Supplemental Material

Journal of Physical Oceanography
A Unifying Approach to Subtidal Salt Intrusion Modeling in Tidal Estuaries
https://doi.org/10.1175/JPO-D-20-0006.1

© Copyright 2020 American Meteorological Society (AMS)
For permission to reuse any portion of this work, please contact permissions@ametsoc.org. Any use of material in this work that is determined to be “fair use” under Section 107 of the U.S. Copyright Act (17 USC §107) or that satisfies the conditions specified in Section 108 of the U.S. Copyright Act (17 USC §108) does not require AMS’s permission. Reproduction, systematic reproduction, posting in electronic form, such as on a website or in a searchable database, or other uses of this material, except as exempted by the above statement, requires written permission or a license from AMS. All AMS journals and monograph publications are registered with the Copyright Clearance Center (https://www.copyright.com). Additional details are provided in the AMS Copyright Policy statement, available on the AMS website (https://www.ametsoc.org/PUBSCopyrightPolicy).
Supplement to: ”A unifying approach to subtidal salt intrusion modelling in tidal estuaries”

YOERI M. DIJKSTRA*, HENK M. SCHUTTELAARS

Delft Institute of Applied Mathematics, Delft University of Technology, Netherlands

This supplement contains an elaborate description of the model, solution method, and numerical implementation.

1. Equations

The equations described in this supplement are written in a slightly more general form than in the main text, allowing for along-channel variations of the width and small variations of the depth. We investigate the following equations for the water motion

\[-(A_v u_z)_z + uu_z + wu_z = -g \zeta_z - gB \int_0^z s_z dz\]  

(momentum),

\[w_z + \frac{1}{B} (Bu)_z = 0\]  

(continuity),

where \(u\) and \(w\) are the along-channel and vertical flow velocities, \(A_v\) is the eddy viscosity, \(g\) is the acceleration of gravity, \(\zeta\) is the water level gradient, \(s\) is salinity, and \(B\) is the channel width. The equations are applied to a Cartesian coordinate system \((\hat{x}, z)\) with the horizontal coordinate \(\hat{x}\) from the seaward boundary at \(\hat{x} = 0\) to the landward boundary at \(\hat{x} = L\) and with the bottom boundary at \(z = -H\) and the fixed surface at \(z = 0\) (rigid lid). Any subscripts \(\hat{x}\) or \(z\) denote derivatives with respect to that dimension.

We use the following boundary conditions:

\[A_v u_z = 0\]  

\((z = 0)\),

\[A_v u_z = s_f u\]  

\((z = -H)\),

\[w + uH_\hat{x} = 0\]  

\((z = -H)\),

\[-(A_v u_z)_z = -g \zeta_\hat{x}\]  

\((\hat{x} = L)\).

where \(s_f\) is a linearised roughness coefficient. The first two conditions indicate no-stress and partial slip boundaries are prescribed at the surface and bed. The third condition is a non-permeability condition at the bed. The fourth condition indicates that the flow at the landward boundary is required to satisfy a simplified version of the momentum equation, only containing barotropic pressure and turbulent mixing. Integrating the continuity equation (2) over depth and requiring that the integrated flow equals a prescribed discharge \(Q\) additionally yields the closure condition

\[B \int_{-H}^0 u dz = -Q.\]  

The following equation is used for salinity

\[-(K_v s_z)_z = -us_\hat{x} - ws_z + \frac{1}{B} (BK_\hat{x} s_\hat{x})_\hat{x},\]  

\(\hat{x} = L\).

* Corresponding author address: Delft Institute of Applied Mathematics, Delft University of Technology, Delft, Netherlands
E-mail: y.m.dijkstra@tudelft.nl

©2020
where $K_v$ and $K_h$ denote the vertical and horizontal eddy diffusivity. This equation has the following boundary conditions

\begin{equation}
K_v s_z - K_h s_z \xi_z = 0 \quad (z = 0),
\end{equation}

\begin{equation}
K_v s_z + K_h s_z H_z = 0 \quad (z = -H),
\end{equation}

\begin{equation}
- \left( K_v s_z \right)_{\hat{x}} = -us_{\hat{x}} - ws_z \quad (\hat{x} = 0, z = -H),
\end{equation}

\begin{equation}
\left. \right|_{s = 0} \left( K_v s_z \right)_{\hat{x}} = \left. \right|_{s = \text{sea}} \left( K_v s_z \right)_{\hat{x}} = \left. \right|_{x = L, \sigma \in [-1,0]}.
\end{equation}

In words, at the seaward boundary, the salinity at the bed is prescribed, while the equation without horizontal diffusion should hold in the rest of the water column. At the landward boundary, the salinity is assumed to vanish.

\subsection*{a. Sigma transformation}

In order to simplify the notation later on, the domain is transformed from $(\hat{x}, z)$-coordinates to $(x, \sigma)$-coordinates, where

\begin{equation}
x(\hat{x}, z) = \hat{x},
\end{equation}

\begin{equation}
\sigma(\hat{x}, z) = \frac{z}{H(x)}.
\end{equation}

As a result, $\sigma \in [-1,0]$ (bed to surface), regardless of the $x$-coordinate. The coordinate transformation satisfies the following conditions

\begin{equation}
\frac{\partial}{\partial \hat{x}} = \frac{\partial x}{\partial \hat{x}} \frac{\partial}{\partial x} + \frac{\partial \sigma}{\partial \hat{x}} \frac{\partial}{\partial \sigma} = \frac{\partial x}{\partial x} \frac{\partial}{\partial x} - \frac{\sigma H_x}{H} \frac{\partial}{\partial \sigma},
\end{equation}

\begin{equation}
\frac{\partial}{\partial z} = \frac{\partial x}{\partial z} \frac{\partial}{\partial x} + \frac{\partial \sigma}{\partial z} \frac{\partial}{\partial \sigma} = \frac{1}{H} \frac{\partial}{\partial \sigma}.
\end{equation}

