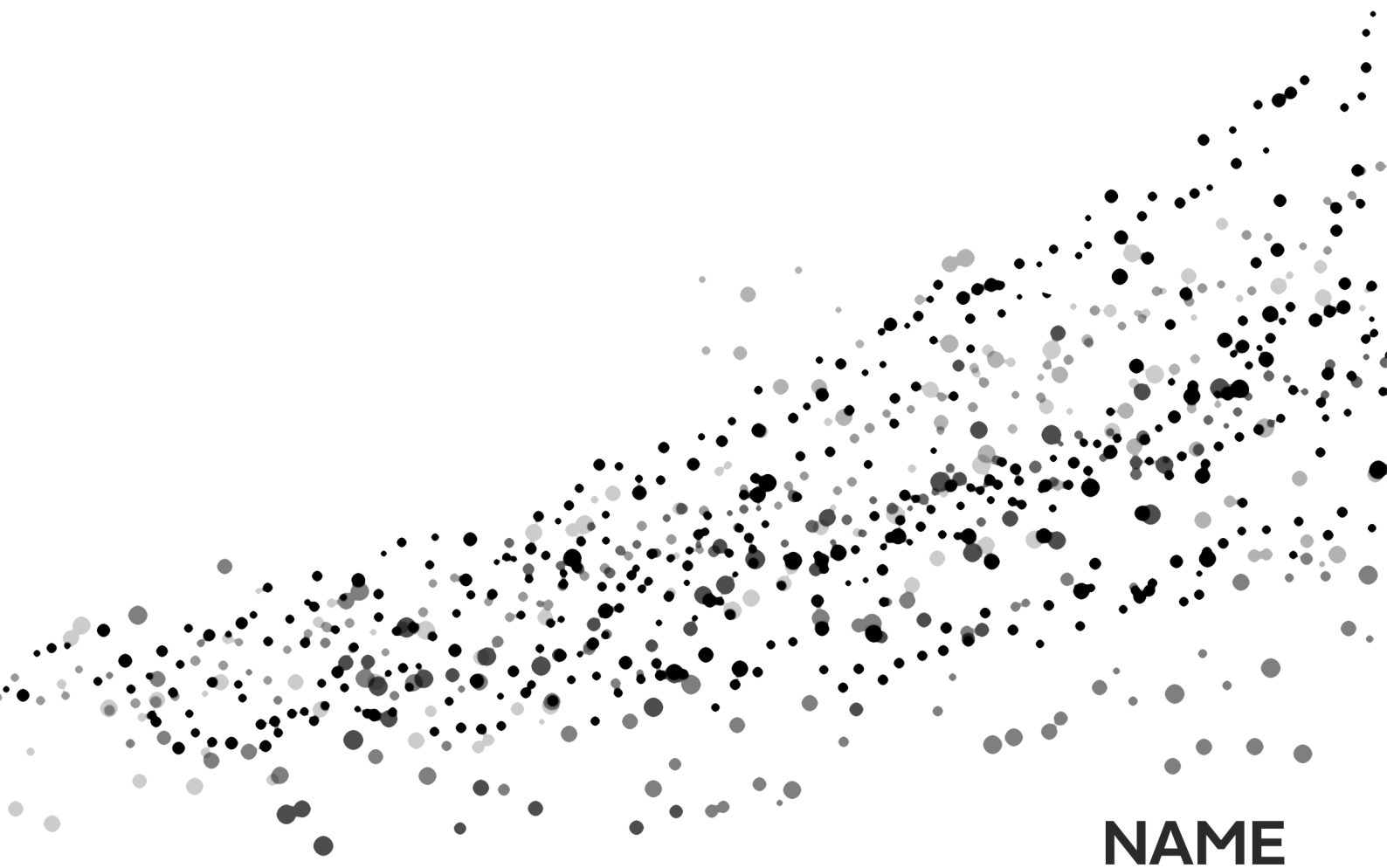
Met Office

NAME Technical Specification Document D03

Eulerian-Lagrangian hybrid model

B. J. Devenish

Documentation issued with	**:**	NAME Version 8.7
Document last updated for	**:**	NAME Version 8.2
Last updated on	**:**	06/10/2023



