 Океанологические исследования, 2020, Том 48. № 4 С. 45-111

УДК 551.468.4 DOI:10.29006/1564-2291.JOR-2020.48(4).3

ПОСТРОЕНИЕ ОРТОГОНАЛЬНЫХ КРИВОЛИНЕЙНЫХ СЕТОК ДЛЯ РЕГИОНАЛЬНОГО МОДЕЛИРОВАНИЯ ОКЕАНА: МАТЕМАТИЧЕСКИЙ АЛГОРИТМ И РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Щепёткин А.Ф.

Институт океанологии им. П.П. Ширшова РАН,
Россия, 117997, Москва, Нахимовский проспект, 36
и *Московский физико-технический институт,*
Россия, 141700, Московская обл., г. Долгопрудный, Институтский пер. 9
e-mail: old_galaxy@yahoo.com
Статья поступила в редакцию: 12.06.2020, одобрена к печати 30.11.2020.

Новый алгоритм построения ортогональных криволинейных сеток на сфере для достаточно общей геометрической формы моделируемой области реализован в виде пакета программ по принципу *compile-once – use forever*. В основе лежит численное решение обратной задачи итерационным методом – нахождение такого распределения точек-узлов сетки вдоль её периметра, чтобы конформное преобразование периметра в прямоугольник превратило это распределение в равномерное. Сама же итерационная процедура при этом получается многоуровневой – т.е. итерационный цикл, построенный вокруг другого (внутреннего) итерационного цикла. В дальнейшем, зная это распределение, узлы сетки внутри области получаются решением эллиптической задачи. При этом удалось получить точную ортогональность периметра в углах сетки, добиться малого, не достижимого ранее, уровня ошибок ортогональности сетки, а так же сделать её изотропной – локальные расстояния между узлами сетки в обоих направлениях равны друг другу.

Ключевые слова: построение ортогональных криволинейных сеток на сфере, конформные отображения, дискретное преобразование Шварца–Кристоффеля, вложенные итерационные методы решения обратной задачи, региональное моделирование океана

1. Введение

Несмотря на то, что практически все современные модели океана написаны в общих ортогональных криволинейных координатах для горизонтальных направлений, и, следовательно, совместимы с криволинейными сетками произвольной формы, на практике применение таких сеток очень ограниченно, что, в свою очередь, приводит к неполному использованию всего потенциала возможностей, которые дают криволинейные координаты.

Стоит отметить, что в начале 1990-х годов построение криволинейных сеток, в которых одна из сторон следует вдоль береговой линии, было довольно типичным для практики моделирования прибрежных явлений, в качестве примера см. Fig. 4 из

(Haidvogel et al., 1991) или Fig. 1 в (Gan, Allen, 2005). Однако постепенно от подобных сеток стали отказываться, предпочитая их более простым – как правило, простой прямоугольный участок Меркаторской сетки, повёрнутый к нужной ориентации в сочетании с использованием земляной маски.

Одной из причин такого выбора являлось отсутствие достаточно надёжных и точных математических алгоритмов, а также их практических программных реализаций для создания криволинейных сеток. Связанное с этим численное решение задач нахождения конформного преобразования контура в прямоугольник было недостаточно точным, что в результате приводило к ошибкам ортогональности за пределами допустимого уровня. Ещё один существенный недостаток – это стремление точно следовать береговой линии в контексте моделирования океана, что приводит к локальному сгущению сетки вблизи выпуклостей береговой линии, и это, в свою очередь, вызывает ограничения на шаг по времени, чтобы не превысить критические значения числа Куранта в этих местах. Более оптимальной является стратегия, заключающаяся в использовании криволинейной сетки в сочетании с земляной маской так, чтобы получить более высокое разрешение в нужном месте, но, в то же время, избежать неконтролируемых ограничений по числу Куранта. Примером удачного построения такой сетки является сетка, представленная на Fig. 1 в работе Ezer & Mellor (2000), где удалось сконцентрировать разрешение в зоне Гольфстрима, но в то же время обойтись умеренным разрешением, и, следовательно, уменьшить вычислительные затраты для модели в целом.

Парадоксально, но в настоящее время в сообществе пользователей ROMS (Regional Oceanic Modeling System, Shchepetkin & McWilliams, 2005) возможность построения ортогональных криволинейных сеток фактически утрачена. Так, исторически наиболее ранний работоспособный инструмент для построения сеток для SPEM–SCRUM–ROMS

https://marine.rutgers.edu/po/tools/gridpak/grid_manual.ps.gz

появился в начале 1990-х годов и был основан на работе Ives & Zacharias (1989), которую для краткости мы далее обозначим IZ89. По существу, этот алгоритм состоит из дискретного конформного преобразования криволинейного контура (фактически ломаной линии) в прямоугольник, с последующим решением задачи Дирихле, чтобы заполнить сетку внутри. Он довольно успешно применялся для создания прибрежных конфигураций модели (одна сторона сетки шла вдоль береговой линии; открытые границы на остальных трёх). В частности, в работе Marchesiello et al. (2003) использовалась именно такая сетка. Сам же контур задавался набором точек, которые служили основой для построения кубических сплайнов отдельно на каждой стороне. Сплайны строились, подразумевая так называемые естественные граничные условия на концах (то есть в углах контура), что эквивалентно приравниванию вторых производных к нулю. Никакого механизма для достижения перпендикулярности сторон контура в угловых точках предусмотрено не было – фактически пользователь должен был задать точки определённым образом, надеясь на глазомер, чтобы добиться ортогональности сопряжения сторон контура. При этом необходимо отметить, что особенностью алгоритма IZ89 (как, впрочем, и любого другого,

основанного на отображении Шварца–Кристоффеля) является то, что ортогональность сопряжения сторон контура формально не требуется: алгоритм применим к любому замкнутому контуру, состоящему из сегментов ломаной линии, однако последствиями нарушения ортогональности является неконтролируемое сгущение или разбег узлов сетки вблизи углов. Распределение же узлов на двух перпендикулярных сторонах создаваемой сетки задавалось (как правило, делалось равномерным), а на соответствующих им противоположных сторонах – находилось алгоритмом IZ89 из условия одинаковости расположения узлов на противоположных сторонах прямоугольника, полученного конформным преобразованием исходного контура (Fig. 8 в Ives & Zacharias, 1989). При таком подходе желательно иметь плавные линии контура (прямые или близкие к окружностям большого радиуса), по крайней мере, на двух смежных сторонах, что, в свою очередь, и ограничивало применимость береговыми конфигурациями, в которых такие плавные линии всегда можно выбрать на открытых границах.

Подобные сетки также использовались в модели POM, и к началу 2000-х появился пакет `SeaGrid`,

https://archive.usgs.gov/archive/sites/woodshole.er.usgs.gov/operations/modeling/seagrid/tutorial.html

а так же,

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

который поддерживался в то время United States Geographical Servey и использовал интерфейс `Matlab`. Он был достаточно универсален – позволял строить сетки совместимые сразу с несколькими моделями. А благодаря удобству использования пакет стал довольно популярен. Вместе с тем он обладал фактически теми же недостатками и ограничениями: трудностью соблюдения ортогональности углов, недостаточно высокой, чрезмерно чувствительной к выбору опорных точек, точностью алгоритма конформного отображения, несовершенством распределения точек сетки вдоль контура. В 2014 г., в результате изменений, произошедших в `Matlab`, графический интерфейс `SeaGrid` стал несовместим с современными версиями `Matlab` и фактически оказался нежизнеспособным из-за прекращения поддержки.

Пакет `GridBuilder`, появившийся во второй половине 2010-х годов,

https://austides.com/downloads

позиционировался его разработчиками как замена `SeaGrid`, совместимая с современным графическим интерфейсом `Matlab`. Однако, при ближайшем рассмотрении, оказывается, что он может использоваться только для построения простых сеток, представляющих собой прямоугольный участок Меркаторской сетки с возможностью её поворота на произвольные углы. Возможность построения криволинейных сеток для общего случая находится в стадии разработки, но, как и прежде, применяемые там алгоритмы обладают теми же самыми недостатками, что и `SeaGrid`.

Стоит так же упомянуть ещё одну ветвь развития алгоритмов данного назначения – пакет `gridgen`. Изначально он появился в инженерном сообществе (Driscoll, Vavasis, 1998), но в своё время был адаптирован Павлом Саковым для геофизических приложений,

https://github.com/sakov/gridgen-c

Здесь основной акцент сделан на возможности моделировать объекты сложной формы (в оригинальных инженерных задачах граница сетки, как правило, проходит по границе обтекаемого тела, без использования маски), при этом точность соблюдения ортогональности сетки является несколько меньшим приоритетом. В настоящее время основные усилия в сообществе ROMS (так же GETM и NEMO) ведутся по созданию интерфейса для удобства использования,

https://phobson.github.io/pygridgen/

в то время, как основные алгоритмы практически остались без изменений за фактически десятилетие. В частности, в недавно вышедшей работе Bruciaferri et al. (2020) использовался именно этот пакет для создания горизонтальной сетки.

Вместе с тем, за последние 10–15 лет подавляющее большинство работ, выполненных с моделью ROMS, использовало простые аналитические сетки, как правило, Меркаторские (так что расстояние между узлами по широте делается пропорциональным косинусу широты с тем, чтобы локально расстояния в обоих направлениях были равны или хотя бы пропорциональны друг другу), а так же повернутые Меркаторские сетки – в этом случае прямоугольный участок сетки нужных размеров в сферических координатах создаётся симметрично относительно экватора, так что пространственная неоднородность разрешения минимальна (по сути такая сетка стремится к декартовой, если область моделирования мала по сравнению с радиусом Земли), затем твёрдотельным вращением на нужные углы приводится в нужное место и азимутально поворачивается, чтобы наилучшим образом (насколько это возможно в рамках ограничений прямоугольной сетки) соответствовать конфигурации данной задачи. На это существует несколько причин, самая очевидная из которых – это отсутствие достаточно хороших средств для построения криволинейных сеток, а так же более сложный анализ и визуализация полученных решений. Одновременно появилось сразу несколько пакетов для создания простых сеток: `EasyGrid` и им подобные. Один из них – наш собственный,

https://www.myroms.org/forum/viewtopic.php?f=14&t=4775&p=18530

который, помимо повернутых Меркаторских сеток, всё же даёт ограниченную возможность создания криволинейных сеток при помощи набора последовательных *аналитических* конформных отображений. Получившиеся при этом сетки обладают идеальной ортогональностью (так как по сути они аналитические), но возможность создания сетки произвольной формы довольно ограничена. Ещё одним недостатком является трудоёмкий и непростой в освоении способ задания геометрических параметров – изменение любого из них оказывает влияние сразу на всю сетку, поэтому на практике это приводит к длительному итеративному подбору методом проб и ошибок, каждый раз отслеживая изменение, вызванное малыми приращениями.

В настоящей работе мы поставили задачу построения криволинейных ортогональных сеток именно произвольной формы, но при этом так, чтобы наш алгоритм автоматически гарантировал ортогональность с высокой точностью, *независимо* от значений параметров, заданных пользователем.

