MEDM Lab
Marine Ecosystem Dynamics Modeling

The Next Generation FVCOM: Algorithms and Progresses

Siqi Li, Changsheng Chen, Jianhua Qi, Geoffrey Cowles

University of Massachusetts School of Marine Science
New Bedford, MA 02744

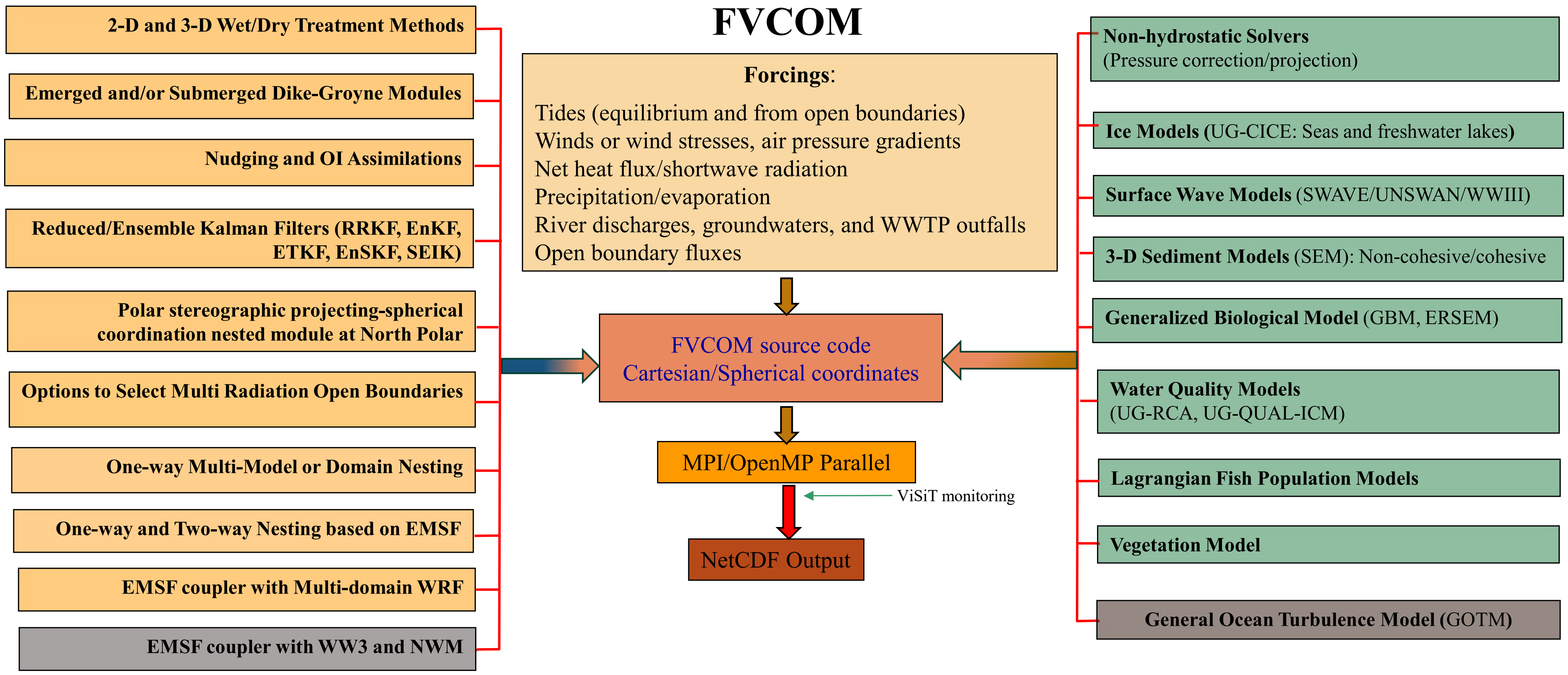
Version 6.0
FVCOM

http://fvcom.smast.umassd.edu/
http://www.fvcom.org

Outline

- **A brief review of current FVCOM mode-split and semi-implicit solvers**.
- **The next-generation FVCOM**: **Hybrid Eulerian-Lagrangian, semi-implicit/implicit time integration**. **In particular,**
  - The second-order backward difference time integration scheme (BDF2).
  - The operator-integration-factor splitting (OIFS) algorithm for horizontal advection.
  - Fully implicit schemes for vertical advection.
  - Semi-implicit scheme for water elevation.
- **Update of MPI-OpenMP combined parallelization**.

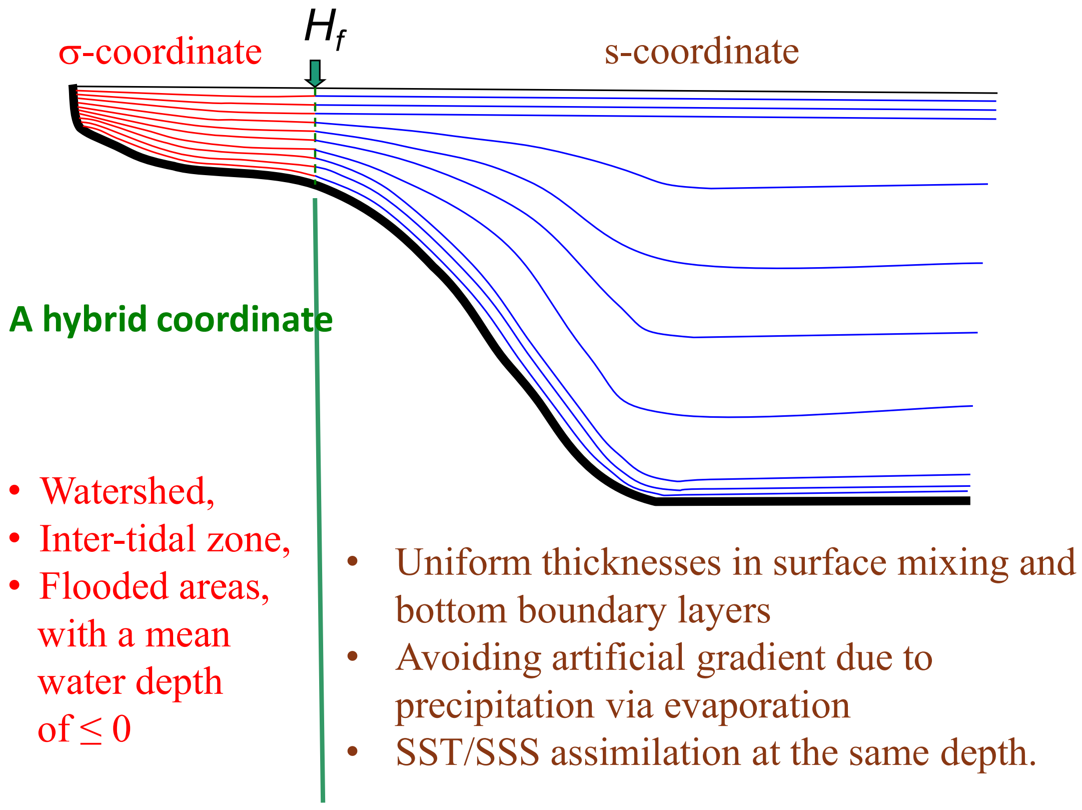
FVCOM


2-D and 3-D Wet/Dry Treatment Methods
Emerged and/or Submerged Dike-Groyne Modules
Nudging and OI Assimilations
Reduced/Ensemble Kalman Filters (RRKF, EnKF, ETKF, EnSKF, SEIK)
Polar stereographic projecting-spherical coordination nested module at North Polar
Options to Select Multi Radiation Open Boundaries
One-way Multi-Model or Domain Nesting
One-way and Two-way Nesting based on EMSF
EMSF coupler with Multi-domain WRF
EMSF coupler with WW3 and NWM
Forcings:
Tides (equilibrium and from open boundaries)
Winds or wind stresses, air pressure gradients
Net heat flux/shortwave radiation
Precipitation/evaporation
River discharges, groundwaters, and WWTP outfalls
Open boundary fluxes
FVCOM source code
Cartesian/Spherical coordinates
MPI/OpenMP Parallel
ViSiT monitoring
NetCDF Output
Non-hydrostatic Solvers (Pressure correction/projection)
Ice Models (UG-CICE: Seas and freshwater lakes)
Surface Wave Models (SWAVE/UNSWAN/WWIII)
3-D Sediment Models (SEM): Non-cohesive/cohesive
Generalized Biological Model (GBM, ERSEM)
Water Quality Models (UG-RCA, UG-QUAL-ICM)
Lagrangian Fish Population Models
Vegetation Model
General Ocean Turbulence Model (GOTM)

Key: 🟩 Existing Model Modules | 🟧 Existing Technology Modules | ▨ Library linking to other models | 🟩 **New Modules**

Vertical Coordinates

Generalized terrain-following coordinate transformations based on the approach described in Pietrzak et al. (2002) (*Ocean Modeling*, 3, 173-205).



- Watershed,
- Inter-tidal zone,
- Flooded areas, with a mean water depth of ≤ 0

- Uniform thicknesses in surface mixing and bottom boundary layers
- Avoiding artificial gradient due to precipitation via evaporation
- SST/SSS assimilation at the same depth.

FVCOM Solvers

Time integration:

1) Mode-splitting and 2) semi-implicit solvers

Mode-splitting:

2-D barotropic mode (ΔT_E) and 3-D baroclinic mode (ΔT_I), a recommended ratio of $\Delta T_I/\Delta T_E$ is 10.

Advantages: Easily set; able to run a 2-D model separately; does not depend on libraries of matrix solvers, etc.