NAME

Numerical Atmospheric-Dispersion Modelling Environment

Met Office, FitzRoy Road, Exeter, Devon, EX1 3PB, United Kingdom

Contents

1 Introduction

These notes give a brief overview of the Eulerian model that is embedded within NAME. The model describes the evolution of a scalar field rather than the motion of individual Lagrangian particles. As such it gives the concentration of a particular species directly rather than by averaging over the total mass of all the particles in an arbitrary box. The model uses a semi-Lagrangian method to solve the advective part of the scalar transport equation. The functionality of the model includes vertical diffusion, sedimentation of solid species, wet and dry deposition. It is a hybrid model in that particles are released from a source and the material associated with each particle is then transferred to the scalar field after a user-specified time. This time can vary from one source to another and can be zero. The model can be used either as a global model or as a limited-area model. The model is integrated with NAME's chemistry scheme and is well suited to modelling the dispersion of species that evolve smoothly over large spatial scales such as ozone.

The main features of the model are described in the following sections.

2 Semi-Lagrangian scheme

The semi-Lagrangian scheme used here is based on that used in the EndGame version of the UM. The method calculates the departure point of a fictitious particle arriving at a grid point (the combination of Eulerian grid and Lagrangian particle gives the method its name). The values of the scalar field at the grid point and the departure point are then used to calculate the change in the scalar concentration (and hence its advection). The method is computationally stable even for large time steps. Further advantages of this method include monotonicity and positivity (sign preservation) i.e. spurious maxima or minima or changes of sign cannot be introduced in the advection of the scalar field. However, disadvantages of the method include potentially significant numerical diffusion which can be partly mitigated by using high-order interpolation schemes. For more information on semi-Lagrangian methods see e.g. Staniforth and Coté (1991).

The code for the semi-Lagrangian scheme can be found in the modules *euSLA.f90* and *euSLAUpdate.f90*.

3 Vertical diffusion

The model uses a Crank-Nicolson method to calculate the vertical diffusion of the scalar field. This method is a finite difference method that is typically used for solving (numerically) a parabolic partial differential equation (pde) such as the heat (or diffusion) equation. It is second-order convergent in time so that the local truncation error associated with the time step is the same order as the spatial step. In practice, this means that the time step and spatial step can be the same order of magnitude (in some non-dimensional sense). The Crank-Nicolson method uses the average value of the spatial derivative at the beginning and end of the time step. This makes it unconditionally stable i.e. errors are damped rather than magnified. It is an implicit method which requires the solution of a system of algebraic equations. This is achieved by means of Crout factorisation. The application of this method to a spatially irregular grid can be found in the appendix.

The lower boundary condition is a flux of material to the ground with a species and size-dependent (for solid particles) deposition velocity (see §5 for more details). If dry deposition is not activated then the deposition velocity is zero. The upper boundary condition requires zero flux of the scalar for all times.

The numerical subroutines are called *euModelUpdateDiffusion* and *euModelVerticalDiffusion* which can be found in *euModelUpdate.f90*. Vertical diffusion is activated by setting the *Turbulence* flag in *Sets of Dispersion Options* to be *true*.

4 Sedimentation

Sedimentation occurs automatically for solid particles i.e. those particles that are defined to have a particle size distribution (which can include a single size for all particles). In a Lagrangian model the sedimentation velocity is calculated separately for each particle (since it is size dependent). For a field of particles this is clearly not possible. In the Eulerian model the sedimentation of a solid species is calculated using the geometric mean of each bin in the particle size distribution. The calculation is performed in the subroutine *euModelFlowUpdateW-Modified* in *euModelFlow.f90* making use of the function *TerminalVelocity* (in *Physics.F90*) as for Lagrangian particles. The modified vertical velocity field is then passed to the semi-Lagrangian advection scheme.