Ещё одно свойство, которое желательно получить, но практически такая цель

никогда ранее не ставилась, – это сделать так, чтобы расстояние между узлами сетки в обоих направлениях было одинаковым в каждом месте. Это связано с обычной практикой применения модели ROMS без явных членов для горизонтальной вязкости – диссипативные свойства численного алгоритма горизонтальной адвекции делают модель численно устойчивой, а, как правило, целью такого выбора является получение как можно менее вязких решений. В результате диссипация энергии и энстрофии идёт на масштабе сетки, и, в случае наличия свободных оторвавшихся вихрей, необходимо создать условия отсутствия искусственного нефизического усиления или уменьшения диссипации в одном из направлений. Мы так же покажем, что на практике возможно сделать сетку изотропной.

В целом, последовательность действий состоит из построения сетки на плоскости с последующим конформным преобразованием к сферическим координатам. Ортогональная криволинейная сетка, вписанная в данный криволинейный контур, на плоскости порождается обратимым конформным преобразованием

$$(x, y) \leftrightarrow (\xi, \eta) \tag{1}$$

этого контура на прямоугольник, с тем, чтобы внутри него можно было построить прямоугольную декартову сетку, а затем превратить её в искомую криволинейную сетку обратным конформным преобразованием (рис. 1).

В силу конформности обратного преобразования криволинейная сетка сохраняет такие свойства декартовой, как соотношение сторон ячеек, в частности, если разрешение декартовой сетки в обоих направлениях одинаково, $\Delta\xi = \Delta\eta$, то криволинейная сетка тоже будет обладать свойством локальной изотропии разрешения, $\Delta x = \Delta y$, хотя, конечно, размер ячеек неоднороден по пространству. В частности, сетка, изображенная на рис. 1, принадлежит именно к такому классу.

Что касается конформного преобразования, то оно существует в общем виде и связано с интегралом Шварца–Кристоффеля (Schwarz, 1890; Driscoll & Trefethen, 2002; Vorobieff). Основная же трудность заключается именно в нахождение обратно-

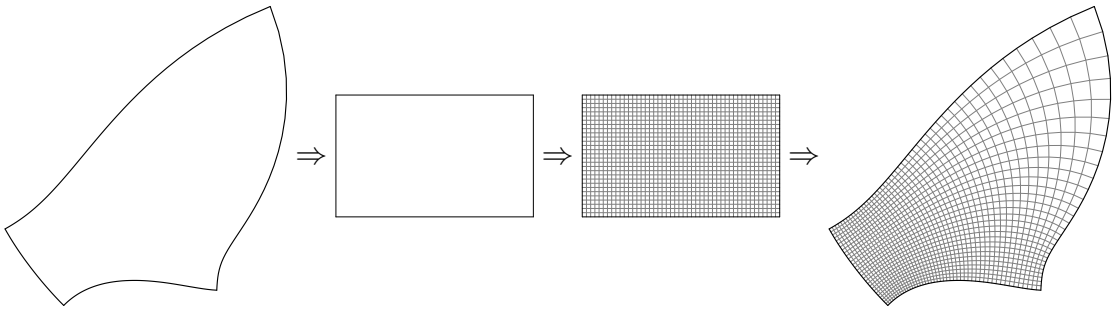


Рис. 1. Математический принцип построения ортогональной криволинейной сетки на плоскости при помощи конформного преобразования: для любого криволинейного контура с четырьмя углами существует обратимое конформное преобразование, отображающее его на прямоугольник, причем соотношение сторон этого прямоугольника не произвольно, а однозначно определяется формой контура. Декартова сетка, построенная в прямоугольнике, отображается в искомую криволинейную обратным преобразованием.

го преобразования, которое (впрочем, как и прямое) необходимо строить численно с применением итерационных методов. Новизной, т.е. принципиальным отличием от всех прототипов, настоящей работы является построение практически реализуемого алгоритма для решения возникающей здесь *обратной задачи*, так, чтобы на дискретном уровне обеспечить сходимость с точностью вплоть до ошибок округления компьютерного счёта.

Конформное преобразование удобно выразить в терминах функций комплексного переменного: $z = z(\zeta)$, где $z = x + iy$ и $\zeta = \xi + i\eta$, причём функция z и ей обратная $\zeta = \zeta(z)$ голоморфны во всей области $\mathscr{D}$,

$$\partial_\xi x = \partial_\eta y \quad \text{и} \quad \partial_\xi y = -\partial_\eta x\,, \tag{2}$$

что, с одной стороны, является условием дифференцируемости функции $z = z(\zeta)$ как функции комплексного переменного (т.е. (2) это условия Коши–Римана), а с другой, одновременно является и условием сохранения ортогональности для системы координат (x, y). Так, для любой пары векторов бесконечно малых приращений, $\boldsymbol{\ell}_1 = (\delta_1\xi, \delta_1\eta)$ и $\boldsymbol{\ell}_2 = (\delta_2\xi, \delta_2\eta)$, их ортогональность, $(\boldsymbol{\ell}_1 \cdot \boldsymbol{\ell}_2) = \delta_1\xi \cdot \delta_2\xi + \delta_1\eta \cdot \delta_2\eta = 0$, в силу условий (2), так же означает ортогональность и соответствующих им векторов приращений в пространстве (x, y). Действительно, если

$$\begin{array}{rcl} \boldsymbol{n}_1 = (\delta_1 x, \delta_1 y) & = & (\partial_\xi x\,\delta_1\xi + \partial_\eta x\,\delta_1\eta\,,\ \partial_\xi y\,\delta_1\xi + \partial_\eta y\,\delta_1\eta) \\ \boldsymbol{n}_2 = (\delta_2 x, \delta_2 y) & = & (\partial_\xi x\,\delta_2\xi + \partial_\eta x\,\delta_2\eta\,,\ \partial_\xi y\,\delta_2\xi + \partial_\eta y\,\delta_2\eta)\ , \end{array} \tag{3}$$

то прямой подстановкой можно показать, что $(\boldsymbol{n}_1 \cdot \boldsymbol{n}_2) = 0$. При этом заметим, что вектора приращений $\boldsymbol{\ell}_1$, $\boldsymbol{\ell}_2$, $\boldsymbol{n}_1$, $\boldsymbol{n}_2$ могут иметь произвольное направление, не обязательно вдоль координатных линий. Перекрестным дифференцированием условий (2) легко показать, что, для того чтобы они выполнялись, необходимо, чтобы функции $x = x(\xi, \eta)$ и $y = y(\xi, \eta)$ были гармоническими,

$$x_{\xi\xi} + x_{\eta\eta} = 0 \quad \text{и} \quad y_{\xi\xi} + y_{\eta\eta} = 0\,, \tag{4}$$

и, таким образом, задачу заполнения криволинейного контура конформной ортогональной сеткой можно свести к решению задачи Дирихле в прямоугольной области на плоскости (ξ, η), однако для этого по-прежнему необходимо, чтобы координаты $(x, y) = \big(x(\xi, \eta), y(\xi, \eta)\big)$ были заданы на границе области $\partial\mathscr{D}$ вполне определенным образом из конформного преобразования. В свою очередь, это означает, что необходимо расположить точки-узлы будущей сетки на криволинейном контуре так, чтобы в результате *прямого конформного преобразования* контура на прямоугольник эти точки оказались на *одинаковом* расстоянии друг от друга, с тем, чтобы на них можно было построить прямоугольную декартову сетку (рис. 1). После того, как такое распределение узлов найдено, оно используется в качестве краевых условий дискретной задачи Дирихле для $x = x_{i,j}$, $y = y_{i,j}$, решение которой и является искомым заполнением контура ортогональной криволинейной сеткой.

Резюмируя вышесказанное, построение сетки состоит из следующих шагов: построение карты района моделирования; построение контура; нахождение распределения узлов сетки на контуре; решение задачи Дирихле; обратное преобразование к сферическим координатам; вычисление метрических коэффициентов сетки.

2. Построение ортогональной криволинейной сетки

Разработанная нами программа `izogrid`[1] для построения сеток устроена таким образом, что её можно остановить после каждого шага, установив значение входного параметра `mode=0, 1, 2, 3, 4` или `5`. После её выполнения получается графический файл, по которому можно судить о соответствии результата последней операции (в зависимости от значения `mode`) желаемым требованиям, и, при необходимости, сделать соответствующие корректировки входных параметров. Это сделано для удобства, так как первые два шага подразумевают ручную настройку параметров, и, соответственно, большое количество повторений. Но вместе с тем они выполняются достаточно быстро – на практике это от долей до единиц секунд, что существенно ускоряет процесс подбора. Дальнейшие же шаги, `mode=2,...,5`, наоборот, гораздо более вычислительно затратны, но не требуют большого числа повторений и не имеют смысл, пока оптимальные значения параметров не найдены. Если `mode=4` или `5`, то создаётся файл сетки в `netCDF` формате, который и является конечной целью. В случае `mode=5`, все вычислительные стадии такие же, как и для `4`, но графический файл не создаётся. Эта возможность полезна для создания сеток высокого разрешения, т.к. в этом случае графический файл не имеет практической ценности из-за слияния линий, сам он оказывается очень большим, а его создание требует значительного процессорного времени. Заметим, что большинство рисунков в этой части статьи созданы самой программой, не требуя никаких дополнительных действий, редактирования и т.д.

Далее в этой части мы проследим всю последовательность операций шаг за шагом, используя в качестве реалистического примера построение ортогональной криволинейной сетки изотропного разрешения для модели Черного моря.

2.1. Привязка к географической области и задание системы координат

Для начала, `mode=0`, необходимо построить плоскую карту выбранной географической области при помощи обратимого конформного преобразования сферы на плоскость. Такими отображениями могут быть проекции Меркатора (в том числе её варианты с поворотом экватора), Ламберта, или стереографическая (Snyder, 1987). Предпочтение отдается проекции, которая даёт минимальные геометрические искажения, но строгого требования или использования каких-либо малых параметров, связанных с этим, рассматриваемый алгоритм не накладывает.

