Orthogonal curvilinear grid generation

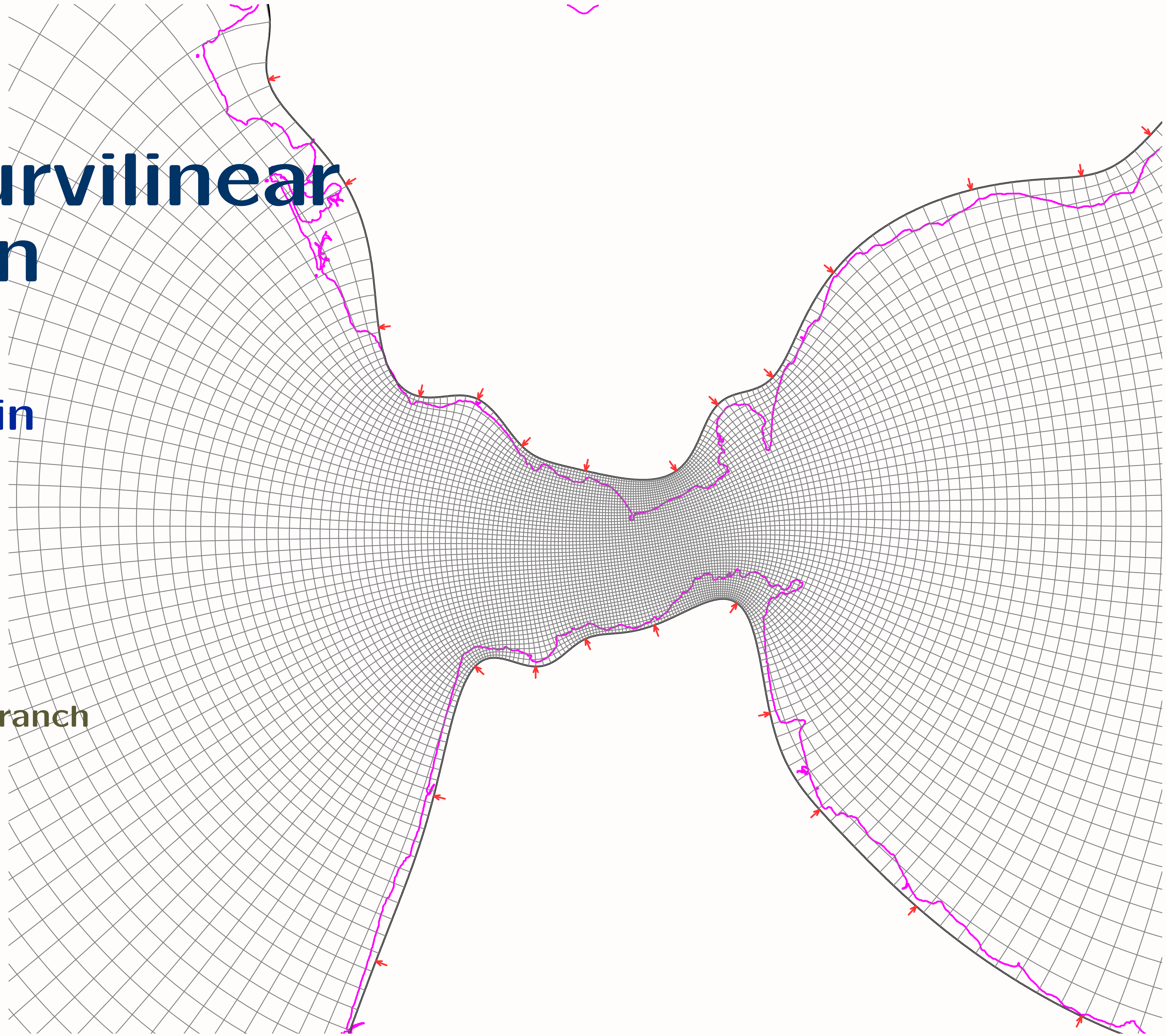
Alexander F. Shchepetkin

Shirshov Institute of Oceanology

NOAA Coastal Marine Modeling Branch

CMMB seminar, 25 May 2021

https://coastaloceanmodels.noaa.gov/seminar/



Officially the problem is solved 30 years ago:

Ives, D. C. and R. M. Zacharias, Conformal mapping and orthogonal grid generation. *Paper No. 87-2057, AIAA/SAE/ASME/ASEE 23RD Joint Propulsion Conference*, San Diego, CA, June 1987.
——— *J. Propulsion and Power*, 1989, Vol. 5, No. 3, pp. 327–333. `DOI:10.2514/3.23156 (AIAA-87-2057)`.

POM suite at Old Dominion University `http://www.ccpo.odu.edu/POMWEB/GRID-DATA/GRID.f`

Wilkin, J. L., 1987: A computer program for generating two-dimensional orthogonal curvilinear coordinate grids. *Unpublished report*, Woods Hole Oceanographic Institution, Woods Hole, MA 02543

Wilkin, J. and K. S. Hedström, User's Manual for an orthogonal curvilinear grid generation package. *IMCS, Rutgers University*, 1998, `https://marine.rutgers.edu/po/tools/gridpak/grid_manual.ps.gz`

`Gridpak` maintained by K. S. Hedström. `https://github.com/kshedstrom/gridpak`

`SeaGrid`, by Rich Signell, ?? `https://github.com/sea-mat/seagrid`

`Gridgen` by Pavel Sakov, `https://github.com/sakov/gridgen-c`

`Pygridgen` (Python interface to gridgen) by Rich Signell, Robert Hetland, ??
`https://github.com/pygridgen/pygridgen`

`GridBuilder`, by Charles James (PIRSA-SARDI) `https://austides.com/downloads`

`MIKE21C` (formerly `MIKE3D`) Curvilinear Grid Generator
`https://manuals.mikepoweredbydhi.help/2017/Water_Resources/MIKE21C_Scientific_documentation.pdf`

Unofficially, when newcomers to ROMS community ask about grid generation tool, we do not have much to offer.

IZOGRID[†]: A new tool for setting up orthogonal curvilinear grids for oceanic modeling

The dilemma: Virtually all modern structured-grid ocean modeling codes are written in orthogonal curvilinear coordinates in horizontal directions, yet the overwhelming majority of modeling studies are done using very simple grid setups - mostly rectangular patches of Mercator grids rotated to proper orientation. Furthermore, **in communities like ROMS, we even observe decline in both interest and skill of setting up curvilinear grids** over the long term. This is caused primarily by the dissatisfaction with the existing tools and procedures for grid generation due to inability to achieve acceptable level of orthogonality errors. This means underutilization of the full potential of the modeling codes.

To address these issues, a new algorithm for constructing orthogonal curvilinear grids on a sphere for a fairly general geometric shape of the modeling region is implemented as a *compile-once–use forever* software package.

Theoretically, one can use Schwartz-Christoffel conformal mapping to project a curvilinear contour onto rectangle, then draw Cartesian grid in it, and, finally, apply the inverse transform (the one which maps the rectangle back to the original contour) to the Cartesian grid in order to obtain the orthogonal curvilinear grid which fits the contour.

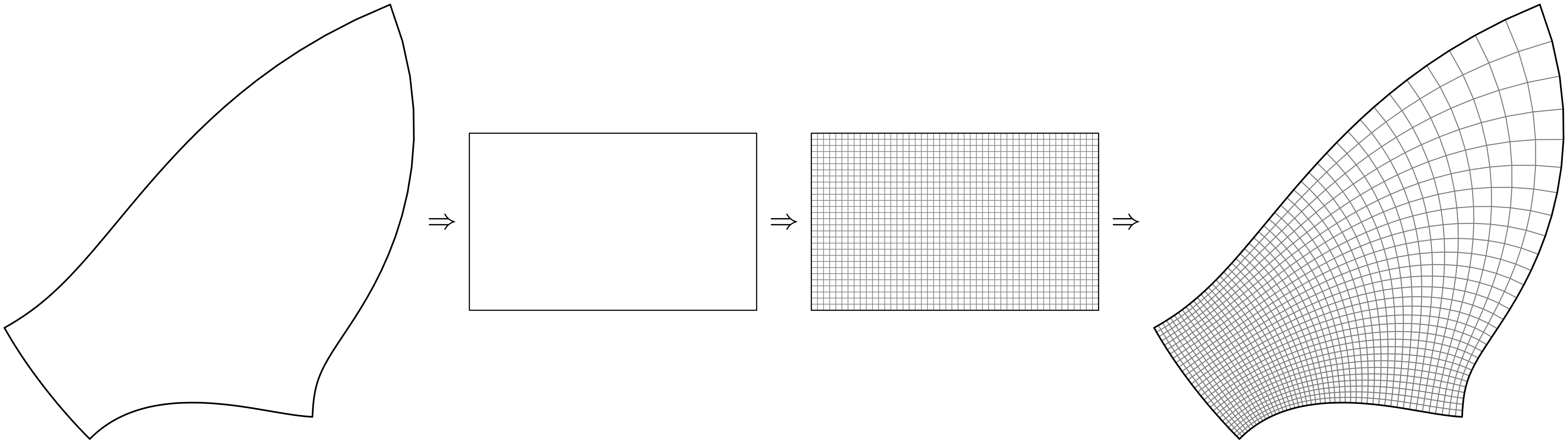
However, in the general case, the forward transform is an iterative algorithm of Ives and Zacharias (1989), and it is not easily invertible, nor it is feasible to apply it to a two-dimensional object (grid) as opposite to just one-dimensional (contour) because of very large number of operations.

To circumvent this, **the core of the new algorithm is essentially an iterative numerical solver for the inverse problem** – to find such distribution of grid points along the sides of curvilinear contour, that the direct conformal mapping of it onto rectangle turns this distribution into uniform one along each side of the rectangle. Along the way, this procedure also finds **the correct aspect ratio**, which makes it possible to automatically chose the number of grid points in one of the directions to yield locally the same grid spacing in both horizontal directions. The iterative procedure itself turns out to be multilevel - i.e. an iterative loop built around another, internal iterative loop. Thereafter, once the distribution of grid points along the perimeter becomes known, the interior of the grid is filled in by solving a Dirichlet problem.

[†]**IZOGRID = isotropic-resolution, Ives and Zacharias, orthogonal curvilinear grid generator**

Orthogonal Curvilinear Grid Generation

basic idea



$$(lat, lon) \rightarrow (x, y) \rightarrow (\xi, \eta) \rightarrow (x, y) \rightarrow (lat, lon)$$

$(lat, lon) \leftrightarrow (x, y)$ is **conformal** sphere-to-plane projection: Mercator, Lambert, stereographic, ...

Needs **reversible** conformal mapping $(x, y) \leftrightarrow (\xi, \eta)$ of plane-to-pane for **fairly general geometric shape.**

Conformal mapping basics

$$(x, y) \leftrightarrow (\xi, \eta)$$

introduce $z = x + iy$ and $\zeta = \xi + i\eta$, then z and ζ **must be related by an analytic (i.e. differentiable) function of complex variable**

$$z = z(\zeta) \qquad \text{and, conversely} \qquad \zeta = \zeta(z)$$

differentiable means existence of the limit

$$\frac{dz}{d\zeta} = \lim_{\Delta\zeta \to 0} \frac{z(\zeta + \Delta\zeta) - z(\zeta)}{\Delta\zeta} = \lim_{\Delta\zeta \to 0} \frac{x(\xi + \Delta\xi\,, \eta + \Delta\eta) + iy(\xi + \Delta\xi\,, \eta + \Delta\eta) - x(\xi\,, \eta) - iy(\xi\,, \eta)}{\Delta\xi + i\Delta\eta}$$

$$= \lim_{\Delta\zeta \to 0} \frac{x + \frac{\partial x}{\partial \xi}\Delta\xi + \frac{\partial x}{\partial \eta}\Delta\eta + iy + i\frac{\partial y}{\partial \xi}\Delta\xi + i\frac{\partial y}{\partial \eta}\Delta\eta - x - iy}{\Delta\xi + i\Delta\eta}$$

$$= \lim_{\Delta\zeta \to 0} \frac{\frac{\partial x}{\partial \xi}\Delta\xi + \frac{\partial x}{\partial \eta}\Delta\eta + i\frac{\partial y}{\partial \xi}\Delta\xi + i\frac{\partial y}{\partial \eta}\Delta\eta}{\Delta\xi + i\Delta\eta} = \lim_{\Delta\zeta \to 0} \frac{\left(\frac{\partial x}{\partial \xi} + i\frac{\partial y}{\partial \xi}\right)\Delta\xi + \left(-i\frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \eta}\right)i\Delta\eta}{\Delta\xi + i\Delta\eta}$$

the limit exists only if the two brackets $\left(.\right)$ **are the same**,

hence $\quad \dfrac{\partial x}{\partial \xi} = \dfrac{\partial y}{\partial \eta} \quad$ and $\quad \dfrac{\partial y}{\partial \xi} = -\dfrac{\partial x}{\partial \eta}$

$$\frac{dz}{d\zeta} = \frac{\partial x}{\partial \xi} + i\frac{\partial y}{\partial \xi} = -i\frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \eta}$$
$$= \frac{\partial x}{\partial \xi} - i\frac{\partial x}{\partial \eta} = \frac{\partial y}{\partial \eta} + i\frac{\partial y}{\partial \xi}$$

these are **Cauchy–Riemann conditions**

Such functions $x = x(\xi, \eta)$ and $y = y(\xi, \eta)$ are called holomorphic.

Conformal mapping basics: How mapping by an analytic (i.e. differentiable) function of complex-variable yields preservation of orthogonality?

Let $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2 \in \{\xi, \eta\}$-plane be pair of infinitely small vectors,

$$\left.\begin{aligned} \boldsymbol{\ell}_1 &= (\delta_1\xi, \delta_1\eta) \\ \boldsymbol{\ell}_2 &= (\delta_2\xi, \delta_2\eta) \end{aligned}\right\} \quad \text{such that} \quad \boldsymbol{\ell}_1 \perp \boldsymbol{\ell}_2\,, \quad \text{hence} \quad (\boldsymbol{\ell}_1 \cdot \boldsymbol{\ell}_2) = \delta_1\xi \cdot \delta_2\xi + \delta_1\eta \cdot \delta_2\eta = 0$$

$(\xi, \eta) \to (x, y)$ transforms them into

$$\left.\begin{aligned} \boldsymbol{\ell}_1 &\to \boldsymbol{n}_1 \\ \boldsymbol{\ell}_2 &\to \boldsymbol{n}_2 \end{aligned}\right\} \boldsymbol{n}_1, \boldsymbol{n}_2 \in \{x, y\}\text{-plane} \qquad \begin{aligned} \boldsymbol{n}_1 = (\delta_1 x, \delta_1 y) &= \left(\partial_\xi x\, \delta_1\xi + \partial_\eta x\, \delta_1\eta\,,\ \partial_\xi y\, \delta_1\xi + \partial_\eta y\, \delta_1\eta\right) \\ \boldsymbol{n}_2 = (\delta_2 x, \delta_2 y) &= \left(\partial_\xi x\, \delta_2\xi + \partial_\eta x\, \delta_2\eta\,,\ \partial_\xi y\, \delta_2\xi + \partial_\eta y\, \delta_2\eta\right) \end{aligned}$$

their scalar product

$$(\boldsymbol{n}_1 \cdot \boldsymbol{n}_2) = \left(\partial_\xi x\, \delta_1\xi + \partial_\eta x\, \delta_1\eta\right) \cdot \left(\partial_\xi x\, \delta_2\xi + \partial_\eta x\, \delta_2\eta\right) + \left(\partial_\xi y\, \delta_1\xi + \partial_\eta y\, \delta_1\eta\right) \cdot \left(\partial_\xi y\, \delta_2\xi + \partial_\eta y\, \delta_2\eta\right)$$

or

$$\begin{aligned} (\boldsymbol{n}_1 \cdot \boldsymbol{n}_2) = &\left(\partial_\xi x\, \partial_\xi x + \partial_\xi y\, \partial_\xi y\right) \cdot \delta_1\xi \cdot \delta_2\xi \quad + \quad \left(\partial_\xi x\, \partial_\eta x + \partial_\xi y\, \partial_\eta y\right) \cdot \delta_1\xi \cdot \delta_2\eta \\ &+ \left(\partial_\eta x\, \partial_\xi x + \partial_\eta y\, \partial_\xi y\right) \cdot \delta_1\eta \cdot \delta_2\xi \quad + \quad \left(\partial_\eta x\, \partial_\eta x + \partial_\eta y\, \partial_\eta y\right) \cdot \delta_1\eta \cdot \delta_2\eta \end{aligned}$$

using **Cauchy–Riemann conditions**, substitute $\partial_\eta y \to \partial_\xi x$ and $\partial_\xi y \to -\partial_\eta x$,

$$\begin{aligned} (\boldsymbol{n}_1 \cdot \boldsymbol{n}_2) = &\left(\partial_\xi x\, \partial_\xi x + \partial_\eta x\, \partial_\eta x\right) \cdot \delta_1\xi \cdot \delta_2\xi \quad + \quad \left(\partial_\xi x\, \partial_\eta x - \partial_\eta x\, \partial_\xi x\right) \cdot \delta_1\xi \cdot \delta_2\eta \\ &+ \left(\partial_\eta x\, \partial_\xi x - \partial_\xi x\, \partial_\eta x\right) \cdot \delta_1\eta \cdot \delta_2\xi \quad + \quad \left[\partial_\eta x\, \partial_\eta x + \partial_\xi x\, \partial_\xi x\right] \cdot \delta_1\eta \cdot \delta_2\eta \\ &= \quad \left((\partial_\xi x)^2 + (\partial_\eta x)^2\right) \cdot \left(\delta_1\xi \cdot \delta_2\xi + \delta_1\eta \cdot \delta_2\eta\right) = 0 \end{aligned}$$

Hence $\boldsymbol{\ell}_1 \perp \boldsymbol{\ell}_2 \;\Leftrightarrow\; \boldsymbol{n}_1 \perp \boldsymbol{n}_2$ as long as C.-R.s hold.

This applies to any orientation of vectors ℓ_1, ℓ_2, not necessarily along grid-coordinate lines.

Conformal mapping basics: preservation of aspect ratio

Let $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2 \in \{\xi, \eta\}$-plane be pair of infinitely small vectors,

$$\left.\begin{aligned} \boldsymbol{\ell}_1 &= (\delta_1\xi, \delta_1\eta) \\ \boldsymbol{\ell}_2 &= (\delta_2\xi, \delta_2\eta) \end{aligned}\right\} \quad \text{hence} \quad |\boldsymbol{\ell}_1| : |\boldsymbol{\ell}_2| = \sqrt{(\delta_1\xi)^2 + (\delta_1\eta)^2} : \sqrt{(\delta_2\xi)^2 + (\delta_2\eta)^2}$$

$(\xi, \eta) \rightarrow (x, y)$ transforms them into

$$\left.\begin{aligned} \boldsymbol{\ell}_1 &\rightarrow \boldsymbol{n}_1 \\ \boldsymbol{\ell}_2 &\rightarrow \boldsymbol{n}_2 \end{aligned}\right\} \boldsymbol{n}_1, \boldsymbol{n}_2 \in \{x, y\}\text{-plane} \qquad \begin{aligned} \boldsymbol{n}_1 = (\delta_1 x, \delta_1 y) &= \big(\partial_\xi x\, \delta_1\xi + \partial_\eta x\, \delta_1\eta\,,\ \partial_\xi y\, \delta_1\xi + \partial_\eta y\, \delta_1\eta\big) \\ \boldsymbol{n}_2 = (\delta_2 x, \delta_2 y) &= \big(\partial_\xi x\, \delta_2\xi + \partial_\eta x\, \delta_2\eta\,,\ \partial_\xi y\, \delta_2\xi + \partial_\eta y\, \delta_2\eta\big) \end{aligned}$$

what about $|\boldsymbol{n}_1| : |\boldsymbol{n}_2|$?

$$\begin{aligned} |\boldsymbol{n}_1|^2 &= \big(\partial_\xi x\, \delta_1\xi + \partial_\eta x\, \delta_1\eta\big)^2 + \big(\partial_\xi y\, \delta_1\xi + \partial_\eta y\, \delta_1\eta\big)^2 \\ &= \Big(\big(\partial_\xi x\big)^2 + \big(\partial_\xi y\big)^2\Big) \cdot (\delta_1\xi)^2 + 2 \cdot \big(\partial_\xi x \cdot \partial_\eta x + \partial_\xi y \cdot \partial_\eta y\big) \cdot \delta_1\xi \cdot \delta_1\eta + \Big((\partial_\eta x)^2 + (\partial_\eta y)^2\Big) \cdot (\delta_1\eta)^2 \end{aligned}$$

using **Cauchy–Riemann conditions**, substitute $\partial_\eta y \rightarrow \partial_\xi x$ and $\partial_\xi y \rightarrow -\partial_\eta x$,

$$\begin{aligned} |\boldsymbol{n}_1|^2 &= \Big(\big(\partial_\xi x\big)^2 + (\partial_\eta x)^2\Big) \cdot (\delta_1\xi)^2 + 2 \cdot \big(\partial_\xi x \cdot \partial_\eta x - \partial_\eta x \cdot \partial_\xi x\big) \cdot \delta_1\xi \cdot \delta_1\eta + \Big((\partial_\eta x)^2 + \big(\partial_\xi x\big)^2\Big) \cdot (\delta_1\eta)^2 \\ &= \Big(\big(\partial_\xi x\big)^2 + (\partial_\eta x)^2\Big) \cdot \Big((\delta_1\xi)^2 + (\delta_1\eta)^2\Big) \end{aligned}$$

Similarly $|\boldsymbol{n}_2|^2 = \Big(\big(\partial_\xi x\big)^2 + (\partial_\eta x)^2\Big) \cdot \Big((\delta_2\xi)^2 + (\delta_2\eta)^2\Big)$

Hence $|\boldsymbol{\ell}_1| : |\boldsymbol{\ell}_2| = |\boldsymbol{n}_1| : |\boldsymbol{n}_2|$ **as long as C.-R.s hold**.

This applies to any orientation of vectors ℓ_1, ℓ_2, not necessarily along grid-coordinate lines.

$$\begin{aligned} \big(\partial_\xi x\big)^2 + \big(\partial_\xi y\big)^2 &= \big(\partial_\xi x\big)^2 + (\partial_\eta x)^2 \\ &= (\partial_\eta y)^2 + (\partial_\eta x)^2 = (\partial_\eta y)^2 + \big(\partial_\xi y\big)^2 \end{aligned}$$

Compute $\begin{cases} 1/\mathrm{pm} = \Delta\xi \cdot \sqrt{\big(\partial_\xi x\big)^2 + \big(\partial_\xi y\big)^2} \\ 1/\mathrm{pn} = \Delta\eta \cdot \sqrt{(\partial_\eta x)^2 + (\partial_\eta y)^2} \end{cases}$

Conformal mapping basics...

Cauchy–Riemann conditions $\partial_\xi x = \partial_\eta y$ and $\partial_\xi y = -\partial_\eta x$ yield

$$\frac{\partial^2 x}{\partial\xi^2} + \frac{\partial^2 x}{\partial\eta^2} = \frac{\partial}{\partial\xi}\left(\frac{\partial x}{\partial\xi}\right) + \frac{\partial}{\partial\eta}\left(\frac{\partial x}{\partial\eta}\right) = \frac{\partial}{\partial\xi}\left(\frac{\partial y}{\partial\eta}\right) + \frac{\partial}{\partial\eta}\left(-\frac{\partial y}{\partial\xi}\right) = 0$$

similarly

$$\frac{\partial^2 y}{\partial\xi^2} + \frac{\partial^2 y}{\partial\eta^2} = 0$$

If transform $(x,y) \leftrightarrow (\xi,\eta)$ is known only for the boundary,

$$\partial\mathscr{D} \leftrightarrow \partial\mathscr{G}\,, \qquad (x,y) = \big(x(\xi,\eta)\,, y(\xi,\eta)\big), \qquad (x,y)\in\partial\mathscr{D}, \quad (\xi,\eta)\in\partial\mathscr{G}$$

then *inside the domain* $(x,y) = \big(x(\xi,\eta)\,, y(\xi,\eta)\big) \in \mathscr{D}\setminus\partial\mathscr{D}$ may be computed by solving a pair of Dirichlet problems

$$\begin{aligned} &\partial^2_{\xi\xi}x + \partial^2_{\eta\eta}x = 0 \qquad \text{subject to b.c.} \qquad && x = x(\xi,\eta), \qquad (\xi,\eta)\in\partial\mathscr{G} \\ &\partial^2_{\xi\xi}y + \partial^2_{\eta\eta}y = 0 && y = y(\xi,\eta), \qquad (\xi,\eta)\in\partial\mathscr{G} \end{aligned}$$

Therefore, for any domain $\mathscr{D}\in\{x,y\}$, which is to be conformally mapped onto domain $\mathscr{G}\in\{\xi,\eta\}$, there is one-to-one correspondence between mapping just the outer boundary, and mapping boundary+interior.

When constructing transform $(x,y)\leftrightarrow(\xi,\eta)$, **it is sufficient to do it only for the boundary**, $\partial\mathscr{D}\leftrightarrow\partial\mathscr{G}$
$\Rightarrow$ huge computational savings, $\mathscr{O}\big(2(L+M)\big)$ vs. $\mathscr{O}\big(L\cdot M\big)$ operations

Assuming that in the $\{\xi,\eta\}$-plane the placement of points $\{\xi_{i,j},\eta_{i,j}\}$ corresponds to uniform equally-spaced Cartesian grid, the resultant placement of all grid points $\{x_{i,j},y_{i,j}\}$ of curvilinear grid, both its perimeter and interior, **is entirely and uniquely defined by the shape of its its curvilinear perimeter.**

The outline for orthogonal curvilinear grid generation is therefore as follows:

Construct curvilinear perimeter – specify a set of coefficients for spline polynomials in such a way that locations of the points on the contour line $(x, y) = \big(x(s), y(s)\big)$ **may be computed for any value of** s, which is the coordinate along the curve, **yet no set of discrete values** $\{s_k\}$ **corresponding to the actual grid points, belonging to the edge of the grid, is specified**. This means that at this stage points can be moved freely along the curve. No conformal mapping is needed at this point.

Populate the curvilinear contour with grid points in such a way, that when applying discrete Schwartz – Christoffel transform to conformally project the polygon, made of these points, onto a rectangle, **the points ended up equidistantly distributed on the sides of the rectangle**. This implies moving the points along the contour, via an iterative procedure involving algorithm of Ives and Zacharias (IZ) to conformally map the discrete polygon onto the rectangle, calculate the mismatches between the resultant and the desired locations of the points on the perimeter of the rectangle, adjust the initial positions of the points on the curvilinear contour, repeat IZ-transform, recalculate mismatch, re-adjust, and so on...

On the way during these iterations, also adjust the number of grid points in one of the directions (usually this is η-direction) to match the aspect ratio of the rectangle, $L_\xi : L_\eta \approx N_\xi : N_\eta$ as accurately as possible. This yields $\Delta\xi \approx \Delta\eta$, and, correspondingly, local distances between the adjacent grid points in both directions of the resultant curvilinear grid will be almost equal to each other as well (isotropic resolution).

Once the above converges to yield the desired placement of grid points on the perimeter, **fill the interior of the grid by solving discrete Dirichlet problem**.

Using splines to construct curvilinear contour from a set of user-specified reference points

To constrict *cubic spline* means that for a given set of values

$$\{f_k = f(s_k),\ k = 1, ..., N\}$$

defined at locations $\{s_k,\ k = 1, ..., N\}$, not necessarily equidistant, $\Delta s_{k+1/2} = s_{k+1} - s_k > 0$, hence $\Delta s_{k+1/2} \neq$ const, **find set of derivatives**

$$\left\{d_k = \left.\frac{\partial f}{\partial s}\right|_{s=s_k},\ k = 1, ..., N\right\}$$

such that, assuming that function $f = f(s)$ is represented by a set of cubic polynomials defined individually within each interval $\Delta s_{k+1/2}$, its second derivative, $\delta''_k = \left.\frac{\partial^2 f}{\partial s^2}\right|_{s=s_k}$ is continuous at every junction point $s = s_k$.

To constrict *quintic spline* means to find a set of first and second derivatives, $\{d_k\,,\ \delta''_k\,,\ k = 1, ..., N\}$, such that, assuming that $f = f(s)$ is made of pieces of fifth-order polynomials within each $\Delta s_{k+1/2}$, its third and fourth derivatives are continuous at junction points $s = s_k$.

Either way, constructing spline is nothing else, but a special way to find derivatives (first, or both first and second) at the same locations where the values of function are specified.

The quickest ever derivation of cubic spline

do not try to find anything like this in textbooks

Any cubic polynomial $f = f(\xi)$ defined within interval $\xi \in [-1/2, 1/2]$ may be expressed as

$$f(\xi) = f^L \cdot h^L(\xi) + f^R \cdot h^R(\xi) + \widehat{d}^L \cdot g^L(\xi) + \widehat{d}^R \cdot g^R(\xi)$$

where $h^L = h^L(\xi)$, $h^R = h^R(\xi)$, $g^L = g^L(\xi)$, and $g^R = g^R(\xi)$ are cubic polynomials, which satisfy side b.c. according to the table

	value		$\partial/\partial\xi$	
ξ	$-1/2$	$+1/2$	$-1/2$	$+1/2$
$h^L(\xi)$	1	0	0	0
$h^R(\xi)$	0	1	0	0
$g^L(\xi)$	0	0	1	0
$g^R(\xi)$	0	0	0	1

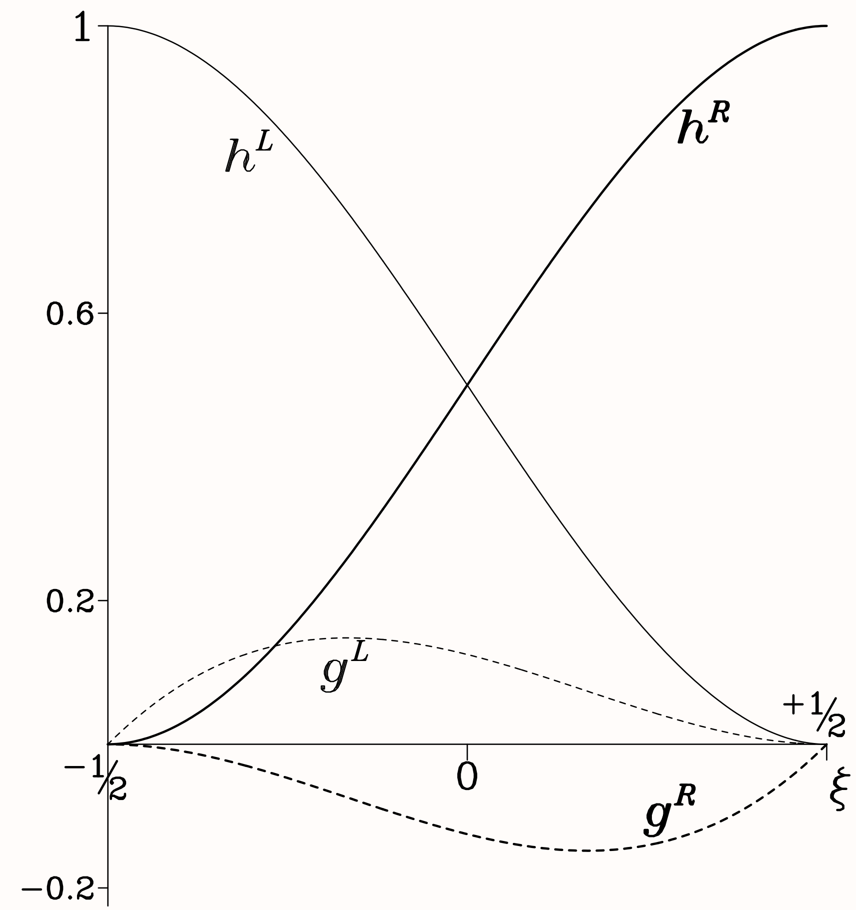
are uniquely defined by the table itself,

$$h^R(\xi) = \frac{1}{2} + \xi\left(\frac{3}{2} - 2\xi^2\right) \qquad g^R(\xi) = \left(\xi^2 - \frac{1}{4}\right)\left(\xi + \frac{1}{2}\right)$$

$$h^L(\xi) = \frac{1}{2} - \xi\left(\frac{3}{2} - 2\xi^2\right) \qquad g^L(\xi) = \left(\xi^2 - \frac{1}{4}\right)\left(\xi - \frac{1}{2}\right)$$

and are known as **Hermite basis functions**, hence

$$f(\xi) \begin{cases} \nearrow f^R, & \xi \to +1/2 \\ \searrow f^L, & \xi \to -1/2 \end{cases} \qquad \frac{\partial f}{\partial \xi} \begin{cases} \nearrow \widehat{d}^R, & \xi \to +1/2 \\ \searrow \widehat{d}^L, & \xi \to -1/2 \end{cases}$$



Once h^L, h^R g^L, and g^R are known, append table with side values of their second derivative

	value		$\partial/\partial\xi$		$\partial^2/\partial\xi^2$	
ξ	$-1/2$	$+1/2$	$-1/2$	$+1/2$	$-1/2$	$+1/2$
$h^L(\xi)$	1	0	0	0	-6	+6
$h^R(\xi)$	0	1	0	0	+6	-6
$g^L(\xi)$	0	0	1	0	-4	+2
$g^R(\xi)$	0	0	0	1	-2	+4

matching condition for the first derivative at $s = s_k$

$$\frac{1}{\Delta s_{k-1/2}} \cdot \underbrace{\lim_{\xi\to+1/2} \frac{\partial f}{\partial \xi}}_{s\in[s_{k-1},s_k]} = \lim_{s\to s_k} \frac{\partial f}{\partial s} \equiv d_k \equiv \lim_{s_k\leftarrow s} \frac{\partial f}{\partial s} = \frac{1}{\Delta s_{k+1/2}} \cdot \underbrace{\lim_{-1/2\leftarrow\xi} \frac{\partial f}{\partial \xi}}_{s\in[s_k,s_{k+1}]}$$

$$\frac{\widehat{d}^R_{s\in[s_{k-1},s_k]}}{\Delta s_{k-1/2}} = d_k = \frac{\widehat{d}^L_{s\in[s_k,s_{k+1}]}}{\Delta s_{k+1/2}} \,,$$

$s \in [s_k, s_{k+1}]$

$$\xi = \frac{s - (s_k + s_{k+1})/2}{\Delta s_{k+1/2}}$$

$\xi \in [-1/2, +1/2]$

$$s = \xi \cdot \Delta s_{k+1/2} + \frac{s_k + s_{k+1}}{2}$$

$$\Delta s_{k+1/2} = s_{k+1} - s_k$$

matching second derivative

$$\frac{1}{\Delta s^2_{k-1/2}} \cdot \underbrace{\lim_{\xi\to+1/2} \frac{\partial^2 f}{\partial \xi^2}}_{s\in[s_{k-1},s_k]} = \lim_{s\to s_k} \frac{\partial^2 f}{\partial s^2} = \lim_{s_k\leftarrow s} \frac{\partial^2 f}{\partial s^2} = \frac{1}{\Delta s^2_{k+1/2}} \cdot \underbrace{\lim_{-1/2\leftarrow\xi} \frac{\partial^2 f}{\partial \xi^2}}_{s\in[s_k,s_{k+1}]}$$

express $f = f(\xi)$ via h^L, h^R, g^L, g^R, take second derivative, and substitute side values of $\partial^2/\partial\xi^2$ from table,

$$\frac{\overbrace{6f_{k-1} - 6f_k + (2d_{k-1} + 4d_k) \cdot \Delta s_{k-1/2}}^{s\in[s_{k-1},s_k],\ s\to s_k}}{\Delta s^2_{k-1/2}} = \frac{\overbrace{-6f_k + 6f_{k+1} - (4d_k + 2d_{k+1}) \cdot \Delta s_{k+1/2}}^{s_k\leftarrow s,\ s\in[s_k,s_{k+1}]}}{\Delta s^2_{k+1/2}}$$

arriving to

$$\frac{d_{k-1}}{\Delta s_{k-1/2}} + \left(\frac{2}{\Delta s_{k-1/2}} + \frac{2}{\Delta s_{k+1/2}}\right) \cdot d_k + \frac{d_{k+1}}{\Delta s_{k+1/2}} = 3 \cdot \left(\frac{f_k - f_{k-1}}{\Delta s^2_{k-1/2}} + \frac{f_{k+1} - f_k}{\Delta s^2_{k+1/2}}\right)$$

two boundary conditions are required, however, for our purposes, for the reason explained later, we are interested only in **periodic b.c.**, via index replacement rules: $k-1 \to N$ if $k=1$, and $k+1 \to 1$ if $k=N$

r.h.s. depends on finite-difference derivatives of the function f, but not the function itself,

$$\frac{d_{k-1}}{\Delta s_{k-1/2}} + \left(\frac{2}{\Delta s_{k-1/2}} + \frac{2}{\Delta s_{k+1/2}}\right)\cdot d_k + \frac{d_{k+1}}{\Delta s_{k+1/2}} = 3\cdot\left(\frac{1}{\Delta s_{k-1/2}}\cdot\left.\frac{\Delta f}{\Delta s}\right|_{k-1/2} + \frac{1}{\Delta s_{k+1/2}}\cdot\left.\frac{\Delta f}{\Delta s}\right|_{k+1/2}\right)$$

where $\Delta f/\Delta s|_{k+1/2} = (f_{k+1}-f_k)/\Delta s_{k+1/2}$ is finite-difference estimate of derivative.

variational principle: cubic spline constructs function $f = f(s)$ which yields minimum possible value of

$$\Phi[f] = \int \left(\frac{\partial^2 f}{\partial s^2}\right)^2 ds = \sum_{k=1}^{N}\left[\int_{s_k}^{s_k+\Delta s_{k+1/2}} \left(\frac{\partial^2 f}{\partial s^2}\right)^2 ds\right]$$

among all piecewise-cubic, continuously differentiable functions going through specified values $f_k = f(s_k)$.