5 Dry deposition

The deposition velocity is calculated at the lowest vertical grid level in the subroutine *euModelFlowUpdateDeposition* in *euModelFlow.f90* using the function *CalcVd* (within *Species.F90*) as for Lagrangian particles. For solid particles the representative diameter of the particles is used which is taken to be the geometric mean of the relevant particle size bin. Dry deposition is activated by setting the *Dry Deposition* flag in *Sets of Dispersion Options* to be *true*.

6 Wet deposition

Wet deposition is performed in *Case.F90*: the scalar field decays exponentially according to

$$\chi = \chi \exp(-\Delta t \lambda)$$

where χ is the scalar concentration per unit volume and Δt is the time step. The position and species-dependent scavenging coefficient, λ, is calculated in the subroutine *euModelFlowUpdateLambda* in *euModelFlow.f90* using *CalcWetScavCoeff* (within *Species.F90*). It is stored within the derived type *EulerianField*. Wet deposition is activated by setting the *Wet Deposition* flag in *Sets of Dispersion Options* to be *true*.

7 References

Burden RL & Faires JD (1997). *Numerical analysis*, 6th ed. Brooks/Cole (Thomson).

Morton KW & Mayers DF (2005). *Numerical solution of partial differential equations: an introduction*. CUP

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992). *Numerical recipes*, 2nd ed. CUP.

Staniforth A & Coté J (1991). Semi-Lagrangian integration schemes for atmospheric models – a review. Monthly Weather Review 119:2206-2223

Appendix: the Crank-Nicolson method for an irregular grid with spatially-dependent density

Discretisation of diffusion equation

Let

$$K'(z) = K(z)\rho(z), \quad \phi'(z) = \frac{\phi(z)}{\rho(z)}.$$

The discretisation of the diffusion equation uses the following results:

$$\left.\frac{\partial \phi'}{\partial z}\right|_{i-1/2} = \frac{\phi'_i - \phi'_{i-1}}{z_i - z_{i-1}}$$

where $\phi'_i = \phi'(z_i)$ and hence

$$\begin{aligned}
\left.\frac{\partial}{\partial z}\left(K'(z)\frac{\partial \phi'}{\partial z}\right)\right|_{i-1/2} &= \frac{1}{z_{i+1/2} - z_{i-1/2}}\left[K'\left.\frac{\partial \phi'}{\partial z}\right|_{i+1/2} - K'\left.\frac{\partial \phi'}{\partial z}\right|_{i-1/2}\right] \\
&= \frac{1}{z_{i+1/2} - z_{i-1/2}}\left[K'_{i+1/2}\left(\frac{\phi'_{i+1} - \phi'_i}{z_{i+1} - z_i}\right) - K'_{i-1/2}\left(\frac{\phi'_i - \phi'_{i-1}}{z_i - z_{i-1}}\right)\right]
\end{aligned}$$

where

$$K_{i-1/2} = \frac{1}{2}(K_i\rho_i + K_{i-1}\rho_{i-1}) = \frac{1}{2}(K(z_i)\rho(z_i) + K(z_{i-1})\rho(z_{i-1})).$$

Crank-Nicolson uses an average of the spatial derivative at the beginning and end of the time step (see e.g. Burden and Faires, p.691; Press *et al.* p.840). Thus, the discretisation of the diffusion equation becomes

$$\begin{aligned}
\frac{\phi_{i,j+1} - \phi_{i,j}}{\Delta t} &= \frac{1}{2(z_{i+1/2} - z_{i-1/2})}\left[K'_{i+1/2,j}\left(\frac{\phi'_{i+1,j} - \phi'_{i,j}}{z_{i+1} - z_i}\right) - K'_{i-1/2,j}\left(\frac{\phi'_{i,j} - \phi'_{i-1,j}}{z_i - z_{i-1}}\right)\right] \\
&\quad + \frac{1}{2(z_{i+1/2} - z_{i-1/2})}\left[K'_{i+1/2,j+1}\left(\frac{\phi'_{i+1,j+1} - \phi'_{i,j+1}}{z_{i+1} - z_i}\right) - K'_{i-1/2,j+1}\left(\frac{\phi'_{i,j+1} - \phi'_{i-1,j+1}}{z_i - z_{i-1}}\right)\right].
\end{aligned}$$