Для построения сетки Чёрного моря (рис. 2) мы использовали повернутую проекцию Меркатора, так, что точка пересечения Гринвичского меридиана с экватором была приведена в точку с координатами 43.75N, 34.50E, которая находится в центре моря. Азимутального вращения относительно этой точки не производилось (но, в принципе, наша процедура подразумевает такую возможность). Географическая сетка с ячейками 2 градуса широты и долготы нанесена для наглядности. Береговая линия построена с использованием данных GSHHS с максимальным доступным раз-

[1] Ives & Zakharias, isotropic-resolution grid

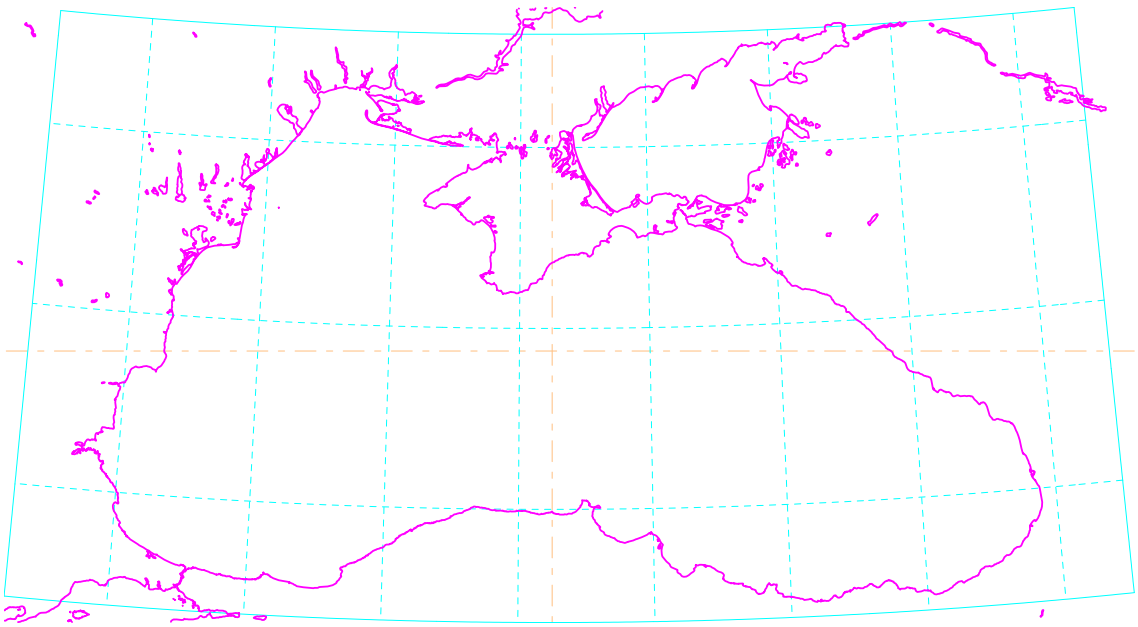


Рис. 2. Карта Чёрного моря в повёрнутой проекции Меркатора. Горизонтальная и вертикальная штрих-пунктирные прямые соответствуют «экватору» и центральному меридиану проекции (для обычной проекции Меркатора это настоящий экватор и меридиан Гринвича).

решением,

`http://www.ngdc.noaa.gov/mgg/shorelines/data/gshhs/latest/`.

Далее, `mode=1`, вводится декартова система координат, изображенная на рис. 3 как «миллиметровка», которая соответствует координатам проекции, для удобства умноженным на масштабирующий множитель с тем, чтобы привести значения чисел в диапазон примерно от -100 до $+100$ по порядку величины (начало этой системы координат всегда совпадает с центром проекции, пересечение штрих-пунктирных линий на рис. 2, поэтому ожидается, что максимальные положительные и отрицательные величины примерно одинаковы). Это масштабирование так же определяет шаг миллиметровки: он должен быть не слишком грубым, чтобы обеспечить достаточное разрешение, но и не слишком мелким, чтобы слишком частые линии не сливались. После этого необходимо задать координаты точек для построения контура криволинейной сетки. На рис. 3 эти точки помечены красными стрелками и красными уголками для угловых точек. Число точек, не связанных с углами, может быть произвольным (в том числе они могут отсутствовать совсем), но число угловых точек должно быть всегда четыре. Сами же точки упорядочены и пронумерованы, двигаясь против часовой стрелки, начиная с юго-западного угла.

На практике пользователь редактирует небольшой файл (для данного примера он приведен в описании рис. 3), задавая координаты точек и каждый раз проверяя получающийся контур (подробное описание всех управляющих параметров во входном файле приведено в Дополнении 1 в конце статьи). Целью является окружить геометрический объект как можно точнее, но при этом избегая нежелательных изломов

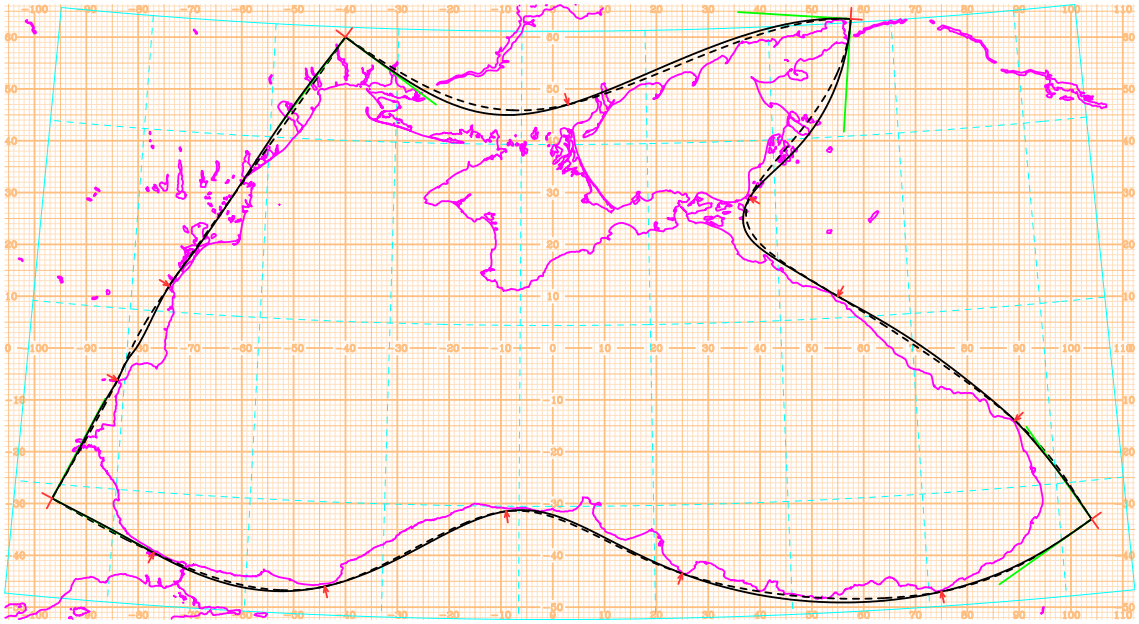


Рис. 3. Декартова сетка и периметр будущей криволинейной сетки. Красными стрелками показаны опорные точки контурного сплайна, координаты которых задаются пользователем интерактивно вручную. В данном случае используются всего 15 точек, их координаты, равно как и все остальные параметры, заданы во входном файле программы, который полностью приведен ниже. Так же, для сравнения, показаны оба варианта контура, построенные кубическим (пунктирная линия) и сплайном пятого порядка (сплошная).

```
mode=1 latlongrid=2 spline_type=4 npass=4      выбор режима
proj=ME rlat=43.75 rlon=34.5 rota=0            параметры проекции карты
 west_edge=26.5     east_edge=43               географические пределы карты
 south_edge=40.75   north_edge=47.25
nx=65 ny=50                                    размерность будущей сетки
uscale=0.001                                   масштабирующий множитель
---                                            метка конца преамбулы
   -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
   +38     +29
   +57.5   +63.5   <  north-east
   +3      +47
   -40     +60     <  north-west
   -74     +12
   -84     -6.2
```

декартовы координаты опорных точек сплайна

контура и/или избыточной кривизны. Для построения самого контура используется сплайн-интерполяция – кубическая или полиномами пятого порядка. На рис. 3 показаны контуры, построенные обоими типами сплайнов (для этого нужно установить `spline_type=4` («посередине» между 3 и 5 для кубического и пятого порядка). Эти контуры почти совпадают, из-за чего может создаться впечатление, что кубического сплайна вполне достаточно. На самом деле, их близость означает удачное, тщательно подобранное расположение опорных точек. Дело в том, что сплайн пятого порядка обеспечивает более высокую степень гладкости, но вместе с тем он гораздо более капризен из-за тенденции к появлению колебаний и резких перегибов. Поэтому на практике оказалось полезным иметь оба типа сплайнов одновременно именно для того, чтобы обнаружить и подавить эти тенденции.

В случае кубического сплайна задача сводится к решению трёхдиагональной системы линейных уравнений. Неизвестными являются значения производных

$$d_k^{(x)} = \partial x/\partial s|_{s=s_k} \quad \text{и} \quad d_k^{(y)} = \partial y/\partial s|_{s=s_k} \tag{5}$$

в опорных точках сплайна $(x_k, y_k)\,,\ k = 1, ..., N$, где s это координата вдоль кривой контура (её определение будет рассмотрено ниже). Уравнения для вычисления $d_k^{(x)}$ и $d_k^{(y)}$ получаются из условий равенства правых и левых пределов вторых производных,

$$\lim_{s\to s_k} {}^{\partial^2 x}/_{\partial s^2} = \lim_{s_k\leftarrow s} {}^{\partial^2 x}/_{\partial s^2} \quad \text{и} \quad \lim_{s\to s_k} {}^{\partial^2 y}/_{\partial s^2} = \lim_{s_k\leftarrow s} {}^{\partial^2 y}/_{\partial s^2} \tag{6}$$

вычисленных в последовательных сегментах $[s_{k-1}, s_k]$ и $[s_k, s_{k-1}]$ для всех $k = 2, ..., N-1$. Для оставшихся неизвестных с индексами $k = 1$ и $k = N$ эти условия неприменимы и требуются специальные уравнения, которые получаются из граничных условий, рассматриваемых ниже. В случае полиномов пятой степени, ситуация в целом аналогична, за исключением того, что неизвестными являются уже пары чисел – значений первой и второй производных, а получающая система носит блочный характер, где вместо обычных скалярных коэффициентов появляются матрицы 2×2. И в том, и в другом случае применим прямой метод решения. Более подробно построение сплайнов, кубических и пятого порядка, рассмотрено в Дополнении 2.

2.2. Ортогональность сторон контурного сплайна

Особое внимание стоит уделить тому, как добиться *точной ортогональности* сторон криволинейной сетки в её углах. Как сказано выше, в «прямых» точках s_k математическими условиями построения сплайна является непрерывность самих функций, $x = x(s)$ и $y = y(s)$, где s – это координата вдоль кривой контура, а так же их первых и вторых производных по s (в случае кубического сплайна), либо всех производных до четвертого порядка включительно (для сплайнов пятого порядка).