Disadvantage: 2D-3D mode adjustments could cause numerical oscillations in the deep ocean.

Semi-implicit:

FVCOM is solved in a single time step (ΔT_I)

Advantage: faster, no adjustment needed, suitable for basin and global applications

Disadvantage: Requires PETSc (scalable sparse matrix solver library).

Remarks on FVCOM Performances

Merits

- Unstructured-grid.
- Volume conservations.
- Hydrostatic/non-hydrostatic dynamics.
- Options for multi-finite-volume advection and time integration schemes.
- Simple module structures of the source.
- Well-validated via various benchmark test problems through inter-model comparisons with ROMS.
- Computationally efficient as finite-difference models.
- Coupled fully with ice, sediments, surface waves, vegetation, WRF, and ecosystem/water quality models.
- Multiple choices in data assimilation, including nudging, OI, and various Kalman filters.
- Successful application to estuaries, coastal to global ocean.
- Allows a straight wall in the terrain-following coordinates

Challenging

- Both mode-split and semi-implicit integration solver is constrained by the Courant-Friedrichs-Lewy (CFL) condition. In general,

$$u\frac{\Delta t}{\Delta s} \le C_{max}$$

u: the magnitudes of the velocity
Δt: the internal time step
Δs: the horizontal resolution.
$C_{max} \sim 1$, when either mode-splitting or semi-explicit is used.

When applying FVCOM to an estuary with a horizontal resolution of a few meters, the time step is required to be very small, especially in a narrow water passage with strong flow.

Proposed solutions

Solution 1:

- Adjustable Δt based on the magnitude of the velocity.

Advantages:

- Simple to be implemented, since the adjustment can be done based on the CFL number constraint.
- Ensure numerical stability during unusual extreme storm events.

Disadvantages:

- In most cases, the model blows up at a local single cell or node, and the adjustment is made over the entire domain, affecting computational efficiency.
- The time step is still constrained by the CFL number.

Solution 2:

- Upgrade by implementing an implicit or semi-Lagrangian solver.

Advantages:

- The time step is not constrained by the CFL condition or allows a larger CFL number.
- Significantly improvement in numerical stability.

Disadvantages:

- Enhancing numerical diffusion.
- Affecting computational efficiency.
- Although the Lagrangian method has no numerical dispersion, the inaccuracy of inverse particle-tracking and interpolation could lead to numerical bias that may be larger than dispersion.

An Illustrative Example of the Eulerian-Lagrangian Method

The diffusion equation for a concentration C is given as

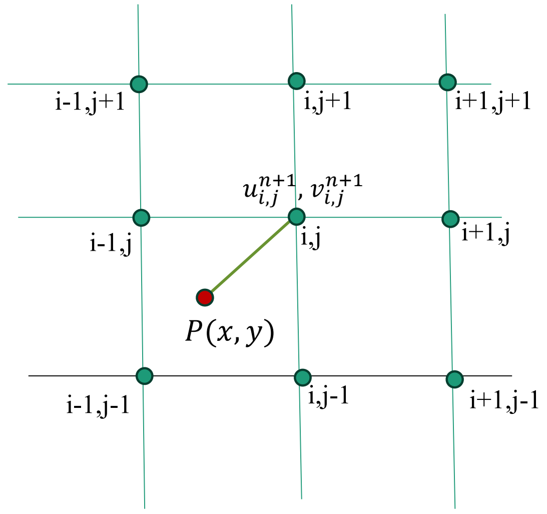
$$\frac{dC}{dt} = K_H\left(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2}\right)$$

Discretizing using Lagrangian inverse tracking on the left side and Eulerian central difference scheme on the right side, we have

$$\frac{C_{i,j}^{n+1} - C^n(P)}{\Delta t} = K_H\left(\frac{C_{i+1,j}^n - 2C_{i,j}^n + C_{i-1.j}^n}{\Delta x^2} + \frac{C_{i,j-1}^n - 2C_{i,j}^n + C_{i,j+1}^n}{\Delta y^2}\right)$$

Then, C at the ith and jth node can be determined by

$$C_{i,j}^{n+1} = C^n(P) + \frac{K_H \Delta t}{\Delta x^2}(C_{i+1,j}^n - 2C_{i,j}^n + C_{i-1.j}^n) + \frac{K_H \Delta t}{\Delta y^2}(C_{i,j-1}^n - 2C_{i,j}^n + C_{i,j+1}^n)$$



The key point is how to determined $C^n(P)$. Two steps:
1) Find the location of P using the Lagrangian inverse tracking.
2) Determine C at P at the time step n by interpolation.

$$P(x,y) = \left[\left(x_{i,j} - u_{i,j}^{n+1}\Delta t\right), \left(y_{i,j} - v_{i,j}^{n+1}\Delta t\right)\right], \begin{cases} \dfrac{dx}{dt} = -u_{i,j}^{n+1} \Rightarrow x_p = x_{i,j} - u_{i,j}^{n+1}\Delta t \\ \dfrac{dy}{dt} = -v_{i,j}^{n+1} \Rightarrow y_p = x_{i,j} - v_{i,j}^{n+1}\Delta t \end{cases}$$

After using the velocity at the time step (n+1) backward to find P, one can use the interpolation method to determine $C^n(P)$ from surrounding nodes.

Merits:

- No advection equations are involved, so it does not have numerical dispersion!
- It is numerically stable with a larger CFL number.
- Allowing a larger time step in the numerical integration.

Challenging:

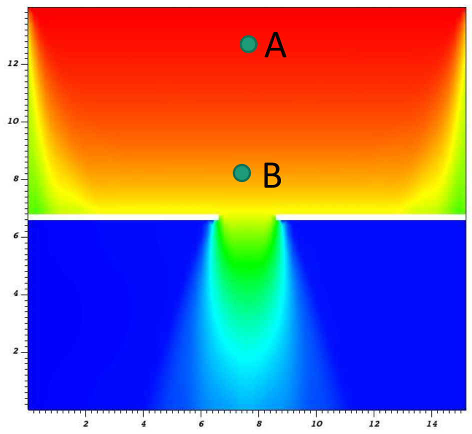
- Numerical diffusions due to inaccurate Lagrangian particle tracking and interpolation could be significant. Implementing the high-order particle tracking and interpolation methods, like the 4th-order Runge-Katta method or the Discontinuous Galerkin method, could be helpful.
- Computational efficiency.
- Boundary treatments.
- Particle path lines often intersect and/or disperse, making it difficult to resolve the fluid interactions accurately.
- Ensuring particle tracking accuracy in sharp gradient areas is crucial, as it can significantly impact the overall simulation results.
- Since the horizontal velocity is usually one or two orders of magnitude larger than the vertical velocity, the characteristic trajectories in a 3-D space are at distinct scales of the motion in the horizontal and vertical.

MEDM Lab Marine Ecosystem Dynamics Modeling

Example: Semi-Lagrangian scheme

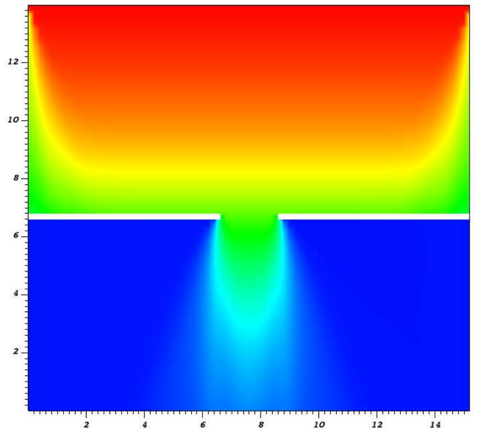
FVCOM-SWAVE
Explicit

$\Delta t = 5\ sec$

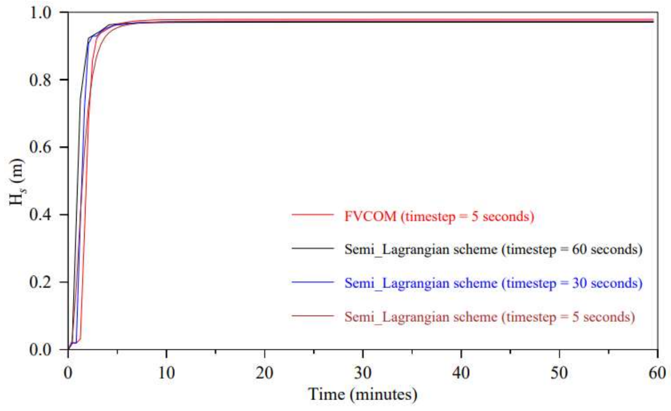


SWAVE
Semi-Lagrangian

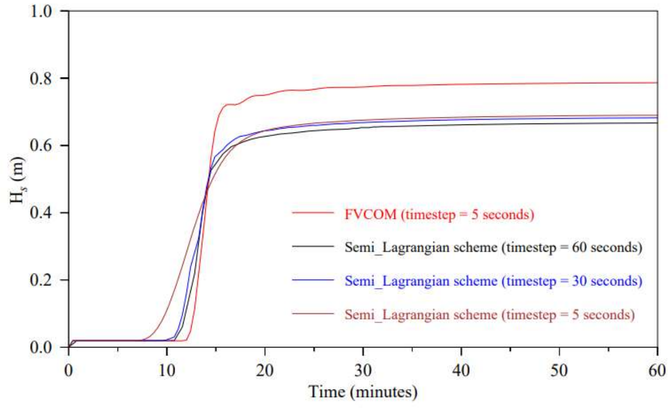
$\Delta t = 30\ sec$



Site A



Site B



The time required
(1 processor)