Re-arranging we get

$$\phi_{i,j+1} - \frac{\Delta t}{2\Delta z_{i+1/2}}\left[K'_{i+1/2,j+1}\left(\frac{\phi'_{i+1,j+1}-\phi'_{i,j+1}}{\Delta z_{i+1}}\right) - K'_{i-1/2,j+1}\left(\frac{\phi'_{i,j+1}-\phi'_{i-1,j+1}}{\Delta z_i}\right)\right]$$
$$= \phi_{i,j} + \frac{\Delta t}{2\Delta z_{i+1/2}}\left[K'_{i+1/2,j}\left(\frac{\phi'_{i+1,j}-\phi'_{i,j}}{\Delta z_{i+1}}\right) - K'_{i-1/2,j}\left(\frac{\phi'_{i,j}-\phi'_{i-1,j}}{\Delta z_i}\right)\right]$$

and hence

$$\phi_{i,j+1}\left(1 + \frac{\Delta t}{2\Delta z_{i+1/2}}\frac{1}{\rho_{i,j+1}}\left(\frac{K_{i+1/2,j+1}}{\Delta z_{i+1}} + \frac{K_{i-1/2,j+1}}{\Delta z_i}\right)\right)$$
$$-\frac{\Delta t}{2\Delta z_{i+1/2}}\frac{K'_{i+1/2,j+1}}{\Delta z_{i+1}}\frac{\phi_{i+1,j+1}}{\rho_{i+1,j+1}} - \frac{\Delta t}{2\Delta z_{i+1/2}}\frac{K_{i-1/2,j+1}}{\Delta z_i}\frac{\phi_{i-1,j+1}}{\rho_{i-1,j+1}}$$
$$= \phi_{i,j}\left(1 - \frac{\Delta t}{2\Delta z_{i+1/2}}\frac{1}{\rho_{i,j}}\left(\frac{K_{i+1/2,j}}{\Delta z_{i+1}} + \frac{K_{i-1/2,j}}{\Delta z_i}\right)\right)$$
$$+\frac{\Delta t}{2\Delta z_{i+1/2}}\frac{K_{i+1/2,j}}{\Delta z_{i+1}}\frac{\phi_{i+1,j}}{\rho_{i+1,j}} + \frac{\Delta t}{2\Delta z_{i+1/2}}\frac{K_{i-1/2,j}}{\Delta z_i}\frac{\phi_{i-1,j}}{\rho_{i-1,j}} \quad (1)$$

Boundary conditions

Lower boundary

At the ground, $z = z_1$, we assume that

$$K(z_1)\frac{\partial \phi}{\partial z}\bigg|_{z=z_1} = \phi(z_1)v_d$$

where v_d is the deposition velocity. We then have

$$\frac{\partial}{\partial z}\left(K'(z)\frac{\partial \phi'}{\partial z}\right)\bigg|_{i=1} = \frac{1}{\Delta z_{3/2}}\left[K'_{3/2}\left(\frac{\phi'_2-\phi'_1}{z_2-z_1}\right) - \phi_1 v_d\right]$$

and so

$$\phi_{1,j+1}\left(1 + \frac{\Delta t}{2\Delta z_{3/2}}\left(\frac{K'_{3/2,j+1}}{\Delta z_2\,\rho_{1,j+1}} + v_d\right)\right) - \frac{\Delta t}{2\Delta z_{3/2}}\frac{K'_{3/2,j+1}}{\Delta z_2}\frac{\phi_{2,j+1}}{\rho_{2,j+1}}$$
$$= \phi_{1,j}\left(1 - \frac{\Delta t}{2\Delta z_{3/2}}\left(\frac{K'_{3/2,j}}{\Delta z_2\,\rho_{1,j}} + v_d\right)\right) + \frac{\Delta t}{2\Delta z_{3/2}}\frac{K'_{3/2,j}}{\Delta z_2}\frac{\phi_{2,j}}{\rho_{2,j}}$$