For the following, we will assume that $H_x \approx 0$ and thus ignore the term $-\sigma \frac{H_x}{H} \frac{\partial}{\partial \sigma}$ in the above transformation. This restricts the model to small variations in bed level. The equations for hydrodynamics then become

\begin{equation}
- \frac{1}{H^2} A_v u_{\sigma \sigma} = -g \xi_z - g \beta H \int_0^\sigma s_x d\sigma',
\end{equation}

\begin{equation}
\frac{1}{H} w_{\sigma} + \frac{1}{B}(Bu)_x = 0.
\end{equation}

The boundary conditions read as

\begin{equation}
\frac{1}{H} A_v u_{\sigma} = 0 \quad (\sigma = 0),
\end{equation}

\begin{equation}
\frac{1}{H} A_v u_{\sigma} = s_f u \quad (\sigma = -1),
\end{equation}

\begin{equation}
w = 0 \quad (\sigma = -1),
\end{equation}

\begin{equation}
- \frac{1}{H^2} (A_v u_{\sigma})_{\sigma} = -g \xi_x \quad (x = L),
\end{equation}

with depth-integrated continuity equation

\begin{equation}
BH \int_{-1}^0 u d\sigma = -Q.
\end{equation}

The transformed salinity equation reads

\begin{equation}
- \frac{1}{H^2} (K_v s_{\sigma \sigma})_{\sigma} + \frac{1}{H} w_{\sigma} + us_x - \frac{1}{B} (BK_h s_x)_x = 0,
\end{equation}

\begin{equation}
\left. \right|_{s = \text{sea}} (K_v s_{\sigma \sigma})_{\sigma} = \left. \right|_{s = \text{sea}} w_{\sigma} + us_x - \frac{1}{B} (BK_h s_x)_x = 0.
\end{equation}
with boundary conditions

\[
\frac{1}{H} K_v s_\sigma - K_v \zeta x = 0
\]  
\( (\sigma = 0), \tag{15a} \)

\[
\frac{1}{H} K_v s_\sigma = 0
\]  
\( (\sigma = -1), \tag{15b} \)

\[
-\frac{1}{H^2} (K_v s_\sigma)_\sigma = -u s_x - \frac{1}{H} w s_\sigma
\]  
\( (x = 0, \sigma = -1), \tag{15c} \)

\[
-\frac{1}{H} \frac{1}{K_v} (s_\sigma)_\sigma = -s_x - \frac{1}{H} w s_\sigma
\]  
\( (x = 0, \sigma \in (-1, 0]), \tag{15d} \)

\[
s = 0
\]  
\( (x = L, \sigma \in [-1, 0]), \tag{15e} \)

It is assumed that \( \zeta x \approx 0 \) in these boundary conditions. Hence, \( \zeta x \) is ignored everywhere but in the barotropic pressure term in the momentum equation.

**b. Turbulence model**

The eddy viscosity \( A_v \) and eddy diffusivity \( K_v \) are assumed to be time independent and depth uniform. To parametrize \( A_v \) and \( K_v \), we use the KEFittedLead module of iFlow with roughness parameter \( s_f \). This implements the expression

\[
A_v = 0.49 s_f H.
\] \( \tag{16} \)

The eddy diffusivity is related to the eddy viscosity using a Prandtl-Schmidt number \( \sigma_p \), i.e.

\[
K_v = \frac{A_v}{\sigma_p}
\] \( \tag{17} \)

Throughout this document, we will use the relative roughness parameter \( R \), which we define as

\[
R = \frac{A_v}{s_f H}.
\] \( \tag{18} \)

Using our turbulence closure, \( R \) is constant and has a value \( R = 0.49 \).

**2. Eigenfunction expansion**

**a. Horizontal velocity**

We will develop a numerical solver for the fully non-linear equations using a truncated series expansion. To this end, the velocity is written as a series expansion

\[
u(x, \sigma) = \sum_{m=0}^{\infty} (-1)^m \beta_m(x) f_m(\sigma).
\] \( \tag{19} \)

The solutions are written as an infinite sum, which is truncated at some finite number \( M \) for the numerical computation. Hence this solution method is known as a truncation method. If the factor \( (-1)^m \) were not used, we found that consecutive rows in the resulting matrix equation have alternating signs. The extra factor \( (-1)^m \) prevents this from occurring in certain situations. The functions \( f_m \) are chosen equal to the eigenfunctions of the vertical turbulent viscosity operator in such a way that \( f_m \) solves for

\[
\frac{A_v}{H^2} f_m, \sigma = \frac{A_v}{H^2} \lambda_m^2 f_m
\] \( \tag{20} \)

with eigenvalues \( \lambda_m \) and satisfying the boundary conditions at \( \sigma = -1 \) and \( \sigma = 0 \),

\[
\frac{A_v}{H} f_m, \sigma = 0 \quad \text{at } \sigma = 0, \tag{21a}
\]

\[
\frac{A_v}{H} f_m, \sigma - s f_m = 0 \quad \text{at } \sigma = -1. \tag{21b}
\]
The eigenfunctions for depth uniform $A_ν$ consist of sines and cosines. The sines do not satisfy the surface boundary condition, such that we have $f_m = \cos(\lambda_m \sigma)$. The eigenvalues $\lambda_m$ should be such that the bottom boundary condition is satisfied. This reads as

$$ \frac{A_ν}{H} \lambda_m \sin(\lambda_m) - s_f \cos(\lambda_m) = 0, \quad (22) $$

which rewrites to

$$ \lambda_m \tan(\lambda_m) = R^{-1}. \quad (23) $$

This implicit equation is solved using a numerical root-finding algorithm using that $\lambda_m \in (m\pi - \frac{1}{2}, m\pi + \frac{1}{2})$ for $m = 0, 1, \ldots$. For the remainder of this memo we will assume $\lambda_0 \neq 0$ (i.e. $\lambda_0 > 0$). This is true as long as $R < \infty$, i.e. $s_f H > 0$.

b. Vertical velocity

Using the continuity equation, the eigenfunction expansion for $u$ can be used to formulate an eigenfunction expansion for $w$. To this end, we solve for $w$ by integrating the continuity equation, i.e.

$$ w(\sigma) - w(-1) = -\int_{-1}^{\sigma} \frac{H}{B} (B w)_x \, d\sigma' $$

$$ = -\frac{H}{B} \sum_{m=0}^{M} \left( \frac{(B \beta_m)_x}{\lambda_m} (-1)^m (\sin(\lambda_m \sigma) - \sin(-\lambda_m)) \right), \quad (24) $$

where $w(-1) = 0$. Hence

$$ w(\sigma) = -H \sum_{m=0}^{M} \frac{1}{\lambda_m} \left( \beta_{m,x} + \frac{B_x}{B} \beta_m \right) (-1)^m (\sin(\lambda_m \sigma) + \sin(\lambda_m)). \quad (25) $$

c. Salinity

Similarly, the salinity is expressed in terms of an eigenfunction expansion, which reads as

$$ s(x, \sigma) = \sum_{m=0}^{\infty} \alpha_m(x)(-1)^m g_m(\sigma). \quad (26) $$

