evaluated at $i=nx-1$ to get:

$$p_{nx-1} = \frac{A_{nx-1}\, dx + \beta_{nx-1}}{1 - \alpha_{nx-1}} \tag{9.3.7}$$

The solution algorithm involves first sweeping upwards in the increasing *i* direction to get the α and β coefficients from (9.3.5). Equation (9.3.6) is used to start this upwards sweep. Then we sweep downwards in the decreasing *i* direction to get *p* from (9.3.4). Equation (9.3.7) is required to start this downward sweep.

Once (9.3.2) is solved, the Thomas algorithm is applied to (9.3.3) with the left hand side updated with the solution of (9.3.2). We then iterate (using over-relaxation) back and forth between (9.3.2) and (9.3.3) until the solution converges to a desired tolerance. Since the solution of the Poisson equation with Neumann boundary conditions is not unique, we subtract off the midpoint value of *p* at every iteration to avoid a potential drift which could affect the convergence of the iterates.

9.4. Evaluation of the Local Derivative Terms

The time derivatives of the velocity field required by the thermodynamic recovery algorithm are calculated with data at three observation times, t_1, t_2 and t_3, where $t_1 < t_2 < t_3$ and the intervals $t_3 - t_2$ and $t_2 - t_1$ need not be equal to each other. To discretize the time derivative at the thermodynamic recovery time, we interpolate quadratically in time data at the 3 observation time levels to the three consecutive model time levels centered on the recovery time and then perform a centered-in-time discretization of the time derivative on the model time steps. The reader not interested in the details of this quadratic interpolation can skip the rest of this section.

Suppose we know a function of time, *f(t)*, at three time levels, t_1, t_2 and t_3, where $t_1 < t_2 < t_3$. We want to interpolate *f(t)* to an arbitrary time *t* between t_1 and t_3 by fitting *f(t)* to a quadratic polynomial, that is,

$$f(t) = \alpha + \beta t + \gamma t^2. \tag{9.4.1}$$

Toward this end, set $t = t_1$, t_2 and t_3 successively in (9.4.1). The result is a set of three equations in the three unknowns α, β and γ:

$$f(t_1) = \alpha + \beta t_1 + \gamma t_1^2, \tag{9.4.2}$$

$$f(t_2) = \alpha + \beta t_2 + \gamma t_2^2, \tag{9.4.3}$$

$$f(t_3) = \alpha + \beta t_3 + \gamma t_3^2. \tag{9.4.4}$$

These equations can be manipulated to obtain expressions for α, β and γ, and the final result for *f(t)* expressed as

$$f(t) = f(t_1)\frac{(t_3 - t)(t_2 - t)}{(t_3 - t_1)(t_2 - t_1)} + f(t_2)\frac{(t - t_1)(t_3 - t)}{(t_2 - t_1)(t_3 - t_2)} + f(t_3)\frac{(t - t_2)(t - t_1)}{(t_3 - t_2)(t_3 - t_1)}. \tag{9.4.5}$$

This is the Lagrange three-point interpolation formula with unevenly spaced abscissas.

9.5. Variational Wind Adjustment with Single-Doppler Data

As described above, in order to recover the pressure and temperature fields we must know both the three-dimensional wind and it's time tendency. Unfortunately, if only one Doppler radar is available then we only have direct measurements of one wind component, the radial wind component. Furthermore, the ARPS model is formulated on a Cartesian grid and the projection of the radial velocity onto its Cartesian counterparts will generally only represent some portion of each of the three Cartesian components. This leaves each wind component only partially specified. The modifications to the Gal-Chen/Liou scheme, the details of which are presented in this section, address the aforementioned difficulties by incorporating the radial velocity directly on ARPS grid and adjusting the cross-beam wind components subject to several physical constraints.