Upper boundary

We have

$$\frac{\partial}{\partial z}\left(K'(z)\frac{\partial \phi'}{\partial z}\right)\bigg|_{i=n} = \frac{1}{\Delta z_{n+1/2}}\left[K'_{n+1/2}\left(\frac{\phi'_{n+1}-\phi'_n}{z_{n+1}-z_n}\right) - K'_{n-1/2}\left(\frac{\phi'_n-\phi'_{n-1}}{z_n-z_{n-1}}\right)\right]$$

and so

$$\phi_{n,j+1}\left(1 + \frac{\Delta t}{2\Delta z_{n+1/2}}\frac{1}{\rho_{n,j+1}}\left(\frac{K'_{n+1/2,j+1}}{\Delta z_{n+1}} + \frac{K'_{n-1/2,j+1}}{\Delta z_n}\right)\right)$$
$$-\frac{\Delta t}{2\Delta z_{n+1/2}}\frac{K'_{n+1/2,j+1}}{\Delta z_{n+1}}\frac{\phi_{n+1,j+1}}{\rho_{n+1,j+1}} - \frac{\Delta t}{2\Delta z_{n+1/2}}\frac{K'_{n-1/2,j+1}}{\Delta z_n}\frac{\phi_{n-1,j+1}}{\rho_{n-1,j+1}}$$
$$= \phi_{n,j}\left(1 - \frac{\Delta t}{2\Delta z_{n+1/2}}\frac{1}{\rho_{n,j}}\left(\frac{K'_{n+1/2,j}}{\Delta z_{n+1}} + \frac{K_{n-1/2,j}}{\Delta z_n}\right)\right) + \frac{\Delta t}{2\Delta z_{n+1/2}}\frac{K'_{n+1/2,j}}{\Delta z_{n+1}}\frac{\phi_{n+1,j}}{\rho_{n+1,j}} + \frac{\Delta t}{2\Delta z_{n+1/2}}\frac{K_{n-1/2,j}}{\Delta z_n}\frac{\phi_{n-1,j}}{\rho_{n-1,j}}$$

NB: level z_{n+1} exists above domain top. One option is to choose $\phi_{n+1,j} = \phi_{n+1,j+1} = 0$ however this could lead to spuriously large gradients near the top of the domain. A better option is to require zero flux at the top boundary i.e. $\phi'_{n+1,j} = \phi'_{n,j}$ for all times (NB: this is now a flux of mixing ratio rather than concentration). In this case, the above equation becomes

$$\phi_{n,j+1}\left(1 + \frac{\Delta t}{2\Delta z_{n+1/2}}\frac{K'_{n-1/2,j+1}}{\Delta z_n}\frac{1}{\rho_{n,j+1}}\right) - \frac{\Delta t}{2\Delta z_{n+1/2}}\frac{K'_{n-1/2,j+1}}{\Delta z_n}\frac{\phi_{n-1,j+1}}{\rho_{n-1,j+1}}$$

$$= \phi_{n,j}\left(1 - \frac{\Delta t}{2\Delta z_{n+1/2}}\frac{K'_{n-1/2,j}}{\Delta z_n}\frac{1}{\rho_{n,j}}\right) + \frac{\Delta t}{2\Delta z_{n+1/2}}\frac{K'_{n-1/2,j}}{\Delta z_n}\frac{\phi_{n-1,j}}{\rho_{n-1,j}}$$