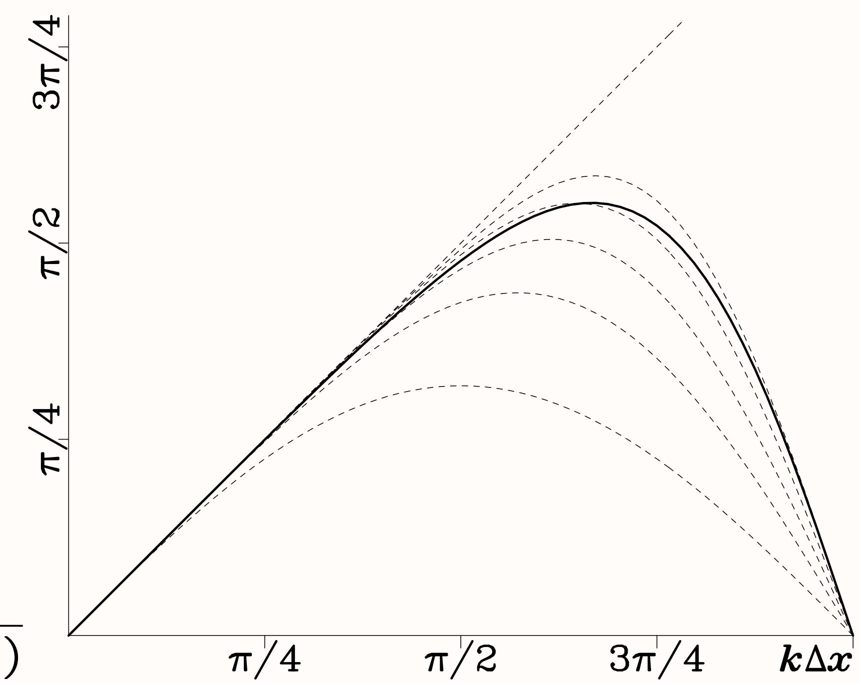
uniform $\Delta s_{j+1/2} = const = \Delta x$ yields

$$\underbrace{\frac{1}{6}\cdot d_{j-1} + \frac{2}{3}\cdot d_j + \frac{1}{6}\cdot d_{j+1}}_{\text{compensate by averaging}} = \underbrace{\frac{1}{2}\cdot\left(\left.\frac{\Delta f}{\Delta x}\right|_{j-1/2} + \left.\frac{\Delta f}{\Delta x}\right|_{j+1/2}\right)}_{\text{averaged } \Delta f/\Delta x} = \frac{f_{j+1}-f_{j-1}}{2\Delta x}$$

d_j computed this way is known as **Padé derivative (a.k.a. compact differencing)**. It is 4th-order accurate, its truncation error 6 times smaller than conventional 4th-order-accurate finite-difference.

Fourier component $f_j = \hat{f}_k \cdot e^{ik\Delta x\cdot j}$

$$\hat{d}\cdot\left(\frac{2}{3}+\frac{1}{3}\cos(k\Delta x)\right) = \hat{f}\cdot\frac{i\cdot\sin(k\Delta x)}{\Delta x} \quad\Rightarrow\quad \hat{d} = ik\cdot\hat{f}\cdot\frac{\sin(k\Delta x)/(k\Delta x)}{(2/3)+(1/3)\cos(k\Delta x)}$$



The quickest derivation of quintic spline ever

Any fifth-order polynomial $f = f(\xi)$, defined within interval $\xi \in [-1/2, +1/2]$ may be cast into form

$$f(\xi) = f^L \cdot H^L(\xi) + f^R \cdot H^R(\xi) + \frac{\partial f}{\partial \xi}\bigg|^L \cdot G^L(\xi) + \frac{\partial f}{\partial \xi}\bigg|^R \cdot G^R(\xi) + \frac{\partial^2 f}{\partial \xi^2}\bigg|^L \cdot D^L(\xi) + \frac{\partial^2 f}{\partial \xi^2}\bigg|^R \cdot D^R(\xi) \qquad -\frac{1}{2} \le \xi \le \frac{1}{2}$$

where the six functions $H^L(\xi)$, $H^R(\xi)$, $G^L(\xi)$, $G^R(\xi)$, $D^L(\xi)$, and $D^R(\xi)$, are polynomials of fifth power, defined in such a way, that their values, first and second derivatives turn into 0 or 1 on the left and right ends according to

	value		$\partial/\partial\xi$		$\partial^2/\partial\xi^2$	
ξ	$-1/2$	$+1/2$	$-1/2$	$+1/2$	$-1/2$	$+1/2$
$H^L(\xi)$	1	0	0	0	0	0
$H^R(\xi)$	0	1	0	0	0	0
$G^L(\xi)$	0	0	1	0	0	0
$G^R(\xi)$	0	0	0	1	0	0
$D^L(\xi)$	0	0	0	0	1	0
$D^R(\xi)$	0	0	0	0	0	1

hence

$$f(\xi) \begin{matrix} \nearrow & f^R, \quad \xi \to +1/2 \\ \searrow & f^L, \quad \xi \to -1/2 \end{matrix} \qquad \frac{\partial f}{\partial \xi} \begin{matrix} \nearrow & \partial f/\partial\xi|^R, \quad \xi \to +1/2 \\ \searrow & \partial f/\partial\xi|^L, \quad \xi \to -1/2 \end{matrix} \qquad \frac{\partial^2 f}{\partial \xi^2} \begin{matrix} \nearrow & \partial^2 f/\partial\xi^2\big|^R, \quad \xi \to +1/2 \\ \searrow & \partial^2 f/\partial\xi^2\big|^L, \quad \xi \to -1/2 \end{matrix}$$

functions $H^L(\xi)$, $H^R(\xi)$, $G^L(\xi)$, $G^R(\xi)$, $D^L(\xi)$, and $D^R(\xi)$, are **Hermite basis functions** (yet unknown).

do not try to find anything like this in textbooks

Let $p = \xi + 1/2$, hence $\quad p \in [0,\, 1] \quad \Leftrightarrow \quad \xi \in [-1/2,\, +1/2]$, hence

$$\mathscr{P}(p) = Ap^5 + Bp^4 + Cp^3 \quad \Rightarrow \quad \mathscr{P}\Big|_{p=0} = 0\,, \quad \frac{\partial \mathscr{P}}{\partial p}\Big|_{p=0} = 0\,, \quad \frac{\partial^2 \mathscr{P}}{\partial p^2}\Big|_{p=0} = 0$$

$$\begin{array}{lcrcrcrcccc}
p = 1 & & & & & & & & H^R & G^R & D^R \\
\mathscr{P}(p) & = & A & + & B & + & C & = & 1 & 0 & 0 \\
\partial \mathscr{P}/\partial p & = & 5A & + & 4B & + & 3C & = & 0 & 1 & 0 \\
\partial^2 \mathscr{P}/\partial p^2 & = & 20A & + & 12B & + & 6C & = & 0 & 0 & 1
\end{array}$$

$$\begin{array}{lcrrr}
H^R(p) & = & 6p^5 & -15p^4 & +10p^3 \\
G^R(p) & = & -3p^5 & +7p^4 & -4p^3 \\
D^R(p) & = & p^5/2 & -p^4 & +p^3/2
\end{array}
\qquad
\begin{array}{l}
H^L(p) = H^R(1-p) \\
G^L(p) = -G^R(1-p) \\
D^L(p) = D^R(1-p)
\end{array}$$

$$H^R(\xi) = \frac{1}{2} + \xi \left(\frac{15}{8} - 5\xi^2 + 6\xi^4 \right)$$

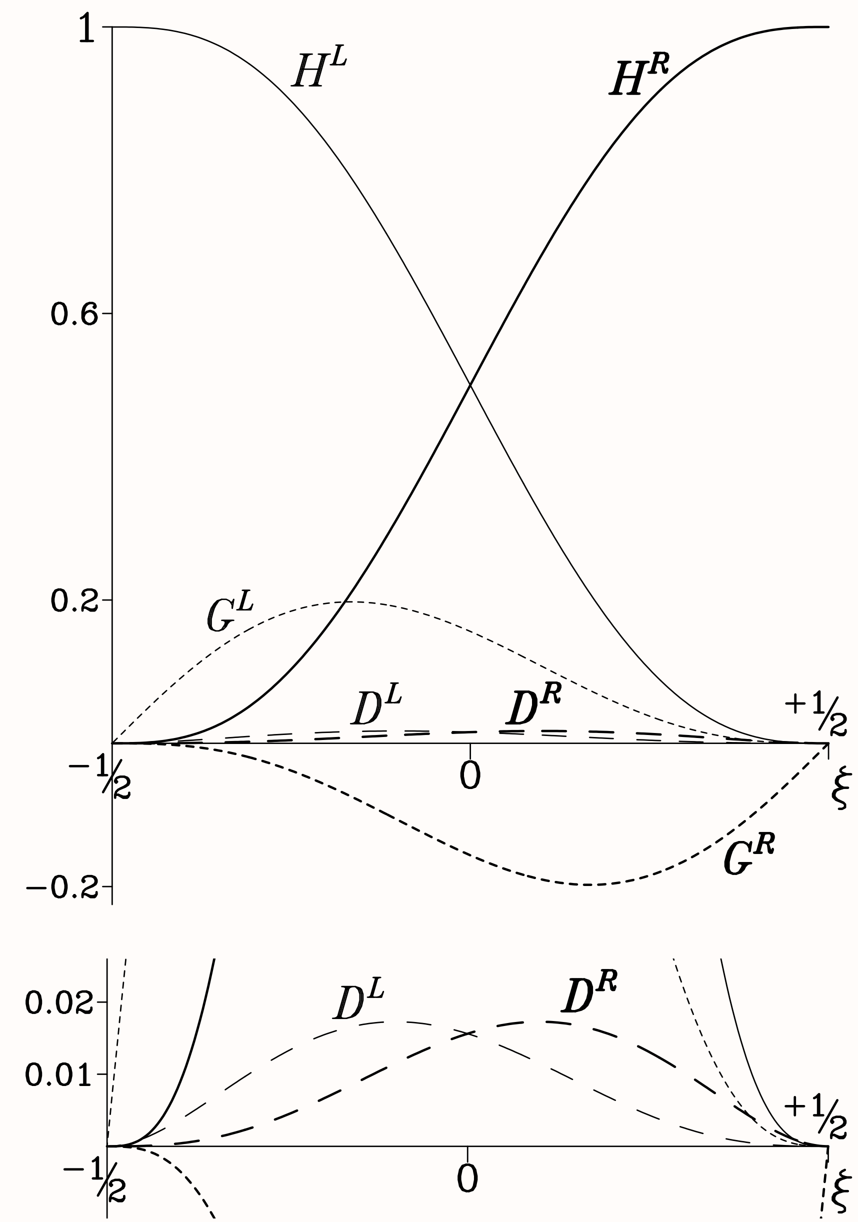
$$H^L(\xi) = \frac{1}{2} - \xi \left(\frac{15}{8} - 5\xi^2 + 6\xi^4 \right)$$

$$G^R(\xi) = \left(\frac{1}{4} - \xi^2 \right) \left[\xi \left(3\xi^2 - \frac{7}{4} \right) + \left(\frac{\xi^2}{2} - \frac{5}{8} \right) \right]$$

$$G^L(\xi) = \left(\frac{1}{4} - \xi^2 \right) \left[\xi \left(3\xi^2 - \frac{7}{4} \right) - \left(\frac{\xi^2}{2} - \frac{5}{8} \right) \right]$$

$$D^R(\xi) = \frac{1}{2} \left(\frac{1}{4} - \xi^2 \right)^2 \left(\frac{1}{2} + \xi \right)$$

$$D^L(\xi) = \frac{1}{2} \left(\frac{1}{4} - \xi^2 \right)^2 \left(\frac{1}{2} - \xi \right)$$



	value		$\partial/\partial\xi$		$\partial^2/\partial\xi^2$		$\partial^3/\partial\xi^3$		$\partial^4/\partial\xi^4$	
ξ	$-1/2$	$+1/2$	$-1/2$	$+1/2$	$-1/2$	$+1/2$	$-1/2$	$+1/2$	$-1/2$	$+1/2$
$H^L(\xi)$	1	0	0	0	0	0	-60	-60	+360	-360
$H^R(\xi)$	0	1	0	0	0	0	+60	+60	-360	+360
$G^L(\xi)$	0	0	1	0	0	0	-36	-24	+192	-168
$G^R(\xi)$	0	0	0	1	0	0	-24	-36	+168	-192
$D^L(\xi)$	0	0	0	0	1	0	-9	-3	+36	-24
$D^R(\xi)$	0	0	0	0	0	1	+3	+9	-24	+36

matching condition for the first derivative at $s = s_k$

$$\lim_{s\to s_k} \frac{\partial f}{\partial s} = \underbrace{\frac{1}{\Delta s_{k-1/2}} \cdot \frac{\partial f}{\partial \xi}\bigg|^{R}}_{s\in[s_{k-1},s_k],\ s\to s_k} = d_k \equiv \frac{\partial f}{\partial s}\bigg|_{s=s_k} = \underbrace{\frac{1}{\Delta s_{k+1/2}} \cdot \frac{\partial f}{\partial \xi}\bigg|^{L}}_{s_k\leftarrow s,\ s\in[s_k,s_{k+1}]} = \lim_{s_k\leftarrow s} \frac{\partial f}{\partial s}$$

matching the second derivative

$$\lim_{s\to s_k} \frac{\partial^2 f}{\partial s^2} = \underbrace{\frac{1}{\Delta s^2_{k-1/2}} \cdot \frac{\partial^2 f}{\partial \xi^2}\bigg|^{R}}_{s\in[s_{k-1},s_k],\ s\to s_k} = \delta''_k \equiv \frac{\partial^2 f}{\partial s^2}\bigg|_{s=s_k} = \underbrace{\frac{1}{\Delta s^2_{k+1/2}} \cdot \frac{\partial^2 f}{\partial \xi^2}\bigg|^{L}}_{s_k\leftarrow s,\ s\in[s_k,s_{k+1}]} = \lim_{s_k\leftarrow s} \frac{\partial^2 f}{\partial s^2}$$

continuity of the third derivative at $s = s_k$

$$\lim_{s\to s_k} \frac{\partial^3 f}{\partial s^3} = \underbrace{\frac{1}{\Delta s^3_{k-1/2}} \cdot \left.\frac{\partial^3 f}{\partial \xi^3}\right|^R}_{s\in[s_{k-1},s_k]} = \left.\frac{\partial^3 f}{\partial s^3}\right|_{s=s_k} = \underbrace{\frac{1}{\Delta s^3_{k+1/2}} \cdot \left.\frac{\partial^3 f}{\partial \xi^3}\right|^L}_{s\in[s_k,s_{k+1}]} = \lim_{s_k\leftarrow s} \frac{\partial^3 f}{\partial s^3}$$

express $f(\xi)$ via $H^L(\xi), \ldots, D^R(\xi)$, take its third derivative, and substitute side-values for $\partial^3/\partial\xi^3$ from the table

$$\overbrace{\frac{60 f_k - 60 f_{k-1} - 36 \Delta s_{k-1/2} d_k - 24 \Delta s_{k-1/2} d_{k-1} + 9 \Delta s^2_{k-1/2} \delta''_k - 3 \Delta s^2_{k-1/2} \delta''_{k-1}}{\Delta s^3_{k-1/2}}}^{\text{substitute } \xi = +1/2,\ k-1 \text{ for } ^L,\ k \text{ for } ^R,\ s\in[s_{k-1},s_k],\ s\to s_k}$$

$$= \underbrace{\frac{60 f_{k+1} - 60 f_k - 24 \Delta s_{k+1/2} d_{k+1} - 36 \Delta s_{k+1/2} d_k + 3 \Delta s^2_{k+1/2} \delta''_{k+1} - 9 \Delta s^2_{k+1/2} \delta''_k}{\Delta s^3_{k+1/2}}}_{s_k\leftarrow s,\ s\in[s_k,s_{k+1}],\ \text{substitute } \xi = -1/2,\ k \text{ for } ^L,\ k+1 \text{ for } ^R}$$

moving unknowns to the left, knowns to the right,

$$-\frac{2}{5} \cdot \frac{d_{k-1}}{\Delta s^2_{k-1/2}} - \frac{3}{5} \cdot \left(\frac{1}{\Delta s^2_{k-1/2}} - \frac{1}{\Delta s^2_{k+1/2}} \right) \cdot d_k + \frac{2}{5} \cdot \frac{d_{k+1}}{\Delta s^2_{k+1/2}}$$
$$-\frac{1}{20} \cdot \frac{\delta''_{k-1}}{\Delta s_{k-1/2}} + \frac{3}{20} \cdot \left(\frac{1}{\Delta s_{k-1/2}} + \frac{1}{\Delta s_{k+1/2}} \right) \cdot \delta''_k - \frac{1}{20} \cdot \frac{\delta''_{k+1}}{\Delta s_{k+1/2}}$$
$$= \frac{f_{k+1} - f_k}{\Delta s^3_{k+1/2}} - \frac{f_k - f_{k-1}}{\Delta s^3_{k-1/2}}$$

continuity of the fourth derivative at $s = s_k$

$$\underbrace{\frac{1}{\Delta s^4_{k-1/2}} \cdot \frac{\partial^4 f}{\partial \xi^4}\bigg|^R}_{s\in[s_{k-1},s_k]} = \frac{\partial^4 f}{\partial s^4}\bigg|_{s=s_k} = \underbrace{\frac{1}{\Delta s^4_{k+1/2}} \cdot \frac{\partial^4 f}{\partial \xi^4}\bigg|^L}_{s\in[s_k,s_{k+1}]}$$

express $f(\xi)$ via $H^L(\xi), \ldots, D^R(\xi)$, take its fourth derivative, and substitute side-values for $\partial^4/\partial\xi^4$ from the table

$$\overbrace{\frac{360 f_k - 360 f_{k-1} - 192\Delta s_{k-1/2} d_k - 168\Delta s_{k-1/2} d_{k-1} + 36\Delta s^2_{k-1/2}\delta''_k - 24\Delta s^2_{k-1/2}\delta''_{k-1}}{\Delta s^4_{k-1/2}}}^{\text{substitute } \xi = +1/2,\ k-1 \text{ for } L,\ k \text{ for } R,\ s\in[s_k, s_{k-1}, s_k],\ s\to s_k}$$

$$= \underbrace{\frac{-360 f_{k+1} + 360 f_k + 168\Delta s_{k+1/2} d_{k+1} + 192\Delta s_{k+1/2} d_k - 24\Delta s^2_{k+1/2}\delta''_{k+1} + 36\Delta s^2_{k+1/2}\delta''_k}{\Delta s^4_{k+1/2}}}_{s_k \leftarrow s,\ s\in[s_k, s_{k+1}],\ \text{for } \xi = -1/2,\ k+1 \text{ for } R,\ k \text{ for } L}$$

moving unknowns to the left, knowns to the right

$$\frac{7}{15} \cdot \frac{d_{k-1}}{\Delta s^3_{k-1/2}} + \frac{8}{15} \cdot \left(\frac{1}{\Delta s^3_{k-1/2}} + \frac{1}{\Delta s^3_{k+1/2}}\right) \cdot d_k + \frac{7}{15} \cdot \frac{d_{k+1}}{\Delta s^3_{k+1/2}}$$
$$+ \frac{1}{15} \cdot \frac{\delta''_{k-1}}{\Delta s^2_{k-1/2}} + \frac{1}{10} \cdot \left(\frac{1}{\Delta s^2_{k-1/2}} - \frac{1}{\Delta s^2_{k+1/2}}\right) \cdot \delta''_k - \frac{1}{15} \cdot \frac{\delta''_{k+1}}{\Delta s^2_{k+1/2}}$$
$$= \frac{f_{k+1} - f_k}{\Delta s^4_{k-1/2}} + \frac{f_k - f_{k-1}}{\Delta s^4_{k+1/2}}$$

Combining continuity conditions the third and fourth derivatives

$$\boldsymbol{a}_k\cdot\boldsymbol{d}_{k-1}+\boldsymbol{b}_k\cdot\boldsymbol{d}_k+\boldsymbol{c}_k\cdot\boldsymbol{d}_{k+1}=\boldsymbol{f}_k \qquad \forall\ k=1,...,N \text{ (see below)}$$

where

$$\boldsymbol{a}_k=\begin{pmatrix}\dfrac{7}{15\Delta s^3_{k-1/2}} & \dfrac{1}{15\Delta s^2_{k-1/2}}\\ -\dfrac{2}{5\Delta s^2_{k-1/2}} & -\dfrac{1}{20\Delta s_{k-1/2}}\end{pmatrix} \qquad \boldsymbol{c}_k=\begin{pmatrix}\dfrac{7}{15\Delta s^3_{k+1/2}} & -\dfrac{1}{15\Delta s^2_{k+1/2}}\\ \dfrac{2}{5\Delta s^2_{k+1/2}} & -\dfrac{1}{20\Delta s_{k+1/2}}\end{pmatrix} \qquad \boldsymbol{d}_k=\begin{pmatrix}d_k\\ \delta''_k\end{pmatrix}$$

$$\boldsymbol{b}_k=\begin{pmatrix}\dfrac{8}{15}\cdot\left(\dfrac{1}{\Delta s^3_{k-1/2}}+\dfrac{1}{\Delta s^3_{k+1/2}}\right) & \dfrac{1}{10}\cdot\left(\dfrac{1}{\Delta s^2_{k-1/2}}-\dfrac{1}{\Delta s^2_{k+1/2}}\right)\\ -\dfrac{3}{5}\cdot\left(\dfrac{1}{\Delta s^2_{k-1/2}}-\dfrac{1}{\Delta s^2_{k+1/2}}\right) & \dfrac{3}{20}\cdot\left(\dfrac{1}{\Delta s_{k-1/2}}+\dfrac{1}{\Delta s_{k+1/2}}\right)\end{pmatrix} \qquad \boldsymbol{f}_k=\begin{pmatrix}\dfrac{f_{k+1}-f_k}{\Delta s^4_{k-1/2}}+\dfrac{f_k-f_{k-1}}{\Delta s^4_{k+1/2}}\\ \dfrac{f_{k+1}-f_k}{\Delta s^3_{k+1/2}}-\dfrac{f_k-f_{k-1}}{\Delta s^3_{k-1/2}}\end{pmatrix}$$

this is a well posed (diagonally dominant) **block tri-diagonal system** of linear equations

two boundary conditions are required at the ends, $k=1$, and $k=N$, however, for our purpose of building contour of the grid, we are interested only in periodic closure conditions, which can be expressed via index-folding rules:

$$\begin{array}{llll}\text{for} & k=1 & \text{replace} & k-1 \rightarrow N\\ \text{for} & k=N & \text{replace} & k+1 \rightarrow 1\end{array}$$

method of solution is similar to tri-diagonal solver for cubic spline, except that operations with numbers are replaced with operation over vectors and 2×2 matrices (division by multiplication of matrix inverse)

Fourier analysis of accuracy for quintic spline: assume uniform $\Delta s_j = const = \Delta x$, hence

$$
\begin{aligned}
\frac{7}{15}d_{j-1} + \frac{16}{15}d_j + \frac{7}{15}d_{j+1} + \frac{\Delta x}{15}\delta''_{j-1} - \frac{\Delta x}{15}\delta''_{j+1} &= \frac{f_{j+1}-f_{j-1}}{\Delta x} \\
-\frac{2}{5\Delta x}d_{j-1} + \frac{2}{5\Delta x}d_{j+1} - \frac{1}{20}\delta''_{j-1} + \frac{3}{10}\delta''_j - \frac{1}{20}\delta''_{j+1} &= \frac{f_{j+1}-2f_j+f_{j-1}}{\Delta x^2}
\end{aligned}
$$

Substitute $f_j = \hat{f}_k \cdot e^{ik\Delta x\, j}$, $d_j = \hat{d}_k \cdot e^{ik\Delta x\, j}$ and $\delta''_j = \hat{\delta''}_k \cdot e^{ik\Delta x\, j}$ into the above,

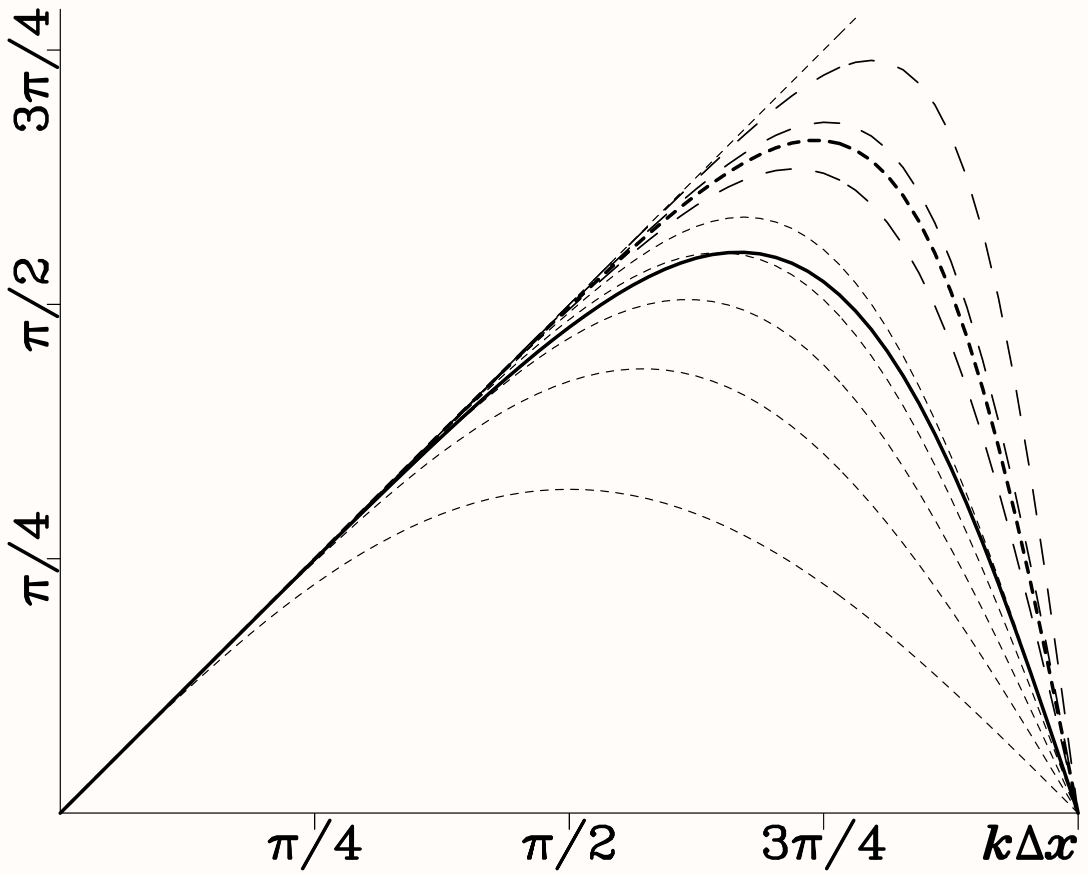
$$
\begin{aligned}
\hat{d}_k \cdot \left(\frac{16}{15} + \frac{14}{15}\cos(k\Delta x)\right) - \frac{2}{15}\Delta x \cdot \hat{\delta''}_k \cdot i\sin(k\Delta x) &= \hat{f}_k \cdot \frac{2i \cdot \sin(k\Delta x)}{\Delta x} \\
\frac{4}{5\Delta x} \cdot \hat{d}_k \cdot i\sin(k\Delta x) + \hat{\delta''}_k \cdot \left(\frac{3}{10} - \frac{1}{10}\cos(k\Delta x)\right) &= 2\hat{f}_k \cdot \frac{\cos(k\Delta x)-1}{\Delta x^2}
\end{aligned}
$$

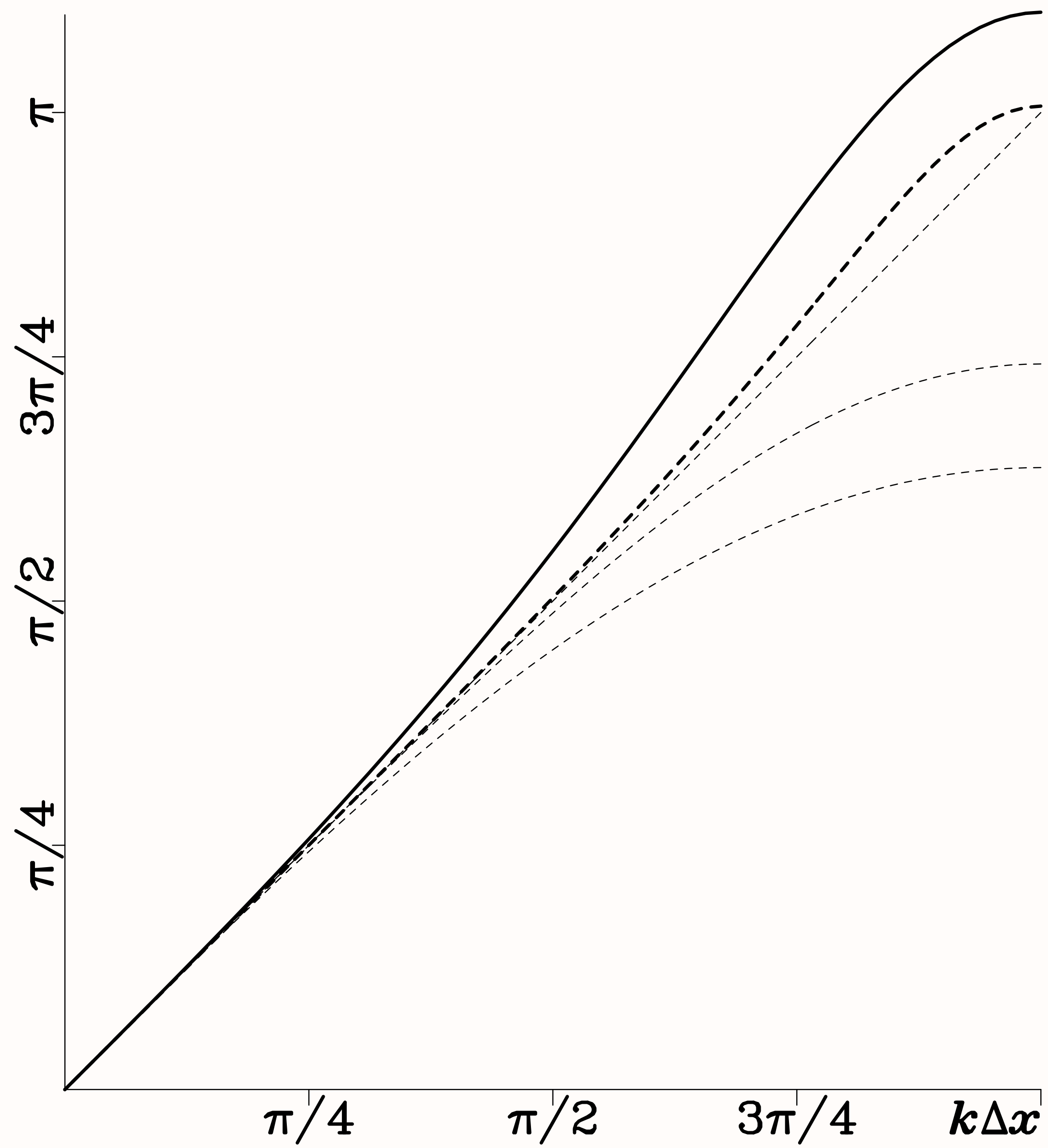
where $\hat{d}_k$ and $\hat{\delta''}_k$ are the unknowns. Solving it as a 2×2 linear system

$$
\begin{aligned}
\hat{d}_k &= \hat{f}_k \cdot \frac{i\sin(k\Delta x)}{\Delta x} \cdot \frac{25 + 5\cos(k\Delta x)}{16 + 13\cos(k\Delta x) + \cos^2(k\Delta x)} \\
\hat{\delta''}_k &= \hat{f}_k \cdot \frac{20}{\Delta x^2} \cdot \frac{\cos(k\Delta x) + \cos^2(k\Delta x) - 2}{16 + 13\cos(k\Delta x) + \cos^2(k\Delta x)}
\end{aligned}
$$

Taylor expansion for $k\Delta x \ll 1$ indicates *the sixth* and *the fourth* order of accuracy respectively,

$$
\begin{aligned}
\hat{d}_k &= ik \cdot \hat{f}_k \cdot \frac{30 - \frac{15}{2}(k\Delta x)^2 + \frac{21}{24}(k\Delta x)^4 - \frac{17}{112}(k\Delta x)^6 + \ldots}{30 - \frac{15}{2}(k\Delta x)^2 + \frac{21}{24}(k\Delta x)^4 - \frac{1}{16}(k\Delta x)^6 + \ldots} \\
\hat{\delta''}_k &= -k^2 \cdot \hat{f}_k \cdot \frac{30 - \frac{15}{2}(k\Delta x)^2 + \frac{11}{12}(k\Delta x)^4 - \ldots}{30 - \frac{15}{2}(k\Delta x)^2 + \frac{21}{24}(k\Delta x)^4 - \ldots}
\end{aligned}
$$



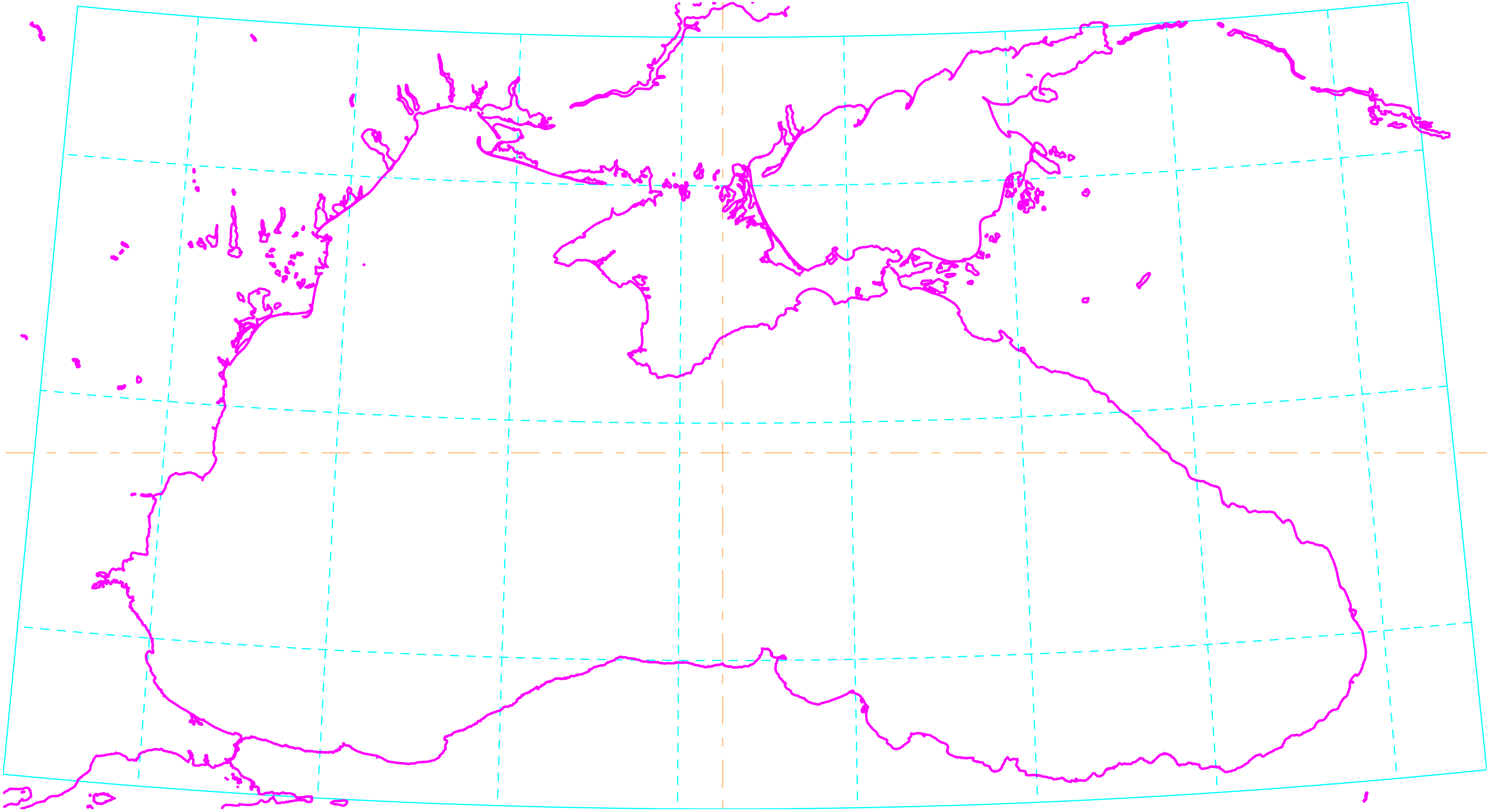


Left: Fourier image multiplier $\mathscr{K} = \mathscr{K}(k\Delta x)$ (ideally should be $\mathscr{K} = k$) for the first derivative computed by *cubic* (**bold solid** line) and *quintic* (**bold with short dashes**) splines. For comparison: five thin lines with short dashes correspond to conventional finite-difference schemes of 2nd, 4th, 5th, 8th, and 10th orders of accuracy. Three thin lines with long dashes above them are for compact-differencing schemes of Lele (1992): 6th- and 8th-order of accuracy tri-diagonal schemes, 10th-order pentadiagonal (correspond to curves e,f,h in Fig 1 from Lele, 1992). *Right*: same, but for image multiplier of the second derivative δ'' shown here as $\mathscr{K} = \sqrt{\mathscr{K}^2(k\Delta x)}$ (ideally $\mathscr{K} = k$).