The functions $g_m$ are chosen to be the eigenfunctions of the vertical diffusivity operator such that

$$ K_v \frac{H^2 g_{m,\sigma}}{H^2} = K_v \frac{H^2 g_m}{H^2} \mu_m \quad (27) $$

with eigenvalues $\mu_m$ and satisfying the boundary conditions at $\sigma = -1$ and $\sigma = 0$,

$$ K_v \frac{H g_{m,\sigma}}{H} = 0 \quad \text{at } \sigma = 0, \quad (28a) $$

$$ K_v \frac{H g_{m,\sigma}}{H} = 0 \quad \text{at } \sigma = -1. \quad (28b) $$

Hence, the eigenfunctions equal

$$ g_m(\sigma) = \cos(m\pi \sigma). \quad (29) $$
3. Galerkin forms

a. Water motion

MOMENTUM EQUATION

The eigenfunction expansions are substituted into the momentum equation, which then reads as

\[ \sum_{m=0}^{M} \sum_{n=0}^{M} \frac{A_v}{H^2} \lambda_m^2 \beta_m (-1)^m \cos(\lambda_m \sigma) + \beta_n \beta_{m,x} (-1)^n (-1)^m \cos(\lambda_m \sigma) \cos(\lambda_n \sigma) + \frac{\lambda_n}{\lambda_m} \left( \beta_{m,x} + \frac{B_s}{B} \beta_m \right) \beta_n (-1)^m (-1)^n \sin(\lambda_m \sigma) \sin(\lambda_n \sigma) \]

\[ = -g \bar{\zeta}_x + g \beta H \left( \alpha_{0,x} + \sum_{m=1}^{M} \frac{\alpha_{m,x}}{m \pi} (-1)^m \sin(m \pi \sigma) \right). \] (30)

To derive this equation, it is used that

- \[ u_\sigma = \sum_{m=0}^{M} -\lambda_m (-1)^m \sin(\lambda_m \sigma) \beta_m, \] (31a)
- \[ \int_{-1}^{0} s_x d' \sigma' = \sum_{m=0}^{M} \int_{-1}^{0} \alpha_m (-1)^m \cos(m \pi \sigma') d' \sigma' \]

\[ = -\alpha_0 \sigma + \sum_{m=1}^{M} \frac{\alpha_m}{m \pi} (-1)^m \sin(m \pi \sigma') \bigg|_{-1}^{0} \]

\[ = - \left( \alpha_0 \sigma + \sum_{m=1}^{M} \frac{\alpha_m}{m \pi} (-1)^m \sin(m \pi \sigma) \right). \] (31b)

A Galerkin method is applied to solve this equation. To this end, the equation and boundary conditions are multiplied by test functions of the type \((-1)^k \cos(\lambda_k \sigma), k = 0, \ldots, M\) and then integrated from \(\sigma = -1\) to 0. The resulting equation for \(k = 0, \ldots, M\) reads as

\[ \sum_{m=0}^{M} \sum_{n=0}^{M} \frac{A_v}{H^2} G_{1,km} \beta_m + G_{2,km} \beta_m \beta_{m,x} + G_{3,km} \left( \beta_{m,x} + \frac{B_s}{B} \beta_m \right) \beta_n = -g \bar{\zeta}_x G_{4,k} + \sum_{m=0}^{M} \beta H G_{5,km} \alpha_{m,x}, \] (32)

where

- \[ G_{1,km} = \lambda_m^2 \int_{-1}^{0} c_m c_k d \sigma, \] (33a)
- \[ G_{2,km} = \int_{-1}^{0} c_n c_k d \sigma, \] (33b)
- \[ G_{3,km} = \frac{\lambda_n}{\lambda_m} \int_{-1}^{0} s_n (s_m + \sin(\lambda_m)) c_k d \sigma, \]

\[ \text{(assuming } \lambda_m > 0) \] (33c)
- \[ G_{4,k} = \int_{-1}^{0} c_k d \sigma, \] (33d)
- \[ G_{5,km} = \begin{cases} \int_{-1}^{0} \sigma c_k d \sigma & \text{for } m = 0, \\ \frac{1}{m \pi} \int_{-1}^{0} \bar{s}_m c_k d \sigma & \text{for } m > 0, \end{cases} \] (33e)

and

- \[ c_j = (-1)^j \cos(\lambda_j \sigma), \] (34a)
- \[ s_j = (-1)^j \sin(\lambda_j \sigma), \] (34b)
- \[ \bar{c}_j = (-1)^j \cos(i \pi \sigma), \] (34c)
- \[ \bar{s}_j = (-1)^j \sin(i \pi \sigma). \] (34d)
BOUNDARY CONDITIONS

The boundary conditions at the bed and surface are automatically satisfied by the choice of the eigenfunctions. The boundary condition at \( x = L \) rewrites to

\[
\sum_{m=0}^{M} \frac{A}{H^2} G_{1,km} \beta_m = -g \xi_s G_{4,k}. \tag{35}
\]

DEPTH-INTEGRATED CONTINUITY

The depth-integrated continuity equation (13) rewrites to

\[
BH \int_{-1}^{0} u d\sigma = BH \sum_{m=0}^{M} \beta_m \int_{-1}^{0} (-1)^m \cos(\lambda_m \sigma) d\sigma,
\]

\[
= BH \sum_{m=0}^{M} \frac{\beta_m}{\lambda_m} (-1)^m \sin(\lambda_m) = -Q. \tag{36}
\]

For consistency of notation, this is written as

\[
\sum_{m=0}^{M} G_{9,m} \beta_m = -\frac{Q}{BH},
\]

where

\[
G_{9,m} = (-1)^n \frac{\sin(\lambda_m)}{\lambda_m}.
\]

b. Salinity

SALINITY EQUATION

Substituting the series expression for \( u, w, \) and \( s \) in the salinity equation, we obtain

\[
\sum_{m=0}^{M} \sum_{n=0}^{N} \frac{K}{H^2} (m\pi)^2 \alpha_m (-1)^m \cos(m\pi \sigma) + \alpha_m, x \beta_n (-1)^m (-1)^n \cos(m\pi \sigma) \cos(\lambda_n \sigma)
\]

\[
+ \frac{m\pi}{B} \frac{(B\beta_n)_x}{\lambda_m} \alpha_m (-1)^m (-1)^n \sin(m\pi \sigma) \sin(\lambda_n \sigma) + \sin(\lambda_n) = \sum_{m=0}^{M} \frac{1}{B} (BK_h \alpha(x,t) \alpha_m) x (-1)^m \cos(m\pi \sigma),
\]

\[
\tag{38}
\]