В угловых точках условия на производные получаются из принципа, что если сегмент контура после угловой точки повернуть на угол 90° по часовой стрелке относительно этой угловой точки, то угол «выпрямится» и должна получиться плавная кривая с теми же самыми свойствами непрерывности производных (вплоть до второй

или четвертой включительно), что и в случае «прямой» точки (рис. 4). Заметим, что сегмент ломаной $[N{+}1, N]$, приближающийся сверху к левому нижнему углу контура (т.е. к углу, который на рис. 4 был разорван – точки $k = 1$ и $k = N + 1$ изначально были тождественны), оказался повернут трижды на 90° против часовой стрелки, а это значит, что касательная к контуру в угловой точке повернулась на такой же угол, и чтобы соблюсти ортогональность касательных в левом нижнем углу, необходимо поставить условие, что после всех поворотов обе касательные окажутся параллельны

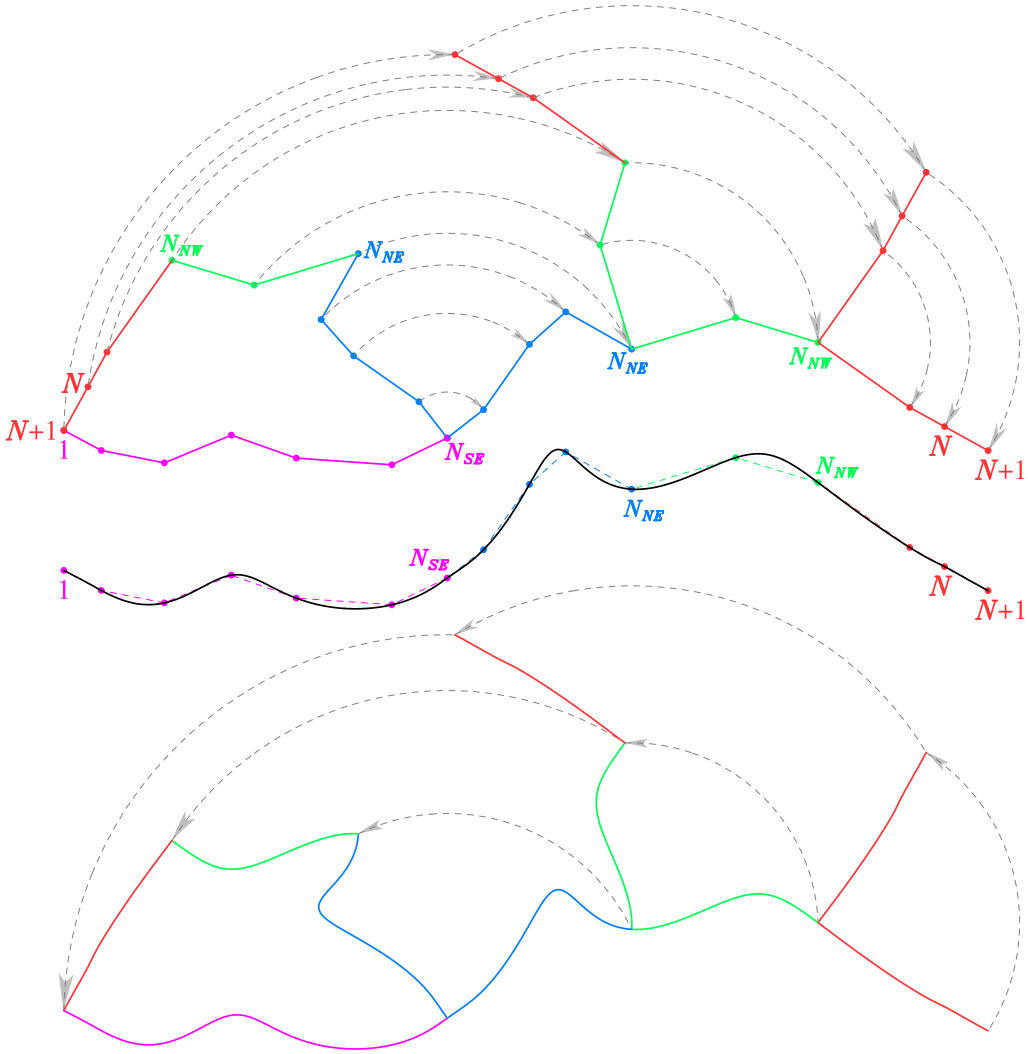


Рис. 4. Построение криволинейного контура с точными прямыми углами сопряжения сторон. Замкнутая ломаная линия (*верхняя панель слева*) проведена по точкам заданным пользователем. Её левый нижний угол размыкается, и три стороны контура (кроме нижней) твердотельным вращением поворачиваются на угол 90° по часовой стрелке – таким образом спрямляется правый нижний угол. Далее аналогичным образом выпрямляются два остальных угла. Через получившиеся точки проводится сплайн $(x, y) = \big(x(s), y(s)\big)$, построенный с учётом периодических граничных условий на концах (*средняя панель*). Обратное преобразование трёх твёрдотельных вращений превращает плавную кривую в замкнутый контур, у которого углы примыкания криволинейных сторон в точности равны 90° (*нижняя панель*).

друг другу. Это выполнится автоматически, если поставить периодические условия на производные на концах. При этом, в силу периодичности, последний и первый сегменты, $[s_N, s_1]$ и $[s_1, s_2]$, считаются последовательными, в том же самом смысле, как и все остальные сегменты, $[s_{k-1}, s_k]$ и $[s_k, s_{k+1}]$, для $k = 2, ..., N-1$, а условие выпрямления разорванного угла *математически ничем не отличаются* от трёх остальных. Фактически это означает, что уравнения для $k = 1$ и для $k = N$ точно такие же, как и для остальных $k = 2, ..., N-1$, за исключением того, что в первом случае, $k = 1$, все величины с индексом $k-1$ заменяется на аналогичные с индексом N; а во втором, $k = N$, величины с $k+1$ заменяется на аналогичные с индексом 1. Таким образом, задача построения криволинейного контура с прямыми углами сопряжения сводится к построению пары одномерных сплайнов с периодическими граничными условиями, соответственно сводится к решению трёхдиагональной (или блочной трёхдиагональной) системы линейных уравнений с условиями *циклического замыкания*. При этом сама процедура вычисления производных сплайна ничего не знает о том, что спрямлённые точки когда-то были углами. Отметим, что несмотря на кажущуюся очевидность этой идеи (рис. 4), ни в одном из известных пакетов для построения сеток подобный алгоритм никогда не применялся.

В качестве координаты s в простейшем случае можно использовать формальную координату $\{s_k = k \,|\, k = 1, ..., N\}$ в опорных точках сплайна – этого достаточно, чтобы вычислить все необходимые производные $d_k^{(x)}$ и $d_k^{(y)}$ в этих точках. Соответственно, дробная часть $s - s_k$ где $k = s_k < s < s_{k+1} = k+1$ используется для интерполяции между точками k и $k+1$. Детальное обоснование такого подхода рассматривается в Дополнении 2, начиная с параграфа содержащего ур. (74).

Альтернативой является использование «настоящей» длины пути вдоль кривой. В этом случае вектор $\boldsymbol{\ell} = (\partial x/\partial s, \partial y/\partial s)$ является настоящим единичным вектором касательной к кривой $(x, y) = \big(x(s), y(s)\big)$ в каждой её точке. Однако некоторое затруднение здесь вызывает тот факт, что для того, чтобы вычислить длину пути вдоль кривой, сама кривая должна быть определена, а в данный момент про неё ничего не известно кроме того, что она проходит через точки (x_k, y_k). Эта дилемма разрешается следующим образом: в качестве начального приближения для s_k берутся длины отрезков ломаной, проведённой через опорные точки: определим

$$\begin{aligned}
&\Delta s_{k+1/2} = \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}\,, && k = 1, ..., N-1 \\
&\Delta s_{N+1/2} = \sqrt{(x_1 - x_N)^2 + (y_1 - y_N)^2}\,, && k = N \\
&s_1 = 0 \\
&s_k = s_{k-1} + \Delta s_{k-1/2} \quad \text{последовательно для всех} && k = 2, ..., N+1.
\end{aligned} \tag{7}$$

При этом конечная величина s_{N+1} является полной длиной замкнутой ломаной, т.к. точка $(x_{+1}, y_{N+1}k)$ совпадает с точкой (x_1, y_1), рис. 4. Далее, на основании этих s_k строится сплайн и отрезки кривых между последовательными парами (x_k, y_k) и (x_{k+1}, y_{k+1}) заполняются интерполированными точками. После этого длины этих отрезков кривых вычисляются аналогичным образом, но, используя уже координаты интерполированных точек, которых во много раз больше чем опорных, и, соответ-

ственно, они дают гораздо более точное приближение к истинной длине кривой. Получаются уточнённые значения s_k. Снова строится сплайн, затем получаются новые значения интерполированных точек, и снова уточняются s_k, и т.д. Этот процесс сходится довольно быстро, и достаточно всего нескольких итераций, чтобы достичь ошибок на уровне округления двойной точности.

Алгоритм на рис. 4 работает следующим образом: входными данными является последовательность координат опорных точек, $\{(x_k, y_k), \quad k = 1, ..., N\}$, четыре из них, $k = 1, N_{SE}, N_{NE}, N_{NW}$ являются угловыми, юго-западный, юго-восточный, северо-восточный, и северо-западный углы, соответственно. Для построение сплайна (т.е. вычисления производных в самих этих точках) нужны не сами координаты, а лишь последовательные разности. Поэтому определим,

$$\begin{array}{l} \Delta x_{k+1/2} = x_{k+1} - x_k \\ \Delta y_{k+1/2} = y_{k+1} - y_k \end{array} \quad k = 1, .., N-1 \quad \text{и} \quad \begin{array}{l} \Delta x_{N+1/2} = x_1 - x_N \\ \Delta y_{N+1/2} = y_1 - y_N \, . \end{array} \tag{8}$$

Заметим что на рис. 4, последняя точка $k = N$ не является угловой, а не дотягивается до юго-западного угла, т.е. последний сегмент приходится вычислять отдельно. Далее «распрямим» ломаную (рис. 4, *средняя панель*), повернув соответствующие сегменты на 90, 180, и 270 градусов по часовой стрелке – это преобразование сводится к взаимным заменам $\Delta x \leftrightarrow \Delta y$ с изменением знака у одной из них,

$$\begin{array}{ll} \begin{array}{l} \Delta x'_{k+1/2} = \Delta x_{k+1/2} \\ \Delta y'_{k+1/2} = \Delta y_{k+1/2} \end{array} & k = 1, ..., N_{SE} - 1 \\ \\ \begin{array}{l} \Delta x'_{k+1/2} = +\Delta y_{k+1/2} \\ \Delta y'_{k+1/2} = -\Delta x_{k+1/2} \end{array} & k = N_{SE}, ..., N_{NE} - 1 \\ \\ \begin{array}{l} \Delta x'_{k+1/2} = -\Delta x_{k+1/2} \\ \Delta y'_{k+1/2} = -\Delta y_{k+1/2} \end{array} & k = N_{NE}, ..., N_{NW} - 1 \\ \\ \begin{array}{l} \Delta x'_{k+1/2} = -\Delta y_{k+1/2} \\ \Delta y'_{k+1/2} = +\Delta x_{k+1/2} \end{array} & k = N_{NW} - 1, ..., N \, . \end{array} \tag{9}$$

Полученные «штрихованные» разности входят в правую часть уравнений, для построения сплайнов (см. детальное описание в Дополнение 2),

$$\begin{array}{rcl} \boldsymbol{a}_k \boldsymbol{d}^{(x\prime)}_{k-1} + \boldsymbol{b}_k \boldsymbol{d}^{(x\prime)}_{k} + \boldsymbol{c}_k \boldsymbol{d}^{(x\prime)}_{k+1} & = & \boldsymbol{f}\left(\Delta x'_{k-1/2}, \Delta x'_{k+1/2}\right) \\ \boldsymbol{a}_k \boldsymbol{d}^{(y\prime)}_{k-1} + \boldsymbol{b}_y \boldsymbol{d}^{(y\prime)}_{k} + \boldsymbol{c}_k \boldsymbol{d}^{(y\prime)}_{k+1} & = & \boldsymbol{f}\left(\Delta y'_{k-1/2}, \Delta y'_{k+1/2}\right) \, , \end{array} \tag{10}$$