Schemes	Time (hours)
FVCOM-SWAVE explicit	0.83
SWAVE-Semi-Lag (60 sec)	3.0
SWAVE-Semi-Lag (30 sec)	6.0
SWAVE Semi-Lag (5 sec)	36

Current versions of FVCOM

The governing equations in the generalized spherical coordinates are given as

$$\frac{\partial uJ}{\partial t} + \frac{1}{R\cos\varphi}\left[\frac{\partial u^2 J}{\partial\lambda} + \frac{\partial u(v\cos\varphi)J}{\partial\varphi}\right] + \frac{\partial\omega u}{\partial r} - \frac{uvJ}{R}\tan\varphi + \frac{wuJ}{R} - fvJ + f'wJ$$

$$= -\frac{gJ}{R\cos\varphi}\frac{\partial\zeta}{\partial\lambda} - \frac{J}{\rho_o R\cos\varphi}\frac{\partial P_a}{\partial\lambda} - \frac{gJ}{\rho_o R\cos\varphi}\int_r^0 J\left(\frac{\partial\rho}{\partial\lambda} + \frac{\partial\rho}{\partial r}\frac{\partial r}{\partial\lambda}\right)dr - \frac{1}{\rho_o}\left(\frac{1}{R\cos\varphi}\frac{\partial qJ}{\partial\lambda} + \frac{\partial qA_1}{\partial r}\right) + \frac{\partial}{\partial r}\left(\frac{K_m}{J}\frac{\partial u}{\partial r}\right) + JF_u$$

$$\frac{\partial vJ}{\partial t} + \frac{1}{R\cos\varphi}\left[\frac{\partial uvJ}{\partial\lambda} + \frac{\partial v^2\cos\varphi J}{\partial\varphi}\right] + \frac{\partial\omega v}{\partial r} + \frac{u^2 J}{R}\tan\varphi + \frac{wvJ}{R} + fuJ$$

$$= -\frac{gJ}{R}\frac{\partial\zeta}{\partial\varphi} - \frac{J}{\rho_o R}\frac{\partial P_a}{\partial\varphi} - \frac{gJ}{\rho_o R}\int_r^0 J\left(\frac{\partial\rho}{\partial\varphi} + \frac{\partial\rho}{\partial r}\frac{\partial r}{\partial\varphi}\right)dr - \frac{1}{\rho_o}\left(\frac{1}{R}\frac{\partial qJ}{\partial\varphi} + \frac{\partial qA_2}{\partial r}\right) + \frac{\partial}{\partial r}\left(\frac{K_m}{J}\frac{\partial v}{\partial r}\right) + JF_v$$

$$\frac{\partial w}{\partial t} + \frac{1}{R\cos\varphi}\left[\frac{\partial uwJ}{\partial\lambda} + \frac{\partial(v\cos\varphi)wJ}{\partial\varphi}\right] + \frac{\partial\omega w}{\partial r} - \frac{(u^2+v^2)J}{R} - f'uJ = -\frac{1}{\rho_o}\frac{\partial q}{\partial r} + \frac{\partial}{\partial r}\left(\frac{K_m}{J}\frac{\partial w}{\partial r}\right) + JF_w$$

$$\frac{\partial J}{\partial t} + \frac{1}{R\cos\varphi}\left[\frac{\partial u}{\partial\lambda} + \frac{\partial(v\cos\varphi)J}{\partial\varphi}\right] + \frac{\partial\omega}{\partial r} = 0$$

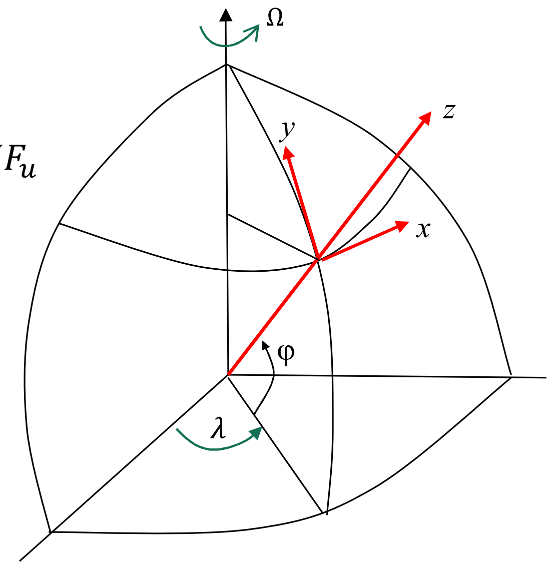
$$\frac{\partial TJ}{\partial t} + \frac{1}{R\cos\varphi}\left[\frac{\partial(uJ)T}{\partial\lambda} + \frac{\partial(v\cos\varphi)JT}{\partial\varphi}\right] + \frac{\partial\omega T}{\partial r} = \frac{\partial}{\partial r}\left(\frac{K_h}{J}\frac{\partial T}{\partial r}\right) + JF_T$$

where $J = \frac{\partial z}{\partial r} = 1/\left(\frac{\partial r}{\partial z}\right)$

$$\begin{cases} r(\zeta) = 0 \\ r(-H) = -1 \end{cases}$$

$$\frac{\partial SJ}{\partial t} + \frac{1}{R\cos\varphi}\left[\frac{\partial(uJ)S}{\partial\lambda} + \frac{\partial(v\cos\varphi)JS}{\partial\varphi}\right] + \frac{\partial\omega S}{\partial r} = \frac{\partial}{\partial r}\left(\frac{K_h}{J}\frac{\partial S}{\partial r}\right) + JF_s$$

$$\rho = \rho(T,S,P)$$



Grid designs

Grid: Non-overlapping unstructured-grid triangles

$u, v\cos\varphi$: at centroids

$\zeta, \omega, w, T, S, \rho$: at nodes

N: The total centroid number

M: The total node number

$[x(i), y(i)], i = 1:N$

$[x_n(j), y_n(j)], j = 1:M$

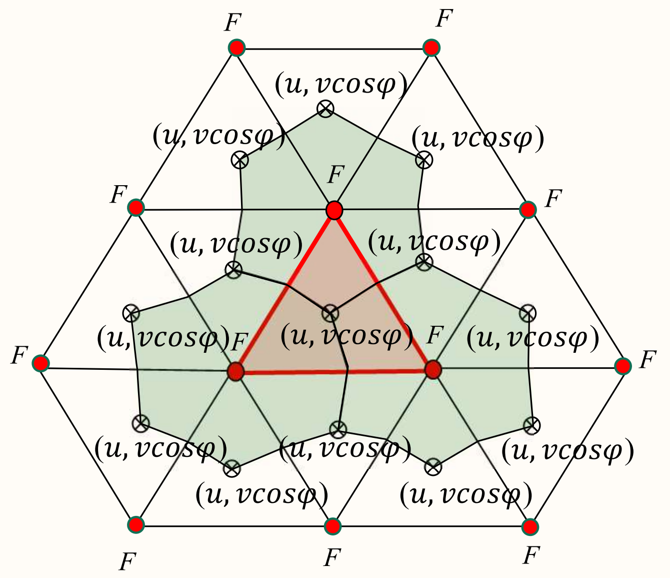
On the ith triangle, three nodes are defined as

$N_i(\hat{j}), \hat{j}$=1:3; counted clockwise.

On the jth node, the total number of the surrounding triangles connected to this node is defined as $NT(j)$, counted by $NB_j(m)$, and

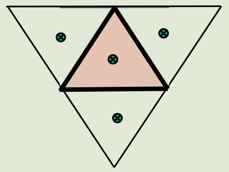
$NB_j(\mathrm{m}) = 1: NT(j)$

The horizontal

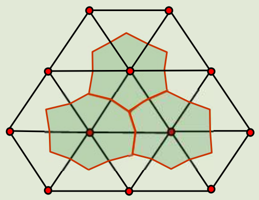


$F: \zeta, \omega, w, T, S, \rho, q^2, q^2l, \varepsilon,$

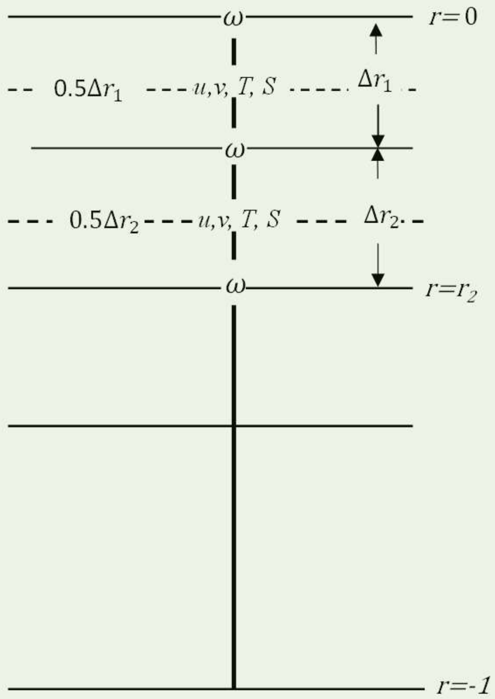
Cell-centered



Cell vertex median



The vertical



Avoid the errors in forcings due to various triangle sizes

Current versions of FVCOM

To simplify the description of numerical algorithms, we use the hydrostatic momentum equations, for example.

$$\frac{\partial uD}{\partial t} + R_u = \frac{1}{D}\frac{\partial}{\partial r}\left(\frac{K_m}{J}\frac{\partial u}{\partial r}\right)$$

$$\frac{\partial vD}{\partial t} + R_v = \frac{1}{D}\frac{\partial}{\partial r}\left(\frac{K_m}{J}\frac{\partial v}{\partial r}\right)$$