This is simplified using a Galerkin method with test functions \((-1)^k \cos(k\pi \sigma)\) for \( k = 0, \ldots, M \), i.e. multiplying the equation with the test function and integrating from \( \sigma = -1 \) to 0, we find

\[
\sum_{m=0}^{M} \sum_{n=0}^{N} \frac{K}{H^2} G_{6,km} \alpha_m + G_{7,kmn} \alpha_m, x \beta_n + \frac{1}{B} G_{8,kmn} (B\beta_n)_x \alpha_m = \sum_{m=0}^{M} \frac{1}{B} G_{6b,km} (BK_h \alpha_m, x)_x , \tag{39}
\]

where

\[
G_{6,km} = (m\pi)^2 \int_{-1}^{0} \bar{e}_m \bar{c}_k d\sigma, \tag{40a}
\]

\[
G_{6b,km} = \int_{-1}^{0} \bar{e}_m \bar{c}_k d\sigma, \tag{40b}
\]

\[
G_{7,kmn} = \int_{-1}^{0} c_n \bar{e}_m \bar{c}_k d\sigma, \tag{40c}
\]

\[
G_{8,kmn} = \frac{m\pi}{\lambda_n} \int_{-1}^{0} (s_n + \sin(\lambda_n)) \bar{s}_m \bar{c}_k, \tag{40d}
\]

(assuming \( \lambda_n > 0 \)).
BOUNDARY CONDITIONS

The boundary condition at the seaward boundary for \( \sigma \in (-1, 0] \) is similar to the salinity equation but without the horizontal dispersion term. In Galerkin form this is applied for \( k = 1, \ldots, M \) (i.e. all but the first element) and reads as

\[
\sum_{m=0}^{M} \sum_{n=0}^{M} K_{\sigma} G_{\sigma, kmn} \alpha_{m} + G_{\sigma, kmn} \beta_{m} + \frac{1}{B} G_{\sigma, kmn} (B \beta_{n}) x \alpha_{m} = 0.
\]  

(41)

The equation for \( k = 0 \) is replaced by the condition \( s = s_{\text{sea}} \) at \( x = 0 \), \( \sigma = -1 \). Using the eigenfunction expansion this reads as

\[
\sum_{m=0}^{\infty} G_{10, m} \alpha_{m} = s_{\text{sea}},
\]

(42)

where

\[
G_{10, m} = (-1)^{m} \cos(m\pi).
\]

(43)

The landward boundary condition \( s = 0 \) is rewritten to Galerkin form for \( k = 0, \ldots, M \) as

\[
\sum_{m=0}^{M} G_{6, mk} \alpha_{m} = 0.
\]

(44)

4. Numerical implementation: Newton-Raphson and discretisation

The Galerkin forms of the equations provide a set of \( M \) non-linear equations for the horizontal velocity (coefficients \( \beta_{0}, \ldots, \beta_{M} \)), \( M \) non-linear equations for the salinity (coefficients \( \alpha_{0}, \ldots, \alpha_{M} \)) and 1 linear equation for \( \zeta_{x} \). It is chosen to solve the entire set of \( 2M + 1 \) equations together in terms of a state vector \( y \) defined as

\[
y = [\alpha, \beta, \zeta_{x}]^{T},
\]

(45a)

\[
\alpha = [\alpha_{0}, \ldots, \alpha_{M}]^{T},
\]

(45b)

\[
\beta = [\beta_{0}, \ldots, \beta_{M}]^{T}.
\]

(45c)

As the system of equations is non-linear, a Newton-Raphson method is used to solve the equations. Furthermore, the equations are solved using a finite volume method on a grid in along-channel direction. To this end, the solution is discretised to \( y^{i,j} \) with iteration number \( i = 0, 1, \ldots \) and grid cell number \( j = 0, \ldots, j_{\text{max}} \).

The system can be solved efficiently because the Galerkin form presented above results in a mode splitting between \( x \) and \( z \). All \( z \) dependency is in the set \( G_{i} (i = 1, \ldots, 10, 6b) \). This set does not depend on \( x \) nor on any element of \( y \). Hence, the expressions for \( G_{i} \) only need to be computed once for all combinations of \( k, m, \) and \( n \). As these computations involve simple integrals of sines and cosines, the results are computed analytically. All \( x \)-dependency is in the \( 2M + 1 \) Galerkin equations, which do not further depend on \( z \).

a. Newton-Raphson method

The Newton-Raphson iteration for some non-linear equation of the form \( F(y) = 0 \) reads as

\[
y^{i+1} = y^{i} - \left( \frac{\partial F(y^{i})}{\partial y} \right)^{-1} F(y^{i}).
\]

(46)

Instead of solving for this equation directly and needing to find the inverse of the Jacobian \( \frac{\partial F(y^{i})}{\partial y} \), we solve for a slightly rewritten form of the equation, which reads as

\[
y^{i+1} = y^{i} + dy^{i+1},
\]

(47a)

\[
\frac{\partial F(y^{i})}{\partial y} dy = -F(y^{i}),
\]

(47b)
where the increment $dy$ is a vector of increments of $\alpha$, $\beta$, and $\zeta$, i.e. $[d\alpha_0, \ldots, d\alpha_M, d\beta_0, \ldots, d\beta_M, d\zeta]^T$.