где неизвестными являются $\boldsymbol{d}^{(x\prime)}_k$ и $\boldsymbol{d}^{(y\prime)}_k$. В случае *кубических сплайнов* это производные, $\boldsymbol{d}^{(x\prime)}_k = \partial x'/\partial s\big|_k$ и $\boldsymbol{d}^{(y\prime)}_k = \partial y'/\partial s\big|_k$. Находящиеся в левой части $\boldsymbol{a}_k$, $\boldsymbol{b}_k$, $\boldsymbol{c}_k$ – это коэффициенты (просто числа, известные на данный момент), а $\boldsymbol{f}$ в правой части – взвешенные суммы своих аргументов (известны). Если же применяются *сплайны пятого порядка*, то $\boldsymbol{d}^{(x\prime)}_k = \left(d^{(x\prime)}_k, \delta^{(x\prime)}_k\right) = \left(\partial x'/\partial s\big|_k, \partial^2 x'/\partial s^2\big|_k\right)$ и

$\boldsymbol{d}_k^{(y\prime)} = \left(d_k^{(y\prime)}, \delta_k^{(y\prime)}\right) = \left(\partial y'/\partial s\big|_k, \partial^2 y'/\partial s^2\big|_k\right)$ – это векторы составленные из двух элементов – первых и вторых производных «штрихованных» координат x' и y' соответственно; $\boldsymbol{a}_k$, $\boldsymbol{b}_k$, $\boldsymbol{c}_k$ – матрицы размером 2×2, состоящие из коэффициентов (известны), $\boldsymbol{f}$ вектор-функция из двух элементов. Уравнения (10) применимы для всех индексов $k = 1, ..., N$, при условии циклической замены: если $k = 1$, то индекс $k - 1$ заменяется на N, а если $k = N$ то $k + 1$ заменяется на 1. Таким образом, верхняя и нижняя строчки (10), взятые по отдельности, представляют собой две замкнутые системы N уравнений относительно N неизвестных (чисел или векторов). Они могут быть решены независимо друг от друга, т.к. их неизвестные не пересекаются. Коэффициенты (или матрицы коэффициентов) для обоих уравнений одни и те же. Конкретные значения коэффициентов и правых частей зависят от того, как определена координата вдоль кривой s, но алгоритмически ход решения одинаков для всех вариантов – решение трёхдиагональной системы с условиями циклического замыкания.

После того как (10) разрешены, $d_k^{(x\prime)}$, $d_k^{(y\prime)}$ становятся известными, так же как и вторые производные $\delta_k^{(x\prime)}$ и $\delta_k^{(y\prime)}$, в случае сплайнов пятого порядка. Эти величины нужно вернуть из «штрихованного» пространства в обычное, (т.е. свернуть линию на *средней панели* рис. 4 обратно в замкнутый контур), сделав преобразование обратное к (9). Одновременно с ним, мы развернем направление на северной и западной сторонах контура, т.к. здесь обход контура против часовой стрелки (возрастание s) соответствует движению против положительных направлений криволинейных координат ξ и η. Получим:

$k = 1, ..., N_{SE}$ вдоль южной стороны контура,

$$\begin{array}{lll} x_{Sk} = x_k & d_{S\ k}^{(x)} = d_k^{(x\prime)} & \delta_{S\ k}^{(x)} = \delta_k^{(x\prime)} \\ y_{Sk} = y_k & d_{S\ k}^{(y)} = d_k^{(y\prime)} & \delta_{S\ k}^{(y)} = \delta_k^{(y\prime)} \end{array} \tag{11}$$

$k = N_{SE}, ..., N_{NE}$, вдоль восточной стороны,

$$\begin{array}{lll} x_{Ek-N_{SE}+1} = x_k & d_{E\ k-N_{SE}+1}^{(x)} = -d_k^{(y\prime)} & \delta_{E\ k-N_{SE}+1}^{(x)} = -\delta_k^{(y\prime)} \\ y_{Ek-N_{SE}+1} = y_k & d_{E\ k-N_{SE}+1}^{(y)} = +d_k^{(x\prime)} & \delta_{E\ k-N_{SE}+1}^{(y)} = -\delta_k^{(x\prime)} \end{array} \tag{12}$$

$k = N_{NE}, ..., N_{NW}$, вдоль северной стороны,

$$\begin{array}{lll} x_{N\,N_{NW}+1-k} = x_k & d_{N\ N_{NW}+1-k}^{(x)} = d_k^{(x\prime)} & \delta_{N\ N_{NW}+1-k}^{(x)} = -\delta_k^{(x\prime)} \\ y_{N\,N_{NW}+1-k} = y_k & d_{N\ N_{NW}+1-k}^{(y)} = d_k^{(y\prime)} & \delta_{N\ N_{NW}+1-k}^{(y)} = -\delta_k^{(y\prime)} \end{array} \tag{13}$$

$k = N_{NW}, ..., N$, вдоль западной стороны, кроме последней точки,

$$\begin{array}{lll} x_{W\,N+2-k} = x_k & d_{W\ N+2-k}^{(x)} = -d_k^{(y\prime)} & \delta_{W\ N+2-k}^{(x)} = \delta_k^{(y\prime)} \\ y_{W\,N+2-k} = y_k & d_{W\ N+2-k}^{(y)} = +d_k^{(x\prime)} & \delta_{W\ N+2-k}^{(y)} = \delta_k^{(x\prime)} \end{array} \tag{14}$$

$k = 1$, замыкаем последний сегмент на западной стороне,

$$\begin{array}{lll} x_{W1} = x_1 & d_{W\ 1}^{(x)} = -d_1^{(y\prime)} & \delta_{W\ 1}^{(x)} = \delta_1^{(y\prime)} \\ y_{W1} = y_1 & d_{W\ 1}^{(y)} = +d_1^{(x\prime)} & \delta_{W\ 1}^{(y)} = \delta_1^{(x\prime)} . \end{array} \tag{15}$$

Для каждой из сторон, $R \in \{S, E, N, W\}$, мы получили четыре (или шесть) последовательностей x_{Rk}, y_{Rk}, $d^{(x)}_{R\ k}$, $d^{(y)}_{R\ k}$, $\delta^{(x)}_{R\ k}$ и $\delta^{(y)}_{R\ k}$, где $k = 1, ..., N_R$ (заметим, что индексы в левой части уравнений (11)–(15) или начинаются с единицы, или приходят к единице, числа точек N_R определены как $N_S = N_{SE}$, $N_E = N_{NE} - N_{SE} + 1$, $N_N = N_{NW} - N_{NE} + 1$ и $N_W = N - N_{NW} + 2$). Эти последовательности есть координаты опорных точек сплайна, их первые и вторые производные вдоль кривой, в *положительном направлении* соответствующей координаты, а крайние индексы $k = 1$ и $k = N_R$ соответствуют угловым точкам. Это позволяет строить сплайн-интерполяцию *отдельно на каждой стороне контура*, но при этом соблюдать точный угол сопряжения двух криволинейных сторон $90°$, т.к. производные в угловых точках определены дважды (для каждой из примыкающих сторон), но так, что $d^{(x)}_{R\ k}$ и $d^{(y)}_{R'\ k}$, приходящие с разных сторон, связаны преобразованием поворота на $90°$.

В заключении этой части стоит проиллюстрировать некоторые свойства построенных таким образом контуров (рис. 5). Если угловые точки разместить точно в углах прямоугольника, то получится прямоугольник, стороны которого в точности прямые. Однако, если сместить хотя бы одну из вершин, то не только две прилежащие, но все четыре стороны, становятся в той или иной степени криволинейными. То же самое произойдет при попытке изогнуть одну из сторон прямоугольника, задав

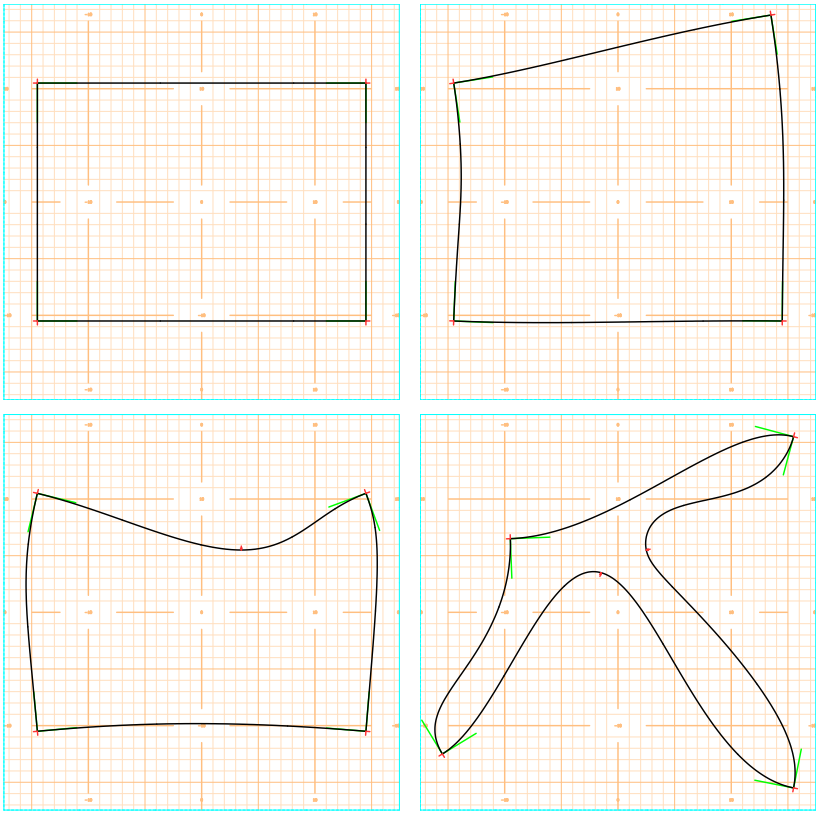


Рис. 5. Некоторые тестовые задачи для алгоритма построения контура. Зелеными линиями показаны касательные к контуру в угловых точках. По взаимному положению зеленых и черных линий можно судить о кривизне контура при подходе в угловую точку.

дополнительную точку. При этом все четыре угла сопряжения сторон всегда остаются строго перпендикулярными *no matter what*: даже в таком экстремальном случае, как на *правой нижней панели* (рис. 5) пользователю не нужно предпринимать никаких специальных мер, чтобы обеспечить эту перпендикулярность. Так же на рис. 5 стоит заметить свойство «сохранение кривизны» в угловых точках контура: обе сопряженные стороны либо выпуклы наружу, либо вогнуты внутрь. Это связано с непрерывностью вторых производных развернутого сплайна (рис. 4). Ситуация, когда одна из сторон выпукла наружу, а другая вогнута внутрь, в принципе возможна, но это означает, что вторые производные меняют знак в угловой точке, при этом плавно переходя через ноль в силу их непрерывности, т.е. кривизна в этом случае исчезающе мала.

