Архитектура гидродинамического ядра Региональной Модели Океана ROMS: Искусство компромисса в пространстве противоречий

Александр Щепёткин

Университет Калифорнии в Лос-Анджелесе (UCLA)

Институт Океанологии им. П. П. Ширшова

Москва, 22 декабря 2017 г.

Моё вовлечение началось 24 августа 1997 с распараллеливания кода SCRUM-3.0 (Hernan Arango, Rutgers University) для суперкомпьютеров SGI Origin 2000. С самого начала было понятно что распараллеливанием дело не ограничится. Там все было не так...

- мы уже тогда знали (*e.g.* Shchepetkin, 1995; Shchepetkin & McWilliams, 1998) что моделировать турбулентный режим разностями второго порядка **бесполезно**;
- предыдущий опыт псевдоспектрального моделирования идеализированной турбулентности резко контрастировал в качестве решений;

Так случилось что я попал именно в то поколение которое ещё застало Cray-YMP и прошло весь путь эволюции суперкомпьютерного счета (в т.ч. векторного и параллельного) наблюдая полемику и концептуальные блуждания 199x.

- мы знали что *формальное* распараллеливание кода насыщающего шину памяти **бесполезно** (Paul Woodward, priv comm; + собственный опыт) для машин современных на то время. *Тем более бесполезно сегодня*;
- "стандартный" подход (*e.g. MPI*) в то время не существовал (и слава Богу: *неведение иногда полезнее знания догмы*);
- *одни слова для кухонь, другие для улиц*: навязывание/пропаганда "лёгкого" (HPF, loop-based directives) распараллеливания с одновременным кулуарным признанием что это не работает и не будет работать никогда;
- В 1995-6 годах US NAVY подарил две Connection Machines университетам: одну в Rutgers University, другую в MIT. **...с катастрофическими последствиями в обоих случаях**;
- мы уже тогда знали что написание параллельного кода проще чем написание кода "понятного" для автоматического компилятора;

Что делать со случайным, неудобным, нежелательным знанием?

- игнорировать, притворяясь что как будто ничего такого нет?
- ...или все таки?

MOM, POP, POM, HYCOM, NEMO, MITgcm - у всех точно такая же дилемма...

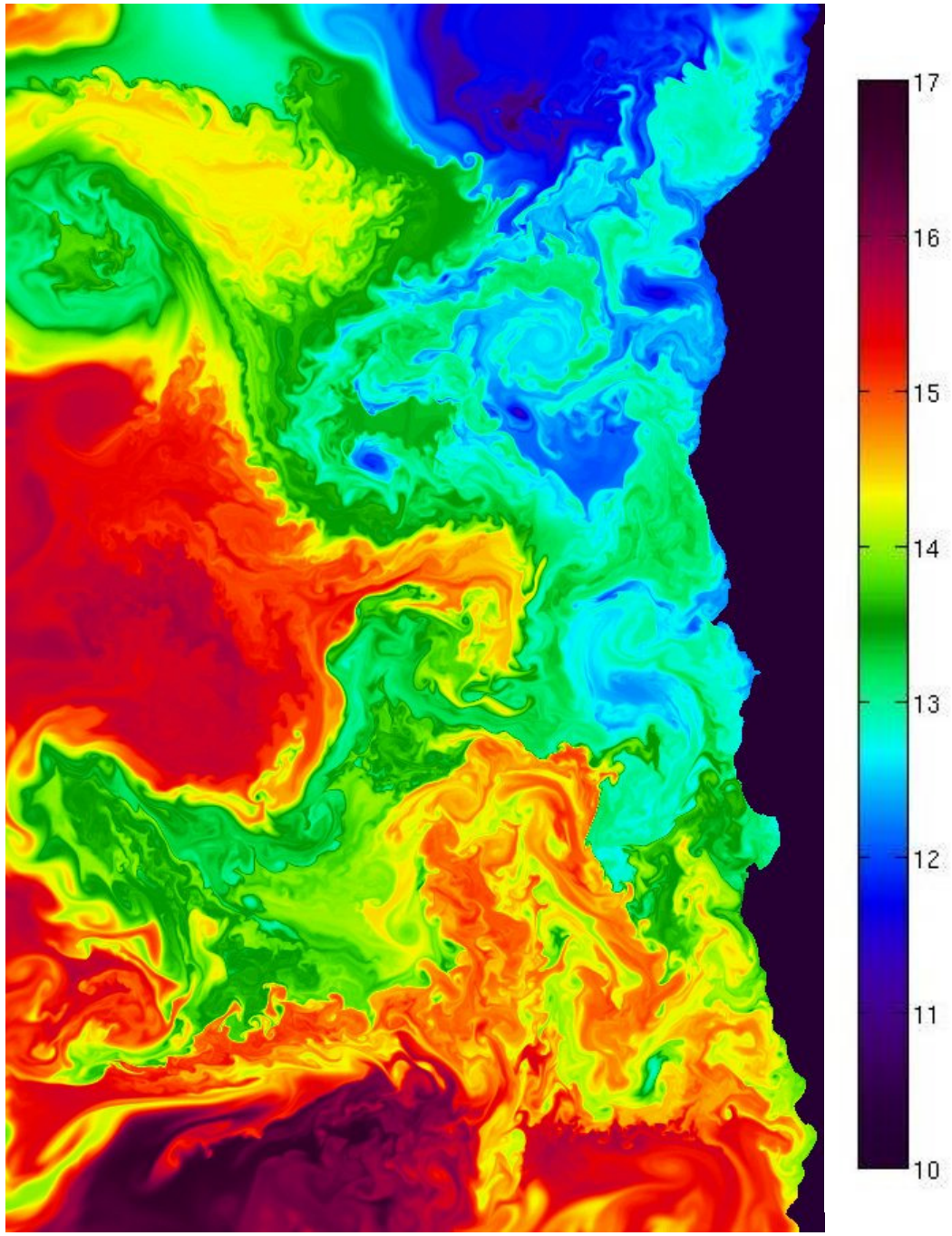
Исходные цели: физика

Моделирование Калифорнийского течения (California Current System), а на самом деле всего Западного Побережья США

Там всё шло не так...

- номинально все компоненты модели присутствовали но работали плохо
- численная устойчивость: необъяснимые ограничения на Δt
- открытые границы не работали
- очевидные ошибки сигма-координат

справа: температура поверхности моря (SST) из модели Средней Калифорнии, $\Delta x = 0.5$km, всего $1200 \times 1800 \times 60$ точек сетки, выполненной в 2009 году (courtesy Jeroen Molemaker)



Цели поставленные уже после начала проекта: т.е. перефокусировка...

Создать модель способную получать **сильно турбулентные решения**: хорошая пространственная точность для уравнений скоростей (наряду с трассерами T,S); малая диссипация; численная устойчивость без вязкости (гипервязкости - последняя используется только для поглощения турбулентного каскада);

Инженерный подход **равнопрочности** ко всему: точность vs. устойчивость vs. вычислительные затраты.

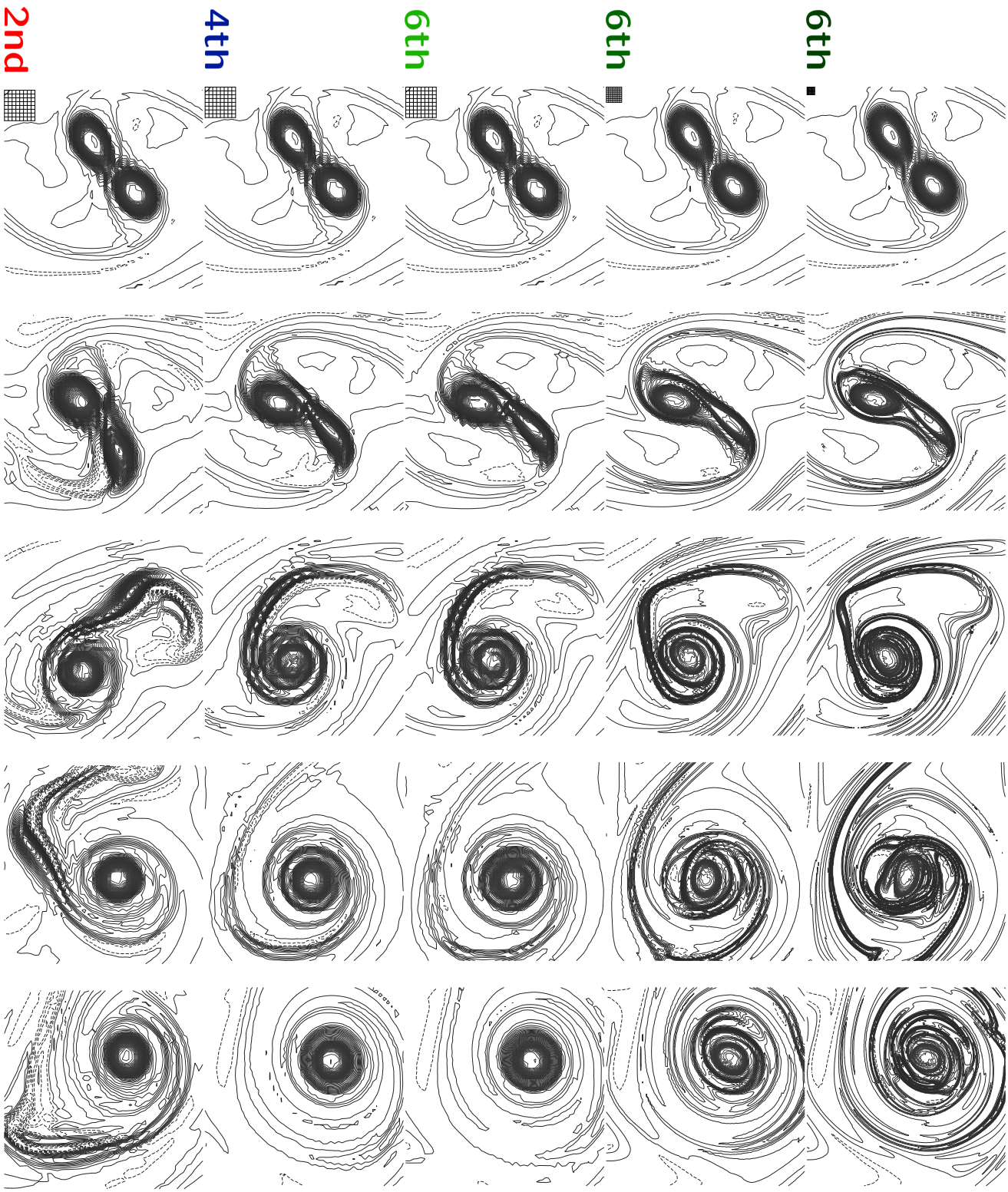
Концептуальный уход от порядка точности к **эффективному разрешению**: т.е. избирательное повышение порядка точности по пространству только там где есть усреднение (фактически внимание к интерполяциям).

Способность получить хорошую точность решения даже если шаг по времени выбран близким к пределу устойчивости (т.е. отсутствие "серой зоны" - принцип *patrick-proofness*).

Осознание иерархии быстроты физических процессов (в порядке уменьшения): баротропные волны $\rightarrow$ внутренние (1я бароклинная мода) $\rightarrow$ горизонтальная адвекция $\rightarrow$ горизонтальная вязкость/диффузия (в т.ч. гипервязкость) $\rightarrow$ Кориолис (инерционные колебания). Вертикальные адвекция и вязкость медленные но могут быть "быстрыми" из-за малости вертикального шага сетки. Фактически **разные алгоритмы шага по времени** для разных процессов.

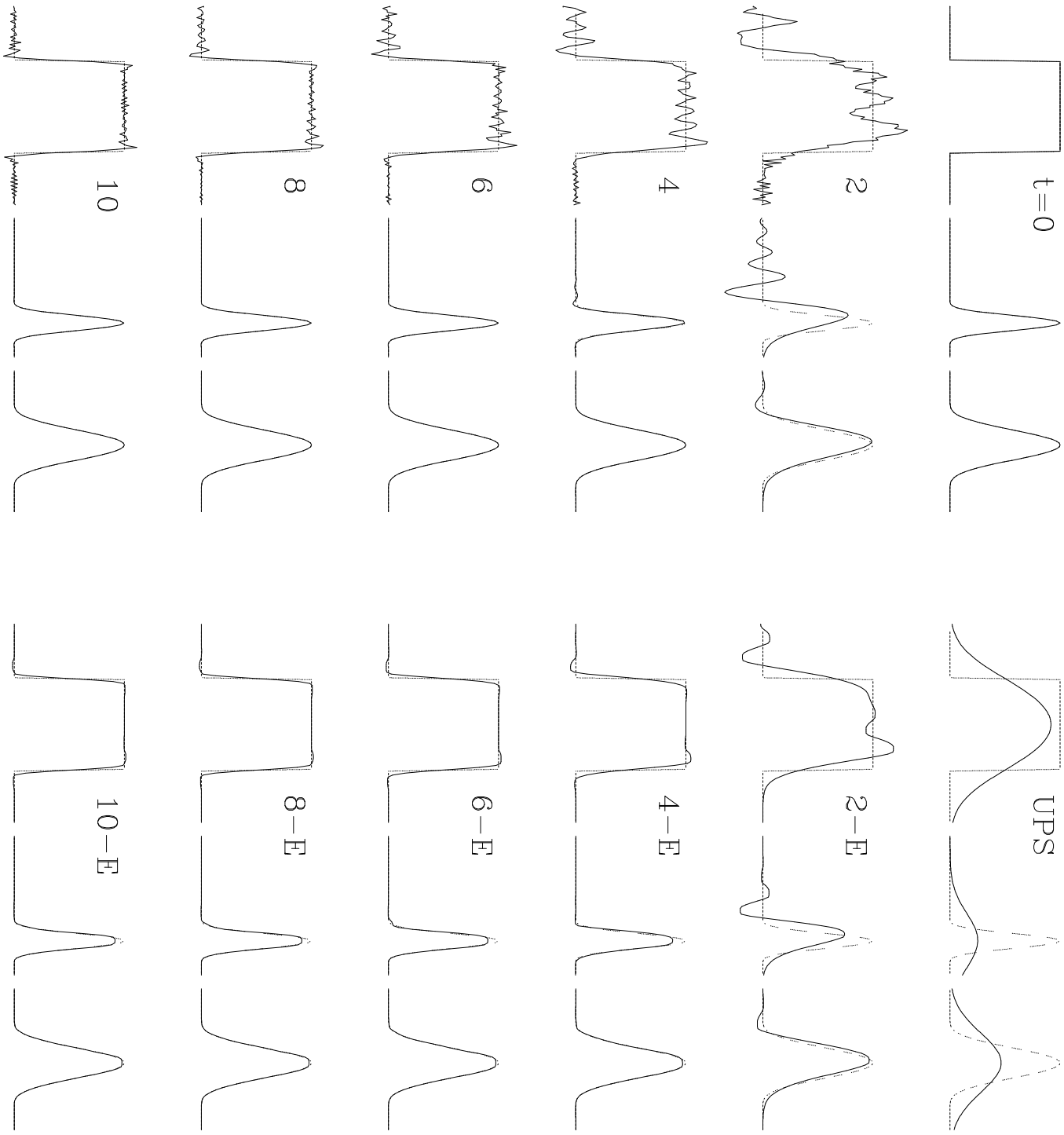
Избирательный подход к порядку точности по времени: третий для адвекции и Кориолиса; второй для волн; первый для вязкости/диффузии (прямой Эйлер для горизонтальных или обратный для вертикальных).

Влияние порядка точности на физику решения



Shchepetkin & McWilliams, 1998

"Сходимость" ...дисперсии по порядку точности



Shchepetkin & McWilliams, 1998

Цели: вычислительные (i.e. Computer Science)

Создать код который бы соответствовал нашим представлениям о компьютерных архитектурах того времени (SGI Origin, cache-based processors, pipelined execution etc.) и мог эволюционировать вместе с ними т.е. как бы "поймать время".

Не только распараллеливание...

Уход от чисто арифметической оценки затрат (по число операций - Flops) характерного для эпохи Cray-YMP по-инерции доминировавшего в то время.

Сильное концептуальное влияние Paul Woodward, LCSE, University of Minnesota; так же Aarpon Sawdey and Mathew O'Keefe - распарраллеливание модели океана MICOM (PhD диссертация Sawdey 1995) так и оставшегося невостребованным (современный HYCOM это во многих проявлениях фактически шаг назад).

Параллельная оболочка и структура внутреннего кода **полностью развязаны и независимы** друг от друга.

Написание внутреннего кода оптимизировано с точки зрения *однопроцессорной скорости* без оглядок и компромиссов: т.е. допустимы любые зависимости данных, рекурсивное использование массивов, оптимизация для кеш, и т.д.

Концептуальный уход от двухступенчого подхода к написанию кода (physics $\rightarrow$ computer science, one-way, a.k.a. *programming slavery*).

Концептуальный уход модульности a'la MOM

Фактически создать новый framework - т.е. набор семантических правил чтобы в последствии создавать "правильный" код особо не задумываясь.

Содержание

Современный ROMS: Обзор физических формулировок и численных алгоритмов

Избранные темы

- Варианты алгоритмов шага по времени
- Обобщённая вертикальная координата
- Адаптивная неявная адвекция по вертикали
- Изопикническая диффузия
- KPP: альтернативная формулировка параметризации вертикального перемешивания
- Вычислительная стратегия для бедного человека

Алгоритмы ROMS: Математическая характеристика

- **вертикальная координата:** формально относится к σ-типу моделей, но код использует 3D массив $z = z(x, y, s)$ не спрашивая откуда он берётся $\Rightarrow$ т.е. фактически **достаточно произвольная** вертикальная координата. В данное время наиболее часто применяется "z-sigma"

- ортогональные криволинейные координаты по горизонтали (все метрические коэффициенты преобразование векторов сохранены)

- приближение Буссинеска со специально модифицированным уравнением состояния морской воды EOS [Sun et. al., 1999; Dukowicz, 2001; Shchepetkin and McWilliams, 2011; Dewar et. al., 2015] - устраняет $\sim 90\%$ неточностей приближение Буссинеска с полным нелинейным сжимаемым EOS; полностью воспроизводит термобарический и каббельный эффекты

- **алгоритм временного обновления:** свободная поверхность конечной амплитуды **(не обязательно малой по отношению к глубине)**; расщепление бароклинной и баротропной мод с явным шагом по времени для обеих

- осреднение (фильтрация) баротропной моды в пределах одного бароклинного шага со специальными **S-образными весами** $\Rightarrow$ сохранение второго порядка точности во времени для осредненных величин

- **точное одновременное** (совместное) сохранение интегральной величины (суммирование по конечным объемам) и свойство сохранение однородного поля для всех трасеров $\Leftarrow$ достигнуто за счет **вторичного осреднения** полных баротропных потоков

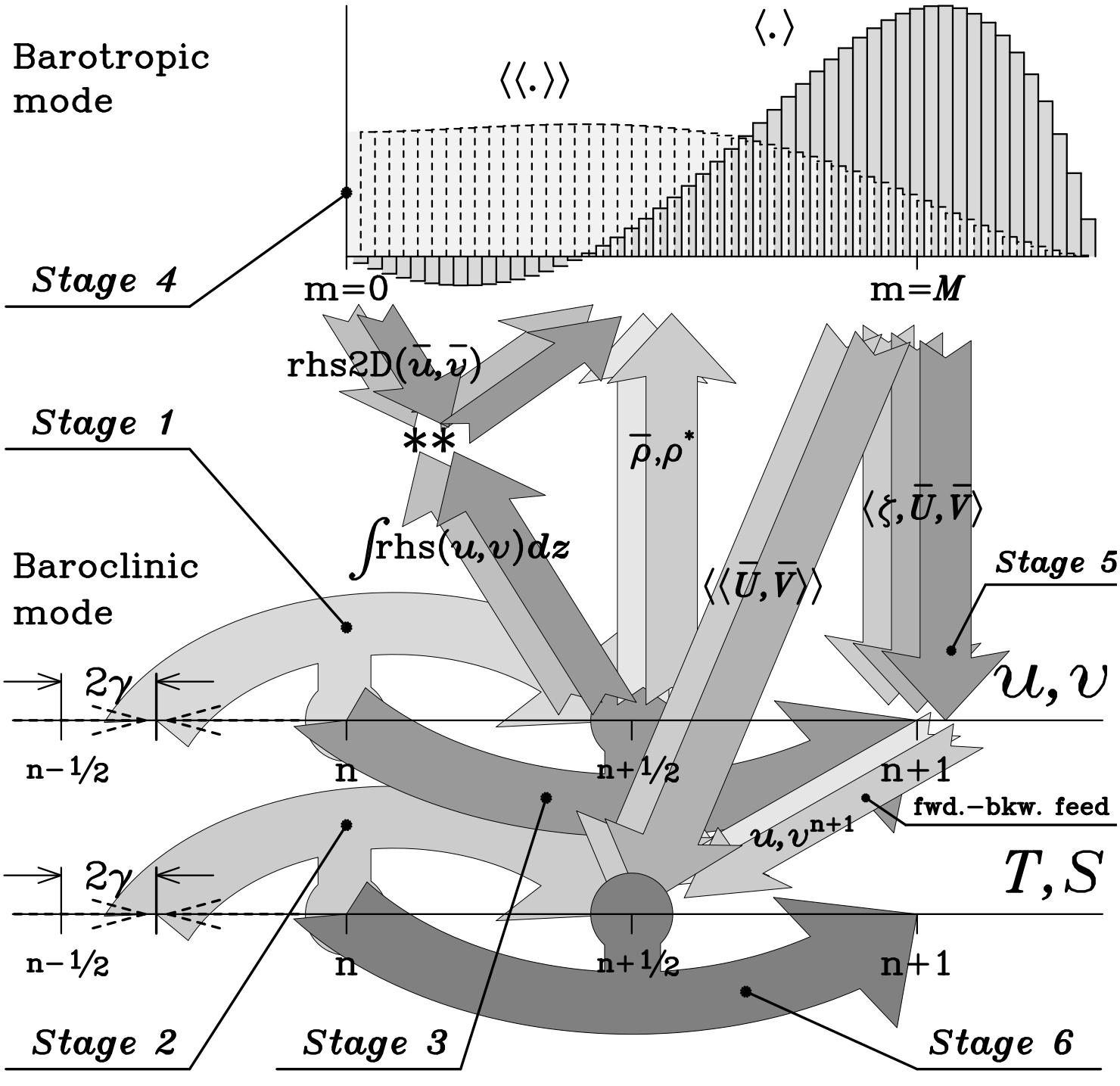
- **расщепление 2D и 3D мод:** баротропный градиент давления выведен как вариационная производная вертикального интеграла от полного градиента давления по отношению к вариации свободной поверхности (т.е. не просто $-gD\nabla_x\zeta$ как в уравнениях мелкой воды). Выражен через $\overline{\rho}$ и ρ_* (двумерные поля). Формальный уход от малости $\epsilon = (\rho - \rho_0)/\rho_0 \ll 1$ для точности расщепления (второй порядок). Фазовая скорость баротропной моды **учитывает её замедление стратификацией**. Так формальный учёт перекрёстного влияния стратификации и топографии на расщепление.