Assuming that the radial wind component is known, we will apply the variational method to adjust the unknown winds (*u*, *v*, and *w*) so that the deviation between the adjusted winds and the first guess (or model) fields is minimized throughout the analysis volume. This adjustment will also enforce two strong constraints, mass conservation:

$$\frac{\partial \bar{\rho} u^a}{\partial x} + \frac{\partial \bar{\rho} v^a}{\partial y} + \frac{\partial \bar{\rho} w^a}{\partial z} = 0 , \tag{9.5.1}$$

and the geometrical constraint that the radial component of the adjusted velocity is equal to that of the observed radial velocity (approximating the radar beam path as a straight line), *i.e.*,

$$u^a x + v^a y + w^a z = r v_r^o \,. \tag{9.5.2}$$

Here the superscript '*a*' denotes an adjusted wind component, v_r^o is the observed radial velocity component, and $\bar{\rho}$ is the height-dependent base-state density. The *x*, *y*, *z* Cartesian coordinates are relative to the radar site.

Thus, we seek to minimize the cost functional, *J*, defined by:

$$J = \iiint \Big[(\bar{\rho}\, u^a - \bar{\rho}\, u^m)^2 + (\bar{\rho}\, v^a - \bar{\rho}\, v^m)^2 + (\bar{\rho}\, w^a - \bar{\rho}\, w^m)^2 +$$

$$\lambda_1 \bar{\rho}\, (u^a x + v^a y + w^a z - r v_r^o) + \lambda_2 \left(\frac{\partial \bar{\rho}\, u^a}{\partial x} + \frac{\partial \bar{\rho}\, v^a}{\partial y} + \frac{\partial \bar{\rho}\, w^a}{\partial z} \right) \Big] dV \tag{9.5.3}$$

where the superscript '*m*' refers to the first guess or model fields, and λ_1, λ_2 are the Lagrange multipliers.

Taking the variation of (9.5.3) with respect to u^a and setting it equal to zero yields:

$$\delta J = \iiint \left[2\,(U^a - U^m)\,\delta U^a + \lambda_1 x \delta U^a + \lambda_2 \frac{\partial \delta U^a}{\partial x} \right] dV = 0\,, \tag{9.5.4}$$

where $U^a \equiv \bar{\rho}\, u^a$. The third term in the integral can be integrated by parts, *i.e.*,

$$\iiint \lambda_2 \frac{\partial \delta\, U^a}{\partial x}\, dV = \iint \left(\lambda_2 \delta U^a \right) \Big|_{x_L}^{x_R} dy\, dz \; - \iiint \delta U^a \frac{\partial \lambda_2}{\partial x}\, dV, \tag{9.5.5}$$

where x_L and x_R denote the left and right sides of the cube (*i.e.*, the *y*-*z* faces). Note that the vanishing of the first term on the right hand side determines the *x*-boundary conditions of the variational problem. This term vanishes if either δU^a or λ_2 equals zero on the *x* boundaries. Assuming that the latter Dirichlet

condition holds (*i.e.*, we do not enforce the original U^a on the lateral boundary), (9.5.5) becomes:

$$\int\int\int \left[2(U^a - U^m) + \lambda_1 x + \frac{\partial \lambda_2}{\partial x} \right] \delta U^a dV = 0. \tag{9.5.6}$$

Since δU^a is arbitrary, the integrand must be identically zero, *i.e.*,

$$2\bar{\rho}(u^a - u^m) + \lambda_1 x + \frac{\partial \lambda_2}{\partial x} = 0 \, . \tag{9.5.7}$$

Equation (9.5.7) is one of the Euler-Lagrange equations for the variational problem given by the functional in (9.5.3). We can also take the variation of (9.5.3) with respect to v^a and, proceeding as above, we choose $\lambda_2 = 0$ on the y boundaries. This yields a second Euler-Lagrange equation, namely:

$$2\bar{\rho}(v^a - v^m) + \lambda_1 y + \frac{\partial \lambda_2}{\partial y} = 0 \, . \tag{9.5.8}$$

When taking the variation of (9.5.3) with respect to w^a we set δw^a to zero on the upper and lower boundaries. This implies that we are not changing the model's boundary conditions on z. This is desirable since we do not want to violate the impermeability condition at ground level. The third Euler-Lagrange equation is:

$$2\bar{\rho}(w^a - w^m) + \lambda_1 z + \frac{\partial \lambda_2}{\partial z} = 0 \, . \tag{9.5.9}$$

The variation of (9.5.3) with respect to λ_1 and λ_2 yields our original equations (9.5.1) and (9.5.2). Equations (9.5.1) and (9.5.2) and (9.5.7)-(9.5.9) represent a system of five equations and five unknowns (u^a, v^a, w^a, λ_1, and λ_2).