Here,

$$R_u = FX_{ADV} + FX_{\mathrm{COR}} + FX_{\mathrm{SP}} + FX_{\mathrm{BCP}} + FX_{\mathrm{HVIS}} + FX\mathrm{c}$$

$$R_v = FY_{ADV} + FY_{\mathrm{COR}} + FY_{\mathrm{SP}} + FY_{\mathrm{BCP}} + FY_{\mathrm{HVIS}} + FY\mathrm{c}$$

$\mathrm{FX_{ADV}}$, $\mathrm{FY_{ADV}}$	Zonal and meridional components of advection terms
$\mathrm{FX_{COR}}$, $\mathrm{FY_{COR}}$	Zonal and meridional components of the Coriolis terms.
$\mathrm{FX_{COR}}$, $\mathrm{FY_{COR}}$	Zonal and meridional components of barotropic pressure gradient terms.
$\mathrm{FX_{SP}}$, $\mathrm{FY_{SP}}$	Zonal and meridional components of barotropic surface pressure forcing terms
$\mathrm{FX_{BCP}}$, $\mathrm{FY_{BCP}}$	Zonal and meridional components of baroclinic pressure gradient forcing terms
$\mathrm{FX_{HVIS}}$, $\mathrm{FY_{HVIS}}$	Zonal and meridional components of horizontal diffusion terms
$\mathrm{FX_C}$, $\mathrm{FY_C}$	The zonal and meridional components of curvature terms in the spherical coordinates.

In the current version of FVCOM, the momentum equations are solved numerically in two steps. In particular,

$$\frac{(uD)^{n+1} - (uD)^{*} + (uD)^{*} - (uD)^{n}}{\Delta t} + R_u = \frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial u}{\partial\sigma}\right)$$

$$\frac{(vD)^{n+1} - (vD)^{*} + (vD)^{*} - (vD)^{n}}{\Delta t} + R_v = \frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial v}{\partial\sigma}\right)$$

$$\frac{(uD)^{*} - (uD)^{n}}{\Delta t} + R_u = 0$$

$$\frac{(vD)^{*} - (vD)^{n}}{\Delta t} + R_v = 0$$

Step 1: Explicit: the 2nd-order accurate 4th Runge-Katta scheme. Constrained by the CFL condition

$$\frac{(uD)^{n+1} - (uD)^{*}}{\Delta t} = \frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial u}{\partial\sigma}\right)$$

$$\frac{(vD)^{n+1} - (vD)^{*}}{\Delta t} = \frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial v}{\partial\sigma}\right)$$

Step 2: Implicit: Absolute stable, no constrained by the CFL condition

1. Explicit method

The advection terms are calculated by an upwind scheme with a modified Runge-Kutta time-stepping scheme for the 2-D mode and an upwind or a modified Runge-Kutta scheme for the 3-D mode (Kobayashi, 1999).

$$u_i(x',y') = \phi_i^u(x',y') = u_i(x_{ic},y_{ic}) + a_i^u(x'-x_{ic}) + b_i^u(y'-y_{ic})$$

$$v_i(x',y') = \phi_i^v(x',y') = v_i(x_{ic},y_{ic}) + a_i^v(x'-x_{ic}) + b_i^v(y'-y_{ic})$$

a_i^u, a_i^v, b_i^u, b_i^v are determined by a least-square method with velocity values at 4 cell centered points. x_c and y_c are location of the triangular central cell.

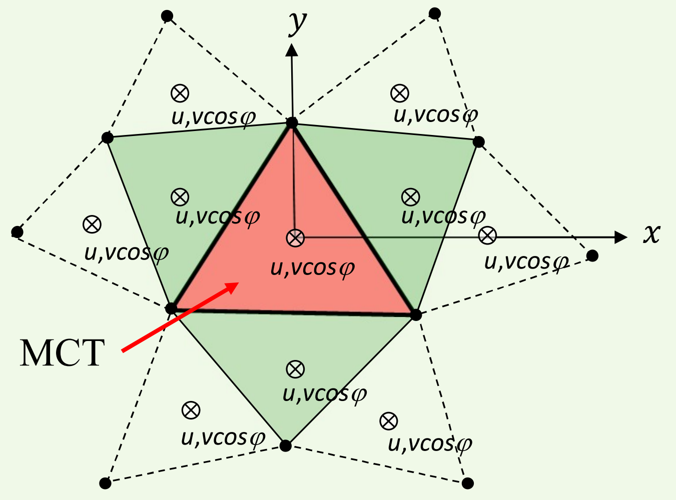
The x and y components of the horizontal advection are calculated numerically by

$$ADVU = \sum_{m=1}^{3} u_{im} D_m v_{nm} l_m \qquad ADVV = \sum_{m=1}^{3} v_{im} D_m v_{nm} l_m$$

u_{im}, v_{im}: velocities on edge m, v_{nm}: normal velocity on edge, l_m: edge length, D_m is depth

$$u_{im} = \begin{cases} \phi_i^u(x_{ic},y_{ic}), & v_{nm} < 0 \\ \phi_{NB_i(m)}^u(x_{im},y_{im}), & v_{nm} \geq 0 \end{cases} \qquad v_{im} = \begin{cases} \phi_i^v(x_{ic},y_{ic}), & v_{nm} < 0 \\ \phi_{NB_i(m)}^v(x_{im},y_{im}), & v_{nm} \geq 0 \end{cases}$$

x_{im}, y_{im}: cell-centered point of the surrounding triangle,
$NB_i(m)$: the neighbor triangle, (x_{ic}, y_{ic}): the centered location of the ith triangle.



$$x = R\cos\varphi(\lambda - \lambda_o); y = R(\varphi - \varphi_o)$$

$$x = R\cos\varphi\cos\lambda; y = R\cos\varphi\sin\lambda$$

MCE:The momentum control element, a triangle with u and $v\cos\varphi$ at its centroid.

For example, the surface elevation is determined by the continuity equations given as

$$\iint_{\Omega_i^\zeta} \frac{\partial \zeta_i}{\partial t} R^2 cos\varphi d\lambda d\varphi = -\iint_{\Omega_i^\zeta} \frac{1}{Rcos\varphi}[\frac{\partial \bar{u} D}{\partial \lambda} + \frac{\partial(\bar{v}cos\varphi)D}{\partial \varphi}]R^2 cos\varphi d\lambda d\varphi$$

$$\Rightarrow \frac{\partial \zeta_i}{\partial t} = -\frac{R}{\Omega_i^\zeta}[\oint_\varphi \bar{u} D d\varphi' - \oint_\lambda (\bar{v}cos\varphi D)d\lambda'] \qquad \Delta x = Rcos\varphi d\lambda, \Delta y = Rd\varphi$$

Numerically integrated using the modified Runge-Kutta time-stepping scheme with a second-order approximation:

$$\zeta_j^0 = \zeta_j^n;\ R_\zeta^0 = R_\zeta^n$$

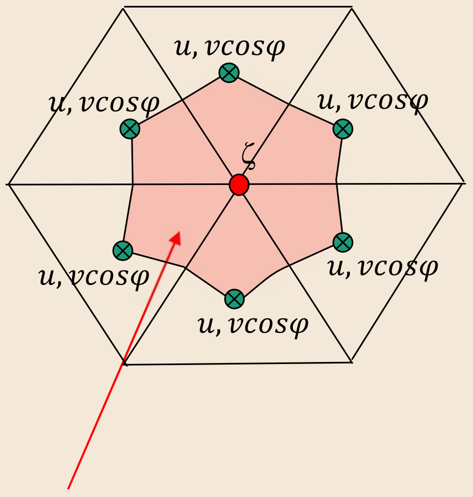
$$R_\zeta^n = \sum_{m-1}^{NT(j)}(\Delta x_{2m-1} v_m^n - \Delta y_{2m-1} u_m^n) D_{2m-1}^n + \sum_{m-1}^{NT(j)}(\Delta x_{2m} v_m^n - \Delta y_{2m} u_m^n) D_{2m}^n$$

$$\zeta_j^k = \zeta_j^0 - \alpha^k \frac{\Delta t R_\zeta^{k-1}}{2\Omega_j^\zeta};\ \zeta_j^{n+1} = \zeta_j^4$$

The same Runge-Kutta scheme is used for horizontal advection terms in the momentum equations

$$u_i^0 = u_i^n; v_i^0 = v_i^n; R_u^0 = R_u^n;\ R_v^0 = R_v^n$$

$$u_i^k = u_i^0 - \alpha^k \frac{\Delta t R_u^{k-1}}{2\Omega_i^u};\ v_i^k = v_i^0 - \alpha^k \frac{\Delta t R_v^{k-1}}{2\Omega_i^v}; \quad u_i^{n+1} = u_i^4;\ v_i^{n+1} = v_i^4$$



TCE: The tracer control element: an area with a node as a center and bounded triangles with u and $vcos\varphi$ at its centroid.