- новые (ранее неизвестные) алгоритмы шага по времени специально для пар гиперболических уравнений ($\partial_t u = -c\cdot\partial_x\zeta, \quad \partial_t\zeta = -c\cdot\partial_x u$) обобщение прямого-обратного алгоритма (forward-backward scheme, Sielecki, 1968; Mesinger & Arakawa, 1976) на более высокий порядок точности, а так же решение проблемы совместимости классического FB (Walters, Lane & Emmanuel, 2009).

- использование анализа устойчивости фон-Неймана как способ поиска и оптимизации: написание алгоритма с произвольными коэффициентами, далее требование сходимости с определёнными порядками точности даёт набор ограничений на коэффициенты; фактически программирование желаемых свойств

- умышленное диссипативное поведение вблизи предела устойчивости

- всегда используется вариант **обобщённого прямого-обратного шага** для 2D (свободная поверхность $\zeta \rightarrow$ скорости баротропные $\overline{u}, \overline{v}$) и 3D (скорости $u, v \rightarrow$ трассеры T, S) мод;


Barotropic mode
⟨⟨.⟩⟩
⟨.⟩
Stage 4
m=0
m=M
rhs2D($\bar{u},\bar{v}$)
Stage 1
**
$\bar{\rho},\rho^*$
$\int$rhs(u,v)dz
$\langle\zeta,\bar{U},\bar{V}\rangle$
Baroclinic mode
$\langle\langle\bar{U},\bar{V}\rangle\rangle$
Stage 5
u,v
2γ
n−1/2
n
n+1/2
n+1
fwd.−bkw. feed
u,v^{n+1}
T,S
2γ
n−1/2
n
n+1/2
n+1
Stage 2
Stage 3
Stage 6

- пространственный **порядок точности выше второго** для критически важных членов: адвекция для **скоростей** u, v и трассеров T, S, бароклинный градиент давления
- возможность использования сплайнов для вертикальной адвекции (eqv. compact differencing): точность + уход от потери точности из-за сильно неоднородного вертикального разрешения
- **адаптивно-неявная** вертикальная адвекция для 3D скоростей и трассеров – сохраняет устойчивость при превышении вертикального CFL, но при этом полностью эквивалентна исходной явной схеме сохраняет её точность (3й порядок) там где CFL не превышено: плавное включение неявности там где необходимо
- возможность **поворота поверхностей** горизонтальной диффузии (или гипердиффузии) для трассеров относительно поверхностей сигма-координат к изопикнам (нейтральным поверхностям, etc...). Алгоритм переключающихся триад с неявностью по вертикали чтобы уйти от жёстких ограничений на устойчивость при сильном наклоне изопикн
- **возможность точного рестарта** (сохранение полей для двух последовательных шагов)

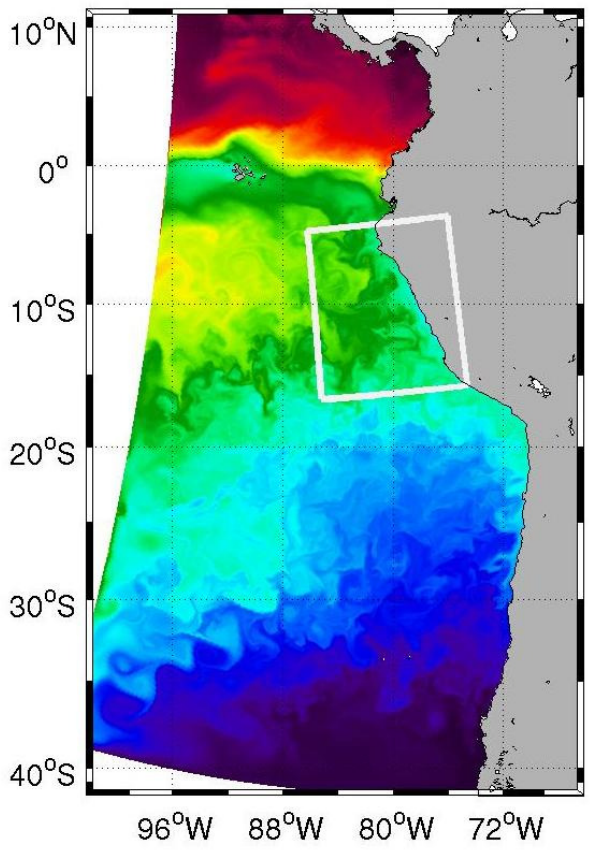
...Алгоритмы: физика

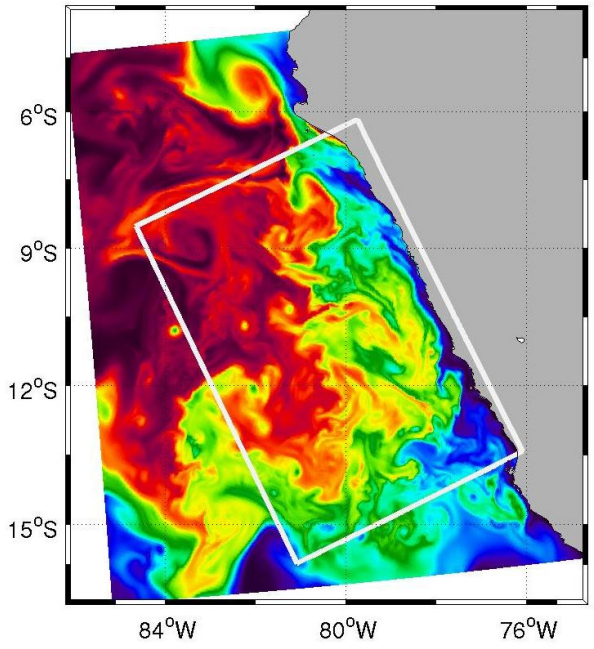
Primarily intended for modeling highly-turbulent flow regimes, sub-mesoscale fronts (sub $1km$-resolution), flow-topography interaction, etc... run in a limited-area

- ability to deal with open boundary conditions and grid nesting
- focus on higher-order advection
- **vortex stretching due to flow over non-uniform topography, topographic Rossby effect**
- bottom-boundary layer phenomena: ability to focus resolution near bottom
- more than one vertical mixing closure scheme
- **"If-less" KPP for top and bottom boundary layers**
- sub-models: biological, WEC, sediments, Sea-Ice, etc...

At the same time

- long-term **water-mass conservation properties** were historically de-emphasised – such errors naturally tend to ventilate through open boundaries in a typical US West Coast configuration. **This is being reconsidered.**
- ROMS never intended to become a climate model – **a sigma-model is unlikely to succeed at 1-degree resolution** – thought precursor sumulations were performed
- ROMS was never run in a global configuration

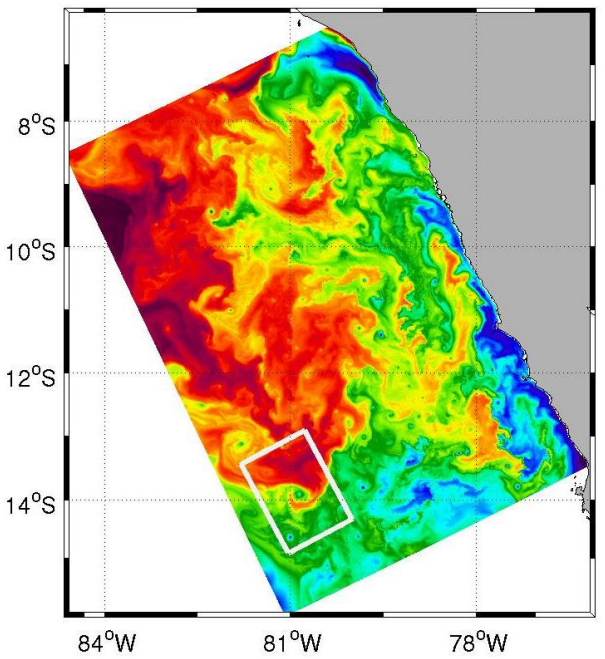


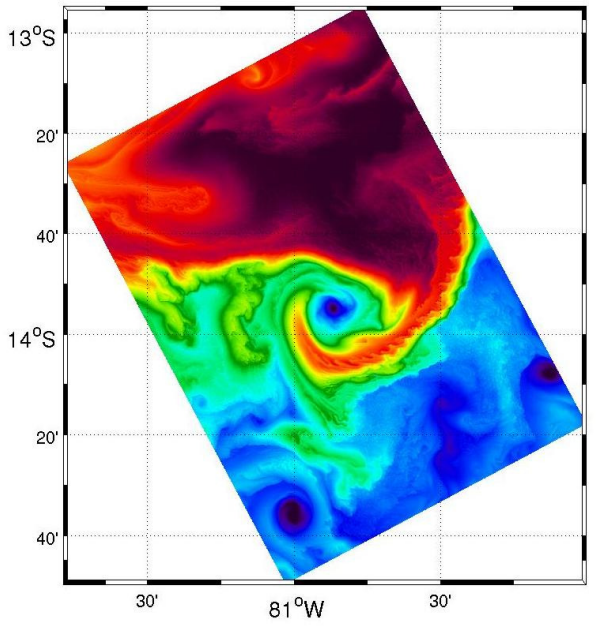


2.2 km

←7.5 km

500 m





180 m

Colas et. al., 2011

Peru-Chile upwelling system: 4-stage downscaling

7.5 km	$384 \times 800 \times 30$	[1992-1999] POP/CCSM ERS COADS Pathfinder
2.2 km	$468 \times 560 \times 30$	[1994-1998] receives 3-day averages, outputs daily
500 m	$1200 \times 1800 \times 42$	[April 1995 - July 1996] receives daily, outputs every 12h
180 m	$694 \times 972 \times 84$	[1st-30th September 1995] receives 12h outputs every 3h

...Алгоритмы: программирование

- параллельный код, двух-уровневое двумерное расщепление на *блоки* (sub-domains), затем каждый и которых ещё и на *плитки* (tiles); MPI или Open MP, либо оба сразу, MPI+OpenMP. Возможность работы на любом компьютере/архитектуре/компиляторе.
- **возможность самоверификации правильности кода:** один или много процессоров должны дать одинаковый результат

- философия *бедного человека* (историческое название из начала 200х когда Linux PC считались дешёвой альтернативой "настоящим" workstation IBM, Sgi, Sun, DecAlpha, etc), внимание на взаимодействие частей кода (включая негативное, т.е. интерференцию); инфраструктура кода сильно отличающаяся от типичных моделей океана (MOM/POP, MITgcm, NEMO, etc) – уход от модульности (в их понимании); характерно объединение мелких блоков в более крупные

- принятие решений подразумевает оптимум учитывающий все факторы: физику, численные алгоритмы, вычислительные затраты, эффективность распараллеливания, и т.д. (уход от *physical scientist* $\rightarrow$ *programming slave*)

- шаг по времени для 3D части построен так чтобы сбалансировать вычислительные затраты между гидродинамическим ядром (адвекция, Кориолис, и градиент давления) и параметеризациями перемешивания (горизонтальными и вертикальными) – для первого используется предиктор-корректор (LF-AM3), для вторых нет (их правые части вычисляются только один раз)
- использование CPP для конфигурации, самотестирования, и т.д.
- **самодокументация** т.е. встроенный механизм учёта статуса всех CPP-ключей без усилия пользователя

ROMS Community

August 1997 as starting date for the project

Аббревиатура ROMS (a.k.a. Regional Oceanic Modeling System) появилась в октябре 1998 г. на конференции в Дависе, Калифорния. Предложена Дейлом Хайдфогелем (Dale Haidvogel)

ROMS evolved (as opposite to *intelligent design*) with full range of consequences of this process:

- from Hernan's SCRUM 3.0 code
- mess, claims, controversy, scrutiny...
- *absence of claims*, ...sometimes
- inheritance, legacies, sometimes irrational choices, branches
- **there is more than one code called ROMS**

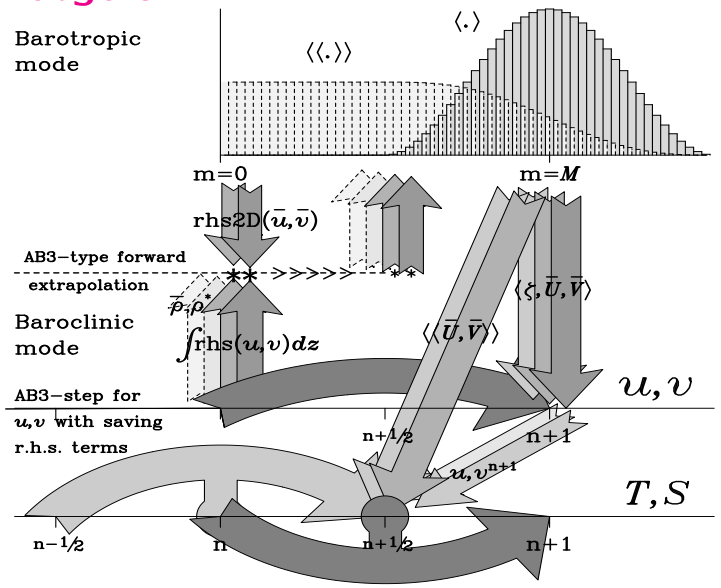
Linux-like-spirited community with multiple development cent*re*s

- exchange of ideas, mutual help network
- **voices to unify and make one "official" code $\rightarrow$ go elsewhere and unify something else**

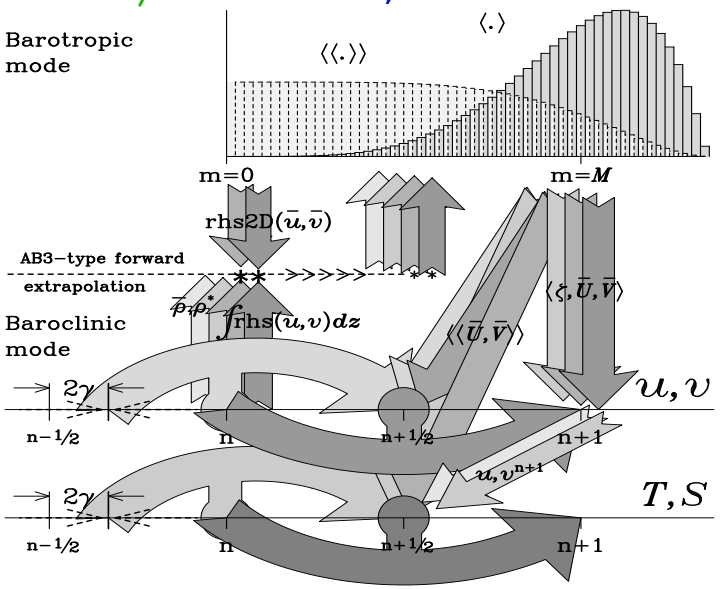
ROMS have survived ... despite (or due?) all of the above

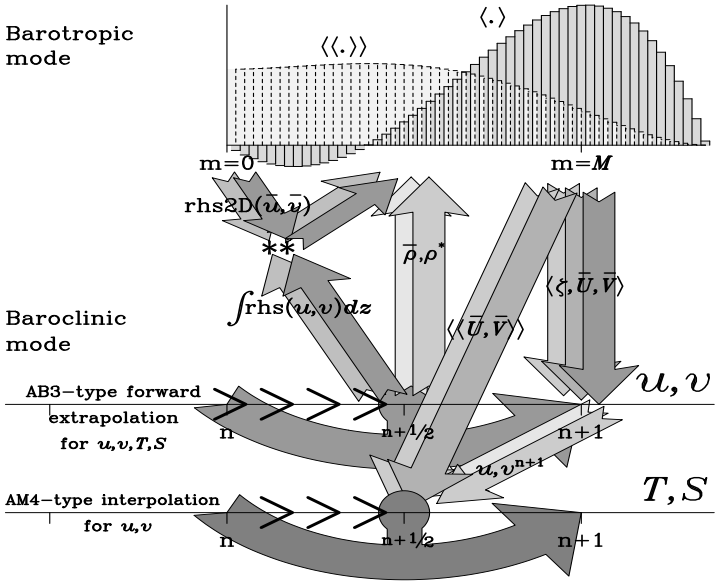
Four variants of ROMS kernel

Rutgers

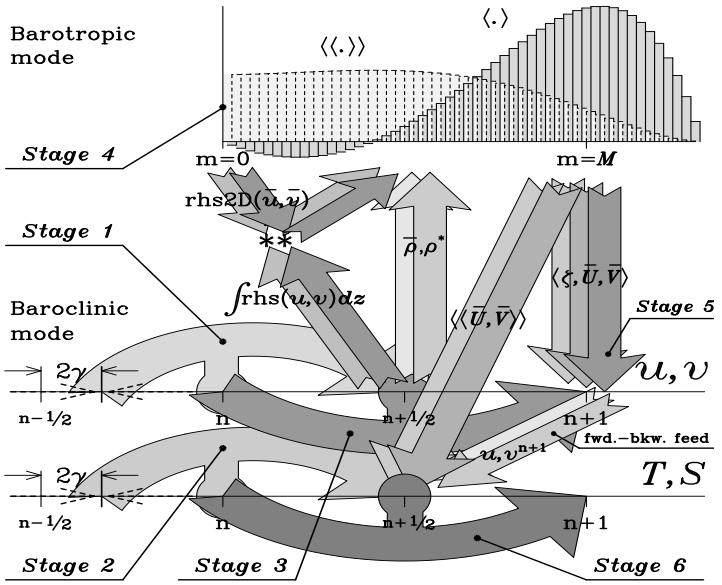


AGRIF / old UCLA; CROCO





Non-hydrostatic code prototype



UCLA (current)

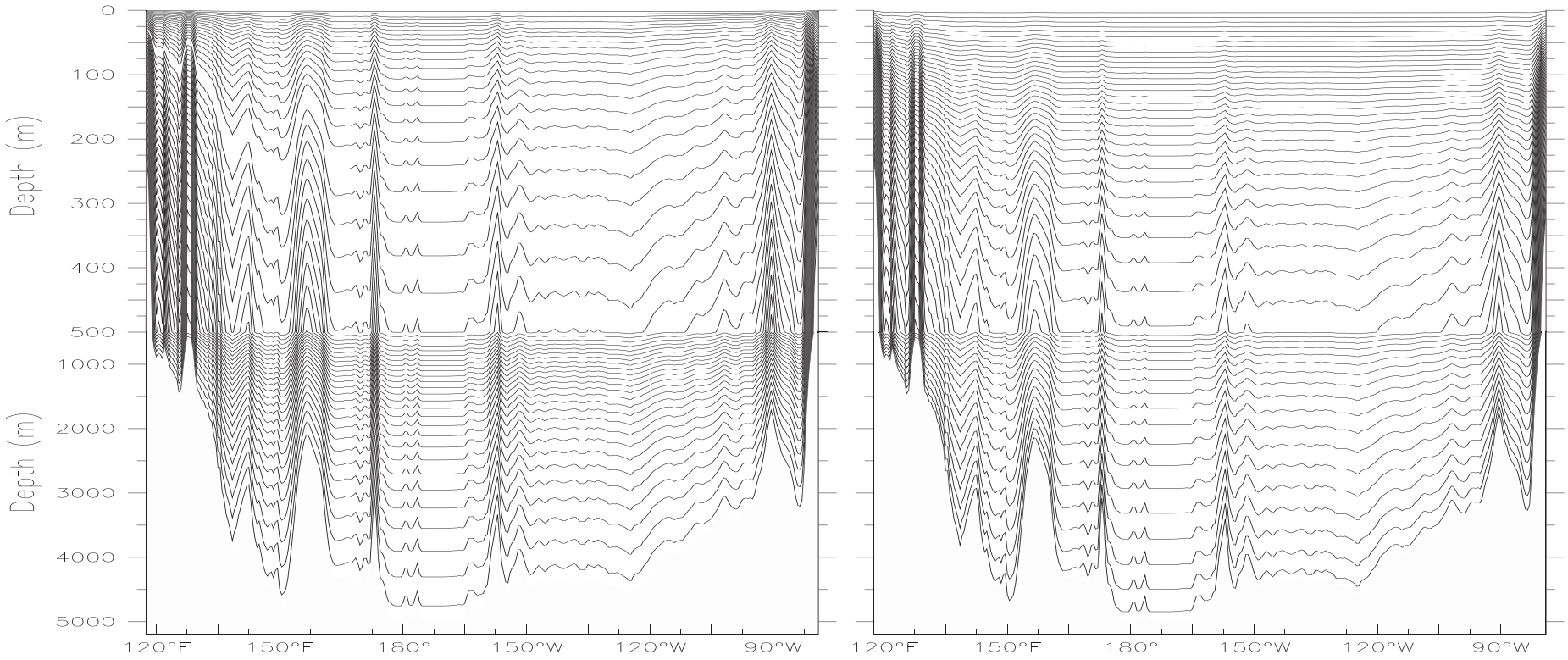
- **z-sigma** $z = z^{(0)} + \zeta\left(1 + \dfrac{z^0}{h}\right)$ $\qquad z^{(0)} = h \cdot \dfrac{h_c \cdot s + h \cdot C(s)}{h + h_c}$ $\qquad h = h(x, y)$