Setting up perimeter of the future grid

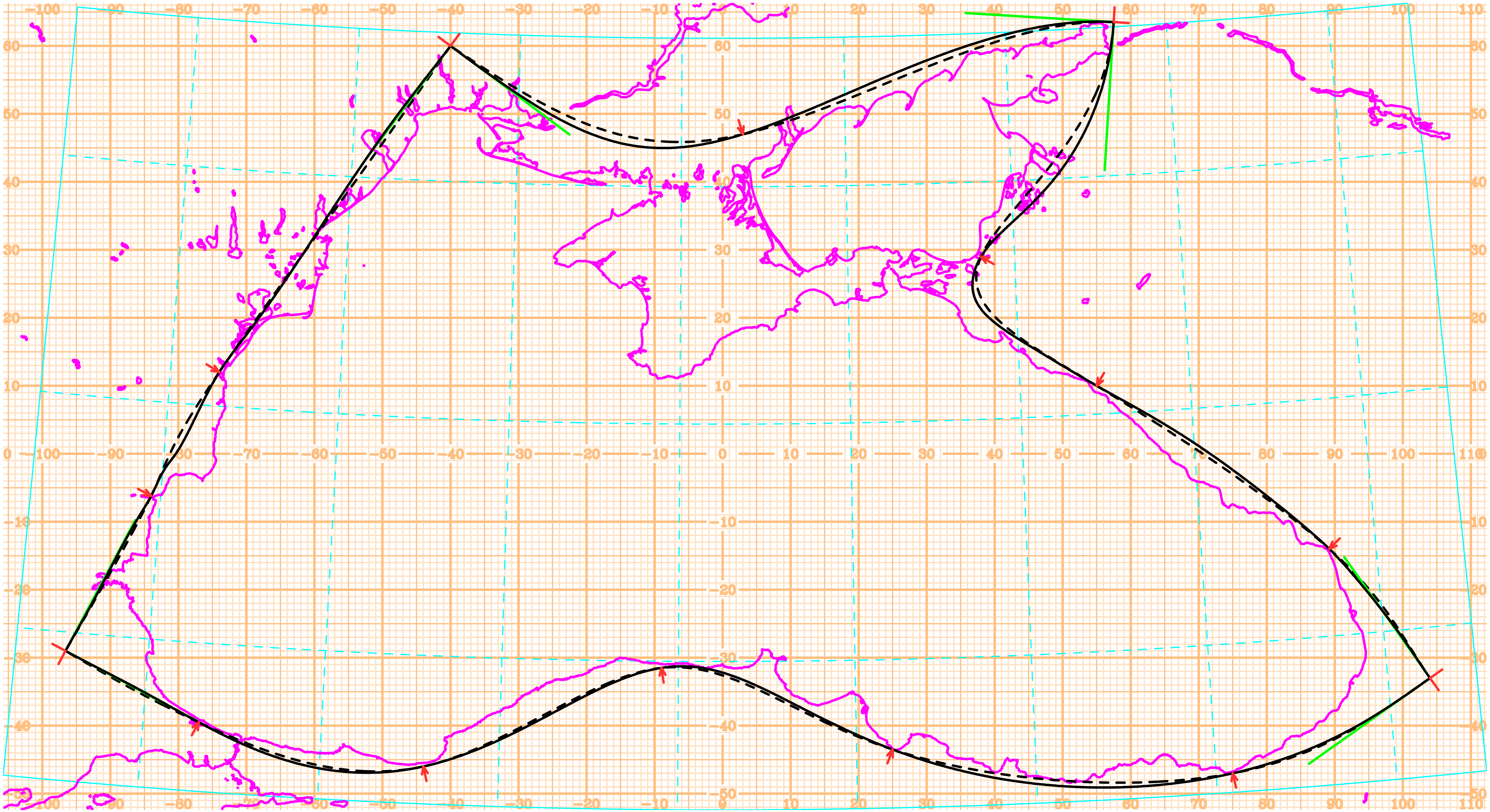
Building map

Black sea. Rotated Mercator projection



Defining contour: cubic (dashed line); quintic (solid) spline

Cartesian system of user coordinates

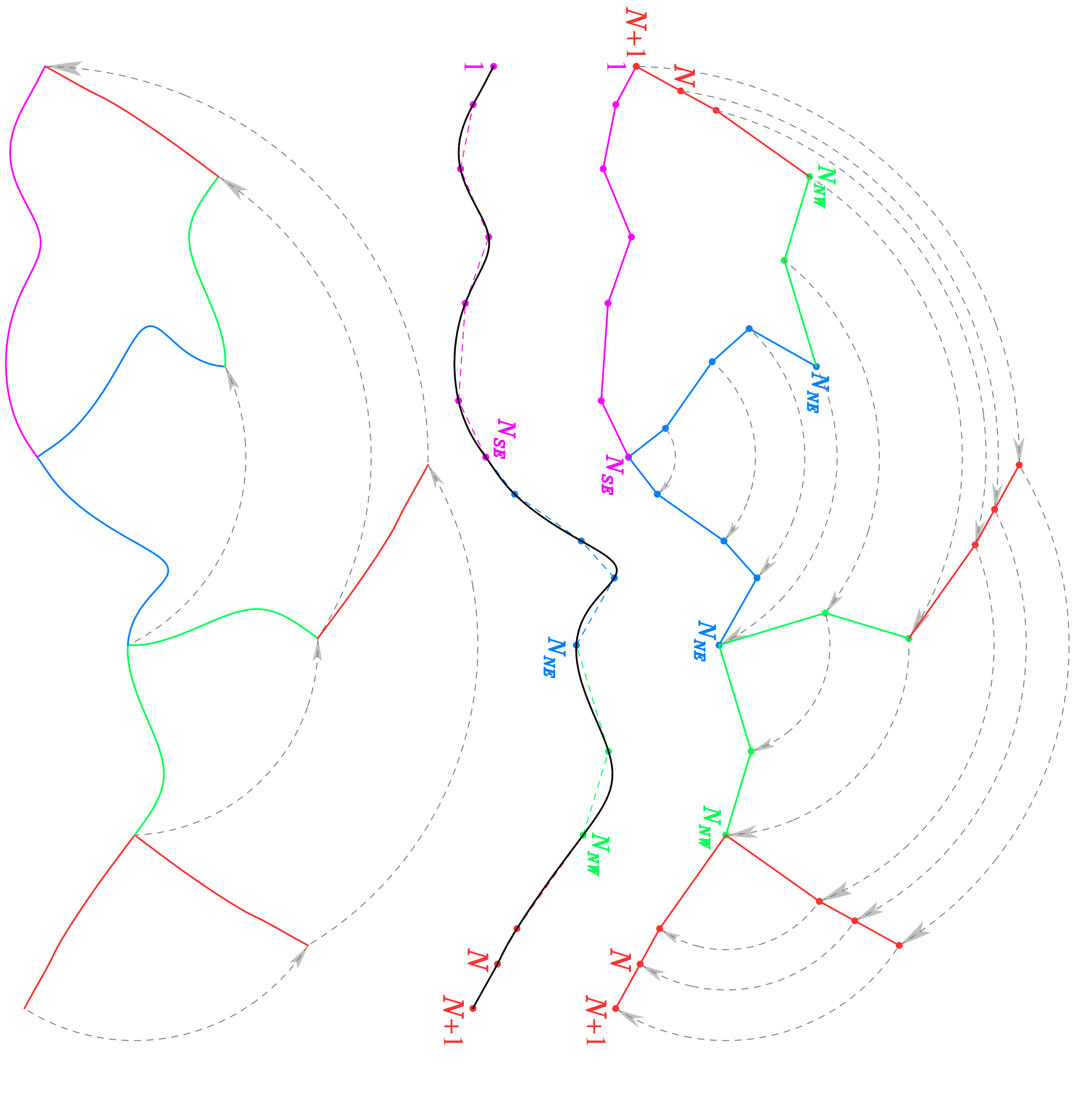


User input file to generate map and contour in the previous page

```
mode=1 latlongrid=2 spline_type=4 npass=4
proj=ME rlat=43.75 rlon=34.5 rota=0
 west_edge=26.5    east_edge=43
 south_edge=40.75  north_edge=47.25
nx=100 ny=66
uscale=0.001
------
   -97    -26        south-west
   -77    -39.5
   -44    -46
   -9     -31.5
   +25    -44
   +75    -47
   +104   -33     <  south-east
   +89.2  -14
   +55    +10
   +40    +30
   +57.5  +63.5   <  north-east
   +3     +47
   -40    +60     <  north-west
   -74    +12
   -84    -6.2
```

selector of the regime
map projection parameters
geographic extents of the map
dimensions of the future grid
scale factor for user coordinates
end of the header

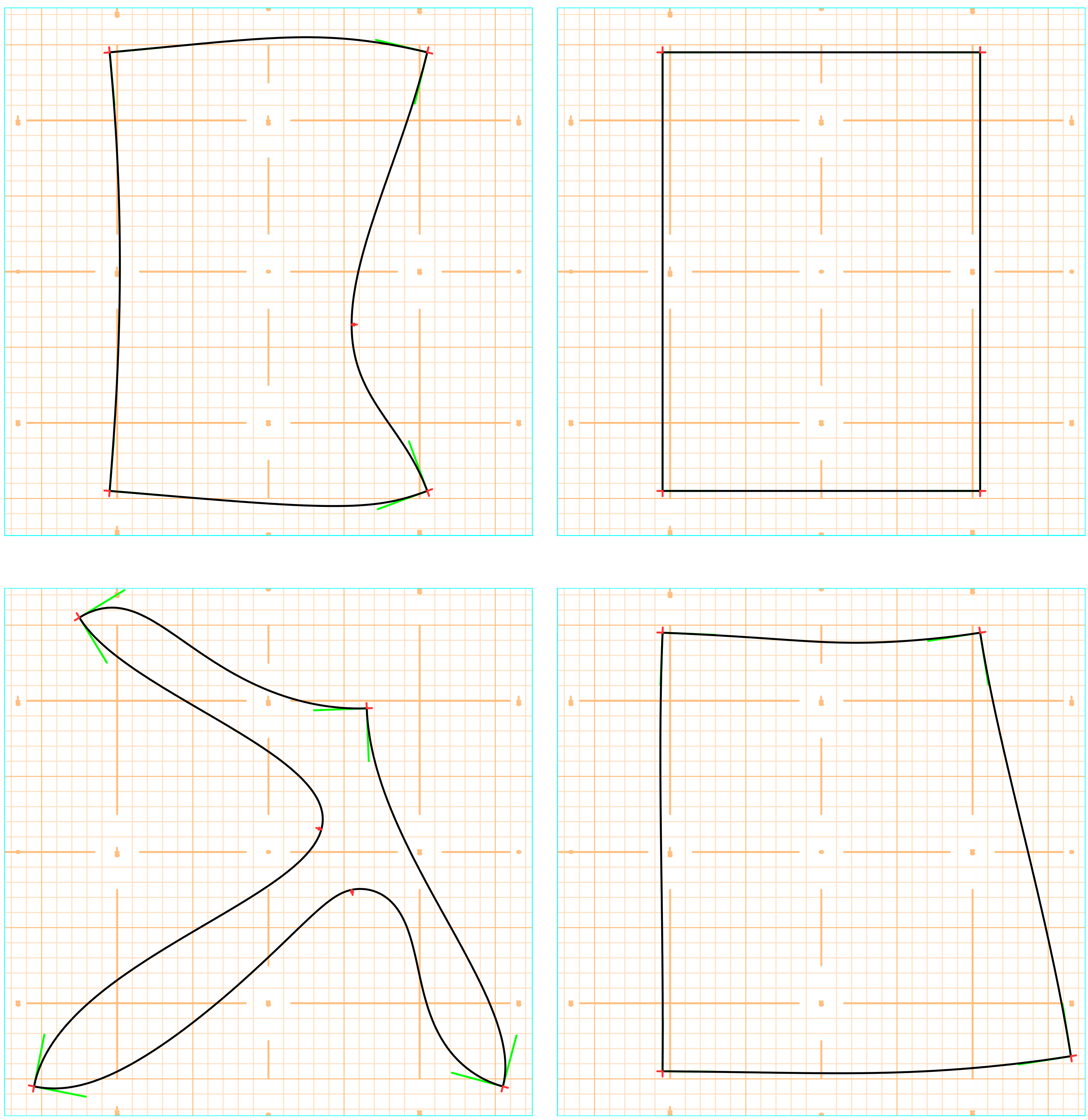
Cartesian coordinates (user system) of reference points for contour spline (tips of red arrows)



once unfolded, periodic b.c. at the ends $\begin{cases} k-1 \to N, & \text{if } k=1 \\ k+1 \to 1, & \text{if } k=N \end{cases}$

$\Rightarrow$ **exact 90-degree angles at the corners**

spline routine actually does not know anything about the corners



Exact 90-degree angles at side junctions no matter what!

...so it looks straightforward, but actually it is not:

Spline in 1D: $f = f(s)$
given set points $\{s_k,\ k = 1, ..., N\}$
and set of values $\{f_k = f(s_k),\ k = 1, ..., N\}$

construct set $\left\{ d_k = df/ds\big|_{s=s_k} \right\}$

such that second derivative d^2f/ds^2
maintain continuity at every $s = s_k$.

2D spline dilemma
given set of points $\{(x_k, y_k)\,, k = 1, ..., N\}$

construct continuous curve $(x, y) = \big(x(s), y(s)\big)$

such that it goes through all (x_k, y_k), and
its derivatives dx/ds, d^2x/ds^2, dy/ds, d^2y/ds^2
satisfy matching conditions to yield
(i) smoothness (no kinks) and
(ii) continuity of curvature at all (x_k, y_k).

derivatives d/ds with respect to what ???

coordinate s is not a-priori defined

...if it can be imagined, it can be done! *Lockheed Martin*

Nowadays this is called BOOTSTRAPPING !

Possible solutions:

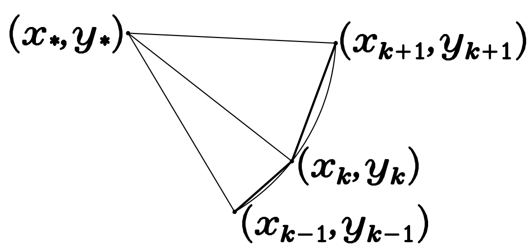
"index" coordinate, $s_k = k$, regardless of the actual distances between consecutive points

coordinate based on straight-line distances

$$s_1 = 0\,, \qquad s_{k+1} = s_k + \sqrt{(x_{k+1}-x_k)^2 + (y_{k+1}-y_k)^2}\,, \qquad k = 1, ..., N-1$$

coordinate using circular ark distances

draw circular arc for each three consecutive points



compute its radius

$$\frac{1}{r_k} = \frac{2\cdot\left(\Delta x_{k-1/2}\cdot\Delta y_{k+1/2} - \Delta x_{k+1/2}\cdot\Delta y_{k-1/2}\right)}{\sqrt{\left(\Delta y_{k+1/2}\cdot\Delta s^2_{k-1/2} + \Delta y_{k-1/2}\cdot\Delta s^2_{k+1/2}\right)^2 + \left(\Delta x_{k+1/2}\cdot\Delta s^2_{k-1/2} + \Delta x_{k-1/2}\cdot\Delta s^2_{k+1/2}\right)^2}}$$

where

$$\left.\begin{array}{l}\Delta x_{k+1/2} = x_{k+1} - x_k \\ \Delta y_{k+1/2} = y_{k+1} - y_k\end{array}\right\} \qquad \Delta s^2_{k+1/2} = \Delta x^2_{k+1/2} + \Delta y^2_{k+1/2}$$

Radius (curvature $1/r_k$) is considered being positive (negative), if turning to the left(right), when going in the direction of index k increase; There is no singularity, $1/r_k = 0$, if the three points are on a straight line.

For each segment $(x,y)_k \to (x,y)_{k+1}$ there are two estimates of curvature, $1/r_k$ and $1/r_{k+1}$. Average them

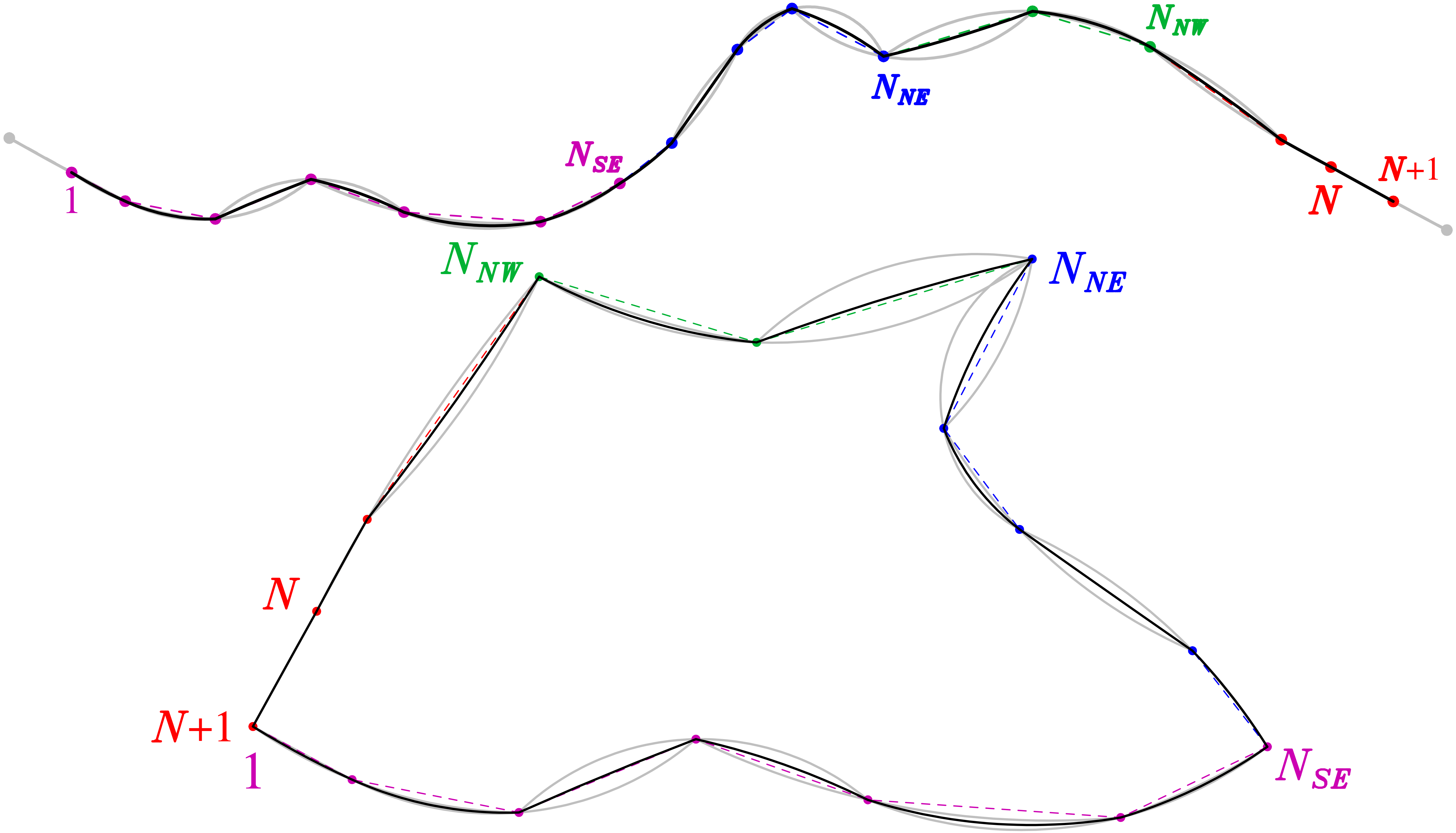
$$\frac{1}{r_{k+1/2}} = \frac{1}{2}\cdot\left|\frac{1}{r_k} + \frac{1}{r_{k+1}}\right|\,,$$

if the two are of the opposite signs, so be it:
the resultant curvature is less than either of them.

Compute the distance along the arc segment

$$\widehat{\Delta s}_{k+1/2} = 2r_{k+1/2}\cdot\arcsin\left(\frac{\Delta s_{k+1/2}}{2r_{k+1/2}}\right)$$

no singularity: when $1/r_{k+1/2} \to 0$, simply $\widehat{\Delta s}_{k+1/2} \to \Delta s_{k+1/2}$



N_{NW}
N_{NE}
N_{SE}
1
N
$N+1$
N_{NW}
N_{NE}
N
$N+1$
1
N_{SE}

"true" distance along the curve: requires iterative solver

set initial approximation for $\left\{\Delta s_{k+1/2} \,\middle|\, k=1,..,N\right\}$

iterate

- construct splines $x = x(s)$ and $y = y(s)$
- recompute $\left\{\Delta s_{k+1/2}\right\}$ by integrating distances *along the curve* within each segment

$$\Delta s^{(\mathrm{new})}_{k+1/2} = \int_{(x_k,y_k)}^{(x_{k+1},y_{k+1})} \sqrt{\left(\frac{dx}{ds}\right)^2 + \left(\frac{dy}{ds}\right)^2}\, ds$$

$$s^{(\mathrm{new})}_1 = 0\,, \qquad s^{(\mathrm{new})}_{k+1} = s^{(\mathrm{new})}_k + \Delta s^{(\mathrm{new})}_{k+1/2}$$

- substitute $\left\{\Delta s^{(\mathrm{new})}_{k+1/2}\right\} \to \left\{\Delta s_{k+1/2}\right\}$ and $\left\{s^{(\mathrm{new})}_k\right\} \to \{s_k\}$

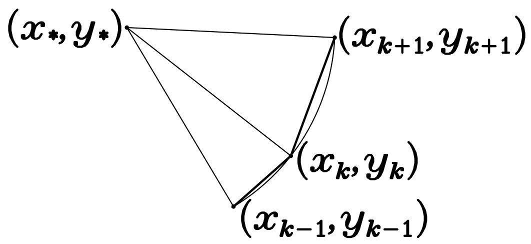
computing integral $\displaystyle\int_{(x_k,y_k)}^{(x_{k+1},y_{k+1})} \sqrt{\left(\frac{dx}{ds}\right)^2 + \left(\frac{dy}{ds}\right)^2}\, ds \approx \sum_{j=1}^{M-1} \sqrt{\left(x_{j+1}-x_j\right)^2 + \left(y_{j+1}-y_j\right)^2}$ implies filling curve with spline-interpolated points $\left\{(x_j,y_j) \,\middle|\, j=1,...,M\right\}$, where $(x_{j=1}, y_{j=1}) = (x_k, y_k)$ and $\left(x_{j=M}, y_{j=M}\right) = (x_{k+1}, y_{k+1})$

to accelerate convergence, summation of straight-line distances may be replaced with circular arc distances;

converges to yield $\left(\frac{dx}{ds}\right)^2 + \left(\frac{dy}{ds}\right)^2 \equiv 1$

Proof of continuity of curvature:

build spline, fill it with points,
compute vector of curvature using consecutive triplets of points



for each point (x_k, y_k) we know radius r_k and direction to the center.

Define **vector of curvature** as

$$-const \cdot \frac{1}{|r_k|} \cdot \boldsymbol{n}_k$$

where $\boldsymbol{n}_k$ is unit vector pointing to the center;
$const$ is a arbitrary scaling factor;
negative sign is to make it point to the convex side of the curve to avoid line interference.

Then track down envelope formed by the ends of all these vectors.

define

$$\left.\begin{array}{l} \Delta x_{k+1/2} = x_{k+1} - x_k \\ \Delta y_{k+1/2} = y_{k+1} - y_k \end{array}\right\} \qquad \Delta s^2_{k+1/2} = \Delta x^2_{k+1/2} + \Delta y^2_{k+1/2}$$

then

$$\frac{1}{r_k} = \frac{2 \cdot \left(\Delta x_{k-1/2} \cdot \Delta y_{k+1/2} - \Delta x_{k+1/2} \cdot \Delta y_{k-1/2}\right)}{\sqrt{\left(\Delta y_{k+1/2} \cdot \Delta s^2_{k-1/2} + \Delta y_{k-1/2} \cdot \Delta s^2_{k+1/2}\right)^2 + \left(\Delta x_{k+1/2} \cdot \Delta s^2_{k-1/2} + \Delta x_{k-1/2} \cdot \Delta s^2_{k+1/2}\right)^2}}$$

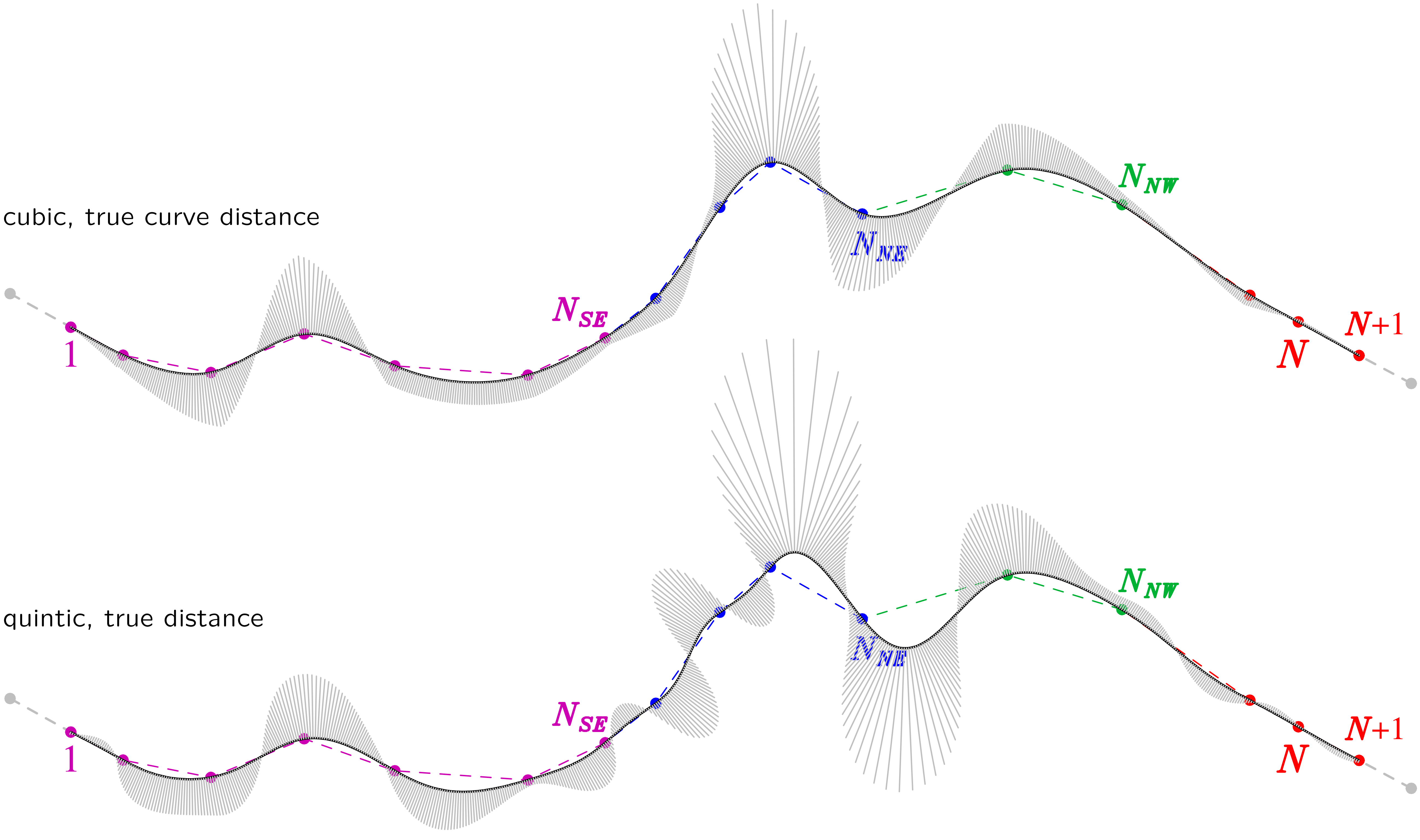
positive, if turning to the left, when going in the direction of index k increase; negative otherwise; 0 if the three points are on a straight line

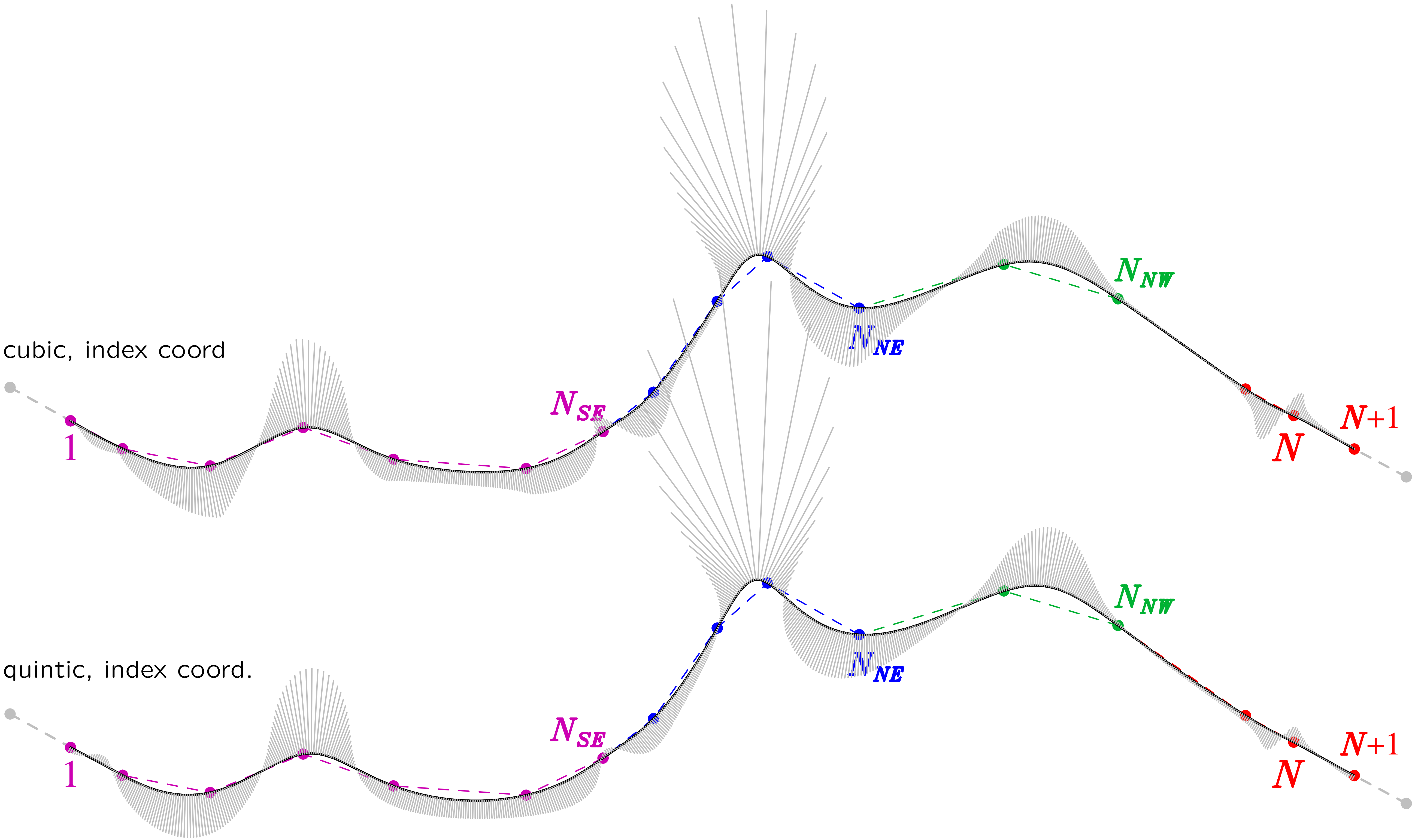
unit vector from (x_k, y_k) to the center

$$n_k^{(x)} = -\frac{\Delta y_{k+1/2} \cdot \Delta s^2_{k-1/2} + \Delta y_{k-1/2} \cdot \Delta s^2_{k+1/2}}{\sqrt{(.)^2 + (.)^2}}$$

$$n_k^{(y)} = +\frac{\Delta x_{k+1/2} \cdot \Delta s^2_{k-1/2} + \Delta x_{k-1/2} \cdot \Delta s^2_{k+1/2}}{\sqrt{(.)^2 + (.)^2}}$$

there is no singularity when curvature $1/r_k \to 0$


cubic, true curve distance
N_{NW}
N_{NE}
N_{SE}
1
N
$N+1$
quintic, true distance
N_{NW}
N_{NE}
N_{SE}
1
N
$N+1$


cubic, index coord
1
N_SE
N_NE
N_NW
N
N+1
quintic, index coord.
1
N_SE
N_NE
N_NW
N
N+1

Conformal mapping: Why do we need it?