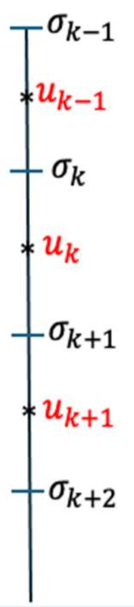
Step 2: Vertical diffusion terms:

$$\iint_{\Omega}\int_{\sigma_{k+1}}^{\sigma_k}\left[\frac{(uD)_k^{n+1}-(uD)_k^{*}}{\Delta t}\right]d\sigma d\Omega=\iint_{\Omega}\int_{\sigma_{k+1}}^{\sigma_k}\left[\frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial u}{\partial\sigma}\right)\right]d\sigma d\Omega$$

$$\iint_{\Omega}\int_{\sigma_{k+1}}^{\sigma_k}\left[\frac{(vD)_k^{n+1}-(vD)_k^{*}}{\Delta t}\right]d\sigma d\Omega=\iint_{\Omega}\int_{\sigma_{k+1}}^{\sigma_k}\left[\frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial v}{\partial\sigma}\right)\right]d\sigma d\Omega$$

$$\Rightarrow (uD)_k^{n+1}=(uD)_k^{*}+\frac{2\Delta t}{D^{n+1}(\sigma_k-\sigma_{k+1})}\left[K_{m,\sigma_k}^{n+1}\frac{u_{k-1}^{n+1}-u_k^{n+1}}{\sigma_{k-1}-\sigma_{k+1}}-K_{m,\sigma_{k+1}}^{n+1}\frac{u_k^{n+1}-u_{k+1}^{n+1}}{\sigma_k-\sigma_{k+2}}\right]$$

$$\Rightarrow (vD)_k^{n+1}=(vD)_k^{*}+\frac{2\Delta t}{D^{n+1}(\sigma_k-\sigma_{k+1})}\left[K_{m,\sigma_k}^{n+1}\frac{v_{k-1}^{n+1}-v_k^{n+1}}{\sigma_{k-1}-\sigma_{k+1}}-K_{m,\sigma_{k+1}}^{n+1}\frac{v_k^{n+1}-v_{k+1}^{n+1}}{\sigma_k-\sigma_{k+2}}\right]$$



$$K_m\frac{\partial u}{\partial z}\Big|_{\sigma_k}^{n+1}=\frac{u_{k-1}^{n+1}-u_k^{n+1}}{\frac{\sigma_{k-1}+\sigma_k}{2}-\frac{\sigma_k+\sigma_{k+1}}{2}}$$

$$K_m\frac{\partial u}{\partial z}\Big|_{\sigma_{k+1}}^{n+1}=\frac{u_k^{n+1}-u_{k+1}^{n+1}}{\frac{\sigma_k+\sigma_{k+1}}{2}-\frac{\sigma_{k+1}+\sigma_{k+2}}{2}}$$

Vertical discretization

$$-A_k u_{k+1}^{n+1}+B_k u_k^{n+1}-C_k u_{k-1}^{n+1}=u_k^{*}$$

$$-A_k v_{k+1}^{n+1}+B_k v_k^{n+1}-C_k v_{k-1}^{n+1}=v_k^{*}$$

$$\begin{cases}A_k=\dfrac{2\Delta t K_{m,\sigma_k+1}^{n+1}}{\left[D^{n+1}\right]^2(\sigma_k-\sigma_{k+1})(\sigma_k-\sigma_{k+2})}\\ C_k=\dfrac{2\Delta t K_{m,\sigma_k}^{n+1}}{\left[D^{n+1}\right]^2(\sigma_k-\sigma_{k+1})(\sigma_{k-1}-\sigma_{k+1})}\\ B_k=1+C_k+A_k\end{cases}$$

They are tridiagonal equations, and the solution can be determined without a matrix solver's request.

$$u^{n+1}(k)=VH(k)u^{n+1}(k+1)+VHP(k)$$

$$v^{n+1}(k)=VH(k)v^{n+1}(k+1)+VHP1(k)$$

$$VH(k)=\frac{A_k}{B_k-C_kVH(k-1)}$$

$$VHP(k)=\frac{u_k^{*}+C_kVHP(k-1)}{B_k-C_kVH(k-1)},\ VHP1(k)=\frac{v_k^{*}+C_kVHP(k-1)}{B_k-C_kVH(k-1)}$$

MEDM Lab Marine Ecosystem Dynamics Modeling

Current versions of FVCOM

Tracer equations (temperature, salinity, sediment concentration, etc.). For example, the temperature equation

$$\frac{\partial TJ}{\partial t}+\frac{1}{R\cos\varphi}\left[\frac{\partial(uJ)T}{\partial\lambda}+\frac{\partial(v\cos\varphi)JT}{\partial\varphi}\right]+\frac{\partial\omega T}{\partial r}=\frac{\partial}{\partial r}\left(\frac{K_h}{J}\frac{\partial T}{\partial r}\right)+JF_T \Rightarrow \frac{(TJ)_i^{n+1}-(TJ)_i^n}{\Delta t}=-(R_T)_i^n+\frac{\Delta}{\Delta r}\left(\frac{(K_h)_i^n}{J_i^n}\frac{\Delta T_i^{n+1}}{\Delta r}\right)$$

Step 1: $(TJ)_i^*=(TJ)_i^n-\frac{\Delta t}{\Omega_i^T}\iint_{\Omega_i^T}(R_T)_i^n R^2\cos\varphi d\lambda d\varphi$ **(Explicit);**

$$(R_T)_i^n=\frac{1}{R\cos\varphi}\left[\frac{\partial(uJ)T}{\partial\lambda}+\frac{\partial(v\cos\varphi)JT}{\partial\varphi}\right]_i^n+\frac{\partial\omega T}{\partial r}-(JF_T)_i^n$$

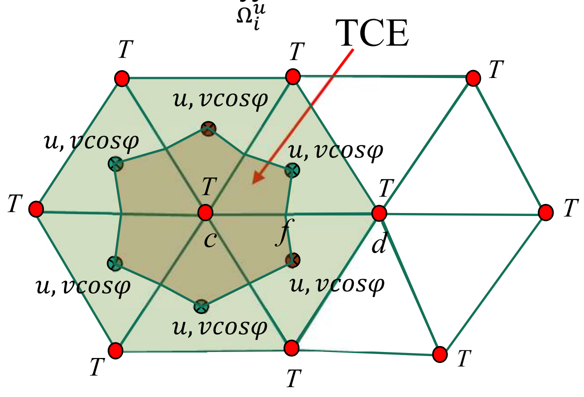
Step 2: $\frac{(TJ)_i^{n+1}-(TJ)_i^*}{\Delta t}=\frac{\Delta}{\Delta r}\left(\frac{(K_h)_i^n}{J_i^n}\frac{\Delta T_i^{n+1}}{\Delta r}\right)$ **(Implicit)**

$$\iint_{\Omega_i^u}(R_T)_i^n R^2\cos\varphi d\lambda d\varphi=\oint(v_N)_j^n J_j^n(T_f)_j^n ds+\iint_{\Omega_i^u}\frac{\partial\omega T}{\partial r}R^2\cos\varphi d\lambda d\varphi-\iint_{\Omega_i^u}J_j^n(F_T)_j^n R^2\cos\varphi d\lambda d\varphi$$

Determining T_f 1. **The second-order upwind scheme**:

$$(T_f)_j^n=T_i^n+\frac{\partial T^n}{\partial\lambda}\Delta\lambda+\frac{\partial T^n}{\partial\varphi}\Delta\varphi;\quad \frac{\partial T}{\partial\lambda}=\frac{R}{\Omega_{T_i^n}}\oint T^n d\varphi;\quad \frac{\partial T}{\partial\varphi}=\frac{R}{\Omega_{T_i^n}}\oint[T^n(\cos\varphi)_{T_i^n}]d\lambda;$$

j is the triangle centroid ID, and i is the index of the central node of $\Omega_{T_i^n}$. $\Omega_{T_i^n}$ is the area where T_f is calculated.
For the inflow, $\Omega_{T_i^n}$ is the area determined by the elements with the i node in the upstream region



2. Total variational diminishing (TVD) scheme (Dr. O. A. Nøst's team, Akvaplan-niva, Norway):

$$T_f=T_c+\frac{1}{2}\psi(T_d-T_c)\qquad \psi=\begin{cases}1 & \text{Central difference}\\ 0 & \text{First-order upwind}\\ \text{Liner function of } r=\frac{T_c-T_b}{T_d-T_c} & \text{Second-order upwind}\end{cases}$$

Flux limiters:

SUPERBEE: $\psi=max(0,min(1,2r),min(2,r))$

MINMOD: $\psi=max(0,min(1,r))$

MUSCL: $\psi=\frac{r+|r|}{1+|r|}$

Updating FVCOM with hybrid Eulerian-Operator-Integration-Factor Splitting (OIFS) method as an alternative solver option.

$$\frac{duD}{dt}+\frac{\partial u\omega}{\partial\sigma}+R_u=\frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial u}{\partial\sigma}\right)$$

$$\frac{dvD}{dt}+\frac{\partial v\omega}{\partial\sigma}+R_v=\frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial v}{\partial\sigma}\right)$$

$$\frac{d}{dt}=\frac{\partial(\;)}{\partial t}+\frac{\partial u(\;)}{\partial x}+\frac{\partial v(\;)}{\partial y}$$

Note: Use the σ-coordinate equation to explain how this method works. The FVCOM code is modified based on the generalized terrain-followed spherical and Cartesian coordinates

Step 1: Solve the horizontal terms using the 2nd-order backward OIFS method.

$$\frac{3(uD)^*-4(\tilde{u}\tilde{D})^n+(\tilde{u}\tilde{D})^{n-1}}{2\delta t}+R_u=0$$

$$\frac{3(vD)^*-4(\tilde{v}\tilde{D})^n+(\tilde{v}\tilde{D})^{n-1}}{2\delta t}+R_v=0$$

Step 2: Solve the vertical diffusion and advection terms implicitly by following the algorithm used in the current version of FVCOM.

$$\frac{(uD)^{n+1}-(uD)^*}{\Delta t}=\frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial u}{\partial\sigma}\right)-\frac{\partial u\omega}{\partial\sigma}$$

$$\frac{(vD)^{n+1}-(vD)^*}{\Delta t}=\frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial v}{\partial\sigma}\right)-\frac{\partial v\omega}{\partial\sigma}$$

Here,

- R_u and R_v do not include terms of horizontal advection on terms.
- δt is the Lagrangian time step.
- $\tilde{u}$ and $\tilde{v}$ are the x and y components of the Lagrangian velocity.