  $h_c = \text{const}$

  $C(s) \equiv C[S(s)]$ where

$$S(s) = \frac{1 - \cosh(\theta_s s)}{\cosh(\theta_s) - 1} \begin{matrix} \nearrow 0 \\ \searrow -1 \end{matrix} \qquad C(S) = \frac{\exp\{\theta_b S\} - 1}{1 - \exp\{-\theta_b\}} \begin{matrix} \nearrow 0, & s = 0, \quad \text{surface} \\ \searrow -1, & s = -1, \quad \text{bottom} \end{matrix}$$

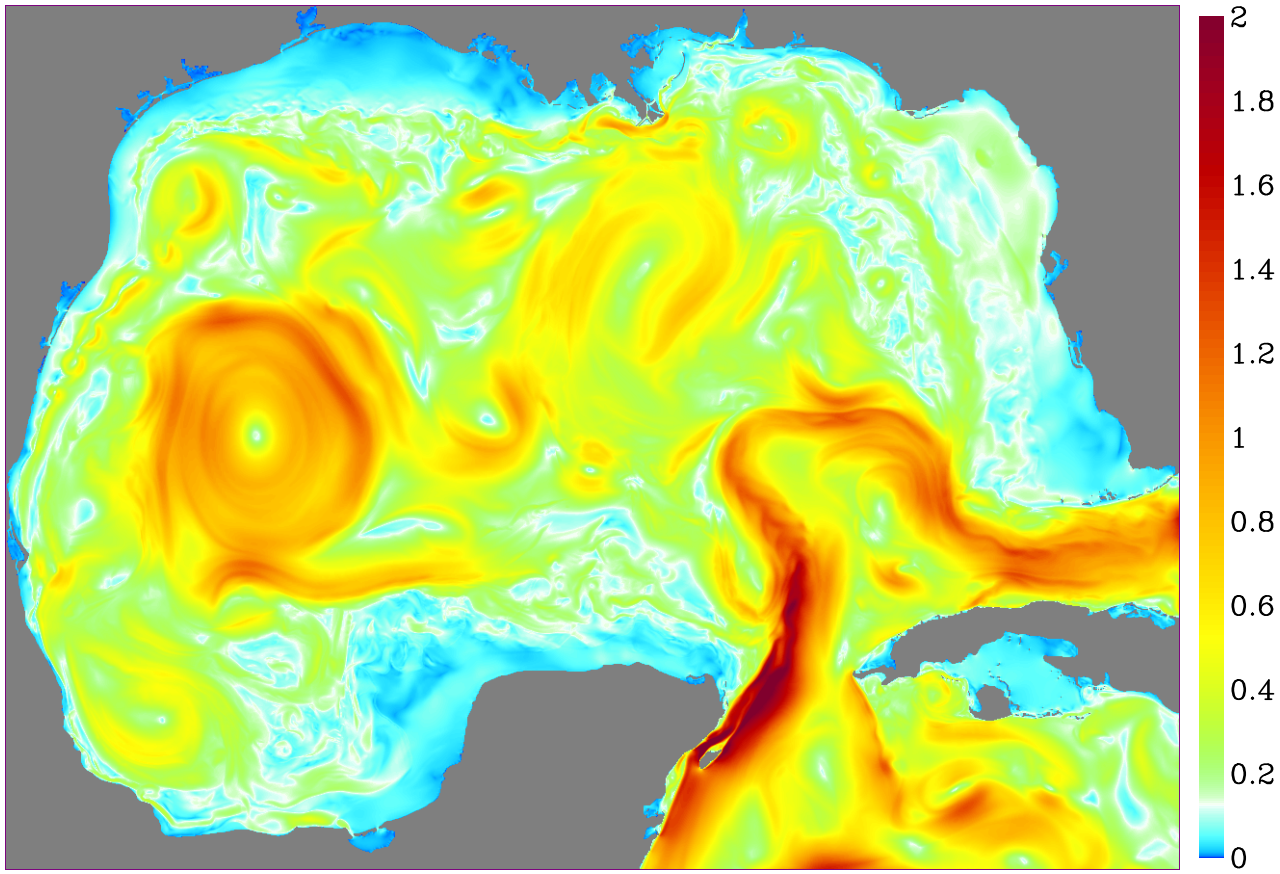
note $\lim\limits_{s \to 0} \dfrac{\partial C(s)}{\partial s} = 0$ and **no restriction** $h_c < h_{\min}$ **!**



Song & Haidvogel (1994), θ_s=6, θ_s=0.15, h_c=50 Lemarié et. al. (2011), θ_s=10, θ_s=2, h_c=400

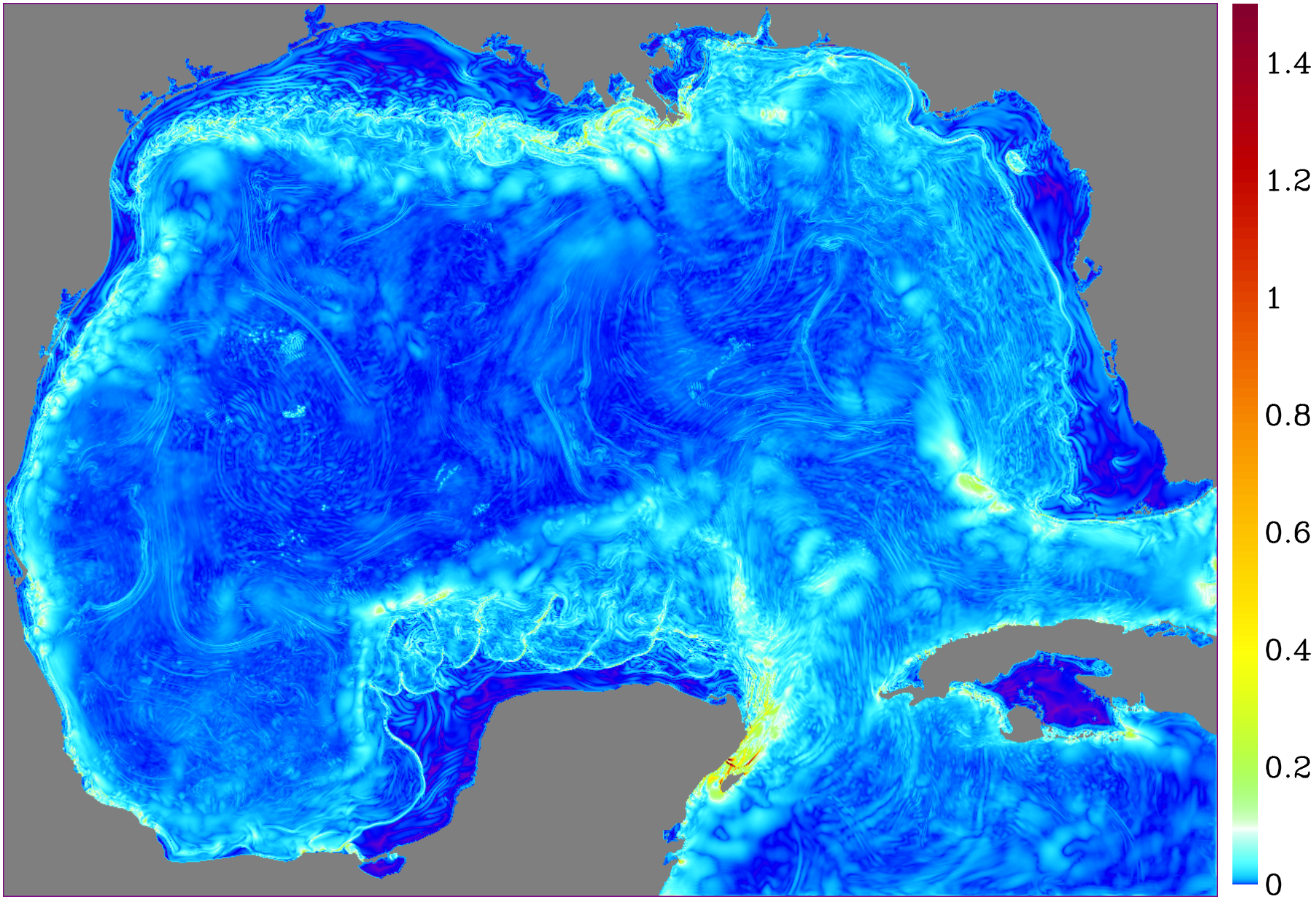
Adaptively implicit vertical advection

The motivation is to keep max horizontal Courant number at healthy 0.6

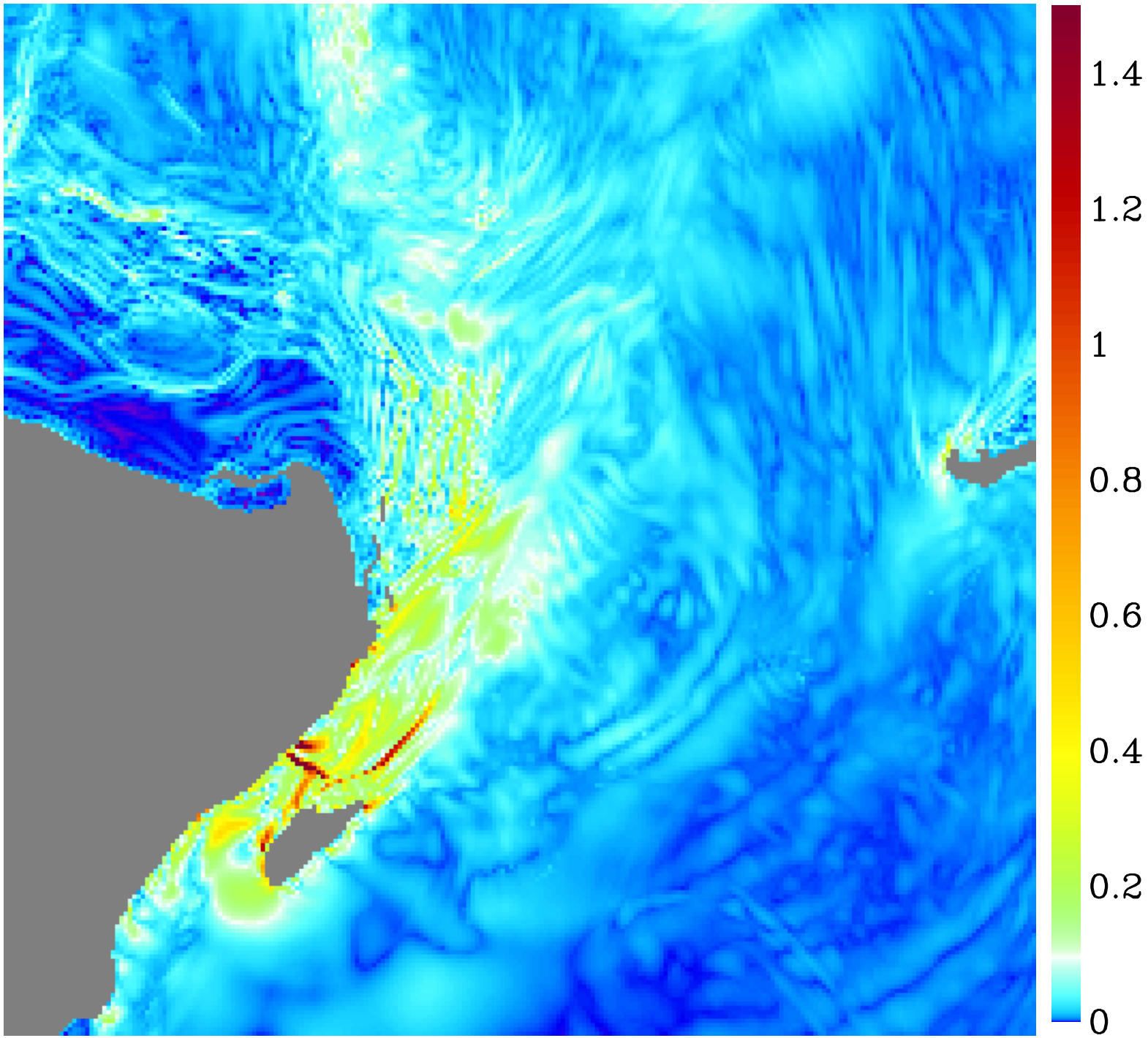


horizontal speed, max in each vertical column

...but it is not always possible.

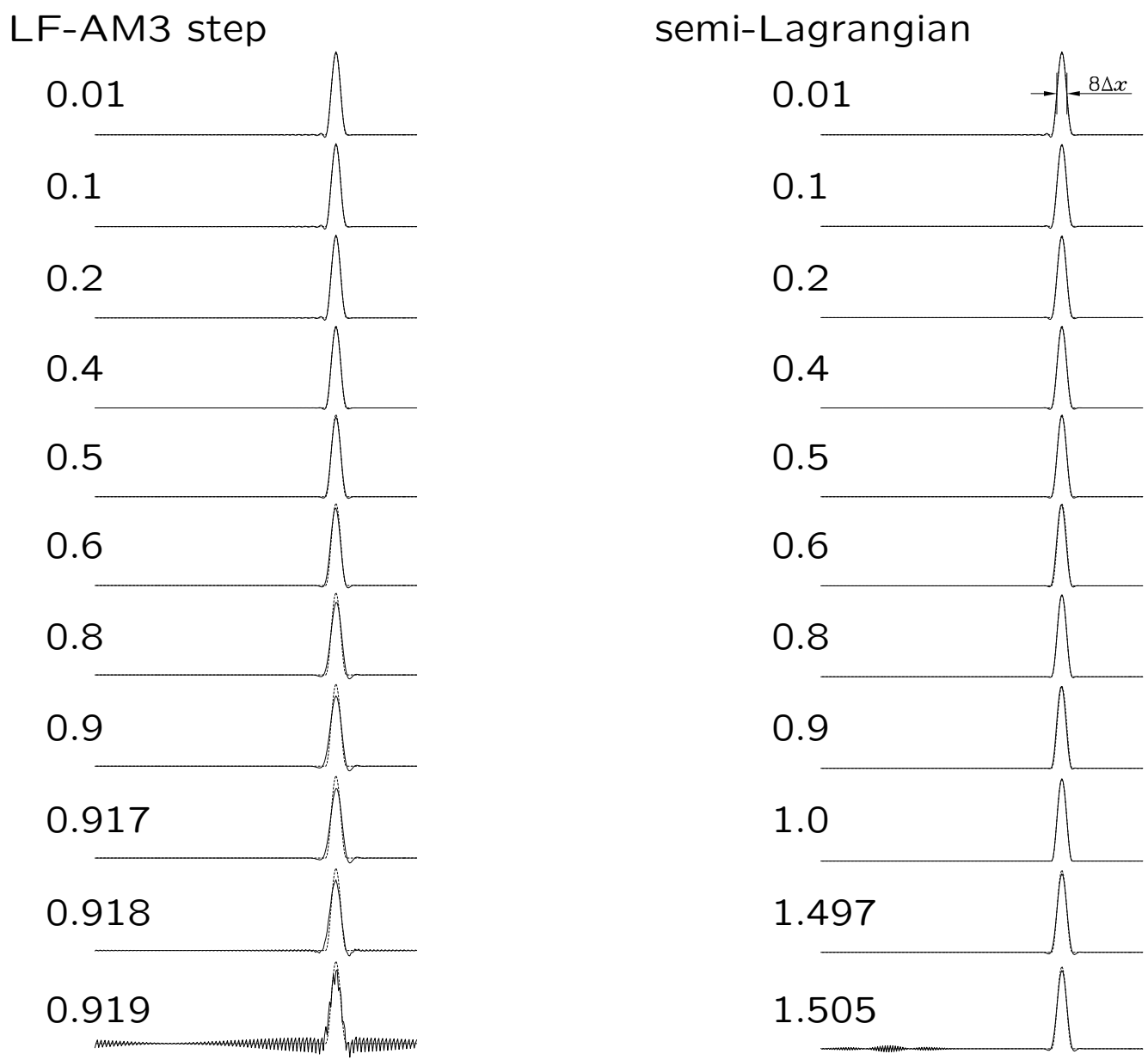


vertical Courant number, max in each vertical column

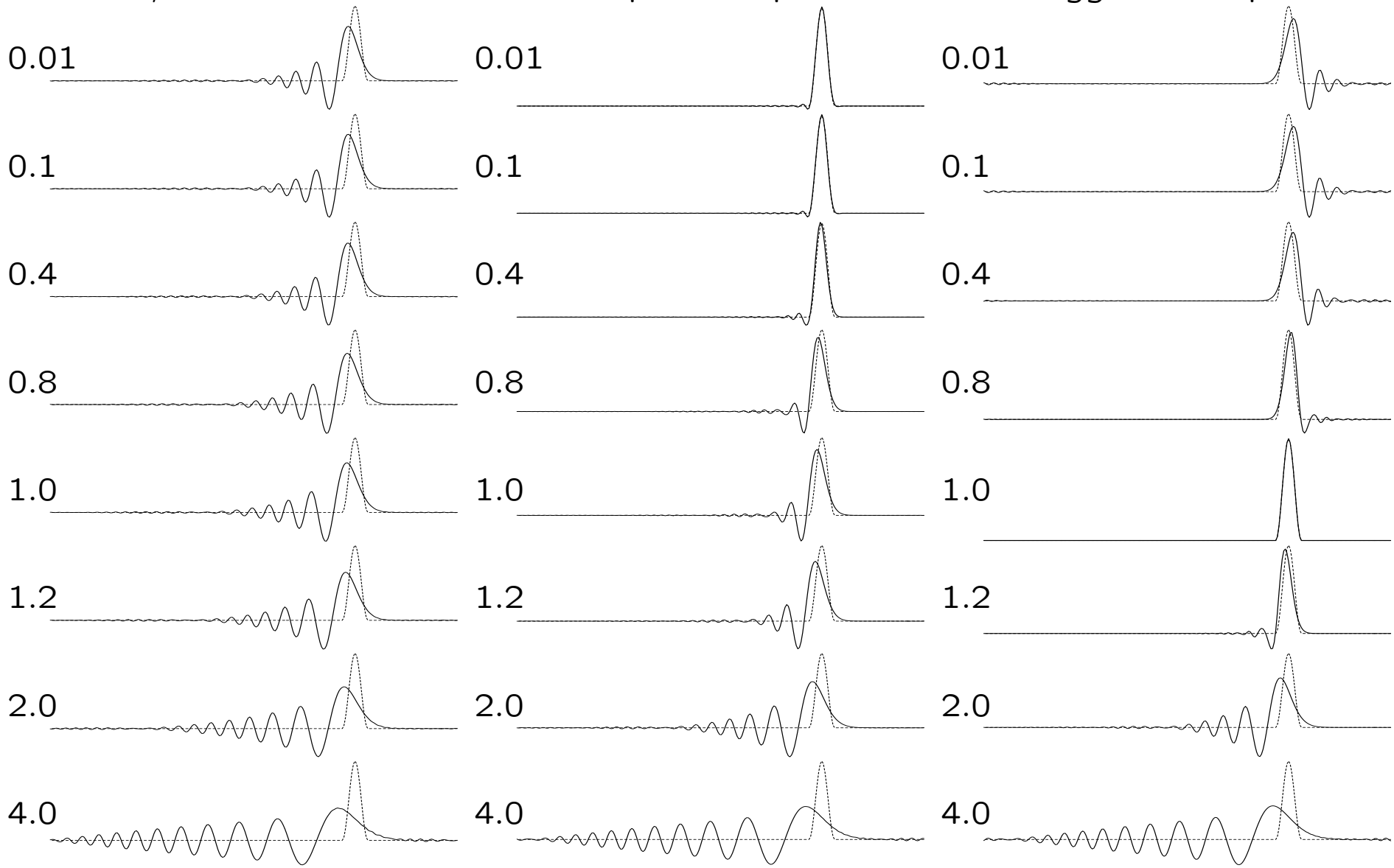

1.4
1.2
1
0.8
0.6
0.4
0.2
0

Why adaptive?

Why not to use an implicit method from a textbook?



Advection a narrow pulse by LF-AM3 (*left column*) and semi-Lagrangian (*right*). **Bold** line numerical solution, *dashed* exact. Number on the left of each panel indicates Courant number, $c\Delta t/\Delta x$.



Advection and dispersive spreading of a narrow pulse by non-dissipative, unconditionally stable implicit schemes using different Courant number regimes. Equal-weight $\theta = 1/2$ Crank-Nicolson stepping in all three cases. *Left column* centered second-order differencing in space; *middle* compact-centered (fourth-order); *right* staggered in space, centered around $(x_j - \Delta x/2, y_n + \Delta t/2)$.