2.3. Конформное преобразование

К настоящему моменту периметр будущей сетки определён в том смысле, что все необходимые производные в опорных точках сплайна вычислены, равно как известны и формулы интерполяции – всё это в совокупности однозначно определяет криволинейный контур как геометрическое положение *всех возможных точек* $(x, y) = (x(s), y(s))$, координаты которых *можно в принципе* построить этим сплайном. При этом остается нерешенным вопрос: *как именно должны быть расположены* краевые точки-узлы будущей сетки вдоль этого контура?

Это расположение не произвольно. Например, если мы распределим точки равномерно вдоль каждой стороны контура (таким образом зададим x и y на границе области), и решим задачи Дирихле для x и y с соответствующими граничными условиями, мы получим сетку, показанную на рис. 6. Эта сетка не ортогональна, несмотря на тот факт, что оба уравнения Лапласа $x_{\xi\xi} + x_{\eta\eta} = 0$ и $y_{\xi\xi} + y_{\eta\eta} = 0$ решены точно (на уровне ошибок округления двойной точности). Здесь (ξ, η) это по сути «индексные» координаты с возможными геометрическими множителями, определяющими соотношение приращений, соответствующих изменению своих индексов на единицу, $\Delta\xi = \xi_{i+1,j} - \xi_{i,j}$ и $\Delta\eta = \eta_{i,j+1} - \eta_{i,j}$, причем $\Delta\xi = const$ и $\Delta\eta = const$ однородны на всей сетке, $\forall(i, j) \in \{[1, ..., L] \times [1, ..., M]\}$. При этом заметим, что для возникающих здесь дискретных уравнений Лапласа,

$$\frac{x_{i+1,j} - 2x_{i,j} + x_{i-1,j}}{\Delta\xi^2} + \frac{x_{i,j+1} - 2x_{i,j} + x_{i,j-1}}{\Delta\eta^2} = 0\,, \tag{16}$$

и такого же уравнения для $y_{i,j}$, важны не сами величины $\Delta\xi$ и $\Delta\eta$, а лишь их отношение, $\Delta\xi/\Delta\eta$, которое совершенно не понятно из каких условий можно определить: варьирование этого отношения, равно как и варьирование отношения числа точек вдоль сторон сетки L/M, если зафиксировать $\Delta\xi/\Delta\eta = 1$, показывает, что результат на рис. 6 чувствителен к этим отношениям, но ни при каком выборе сетка не получается ортогональной. Более того, на рис. 6 в восточной части, где сильная выпуклость контура в сторону моря (вблизи Таманского полуострова), сетка вывернулась наизнанку. Ячейки сетки в этом месте сильно вытянуты с севера на юг, и если

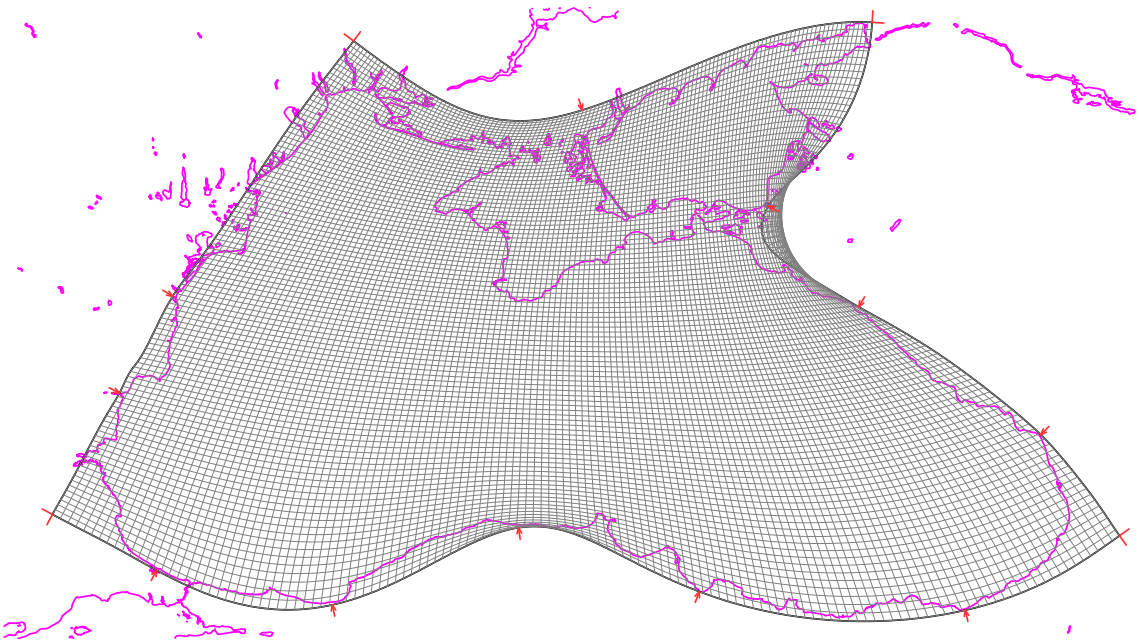


Рис. 6. Сетка, полученная заполнением каждой из сторон контура равноудаленными точками (их количество одинаково на противоположных сторонах), с последующим решением задачи Дирихле внутри области. Очевидно, что эта сетка не является ортогональной.

увеличить число точек в направлении с юга на север, чтобы сделать их менее вытянутыми, то сетка перестанет быть вывернутой в этом месте, но тогда точно такая же проблема появится вблизи выпуклости контура на южной стороне, где ячейки и так вытянуты с запада на восток, а в результате попытки увеличить число точек с юга на север, окажутся ещё более вытянутыми.

Причина этой неудачи в том, что для того, чтобы решение задачи Дирихле гарантировало ортогональность сетки, необходимо, чтобы пространство (x, y) и пространство (ξ, η) были *связаны конформным преобразованием*. Тогда прямому углу любой ориентации в пространстве (ξ, η) будет соответствовать прямой угол в пространстве (x, y), а бесконечно малому прямоугольнику в (ξ, η) будет соответствовать прямоугольник с таким же соотношением сторон в пространстве (x, y). Поэтому если сетка в пространстве (ξ, η) декартова, то ей будет соответствовать ортогональная криволинейная сетка в пространстве (x, y), а если ещё сделать так, чтобы ячейки сетки в (ξ, η) были квадратными, $\Delta\xi = \Delta\eta$, то сетка в (x, y) будет обладать свойством изотропии разрешения – расстояния между соседними точками в обоих направлениях равны друг другу, несмотря на то, что сами эти расстояния непостоянны и зависят от местоположения на сетке.

Область произвольной формы на комплексной плоскости можно конформно отобразить на верхнюю полуплоскость преобразованием Шварца–Кристоффеля. Наиболее простым случаем является полубесконечное пространство, находящееся по одну сторону от несамопересекающейся ломаной линии, концы которой уходят в бесконечность – в этой ситуации применима формула Кристоффеля, представляю-

щая собой совокупность элементарных преобразований выпрямления угла в прямую линию. Конформное отображение области, образованной несамопересекающейся замкнутой ломаной произвольной формы в прямоугольник, возможно применением алгоритма IZ89, в основе которого используется то же самое элементарное преобразование выпрямления углов, но в целом ситуация более сложная, потому что этот алгоритм уже не прямого действия, а требует итерационного подхода. Детальное описание алгоритма IZ89 приведено в их оригинальной статье, мы лишь ограничимся иллюстрацией принципов его построения.

На рис. 7 показано отображение угла на полуплоскость степенной функцией комплексного переменного. Представим себе, что из точки z_k выходят два луча, проходящие через точки z_{k-1} и z_{k+1}, так, что образуется угол $\angle z_{k-1}z_kz_{k+1}$. Возведение комплексного числа в степень $z'-z_k=(z-z_k)^P$ увеличивает его аргумент в P раз, а поэтому, если выбрать $P=\pi/(\pi-\alpha)$, то все три точки, z_{k-1}, z_k и z_{k+1}, окажутся на одной прямой линии. Для удобства (что является достаточно произвольным выбором) возведение в степень можно дополнить поворотом на некоторый угол и масштабированием. На рис. 7 угол и масштаб выбраны таким образом, чтобы точка z_{k-1} не смещалась (очевидно, что точка z_k остается неподвижной при любых P, A и α). При этом расстояние между точками z_k и z_{k+1} сохраняется лишь в случае, если z_{k+1} и z_k равноудалены от z_k, $|z_{k+1}-z_k|=|z_k-z_{k-1}|$. Во всех остальных случаях оно увеличивается или уменьшается в зависимости от угла поворота α и соотношения длин $|z_{k+1}-z_k|\,/\,|z_k-z_{k-1}|$. Отметим, что z_k является *точкой ветвления* функции $z'=z'(z)$, а сама функция является многозначной, поэтому таким преобразованием можно отобразить лишь часть комплексной плоскости (например, область внутри или снаружи угла $\angle z_{k-1}z_kz_{k+1}$ на полуплоскость выше или ниже прямой), но не всю плоскость на плоскость.

Формула на рис. 7 имеет несколько непривычный, по сравнению с известным из литературы, вид (в частности с Ives & Zacharias, 1989), однако если её переписать в терминах модуля и аргумента,

$$z-z_k=|z-z_k|\cdot e^{i\gamma}\,, \tag{17}$$

то получается

$$z'-z_k=|z_k-z_{k-1}|\left(\frac{|z-z_k|}{|z_k-z_{k-1}|}\right)^{\pi/(\pi-\alpha)}\exp\left\{i\left[\beta+\frac{\pi}{\pi-\alpha}\cdot(\gamma-\alpha-\beta)\right]\right\} \tag{18}$$

т.е. $\gamma'=\beta+\dfrac{\pi}{\pi-\alpha}\cdot(\gamma-\alpha-\beta)$. В частности, легко видеть что

$$\begin{array}{llll} z_{k+1}-z_k=|z_{k+1}-z_k|\cdot e^{i(\alpha+\beta)} & \gamma=\alpha+\beta & \rightarrow & \gamma'=\beta \\ z_{k-1}-z_k=|z_k-z_{k-1}|\cdot e^{i(\pi+\beta)} & \gamma=\pi+\beta & \rightarrow & \gamma'=\pi+\beta \end{array} \tag{19}$$

Так же, переписав её в терминах $P=\pi/(\pi-\alpha)$, и избавившись от α подстановкой $\alpha=\pi(P-1)/P$, получим

$$z'-z_k=|z_k-z_{k-1}|\cdot\left(\frac{|z-z_k|}{|z_k-z_{k-1}|}\right)^P\cdot\exp\Big\{i\big[\beta+\pi+P(\gamma-\beta-\pi)\big]\Big\}\,. \tag{20}$$