At this moment curvilinear grid contour is defined in sense that coefficients for all piecewise polynomials of which contour is comprised are known.

But the contour is not populated with grid-points yet.

Suppose we know number of grid points in each direction, L and M, and we populate the contour by distributing points uniformly along each side.

Then solve the Dirichlet problem

$$\frac{x_{i+1,j} - 2x_{i,j} + x_{i-1,j}}{\Delta\xi^2} + \frac{x_{i,j+1} - 2x_{i,j} + x_{i,j-1}}{\Delta\eta^2} = 0 \qquad \forall i,j \in [2,...,L-1] \times [2,...,M-1]$$

similar for $y_{i,j}$

...well, how do we know values of $\Delta\xi$ and $\Delta\eta$?

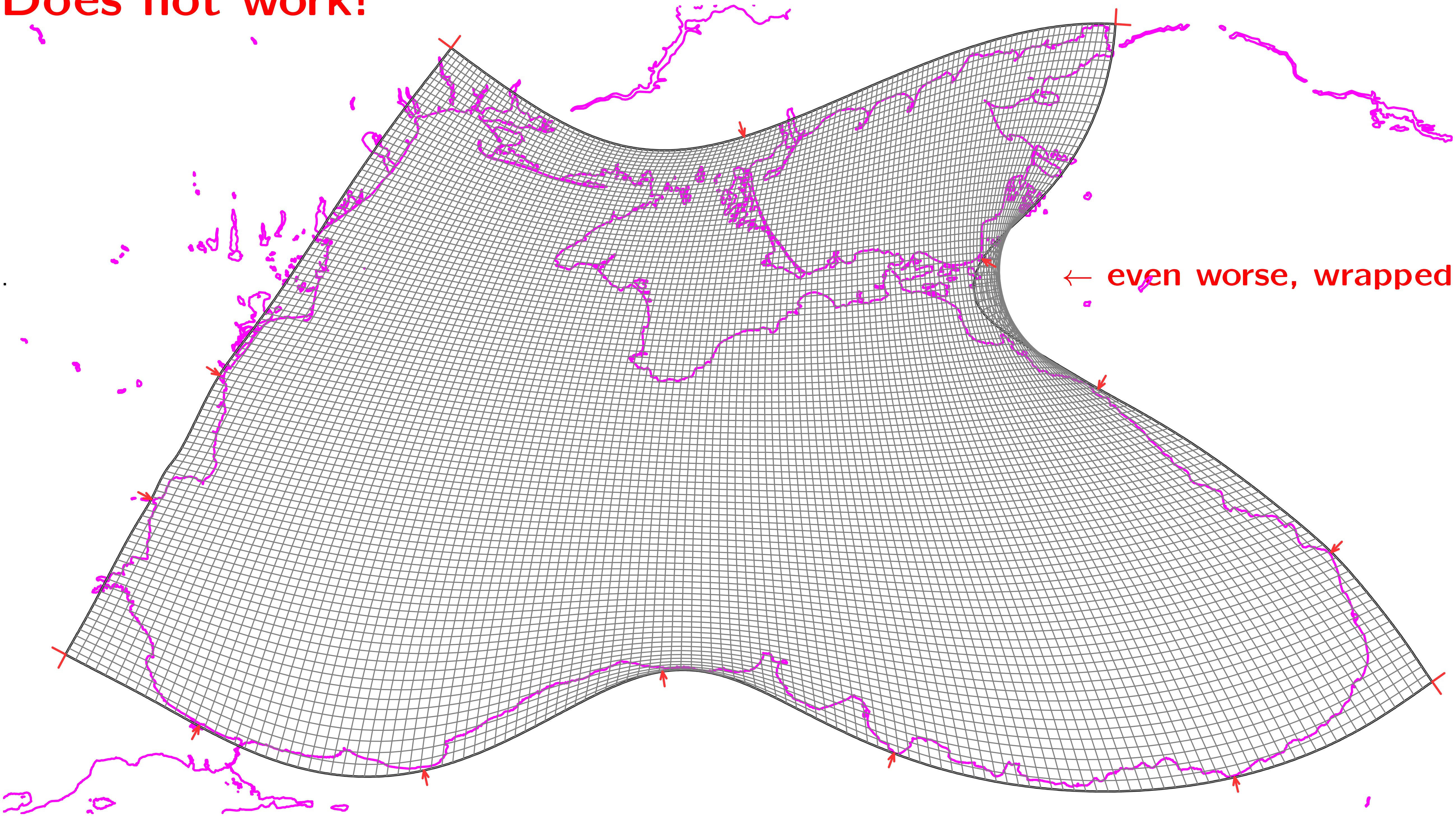
...obviously only their ratio $\Delta\xi/\Delta\eta$ matters.

...let's pretend that $\Delta\xi/\Delta\eta = 1$ and try it any way

in fact, we use L and M that yield *the correct* ratio $\Delta\xi/\Delta\eta$ (to be explained later),
but still we would not get it right

Does not work!

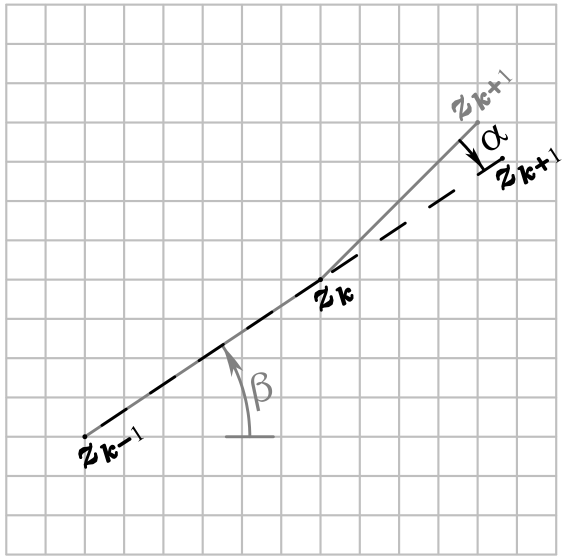
uniform grid-point distribution along each side ⇒ **not orthogonal!**



← **even worse, wrapped**

Conformal mapping: Discrete Schwartz–Christoffel transform explained

Elementary hinge transform in terms of absolute value and argument



Let $z = z_k + |z - z_k| \cdot e^{i\gamma}$ be an arbitrary point on complex plane.

Transform $z' = z_k + \dfrac{(z-z_k)^P}{(z_k - z_{k-1})^{P-1}}$ where $P = \dfrac{\pi}{\pi - \alpha}$, conversely, $\alpha = \pi \dfrac{P-1}{P}$,

hence $$z' - z_k = \frac{|z-z_k|^{\pi/(\pi-\alpha)}}{|z_k - z_{k-1}|^{\pi/(\pi-\alpha)-1}} \cdot \exp\Big\{ i \underbrace{\Big[\beta + \frac{\pi}{\pi-\alpha}\cdot(\gamma - \alpha - \beta)\Big]}_{\gamma'} \Big\}$$

$$\gamma' = \beta + \pi + P(\gamma - \beta - \pi) = \beta + \frac{\pi}{\pi - \alpha}\cdot(\gamma - \alpha - \beta)$$

turns $\angle z_{k-1}\, z_k\, z_{k+1}$ into straight line, while z_k and z_{k-1} do not move,

$$z_{k+1} - z_k = |z_{k+1} - z_k| \cdot e^{i(\alpha+\beta)} \qquad \gamma = \alpha + \beta \qquad \to \qquad \gamma' = \beta$$

$$z_{k-1} - z_k = |z_k - z_{k-1}| \cdot e^{i(\pi+\beta)} \qquad \gamma = \pi + \beta \qquad \to \qquad \gamma' = \pi + \beta$$

Inverse transform: replace $P \to 1/P$

for absolute value it is obvious.

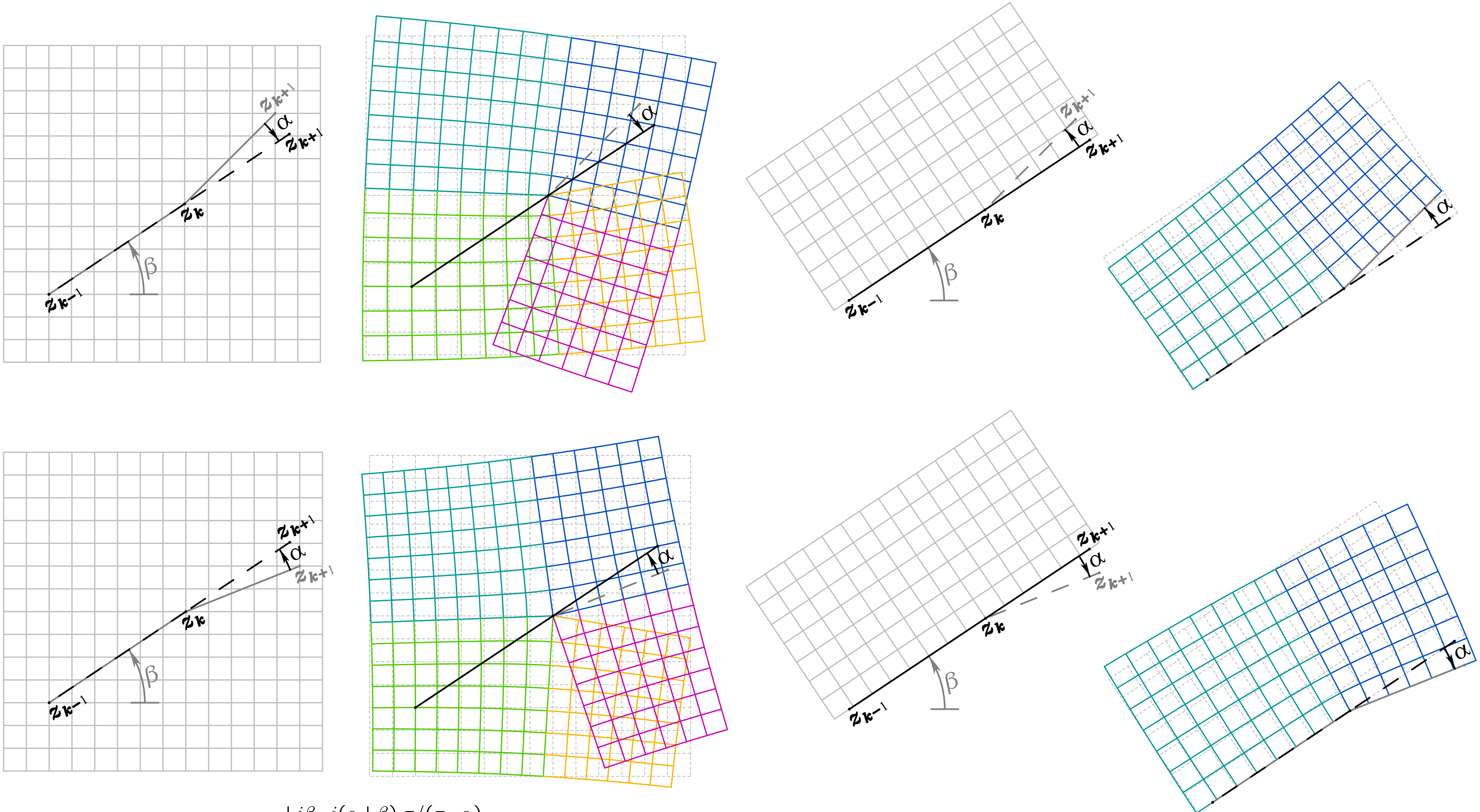
argument: apply both consecutively,
$$\begin{aligned} \gamma' &= \beta + \pi + P\cdot(\gamma - \beta - \pi) \\ \gamma'' &= \beta + \pi + \frac{1}{P}\cdot(\gamma' - \beta - \pi) \end{aligned}$$

substitute γ' from the first into the second yields $\gamma'' = \gamma$ for any P, β

Substitution $\gamma' = \beta$ yields $\gamma'' = \beta + \pi - \pi/P = \beta + \pi - (\pi - \alpha) = \alpha + \beta$,
i.e. direction β to z'_{k+1} rotates back to its original direction to z_{k+1}.

arguments are manipulated separately from absolute values

$\Rightarrow$ any ray, originating from the hinge point z_k, remains straight line



transform $z' = z_k + \frac{e^{+i\beta - i(\alpha+\beta)\cdot\pi/(\pi-\alpha)}}{|z_k - z_{k-1}|^{\pi/(\pi-\alpha)-1}} \cdot \left(z - z_k\right)^{\pi/(\pi-\alpha)}$ to straighten out $\angle z_{k-1}\, z_k\, z_{k+1}$ and its inverse

Straightening out a polyline: the transform function $z' = \mathscr{G}(z)$ is defined by a triplet of points z_{k-1}, z_k, z_{k+1}, therefore $z' = \mathscr{G}_{z_{k-1},z_k,z_{k+1}}(z)$

$z' = \mathscr{G}_{z_{k-1},z_k,z_{k+1}}(z)$ is analytical, except in its hinge point $z = z_k$

the hinge point z_k is **a branch point of a multi-valued function** $z' = \mathscr{G}_{z_{k-1},z_k,z_{k+1}}(z)$, therefore the transform cannot be applied to the entire plane, but only to a single-connected part of the plane on one side from the angle $\angle z_{k-1}, z_k, z_{k+1}$

a tracking algorithm is required to make sure that **the argument changes continuously within the selected semi-plane** (jumps of argument by 2π are not acceptable)

any ray originated from the hinge point z_k remains straight line $\Rightarrow$ if the transform is applied recursively to straighten out vertices of a polyline, defined by $\{z_1, z_2, ..., z_N\}$,

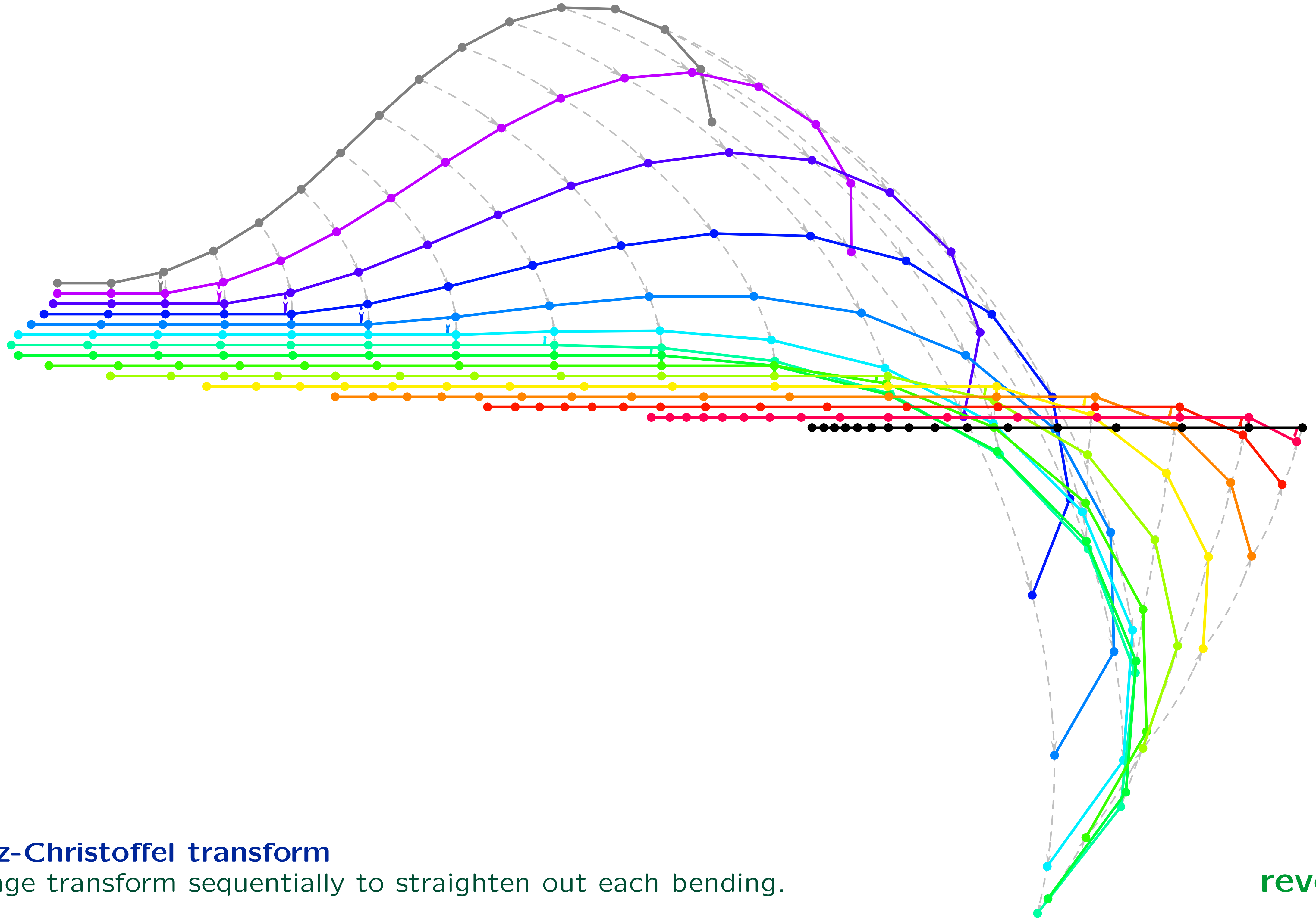
$$z^* = \underbrace{\mathscr{G}_{z^{xx}_{N-2},z^{xx}_{N-1},z^{xx}_N}\left\{\ldots\ \mathscr{G}_{z^{iv}_{k-3},z^{iv}_{k+2},z^{iv}_{k+3}}\left[\mathscr{G}_{z'''_{k-2},z'''_{k+1},z'''_{k+2}}\left(\mathscr{G}_{z''_{k-1},z''_k,z''_{k+1}}\left\{\ldots\ \mathscr{G}_{z'_2,z'_3,z'_4}\left[\mathscr{G}_{z_1,z_2,z_3}(z)\right]\ \ldots\right\}\right)\right]\ \ldots\right\}}_{\text{nested } N-2 \text{ times}}$$

then, at each stage, **the vertices, which already been straightened out, remain on straight.**

$$
\begin{array}{ll}
\texttt{do}\ k = 2, N-1 & \\
\quad \mathscr{G}() = \mathscr{G}_{z_{k-1},z_k,z_{k+1}}() & \leftarrow \text{compute } k\text{-dependent, } j\text{-invariant part of } \mathscr{G}_{z_{k-1},z_k,z_{k+1}}() \\
\quad \texttt{do}\ j = k-1, 1, -1 & \leftarrow \text{data dependency because of tracking} \\
\quad\quad z'_j = \mathscr{G}(z_j) & \\
\quad \texttt{enddo} & \\
\quad \texttt{do}\ j = k+1, N, +1 & \leftarrow \text{data dependency because of tracking} \\
\quad\quad z'_j = \mathscr{G}(z_j) & \\
\quad \texttt{enddo} & \\
\quad z_j = z'_j \quad \forall j = 1:N & \\
\texttt{enddo} &
\end{array}
$$

work from hinge point outward in both directions

$\mathscr{O}\left(N^2\right)$ operations required

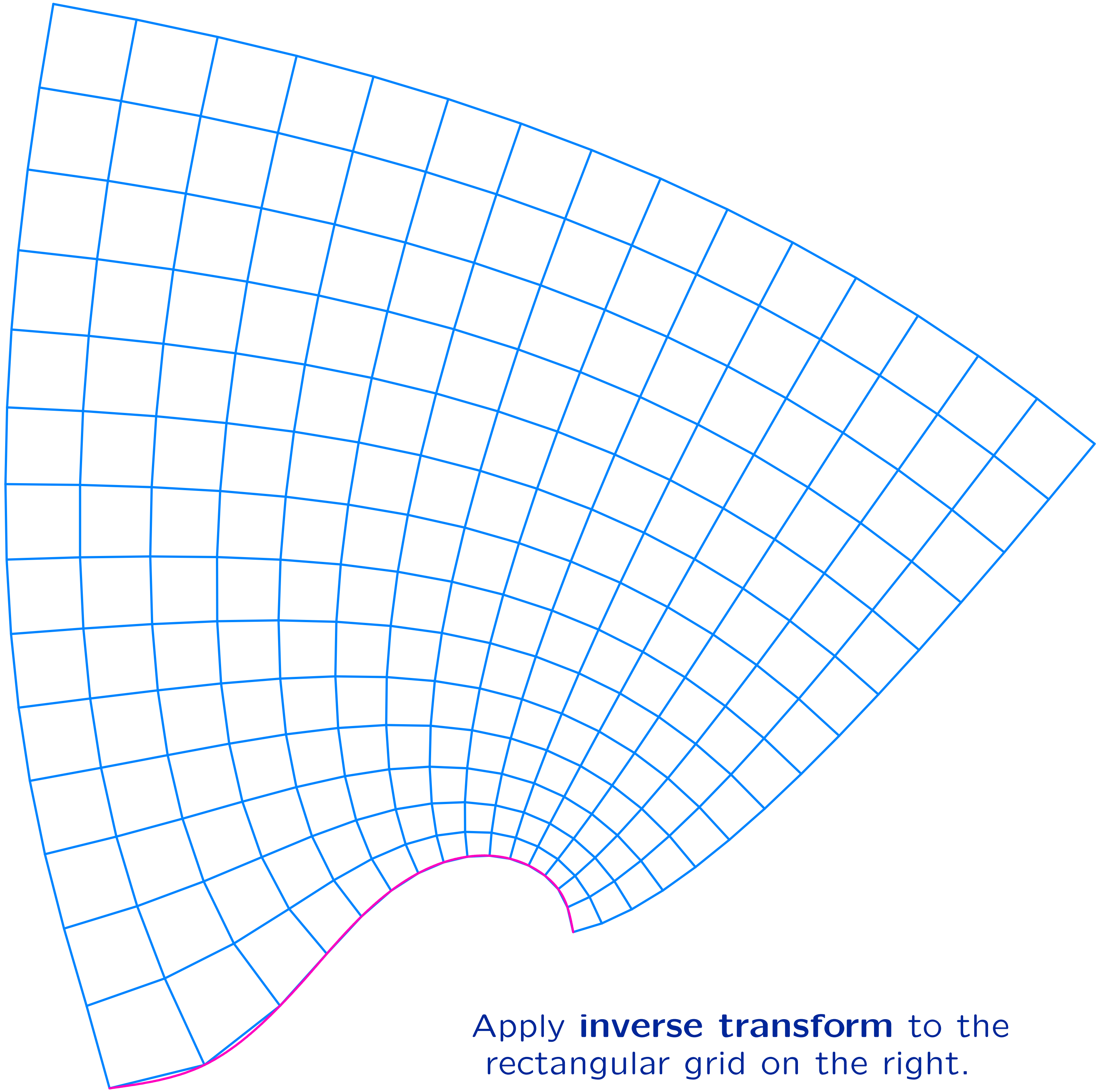


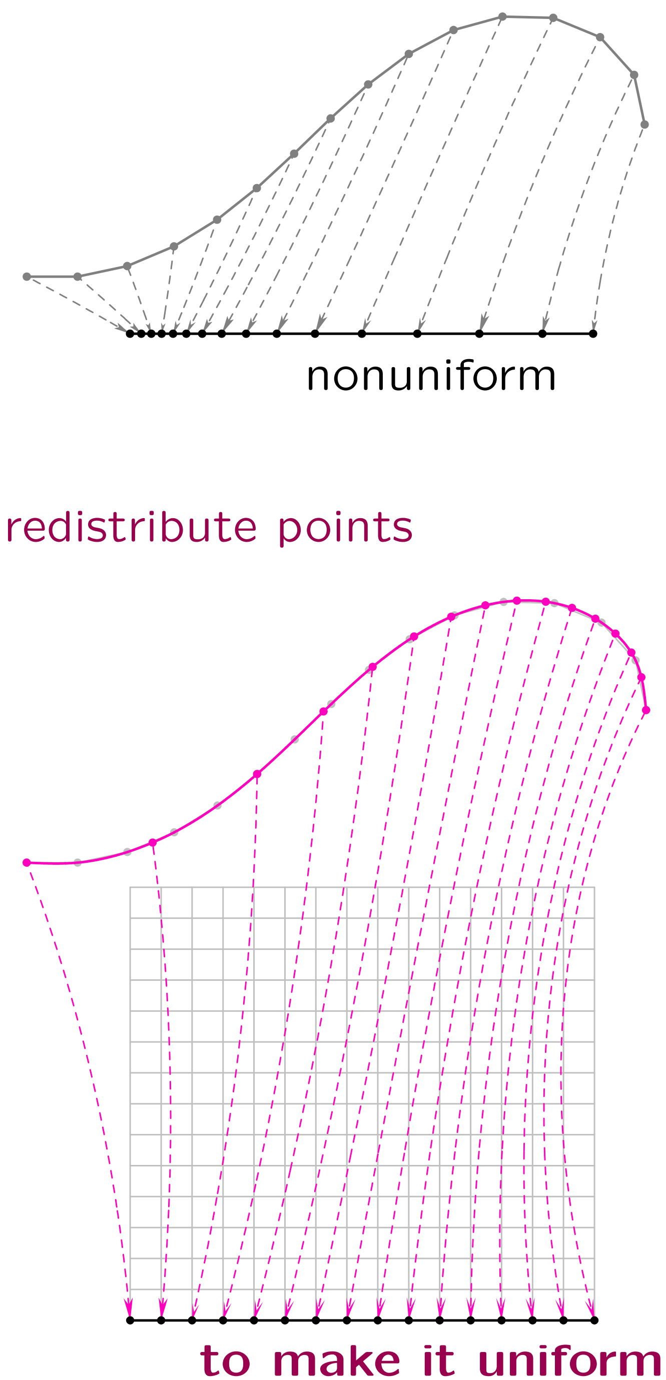
Schwartz-Christoffel transform

Apply hinge transform sequentially to straighten out each bending.

reversible!

Conformal grid by inverse mapping: interpret polyline as curve, build spline...





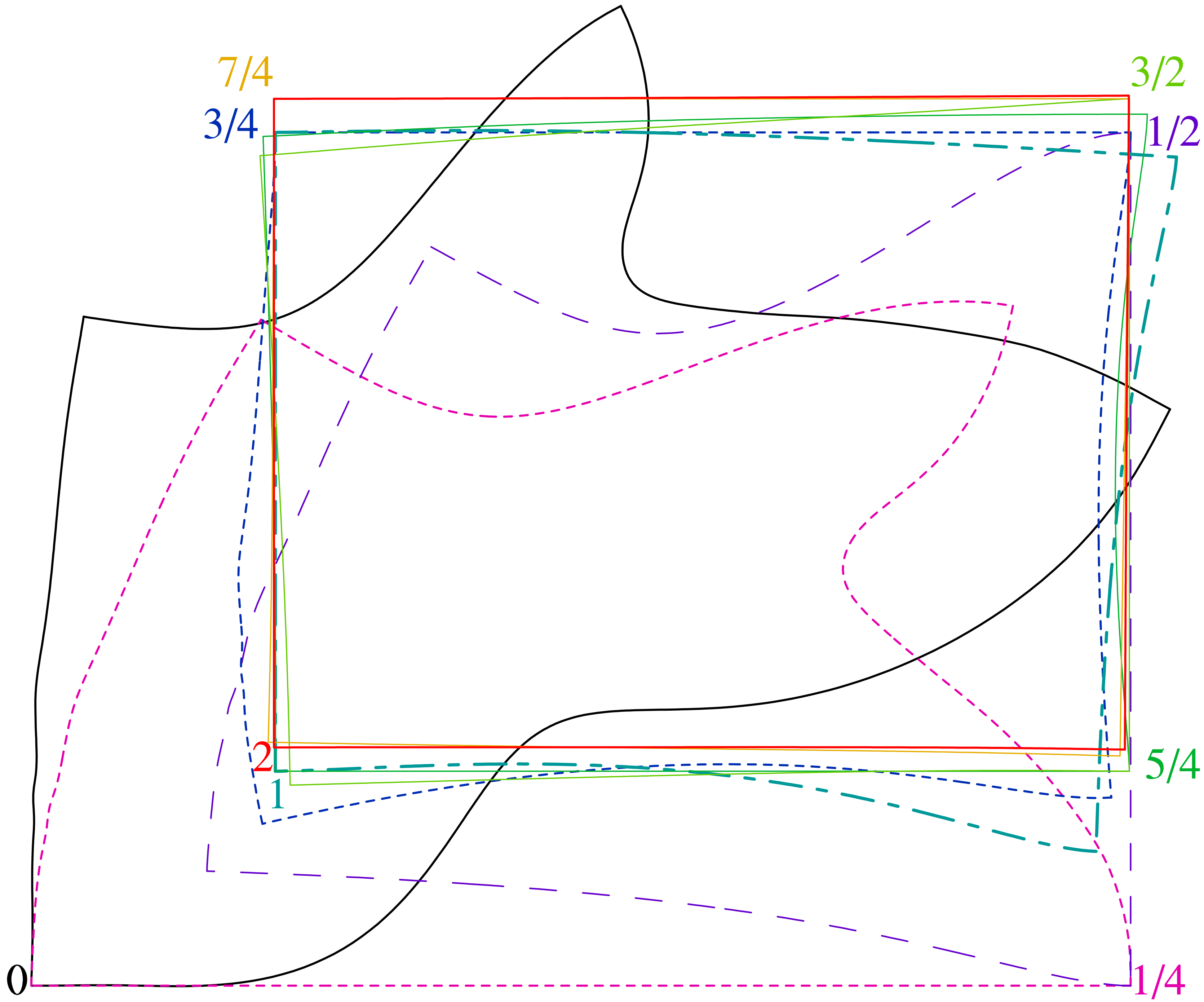
Apply **inverse transform** to the rectangular grid on the right.

```
do mtr=1,nmtrs
  z_j = spline(s_j)    ∀j = 1 : N    ← spline interpolation
  do k = 2, N − 1
    𝒢() = 𝒢_{z_{k−1},z_k,z_{k+1}}()    ← compute k-dependent, j-invariant part of 𝒢_{z_{k−1},z_k,z_{k+1}}()
    do j = k − 1, 1, −1    ← data dependency because of tracking
      z'_j = 𝒢(z_j)
    enddo
    do j = k + 1, N, +1    ← data dependency because of tracking
      z'_j = 𝒢(z_j)
    enddo
    z_j = z'_j    ∀j = 1 : N
  enddo
  s_j = S(z'_j)    ∀j = 1 : N    ← adjust {s_j} to make {z'_j} equidistant
enddo
```

$\mathscr{O}\left(N^2 \cdot \texttt{nmtrs}\right)$ operations

Conformal mapping of curvilinear rectangular contour onto rectangle

algorithm of Ives & Zacharias **needs iterations ⇒ basically irreversible**



```
do mtr=1,nmtrs
  z_j = spline(s_j)   ∀j = 1 : N        ← spline interpolation
  do itr=1,nitrs                          ← iteration loop of Ives and Zacharias
    do k = 1,N                              → compute k-dependent, j-invariant part of 𝒢_{z_{k-1},z_k,z_{k+1}}()
      set up 𝒢() = 𝒢_{z_{k-1},z_k,z_{k+1}}()   ← fold k − 1 and k + 1 into ∈ [1 : N] via periodicity
      if (k == corner) then
        special treatment
        for corner points
      endif
      do j' = 1,N − 1          ← data dependency because of tracking
        j = k + j'
        if (j > N) j = j − N    ← fold j into ∈ [1 : N] via periodicity
        z'_j = 𝒢(z_j)
      enddo
      z_j = z'_j   ∀j = 1 : N
    enddo
  enddo
  adjust L number of grid points in
  ξ-direction to match aspect ratio
  N = 2 · (L + M − 2)
  s_j = S(z_j),  ∀j = 1 : N        ← recompute {s_j} to make transformed {z_j}
enddo                                   equidistant on each side of the rectangle
```

$N = 2 \cdot (L + M - 2)$ number of grid points on perimeter

two-level nested iterations $\Rightarrow \mathcal{O}\left(N^2 \cdot \texttt{nitrs} \cdot \texttt{nmtrs}\right)$ operations

to mitigate computational cost set $\texttt{nitrs} = \mathrm{nint}\left(\sqrt{2^{\texttt{mtr}+5}}\right)$

in practice $\texttt{nmtrs} \sim 10$

for each z_k, tracking consist of computing argument, initially defined $\in [-\pi : \pi]$, but an arbitrary integer number of 2πs may be added, to make sure that it changes continuously when going from z_{k-1} to z_k, and only after that it is multiplied by $\pi/(\pi - \alpha_k)$ to straighten out $\angle z_{k-1}, z_k, z_{k+1}$

```
do mtr=1,nmtrs
  (x_j, y_j) = spline(s_j) ∀ j = 1 : N          ← spline interpolation
  do itr=1,nitrs                                 ← iteration loop of Ives and Zacharias
    do k = 1, N                                  → compute k-dependent, j-invariant part of \mathscr{G}_{z_{k-1},z_k,z_{k+1}}()
      set up \mathscr{G}() = \mathscr{G}_{z_{k-1},z_k,z_{k+1}}()     ← fold k − 1 and k + 1 into ∈ [1 : N] via periodicity
      if (k == corner) then
        special treatment
        for corner points
      endif
      do j' = jstr, jend, +1                     ← data dependency because of tracking
        j = k + j' ; if (j > N) j = j − N        ← fold into ∈ [1 : N]
        (r_j, γ_j) = \mathscr{G}(x_j, y_j)       ← transform and express in terms of
      enddo                                        absolute value r and argument γ
!$OMP BARRIER
      reconcile tracking among the threads
!$OMP BARRIER
      do j' = jend, jstr, −1
        j = k + j' ; if (j > N) j = j − N        ← fold into ∈ [1 : N]
        (x_j, y_j) = \mathscr{C}(r_j, corrected γ_j)   ← switch from (r, γ)
      enddo                                              back to (x, y)
!$OMP BARRIER
    enddo
  enddo
  adjust L number of grid points in
  ξ-direction to match aspect ratio
  N = 2 · (L + M − 2)
  s_j = S(x_j, y_j) ∀ j = 1 : N    ← recompute {s_j} to make transformed {x_j, y_j}
!$OMP BARRIER                        equidistant on each side of the rectangle
enddo
```

the usual OpenMP preambula

```
ntrds=omp_get_num_threads()
trd=omp_get_thread_num()
chnksize=(N − 1+ntrds-1)/ntrds
jstr=1+trd*chnksize
jend=min((1+trd)*chnksize,N − 1)
```

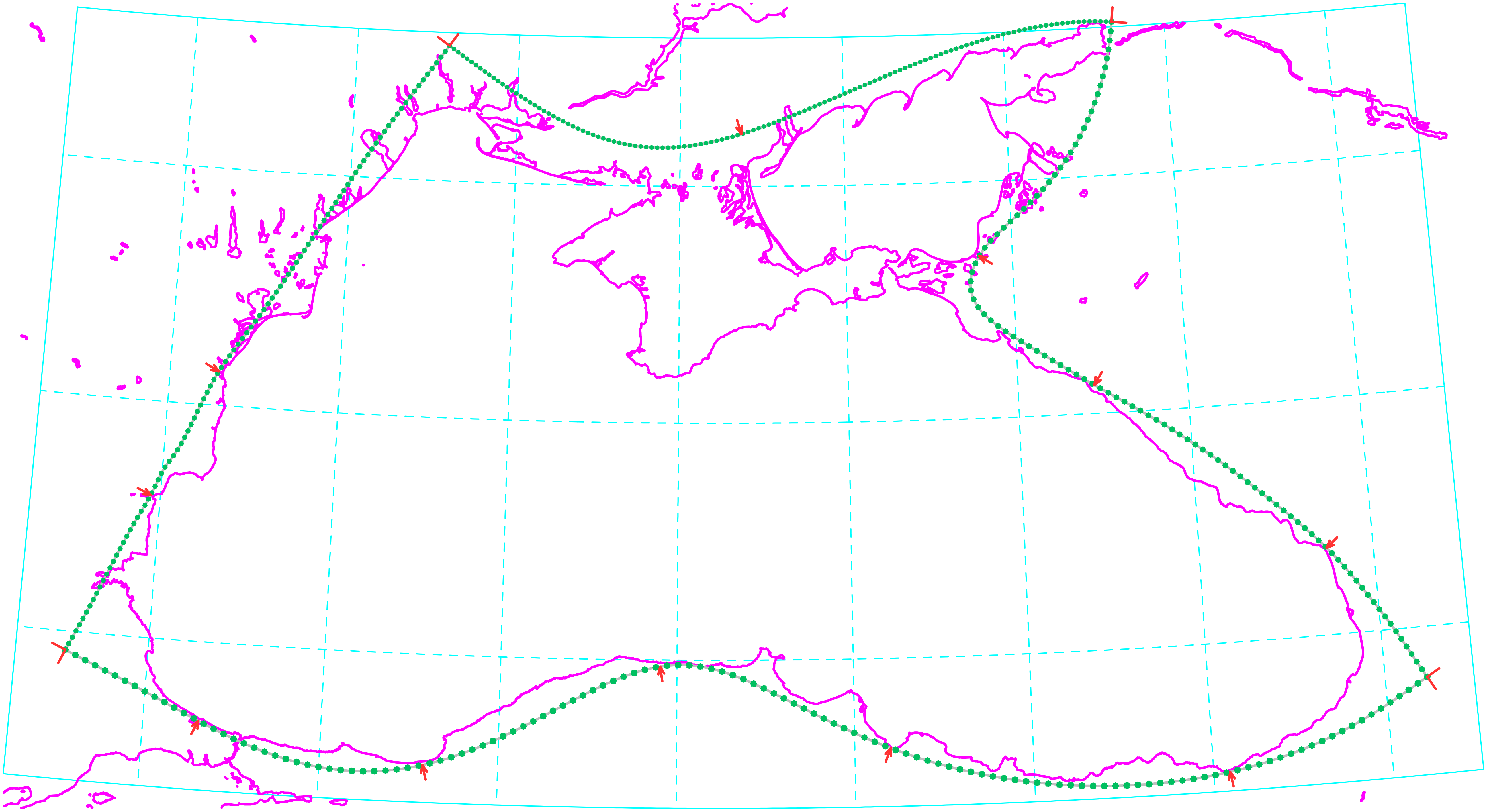
The sole purpose of conformal mapping of curvilinear contour onto rectangle is to *find such distribution of grid points along each side of the curvilinear contour, that the forward mapping turns it into uniform* along the side of the rectangle in transformed space.