Crout factorisation

Equation (1) can be written as a tridiagonal matrix

$$A\boldsymbol{\phi}^{(j+1)} = B\boldsymbol{\phi}^{(j)}$$

where A and B are respectively the $n \times n$ matrices

$$A = \begin{pmatrix} 1+\frac{\lambda_{3/2}}{\rho_{1,j+1}}\left(\frac{K_{3/2,j+1}}{\Delta z_2}+\frac{K_{1/2,j+1}}{\Delta z_1}\right) & -\frac{\lambda_{3/2}}{\rho_{2,j+1}}\frac{K_{3/2,j+1}}{\Delta z_2} & 0 & \cdots\cdots\cdots & 0 \\ -\frac{\lambda_{5/2}}{\rho_{1,j+1}}\frac{K_{3/2,j+1}}{\Delta z_2} & 1+\frac{\lambda_{5/2}}{\rho_{2,j+1}}\left(\frac{K_{5/2,j+1}}{\Delta z_3}+\frac{K_{3/2,j+1}}{\Delta z_2}\right) & -\frac{\lambda_{5/2}}{\rho_{3,j+1}}\frac{K_{5/2,j+1}}{\Delta z_3} & 0 \quad \cdots & \vdots \\ 0 & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots\cdots\cdots\cdots & 0 & -\frac{\lambda_{n+1/2}}{\rho_{n-1,j+1}}\frac{K_{n-1/2,j+1}}{\Delta z_n} & 1+\frac{\lambda_{n+1/2}}{\rho_{n,j+1}}\frac{K_{n-1/2,j+1}}{\Delta z_n} \end{pmatrix}$$

and

$$B = \begin{pmatrix} 1-\frac{\lambda_{3/2}}{\rho_{1,j}}\left(\frac{K_{3/2,j}}{\Delta z_2}+\frac{K_{1/2,j}}{\Delta z_1}\right) & \frac{\lambda_{3/2}}{\rho_{2,j}}\frac{K_{3/2,j}}{\Delta z_2} & 0 & \cdots\cdots & 0 \\ \frac{\lambda_{5/2}}{\rho_{1,j}}\frac{K_{3/2,j}}{\Delta z_2} & 1-\frac{\lambda_{5/2}}{\rho_{2,j}}\left(\frac{K_{5/2,j}}{\Delta z_3}+\frac{K_{3/2,j}}{\Delta z_2}\right) & \frac{\lambda_{5/2}}{\rho_{3,j}}\frac{K_{5/2,j}}{\Delta z_3} & 0 & \vdots \\ 0 & & \ddots & & \vdots \\ \vdots & & & & \vdots \\ 0 & \cdots\cdots\cdots & 0 & \frac{\lambda_{n+1/2}}{\rho_{n-1,j}}\frac{K_{n-1/2,j}}{\Delta z_n} & 1-\frac{\lambda_{n+1/2}}{\rho_{n,j}}\frac{K_{n-1/2,j}}{\Delta z_n} \end{pmatrix}$$

and

$$\lambda_i = \frac{\Delta t}{2\Delta z_{i+1/2}}.$$

This tridiagonal system is most easily solved using Crout factorisation (Burden and Faires, p. 413; Press *et al.*, p. 34+) which involves the LU decomposition of matrix A so that

$$LU\boldsymbol{\phi}^{(j+1)} = b \tag{2}$$

where $b = B\boldsymbol{\phi}^{(j)}$. Equation (2) can be solved in two stages:

$$Lz = b$$

where

$$U\boldsymbol{\phi}^{(j+1)} = z.$$

The components of L and U can be determined as follows:

$$\begin{gathered} L_{11} = A_{11}, \quad U_{12} = A_{12}/L_{11} \\ L_{i,i-1} = A_{i,i-1}, \quad L_{ii} = A_{ii} - L_{i,i-1}U_{i-1,i}, \quad U_{i,i+1} = A_{i,i+1}/L_{ii}, \quad i = 2, \ldots, n-1, \\ L_{n,n-1} = A_{n,n-1}, \quad L_{nn} = A_{nn} - L_{n,n-1}U_{n-1,n} \end{gathered}$$

Note that $U_{ii} = 1$ ($i = 1, \ldots, n$). The values of z are given by

$$\begin{gathered} z_1 = b_1/L_{11}, \\ z_i = (b_i - L_{i,i-1}z_{i-1})/L_{ii}, \quad i = 2, \ldots, n-1, \\ z_n = (b_i - L_{n,n-1}z_{n-1})/L_{nn} \end{gathered}$$

Once z is known, $\boldsymbol{\phi}^{(j+1)}$ can be obtained by back-substitution:

$$\begin{gathered} \phi_n^{(j+1)} = z_n, \\ \phi_i^{(j+1)} = z_i - U_{i,i+1}b_{i+1}, \quad i = n-1, \ldots, 1 \end{gathered}$$