The vector function $F$ consists of the three sets of equations $F_1$, $F_2$, and $F_3$. The equations are repeated below for elements of the vector functions $F_1$ and $F_2$ and the scalar function $F_3$. The notation is changed in two ways: 1) in this section we use Einstein’s summation convention\(^1\), and 2) derivatives of state variables are written using $\nabla$ and $\Delta$:

$$\begin{align*}
F_1, k &= \frac{K_v}{H^2} G_{6,km} \alpha_m + G_{7,km} \nabla \alpha_m \beta_n + \frac{1}{B} G_{8,kmn} (B_x \beta_n + B \nabla \beta_n) \alpha_m - \frac{1}{B} G_{6b,km} ((BK_h)_x \nabla \alpha_m + BK_h \Delta \alpha_m), \\
F_2, k &= \frac{A_v}{H^2} G_{1,km} \beta_m + G_{2,kmn} \nabla \beta_m + G_{3,kmn} \left( \nabla \beta_m + \frac{B_x}{B} \beta_m \right) \beta_n + g \zeta_x G_{4,k} - g \beta H G_{5,km} \nabla \alpha_m,
\end{align*}$$

(49)

$$F_3 = B H G_{9,m} \beta_m + Q,$$

(50)

with boundary conditions

$$\begin{align*}
G_{10,m} \alpha_m &= s_{sea} & \text{at } x = 0, \\
\frac{K_v}{H^2} G_{6,km} \alpha_m + G_{7,km} \nabla \alpha_m \beta_n + \frac{1}{B} G_{8,kmn} (B_x \beta_n + B \nabla \beta_n) \alpha_m &= 0 & \text{at } x = 0, k = 1, \ldots, M, \\
G_{6b,mn} \alpha_m &= 0 & \text{at } x = L, k = 0, \ldots, M, \\
\frac{A_v}{H^2} G_{1,km} \beta_m &= -g \zeta_x G_{4,k} & \text{at } x = L, k = 0, \ldots, M.
\end{align*}$$

(51)

In the following, we will only provide further specification for the equations without the boundary conditions. The numerical implementation of the boundary conditions follows trivially from this.

**Jacobian**

The Jacobian of $F$, required in the Newton-Raphson method can be written as

$$\frac{\partial F}{\partial y} = \begin{bmatrix}
\frac{\partial F_1}{\partial \alpha} \\
\frac{\partial F_1}{\partial \beta} \\
\frac{\partial F_1}{\partial \zeta}
\end{bmatrix}_{km}$$

(52)

These partial derivatives can be easily derived from the equations when considering the differential operators $\nabla$ and $\Delta$ as linear operators that can be specified without knowing the variable to take the derivative of. We can then write:

$$\frac{\partial \nabla y}{\partial y} = \nabla,$nabla
$$

$$\frac{\partial \Delta y}{\partial y} = \Delta.$nabla

It follows that (green-colored terms will be explained below)

$$\begin{align*}
\left( \frac{\partial F_1}{\partial \alpha} \right)_{km} &= \frac{K_v}{H^2} G_{6,km} + G_{7,kmn} \nabla \beta_n + \frac{1}{B} G_{8,kmn} (B_x \beta_n + B \nabla \beta_n) - \frac{1}{B} G_{6b,km} ((BK_h)_x \nabla \alpha_m + BK_h \Delta \alpha_m), \\
\left( \frac{\partial F_1}{\partial \beta} \right)_{km} &= G_{7,kmn} \nabla \alpha_m + \frac{1}{B} G_{8,kmn} \alpha_m (B_x + B \nabla), \\
\left( \frac{\partial F_1}{\partial \zeta} \right)_{km} &= 0,
\end{align*}$$

(53)

$$\begin{align*}
\left( \frac{\partial F_2}{\partial \alpha} \right)_{km} &= -g \beta H G_{5,km} \nabla, \\
\left( \frac{\partial F_2}{\partial \beta} \right)_{km} &= \frac{A_v}{H^2} G_{1,km} + G_{2,kmn} \nabla \beta_m + G_{2,kmn} \nabla G_{3,kmn} \left( \nabla \beta_m + \frac{B_x}{B} \beta_m \right) + G_{3,kmn} \beta_n \left( \nabla + \frac{B_x}{B} \right), \\
\left( \frac{\partial F_2}{\partial \zeta} \right)_{km} &= g G_{4,k},
\end{align*}$$

(54)

\(^1\)In Einstein’s notation, any index that appears twice in one term should be summed over, i.e. for some $n = 0, \ldots, N$, the term $\alpha_n \beta_n$ equals $\sum_{n=0}^{N} \alpha_n \beta_n$.\(^1\)
\[
\begin{align*}
\left( \frac{\partial F_1}{\partial \alpha} \right)_m &= 0, \\
\left( \frac{\partial F_1}{\partial \beta} \right)_m &= BHG_{\alpha,m}, \\
\frac{\partial F_3}{\partial \xi_x} &= 0.
\end{align*}
\]

**Matrix form of \( F \)**

The Newton-Raphson method also requires the function \( F(y) \). To ensure consistency in the numerical implementation of the Jacobian and the function \( F(y) \), \( F(y) \) is written as a matrix-vector product

\[
F(y) = \mathcal{M} y = \begin{bmatrix} \mathcal{M}_{1\alpha} & \mathcal{M}_{1\beta} & \mathcal{M}_{1\xi} \\ \mathcal{M}_{2\alpha} & \mathcal{M}_{2\beta} & \mathcal{M}_{2\xi} \\ \mathcal{M}_{3\alpha} & \mathcal{M}_{3\beta} & \mathcal{M}_{3\xi} \end{bmatrix} y.
\]

The matrix \( \mathcal{M} \) is very similar to the Jacobian and consists of all but the green terms in the Jacobian

\[
\begin{align*}
\mathcal{M}_{1\alpha,m} &= K_x \frac{K_v}{H^2} G_{6,km} + G_{7,kmn} \beta_n \nabla - \frac{1}{B} G_{6b,km} ((BK_h) \nabla + BK_h \Delta) \quad (56a) \\
\mathcal{M}_{1\beta,m} &= \frac{1}{B} G_{8,kmn} \alpha_m (B_x + BV) \quad (56b) \\
\mathcal{M}_{1\xi,k} &= 0, \quad (56c) \\
\mathcal{M}_{2\alpha,m} &= -g \beta HG_{5,km} \nabla, \quad (57a) \\
\mathcal{M}_{2\beta,m} &= \frac{A_x}{H^2} G_{1,km} + G_{2,kmn} \beta_n \nabla + G_{3,kmn} \left( \nabla \beta_m + \frac{B_x}{B} \beta_m \right), \quad (57b) \\
\mathcal{M}_{2\xi,k} &= g G_{4,k}, \quad (57c) \\
\mathcal{M}_{3\alpha,m} &= 0, \quad (58a) \\
\mathcal{M}_{3\beta,m} &= BHG_{9,m}, \quad (58b) \\
\mathcal{M}_{3\xi,k} &= 0. \quad (58c)
\end{align*}
\]

**b. Adaptive grid**

The above momentum, depth-averaged continuity and salinity equations and the boundary conditions only depend on the \( x \)-coordinate. This \( x \)-dependency is solved numerically on a grid. As the model domain needs to be sufficiently long so that the landward boundary does not affect the salt intrusion while still capturing the short length scales in salt wedge estuaries, the grid is non-equidistant and adapts to the solution in every step of the Newton-Raphson iteration.