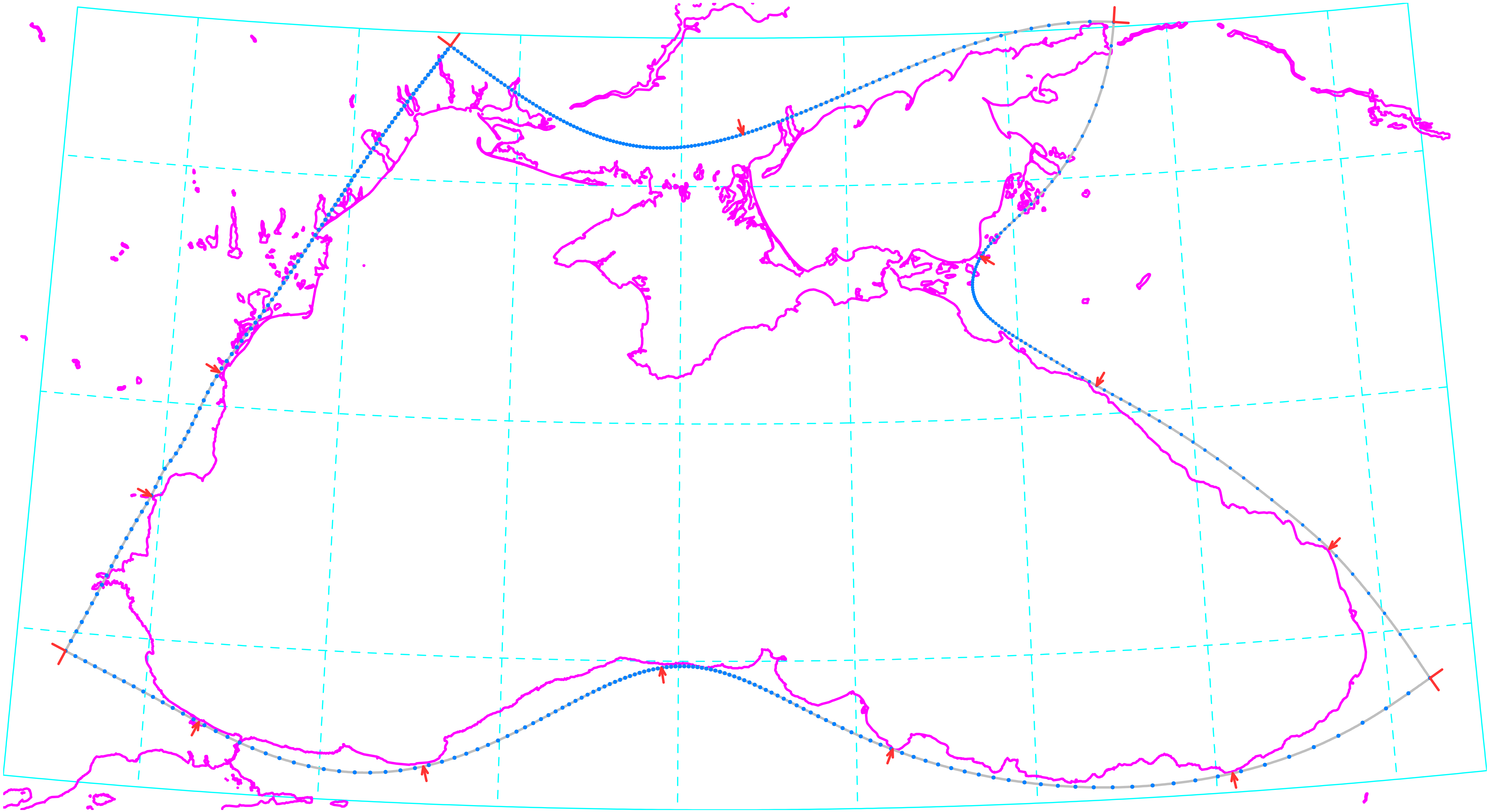
Placed an extra iterative loop around already iterative algorithm of Ives & Zacharias: starting with some distribution of points $(x_k, y_k) = \big(x(s_k), y(s_k)\big)$ along the contour [here s is coordinate along the contour; k is grid-point index], find the corresponding s'_k in the transformed space. On each side separately: because s'_k exists as a set of discrete values defining a smooth monotonic function, use interpolation to find a set σ_k such that $s'(\sigma_k) = k$. Use these σ_k to set new approximation of s_k and repeat the whole cycle again.

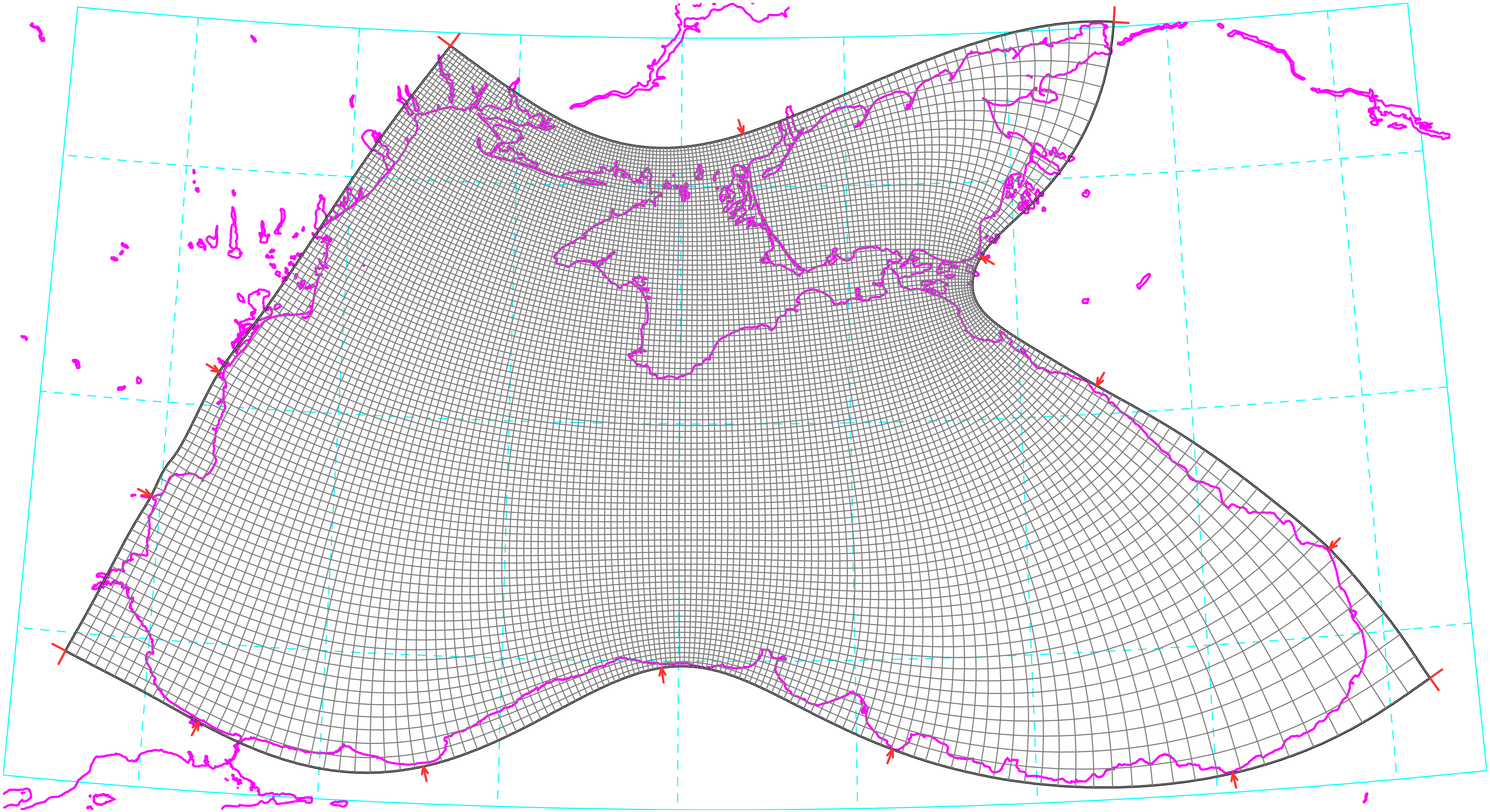
Iterate until convergence; roundoff-lever errors are achievable.

Optimize: set mutually dependent numbers of iterations in inner and outer loops to mitigate computational cost. Both are progressively increasing during the process.

User inputs only one number of grid points along one of the directions: the other will be determined by the code itself to make grid cells in the transformed space as square as possible.







Discretizing Laplace equation

We facing a very basic Dirichlet problem in a rectangular domain and with constant coefficients of elliptic operator.

L. Collatz, *The Numerical Treatment of Differential Equations* (1960) original **"mehrstellenverfahren"** idea:

Suppose we need to solve a Poisson equation, $\nabla^2 q = \frac{\partial q}{\partial \xi^2} + \frac{\partial q}{\partial \eta^2} = f$. If one constructs a second-order-accurate finite-difference operator $\mathscr{L}$, such that its *leading-order truncation term has form of Laplacian of Laplacian*

$$\mathscr{L} q = \nabla^2 q + C \cdot \nabla^2 \left[\nabla^2 q\right] + \mathscr{O}\left(\Delta\xi^4, \Delta\xi^2\Delta\eta^2, \Delta\eta^4\right), \qquad \text{where} \qquad C \sim \mathscr{O}\left(\Delta\xi^2, \Delta\xi\Delta\eta, \Delta\eta^2\right)$$

is just a coefficient. Then, noting that the inner Laplacian may be replaced as $\nabla^2\left[\nabla^2 q\right] \to \nabla^2 f$, introduce a *compensating term* into r.h.s.,

$$\mathscr{L} q = \nabla^2 q + C \cdot \nabla^2 \left[\nabla^2 q\right] + \mathscr{O}\left(\Delta\xi^4 ...\right) = f + C \cdot \nabla^2 f \equiv \mathscr{R} f$$

so the resultant solution of $\mathscr{L} q = \mathscr{R} f$ is fourth-order accurate approximation to the solution of $\nabla^2 q = f$.

$\mathscr{R} f$ is essentially a smoothing operator to compensate for truncation error of finite-difference Laplacian $\mathscr{L}$.

1D version of this idea is known as Boris Numerov method, (1923,1924), e.g.,
`http://gersoo.free.fr/Download/docs/numerov.pdf`

Same applies for Laplace equation, which is just a special case with r.h.s. $f = 0$.

however, making it Laplacian of Laplacian is possible only if $\Delta\xi = \Delta\eta$

A. Samarskii, *The Theory of Finite Difference Schemes* (1971 then 1977, 1989, 2001 multiple editions)
1971, in Russian, page 263, `http://samarskii.ru/books/book1971.pdf`
2001 English translation 2001, page 293, Eq (8), `http://samarskii.ru/books/book2001_2.pdf`
proposed an alternative factorization, making the whole idea **work for non-equal grid spacing**, $\Delta\xi \neq \Delta\eta$, as well,

$$\mathscr{L} q = \nabla^2 q + C \cdot \left(\Delta\xi^2 \cdot \frac{\partial^2}{\partial \xi^2} + \Delta\eta^2 \cdot \frac{\partial^2}{\partial \eta^2}\right)\left[\nabla^2 q\right] + \mathscr{O}\left(\Delta\xi^4, \Delta\xi^2\Delta\eta^2, \Delta\eta^4\right), \qquad \text{where} \qquad C = \mathscr{O}(1)$$

Indeed, consider a nine-point operator $\mathscr{L}_{\ast} = \begin{bmatrix} \gamma & \beta & \gamma \\ \alpha & \delta & \alpha \\ \gamma & \beta & \gamma \end{bmatrix}$ where $\delta = -2\alpha - 2\beta - 4\gamma$

$$\begin{aligned}
\mathscr{L}_{\ast} q\big|_{i,j} &= \alpha\left(q_{i+1,j} - 2q_{i,j} + q_{i-1,j}\right) + \beta\left(q_{i,j+1} - 2q_{i,j} + q_{i,j-1}\right) + \gamma\left(q_{i+1,j+1} + q_{i+1,j-1} + q_{i-1,j+1} + q_{i-1,j-1} - 4q_{i,j}\right) \\
&= \alpha\Delta\xi^2 \cdot \frac{\partial^2 q}{\partial \xi^2} + \beta\Delta\eta^2 \cdot \frac{\partial^2 q}{\partial \eta^2} + 2\gamma\left(\Delta\xi^2 \cdot \frac{\partial^2 q}{\partial \xi^2} + \Delta\eta^2 \cdot \frac{\partial^2 q}{\partial \eta^2}\right) \\
&\quad + 2\alpha \cdot \frac{\Delta\xi^4}{24} \cdot \frac{\partial^4 q}{\partial \xi^4} + 2\beta \cdot \frac{\Delta\eta^4}{24} \cdot \frac{\partial^4 q}{\partial \eta^4} + 4\gamma \cdot \left[\frac{\Delta\xi^4}{24} \cdot \frac{\partial^4 q}{\partial \xi^4} + \frac{\Delta\xi^2\Delta\eta^2}{4} \cdot \frac{\partial^4 q}{\partial \xi^2 \partial \eta^2} + \frac{\Delta\eta^4}{24} \cdot \frac{\partial^4 q}{\partial \eta^4}\right] + \ldots \\
&= \frac{\partial^2 q}{\partial \xi^2} + \frac{\partial^2 q}{\partial \eta^2} + \mathscr{O}\left(\Delta\xi^2\right) + \mathscr{O}\left(\Delta\xi\Delta\eta\right) + \mathscr{O}\left(\Delta\eta^2\right) .
\end{aligned}$$

to make it *second-order consistent*, one needs to set $\alpha = \frac{1}{\Delta\xi^2} - 2\gamma$ and $\beta = \frac{1}{\Delta\eta^2} - 2\gamma$ while γ remains unconstrained thus far. This leads to

$$\mathscr{L}_{\ast} q = \frac{\partial^2 q}{\partial \xi^2} + \frac{\partial^2 q}{\partial \eta^2} + \frac{\Delta\xi^2}{12} \cdot \frac{\partial^4 q}{\partial \xi^4} + \frac{\Delta\eta^2}{12} \cdot \frac{\partial^4 q}{\partial \eta^4} + \gamma \cdot \Delta\xi^2\Delta\eta^2 \cdot \frac{\partial^4 q}{\partial \xi^2 \partial \eta^2} + \ldots$$

where the terms with fourth-order derivatives can be factored into

$$\frac{1}{12}\left(\Delta\xi^2 \cdot \frac{\partial^2}{\partial \xi^2} + \Delta\eta^2 \cdot \frac{\partial^2}{\partial \eta^2}\right)\underbrace{\left(\frac{\partial^2 q}{\partial \xi^2} + \frac{\partial^2 q}{\partial \eta^2}\right)}_{[\nabla^2 q]} = \frac{\Delta\xi^2}{12} \cdot \frac{\partial^4 q}{\partial \xi^4} + \frac{\Delta\eta^2}{12} \cdot \frac{\partial^4 q}{\partial \eta^4} + \underbrace{\left(\frac{\Delta\xi^2}{12} + \frac{\Delta\eta^2}{12}\right)}_{=\gamma \cdot \Delta\xi^2\Delta\eta^2} \cdot \frac{\partial^4 q}{\partial \xi^2 \partial \eta^2}$$

if one chooses $\gamma = \frac{1}{12} \cdot \left(\frac{1}{\Delta\xi^2} + \frac{1}{\Delta\eta^2}\right)$.

Replacing $[\nabla^2 q] \to f$, and moving it to r.h.s.,

$$\begin{aligned}
\mathscr{L}_{\ast} q\Big|_{i,j} &= \frac{1}{6}\cdot\left(\frac{5}{\Delta\xi^2}-\frac{1}{\Delta\eta^2}\right)\cdot(q_{i+1,j}+q_{i-1,j}) + \frac{1}{6}\cdot\left(\frac{5}{\Delta\eta^2}-\frac{1}{\Delta\xi^2}\right)\cdot(q_{i,j+1}+q_{i,j-1}) \\
&\quad + \frac{1}{12}\cdot\left(\frac{1}{\Delta\xi^2}+\frac{1}{\Delta\eta^2}\right)\cdot(q_{i+1,j-1}+q_{i+1,j+1}+q_{i-1,j+1}+q_{i-1,j-1}) \\
&\quad - \frac{5}{3}\cdot\left(\frac{1}{\Delta\xi^2}+\frac{1}{\Delta\eta^2}\right)\cdot q_{i,j} \\
&= f_{i,j} + \frac{1}{12}\cdot\left(\Delta\xi^2\cdot\frac{\partial^2}{\partial\xi^2}+\Delta\eta^2\cdot\frac{\partial^2}{\partial\eta^2}\right) f\Big|_{i,j} = \mathscr{R} f\Big|_{i,j}
\end{aligned}$$

solution of which provides a fourth-order accurate approximation to the solution of $\nabla^2 q = f$, even if $\Delta\xi \neq \Delta\eta$;

imposes maximum limit of grid-spacing inequality, $1/\sqrt{5} < \Delta\xi/\Delta\eta < \sqrt{5}$

without loosing forth-order accuracy, operator $\mathscr{R}$ in r.h.s. can be replaced with its finite-difference analog, e.g.,

$$\mathscr{R} f\Big|_{i,j} = \frac{2}{3}\cdot f_{i,j} + \frac{1}{12}\cdot(f_{i+1,j}+f_{i-1,j}+f_{i,j+1}+f_{i,j-1})$$

$\mathscr{L}_{\ast}$ reverts back to the classical Collatz (1960) “mehrstellenverfahren”, if $\Delta\xi = \Delta\eta = \Delta h$

$$\mathscr{L}_{\ast} = \frac{1}{6}\cdot\frac{1}{\Delta h^2}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} \qquad \text{and, accordingly,} \qquad \mathscr{L}_{\ast} q = \nabla^2 q + \frac{\Delta h^2}{12}\cdot\nabla^4 q + \mathscr{O}\left(\Delta h^4\right)$$

Samarskii claims **six-order convergence** for Laplace equation, $f = 0$, in the case of equal grid spacing $\Delta\xi = \Delta\eta$.

indeed, checking out Samarskii's claim:

$$\mathscr{L}_{*}q = \frac{\partial^2 q}{\partial \xi^2} + \frac{\partial^2 q}{\partial \eta^2} + \left[\textbf{2nd-order terms}\right] + 2 \cdot \left(\frac{1}{\Delta\xi^2} - 2\gamma\right) \cdot \frac{\Delta\xi^6}{720} \cdot \frac{\partial^6 q}{\partial \xi^6} + 2 \cdot \left(\frac{1}{\Delta\eta^2} - 2\gamma\right) \cdot \frac{\Delta\eta^6}{720} \cdot \frac{\partial^6 q}{\partial \eta^6}$$

$$+\gamma \cdot \frac{1}{720} \sum_{\pm} \sum_{\pm} \left(\pm\Delta\xi \cdot \frac{\partial}{\partial \xi} \pm \Delta\eta \cdot \frac{\partial}{\partial \eta}\right)^6 q$$

$$= \nabla^2 q + [...] + 2(...)... + 2(...)...$$

$$+4\gamma \cdot \frac{1}{720} \left(\Delta\xi^6 \cdot \frac{\partial^6 q}{\partial \xi^6} + 15\Delta\xi^4\Delta\eta^2 \cdot \frac{\partial^6 q}{\partial \xi^4 \partial \eta^2} + 15\Delta\xi^2\Delta\eta^4 \cdot \frac{\partial^6 q}{\partial \xi^2 \partial \eta^4} + \Delta\eta^6 \cdot \frac{\partial^6 q}{\partial \eta^6}\right)$$

$$= \nabla^2 q + [...] + \frac{1}{360} \underbrace{\left(\Delta\xi^4 \cdot \frac{\partial^6 q}{\partial \xi^6} + 30\gamma\Delta\xi^4\Delta\eta^2 \cdot \frac{\partial^6 q}{\partial \xi^4 \partial \eta^2} + 30\gamma\Delta\xi^2\Delta\eta^4 \cdot \frac{\partial^6 q}{\partial \xi^2 \partial \eta^4} + \Delta\eta^4 \cdot \frac{\partial^6 q}{\partial \eta^6}\right)}_{= \left(\Delta\xi^4 \cdot \frac{\partial^4}{\partial \xi^4} + A\Delta\xi^2\Delta\eta^2 \cdot \frac{\partial^4}{\partial \xi^2 \partial \eta^2} + \Delta\eta^4 \cdot \frac{\partial^4}{\partial \eta^4}\right)\left(\frac{\partial^2}{\partial \xi^2} + \frac{\partial^2}{\partial \eta^2}\right) q \ ?}$$

Factorization is possible, if, simultaneously

$$\Delta\xi^4 + A\Delta\xi^2\Delta\eta^2 = 30\gamma\Delta\xi^4\Delta\eta^2 \quad \text{and} \quad A\Delta\xi^2\Delta\eta^2 + \Delta\eta^4 = 30\gamma\Delta\xi^2\Delta\eta^4$$

or

$$\Delta\xi^2 + A\Delta\eta^2 = 30\gamma\Delta\xi^2\Delta\eta^2 \quad \text{and} \quad A\Delta\xi^2 + \Delta\eta^2 = 30\gamma\Delta\xi^2\Delta\eta^2$$

where r.h.ss. are the same, so

$$\Delta\xi^2 + A\Delta\eta^2 = A\Delta\xi^2 + \Delta\eta^2$$

hence, in the case of $\Delta\xi \neq \Delta\eta$ one must set $A = 1$, which leads to $\gamma = (1/30) \cdot \left[1/\Delta\xi^2 + 1/\Delta\eta^2\right]$, which contradicts to the previously found $\gamma = (1/12) \cdot \left[1/\Delta\xi^2 + 1/\Delta\eta^2\right]$ needed to factorize **2nd-order terms**.

however, if $\Delta\xi = \Delta\eta = \Delta h$, then the two conditions merge into one, leading to $A = 30\gamma\Delta h^2 - 1, \ \forall\gamma$, which indicates that factorization is possible, and the scheme is **6th-order accurate for Laplace equation**.

it does not work for terms with 8th-order derivatives:

$$\mathscr{L}_{*}q = \frac{\partial^2 q}{\partial \xi^2} + \frac{\partial^2 q}{\partial \eta^2} + [\textbf{2nd and 4th-order terms}] + 2\cdot\left(\frac{1}{\Delta\xi^2} - 2\gamma\right)\cdot\frac{\Delta\xi^8}{8!}\cdot\frac{\partial^8 q}{\partial\xi^8} + 2\cdot\left(\frac{1}{\Delta\eta^2} - 2\gamma\right)\cdot\frac{\Delta\eta^8}{8!}\cdot\frac{\partial^8 q}{\partial\eta^8}$$
$$+\gamma\cdot\frac{1}{8!}\sum_{\pm}\sum_{\pm}\left(\pm\Delta\xi\cdot\frac{\partial}{\partial\xi}\pm\Delta\eta\cdot\frac{\partial}{\partial\eta}\right)^8 q$$

$$= \nabla^2 q + [...] + 2(...)... + 2(...)...$$

$$+4\gamma\cdot\frac{1}{8!}\left(\Delta\xi^8\cdot\frac{\partial^8 q}{\partial\xi^8} + 28\Delta\xi^6\Delta\eta^2\cdot\frac{\partial^8 q}{\partial\xi^6\partial\eta^2} + 70\Delta\xi^4\Delta\eta^4\cdot\frac{\partial^8 q}{\partial\xi^4\partial\eta^4} + 28\Delta\xi^2\Delta\eta^6\cdot\frac{\partial^8 q}{\partial\xi^2\partial\eta^6} + \Delta\eta^8\cdot\frac{\partial^8 q}{\partial\eta^8}\right)$$

$$= \nabla^2 q + [...] + \frac{2}{8!}\underbrace{\left(\Delta\xi^6\cdot\frac{\partial^8 q}{\partial\xi^8} + 56\gamma\Delta\xi^6\Delta\eta^2\cdot\frac{\partial^8 q}{\partial\xi^6\partial\eta^2} + 140\gamma\Delta\xi^4\Delta\eta^4\cdot\frac{\partial^8 q}{\partial\xi^4\partial\eta^4} + 56\gamma\Delta\xi^2\Delta\eta^6\cdot\frac{\partial^8 q}{\partial\xi^2\partial\eta^6} + \Delta\eta^6\cdot\frac{\partial^8 q}{\partial\eta^8}\right)}_{= \left(\Delta\xi^6\cdot\frac{\partial^6}{\partial\xi^6} + A\Delta\xi^4\Delta\eta^2\cdot\frac{\partial^6}{\partial\xi^4\partial\eta^2} + B\Delta\xi^2\Delta\eta^4\cdot\frac{\partial^6}{\partial\xi^2\partial\eta^4} + \Delta\eta^6\cdot\frac{\partial^6}{\partial\eta^6}\right)\left(\frac{\partial^2}{\partial\xi^2} + \frac{\partial^2}{\partial\eta^2}\right)q\ ?}$$

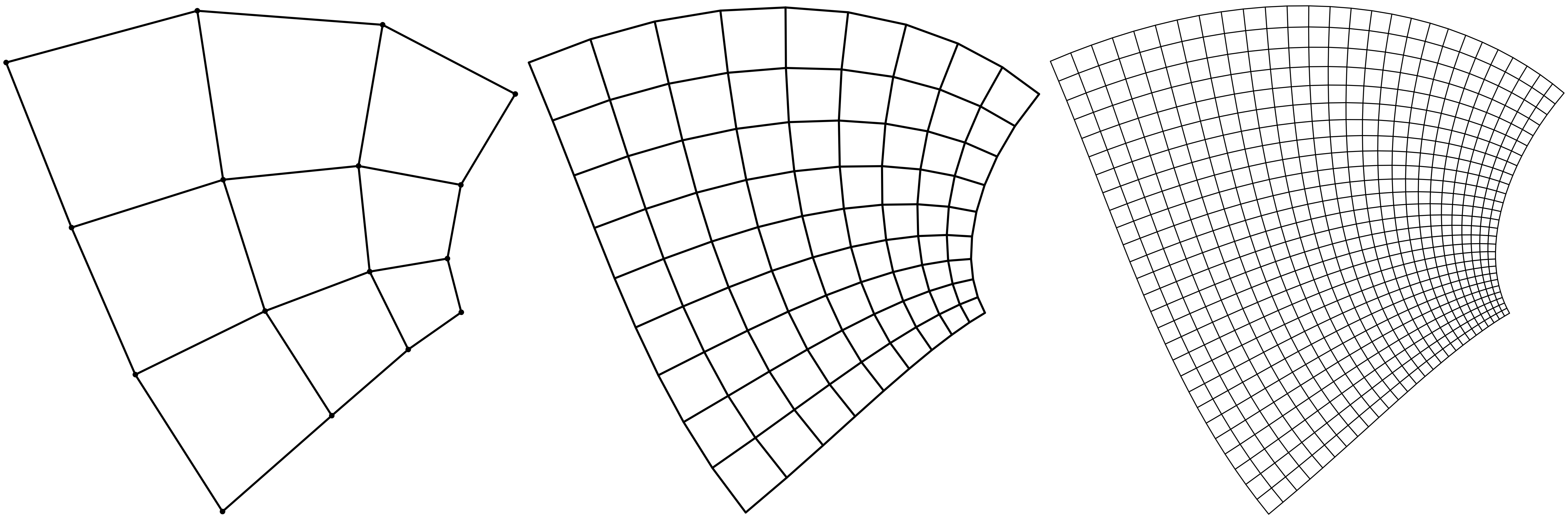
substitute $\Delta\xi = \Delta\eta = \Delta h$ and $\gamma = 1/(6\Delta h^2) \Rightarrow$ still cannot be factored,

$$\frac{\partial^8 q}{\partial\xi^8} + \frac{28}{3}\cdot\frac{\partial^8 q}{\partial\xi^6\partial\eta^2} + \frac{70}{3}\cdot\frac{\partial^8 q}{\partial\xi^4\partial\eta^4} + \frac{28}{3}\cdot\frac{\partial^8 q}{\partial\xi^2\partial\eta^6} + \frac{\partial^8 q}{\partial\eta^8} \neq \left(\frac{\partial^6}{\partial\xi^6} + A\cdot\frac{\partial^6}{\partial\xi^4\partial\eta^2} + B\cdot\frac{\partial^6}{\partial\xi^2\partial\eta^4} + \frac{\partial^6}{\partial\eta^6}\right)\left(\frac{\partial^2 q}{\partial\xi^2} + \frac{\partial^2 q}{\partial\eta^2}\right)$$

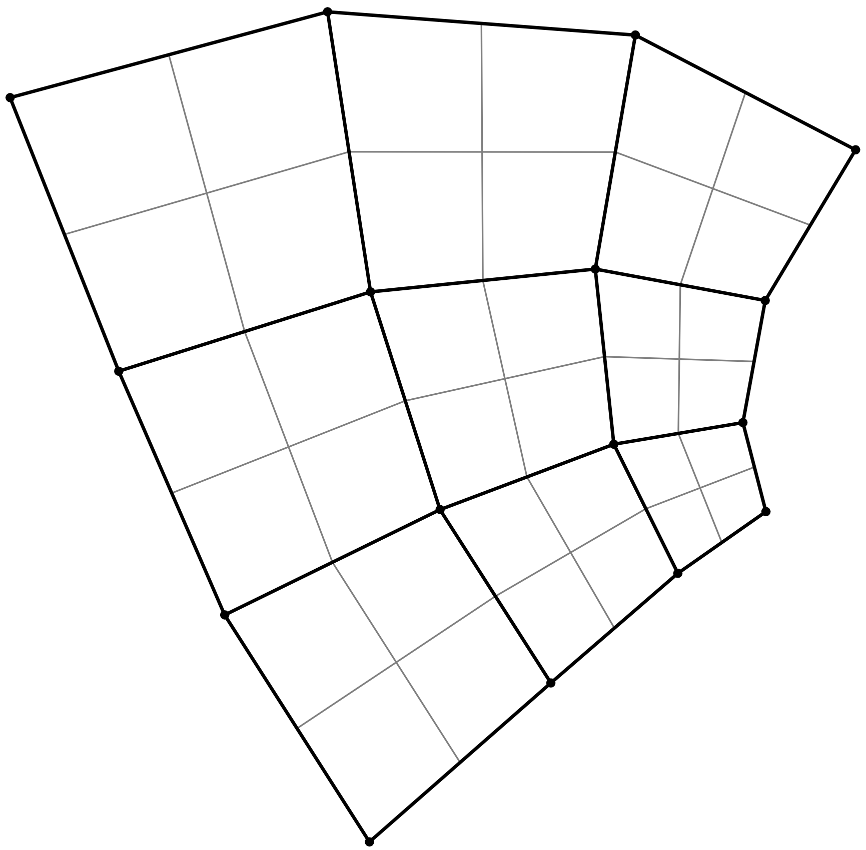
three conditions must be satisfied, $A + 1 = 28/3$, $A + B = 70/3$, and $1 + B = 28/3$. No solution.

How do we know that they are orthogonal?

Need a discrete criterion to measure orthogonality errors.