A single equation in λ_2 is now derived. Differentiate (9.5.7) with respect to x, (9.5.8) with respect to y, and (9.5.9) with respect to z, apply (9.5.1), and add the resulting equations to obtain an equation for λ_2 in terms of λ_1:

$$\nabla^2 \lambda_2 = -2\nabla \cdot \bar{\rho} \vec{V}^m + \nabla \cdot \vec{r} \lambda_1 \, , \tag{9.5.10}$$

where $\vec{r}$ is the position vector (directed from the radar). We also obtain an equation for λ_1 in terms of λ_2 by multiplying equations (9.5.7)-(9.5.9) by *x*, *y*, and *z* respectively and adding them together,

$$\lambda_1 = \frac{2}{r^2}\left[\bar{\rho}(rv_r^m - rv_r^o) + \frac{1}{2}\,\vec{r}\cdot\nabla\lambda_2\right]. \tag{9.5.11}$$

By inserting (9.5.11) into (9.5.10) we arrive at a Poisson equation for λ_2:

$$\nabla^2_{\phi,\theta}\lambda_2 = -2\nabla\cdot\bar{\rho}\,\vec{V}^m - \frac{2}{r^2}\frac{\partial}{\partial r}\left[r^2\bar{\rho}(v_r^o - v_r^m)\right]. \tag{9.5.12}$$

This equation is analogous to equation (2.3.5) in Liou's thesis, and represents the extension of Liou's technique to the ingestion of real single-Doppler radar data (rather than a Cartesian component). Since the intended application of this technique is to forward data assimilation in the ARPS model, we re-express (9.5.12) in Cartesian geometry retaining v_r as the known variable:

$$\begin{aligned}
&\frac{y^2+z^2}{r^2}\frac{\partial^2\lambda_2}{\partial x^2} + \frac{x^2+z^2}{r^2}\frac{\partial^2\lambda_2}{\partial y^2} + \frac{x^2+y^2}{r^2}\frac{\partial^2\lambda_2}{\partial z^2} - \frac{2xy}{r^2}\frac{\partial^2\lambda_2}{\partial x\partial y} \\
&-\frac{2xz}{r^2}\frac{\partial^2\lambda_2}{\partial x\partial z} - \frac{2yz}{r^2}\frac{\partial^2\lambda_2}{\partial y\partial z} - \frac{2x}{r^2}\frac{\partial\lambda_2}{\partial x} - \frac{2y}{r^2}\frac{\partial\lambda_2}{\partial y} - \frac{2z}{r^2}\frac{\partial\lambda_2}{\partial z} = \\
&-2\left(\frac{\partial\bar{\rho}\,u^m}{\partial x} + \frac{\partial\bar{\rho}\,v^m}{\partial y} + \frac{\partial\bar{\rho}\,w^m}{\partial z}\right) - \frac{2}{r^2}\left[\frac{x}{r}\frac{\partial}{\partial x} + \frac{y}{r}\frac{\partial}{\partial y} + \frac{z}{r}\frac{\partial}{\partial z}\right]\{r^2\bar{\rho}(v_r^o - v_r^m)\}.
\end{aligned} \tag{9.5.13}$$