Adaptively implicit vertical advection operator:

Vertical fluxes for the tracer and velocity fields are discretized involving the advected field at $n+1$ time step which are yet unknown,

$$FC_{k+1/2} = W_{k+1/2} \cdot \mathscr{Q}\left(q_k^{n+1}, q_{k\pm1}^{n+1}, q_k^{n+1/2}, q_{k\pm1}^{n+1/2}, ...\right)$$

rearrange by splitting vertical velocity into two parts,

$$W_{k+1/2} = W^{(e)}_{k+1/2} + W^{(i)}_{k+1/2}, \qquad \forall k = 0, 1, ..., N$$

where $W^{(i)}_{k+1/2}$ is for terms involving $q_k^{n+1}, q_{k\pm1}^{n+1}$ only (*i.e.*, implicit part), while $W^{(e)}_{k+1/2}$ for the remaining $q_k^{n+1/2}, q_{k\pm1}^{n+1/2}, ...$, then:

- **$W^{(e)}$-terms are computed within r.h.s via standard algorithm**
- **$W^{(i)}$-terms are integrated into the vertically implicit operator**

Combined advection-diffusion with upstream treatment of the implicit advection part

$$FC^{(i)}_{k+1/2} = W^{(i)}_{k+1/2} \cdot \begin{cases} q^{n+1}_k, & \text{if} \quad W^{(i)}_{k+1/2} > 0 \\ q^{n+1}_{k+1}, & \text{if} \quad W^{(i)}_{k+1/2} < 0 \end{cases}$$

$k = N$, uppermost grid box,

$$H^{n+1}_N q^{n+1}_N = H^n_N q^n_N + \Delta t \cdot \text{rhs}'_N + \Delta t \cdot SRFRC - \Delta t A_{N-1/2} \frac{q^{n+1}_N - q^{n+1}_{N-1}}{\Delta z_{N-1/2}}$$
$$+\Delta t \left[\max\left(W^{(i)}_{N-1/2}, 0\right) q^{n+1}_{N-1} + \min\left(W^{(i)}_{N-1/2}, 0\right) q^{n+1}_N \right]$$

$k = 2, ..., N-1$

$$H^{n+1}_k q^{n+1}_k = H^n_k q^n_k + \Delta t \cdot \text{rhs}'_k + \Delta t A_{k+1/2} \frac{q^{n+1}_{k+1} - q^{n+1}_k}{\Delta z_{k+1/2}}$$
$$-\Delta t \left[\max\left(W^{(i)}_{k+1/2}, 0\right) q^{n+1}_k + \min\left(W^{(i)}_{k+1/2}, 0\right) q^{n+1}_{k+1} \right]$$
$$-\Delta t A_{k-1/2} \frac{q^{n+1}_k - q^{n+1}_{k-1}}{\Delta z_{k-1/2}}$$
$$+\Delta t \left[\max\left(W^{(i)}_{k-1/2}, 0\right) q^{n+1}_{k-1} + \min\left(W^{(i)}_{k-1/2}, 0\right) q^{n+1}_k \right]$$

$k = 1$, bottom grid box,

$$H^{n+1}_1 q^{n+1}_1 = H^n_1 q^n_1 + \Delta t \cdot \text{rhs}'_1 + \Delta t A_{3/2} \frac{q^{n+1}_2 - q^{n+1}_1}{\Delta z_{3/2}}$$
$$-\Delta t \left[\max\left(W^{(i)}_{3/2}, 0\right) q^{n+1}_1 + \min\left(W^{(i)}_{3/2}, 0\right) q^{n+1}_2 \right]$$

The W-splitting works as follows: Compute $W_{k+1/2}$ the standard way; also compute finite-volume Courant number $\alpha_{i,j,k}$ at every grid box H_k as the sum of outgoing fluxes normalized by Δt and grid-box volume,

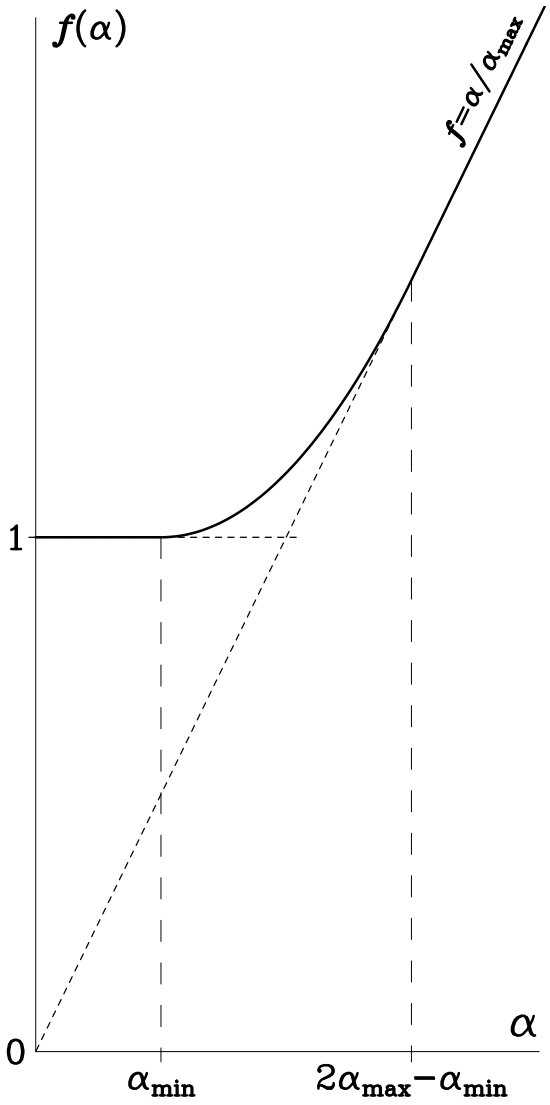
$$\alpha_{i,j,k} = \frac{\Delta t}{\Delta V_{i,j,k}} \cdot \Big[\max(FlxU_{i+1/2,j,k}, 0) - \min(FlxU_{i-1/2,j,k}, 0) + \max(FlxV_{i,j+1/2,k}, 0) - \min(FlxV_{i,j-1/2,k}, 0) + \max(W_{i,j,k+1/2}, 0) - \min(W_{i,j,k-1/2}, 0) \Big]$$

then the explicit part

$$W^{(e)}_{k+1/2} = \frac{W_{k+1/2}}{f(\alpha^*)}, \quad \text{where} \quad \begin{cases} \alpha^* = \alpha_k & \text{if} \quad W_{k+1/2} > 0 \\ \alpha^* = \alpha_{k+1} & \text{if} \quad W_{k+1/2} < 0 \end{cases}$$

and the limiting function

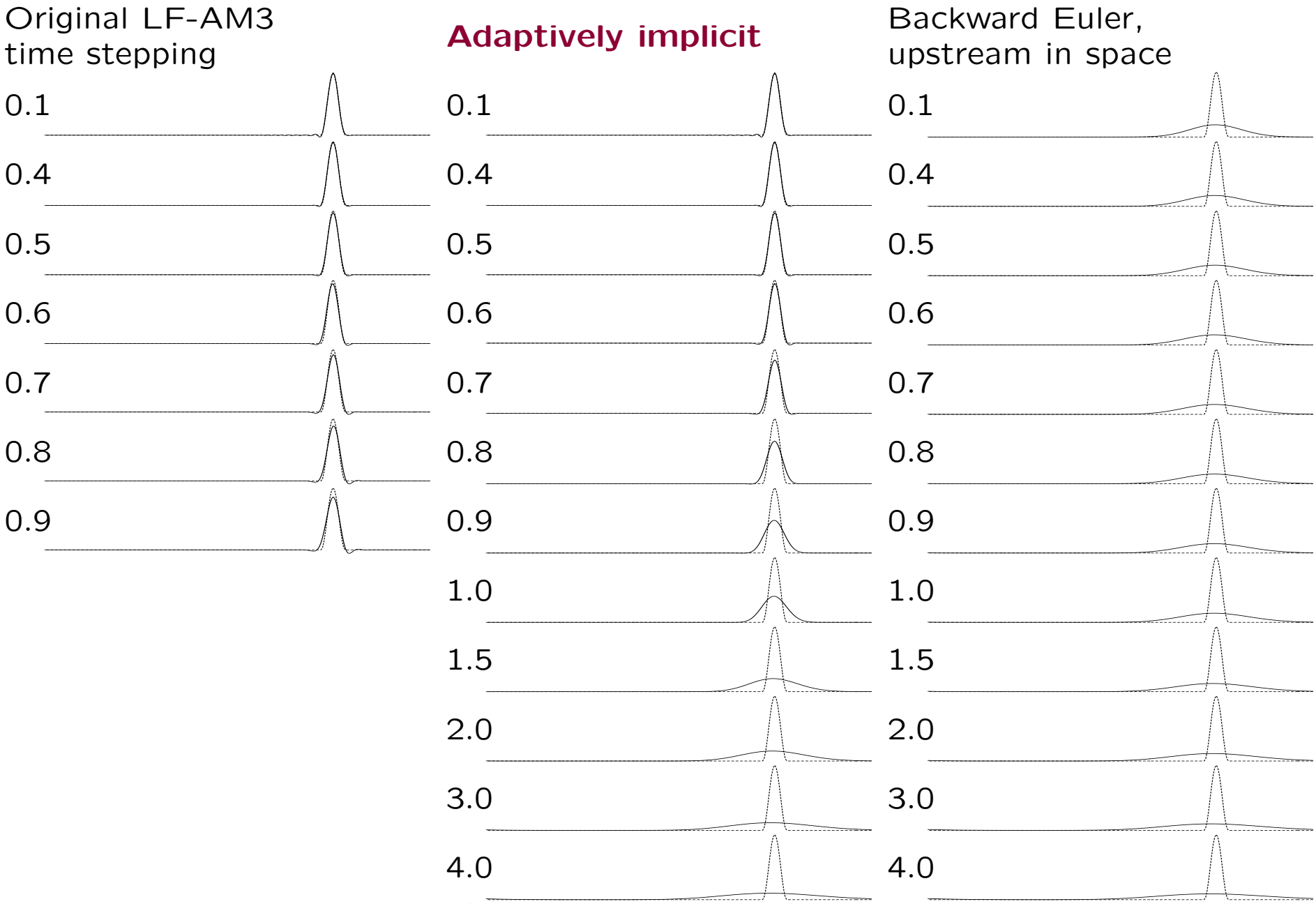
$$f(\alpha) = \begin{cases} 1\,, & \text{if} \quad \alpha \le \alpha_{\min} \\ 1 + \dfrac{(\alpha - \alpha_{\min})^2}{4\alpha_{\max}\,(\alpha_{\max} - \alpha_{\min})}\,, & \text{if} \quad \alpha_{\min} < \alpha < 2\alpha_{\max} - \alpha_{\min} \\ \alpha/\alpha_{\max}\,, & \text{if} \quad \alpha \ge \alpha_{\max} \end{cases}$$



made of three segments – constant, parabolic, and linear – smoothly matched to each other; $\alpha_{\min}$ control the threshold below which the algorithm is fully explicit; $\alpha_{\max}$, and the maximum allowed Courant number **"never exceed speed"** for the explicit part
The implicit part is the "excess" velocity

$$W^{(i)}_{k+1/2} = W_{k+1/2} - W^{(e)}_{k+1/2}$$

Selectable $\alpha_{\min}$ and $\alpha_{\max}$ based on consideration of accuracy and numerical stability of the explicit part.
In the actual code all the above – computing $\alpha = \alpha_{i,j,k}$, then $f(\alpha)$ then splitting W is implanted into the computation of W itself, so none of the intermediates is stored as a 3D array.



Comparison of LF-AM3 algorithm (*left*, adaptively implicit (*middle*, threshold Courant numbers settings $\alpha_{\min} = 0.6$, $\alpha_{\max} = 1.0$), and fully-implicit backward Euler upstream in space advection (*right*).

Prime in rhs'_k means that the usual r.h.s. computed by ROMS code for the corresponding equations, except the replacement $W_{k+1/2} \to W^{(e)}_{k+1/2}$

$A_{k+1/2}$ is vertical viscosity/diffusion coefficient [including the stabilization terms (Lemarié et. al., 2012) in the case when isoneutral lateral diffusion is used]

The above takes into account kinematic b.c. at surface and bottom, $W_{N+1/2} = W_{1/2} = 0$, bottom no-flux b.c. for tracers. There is an extra term for momentum equation associated with bottom drag which also treated implicitly.

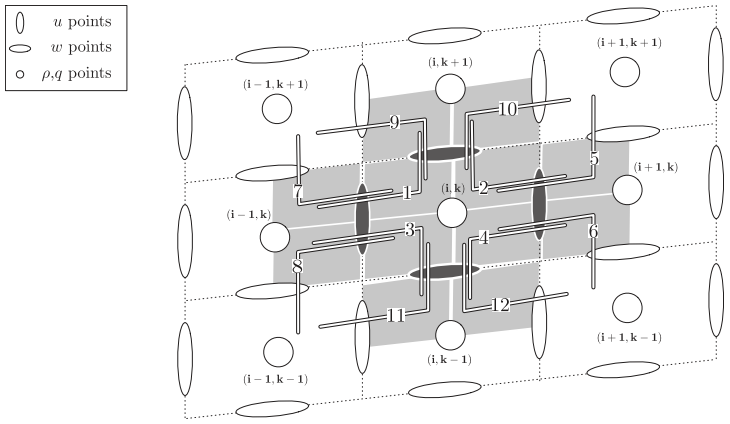
The modified algorithm **retains simultaneous conservation and constancy preservation properties for tracers**, despite the fact that grid box heights change due to changing free surface, $H^{n+1}_k \neq H^n_k$.

The motivation for using upstream discretization for the implicit part comes from the fact that it is monotonic, hence will not cause oscillation. Unavoidably it is diffusive, however this choice is justified by the observation that in practical model solutions large vertical velocities occur only in places with vanishing (or even unstable) stratification and, consequently, already large mixing set by the vertical parameterization scheme.

Well posed, diagonally dominant discrete system.

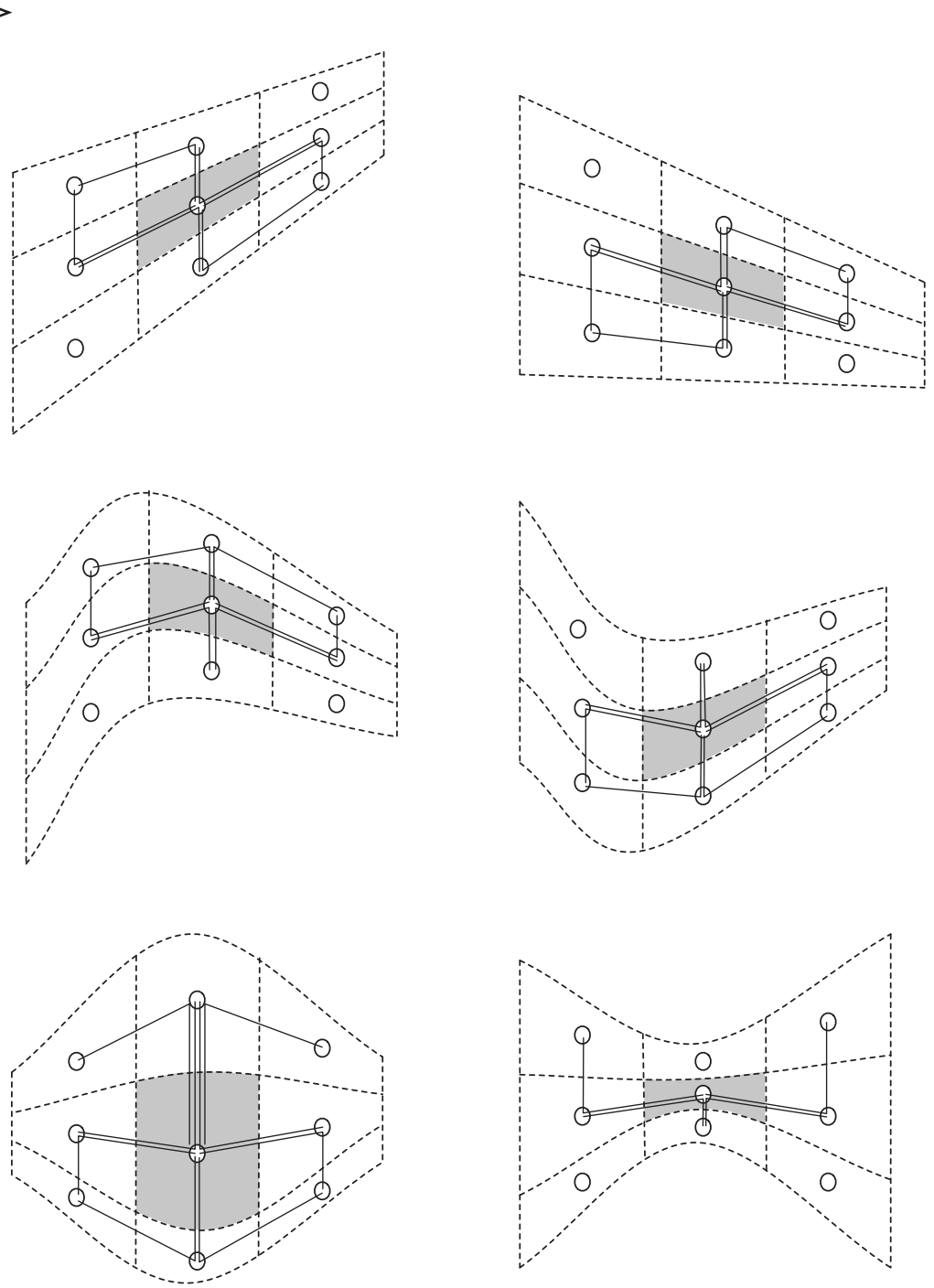
Rotated diffusion in sigma coordinates

Rotated isoneutral diffusion: explaining switching triads

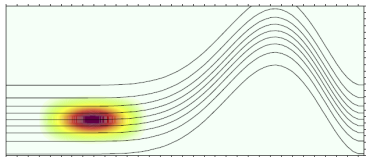


above: Griffies et. al. (1998) discrete isoneutral diffusion operator using all 12 triads around tracer point (they are arranged into two pairs of fluxes, horizontal and "vertical", each involving 4 triads)

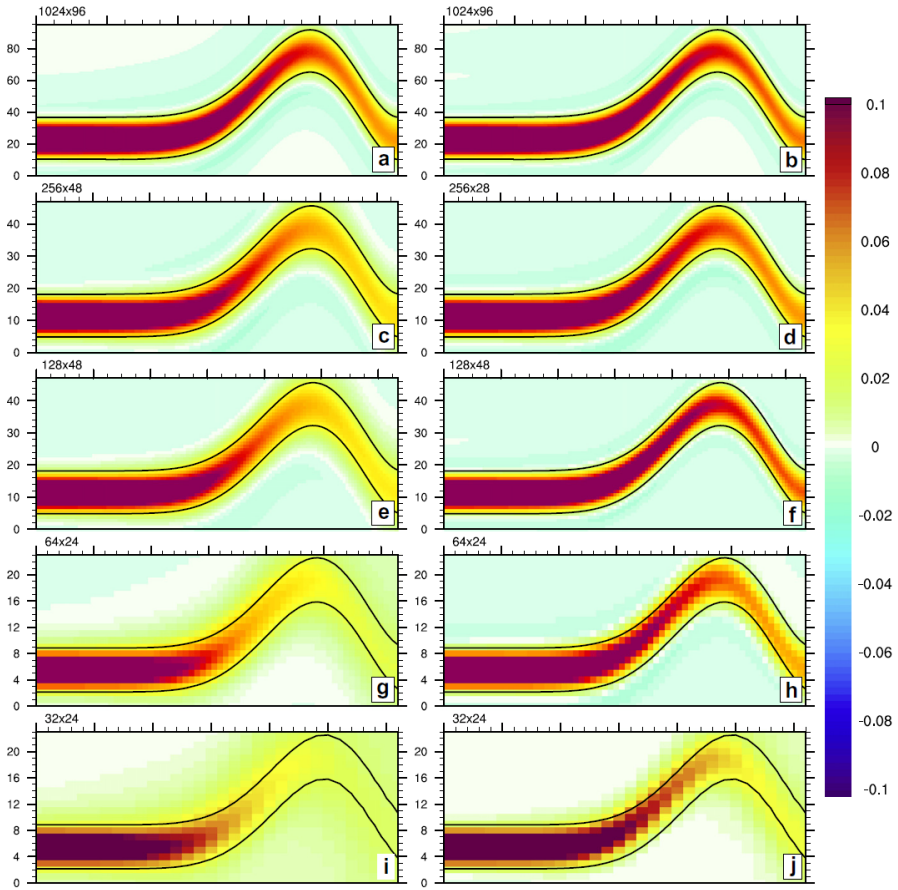
right: Lemarié et. al. (2012) switching triads in all possible situations. Note that vertical axis here is "density", so isoneutral direction is horizontal in all these charts; **only triads with sharp angle** ($< 90^O$) are selected. "exact" if grid-point slope $s = 1$