These grids are generated by an analytical transform. They are meant to be exactly orthogonal by the construction, but none of the angles here seem to be equal to 90°. **?**



midpoint orthogonality criterion: vectors connecting midpoints of the opposite sides of grid cell $(x,y)_{i,j}$, $(x,y)_{i+1,j}$, $(x,y)_{i+1,j+1}$, $(x,y)_{i,j+1}$

$$\boldsymbol{\ell}_\xi = \left(\Delta_\xi x, \Delta_\xi y\right) \quad \text{and} \quad \boldsymbol{\ell}_\eta = \left(\Delta_\eta x, \Delta_\eta y\right)$$

where

$$\Delta_\xi x = \left(x_{i+1,j} + x_{i+1,j+1}\right)/2 - \left(x_{i,j} + x_{i,j+1}\right)/2$$
$$\Delta_\xi y = \left(y_{i+1,j} + y_{i+1,j+1}\right)/2 - \left(y_{i,j} + y_{i,j+1}\right)/2$$
$$\Delta_\eta x = \left(x_{i,j+1} + x_{i+1,j+1}\right)/2 - \left(x_{i,j} + x_{i+1,j}\right)/2$$
$$\Delta_\eta y = \left(y_{i,j+1} + y_{i+1,j+1}\right)/2 - \left(y_{i,j} + y_{i+1,j}\right)/2$$

are orthogonal to each other if

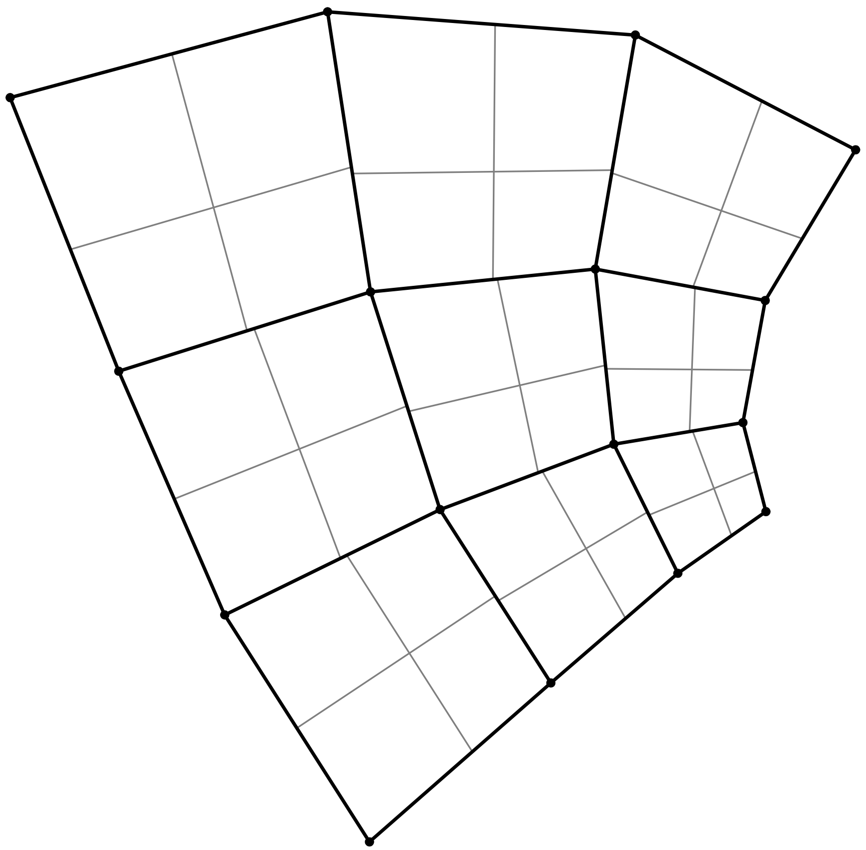
$$(\boldsymbol{\ell}_\xi \cdot \boldsymbol{\ell}_\eta) = \Delta_\xi x \cdot \Delta_\eta x + \Delta_\xi y \cdot \Delta_\eta y = 0$$

hence, the orthogonality error measure

$$\epsilon = \sin\left(\angle \boldsymbol{\ell}_\xi \boldsymbol{\ell}_\eta - \frac{\pi}{2}\right) = \frac{(\boldsymbol{\ell}_\xi \cdot \boldsymbol{\ell}_\eta)}{|\boldsymbol{\ell}_\xi| \cdot |\boldsymbol{\ell}_\eta|} = \frac{\Delta_\xi x \cdot \Delta_\eta x + \Delta_\xi y \cdot \Delta_\eta y}{\sqrt{\left(\Delta_\xi x^2 + \Delta_\xi y^2\right) \cdot \left(\Delta_\eta x^2 + \Delta_\eta y^2\right)}}$$

this criterion is equivalent to having diagonals equal to each other

$$\begin{aligned}(\boldsymbol{\ell}_\xi \cdot \boldsymbol{\ell}_\eta) = \; & \left[\left(x_{i+1,j+1} - x_{i,j}\right)^2 + \left(y_{i+1,j+1} - y_{i,j}\right)^2\right] \\ - \; & \left[\left(x_{i,j+1} - x_{i+1,j}\right)^2 + \left(y_{i,j+1} - y_{i+1,j}\right)^2\right] \; = 0\end{aligned}$$



orthogonality criterion using **cross-directional interpolation biased toward smaller side:** same idea as above, but different definition of vectors $\boldsymbol{\ell}_\xi$, $\boldsymbol{\ell}_\eta$. Let

$$\Delta_\xi x_{i+1/2,j} = x_{i+1,j} - x_{i,j} \qquad \Delta_{\xi_{i+1/2,j}} = \sqrt{\left(\Delta_\xi x_{i+1/2,j}\right)^2 + \left(\Delta_\xi y_{i+1/2,j}\right)^2}$$
$$\Delta_\xi y_{i+1/2,j} = y_{i+1,j} - y_{i,j}$$

$$\Delta_\eta x_{i,j+1/2} = x_{i,j+1} - x_{i,j} \qquad \Delta_{\eta_{i,j+1/2}} = \sqrt{\left(\Delta_\eta x_{i,j+1/2}\right)^2 + \left(\Delta_\eta y_{i,j+1/2}\right)^2}$$
$$\Delta_\eta y_{i,j+1/2} = y_{i,j+1} - y_{i,j}$$

then

$$\Delta_\xi x_{i+1/2,j+1/2} = \frac{\Delta_{\xi_{i+1/2,j}} \cdot \Delta_\xi x_{i+1/2,j+1} + \Delta_{\xi_{i+1/2,j+1}} \cdot \Delta_\xi x_{i+1/2,j}}{\Delta_{\xi_{i+1/2,j}} + \Delta_{\xi_{i+1/2,j+1}}}$$

$$\Delta_\xi x_{i+1/2,j+1/2} = \frac{\Delta_{\xi_{i+1/2,j}} \cdot \Delta_\xi y_{i+1/2,j+1} + \Delta_{\xi_{i+1/2,j+1}} \cdot \Delta_\xi y_{i+1/2,j}}{\Delta_{\xi_{i+1/2,j}} + \Delta_{\xi_{i+1/2,j+1}}}$$

$$\Delta_\eta x_{i+1/2,j+1/2} = \frac{\Delta_{\eta_{i,j+1/2}} \cdot \Delta_\eta x_{i+1,j+1/2} + \Delta_{\eta_{i+1,j+1/2}} \cdot \Delta_\eta x_{i,j+1/2}}{\Delta_{\eta_{i,j+1/2}} + \Delta_{\eta_{i+1,j+1/2}}}$$

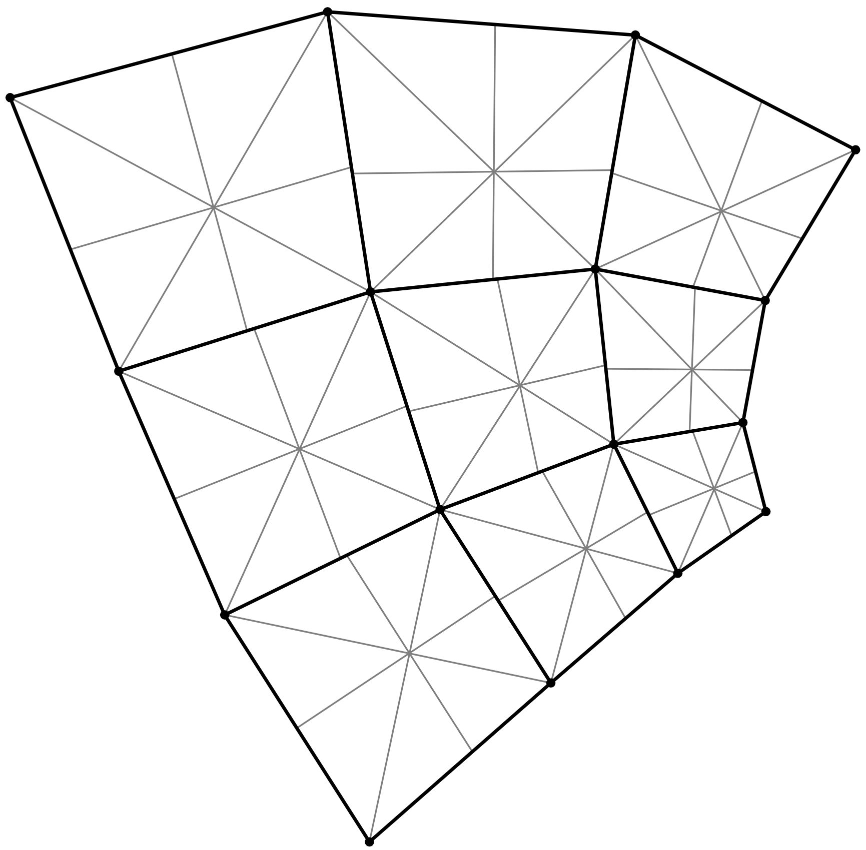
$$\Delta_\eta y_{i+1/2,j+1/2} = \frac{\Delta_{\eta_{i,j+1/2}} \cdot \Delta_\eta y_{i+1,j+1/2} + \Delta_{\eta_{i+1,j+1/2}} \cdot \Delta_\eta y_{i,j+1/2}}{\Delta_{\eta_{i,j+1/2}} + \Delta_{\eta_{i+1,j+1/2}}}$$

and vectors

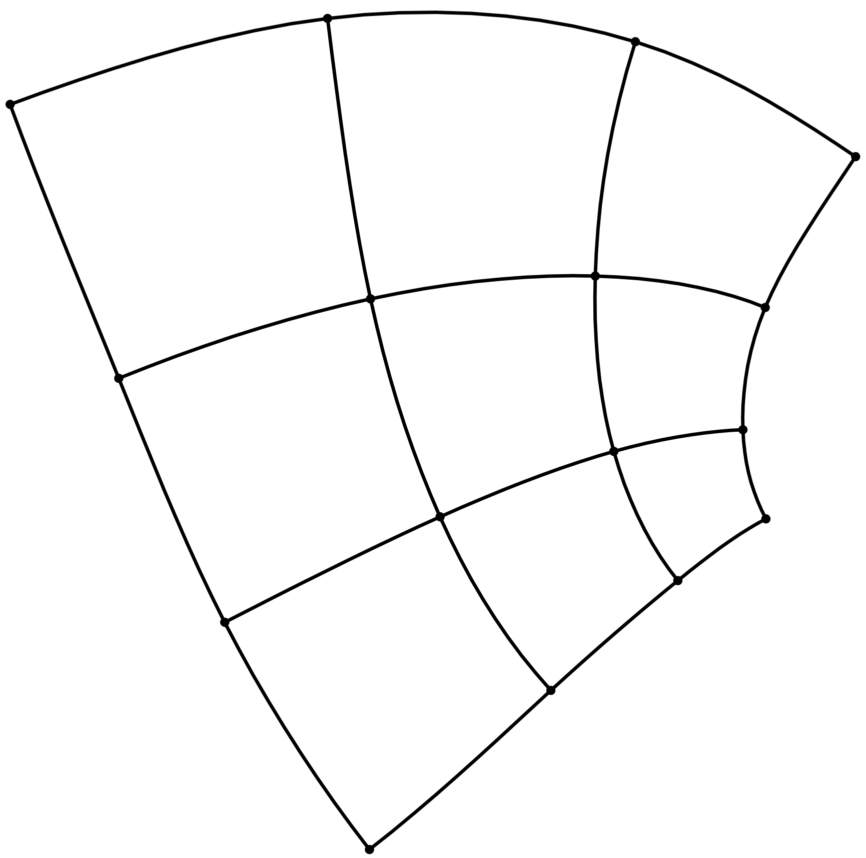
$$\boldsymbol{\ell}_\xi = \left(\Delta_\xi x_{i+1/2,j+1/2}, \Delta_\xi y_{i+1/2,j+1/2}\right) \quad \text{and} \quad \boldsymbol{\ell}_\eta = \left(\Delta_\eta x_{i+1/2,j+1/2}, \Delta_\eta y_{i+1/2,j+1/2}\right)$$

orthogonality error measure

$$\epsilon = \sin\left(\angle \boldsymbol{\ell}_\xi \boldsymbol{\ell}_\eta - \frac{\pi}{2}\right) = \frac{(\boldsymbol{\ell}_\xi \cdot \boldsymbol{\ell}_\eta)}{|\boldsymbol{\ell}_\xi| \cdot |\boldsymbol{\ell}_\eta|}$$



point of intersection of vectors $\boldsymbol{\ell}_\xi$ and $\boldsymbol{\ell}_\eta$ is very close to the point of intersection of diagonals, **but does not coincide with it.**



cubic spline orthogonality criterion: the idea is to compute all four sets of derivatives, $\partial x/\partial\xi$, $\partial x/\partial\eta$, $\partial y/\partial\xi$, $\partial y/\partial\eta$, at every point $(x_{i,j}, y_{i,j})$ of the grid, then define orthogonality error as

$$\epsilon = \sin\left(\angle \boldsymbol{\ell}_\xi \boldsymbol{\ell}_\eta - \frac{\pi}{2}\right) = \frac{\dfrac{\partial x}{\partial\xi}\cdot\dfrac{\partial x}{\partial\eta} + \dfrac{\partial y}{\partial\xi}\cdot\dfrac{\partial y}{\partial\eta}}{\sqrt{\left(\dfrac{\partial x}{\partial\xi}\right)^2 + \left(\dfrac{\partial y}{\partial\xi}\right)^2}\cdot\sqrt{\left(\dfrac{\partial x}{\partial\eta}\right)^2 + \left(\dfrac{\partial y}{\partial\eta}\right)^2}}$$

which is deviation of angles of intersection of curvilinear coordinate lines from 90° angle.

to compute derivatives use cubic spline algorithm:

(*i*) construct spline going through all the grid points along the perimeter of the grid, assuming exact 90° angles at its corners (the same algorithm as the one to construct grid contour from user-specified reference points);

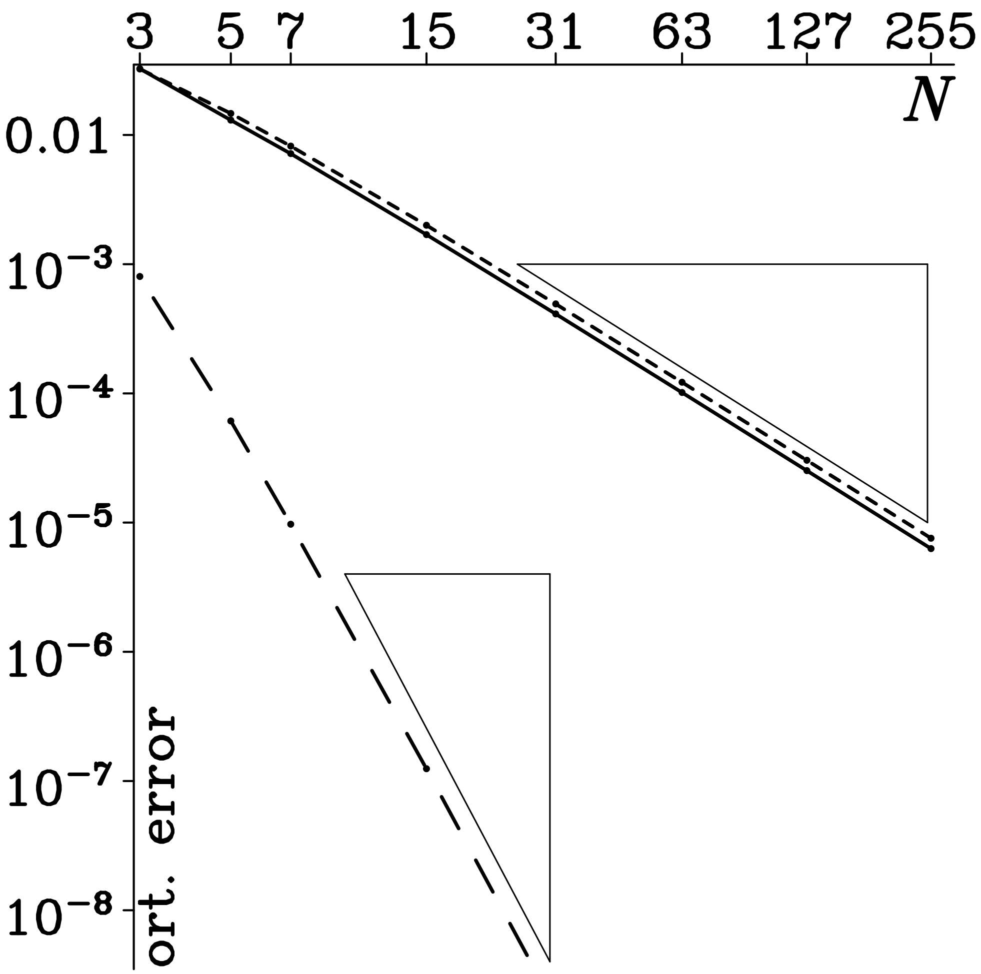
(*ii*) also compute second derivatives along the contour via spline;

(*iii*) construct splines for each coordinate line in both directions. Because $x_{\xi\xi} + x_{\eta\eta} = 0$, (same for y), assume b.c.s prescribing second derivative at both ends:
on western and eastern sides

$$\partial^2 x/\partial\xi^2\big|_{W,E} = -\partial^2 x/\partial\eta^2\big|_{contour} \qquad \text{(same for } y\text{)}$$

on southern and northern sides

$$\partial^2 x/\partial\eta^2\big|_{W,E} = -\partial^2 x/\partial\xi^2\big|_{contour} \qquad \text{(same for } y\text{)}$$



Convergence rates for the three criteria of orthogonality error in the case of analytically-generated orthogonal curvilinear grid:

short dashes – midpoint criterion
solid line – biased toward smaller side
`long dashes` – cubic spline

orthogonality error is the maximum over all grid points.

triangles indicate second and sixth-order convergence rate

this is actually test for the methods of measuring orthogonality error of a grid, rather than grid itself

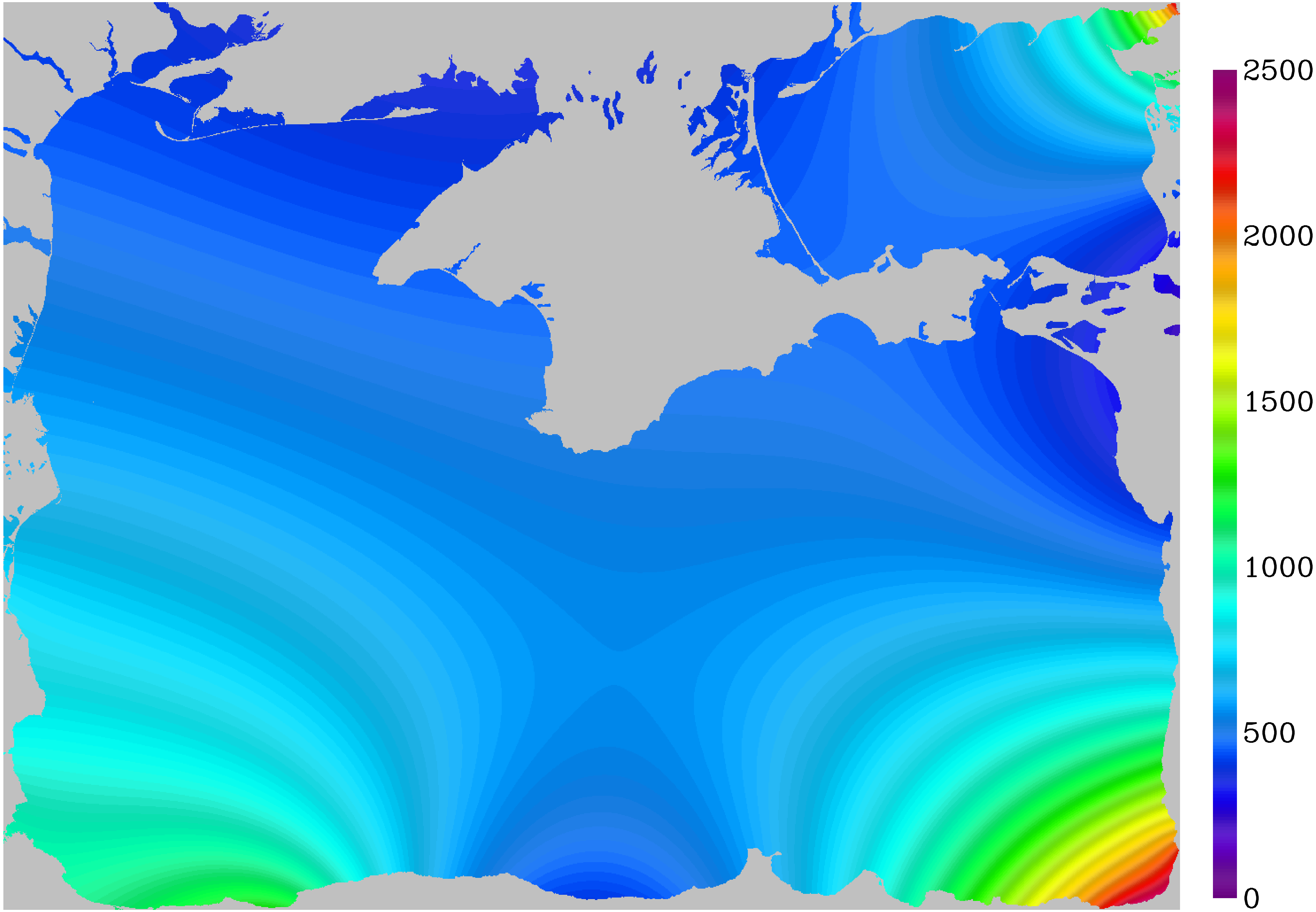
Practical, not-a-toy, Black sea grid

1329×1025 **points**

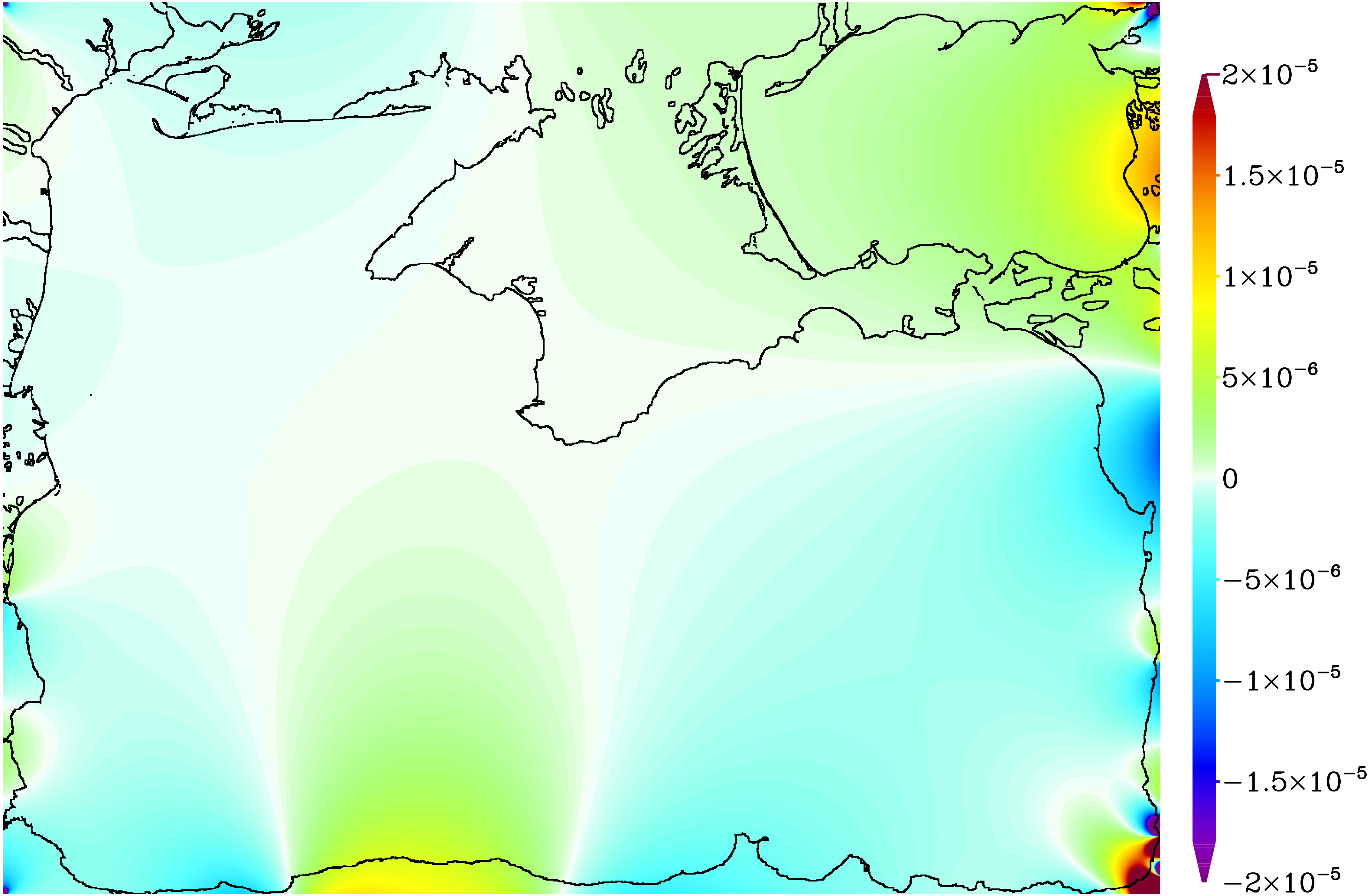
only 1 out of 8 grid lines is shown

```
mode=4 latlongrid=2 spline_type=5 npass=10
proj=ME rlat=43.75 rlon=34.5 rota=0
west_edge=26.5 east_edge=43
south_edge=40.75 north_edge=47.25
lwidth=1.  lonlat=0 rarefy=6

gshhs_data=h

nx=664, ny=512
uscale=0.001
---
-97   -30.5 south-west
-45   -46
-11   -32.
+25   -43.8
+75   -47
+104   -33 < south-east
+89.2 -14.0
+55   +10
+38   +29
+57.5 +63.5 < north-east
+3   +47
-40   +60 < north-west
-74 12
```

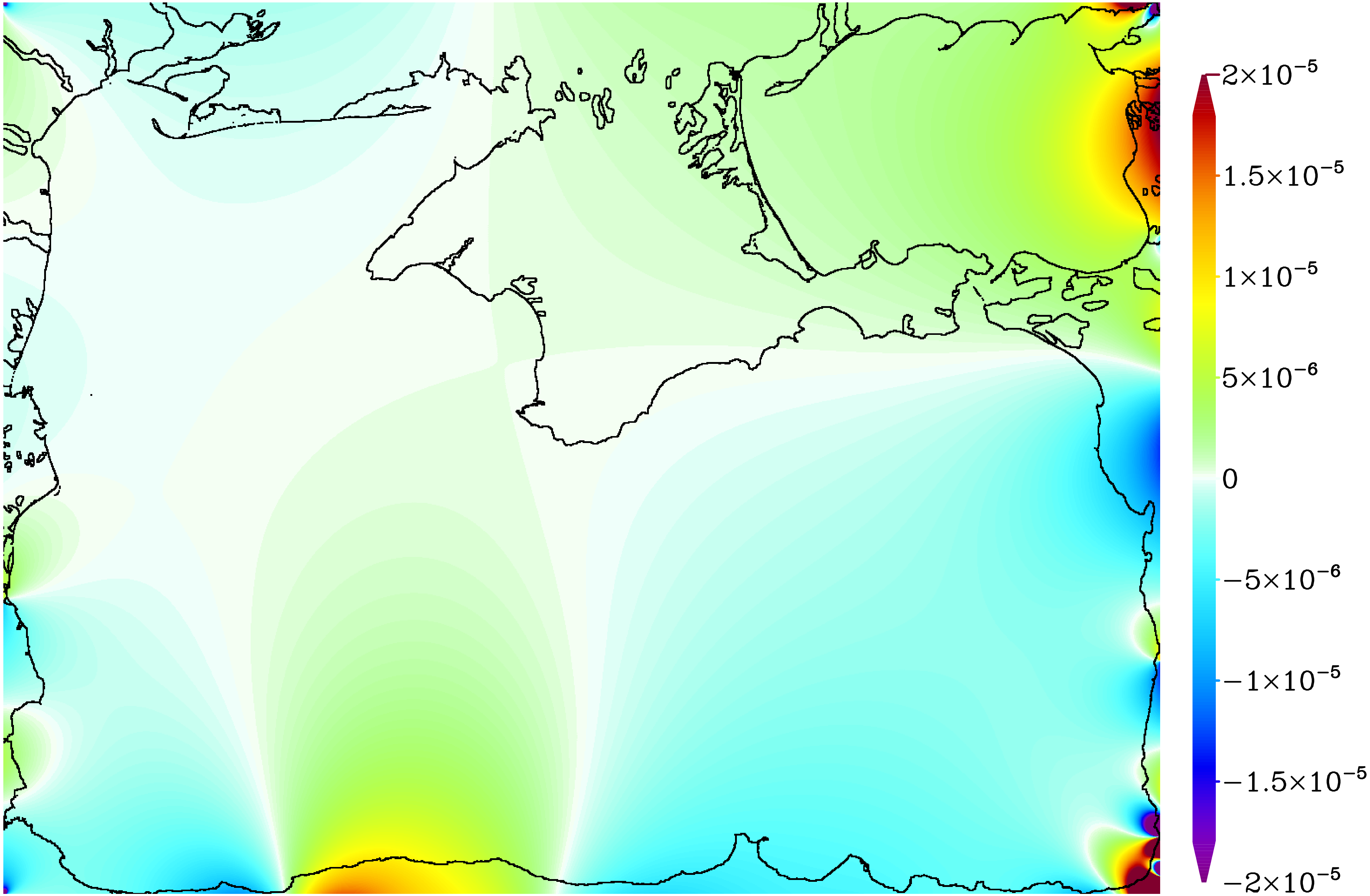
1329 × 1025 points, only 1 out of 8 grid lines is shown



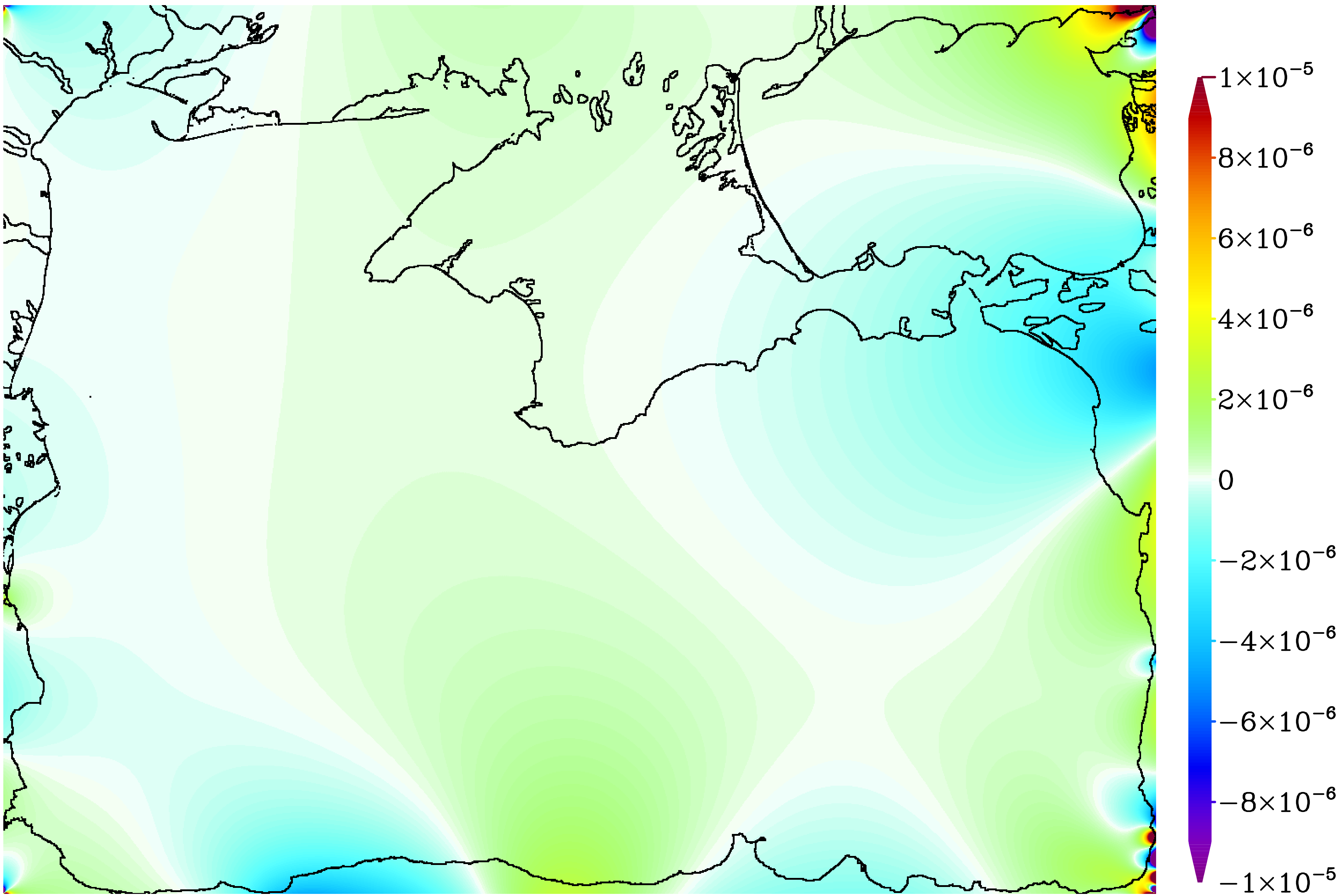
Transformed view with land mask. Grid dimensions 1329 × 1025 points. Grid spacing, *meters*.