Обратное преобразование получается простой заменой $P \to 1/P$, что самоочевидно

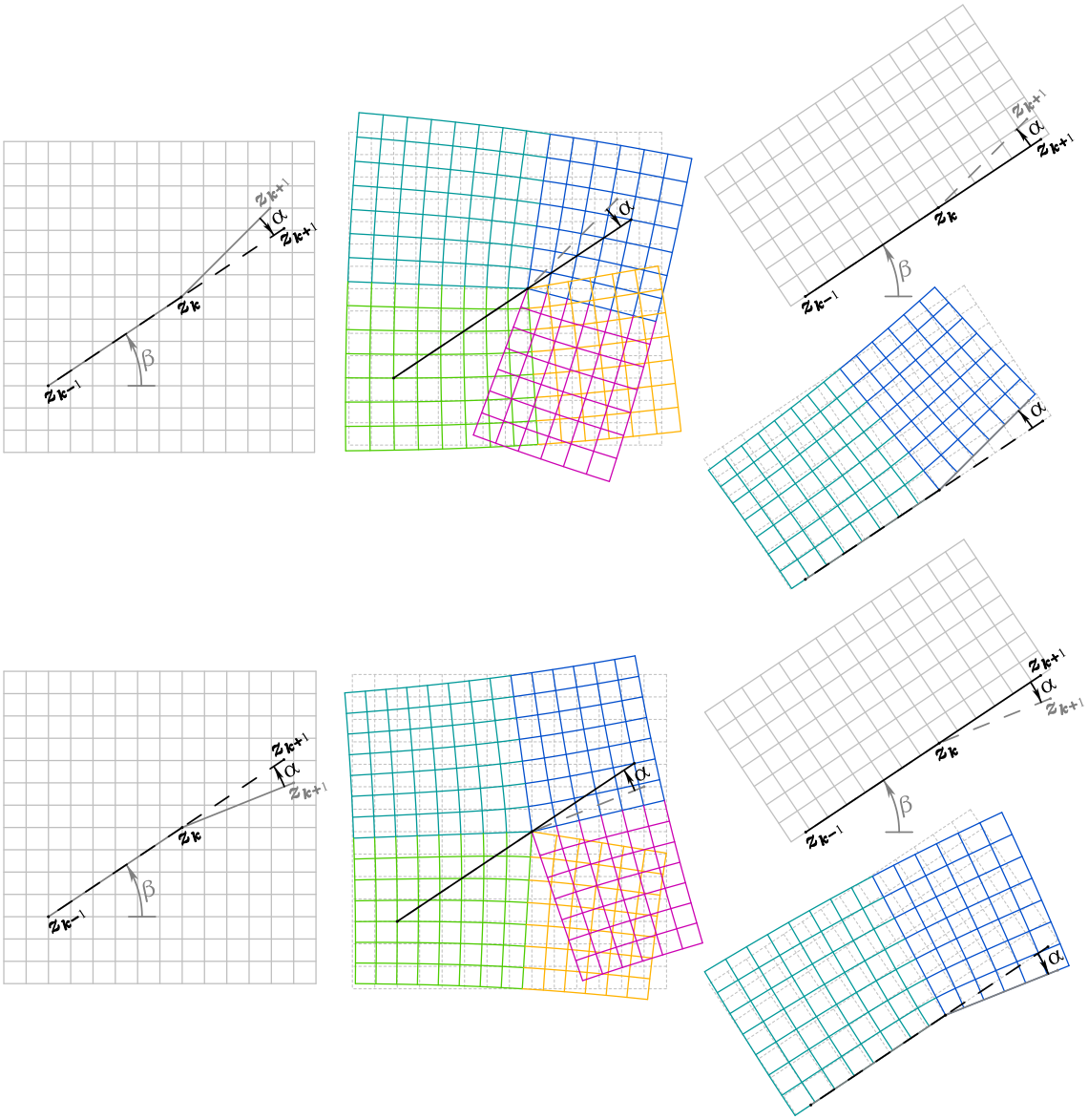


Рис. 7. Преобразование $z' = z_k + A \cdot e^{+i\beta} \cdot \left[e^{-i(\alpha+\beta)} \cdot (z - z_k)\right]^{\pi/(\pi-\alpha)}$, распрямляющее угол $\angle z_{k-1}\, z_k\, z_{k+1}$. Угловые множители $e^{+i\beta}$ и $e^{-i(\alpha+\beta)}$ делают направление β неизменным, а масштабирующий множитель $A = |z_k - z_{k-1}|^{1-\pi/(\pi-\alpha)}$ выбран так, чтобы расстояние между точками z_{k-1} и z_k не изменялось. *В средней колонке* показан результат преобразования равномерной сетки (*левая колонка*) в случае положительного ($\alpha > 0$, *сверху*) или отрицательного ($\alpha < 0$, *внизу*) начального угла изгиба. Отметим, что z_k является точкой ветвления функции $z' = z'(z)$, а сама функция является многозначной. *В правой колонке* показано искривление прямоугольной сетки, построенной уже в преобразованном пространстве (*верхний рисунок в каждой паре*) в результате обратного преобразования (*нижний в паре*).

для модуля, а для аргумента применим последовательно

$$\begin{aligned} \gamma' &= \beta + \pi + P \cdot (\gamma - \beta - \pi) \\ \gamma'' &= \beta + \pi + \frac{1}{P} \cdot (\gamma' - \beta - \pi) \end{aligned} \tag{21}$$

где, подставив выражение для γ' из первой строки во вторую, легко убедиться, что $\gamma'' = \gamma$ независимо от P и β. Так же, подставив $\gamma' = \beta$ во вторую строку, получим $\gamma'' = \beta + \pi - \pi/P = \beta + \pi - (\pi - \alpha) = \alpha + \beta$, т.е. направление β на точку z'_{k+1} поворачивается на угол α к исходному направлению на z_{k+1}.

В правой колонке на рис. 7 показана деформация равномерной прямоугольной сетки, построенной в пространстве z' обратным преобразованием в z в случае положительного ($\alpha > 0$, *сверху*) или отрицательного ($\alpha < 0$, *внизу*) начального угла поворота. Следует обратить внимание на то, что вдали от точки изгиба z_k узлы сетки распределены почти равномерно, однако по мере приближения к z_k они стягиваются к ней, либо наоборот стремятся избегать её, в зависимости от того, выпуклый или вогнутый край сетки. Далее мы увидим, что именно это свойство позволяет избегать выворачивания сетки вблизи выпуклостей контура в сторону моря, пример чего показан на рис. 6.

Так же стоит обратить внимание на то, что любой луч, выходящий из точки z_k в любом направлении, остаётся прямым и после преобразования выпрямления угла – это следует как из формул, приведённых выше, так и из рис. 7. Это свойство очень полезно, так так позволяет построить конформное преобразование ломаной линии в прямую, причем сделать это обратимым образом. На *верхнем фрагменте* рис. 8 показано, как это сделать. Исходная ломаная – это самая верхняя линия серого цвета. Её первый слева сегмент изначально горизонтален (если это не так, то не умаляя общности, всю ломаную можно повернуть как целое). Каждый сегмент, начиная со второго слева, поворачивается к горизонтали применением преобразования выпрямления угла при помощи возведения комплексного числа в степень (рис. 7) с соответствующим значением α (в этом примере $\beta = 0$). Масштабирующий множитель выбран так, что предыдущая точка не смещается (т.е. в точности всё как на рис. 7 – точки z_k и z_{k-1} неподвижны при выпрямлении изгиба в точке z_k). При этом так же поворачиваются и деформируются все сегменты, находящиеся правее, однако уже горизонтальные сегменты левее остаются горизонтальными при всех последующих преобразованиях выпрямления относительно точек правее к концу – может изменяться лишь длина этих сегментов: точки на прямой – это бывшие точки изгибов, и в данном примере, по их меняющемуся положению можно увидеть, как ломаная «поджимает» свой выпрямленный хвост по мере того, как последовательно выпрямляются углы справа.

В принципе, получив прямую линию в преобразованном пространстве, можно построить прямоугольную сетку так, чтобы по одной из осей её узлы совпали с точками выпрямленной ломаной, а по дугой оси выбрать узлы произвольно. Тогда обратное преобразование прямой в ломаную конформным образом превратит эту сетку в ортогональную криволинейную, причём так, что одна из сторон сетки будет совпадать с исходной ломаной.

При этом здесь очень просто построить обратное преобразование: в ходе прямого преобразования необходимо запомнить все степени P_k, для выпрямляемых на каждом шаге углов, $k = 2, ..., N-1$, а обратное преобразование будет точно таким же, только со степенями $1/P_k$, и будет происходить в обратной последовательности, начиная с правого конца, и двигаясь влево.

Однако получившаяся сетка в реальном пространстве будет иметь весьма неоднородное, произвольное распределение узлов, и будет малопригодной для моделирования. Дело в том, что все сегменты исходной ломаной на рис. 8 имеют одинаковую длину, а узлы на выпрямленной линии расположены неравномерно. Для построения хорошей сетки нужно наоборот – в этом случае, если в преобразованном пространстве построить прямоугольную сетку с равномерными распределением узлов, и желательно с ячейками квадратной формы, то соответствующая ей сетка в реальном пространстве будет обладать привлекательными свойствами.

Эта задача решается следующим образом: по точкам исходной ломаной строится сплайн (т.е. вычисляются производные в узловых точках, а сами точки становятся опорными точками сплайна), и далее эта ломаная интерпретируется как гладкая кривая: координаты точек, которые задают её как ломаную, можно интерполировать этим сплайном – фактически это означает, что каждая точка может двигаться вдоль кривой. Каждому набору интерполированных точек ставится в соответствие последовательность $\{s_k,\ k = 1, ..., N\}$, где s – это координата вдоль кривой. В качестве начального распределения точек выбирается $\{s_k\}$, которая даёт исходные опорные точки. Далее выполняется преобразование, описанное выше, результатом которого будет отрезок прямой с некоторым распределением точек вдоль него. Таким образом, конформное отображение *всей ломаной* на отрезок прямой играет роль отображения

$$\{s_k\} \to \{s'_k\}\ , \qquad \text{где} \qquad s'_k = \mathscr{S}\left(s_k\right)\ , \quad \forall\ k = 1, ..., N\ , \tag{22}$$