Before (9.5.13) can be solved, we need appropriate boundary conditions. For the vertical boundary condition, we apply $\delta w^a = 0$ ($w^a = w^m$) to (9.5.9) and substitute (9.5.11) for λ_1 to get,

$$\frac{\partial\lambda_2}{\partial z} = \frac{1}{x^2+y^2}\left\{2\,zr\bar{\rho}(v_r^m - v_r^o) + z\left(x\frac{\partial\lambda_2}{\partial x} + y\frac{\partial\lambda_2}{\partial y}\right)\right\}. \tag{9.5.14}$$

For our lateral boundary conditions, recall that the vanishing of the area integral in (9.5.5) (and of the corresponding integral in the equation for the variation of *J* with respect to v^a) led to,

$\lambda_2 = 0$ on the *x* faces, and $\lambda_2 = 0$ on the *y* faces. (9.5.15)

The algorithm that solves for the adjusted velocity components can be summarized as follows:

(i) Ingest the radial wind component and model (or background) fields at the assimilation time.

(ii) Use successive over-relaxation to solve (9.5.13) for λ_2 on ARPS grid using boundary conditions (9.5.14) and (9.5.15).

(iii) From the λ_2 solution (ii) above, solve for λ_1 on ARPS grid using (9.5.11).

(iv) Once λ_1 and λ_2 are known, solve the Euler-Lagrange equations (9.5.7)-(9.5.9) for u^a,v^a, and w^a.

9.6. Direct Insertion Technique

As an alternative to the modified Liou/Gal-Chen technique presented above, the user may just want to insert the radial velocity component directly into the model while preserving the model's polar and tangential components. This, in effect, amounts to directly overwriting the model's radial velocity component with that of the observed radial velocity. Hence we denote this technique 'direct insertion.' It should be noted that the mass conservation equation will almost certainly be violated by this procedure.

Let v_r^o denote the observed radial velocity component. The model winds prior to the insertion have a superscript "*m*" and can be written in Cartesian and spherical polar coordinates (approximating the radar beam path as a straight line) as:

$$\vec{V}^m = u^m\hat{i} + v^m\hat{j} + w^m\hat{k} = v_r^m\hat{r} + v_\phi^m\hat{\phi} + v_\theta^m\hat{\theta}, \tag{9.6.1}$$

where $\hat{i}$, $\hat{j}$ and $\hat{k}$ are the (*x*, *y*, *z*) unit basis vectors in the Cartesian system and $\hat{r}$, $\hat{\phi}$ and $\hat{\theta}$ are the (radial, azimuthal, polar) unit basis vectors in the spherical polar system. The spherical polar components can be expressed in terms of the Cartesian components as:

$$v_r^m = \vec{V}^m \cdot \hat{r} = u^m\hat{i} \cdot \hat{r} + v^m\hat{j} \cdot \hat{r} + w^m\hat{k} \cdot \hat{r}, \tag{9.6.2}$$

$$v_\phi^m = \vec{V}^m \cdot \hat{\phi} = u^m \hat{i} \cdot \hat{\phi} + v^m \hat{j} \cdot \hat{\phi} + w^m \hat{k} \cdot \hat{\phi}, \tag{9.6.3}$$

and

$$v_\theta^m = \vec{V}^m \cdot \hat{\theta} = u^m \hat{i} \cdot \hat{\theta} + v^m \hat{j} \cdot \hat{\theta} + w^m \hat{k} \cdot \hat{\theta}. \tag{9.6.4}$$

Let the final "adjusted" winds (*i.e.*, after the insertion) have a superscript "*a*." These winds are such that the polar and azimuthal components are preserved but the radial component is given by observed data, that is,

$$\vec{V}^a = v_r^a \hat{r} + v_\phi^a \hat{\phi} + v_\theta^a \hat{\theta}, \tag{9.6.5}$$

where,

$$v_r^a = v_r^o, \quad v_\phi^a = v_\phi^m, \quad \textit{and} \quad v_\theta^a = v_\theta^m. \tag{9.6.6}$$