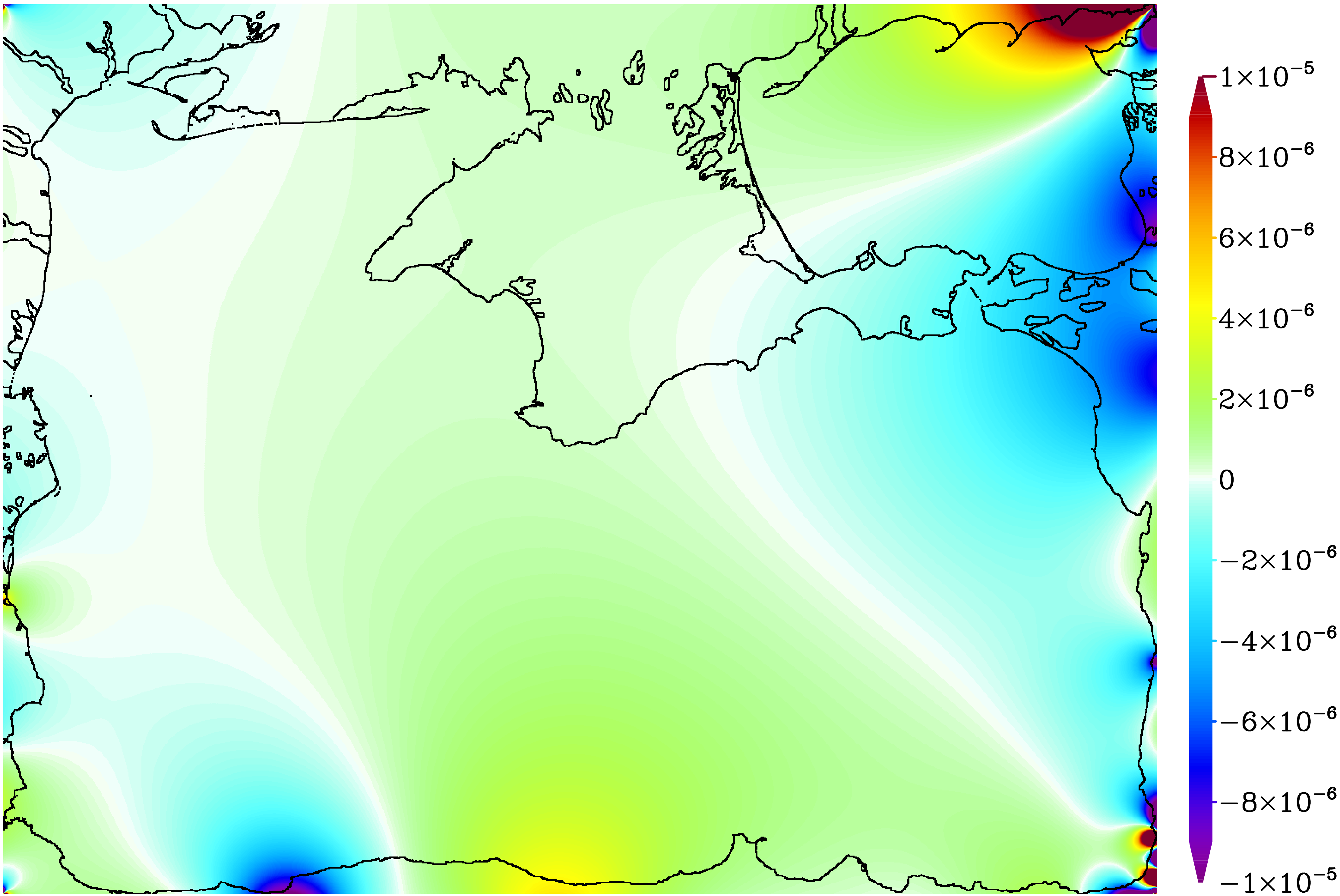
Black sea, 1329×1025 grid points, orthogonality error, radians



Black sea, 1329×1025 grid points, orthogonality error, 5-point, 2nd-order Laplacian



Black sea, 1329×1025, error in nonuniformity of grid spacing ratio



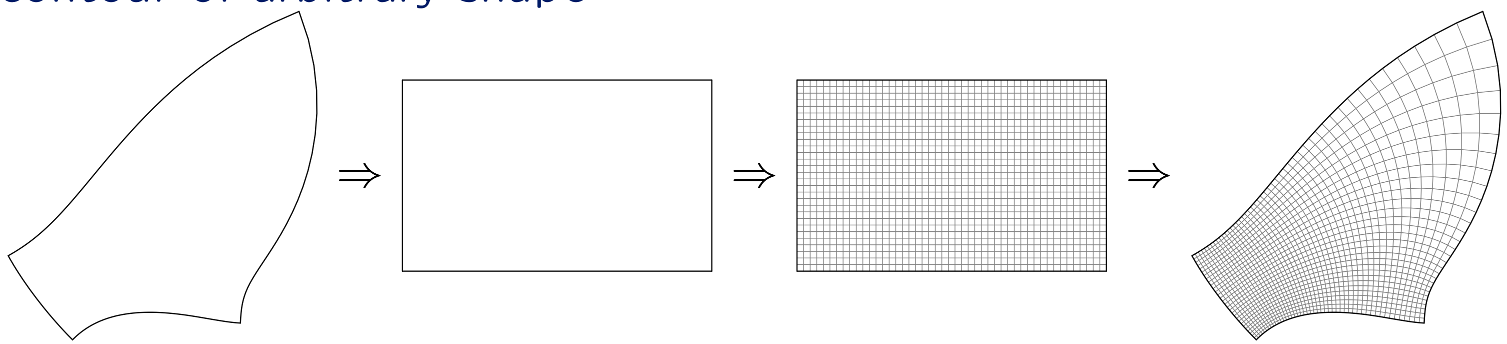
Black sea, 1329×1025, error in nonuniformity of grid spacing ratio, 5-point, 2nd-order Laplacian

What is achieved?

a completely new *compile once – use forever* **tool built from scratch;**

based on Schwartz–Christoffel transform;

in its core, it is a novel *two-level-nested iterative procedure* to construct *reversible* conformal mapping, $(lat, lon) \to (x, y) \to (\xi, \eta) \to (x, y) \to (lat, lon)$ for contour of arbitrary shape



the "corner problem" is solved completely: *guarantees exact 90-degree angles* at the side junctions, regardless of how user specifies reference points for contour spline;

cubic or *quintic* splines to construct grid contour;

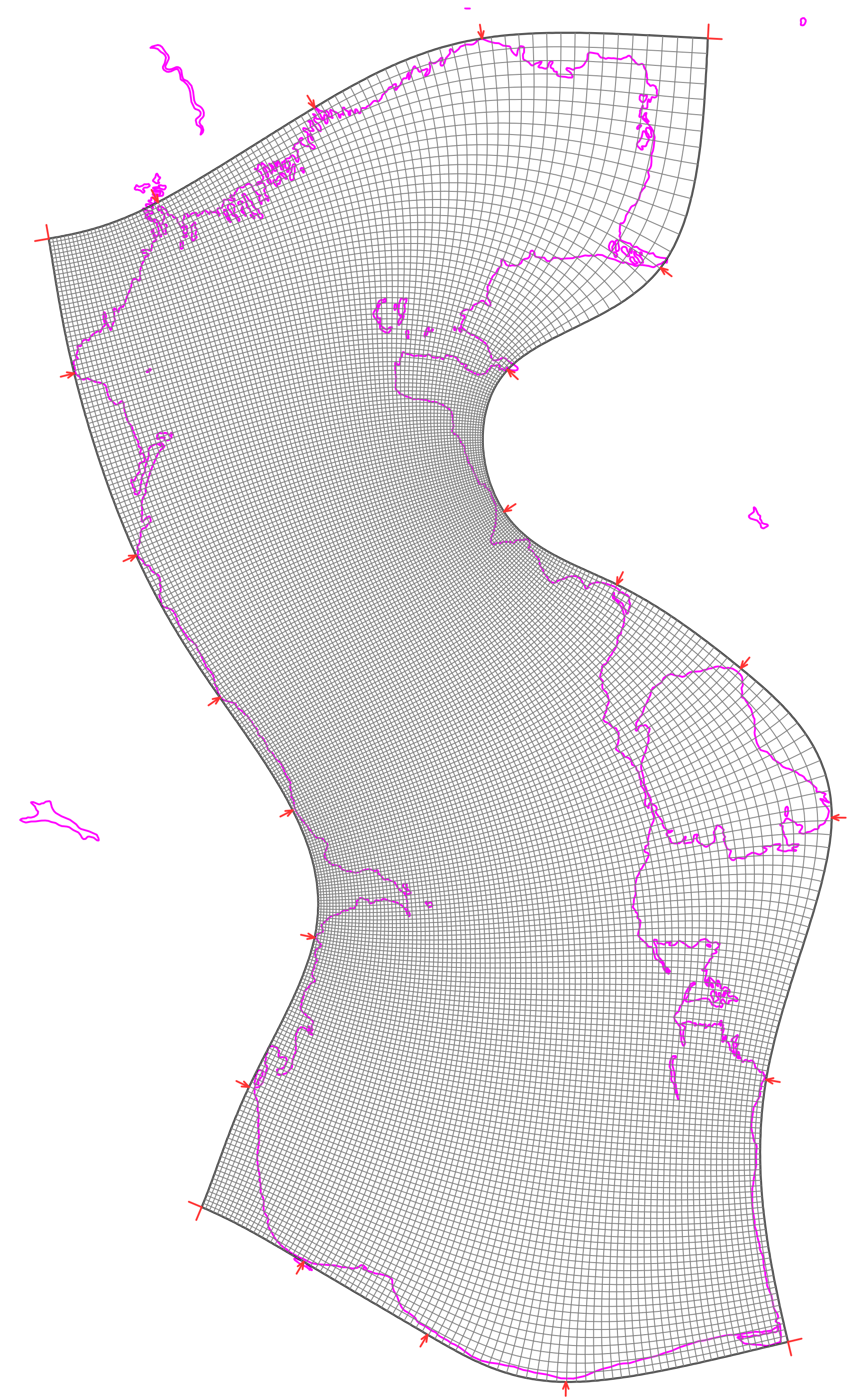
algorithm of Ives & Zacharias is rewritten from the first principles, from scratch, completely free of complex-number arithmetics, and is parallelized;

nine-point "mehrstellenverfahren" discretization of Laplacian operator, compact fourth-order accuracy, parallel solver;

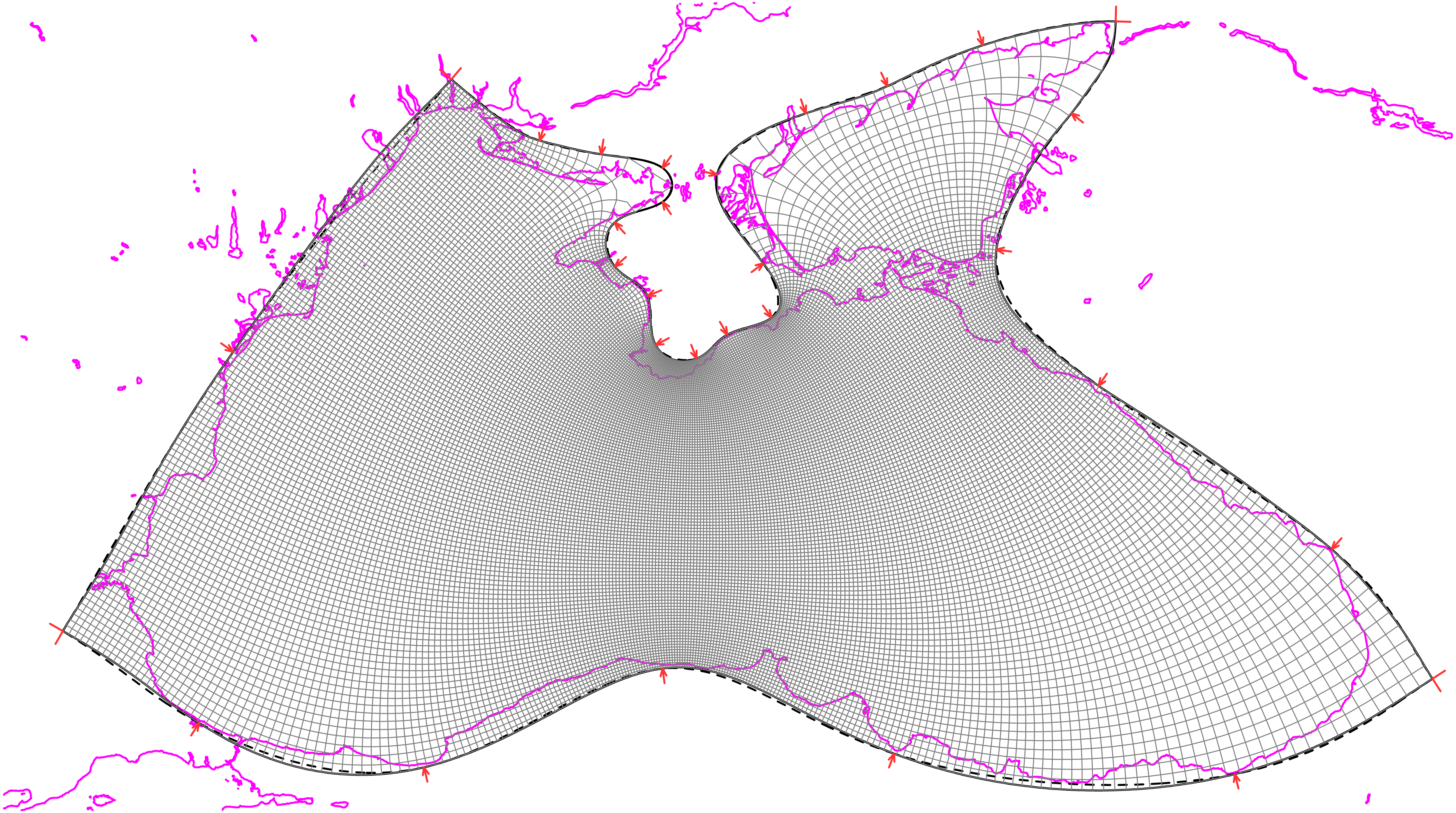
very small orthogonality error, $\sim 10^{-5}$ radians in realistic applications; converges to nothing as the number of grid points increases;

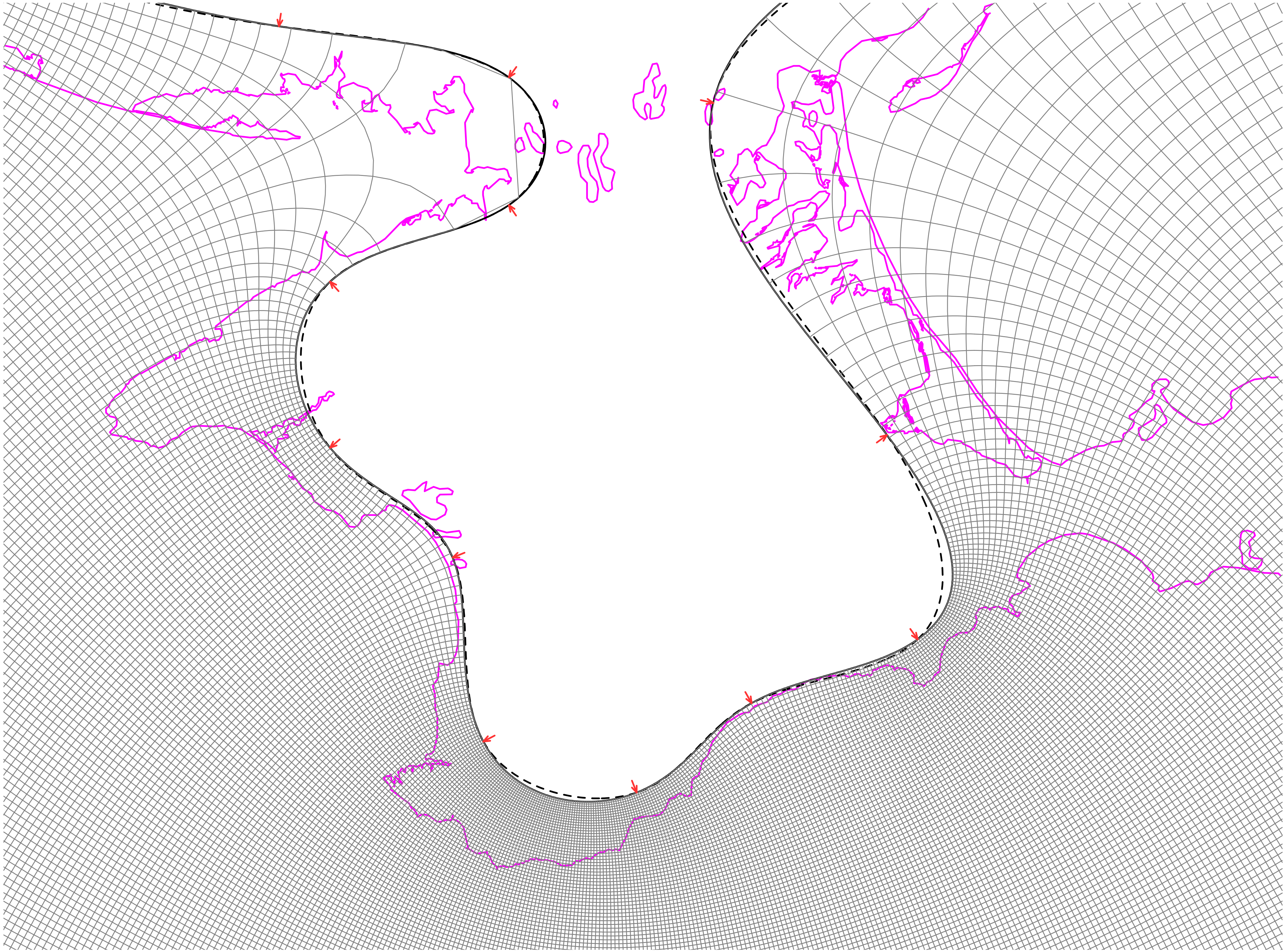
yields **locally equal grid spacing** in both directions, $\Delta x_{i,j} = \Delta y_{i,j}\,,\ \forall i, j$;

extremely robust

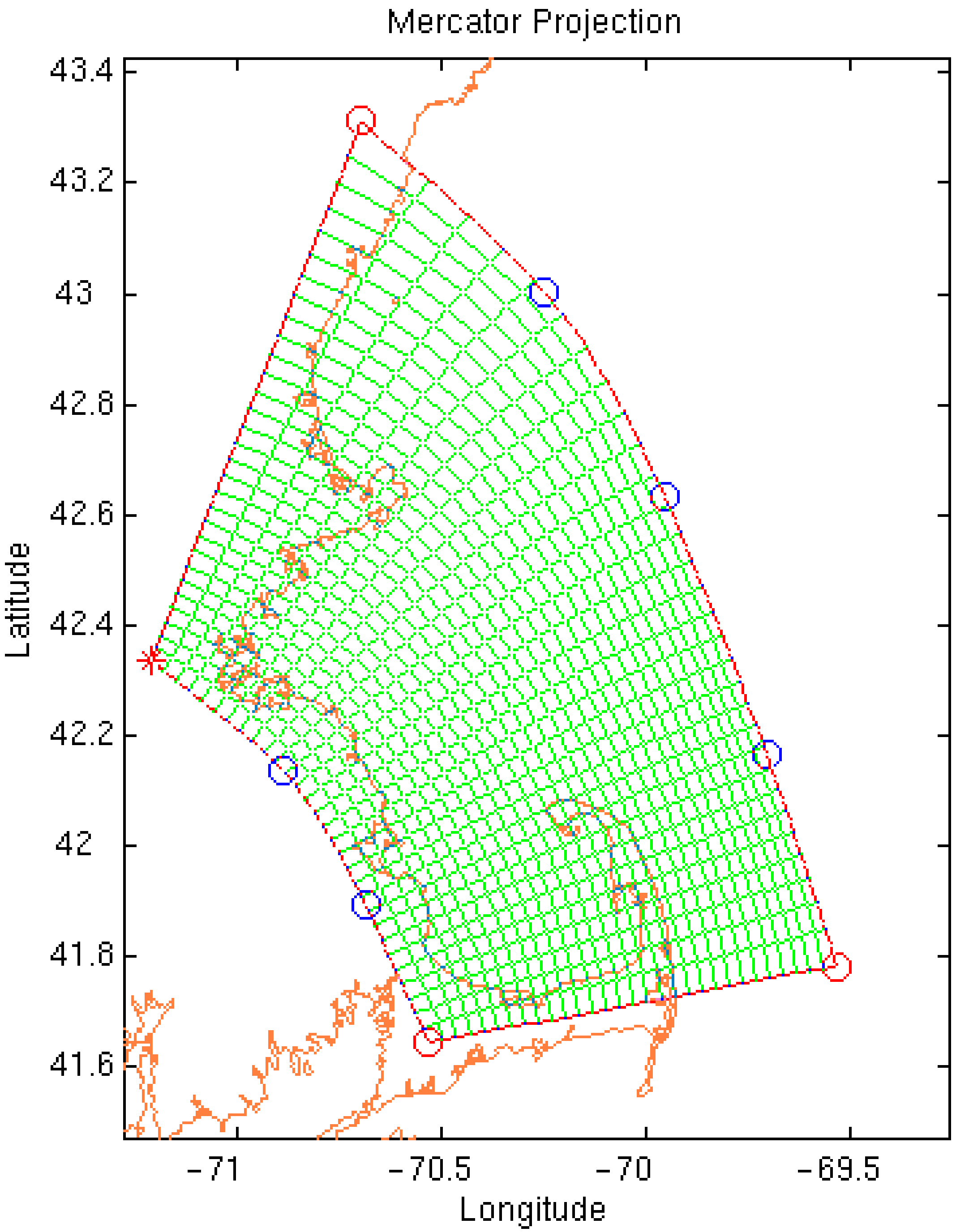


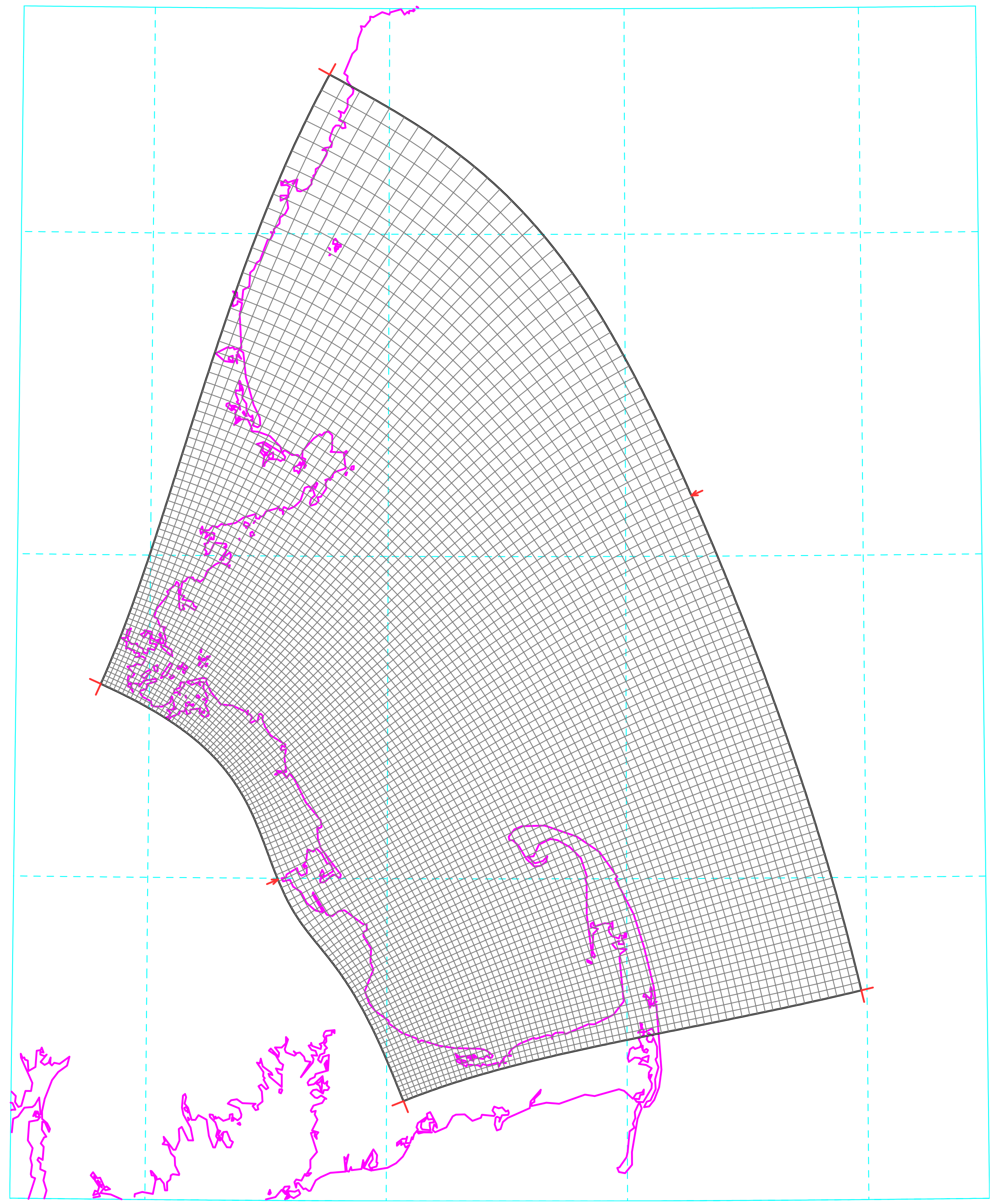
How robust?





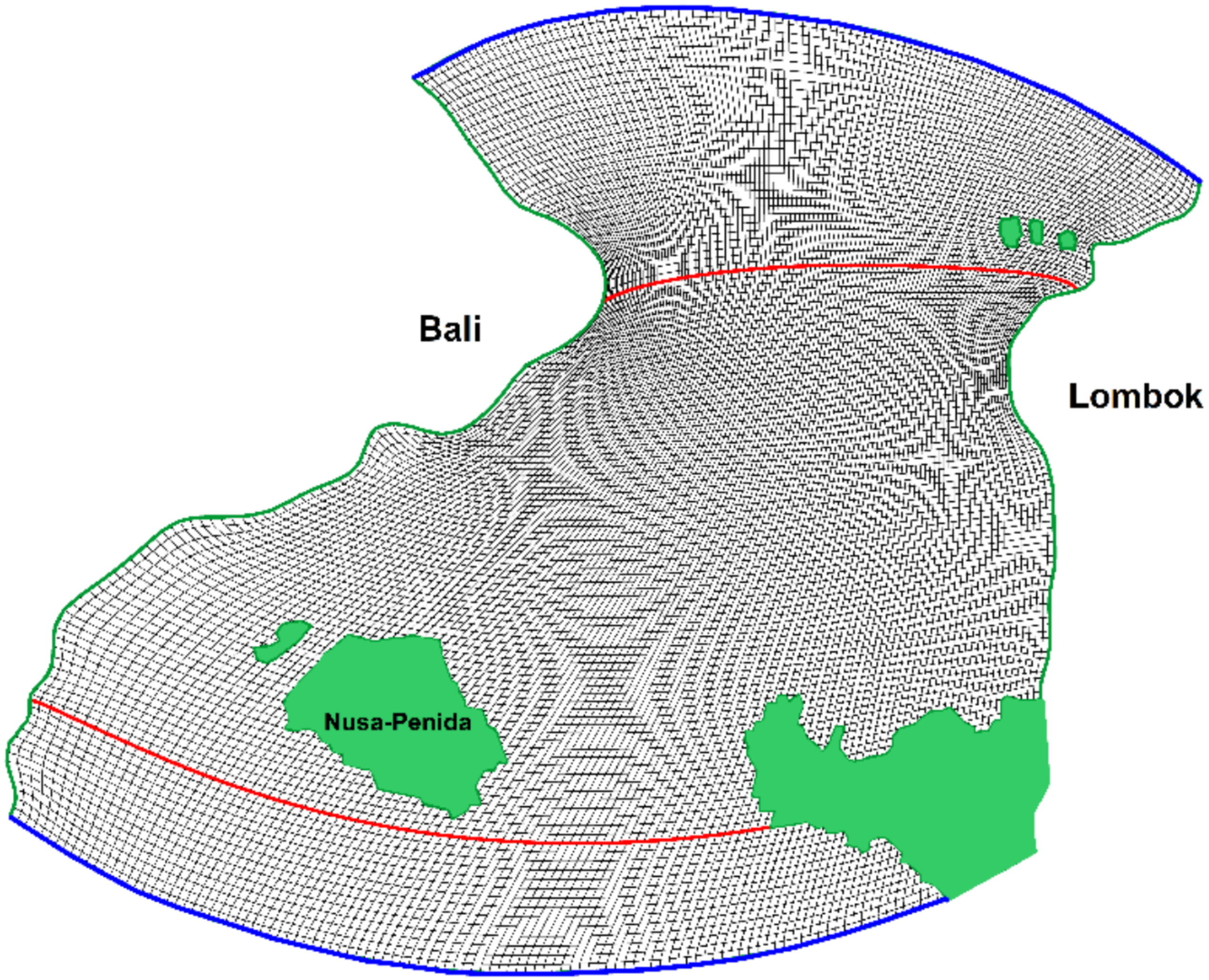
Massachusetts Bay & Cape Cod Bay

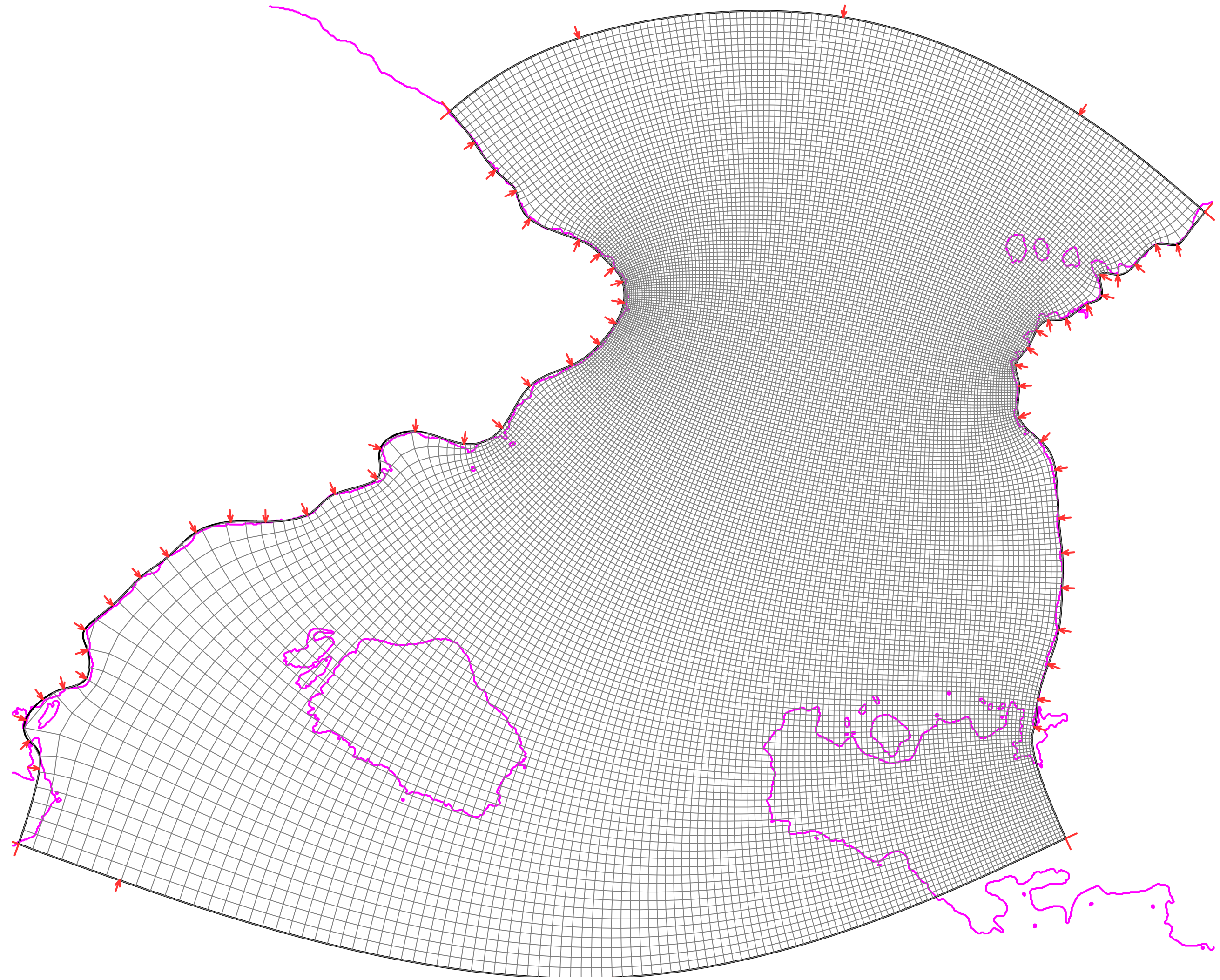




De facto this is seagrid logo.

https://github.com/sea-mat/seagrid





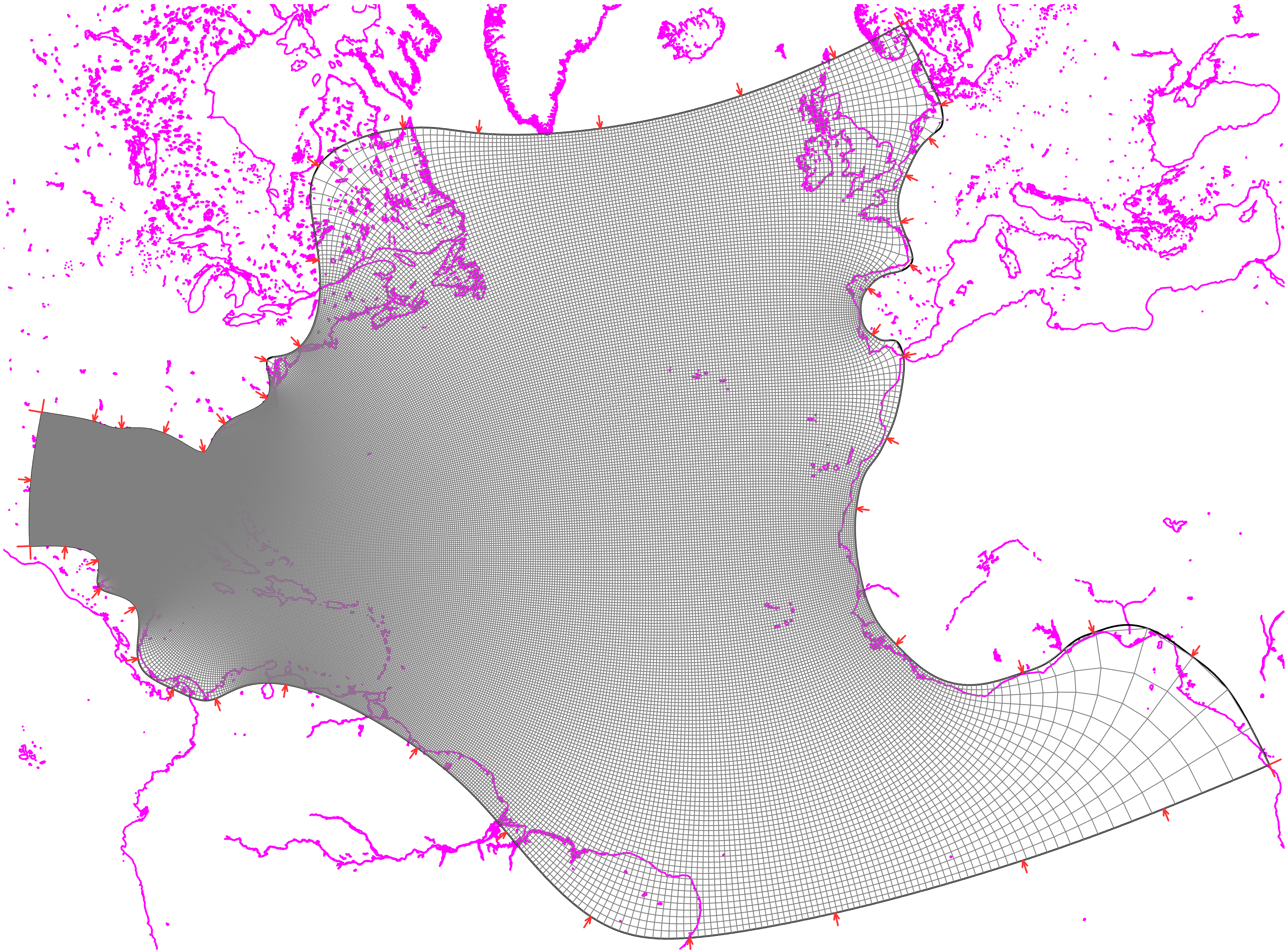
A. Androsov, N. Voltzinger, I. Kuznetsov, V. Fofonova (2020) Modeling of nonhydrostatic dynamics and hydrology of the Lombok Strait `https://www.mdpi.com/2073-4441/12/11/3092`