The grid contains \( j_{\text{max}} \) grid cells of length \( \Delta x_j \) (\( j = 1, \ldots, j_{\text{max}} \)). To determine the size of each grid cell, we define a weight function \( \tilde{v}(x) \) according to the salinity gradient computed in the last iteration

\[
\begin{align*}
\tilde{v}(x) &= \left( \frac{\max(s_x(x)) + L^{-1}}{\sigma} \right)^n, \quad (59a) \\
v(x) &= \min \left( \tilde{v}(x), \min_{x'}(\tilde{v}(x')) \frac{\Delta x_{\text{max}}}{\Delta x_{\text{min}}} \right), \quad (59b)
\end{align*}
\]

where we choose \( n = 3 \) and the term \( L^{-1} \) (for domain size \( L \)) was arbitrarily chosen to prevent any division by zero. \( v(x) \) is determined at each \( x \)-coordinate by linear interpolation between the grid points. The parameters \( \Delta x_{\text{min}} \) and \( \Delta x_{\text{max}} \) are introduced to set maximum and minimum grid sizes and are used here to make sure that \( v(x) \) is of the same order of magnitude in the entire domain. This improves the convergence of the algorithm below.
The grid cell size is then computed as (for \( j = 1, \ldots, j_{\text{max}} - 1 \)):

\[
\Delta \bar{x}_{j+1} = \max(0.97 \Delta x_j, \min(1.03 \Delta x_j, \frac{\gamma}{v(x_j)})),
\]

\[
\Delta x_{j+1} = \max(\Delta x_{\text{min}}, \min(\Delta x_{\text{max}}, \Delta \bar{x}_{j+1})),
\]

\[
x_{j+1} = x_j + \Delta x_{j+1}
\]

and with

\[
\Delta x_1 = \max(\Delta x_{\text{min}}, \min(\Delta x_{\text{max}}, \frac{\gamma}{v(0)})).
\]

The first condition defines a temporary value of the grid spacing on the basis of the weight function \( v(x) \), corrected so that the change in grid spacing from one cell to the next is not more than 3 %. The second condition then applies a correction to ensure that the grid spacing is not smaller or bigger than prescribed minimum and maximum grid spacings \( \Delta x_{\text{min}} \) and \( \Delta x_{\text{max}} \). The value of \( \gamma \) determined iteratively so that \( \sum_{j=1}^{j_{\text{max}}} \Delta x_j = L \).

c. Discretisation

To discretise the equations it is chosen to use a fixed numerical scheme for each operator. The equations contain five types of operators:

1. local operators containing no derivatives of the state vector,
2. diffusive operators containing \( \Delta \),
3. advective matrix operators containing \( \nabla \) but no use of state vectors,
4. implicit advective tensor operators containing a state variable and implicit use of \( \nabla \) (i.e. \( \nabla \) without specifying the variable to take the gradient of),
5. explicit advective tensor operators containing the gradient (\( \nabla \)) of a state variable and no other gradient operators or state variables.

Local operators are seen as operators on the cell average and require no further explanation. The other operators are discussed below after first introducing the flux vector splitting method that we will use for some of the terms.

Flux vector splitting

As the equations are advection-diffusion equations with potentially dominant advection terms, central operators for advection may generate spurious oscillations. To prevent such oscillations, we will use upwind methods for advective terms wherever possible. To this end, the upwind direction of a matrix operator needs to be determined. Two main categories of methods exists for this purpose: Godunov’s method (also Riemann approach or flux difference splitting) and flux vector splitting method (FVS) (also Boltzmann approach). While many different implementations of both methods exist, as a first guideline, the Godunov method requires more effort for determining the upwind direction than the FVS method, while it may lead to better reproduction of shock waves (e.g Toro 1999). For our purposes, a FVS method suffices.

The FVS method works as follows. Consider a hyperbolic matrix operator \( A \) in an advective term \( A_{\vec{u}} \), for a vector \( \vec{u} \). Let \( K \) be a matrix of right-eigenvectors and \( \Lambda \) be a diagonal matrix with the corresponding eigenvalues, then

\[
A = K \Lambda K^{-1}.
\]

Multiplying the advective term by \( K^{-1} \) and defining \( \vec{v} = K^{-1} \vec{u} \) we obtain

\[
K^{-1} A_{\vec{u}} = K^{-1} K \Lambda K^{-1} \vec{u} = \Lambda \vec{v}.
\]

The matrix \( \Lambda \) is a real diagonal matrix, because \( A \) is hyperbolic. Therefore, it is easy to determine the upwind direction of \( \Lambda \). Let \( \Lambda^+ \), \( \Lambda^- \) be the matrix with positive and negative eigenvalues, and let \( \vec{v}^+_\alpha \), \( \vec{v}^-_\alpha \) be the gradients in corresponding upwind directions. Then, the above equation can be written as

\[
K^{-1} A_{\vec{u}} = \Lambda^+ \vec{v}^+_\alpha + \Lambda^- \vec{v}^-_\alpha.
\]
Next, multiplying the equation by $K$ and rewriting yields

$$Au_t = K \Lambda^+ K^{-1} u_t^+ + K \Lambda^- K^{-1} u_t^-.$$ 

Thus we have expressed the advective terms in terms of derivatives in specific upwind directions.

**Diffusive operators**

The equations feature one diffusive operator: $G_{6b,km} K_h \Delta \alpha_{m}^{i+1}$. This is discretised using a second-order central method, i.e.

$$(G_{6b,km} K_h \Delta \alpha_{m}^{i+1})^j \approx G_{6b,km} K_h \frac{1}{2(\Delta x_{j+1} + \Delta x_j)} \left( \frac{\alpha_m^{i+1,j+1} - \alpha_m^{i+1,j}}{\Delta x_{j+1}} - \frac{\alpha_m^{i+1,j} - \alpha_m^{i+1,j-1}}{\Delta x_j} \right).$$

(62)

As this operator is not used on the boundaries, adapted versions of this scheme for the boundaries are not necessary.