The adjusted winds are required in Cartesian components and, using (9.6.5) and (9.6.6), can be written as:

$$u^a = \vec{V}^a \cdot \hat{i} = v_r^a \hat{r} \cdot \hat{i} + v_\phi^a \hat{\phi} \cdot \hat{i} + v_\theta^a \hat{\theta} \cdot \hat{i} = v_r^o \hat{r} \cdot \hat{i} + v_\phi^m \hat{\phi} \cdot \hat{i} + v_\theta^m \hat{\theta} \cdot \hat{i}, \tag{9.6.7}$$

$$v^a = \vec{V}^a \cdot \hat{j} = v_r^a \hat{r} \cdot \hat{j} + v_\phi^a \hat{\phi} \cdot \hat{j} + v_\theta^a \hat{\theta} \cdot \hat{j} = v_r^o \hat{r} \cdot \hat{j} + v_\phi^m \hat{\phi} \cdot \hat{j} + v_\theta^m \hat{\theta} \cdot \hat{j}, \tag{9.6.8}$$

$$w^a = \vec{V}^a \cdot \hat{k} = v_r^a \hat{r} \cdot \hat{k} + v_\phi^a \hat{\phi} \cdot \hat{k} + v_\theta^a \hat{\theta} \cdot \hat{k} = v_r^o \hat{r} \cdot \hat{k} + v_\phi^m \hat{\phi} \cdot \hat{k} + v_\theta^m \hat{\theta} \cdot \hat{k}. \tag{9.6.9}$$

The direct insertion algorithm can be summarized as follows:

(i) Compute the model's azimuthal and polar velocity components from (9.6.3) and (9.6.4).

(ii) With v_ϕ and v_θ determined from step (i) above and with the radial velocity known from measurements, solve (9.6.6) - (9.6.8) for u^a, v^a, and w^a.

It is important to note that we need to know the location of the radar in order to evaluate the dot products in these equations.

9.7. File and Subroutine List for the Assimilation Package

This is a list of assimilation subroutines and the files that contain them.

FILE	PROGRAM/SUBROUTINE
assimbat.f	SUBROUTINE ASSIMBAT
assimbat.f	SUBROUTINE ASSIMPAR
assimcon.f	SUBROUTINE ASSIMCON
assimptpr.f	SUBROUTINE ASSIMPTPR
assimptpr.f	SUBROUTINE DABLEND
assimptpr.f	SUBROUTINE GETMNSD
assimptpr.f	SUBROUTINE SUBTR
assimrd.f	SUBROUTINE ASSIMRD
assimvel.f	SUBROUTINE ASSIMVEL
assimvel.f	SUBROUTINE POIS3D
head3d.f	SUBROUTINE BINHEAD
head3d.f	SUBROUTINE BN2HEAD
head3d.f	SUBROUTINE DTAHEAD
head3d.f	SUBROUTINE HDFHEAD
head3d.f	SUBROUTINE PAKHEAD
radread.f	SUBROUTINE RADAREAD
retrchk.f	SUBROUTINE RETRCHK
retrint.f	SUBROUTINE RETRINT
retrint.f	SUBROUTINE INTERP
retrpois.f	SUBROUTINE RETRPOIS
retrptpr.f	SUBROUTINE RETRPTPR