Rotated isoneutral diffusion: convergence comparison

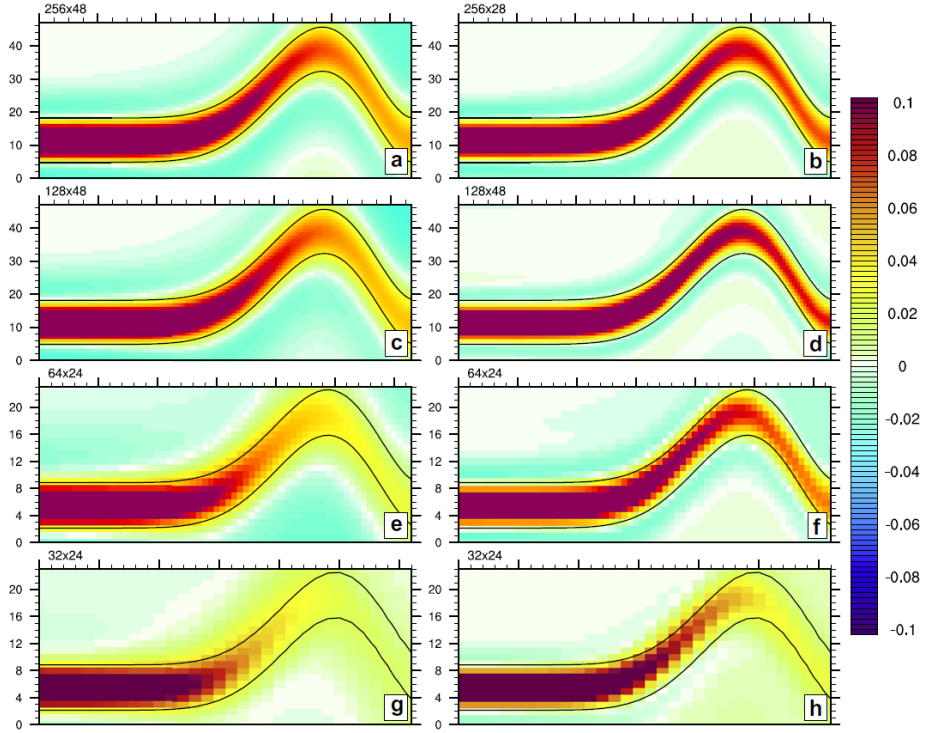


initial condition for smooth field test problem



non-switched switched

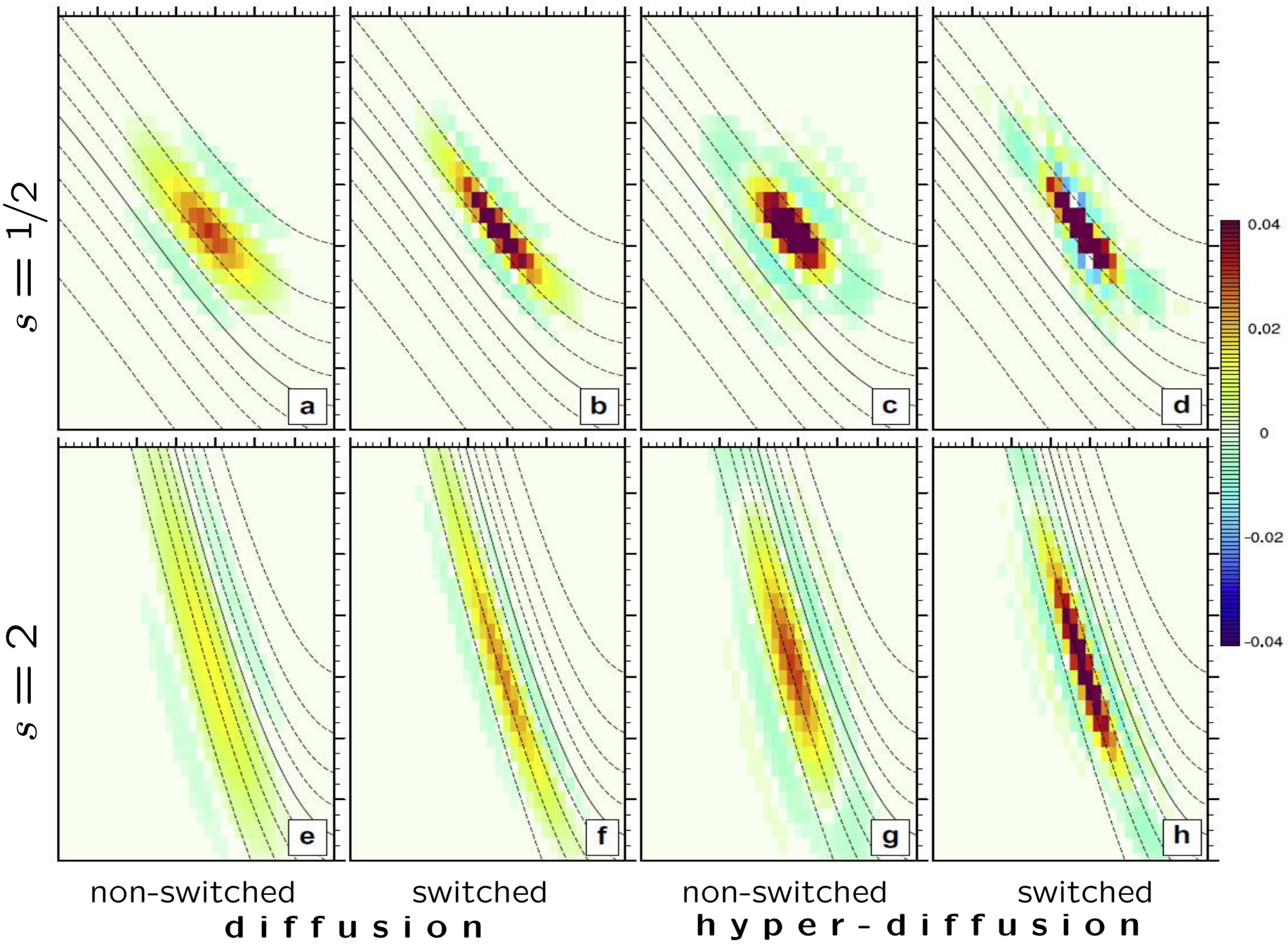
d i f f u s i o n



non-switched switched

h y p e r - d i f f u s i o n

Rotated isoneutral diffusion: spread of a single dot



"If-less" KPP via integral criterion alternative to Ri_b

Criterion for finding $h_{\rm bl}$: Define surface PBL as an integral layer within which net production of turbulence due to shear-layer instability is balanced by dissipation due to stratification,

$$Cr(z) = \int_z^{\rm surface} \mathscr{K}(z) \left\{ \left| \frac{\partial \mathbf{u}}{\partial z} \right|^2 - \frac{N^2}{\mathtt{Ri}_{\rm cr}} - C_{\rm Ek} \cdot f^2 \right\} dz' + \frac{V_t^2(z)}{z}$$

and search for crossing point $Cr(z) = 0$.

$N^2 = -\dfrac{g}{\rho_0} \cdot \left. \dfrac{\partial \rho}{\partial z} \right|_{ad}$ is B-V frequency ($ad \equiv$ adiabatic);

f is Coriolis parameter; $C_{\rm Ek}$ is a nondimensional constant;

$V_t^2(z)$ is unresolved turbulent velocity shear (same as in LMD94);

Integration Kernel $\mathscr{K}(z) = \dfrac{\zeta - z}{\epsilon h_{\rm bl} + \zeta - z}$ is to ignore contribution from near-surface sublayer $\epsilon h_{\rm bl}$ where M-O similarity law is not valid (plays the same role as to distinguish between $\rho_{\rm ref}$ vs. $\rho_{\rm surf}$ in Ri_b of LMD94). ζ is free surface; $\epsilon = 0.1$.

"If-less" KPP...

- Same result as Ri_b the case of linear velocity profile, but otherwise

$$\int_{z'}^{z''} \left| \frac{\partial \mathbf{u}}{\partial z} \right|^2 dz \geq \frac{|\mathbf{u}'' - \mathbf{u}'|^2}{z'' - z'}$$

- $Cr(z)$ is **monotonic** for Ekman spiral $\Rightarrow$ no sudden jumps of h_{bl}
- Numerically more attractive, as $\mathbf{u}(z)$ and $\rho(z)$ can be reconstructed as continuous functions
- Avoids introduction of *reference* potential density: basically integration square of Brunt-Väisäla frequency. Allows formalism of *adiabatic* derivatives and differences to achieve monotonicity
- Correct account for thermobaric effect: Bill Large: to determine extent of BL one must bring water parcel **from reference depth to** $z = -h_{\mathsf{bl}}$ and compare its density with the ambient fluid **there**. We never did it this way in ROMS community (?)
- Avoids ambiguity for merging top and bottom BLs

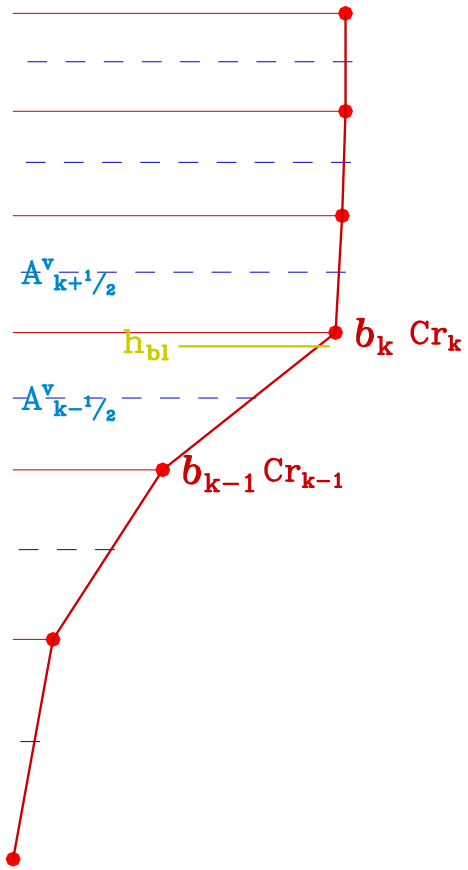
"If-less" KPP...

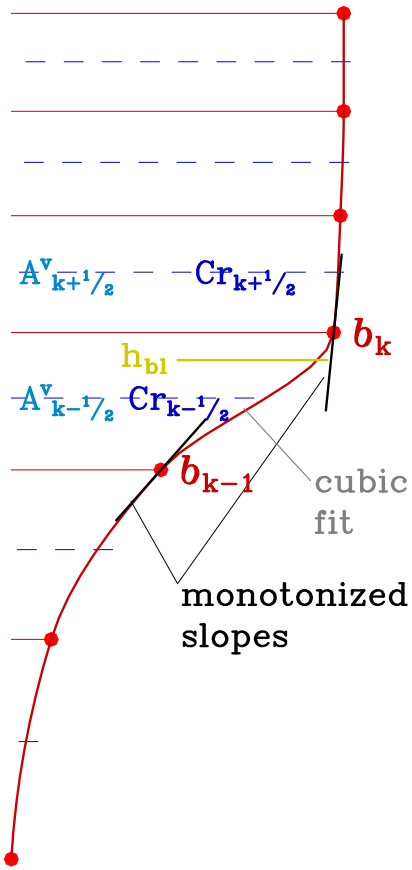
Pure physical limits: destabilizing vs. stabilizing effects:

- balance $\left|\frac{\partial \mathbf{u}}{\partial z}\right|^2$ vs. $\frac{N^2}{\mathtt{Ri}_{\mathsf{cr}}}$ $\Rightarrow$ shear layer instability

- $\left|\frac{\partial \mathbf{u}}{\partial z}\right|^2$ vs. $C_{\mathsf{Ek}} \cdot f^2$ $\Rightarrow$ turbulent Ekman layer

- *negatively* forced $\frac{N^2}{\mathtt{Ri}_{\mathsf{cr}}}$ vs. V_t^2 $\Rightarrow$ free convection





Monotonized reconstruction to compute $V_t^2{}_{k+1/2}$ and $\rho_{k+1/2}$, but **not** to interpolate Cr to find h_{bl}: because

$$Cr \sim w_s \sqrt{N^2 - N^2 d}$$

$Cr(z)$ is **not monotonic** near

$$z = -h_{\text{bl}}$$

even if $\rho(z)$ and $u, v(z)$ are

$\Rightarrow$ quadratic (cubic) interpolation for Cr is dangerous

Overall this is by far the largest cause of numerical sensitivities in KPP.

"If-less" KPP... Monin-Obukhov depth limit $h_{\mathrm{bl}} \leq h_{\mathrm{MO}} = \dfrac{C^{\mathrm{MO}} \cdot u_*^3}{\kappa \cdot B_{\mathrm{f}}}$ if $B_{\mathrm{f}}(z) > 0$.

Because of solar radiation absorption, buoyancy forcing $B_{\mathrm{f}} = B_{\mathrm{f}}(z)$ increases with depth, possibly changing sign *from unstable to stable*
$\Rightarrow$ a case when $B_{\mathrm{f}}\left(-\text{overestimated } h_{\mathrm{bl}}\right) > 0$, but $B_{\mathrm{f}}\left(-h_{\mathrm{MO}}\right) < 0$
$\Rightarrow$ **hysteresis** and oscillations in h_{bl}

solution # 1: use $B_{\mathrm{f}} = B_{\mathrm{f}}\left(-h_{\mathrm{MO}}\right)$ in computation of h_{MO}, i.e. implicit search for k enclosing z^*, such that

$$z_k \leq z^* \leq z_{k+1} \qquad \text{and} \qquad h_{\mathrm{MO}}\left(z_k\right) \leq |z^*| \leq h_{\mathrm{MO}}\left(z_{k+1}\right)$$

then solve $$\frac{h_{\mathrm{MO}k}\left(z_{k+1} - z^*\right) + h_{\mathrm{MO}k+1}\left(z^* - z_k\right)}{z_{k+1} - z_k} + z^* = 0$$

resulting in $$h_{\mathrm{MO}} = -z^* = \frac{\frac{C^{\mathrm{MO}} u_*^3}{\kappa}\left(B_{\mathrm{f}}{}'_{k+1} z_{k+1} - B_{\mathrm{f}}{}'_k z_k\right)}{B_{\mathrm{f}}{}'_{k+1} B_{\mathrm{f}}{}'_k \left(z_{k+1} - z_k\right) + \frac{C^{\mathrm{MO}} u_*^3}{\kappa}\left(B_{\mathrm{f}}{}'_k - B_{\mathrm{f}}{}'_{k+1}\right)}$$

above $B_{\mathrm{f}}{}' = \max\left(B_{\mathrm{f}}, 0\right)$; if k not found $\Rightarrow$ no limit; **no singularity** if either $B_{\mathrm{f}} \to 0$; limit applied **outside** $B_{\mathrm{f}} > 0$ **logic**: it is already taken into account in computing h_{MO}; **since** h_{bl} **is not involved** $\Rightarrow$ **no possibility of hysteresis**

solution # 2: Eliminate M-O limit altogether.

"If-less" KPP... Ekman depth limitation: $h_{\rm bl} \le h_{\rm Ek} = 0.7\dfrac{u_*}{f}$ applicable **only for stable** boundary layer; should be $h_{\rm bl} = h_{\rm Ek}$ for **neutral** forcing and stratification

Length $\mathscr{L} = u_*/f$ and velocity $\mathscr{U} = u_*$ are natural scaling parameters for neutrally stratified problem

$$i \cdot f\mathbf{u} = \frac{\partial}{\partial z}\left(w_m|z|\frac{\partial \mathbf{u}}{\partial z}\right)$$

where $\mathbf{u} = u + iv$, and $w_m \equiv \kappa u_*$, and κ is von Karman constant.

- Most vertical mixing schemes are "Coriolis-blind" any way.
- Coriolis effect plays no role in determining $h_{\rm bl}$ via bulk Ri criterion; $h_{\rm Ek}$-limit is applied *a'posteriori*, and only for stable buoyancy forcing.
- Because of light absorption, **stability increases downward resulting in hysteresis** if $B_{\rm f}$ (unlimited $h_{\rm bl}$) > 0, but $B_{\rm f}\,(h_{\rm Ek}) < 0$ which is manifested by $h_{\rm bl}$ oscillations and jumps

??? integrate $h_{\rm Ek}$-limit into KPP BL criterion, balance

$$\int \left|\frac{\partial \mathbf{u}}{\partial z}\right|^2 dz' \qquad \text{vs.} \qquad \int f^2 dz' \qquad ???$$

"If-less" KPP... Modified Ekman problem $\quad i \cdot f\mathbf{u} = \dfrac{\partial}{\partial z}\left[w_m \mathscr{L} G\left(\dfrac{z}{\mathscr{L}}\right)\dfrac{\partial \mathbf{u}}{\partial z}\right]$

G is KPP non-dimensional shape function

$$G(\sigma) = |\sigma|\,(1-\sigma)^2 + \begin{cases} \frac{(\sigma-\sigma_0)^2}{2\sigma_0}, & \sigma < \sigma_0 \\ 0 & \text{otherwise} \end{cases} \qquad \sigma_0 = 0.1$$

B.C.:
$$w_m \mathscr{L} G\left(\frac{z}{\mathscr{L}}\right)\left.\frac{\partial \mathbf{u}}{\partial z}\right|_{z=0} = u_*^2 \mathbf{1}_\tau \qquad \Rightarrow \qquad \left.\frac{\partial \mathbf{u}}{\partial z}\right|_{z=0} = \frac{u_* \mathbf{1}_\tau}{\kappa \mathscr{L} \sigma_0/2}$$

$\mathbf{u} = 0$, if $z < -\mathscr{L}$

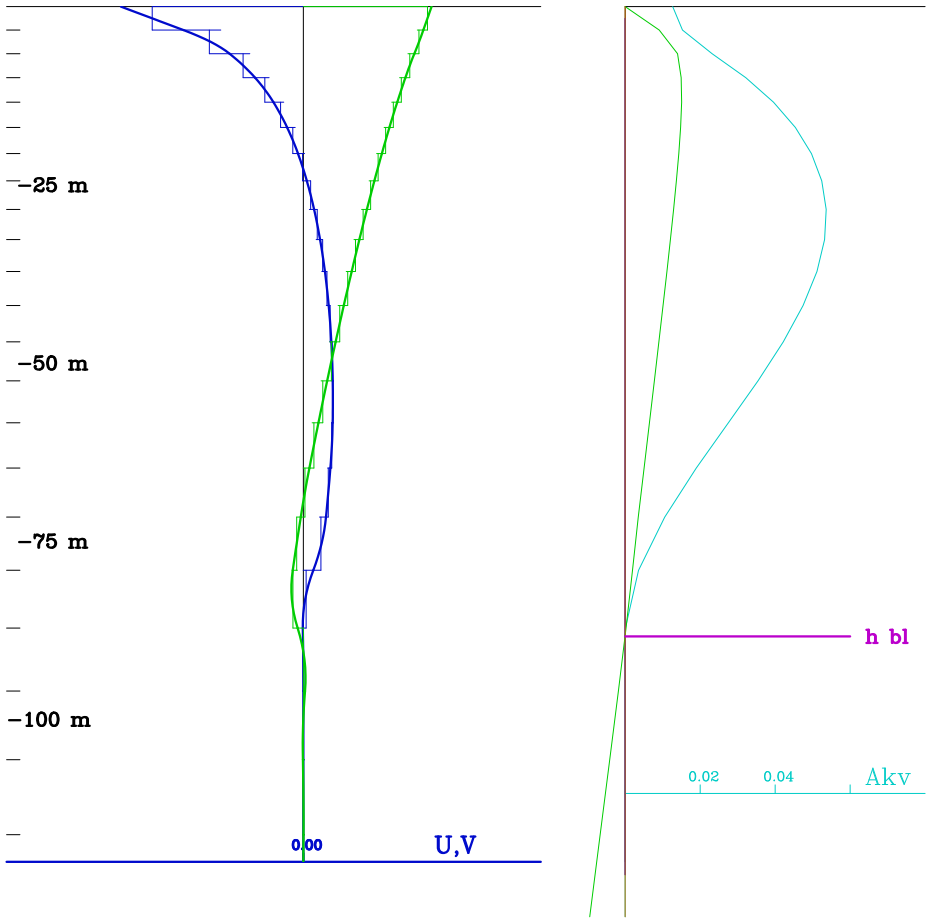
Nondimensionalization: Postulate that depth of generated this way boundary layer is equal to Ekman length and introduce scaling,

$$z = \mathscr{L}\sigma = \sigma \cdot 0.7 u_*/f \qquad\qquad \mathbf{u} = u_* \cdot \tilde{\mathbf{u}}\,,$$

hence

$$\frac{\partial}{\partial \sigma}\left(G(\sigma)\frac{\partial \tilde{\mathbf{u}}}{\partial \sigma}\right) = i \cdot \frac{\kappa}{0.7}\tilde{\mathbf{u}}\,, \qquad \left.\frac{\partial \tilde{\mathbf{u}}}{\partial \sigma}\right|_{\sigma=0} = \frac{2}{\kappa \sigma_0}\,, \qquad \left.\tilde{\mathbf{u}}\right|_{\sigma<-1} = 0$$

everything has been scaled out.



Recognize Coriolis force as a *stabilizing* factor (balancing vertical shear production), construct

$$Cr(z) = \int_z^{\text{surf}} \mathscr{K}(z') \left\{ \left| \frac{\partial \mathbf{u}}{\partial z} \right|^2 - C_{\text{Ek}} \cdot f^2 \right\} dz'$$

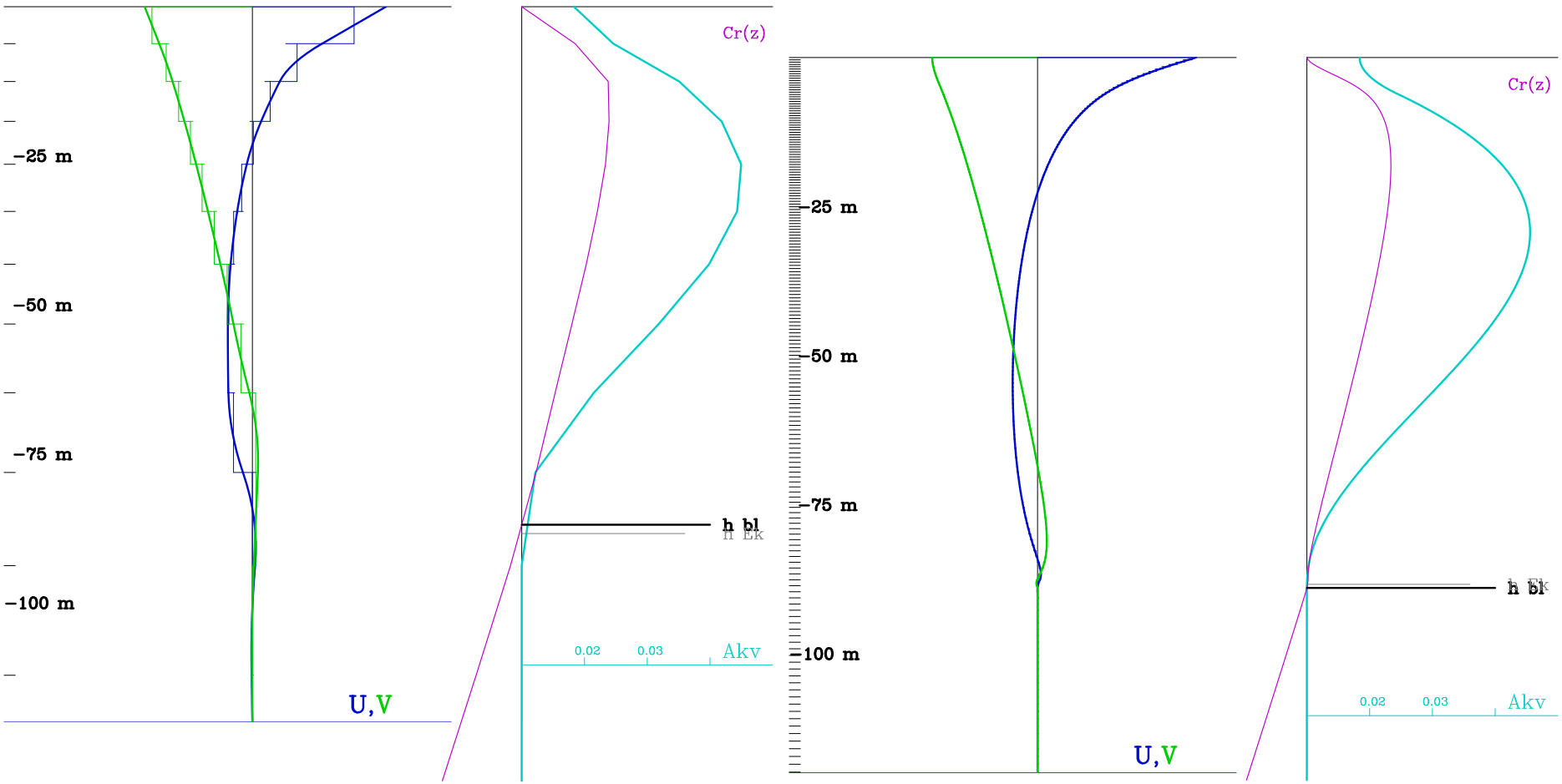
apply the same scaling

$$\widetilde{Cr}(\sigma) = \frac{1}{(0.7)^2} \int_\sigma^0 \mathscr{K}(\sigma) \left\{ \left| \frac{\partial \tilde{\mathbf{u}}}{\partial \sigma} \right|^2 - C_{\text{Ek}} \right\} d\sigma'$$