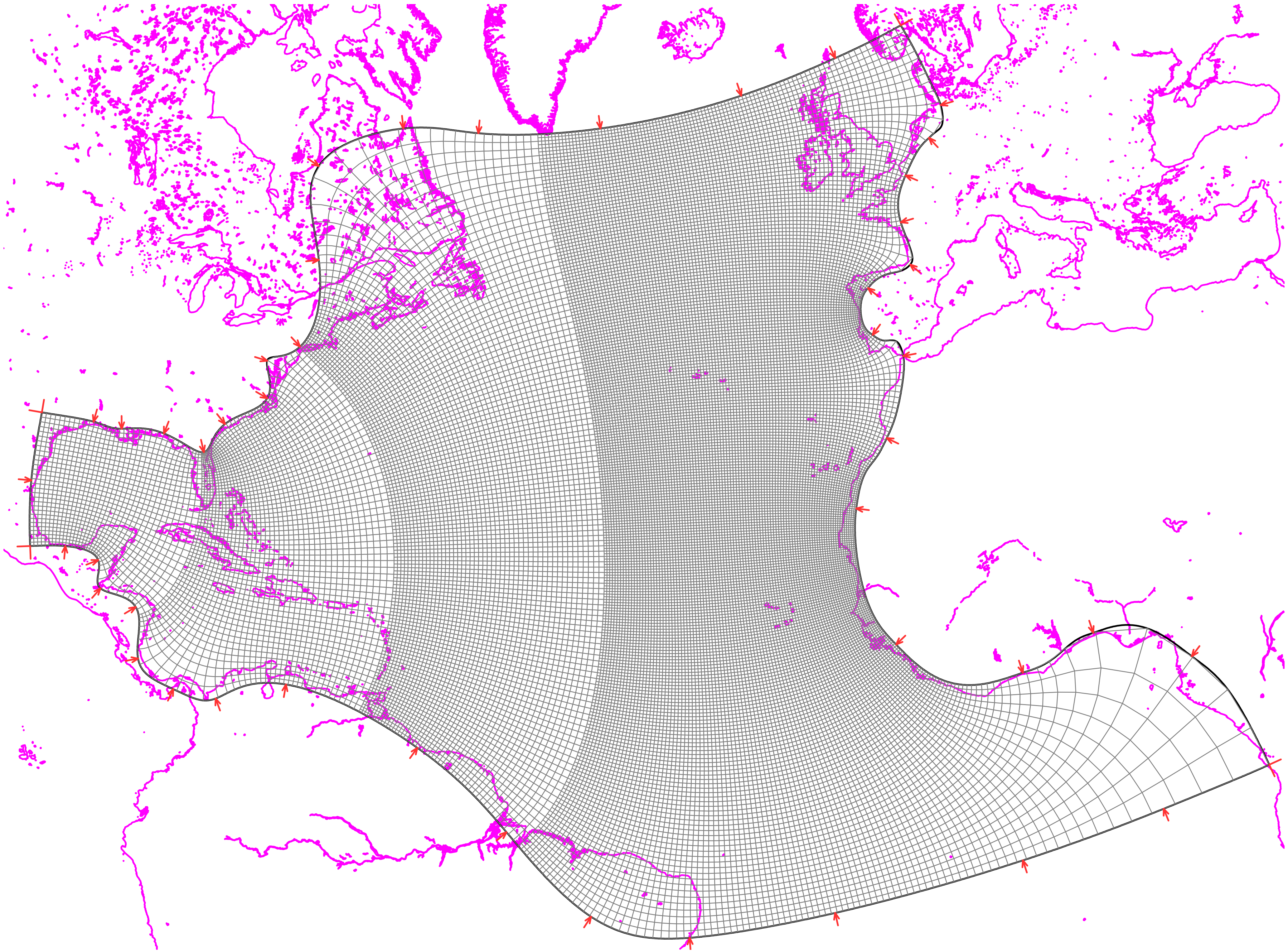
An example: Atlantic ocean with focus on Gulf of Mexico:

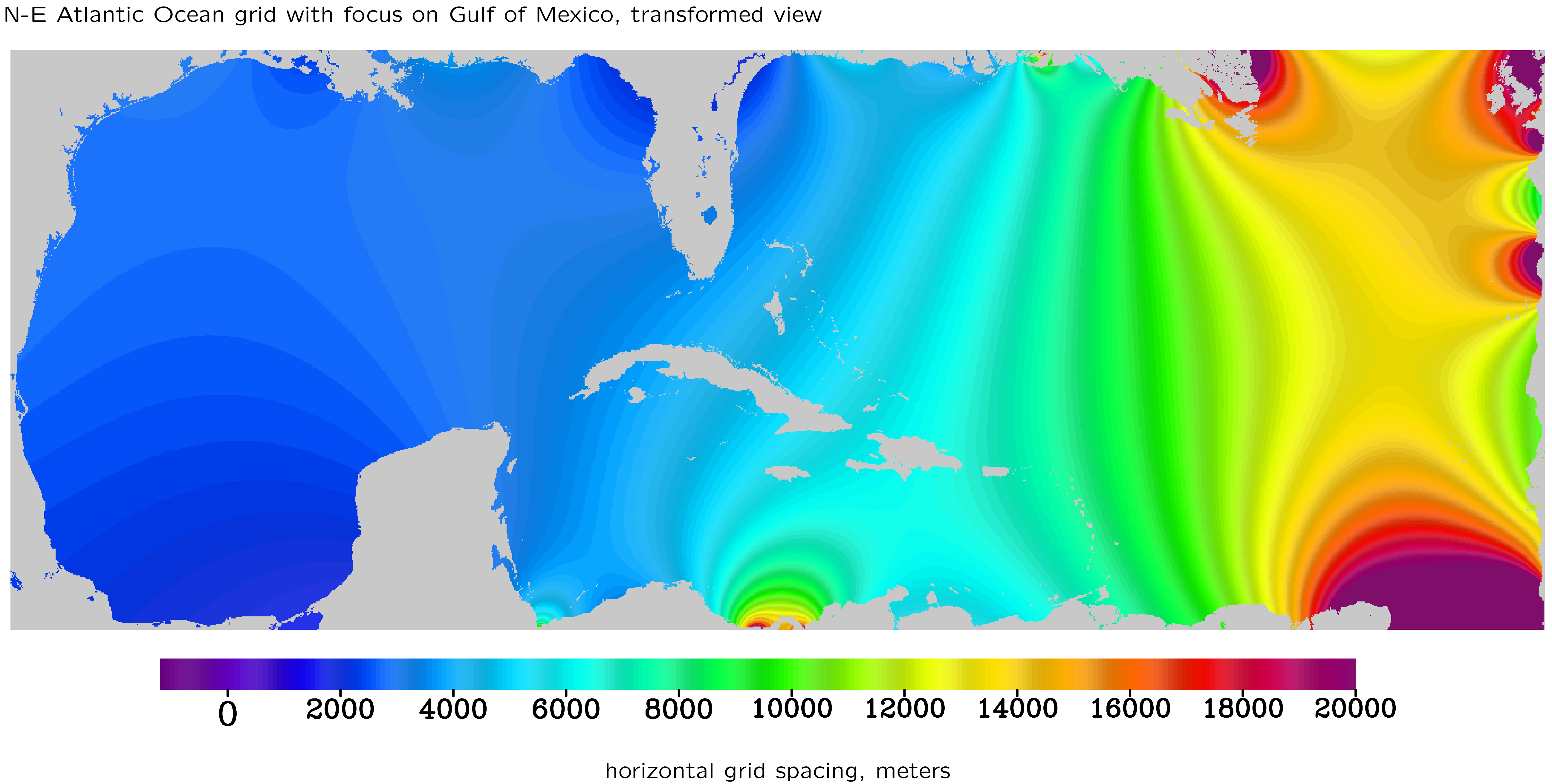
an alternative to multistage 2-way nesting

Shown on next slide: grid dimensions 637×241 points

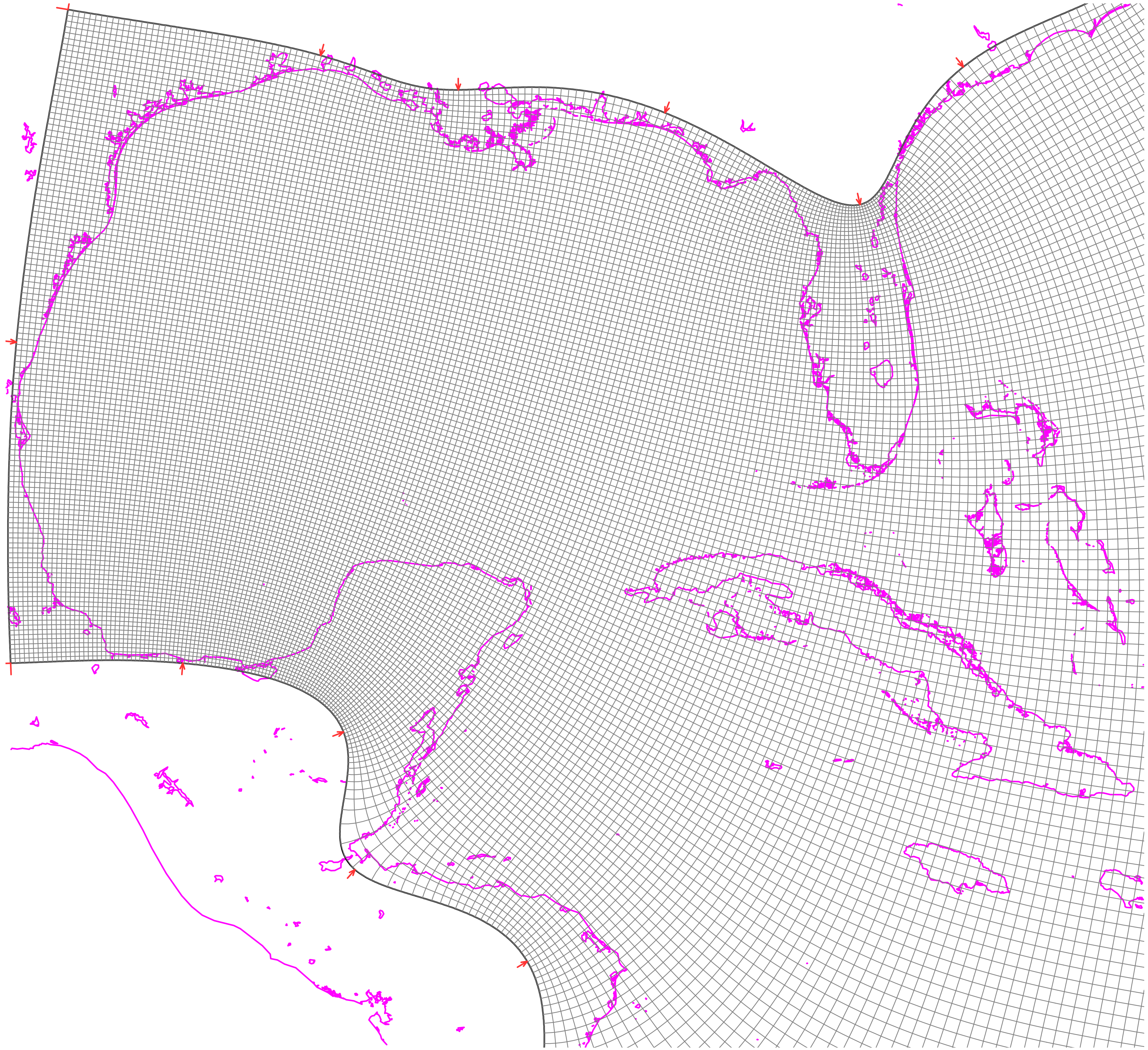
For illustrative purposes divide it into 4 zones showing:
1 out of 8 coordinate lines, 1 out of 4, every other one, and all lines.

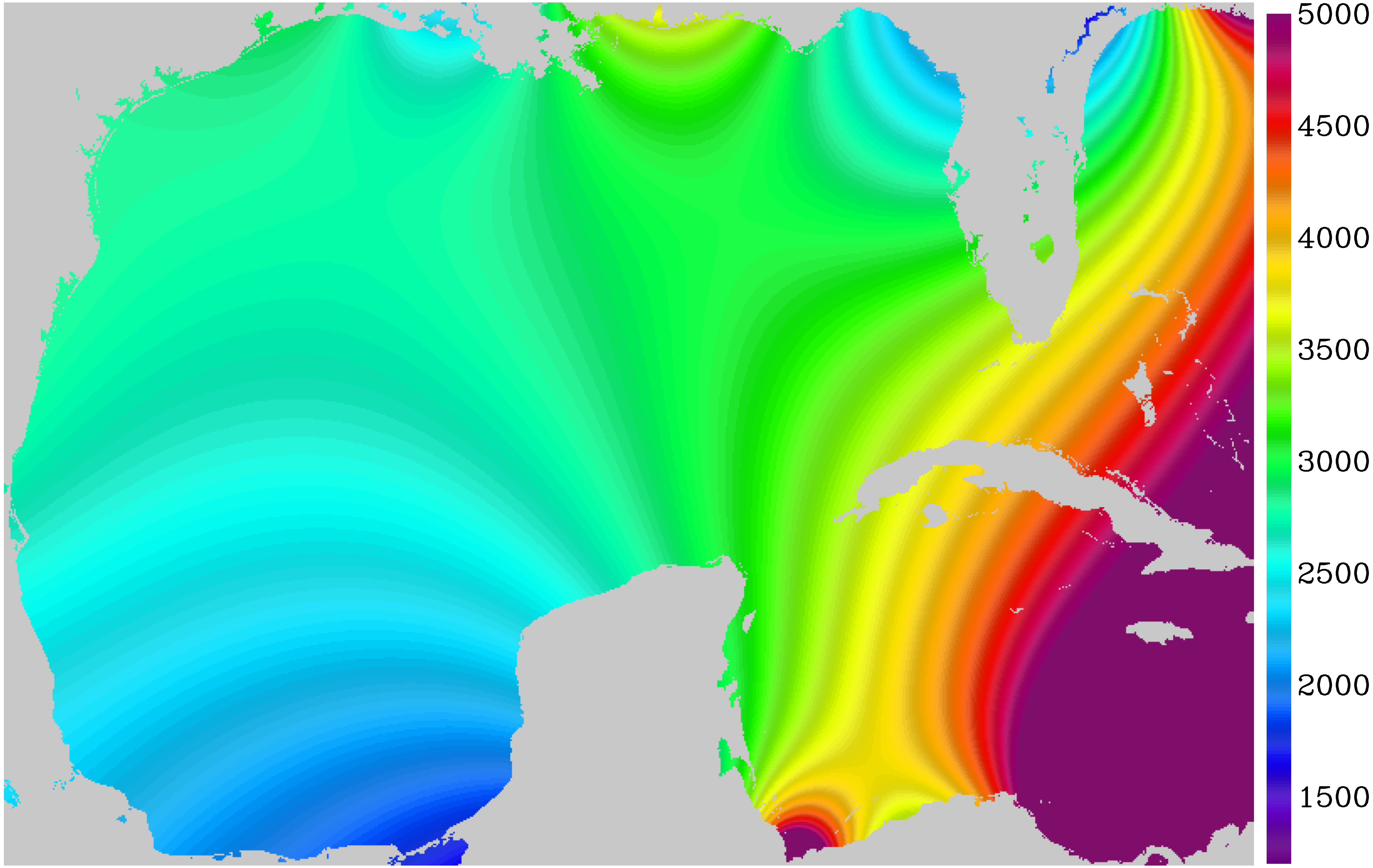




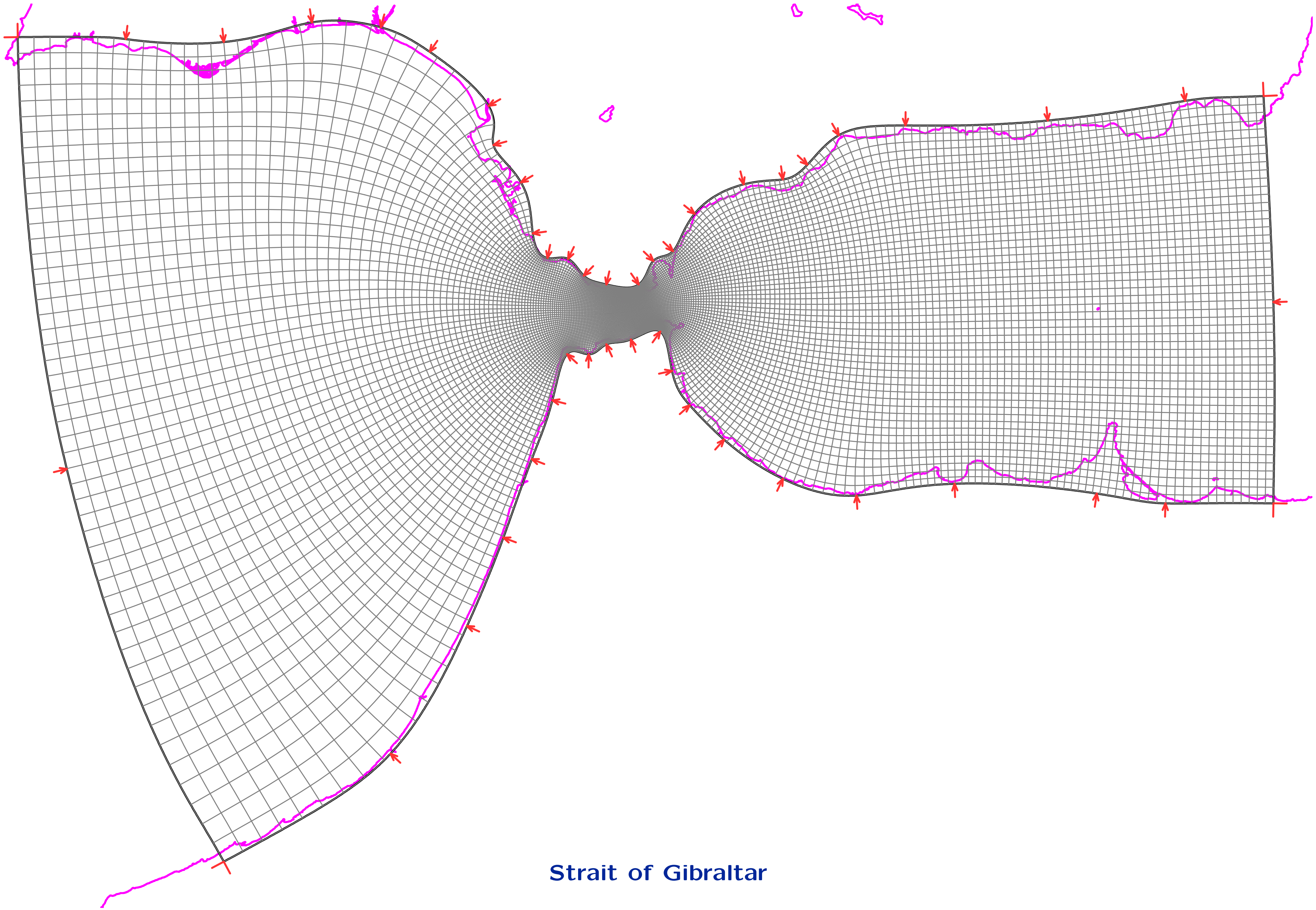


N-E Atlantic Ocean grid with focus on Gulf of Mexico, transformed view
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
horizontal grid spacing, meters

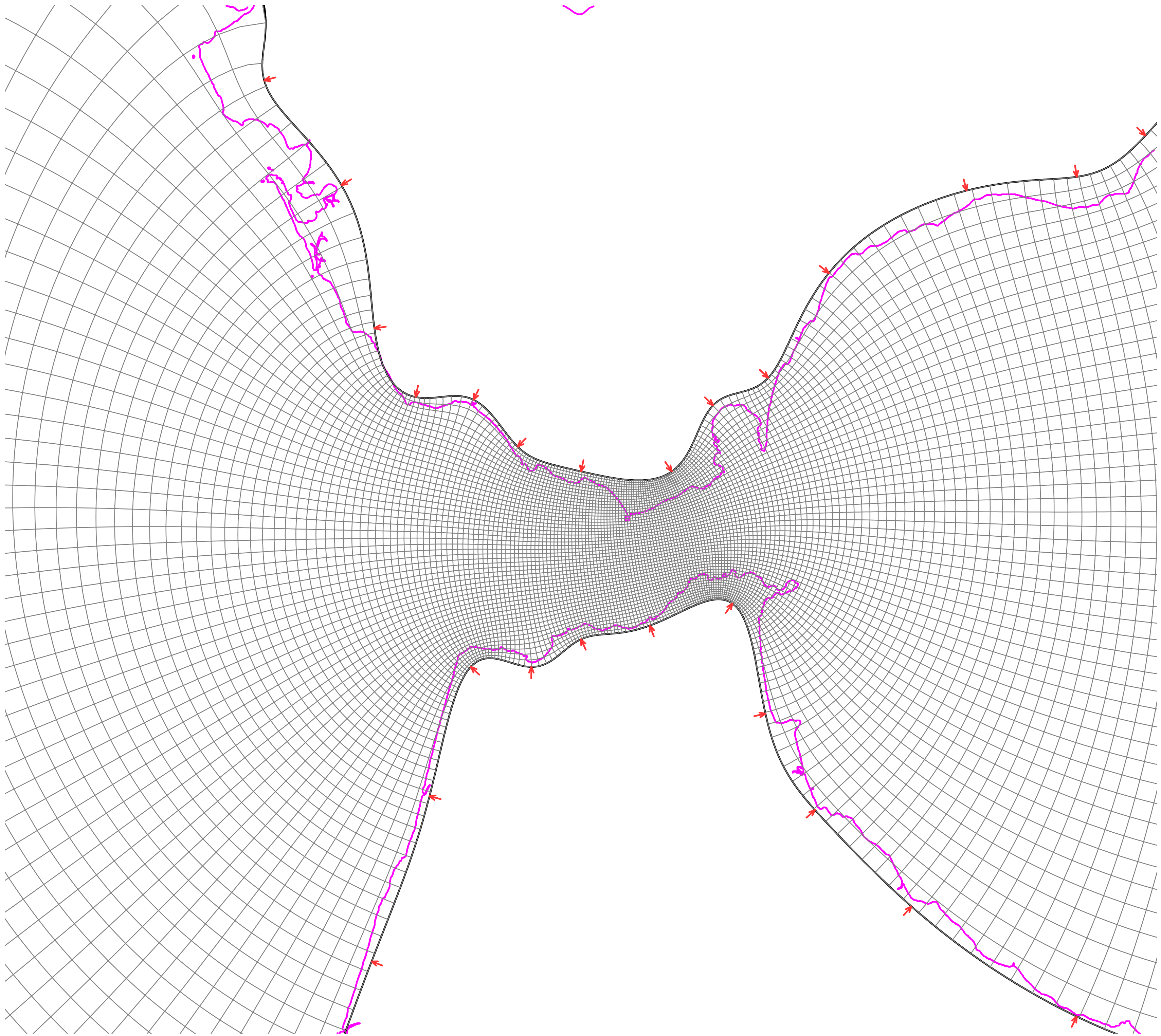




horizontal grid spacing, meters



Strait of Gibraltar



South Africa

focus on Benguela and Agulhas currents

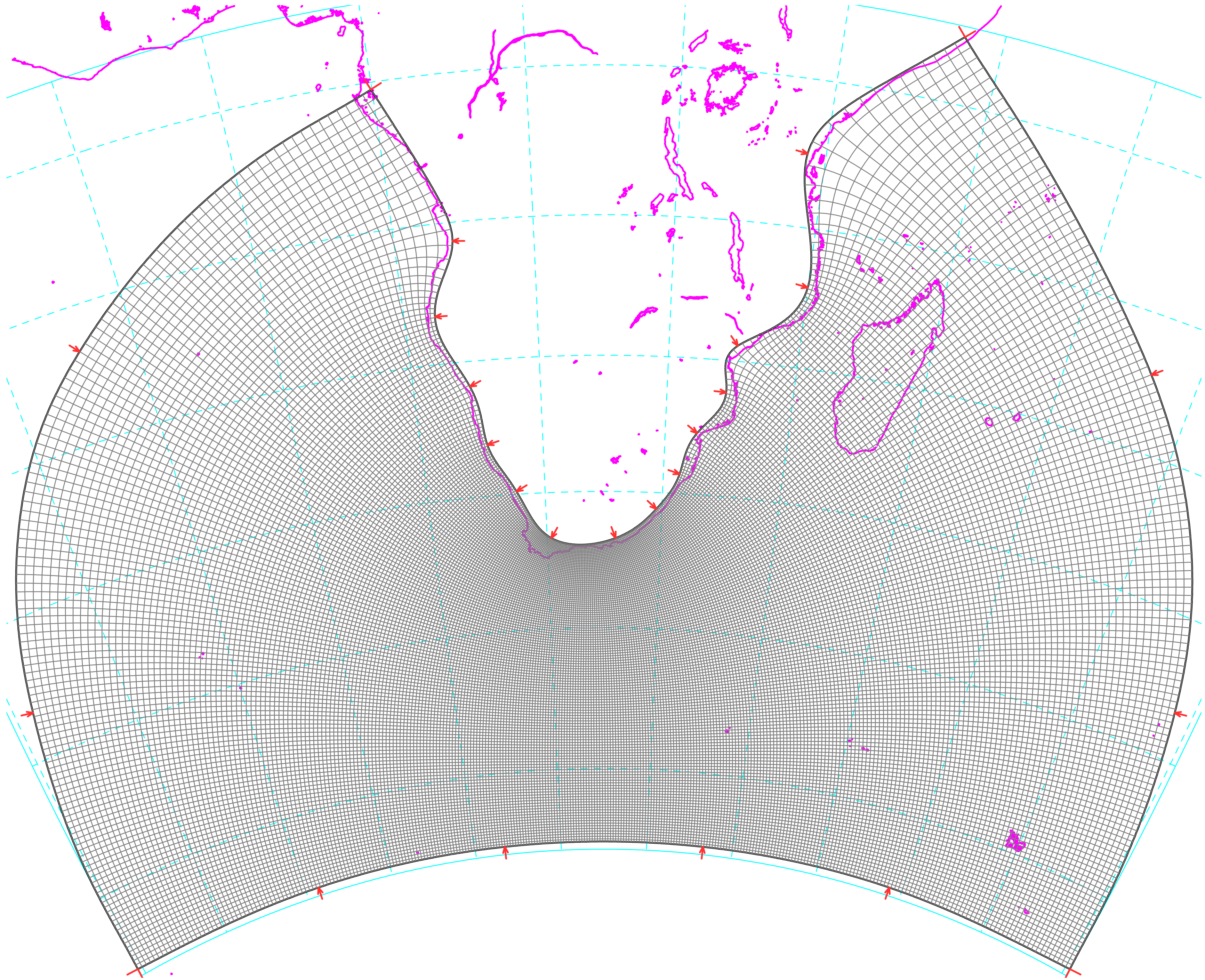
design goals:
place open boundaries as far as possible from the area of interest

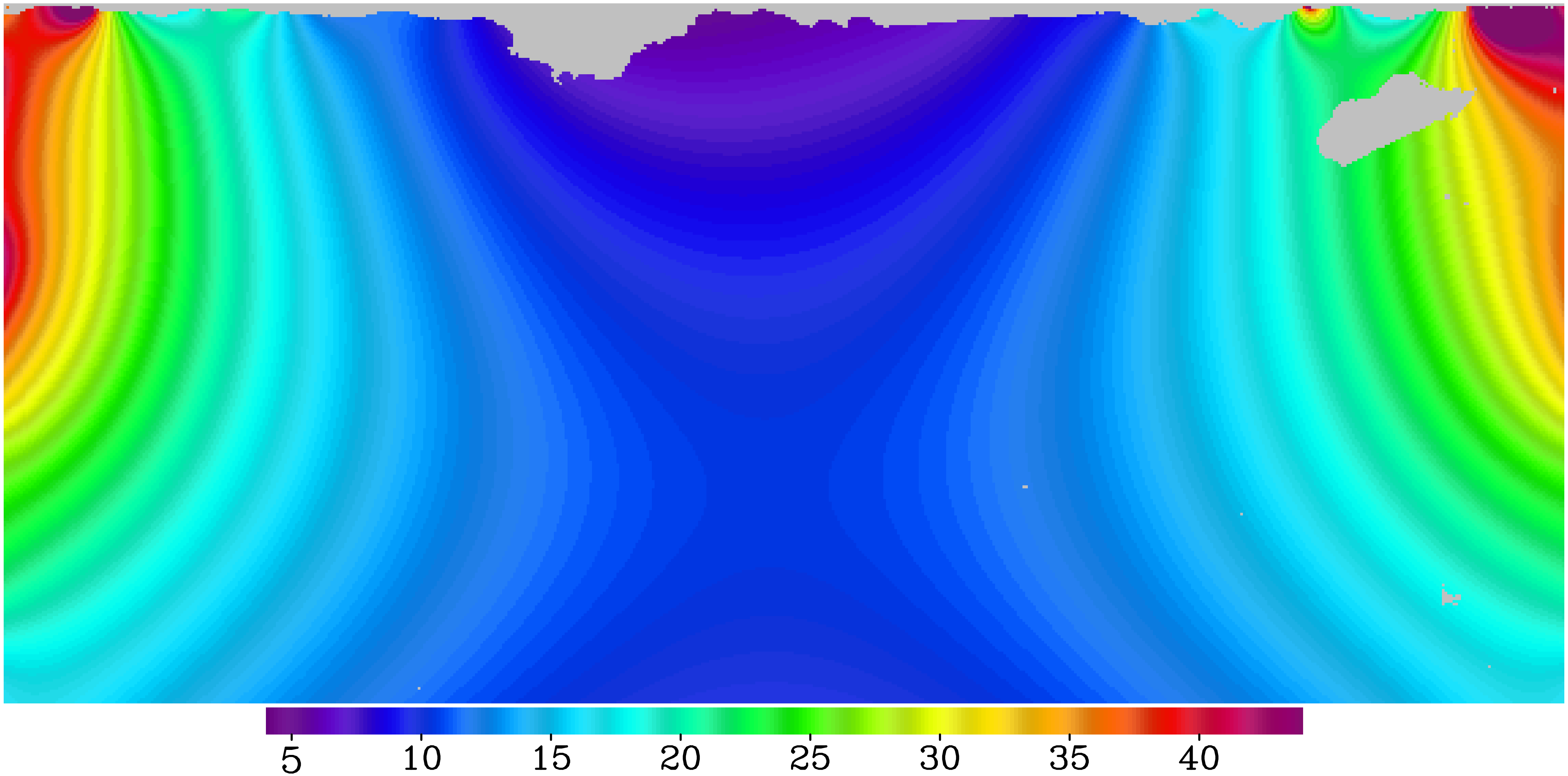
preferably make them perpendicular to the directions of major currents (best situation for inflo and outflow b.c. algorithms)

south open boundary is alinged with 55°S parallel

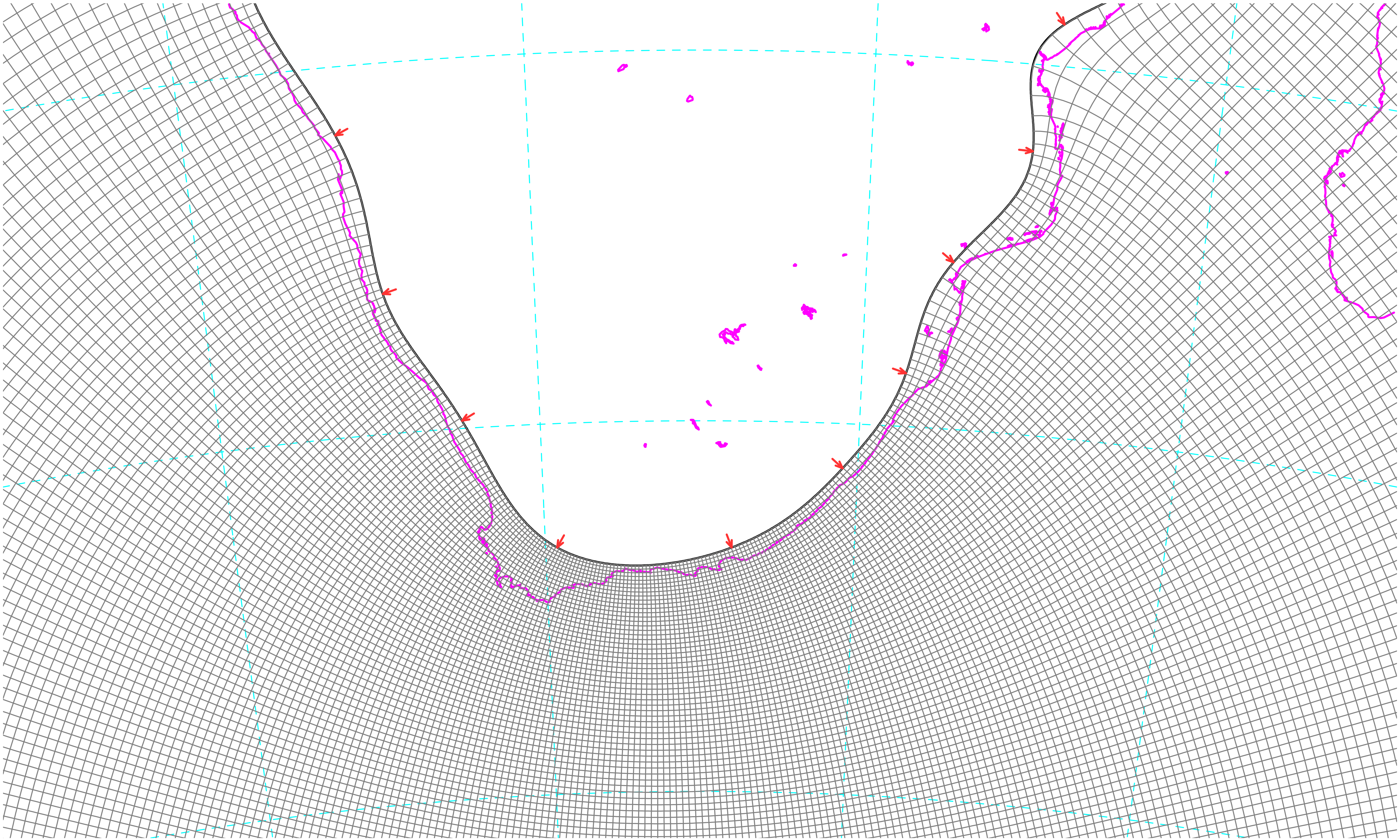
572×256 grid points

every other grid line is shown





South Africa, Grid spacing in transformed coordinates, *km*



Summary: Strategy of usage

The observed loss of interest in using orthogonal curvilinear grids in ROMS community is mainly explained by the absense of sufficiently good means to generate them. **Hopefully this situation will be improved by the present work.**

orthogonal curvilinear grids are **no substitute for unstructured** grids

orthogonal curvilinear grids **share all the advantages and disadvantages** of structured-grid modeling codes, i.e.:

- it is easier to do develop higher-order numerical schemes for structured model than for unstructured;
- for the same number of degrees of freedom, structured-grid models are much faster than unstructured;
- smoothness of curvilinear grid is essential to maintain accuracy of numerical algorithms of the model $\rightarrow$ do no attempt to "over-fit" to follow details of coast line;
- curvilinear grids are **not substitute for masking**, although the use of land mask can be significantly reduced;

as conformal mapping **is entirely controlled by the shape of the perimeter** of the future grid, so do the places where it puts fine and where coarse resolution. Therefore, by judicially bending the contour line (while deliberately placing it within land masked area, and therefore, allowing some freedom to choose how it goes), one can focus resolution in the places where it is desired most, or, conversely, make it as uniform as possible within the area of interest.

behavior of splines – depending how one places spline reference points – sometimes is not very intuitive. Therefore, it is advisable to **capture first the general geometrical shape** of the area of interest, using *as fewer reference points as possible*, and only after that "push" into capturing finer detail;

Finally, IZOGRID, as any other grid generator is **just an instrument in hand**. Depending on the geometry of the area, always look for *ad hoc approach* to construct optimal grid.