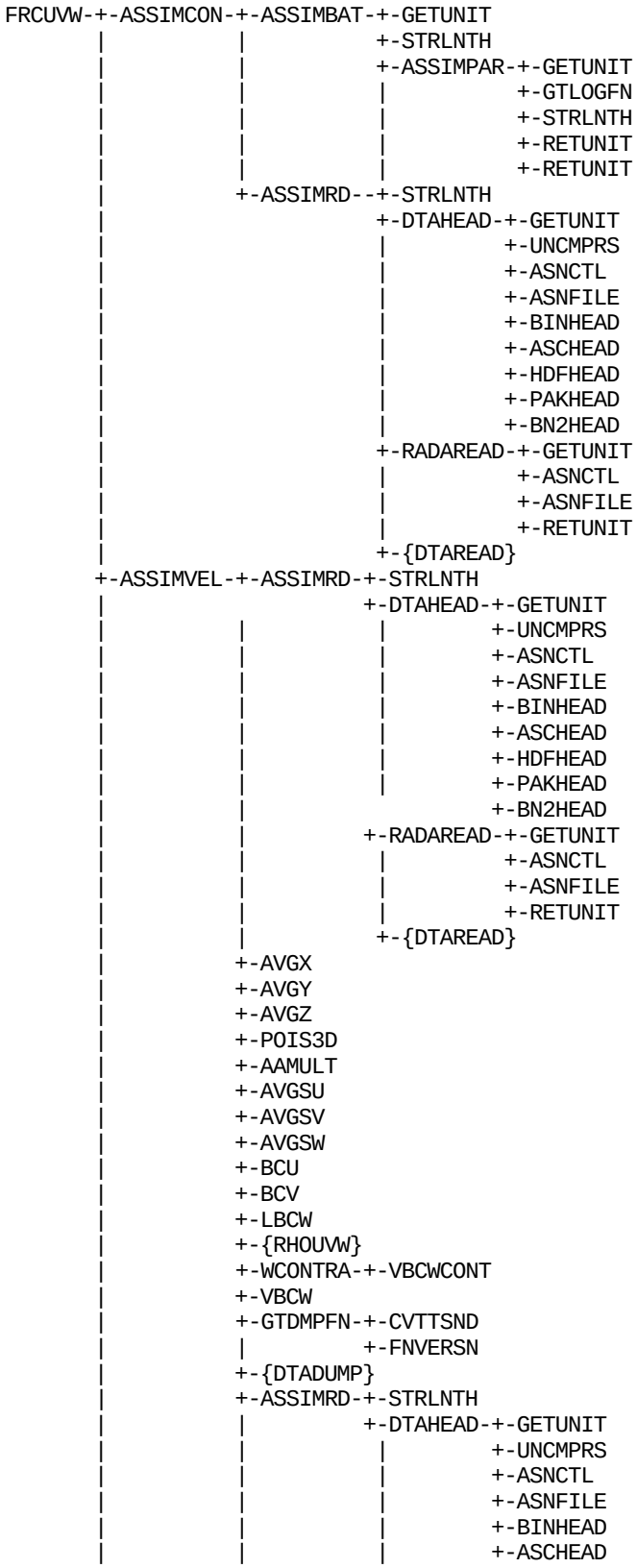
9.8. Assimilation Subroutine Glossary

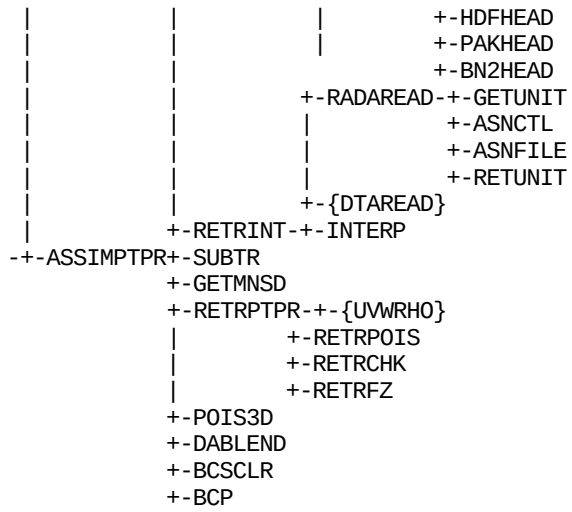
SUBROUTINE	DESCRIPTION
ASCHEAD	Read history data header ONLY in ASCII format.
ASSIMBAT	Initialize thermodynamic recovery and assimilation control parameters.