and demand that $\widetilde{Cr}(-1) = 0$
which can be achieved by tuning

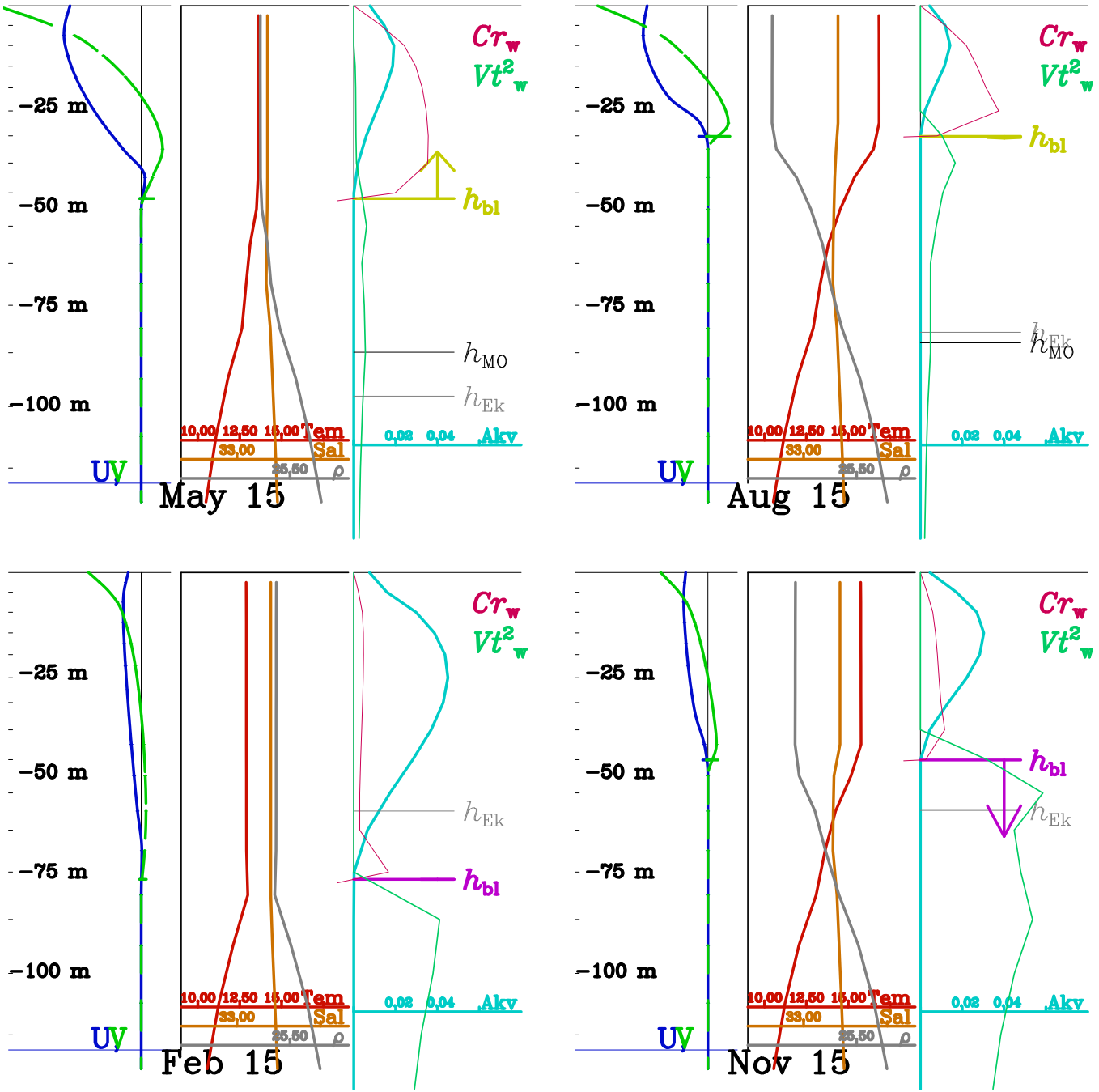
$C_{\text{Ek}} = 258$

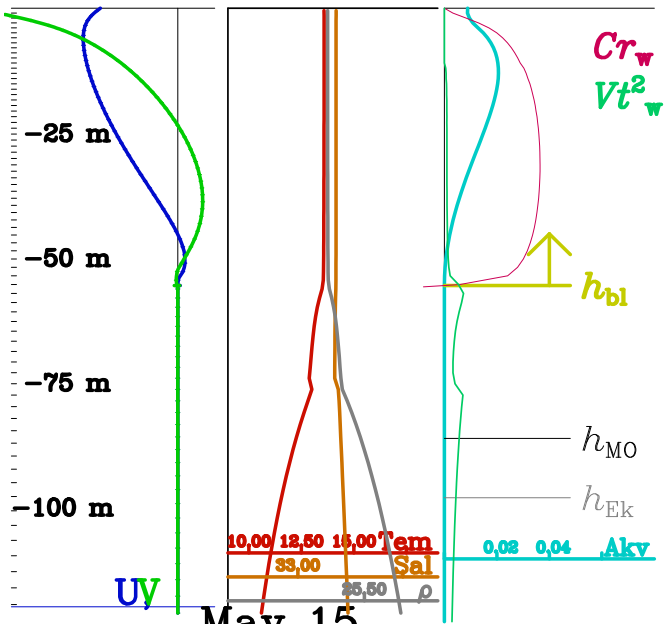
provided that $\mathscr{K}(\sigma) = |\sigma|/(|\sigma| + \epsilon)$
where $\epsilon = 0.1$



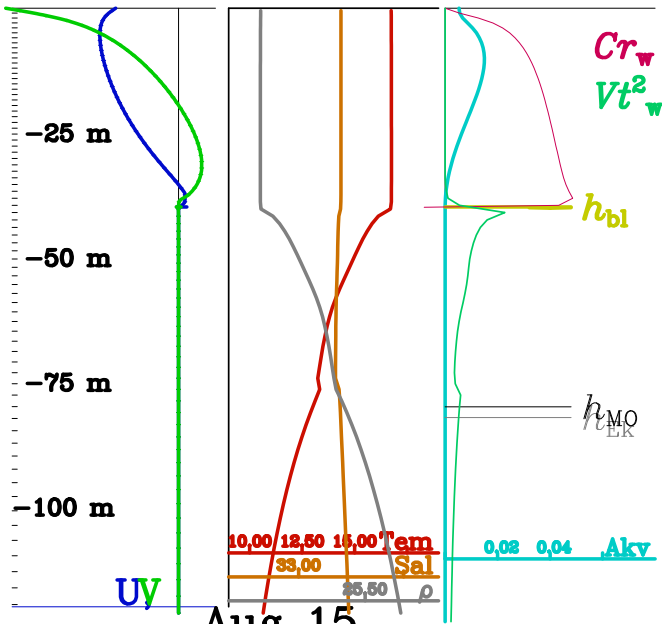
Coarse, $N = 32$ and fine, $N = 512$ resolution. h_{Ek} is shown for reference only and does not participate in determining h_{bl}.

- **presence of $\mathscr{K}(\sigma)$ is essential for convergence**
- **overall extremely robust**

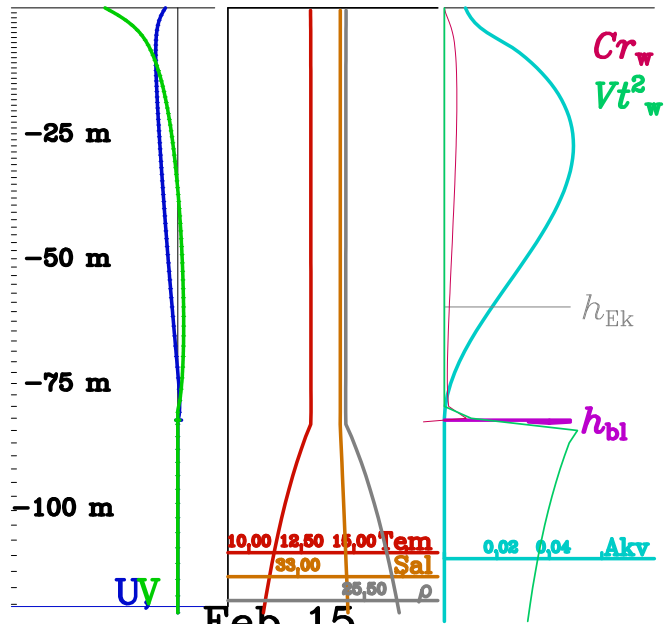

−25 m
−50 m
−75 m
−100 m
UV
10,00 12,50 15,00 Tem
33,00 Sal
25,50 ρ
0,02 0,04 Akv
Cr_w
Vt^2_w
h_{bl}
h_{MO}
h_{Ek}
May 15
Aug 15
Feb 15
Nov 15



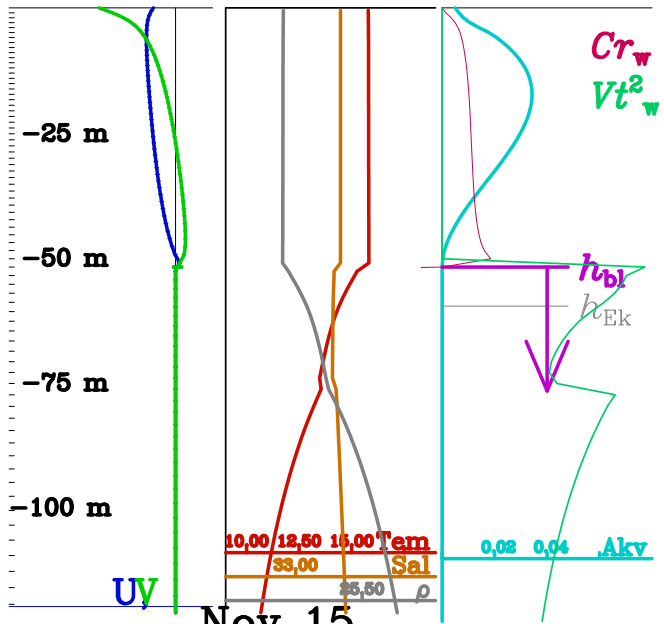
May 15



Aug 15



Feb 15



Nov 15

Poor Man's Computing: Overlooked and underutilized resources

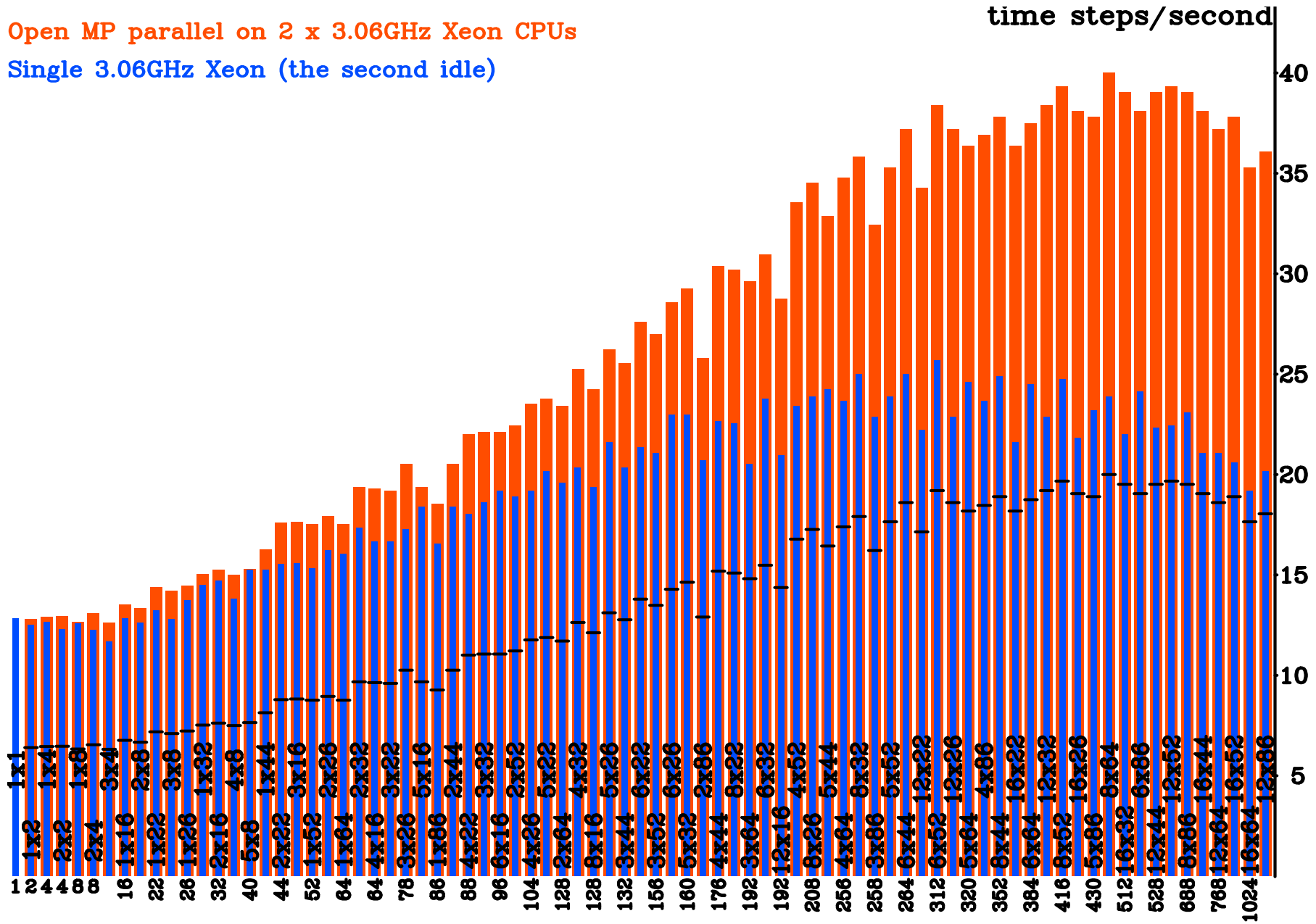
From its day 1 ROMS started as a coarse-grain multi-threaded (proprietary SGI directives, later OpenMP standard) parallel code with 2D subdomain decomposition (tiling) intended for SMP architecture.

- Loop-based parallelization approach **was rejected** from the start. Tiling with **ability to set number of tiles independently** from the number of CPUs.
- **False sharing avoidance** was the key for success/failure back then.

Departure SGI Origin 2000 *de-facto* downplayed the importance of SMP computing. MPI capability was added, but OpenMP always kept.

- SMPs are coming back in form of multi-core CPUs.
- ROMS in OpenMP mode optimally tiled for cache utilization **easily outperforms** the same ROMS code run as MPI by at least 2:1 ratio *within* a single SMP computer. Same applies to even a single-processor machine, tiling vs. single block.
- We knew this for over 10 years, others have no means to discover.
- Efficient Open MP played a key role (at least one of the) in successful proliferation.
- However, Open MP is limited to a single node only. Can we build a **dream code** which combines OpenMP-level single-processor performance with MPI scaling on multiple nodes?

Cache effects on SMP computer, back in 2004



two-dimensional Soliton problem on 768×256 grid

Poor Man's Computing: Why it is? Is it still relevant today?

This is why:

CPU family	Prescott Pentium IV HT	Kentsfield Core2 Quad	Nehalem Core i7 920	SandyBridge E Core i7 3820	IvyBridge E Core i7 4820K	Haswell E Core i7 5930K
year introduced	2004	2007	Nov. 2008	late 2011	2013	Aug. 2014
CPU clock speed	3.2GHz	2.4GHz	2.66GHz	3.2GHz	3.7GHz	3.5GHz
cores (threads)	1(2)	4	4(8)	4(8)	4(8)	6(12)
cache	1M L2	8M (4+4)	8M L3	10M	10M	15M
memory type	DDR 400	DDR2 667	DDR3 1333	DDR3 1600	DDR3 1600	DDR4 2400
CAS latency	2	5	8	9	9	15
latency, ns	10	15	12	11.2	11.2	12.5
number of channels	2	2	3	4	4	4
aggregate bandwidth, GB/sec	6.4	10.6	31.8	51.2	51.2	76.8
$\frac{\text{loads or stores}}{\text{per cycle, each core}}$	1	1	1	2	2	2
$\frac{\text{loads or stores}}{\text{per sec, all cores}}$, G/sec	3.2	9.6	10.6	25.6	29.6	42
$\frac{\text{load-store bandwidth}}{\text{for 8Byte operands}}$, GB/sec	25.6	76.8	84.8	204.8	236.8	336
$\frac{\text{load-store}}{\text{vs. memory}}$ bandwidth ratio	**4**	**7.25**	**2.666**	**4**	**4.625**	**4.375**

Poor Man's continued ...

Most today's Linux clusters are made of 8 ... 64-way SMP nodes

- dual or quad CPU sockets; 6, 8, ..., 16-core CPUs
- shared outermost-level cache in each CPU chip
- shared memory bus by all cores within each CPU socket
- interconnect interface (Infiniband or whatever) is shared by all CPUs/cores on the board and become scarce resource
- hyper-threading is back for Intel chips

Most MPI codes are designed to

- separate communications from computations in time resulting in peak loads on interconnect followed periods of by inactivity
- treat multiple processors (cores) within each physical node as separate compute nodes, ignoring non-uniform topology of the machine, (if any)
- introduce competition for shared interconnect interface
- single subdomain — single processor policy motivated by "perimeter vs. area argument"
- ignore cache effects

MPI with Threads Inside and MPI without them

Now, at last, multiple threads are **officially included** into MPI-2 standard

Replaces `MPI_Init` with `MPI_Init_thread(requested, provided)`, where

$$\texttt{requested}, \texttt{provided} = \begin{cases} 0 = \texttt{MPI\_THREAD\_SINGLE} \text{ means no thread support} \\ 1 = \texttt{MPI\_THREAD\_FUNNELED} \text{ means threads are allowed,} \\ \quad \text{but only master thread can execute MPI calls} \\ 2 = \texttt{MPI\_THREAD\_SERIALIZED} \text{ multiple threads can do} \\ \quad \text{MPI calls, but the calls are serialized} \\ 3 = \texttt{MPI\_THREAD\_MULTIPLE} \text{ means multiple threads can} \\ \quad \text{execute concurent MPI calls} \end{cases}$$

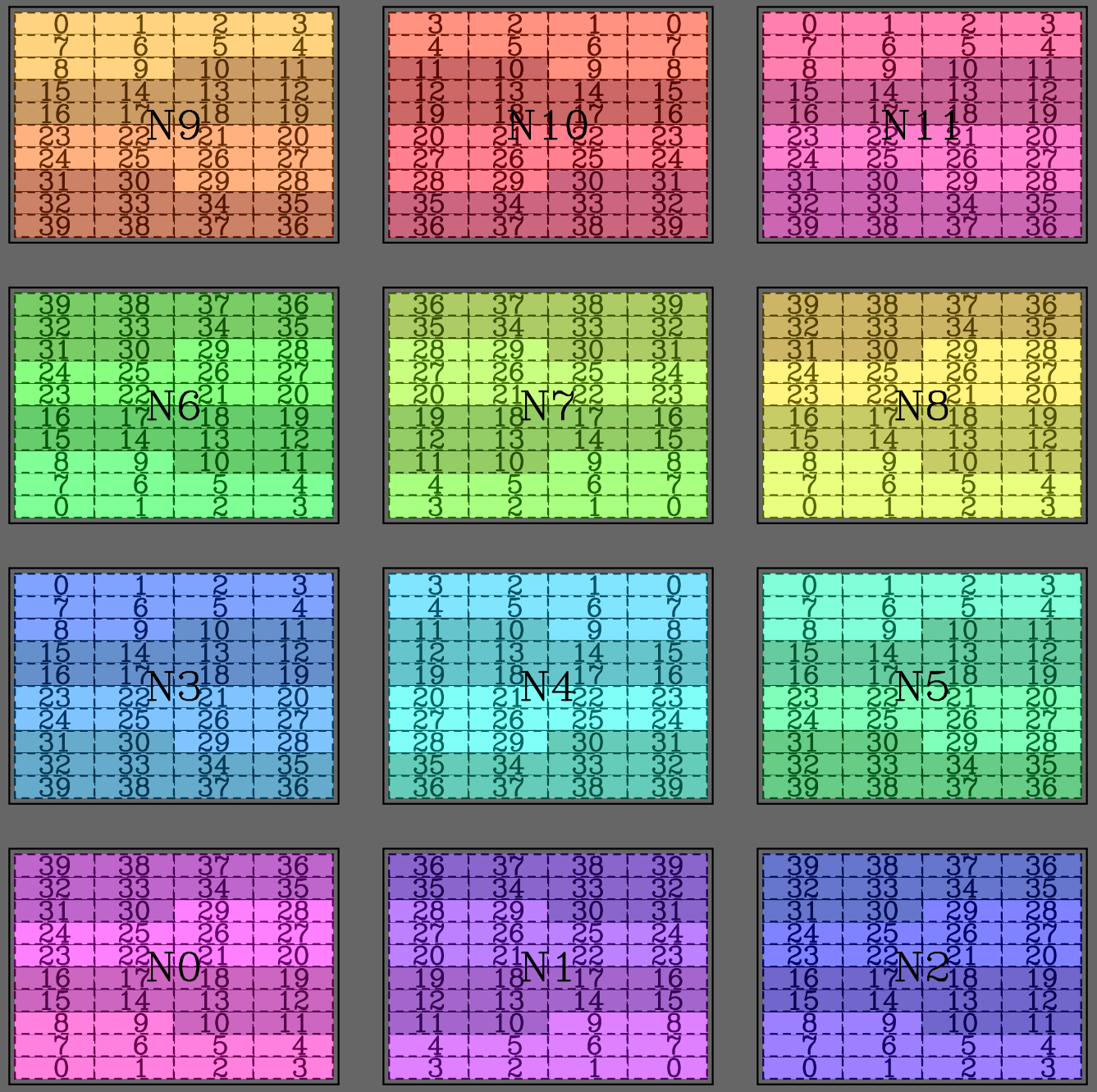
- The original motivation is to allow multiple subdomains for cache management to recover cache efficiency of optimally tiled code.
- But it turns out that there is more in it: tiling and threads can be used for tuning of scheduling of messages to alleviate competition for access to Network Interface within SMP nodes, simply put, when one thread sends messages, the other(s) compute and, and vice versa.