**Advective matrix operators**

Our equations contain one advective matrix term, which is for the baroclinic pressure: $g \beta H G_{5,km} \nabla \alpha_{m}$. It turns out that this operator is not hyperbolic due to the specific combination of non-orthogonal eigenfunctions for $u$ and $s$ in the Galerkin coefficient matrix $G_5$. Therefore, the FVS method introduced above cannot be applied and we resort to a central scheme:

$$(g \beta H G_{5,km} \nabla \alpha_{m}^{i+1})^j = g \beta H \Delta x_j \frac{\alpha_m^{i+1,j+1} - \alpha_m^{i+1,j-1}}{\Delta x_j + \Delta x_{j+1}}.$$

(63)

At the boundary points $j = 0, j_{max}$, this is replaced by a second-order scheme toward the interior following a second-order symmetry preserving scheme (Veldman and Lam 2008), i.e.

$$(g \beta H G_{5,km} \nabla \alpha_{m}^{i+1})^j = \begin{cases} g \beta H \frac{3 \alpha_m^{i+1,max} - 4 \alpha_m^{i+1,1} + \alpha_m^{i+1,0}}{\Delta x_1 + \Delta x_2} & (j = j_{max}), \\ g \beta H \frac{3 \alpha_m^{i+1,0} - 4 \alpha_m^{i+1,j} + \alpha_m^{i+1,max}}{\Delta x_1 + \Delta x_2} & (j = 0). \end{cases}$$

(64)

**Implicit advective tensor operators**

We describe implicit advective tensor operators as terms of the type $G_{7,kmn} \beta_i \nabla \alpha_{m}^{i+1}$, i.e. where the term following the gradient, $\alpha_m^{i+1}$, is implicit. These terms are discretised using the FVS scheme. However, the eigenvectors and eigenvalues should now be computed for the matrix $G_{7,kmn}$. If this were done by applying an FVS method directly to $G_{7,kmn} \beta_i$, this would become computationally expensive as this needs to be repeated for every grid point in $x$-direction and for every Newton-Raphson iteration. The computational effort is greatly reduced by first determining the eigenvalues and eigenvectors of $G_{7,kmn}$ for fixed values of $n$ and then applying flux splitting. To this end, we define $K_{kmn}^{(km)}$, $\Lambda_{kmn}^{(km)}$ as the elements of the tensors with right eigenvectors and corresponding eigenvalues with respect to the dimensions belonging to indices $k$ and $m$. In other words, they are the right eigenvectors and eigenvalues of the matrices $G_{7,km0}, \ldots, G_{7,kmM}$. We thus write

$$G_{7,kmn} \beta_i^{i} = \left( K_{kmn}^{(km)} \Lambda_{kmn}^{(km)} K_{kmn}^{(km),-1} \right)_{kmn}^{i} \beta_i^{i}$$

(65)

The computational effort of this is to compute the eigenvalues and eigenvectors of an $M \times M$ matrix $M$ times without any dependence on $x$ or the Newton-Raphson iteration. Only the matrix-vector product above needs to be repeated for each grid point and every iteration.

To define the splitting, consider one fixed number $n = n_0$ (instead of a sum over $n$) and note that the above can be written as $K_{q4q2n0}^{(km)} \Lambda_{q4q2n0}^{(km)} \beta_i^{i} K_{q4q2n0}^{(km),-1}$. The upwind direction should thus be based on the sign of $\Lambda_{q4q2n0}^{(km)}$. This is
written as

$$G_{7,kmn}^i \beta_n^i = \frac{1}{2} (K^{(km)} \Lambda^{(km)}, + K^{(km)}, -1)_{kmn} (\beta^i + |\beta^i|)_n + \frac{1}{2} (K^{(km)} \Lambda^{(km)}, - K^{(km)}, -1)_{kmn} (\beta^i - |\beta^i|)_n$$

$$= (G_{7,kmn}^i \beta_n^i)^+, \text{ positive direction}$$

$$= (G_{7,kmn}^i \beta_n^i)^-, \text{ negative direction}$$

For the discretisation we use a symmetry preserving upwind scheme (Veldman and Lam 2008) resulting in

$$\left( G_{7,kmn}^i \nabla \alpha_m^{i+1} \right)^j = \left( G_{7,kmn}^i \nabla \alpha_m^{i+1} \right)^j + \frac{3 \alpha_m^{i+1,j} - 4 \alpha_m^{i+1,j-1} + \alpha_m^{i+1,j-2}}{\Delta x_j + \Delta x_{j+1}} + \left( G_{7,kmn}^i \beta_n^i \right) - \frac{3 \alpha_m^{i+1,j} + 4 \alpha_m^{i+1,j+1} - \alpha_m^{i+1,j+2}}{\Delta x_j + \Delta x_{j+1}}$$

(67)

At the points $j = 1, j_{max} - 1$, this is replaced by a first-order upwind scheme

$$\left( G_{7,kmn}^i \nabla \alpha_m^{i+1} \right)^j = \left( G_{7,kmn}^i \nabla \alpha_m^{i+1} \right)^j + \frac{\alpha_m^{i+1,j} - \alpha_m^{i+1,j-1}}{\frac{1}{2} (\Delta x_j + \Delta x_{j+1})} + \left( G_{7,kmn}^i \beta_n^i \right) - \frac{-\alpha_m^{i+1,j} + \alpha_m^{i+1,j+1}}{\frac{1}{2} (\Delta x_j + \Delta x_{j+1})}$$

(68)

At the boundary points $j = 0, j_{max}$, the upwinding direction is ignored and replaced by a second-order scheme toward the interior, i.e.

$$\left( G_{7,kmn}^i \nabla \alpha_m^{i+1} \right)^j = \begin{cases} 
G_{7,kmn}^i \beta_n^i \frac{3 \alpha_m^{i+1,j} + 4 \alpha_m^{i+1,j+1} - \alpha_m^{i+1,j+2}}{\Delta x_{j+1} + \Delta x_{j+2}} & (j = j_{max}), \\
G_{7,kmn}^i \beta_n^i \frac{-3 \alpha_m^{i+1,j} + 4 \alpha_m^{i+1,j+1} - \alpha_m^{i+1,j+2}}{\Delta x_{j+1} + \Delta x_{j+2}} & (j = 0).
\end{cases}$$

(69)

An exception is made for the implicit advective tensor term $G_{8,kmn}^i \nabla \beta_n^i \alpha_m^{i+1}$, which is not hyperbolic. This term is discretised using a central method similar to the advective matrix operator.