ASSIMCON	Assimilation package driver. Coordinate direct insertions, variational adjustments and thermodynamic retrievals.
ASSSIMPAR	Print and write log file of assimilation parameters.
ASSIMPTPR	Coordinate the thermodynamic retrieval and data blending.
ASSIMRD	Coordinate the ingestion of all input data for the assimilation package.
ASSIMVEL	Perform direct insertions, variational adjustments, and coordinate the hole-filling.
BINHEAD	Read history data header ONLY in binary format.
BN2HEAD	Read history data header ONLY in binary format created by ARPS.
DABLEND	Calculate a specified weighted average of an original model forecast variable and the new forecast variable (NOT operational).
DTAHEAD	History data header driver. Coordinate the ingestion of history data file headers of various formats.
GTMNSD	Calculate the statistics comparing the differences between 2 input fields.
HDFHEAD	Read history data header ONLY in an HDF scientific format.
PAKHEAD	Read history data header ONLY in a packed binary format created by ARPS.
POIS3D	Solve a three-dimensional Poisson equation to hole-fill missing data on a Cartesian grid.
RADAREAD	Read in Lincoln Lab radar data.
RETRCHK	Check the accuracy of the retrieved pressure gradients via Gal-Chen consistency check.
RETRINT	Coordinate the quadratic interpolation of input data at three time levels to three intermediate time levels.
RETRPOIS	Solve two-dimensional Poisson equation for pressure.
RETRPTPR	Retrieve temperature via a modified Gal-Chen thermodynamic technique (1978).
SUBTR	Subtract two three-dimensional arrays.

9.9. Subroutine Calling Tree for the Assimilation package

Note that the assimilation package is connected to ARPS via the subroutine FRCUVW. See the subroutine calling tree of ARPS 4.0 in Chapter 5 for other (non-assimilation) routines called by FRCUVW. A subroutine enclosed by curly brackets {} indicates that the branch stemming from that subroutine is not shown here, but can be found in Chapter 5, section 2.


FRCUVW-+-ASSIMCON-+-ASSIMBAT-+-GETUNIT
+-STRLNTH
+-ASSIMPAR-+-GETUNIT
+-GTLOGFN
+-STRLNTH
+-RETUNIT
+-RETUNIT
+-ASSIMRD--+-STRLNTH
+-DTAHEAD-+-GETUNIT
+-UNCMPRS
+-ASNCTL
+-ASNFILE
+-BINHEAD
+-ASCHEAD
+-HDFHEAD
+-PAKHEAD
+-BN2HEAD
+-RADAREAD-+-GETUNIT
+-ASNCTL
+-ASNFILE
+-RETUNIT
+-{DTAREAD}
+-ASSIMVEL-+-ASSIMRD-+-STRLNTH
+-DTAHEAD-+-GETUNIT
+-UNCMPRS
+-ASNCTL
+-ASNFILE
+-BINHEAD
+-ASCHEAD
+-HDFHEAD
+-PAKHEAD
+-BN2HEAD
+-RADAREAD-+-GETUNIT
+-ASNCTL
+-ASNFILE
+-RETUNIT
+-{DTAREAD}
+-AVGX
+-AVGY
+-AVGZ
+-POIS3D
+-AAMULT
+-AVGSU
+-AVGSV
+-AVGSW
+-BCU
+-BCV
+-LBCW
+-{RHOUVW}
+-WCONTRA-+-VBCWCONT
+-VBCW
+-GTDMPFN-+-CVTTSND
+-FNVERSN
+-{DTADUMP}
+-ASSIMRD-+-STRLNTH
+-DTAHEAD-+-GETUNIT
+-UNCMPRS
+-ASNCTL
+-ASNFILE
+-BINHEAD
+-ASCHEAD



9.10 Examining the Assimilation Code

In addition to the FORTRAN files listed in the Subroutine List, you will also need the following assimilation files:

assim.inc Include file that declares parameters, assimilation/retrieval flags and other information.

assim.input Input file that defines assimilation flags, parameters and input data file names. A complete description of the input parameters and flags can be found in the subroutine ASSIMBAT.

The ARPS assimilation package runs as a component of the ARPS model. No additional compilation or executable procedures are required. The flags set in the file *assim.input* control whether or not the assimilation/ retrieval is turned on or off (the default is OFF). Data sets dumped out by the assimilation package may be specified by a different name other than the *runname* (see *assim.input*).