- Moving the vertical advection to the vertical diffusion equation to avoid the OIFS solvers in two distinct motion scales in the horizontal and vertical.
- Determining the Lagrangian velocity at the nth and (n-1)th time steps using the OIFS method can avoid deficiencies in Lagrangian tracking.

Similarities and differences between semi-Lagrangian and OIFS methods

- Both solve the conservative advection equations to determine the Lagrangian velocity: $\frac{du}{dt} = 0$ and $\frac{dv}{dt} = 0$.
- The semi-Lagrangian method uses forward/backward particle tracking on characteristic lines.
- The OIFS method solves the conservative advection equations in the Eulerian space (Maday et al. 1990).

An example to illustrate the similarities of these two methods

Let us consider a one-dimensional case, for example.

$$\frac{dT}{dt} = 0$$

For the semi-Lagrangian method, we have:

$$\frac{dT}{dt} = 0;\ \frac{dx}{dt} = u \Longrightarrow x_d = x_i - u\Delta t, \Longrightarrow T(x_d, t_n) = T(x_i - u\Delta t, t_n)$$

For OIFS method, we have:

$$T(x_d) = T(x_i) - \Delta t u \frac{\partial T}{\partial x} \Longrightarrow \frac{T(x_d) - T(x_i)}{\Delta t} = -u \frac{\partial T}{\partial x}$$

This equation can be obtained using the Taylor expansion from the above semi-Lagrangain equation , i.e.

$$T(x_d, t_n) = T(x_i - u\Delta t, t_n) = T(x_i) - \Delta t u \frac{\partial T}{\partial x} \Longrightarrow \frac{T(x_d) - T(x_i)}{\Delta t} = -u \frac{\partial T}{\partial x}$$

Advantages and Challenging of Implementing semi-Lagrangian and OIFS Methods into FVCOM

The semi-Lagrangian Method	The OIFS Method
• Becomes increasingly more cost-effective with increasing time-step. On a distributed-memory computer, the departure point of a specific grid point may lie off-processor. • The online 2nd-order accurate 4th-order Runge-Katta particle tracking program in FVCOM can be directly used for the forward/backward tracking to determine the departure point. However, particle tracking requires the treatment of various open and solid boundaries, which can increase numerical uncertainty around boundaries. • An interpolation stencil, which is a set of points used to perform the interpolation, must be coded in the grid. The interpolation accuracy depends on the interpolation scheme, with high-order interpolation functions being preferred for their accuracy.	• The success of the method relies on solving an advection problem efficiently. Multi-advection operators can be implemented. • OIFS can take advantage of unstructured grids in the current versions of FVCOM. • The MPI/OpenMP parallelization methods, which are seamlessly integrated into the current version of FVCOM, can be directly applied to the OIFS method, ensuring compatibility and ease of use. • The departure point values do not rely on interpolation, as in the semi-Lagrangian method, which could avoid the numerical errors caused by interpolation. • OIFS can be one option for FVCOM solvers without changing the grid setup.

The approaches used for Step 1

$$\frac{3(uD)^* - 4(\tilde{u}\tilde{D})^n + (\tilde{u}\tilde{D})^{n-1}}{2\delta t} + R_u = 0$$

$$\frac{3(vD)^* - 4(\tilde{v}\tilde{D})^n + (\tilde{v}\tilde{D})^{n-1}}{2\delta t} + R_v = 0,$$

$$\Rightarrow$$

$$(uD)^* = \frac{2}{3}(\tilde{u}\tilde{D})^n - \frac{1}{6}(\tilde{u}\tilde{D})^{n-1} - \frac{2}{3}\delta t R_u$$

$$(vD)^* = \frac{2}{3}(\tilde{v}\tilde{D})^n - \frac{1}{6}(\tilde{v}\tilde{D})^{n-1} - \frac{2}{3}\delta t R_v$$

Determining the backward Lagrangian velocities at the nth and (n-1)th time steps by solving the two-dimensional advection equation within the time interval of $t_n < t \le t_{n+1}$:

$$\frac{\partial Du}{\partial t} = -\left(\frac{\partial Duu}{\partial x} + \frac{\partial Dvu}{\partial y}\right)$$

$$\frac{\partial Dv}{\partial t} = -\left(\frac{\partial Duv}{\partial x} + \frac{\partial Dvv}{\partial y}\right)$$

$$\Rightarrow$$

$$\iint_{\Omega} \int_{\sigma_{k+1}}^{\sigma_k} \frac{\partial Du}{\partial t} d\sigma dx dy = -\iint_{\Omega} \int_{\sigma_{k+1}}^{\sigma_k} \left(\frac{\partial Duu}{\partial x} + \frac{\partial Dvu}{\partial y}\right) d\sigma dx dy \Longrightarrow \boxed{\frac{\partial Du}{\partial t} = -\frac{1}{\Omega}\oint (Du)\, v_n dl}$$

$$\iint_{\Omega} \int_{\sigma_{k+1}}^{\sigma_k} \frac{\partial Dv}{\partial t} d\sigma dx dy = -\iint_{\Omega} \int_{\sigma_{k+1}}^{\sigma_k} \left(\frac{\partial Duv}{\partial x} + \frac{\partial Dvv}{\partial y}\right) d\sigma dx dy \Longrightarrow \boxed{\frac{\partial Dv}{\partial t} = -\frac{1}{\Omega}\oint (Dv)\, v_n dl}$$

Note: One can solve these two equations using the same 2nd-order Runge-Katta explicit method from the current version of FVCOM with a sub-time step within the time interval of $t_n < t \le t_{n+1}$or implementing an implicit Runge-Katta method.

1. Explicit solver with a sub-time step within the time interval of $t_n < t \le t_{n+1}$

The current version of FVC OM has a 2nd KR integration method to solve the horizontal advection terms. The difference here is that we only solve the horizontal advection equations over the time interval of $t_n < t \le t_{n+1}$. One can easily use the same explicit 2nd KR method to solve the conservative advection equations.

$$\left(\widehat{D}\hat{u}\right)_1^{n+\frac{1}{2}} = (Du)^n - \frac{\Delta t}{2\Omega}\oint (Du)^n\, v_n^n dl \Rightarrow \left(\widehat{D}\hat{u}\right)_1^{n+1} = (Du)^n - \frac{\Delta t}{\Omega}\oint (Du)^{n+\frac{1}{2}}\, v_n^{n+\frac{1}{2}} dl$$

$$\left(\widehat{D}\hat{v}\right)_1^{n+\frac{1}{2}} = (Dv)^n - \frac{\Delta t}{2\Omega}\oint Dv_1^n\, v_n^n dl \Rightarrow \left(\widehat{D}\hat{v}\right)_1^{n+1} = (Dv)^n - \frac{\Delta t}{\Omega}\oint (Dv)^{n+\frac{1}{2}}\, v_n^{n+\frac{1}{2}} dl$$

$$\left(\widehat{D}\hat{u}\right)_2^{n-\frac{1}{2}} = (Du)^{n-1} - \frac{\Delta t}{2\Omega}\oint (Du)^{n-1}\, v_{n,}^{n-1} dl \Rightarrow \left(\widehat{D}\hat{u}\right)_1^{n} = (Du)^{n-1} - \frac{\Delta t}{\Omega}\oint (Du)^{n-\frac{1}{2}}\, v_n^{n-\frac{1}{2}} dl$$

$$\left(\widehat{D}\hat{u}\right)_2^{n+\frac{1}{2}} = (Du)_2^n - \frac{\Delta t}{2\Omega}\oint (Du)_2^n\, v_n^n dl \Rightarrow \left(\widehat{D}\hat{u}\right)_2^{n+1} = (Du)_2^n - \frac{\Delta t}{\Omega}\oint (Du)_2^{n+\frac{1}{2}}\, v_{n2}^{n+\frac{1}{2}} dl$$

$$\left(\widehat{D}v\right)_2^{n+\frac{1}{2}} = (Dv)_2^n - \frac{\Delta t}{2\Omega}\oint (Dv)_2^n\, v_n^n dl \Rightarrow \left(\widehat{D}\hat{v}\right)_2^{n+1} = (Dv)_2^n - \frac{\Delta t}{\Omega}\oint (Dv)_2^{n+\frac{1}{2}}\, v_{n2}^{n+\frac{1}{2}} dl$$

$$\left(\widehat{D}v\right)_2^{n-\frac{1}{2}} = (Dv)^{n-1} - \frac{\Delta t}{2\Omega}\oint (Dv)^{n-1}\, v_{n,}^{n-1} dl \Rightarrow \left(\widehat{D}\hat{v}\right)_1^{n} = (Dv)^{n-1} - \frac{\Delta t}{\Omega}\oint (Dv)^{n-1/2}\, v_n^{n-\frac{1}{2}} dl$$