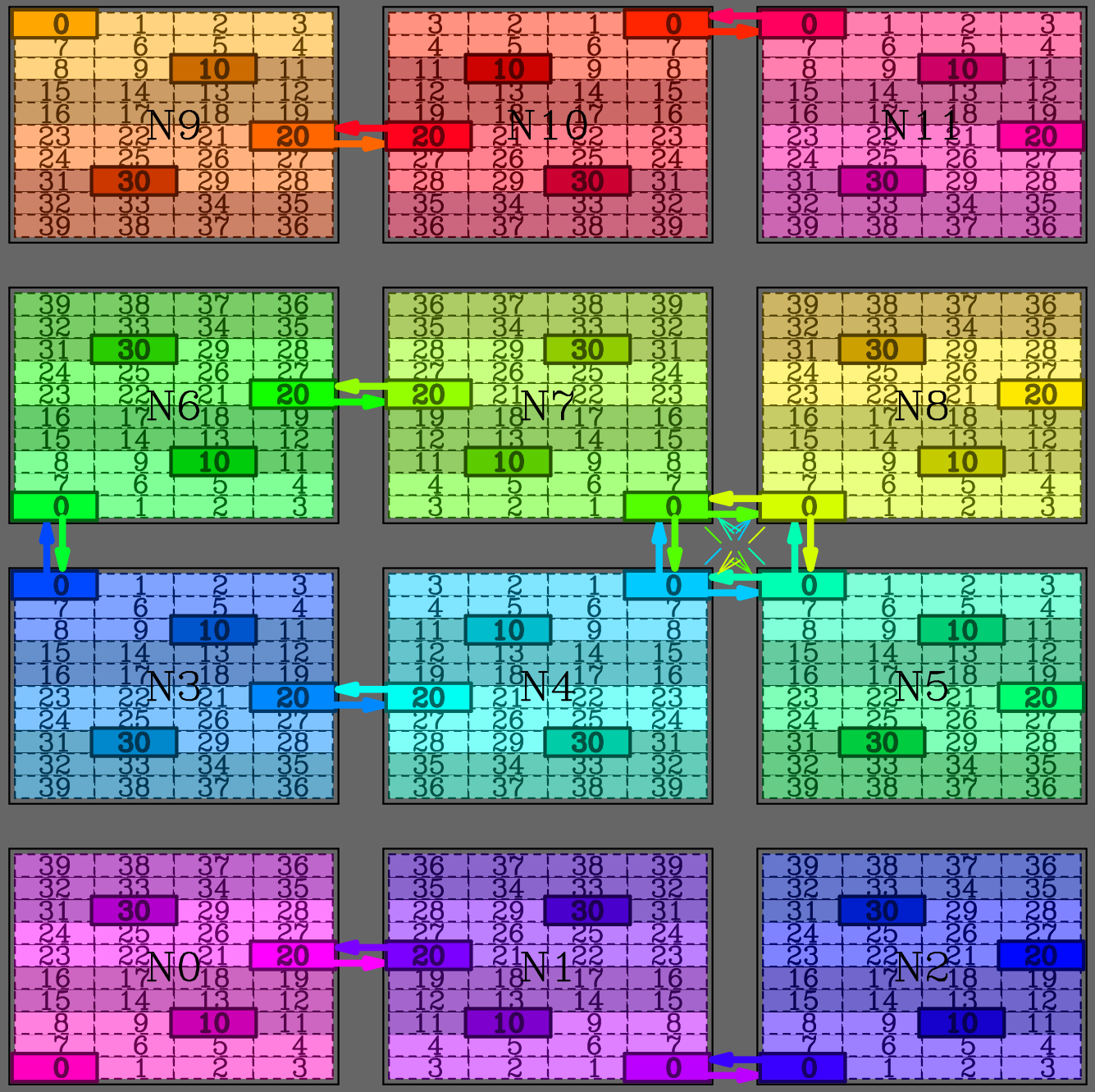
Approach:

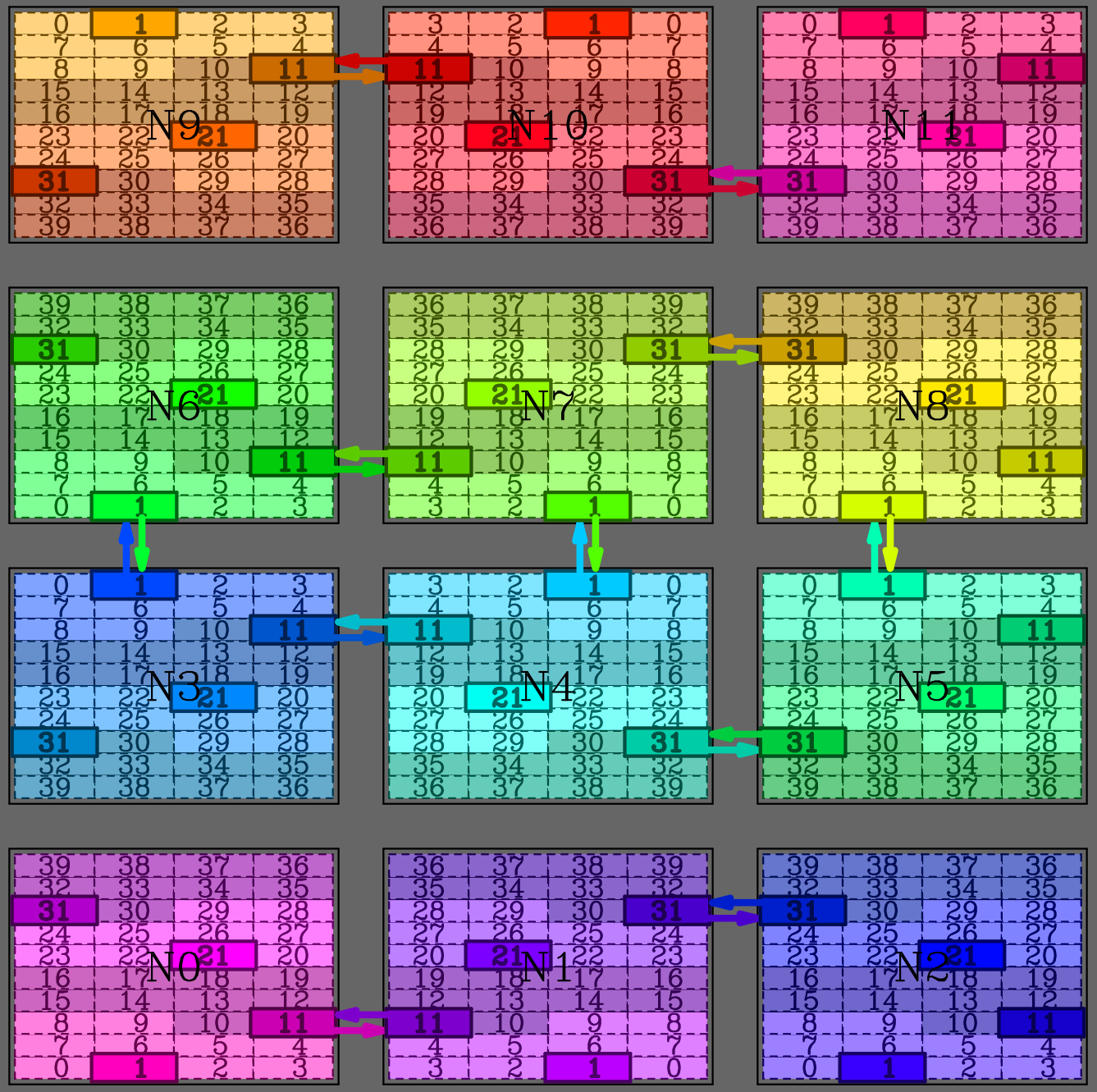
- relies on the highest level of MPI thread support, `requested, provided` = 3,3
- 2-level 2-dimensional subdomain decomposition: tiles within MPI subdomains;
- once a thread completes working on a tile, it sends/receives relevant messages to its MPI-neighbor (if any). Because now there is (may be) more than one neighbor of each side, and because MPI *knows nothing* about threads (i.e., thread receiving an MPI message from MPI neighbor does not know from which thread on that node the message is coming) use **unique tags** to label messages sent when processing different tiles
- **mirror** thread trajectories for adjacent MPI subdomains: that is the key to avoid deadlocks
- inner tiling unavoidably fragments MPI messages into smaller ones. This is off-set by combining messages exchanging halos for different arrays into unified messages
- prepare to receive all $\rightarrow$ send all $\rightarrow$ check and unpack whatever arrives (no pre-determined order)

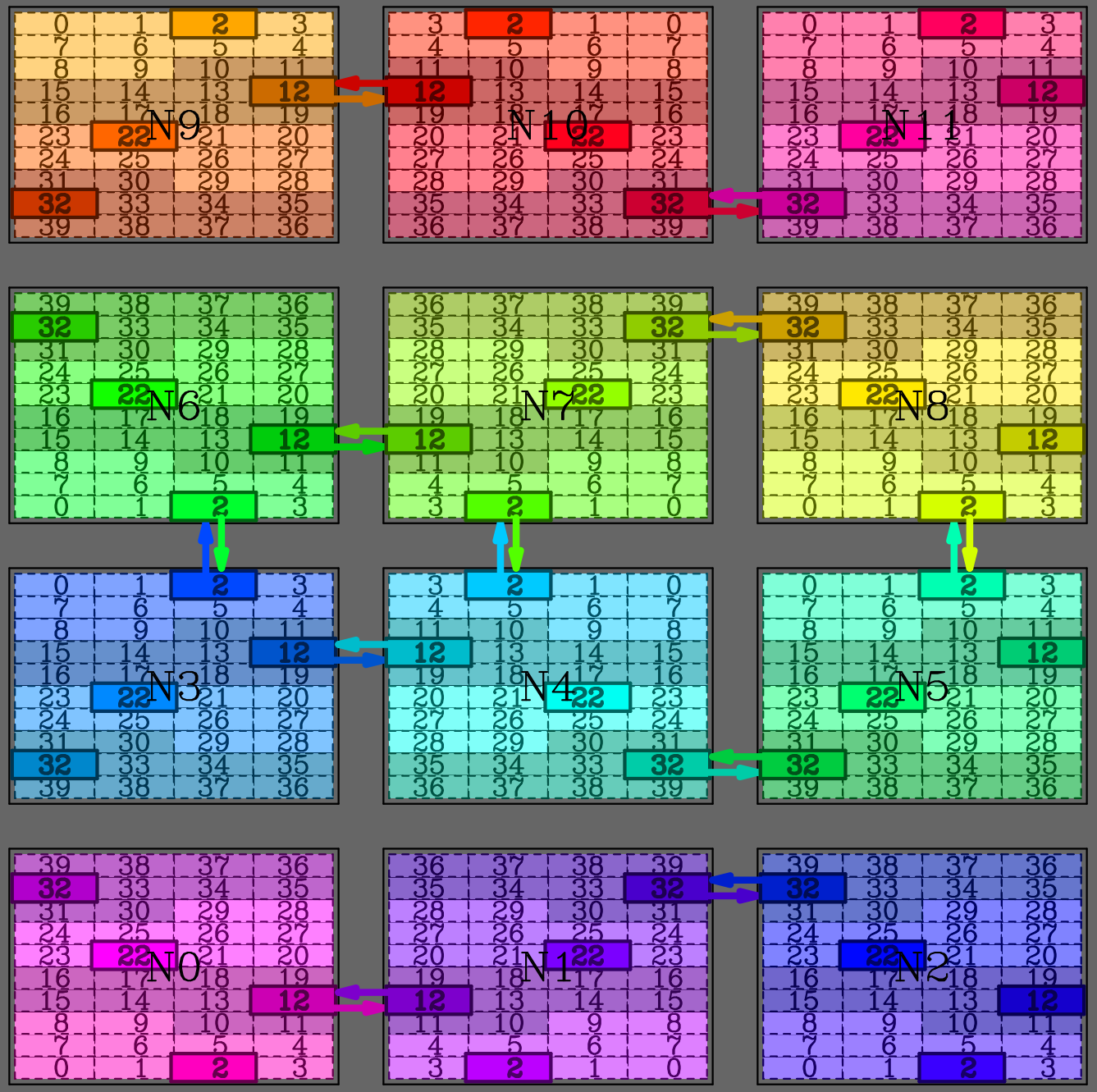
And, finally, the code is just a tool; **the art is in its usage:** the number of possible permutations is now too large to be quickly explored.

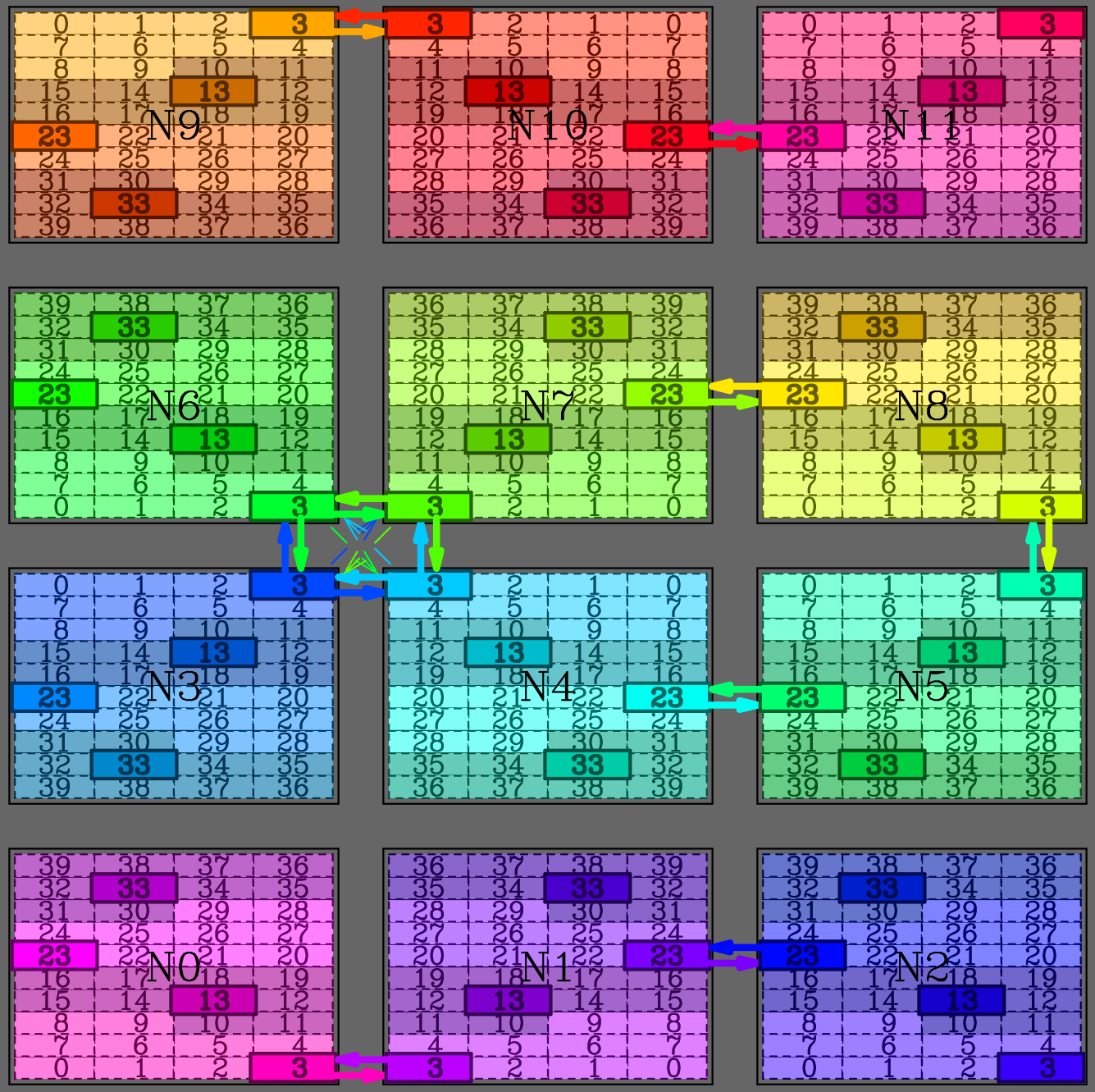
Scheduling messages by inner tiling

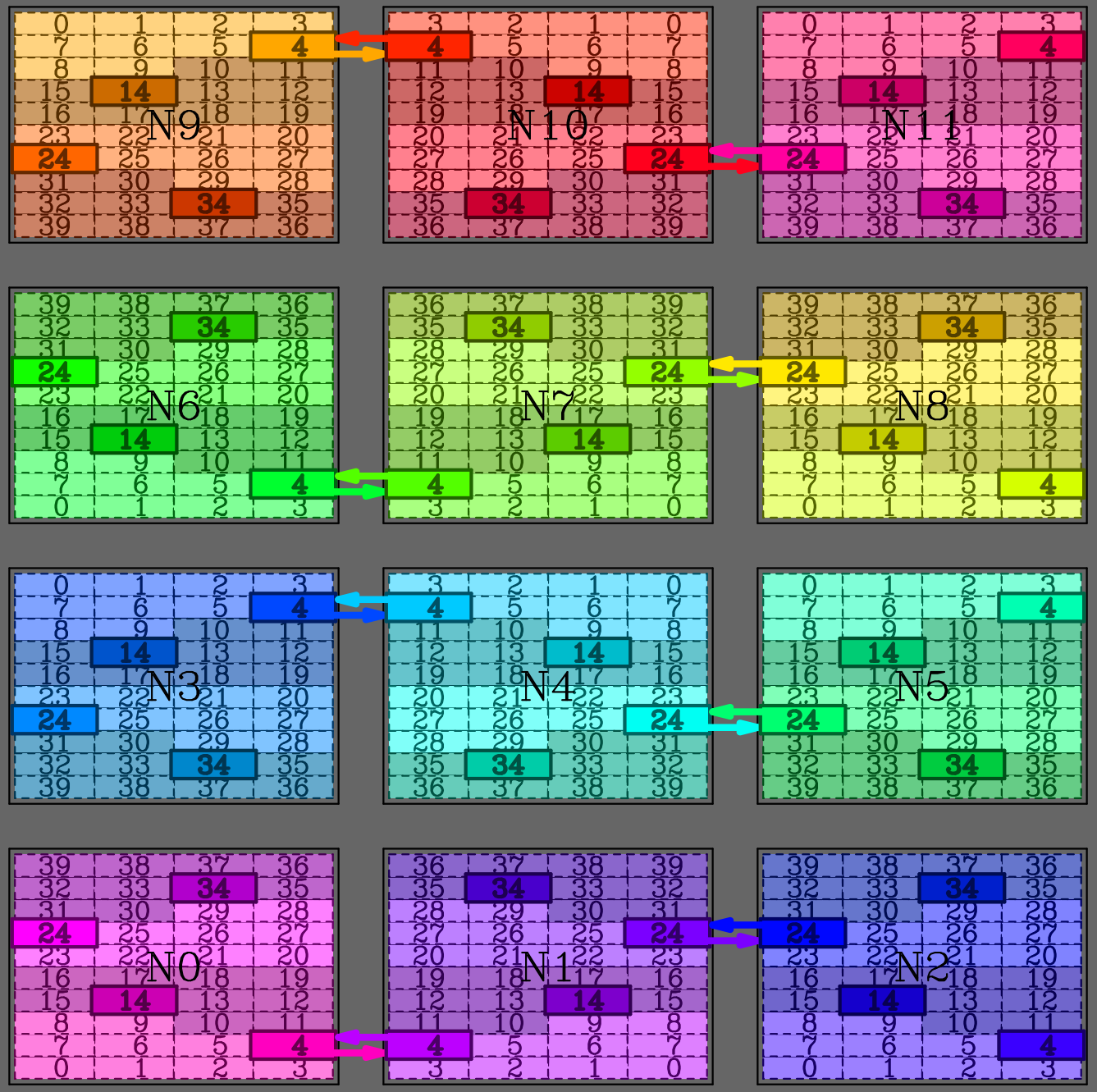

0 1 2 3
7 6 5 4
8 9 10 11
15 14 13 12
16 17 18 19
23 22 21 20
24 25 26 27
31 30 29 28
32 33 34 35
39 38 37 36
N9
3 2 1 0
4 5 6 7
11 10 9 8
12 13 14 15
19 18 17 16
20 21 22 23
27 26 25 24
28 29 30 31
35 34 33 32
36 37 38 39
N10
0 1 2 3
7 6 5 4
8 9 10 11
15 14 13 12
16 17 18 19
23 22 21 20
24 25 26 27
31 30 29 28
32 33 34 35
39 38 37 36
N11
39 38 37 36
32 33 34 35
31 30 29 28
24 25 26 27
23 22 21 20
16 17 18 19
15 14 13 12
8 9 10 11
7 6 5 4
0 1 2 3
N6
36 37 38 39
35 34 33 32
28 29 30 31
27 26 25 24
20 21 22 23
19 18 17 16
12 13 14 15
11 10 9 8
4 5 6 7
3 2 1 0
N7
39 38 37 36
32 33 34 35
31 30 29 28
24 25 26 27
23 22 21 20
16 17 18 19
15 14 13 12
8 9 10 11
7 6 5 4
0 1 2 3
N8
0 1 2 3
7 6 5 4
8 9 10 11
15 14 13 12
16 17 18 19
23 22 21 20
24 25 26 27
31 30 29 28
32 33 34 35
39 38 37 36
N3
3 2 1 0
4 5 6 7
11 10 9 8
12 13 14 15
19 18 17 16
20 21 22 23
27 26 25 24
28 29 30 31
35 34 33 32
36 37 38 39
N4
0 1 2 3
7 6 5 4
8 9 10 11
15 14 13 12
16 17 18 19
23 22 21 20
24 25 26 27
31 30 29 28
32 33 34 35
39 38 37 36
N5
39 38 37 36
32 33 34 35
31 30 29 28
24 25 26 27
23 22 21 20
16 17 18 19
15 14 13 12
8 9 10 11
7 6 5 4
0 1 2 3
N0
36 37 38 39
35 34 33 32
28 29 30 31
27 26 25 24
20 21 22 23
19 18 17 16
12 13 14 15
11 10 9 8
4 5 6 7
3 2 1 0
N1
39 38 37 36
32 33 34 35
31 30 29 28
24 25 26 27
23 22 21 20
16 17 18 19
15 14 13 12
8 9 10 11
7 6 5 4
0 1 2 3
N2