**EXPLICIT ADVECTIVE TENSOR OPERATORS**

We describe explicit advective tensor operators as terms of the type $G_{8,kmn}^i \nabla \beta_n^i \alpha_m^{i+1}$. The most consistent discretisation of this term would be similar to the implicit advective tensor operator, i.e. splitting the flux on the basis of the eigenvalues of $G_{8,kmn}$ and the sign of $\alpha_m^{i+1}$ and then using an upwind scheme for $\nabla \beta_n^i$. However, as $\alpha_m^{i+1}$ is implicit, the upwind direction cannot be determined. It has been tried to determine the upwind direction on the basis of the sign of the explicit $\alpha_m$, but first experiments showed that this did not lead to convergent results. Hence, this term is discretised without flux splitting using a second-order central scheme as

$$\left( G_{8,kmn}^i \nabla \beta_n^i \alpha_m^{i+1} \right)^j = G_{8,kmn}^i \beta_n^i \frac{\beta_n^{i+1,j} - \beta_n^{i-1,j}}{\Delta x_j} \alpha_m^{i+1,j}$$

(70)

in the interior and a first-order scheme on the boundaries

$$\left( G_{8,kmn}^i \nabla \beta_n^i \alpha_m^{i+1} \right)^j = \begin{cases} 
G_{8,kmn}^i \beta_n^i \frac{\beta_n^{i+1,j} - \beta_n^{i+1,j-1}}{\Delta x_{j+1}} \alpha_m^{i+1,j} & (j = j_{max}), \\
G_{8,kmn}^i \beta_n^i \frac{-3 \beta_n^{i+1,j} + 4 \beta_n^{i+1,j+1} - \beta_n^{i+1,j+2}}{\Delta x_{j+1}} \alpha_m^{i+1,j} & (j = 0).
\end{cases}$$

(71)

d. Solution to the Newton-Raphson method

To solve the Newton-Raphson step, the above equations are implemented in terms of one state vector that contains the unknowns of all the modes and grid cells. This state vector $y^i$ is defined as

$$y^i = [\alpha_m^{i,0}, \alpha_m^{i,1}, \ldots, \alpha_m^{i, M}, \beta_n^{i,0}, \beta_n^{i,1}, \ldots, \beta_n^{i, M}, \xi_x^{i,0}, \xi_x^{i,1}, \ldots, \xi_x^{i, j_{max}}]^T$$

(72)

and has length $(2M + 1)j_{max}$. 


The discretised expressions for $\mathcal{A}$ and $\frac{\partial F(y)}{\partial y}$ are both sets of five $(2M + 1) \times (2M + 1)$ matrices for each grid point; one matrix as coefficient for each implicit element at $j-2$, $j-1$, $j$, $j+1$, and $j+2$. For example, the matrix for elements at $j$ is obtained by gathering the coefficients in front of any implicit variable at the grid point $j$. The thus obtained matrices are written as $\mathcal{A}^{i,j}_{(j+q)}\left(\frac{\partial F(y)}{\partial y}\right)^j_{(j+q)}$ for $q = -2, -1, 0, 1, 2$. These matrices are then written as a block-penta-diagonal matrix using block-storage notation in numpy (see documentation of scipy.linalg.solve_banded). This matrix has $12M + 5$ diagonals of maximum length $(2M + 1)_{\text{max}}$.

In the numerical implementation this is done for each operator. Thus, for each operator in the equations, the coefficients are gathered on the basis of the numerical discretisation scheme and then formed into a block-penta-diagonal matrix using block-storage notation in numpy (see documentation of scipy.linalg.solve_banded). The block-penta-diagonal matrices for each operator are then added to find $\mathcal{M}$ and $\frac{\partial F}{\partial y}$ and used to solve each Newton-Raphson step.

5. Continuation method

The Newton-Raphson iteration requires an initial solution to start the iteration. The first time the Newton-Raphson iteration is used, it is started from a trivial initial state. This leads to a converging result as long as the model experiment is only weakly nonlinear (i.e. momentum advection terms and baroclinic pressure are relatively weak). To obtain the solution for other cases, the model is run several times starting from parameter settings leading to weakly nonlinear results and gradually changing the parameter settings. This method is called a continuation procedure. Below we discuss the implementation of the continuation procedure for gradually changing eddy viscosity/eddy diffusivity.

Let $p$ define the eddy viscosity, eddy diffusivity, and partial slip parameter, linked through Eqs. (16)-(17) and let $y^0$ be a solution for parameter value $p^0$ obtained by a Newton-Raphson iteration as described above. The solutions for different values of $p$ are computed using a continuation procedure. This procedure starts from $y^0$, then extrapolates this to an estimated solution for a different $p^1$ (the predictor) and then starts another Newton-Raphson iteration to make this estimate more accurate (the corrector). The predictor reads as (in the notation of Eq. (52))

$$
F_1(y^0)\Delta y = -F_p(y^0)\Delta p.
$$

We fix the step $\Delta p$ and compute $\Delta y$. To this end we need the derivative of $F$ with respect to $p$. Using the notation of Eq. (52), this reads as

$$
\left(\frac{\partial F_1}{\partial p}\right)_k = \frac{1}{H^2}G_{6,km}\alpha_m,
\left(\frac{\partial F_2}{\partial p}\right)_k = \frac{1}{H^2}G_{1,km}\beta_m.
$$

The corrector is given by the Newton-Raphson iteration. We do not extend the iteration to include $\Delta p$ as an unknown as in usual pseudo-arclength continuation but instead fix $\Delta p$.

6. Summary of the numerical procedure

The total numerical routine is of the following form

1. Compute the eigenvalues $\lambda_m$ ($m = 0, \ldots, M$) and eigenfunctions $f_m$, $g_m$.
2. Compute $G_i$ ($i = 1, \ldots, 10, 6b$).
3. Set the initial state $y^0$ equal to zero or to a previously obtained solution.
4. Continuation procedure:
   (a) Compute continuation step (not in the first iteration).
   (b) Compute Newton-Raphson iteration:
      i. Adapt the numerical grid to the previous solution and linearly interpolate $y^i$ to the new grid.
      ii. Compute the operators in the discretised equations.
      iii. Compute $F$ and $\frac{\partial F}{\partial y}$ based on the computed terms.
iv. Solve a Newton-Raphson step to compute $y^{i+1}$.

v. Compute the norm $\frac{||\alpha^{i+1,j}(\Delta t^{j+1}+\Delta t^{j+1})-\alpha^{i,j}(\Delta t^{j+1}+\Delta t^{j+1})||_2}{||\alpha^{i+1,j}(\Delta t^{j+1}+\Delta t^{j+1})||_2}$ and do another iteration if this is smaller than $10^{-5}$.

(c) Post-calculate a decomposition of the equations (see main text).

(d) Save the result.

References