According to the momentum conservation, the Lagrangian velocity at nth and (n-1)th time steps equals

$$(\tilde{u}\widetilde{D})^n = \left(\widehat{D}\hat{u}\right)_1^{n+1}$$
$$(\tilde{v}\widetilde{D})^n = \left(\widehat{D}\hat{v}\right)_1^{n+1}$$

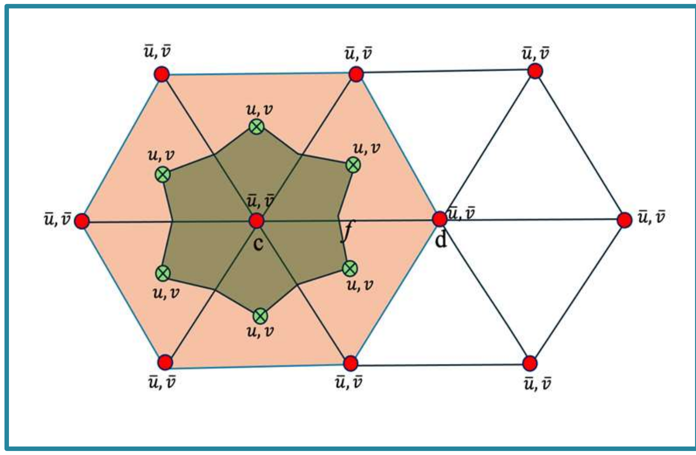
And

$$(\tilde{u}\widetilde{D})^{n-1} = \left(\widehat{D}\hat{u}\right)_2^{n+1}$$
$$(\tilde{v}\widetilde{D})^{n-1} = \left(\widehat{D}\hat{v}\right)_2^{n+1}$$

This approach can increase the maximum CFL number.

2. Implicit solver with a sub-time step within the time interval of $t_n < t \le t_{n+1}$

Solving the advection equation using the tracer control element using the combined implicit TVD (spatial) and RK (time integration) algorithms (Note: This helps reduce the matrix size for the implicit solver).



For momentum equation,

Step 1: interpolate the velocity from individual triangular centroids to the triangle's nodes.

$$\bar{u}(N_i(\hat{j}_m)) = \frac{1}{NT(\hat{j})}\sum_{NB_i=1}^{NT(\hat{j})} u(NB_i)\; NB_i = 1,2,\dots.NT(\hat{j})$$

$$\bar{v}(N_i(\hat{j}_m)) = \frac{1}{NT(\hat{j})}\sum_{NB_i=1}^{NT(\hat{j})} v(NB_i)\; NB_i = 1,2,\dots.NT(\hat{j})$$

$NT(\hat{j})$: the total number of the surrounding triangle with a connection to the jth node.

Step 2: Construct the matrix based on the finite-volume control element centered at the tringle's nodes.

$$\frac{\partial Du}{\partial t} = -\frac{1}{\Omega}\oint(D\bar{u})\,v_n dl = -\frac{1}{\Omega}\sum_{m=1}^{2NT} D^n \bar{u}_f^{n+1} v_n\,\Delta l$$

$$\frac{\partial Dv}{\partial t} = -\frac{1}{\Omega}\oint(D\bar{v})\,v_n dl = -\frac{1}{\Omega}\sum_{m=1}^{2NT} D^n \bar{v}_f^{n+1} v_n\,\Delta l$$

$$\bar{u}_f^{n+1} = \bar{u}_c^{n+1} + \frac{1}{2}\psi_1(\bar{u}_d^{n+1} - \bar{u}_c^{n+1})$$

$$\bar{v}_f^{n+1} = \bar{v}_c^{n+1} + \frac{1}{2}\psi_2(\bar{v}_d^{n+1} - \bar{v}_c^{n+1})$$

Flux limiters

$$\psi_1 = \begin{cases} \max[0,\min(1,2r),\min(2,r)] & \text{Superbee} \\ \max[0,\min(1,r)] & \text{Minmod} \\ \dfrac{r+|r|}{1+|r|} & \text{Musel} \end{cases}$$

$$\psi_1 \to \psi_2 \text{ as } r \to r_1, r = \frac{\bar{u}_c^n - \bar{u}_b^n}{\bar{u}_d^n - \bar{u}_c^n}; r_1 = \frac{\bar{v}_c^n - \bar{v}_b^n}{\bar{v}_d^n - \bar{v}_c^n}$$

v_n is the normal velocity calculated by interpolation or extrapolation between the time interval n-1 and n.

Introducing the second-order accuracy of the total variational diminishing (TVD) method to construct the matrix for u^{n+1} and v^{n+1}.

Vertical diffusion and advection equations:

$$\iint_{\Omega}\int_{\sigma_{k+1}}^{\sigma_k}\left[\frac{(uD)_k^{n+1}-(uD)_k^*}{\Delta t}\right]d\sigma d\Omega=\iint_{\Omega}\int_{\sigma_{k+1}}^{\sigma_k}\left[\frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial u}{\partial\sigma}\right)-\frac{\partial u\omega}{\partial\sigma}\right]d\sigma d\Omega\ ,\quad \iint_{\Omega}\int_{\sigma_{k+1}}^{\sigma_k}\left[\frac{(vD)_k^{n+1}-(vD)_k^*}{\Delta t}\right]d\sigma d\Omega=\iint_{\Omega}\int_{\sigma_{k+1}}^{\sigma_k}\left[\frac{1}{D}\frac{\partial}{\partial\sigma}\left(K_m\frac{\partial v}{\partial\sigma}\right)-\frac{\partial v\omega}{\partial\sigma}\right]d\sigma d\Omega$$

$$\Longrightarrow (uD)_k^{n+1}=(uD)_k^*+\frac{2\Delta t}{D^{n+1}(\sigma_k-\sigma_{k+1})}\left[K_{m,\sigma_k}^{n+1}\frac{u_{k-1}^{n+1}-u_k^{n+1}}{\sigma_{k-1}-\sigma_{k+1}}-K_{m,\sigma_{k+1}}^{n+1}\frac{u_k^{n+1}-u_{k+1}^{n+1}}{\sigma_k-\sigma_{k+2}}\right]-\Delta t\frac{(u_{k-1}^{n+1}+u_k^{n+1})\overline{\omega}_k^{n+1}-(u_k^{n+1}+u_{k+1}^{n+1})\overline{\omega}_{k+1}^{n+1}]}{2(\sigma_k-\sigma_{k+1})}$$

$$\Longrightarrow (vD)_k^{n+1}=(vD)_k^*+\frac{2\Delta t}{D^{n+1}(\sigma_k-\sigma_{k+1})}\left[K_{m,\sigma_k}^{n+1}\frac{v_{k-1}^{n+1}-v_k^{n+1}}{\sigma_{k-1}-\sigma_{k+1}}-K_{m,\sigma_{k+1}}^{n+1}\frac{v_k^{n+1}-v_{k+1}^{n+1}}{\sigma_k-\sigma_{k+2}}\right]-\Delta t\frac{(v_{k-1}^{n+1}+v_k^{n+1})\overline{\omega}_k^{n+1}-(v_k^{n+1}+v_{k+1}^{n+1})\overline{\omega}_{k+1}^{n+1}]}{2(\sigma_k-\sigma_{k+1})}$$

$$-A_k u_{k+1}^{n+1}+B_k u_k^{n+1}-C_k u_{k-1}^{n+1}=u_k^*$$

$$-A_k v_{k+1}^{n+1}+B_k v_k^{n+1}-C_k v_{k-1}^{n+1}=v_k^*$$

$$\begin{cases} A_k=\dfrac{2\Delta t K_{m,\sigma_k+1}^{n+1}}{\left[D^{n+1}\right]^2(\sigma_k-\sigma_{k+1})(\sigma_k-\sigma_{k+2})}+\dfrac{\Delta t\overline{\omega}_{k+1}^{n+1}}{2D^{n+1}(\sigma_k-\sigma_{k+1})} \\ C_k=\dfrac{2\Delta t K_{m,\sigma_k}^{n+1}}{\left[D^{n+1}\right]^2(\sigma_k-\sigma_{k+1})(\sigma_{k-1}-\sigma_{k+1})}-\dfrac{\Delta t\overline{\omega}_k^{n+1}}{2D^{n+1}(\sigma_k-\sigma_{k+1})} \\ B_k=1+C_k+A_k+\dfrac{\Delta t}{D^{n+1}(\sigma_k-\sigma_{k+1})}(\overline{\omega}_k^{n+1}-\overline{\omega}_{k+1}^{n+1}) \end{cases}$$

They are tridiagonal equations, and the solution can be determined without a matrix solver's request.

$$u^{n+1}(k)=VH(k)u^{n+1}(k+1)+VHP(k)$$

$$v^{n+1}(k)=VH(k)v^{n+1}(k+1)+VHP1(k)$$

$$VH(k)=\frac{A_k}{B_k-C_kVH(k-1)}$$

$$VHP(k)=\frac{u_k^*+C_kVHP(k-1)}{B_k-C_kVH(k-1)},\qquad VHP1(k)=\frac{v_k^*+C_kVHP(k-1)}{B_k-C_kVH(k-1)}$$