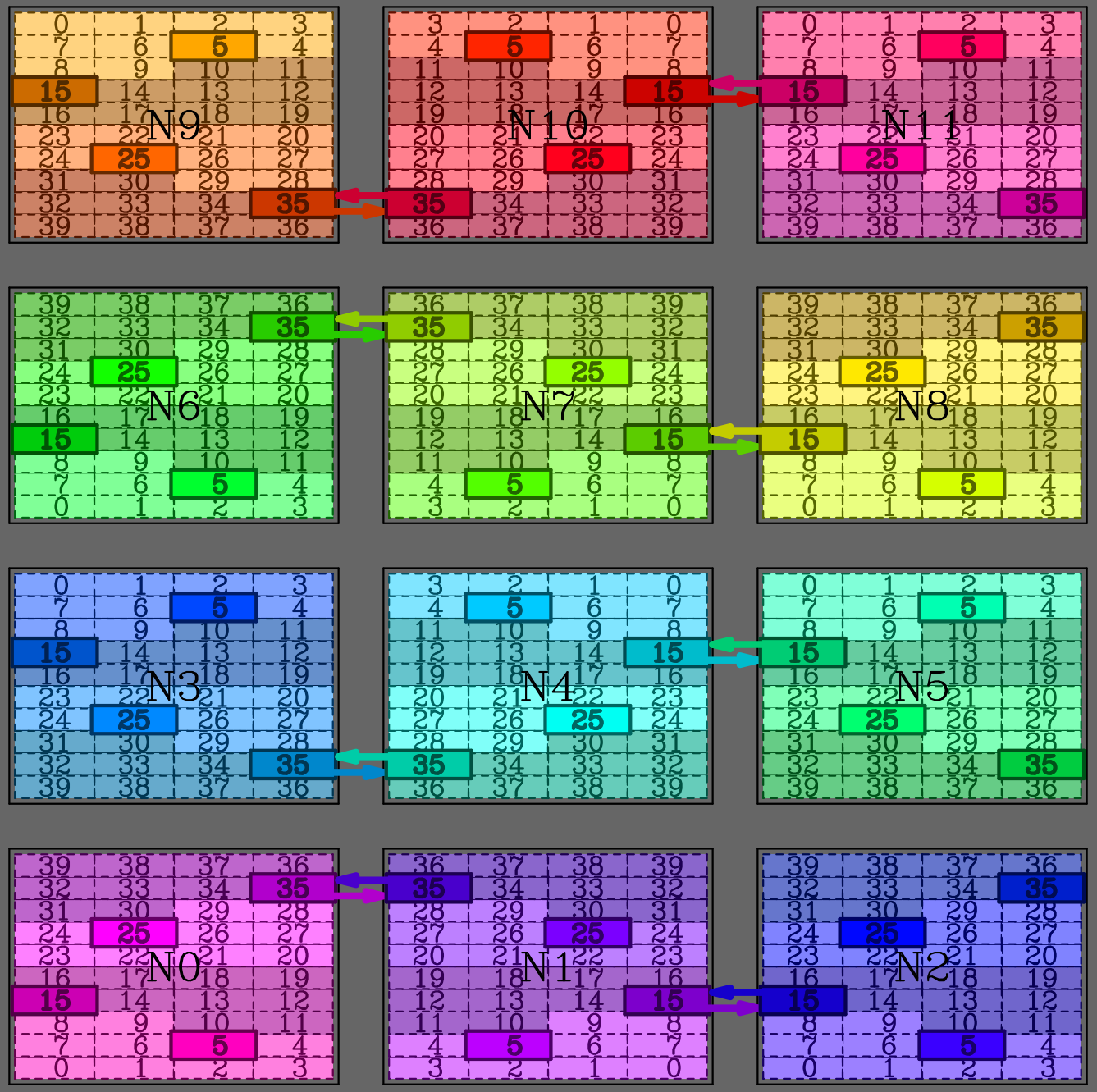

N9
N10
N11
N6
N7
N8
N3
N4
N5
N0
N1
N2

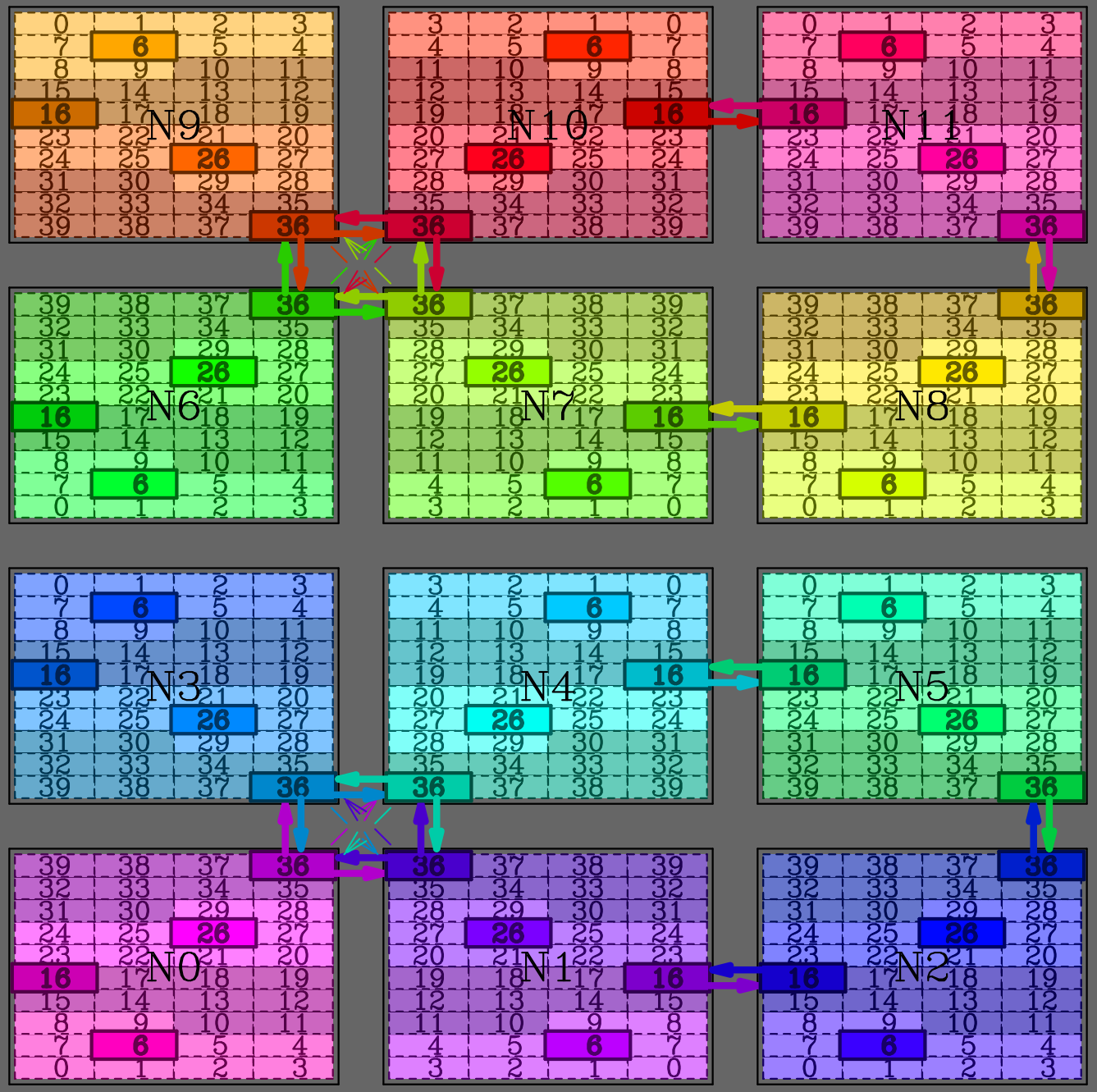

N9
N10
N11
N6
N7
N8
N3
N4
N5
N0
N1
N2

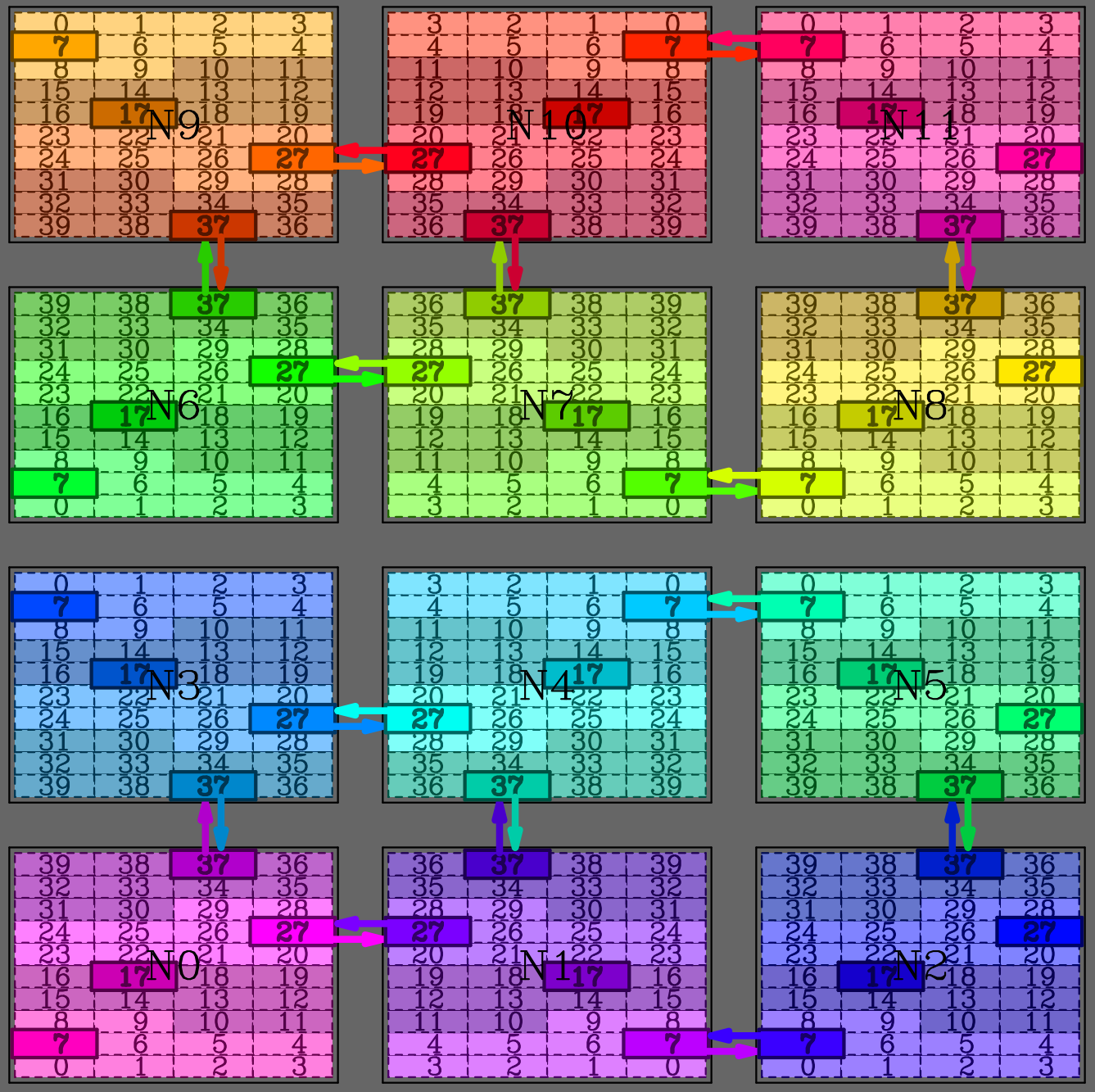

N9
N10
N11
N6
N7
N8
N3
N4
N5
N0
N1
N2

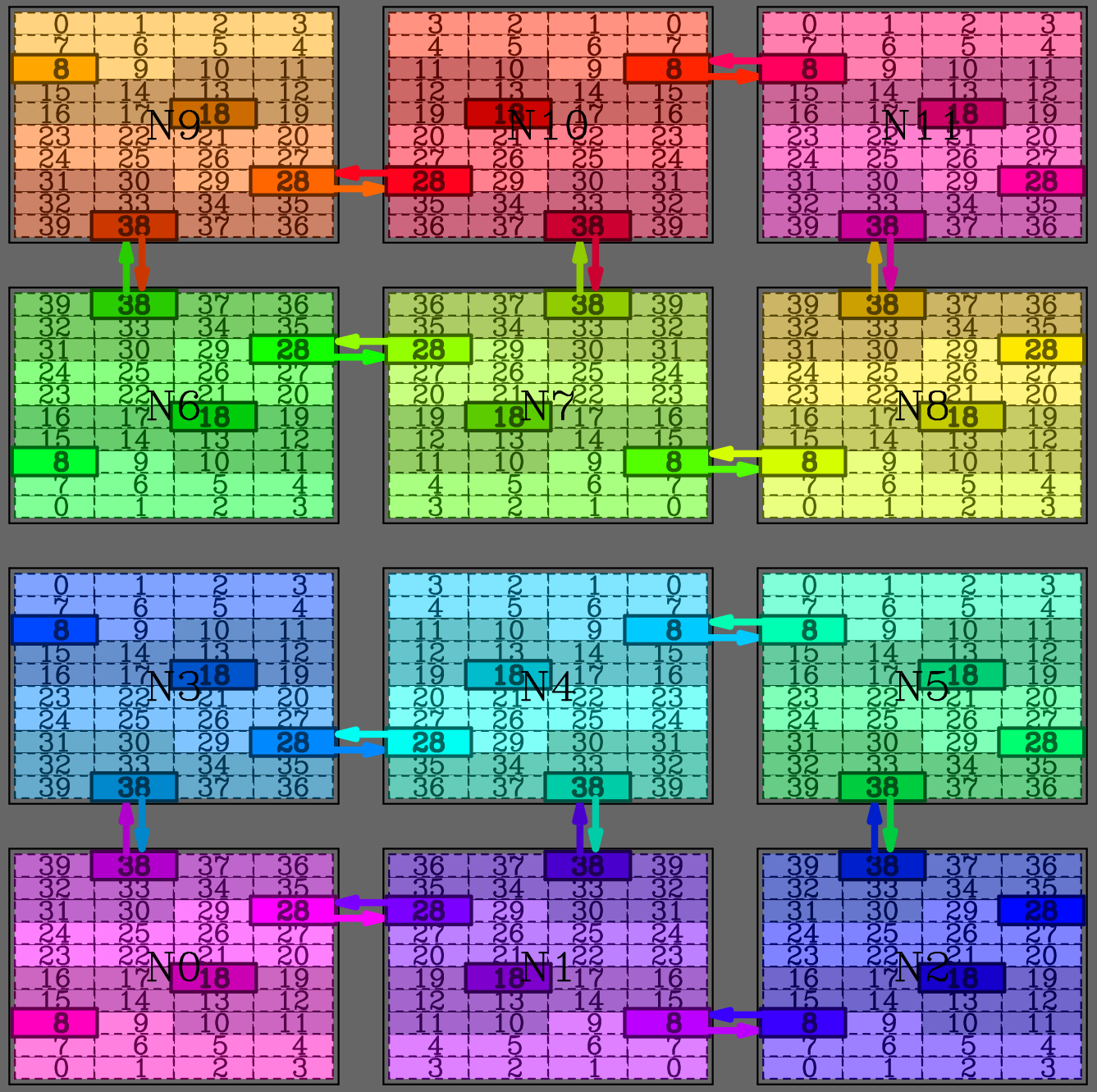

N9
N10
N11
N6
N7
N8
N3
N4
N5
N0
N1
N2

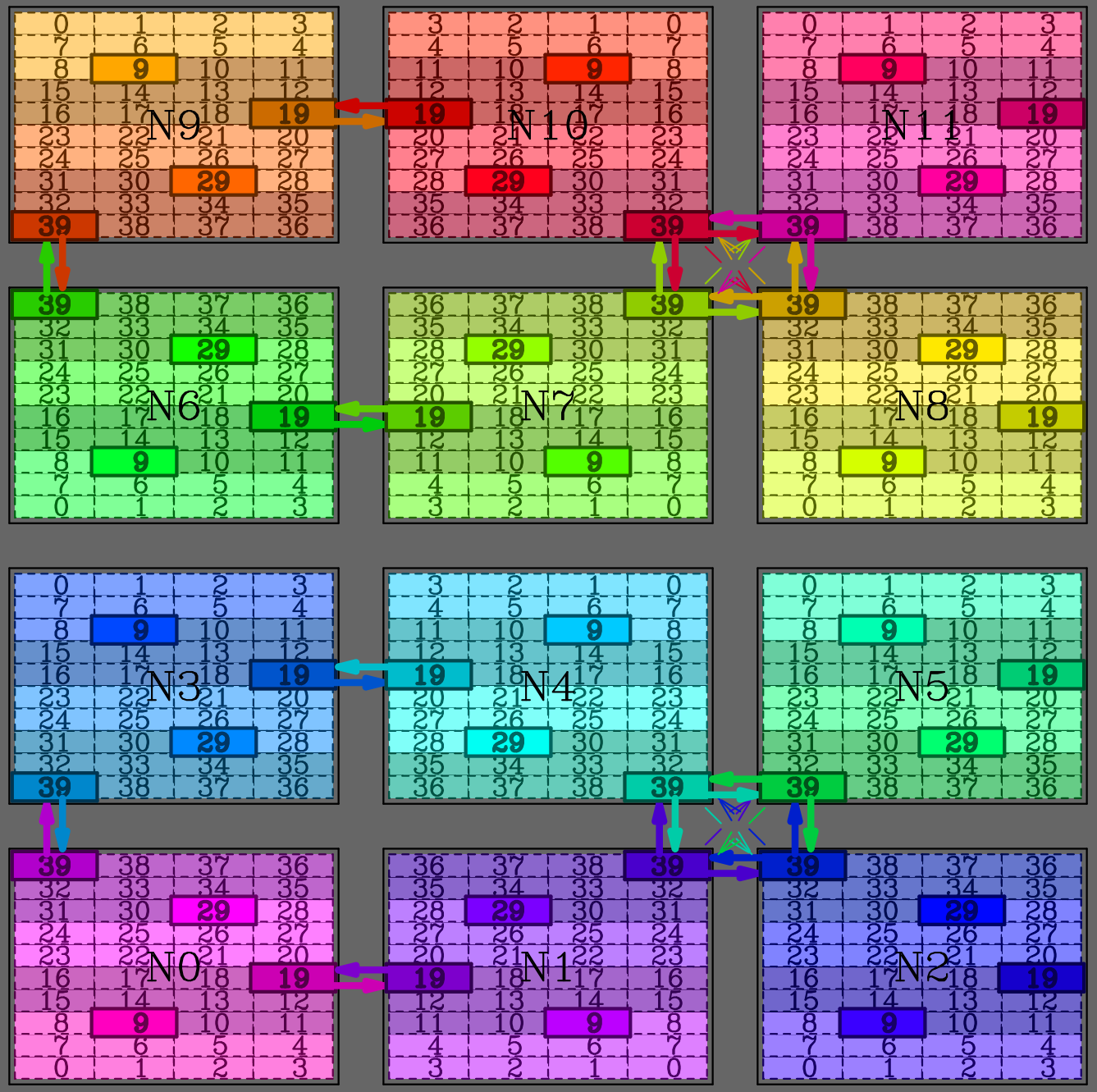

N9
N10
N11
N6
N7
N8
N3
N4
N5
N0
N1
N2


N9
N10
N11
N6
N7
N8
N3
N4
N5
N0
N1
N2


N9
N10
N11
N6
N7
N8
N3
N4
N5
N0
N1
N2


N9
N10
N11
N6
N7
N8
N3
N4
N5
N0
N1
N2


N9
N10
N11
N6
N7
N8
N3
N4
N5
N0
N1
N2


N9
N10
N11
N6
N7
N8
N3
N4
N5
N0
N1
N2

Poor Man's Conclusion

- besides cache optimization, inner tiling can be used to control scheduling of communications to alleviate competition between/among CPUs for network access
- works in "soft mode": threads are free-running, and there are no barriers around MPI calls, and no extra barriers relatively to the original OpenMP-only code.
- **thread trajectory does matter** by itself, and relatively to other threads, and it is no longer a simple zig-zag pattern
- partially overlaps computations and communications in time
- softens peak loads on network switch
- finer tiling also means smaller MPI-messages, which negatively affects performance. compromise is needed, and experience tells that message scheduling somewhat is more critical than cache optimization

Does it work?

- Yes, the code is functional and produces the correct result.
- Yes, it may exceed performance of pure MPI code sticking with one-subdomain-per processor(core) policy for problems of our interest.
- Yes, it is useful even if one ends up not using Open MP at all – it is still worth do tiling: **sparse mode**.
- Looking back: never trust you intuition about ideas for improving performance of MPI code: always try all the possibilities and select what works the best.
- Swappable MPI halo-exchange routines.