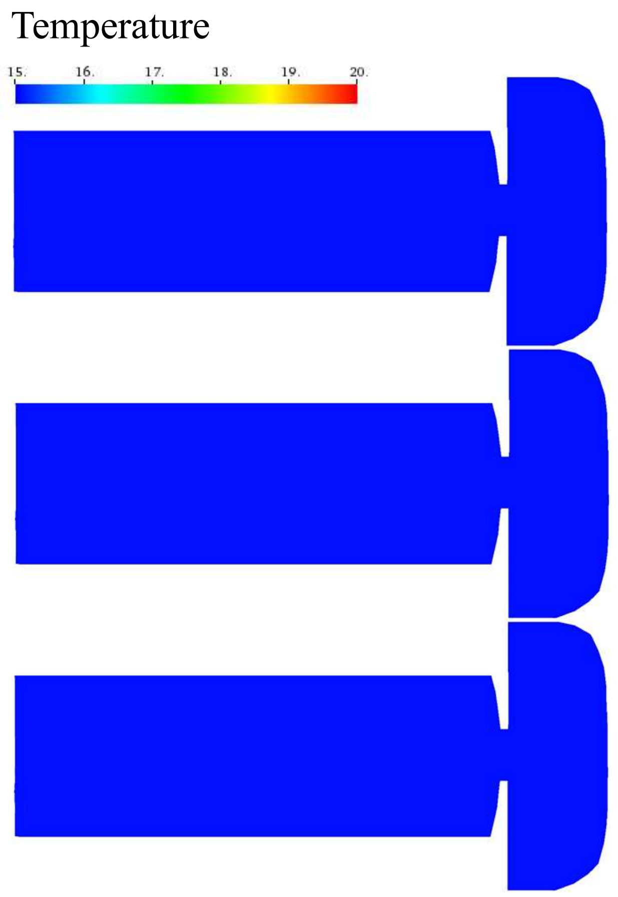
Team members and their contributions

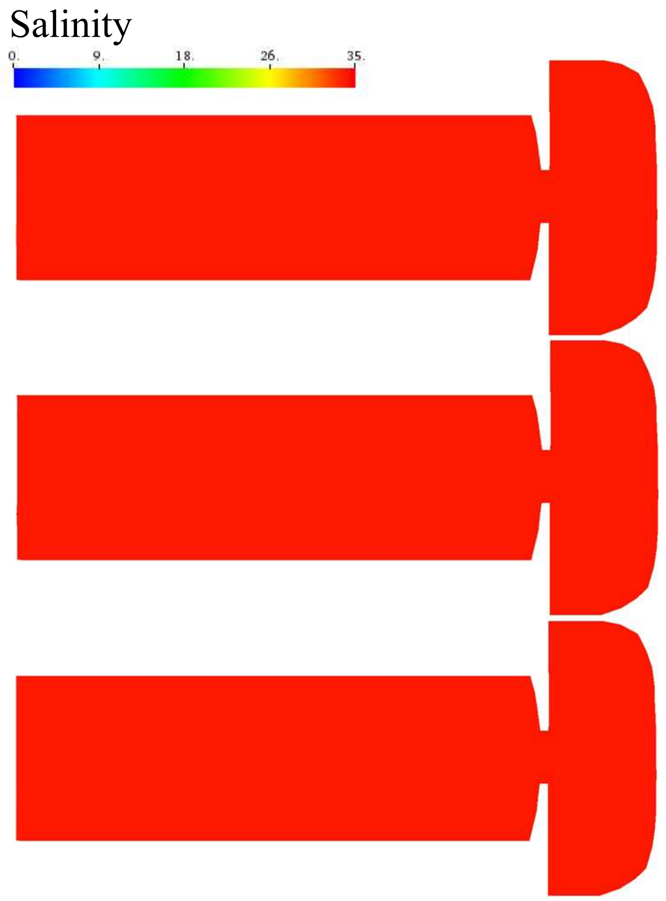
Team members	Contributions
Dr. S. Li	Leading the development of implementing the OIFS methods: the code modification and benchmark tests.
Dr. C. Chen	Supervising the developments in numerical schemes and code structures.
Dr. J. Qi	Modifying the implicit solver of the diffusion-advection equation: the code modification and benchmark tests.
Dr. G. Cowles	Supervising the new code parallelization.

Updates of the code modifications

Time	Completions or ongoing tasks
March-April 2024	Completed numerical algorithm designs in Mathematics.
May-June 2024	Completed the code modification to include the vertical advection in the implicit solver and conducted an initial benchmark test by comparing it with the current version of FVCOM.
May-June 2024	Completed the code development of the implicit OIFS method.
July-August 2024	• Test the implicit OIFS method on a benchmark test problem and evaluate the code by comparing the results with the current version of FVCOM. • Continue to debug the implicit diffusion-advection solvers for benchmark test problems.

Example: Comparisons between the implicit vertical diffusion-advection solver and upwind and TVD/MPDATA solvers.



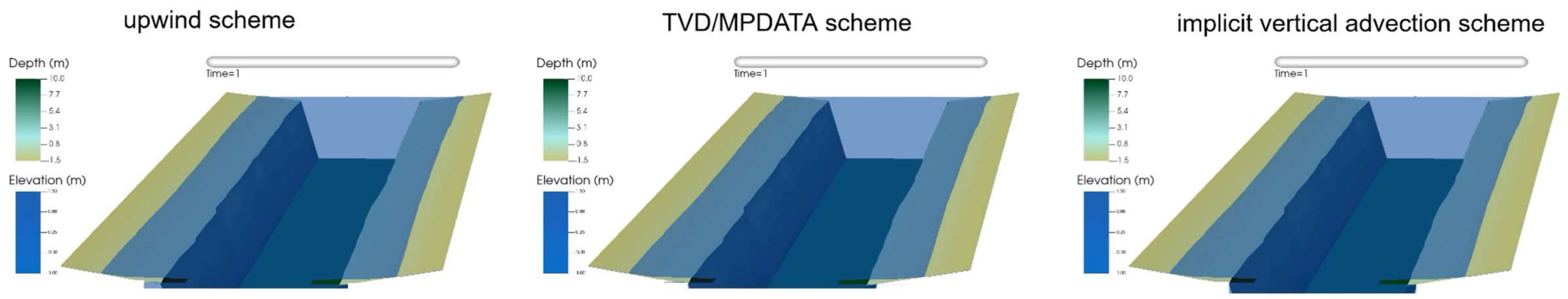


Upwind scheme

TVD/MPDATA scheme

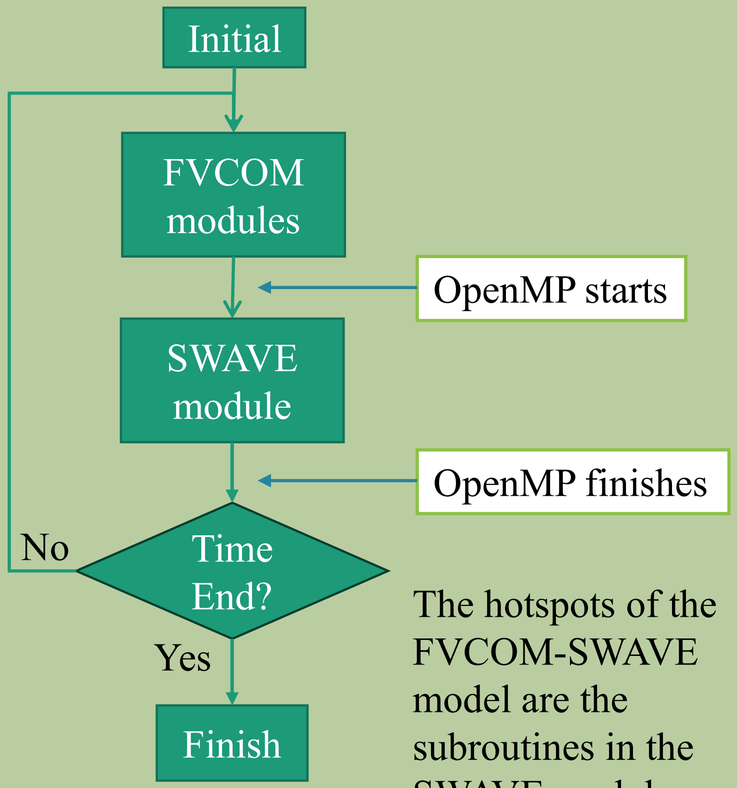
Implicit vertical advection scheme

Example: Comparisons between the implicit vertical diffusion-advection solver and upwind and TVD/MPDAT solvers.



Update of MPI/Open MP code debugging

OpenMP for FVCOM-SWAN model



The hotspots of the FVCOM-SWAVE model are the subroutines in the SWAVE module.

Updated from Dr. J. K. Chen

Implementation was initially done for one OpenMP parallel block in the SWAVE subroutines to reduce the start-up time of OpenMP threads

Time-consuming statistics in each step of the FVCOM-SWAN model with OpenMP

Subroutine	**Wall time**				
	Threads: 1 Task(s): 1	Threads: 2 Task(s): 1	Threads: 4 Task(s): 1	Threads: 1 Task(s): 2	Threads: 1 Task(s): 4
SWOMPU1	0.1179	0.0778	0.0437	0.0598	0.0314
ADV_N	0.1008	0.0555	0.0322	0.0694	0.0482
SWOMPU2	0.2073	0.1636	0.1636	0.1055	0.0552
Total					
SWCOMP	0.4524	0.3250	0.2669	0.2770	0.1695
internal_step	0.5291	0.3901	0.3259	0.3171	0.1928

Summary

- The new OIFS solvers, which are being implemented in FVCOM, will expand FVCOM's capability of solving the problem with a meter-order resolution with a large CFL number or no CFL constraints.

- The next-generation FVCOM is designed to cater to a wide range of user needs. It will provide users with various multi-solvers, including RK-upwind, TVD/MPDATA, and OIFS. The inter-solver comparisons could help the solver's selection with the required accuracy.

- The combined MPI/OpenMP parallelization could significantly improve the FVCOM's computational efficiency.