причем на концах $\mathscr{S}(s_1) = s_1$ и $\mathscr{S}(s_N) = s_N$, т.е. отображение тождественно, а о функции $\mathscr{S}(s)$ как функции непрерывного аргумента известно лишь то, что она является монотонной и достаточно гладкой, поскольку исходная кривая представима сплайном. Нужно сделать так, чтобы распределение $\{s'_k\}$ стало равномерным, т.е. подобрать такие значения $\{s_k\}$, чтобы $\{s'_k\}$ была линейной функцией индекса k. Учитывая значения на концах, единственно возможным является

$$s'_k = s_1 + (k-1)\cdot(s_N - s_1)/(N-1) = s'^{\infty}_k\ , \tag{23}$$

а это означает, что $s_k = \mathscr{S}^{-1}\left(s'_k\right)$, где $\mathscr{S}^{-1}$ функция обратная к $\mathscr{S}$, обратимость которой следует из её монотонности. Поскольку $\mathscr{S}^{-1}$, неизвестна, задача решается численно итерационным методом: зная приближение $\left\{s^{(m)}_k\right\}$ для $\{s_k\}$ получить $\left\{s^{(m)\prime}_k\right\}$ прямым преобразованием. Это полученное распределение не совпадает с идеальным $\{s'^{\infty}_k\}$, поэтому для каждого индекса $k = 2, ..., N-1$ нужно найти индекс k_m, так чтобы

$$s^{(m)\prime}_{k_m} \le s'^{\infty}_k < s^{(m)\prime}_{k_m+1}\ , \tag{24}$$

а далее найти интерполяционный коэффициент p, так, чтобы

$$s_k'^{\infty} = (1-p)\cdot s_{k_m}^{(m)\prime} + p\cdot s_{k_m+1}^{(m)\prime}\,, \tag{25}$$

(что в простейшем случае подразумевает линейную интерполяцию). Далее обнов-

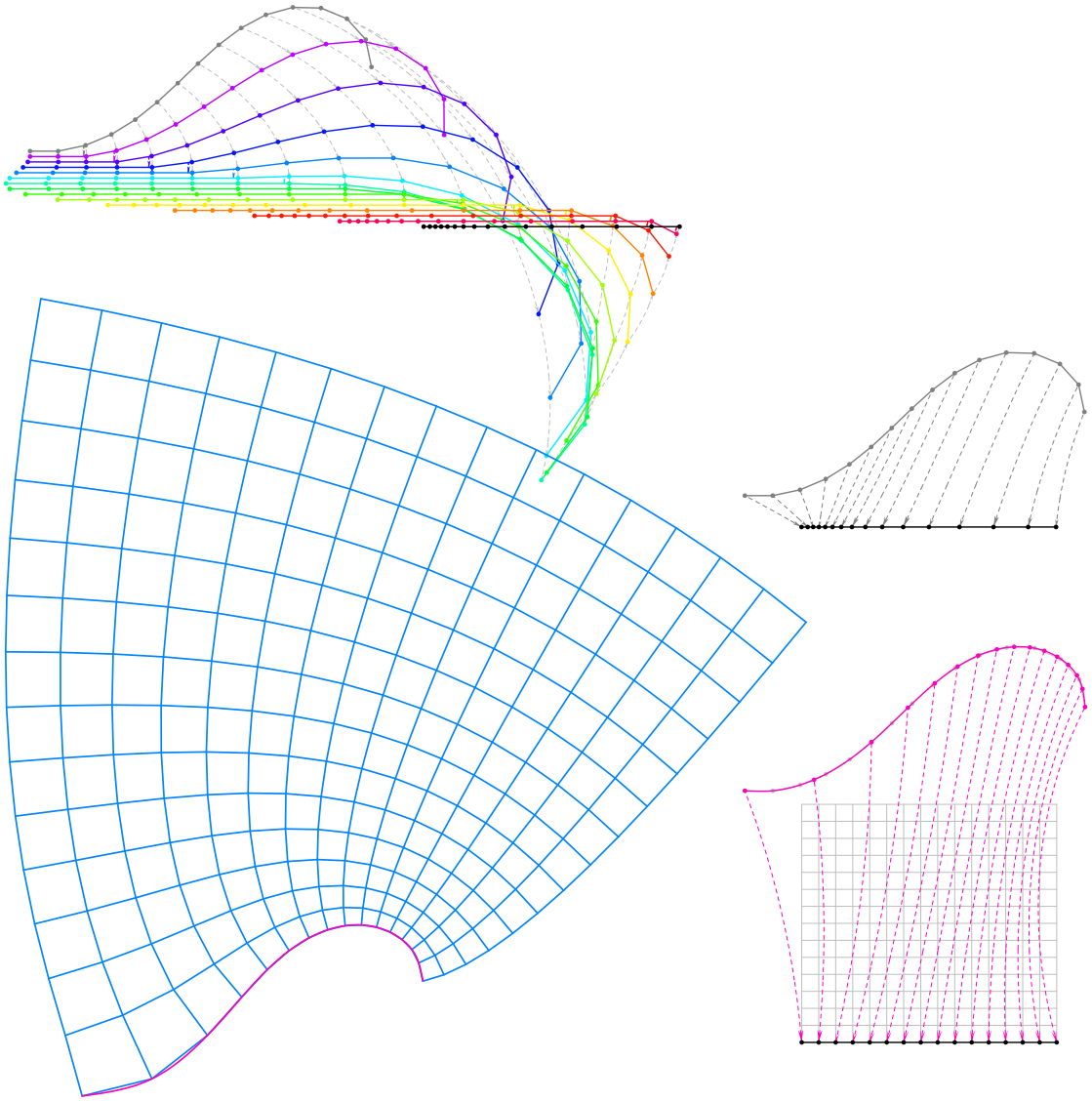


Рис. 8. *Вверху*: Пошаговое выпрямление ломаной преобразованием возведения в степень (рис. 7). Начальное состояние – самая верхняя линия, показанная серым цветом; конечное – черная горизонтальная прямая. Цветные линии соответствуют промежуточным состояниям. Для иллюстрации, чтобы избежать наложения линий друг на друга, ломаная слегка смещается вниз (но не вправо – влево) после каждого шага. *Справа посередине*: То же самое, что и выше, но показано только начальное и конечное состояния, соответствующие точки соединены пунктирными стрелками. Исходная ломаная состоит из сегментов равной длины, однако после выпрямления точки оказались расположены неравномерно. *Справа внизу*: Решение обратной задачи – по исходным точкам строится сплайн и ищется такое распределение точек вдоль кривой, чтобы преобразование выпрямления дало равноудаленные точки. По этим точкам строится прямоугольная сетка с квадратными ячейками. Обратным конформным преобразованием она превращается в криволинейную ортогональную сетку *внизу слева*.

ленные величины $\{s_k\}$ находятся как

$$s_k^{(m+1)} = (1-p)\cdot s_{k_m}^{(m)} + p\cdot s_{k_m+1}^{(m)}\,, \tag{26}$$

после чего весь процесс повторяется снова и снова. Этот итерационный процесс сходится, в результате чего появляется такое распределение точек, что соответствующее ему распределение вдоль прямой будет равномерным (рис. 8, *справа внизу*). После того, как распределение $\{s_k\}$ построено, делается ещё раз конформное преобразование с запоминанием степеней P_k на каждом шаге (т.е. $N-2$ значений), далее в пространстве z' строится прямоугольная декартова сетка (рис. 8, *справа внизу*), и обратным преобразованием она переводится в пространство z (рис. 8, *слева*).

Заметим, что за исключением количества узлов в вертикальном направлении (подразумевается, что исходные ячейки имеют квадратную форму), вся сетка *полностью определяется свойствами исходной кривой*. Сетка, построенная таким образом, фактически является аналитической: всё, что требуется знать, чтобы её построить, это последовательность степеней P_k для обратного преобразования. Здесь практически удалось реализовать сценарий рис. 1 построения ортогональной криволинейной сетки обратным конформным преобразованием. Ожидаемые ошибки ортогональности при таком подходе зависят только от гладкости исходных сплайнов – от которой, в конечном счете, зависит правомерность приближения непрерывной кривой ломаной линией. Такой подход применим для создания сеток моделей в береговой конфигурации – когда есть береговая линия и ставятся три открытые границы, фактическое положение которых может быть задано достаточно произвольно. Однако, несмотря на очень большое количество работ с использованием подобных конфигураций, нам неизвестны случаи применения подобного подхода.

2.4. Построение ортогональной криволинейной сетки изотропного разрешения для замкнутого контура

Алгоритм IZ89 конформно отображает произвольный замкнутый многоугольник в прямоугольник. При этом четыре вершины многоугольника должны быть заранее определены как будущие углы прямоугольника. Подобно алгоритму выпрямления ломаной линии (рис. 8), алгоритм IZ89 заключается в последовательном выпрямлении углов $\angle z_{k-1}\, z_k\, z_{k+1}$, двигаясь вдоль контура и применяя преобразование возведения в степень $z' = z_k + (z - z_k)^{P_k}$ для каждой вершины z_k (т.е. $z_j \to z'_j$ применяется ко всем точкам контура z_j, $j = 1, ..., N$, $j \neq k$ для каждого k). Степень P_k определяется углом изгиба α_k – это внешний угол для $\angle z_{k-1}\, z_k\, z_{k+1}$ для всех неугловых точек z_k – в этом случае $P_k = \pi/(\pi - \alpha_k)$. Для угловых точек $P_k = (\pi/2)/(\pi/2 - \alpha'_k)$, где α'_k – это угол поворота сегмента $[z_k\, z_{k+1}]$ относительно направления перпендикуляра к сегменту $[z_{k-1}\, z_k]$ в направлении против часовой стрелки от него. Поскольку возведение в степень комплексного числа подразумевает умножение его аргумента на показатель степени, то после каждой такой операции угол поворота в точке z_k становится либо π (т.е. ломаная «выпрямляется» в точке z_k, рис. 7), либо приводится к углу в точности равному $\pi/2$ в угловых точках.

Очевидно, что если точки z_{k-1}, z_k, z_{k+1} находятся на одной прямой, либо уже образуют прямой угол, если точка z_k угловая, то в обоих случаях $\alpha_k = 0$ и $P_k = 1$, что соответствует тождественному преобразованию. Последовательное применение преобразования возведения в нужную степень ко всем точкам одной стороны криволинейного контура, от угла до угла (включая их), выпрямляет эту сторону, однако, после прохождения угловой точки, выпрямление следующей стороны приводит к искривлению предыдущей, хотя в меньшей степени, чем её исходное состояние (рис. 9). Таким образом, для того, чтобы выпрямить все четыре стороны, необходимы итерации, повторяющие весь процесс снова и снова. Их конечным результатом является прямоугольный контур с определённым соотношением длин сторон и определённым распределением узловых точек на каждой стороне – в соответствии с выпрямляющими степенями $P_k > 1$ или $P_k < 1$, в зависимости от углов поворота в каждой точке z_k, происходит растяжение или сжатие точек вдоль контура, которое, по сути, и является целью алгоритма IZ89, описание которого на этом заканчивается.

В конечном счете, нам нужно добиться, чтобы получающееся в результате преобразования алгоритмом IZ89 распределение точек оказалось равномерным на каждой стороне прямоугольника. Это означает, что исходные точки $\{z_k\}$ нужно каким-то образом перераспределить вдоль криволинейного контура, заданного сплайном, но так, чтобы они при этом оставались на контуре. Таким образом, мы имеем дело с обратной задачей. Её решение принципиально возможно, поскольку каждый элементарный шаг алгоритма IZ89 обратим, и, следовательно, существует обратное конформное преобразование прямоугольника в контур, состоящее из последовательности обратных операций (то есть возведения в степень $1/P_k$ всех разностей $z_j - z_k$, где $j \neq k$), в обратном порядке, и, если начать с прямоугольника с равномерным заполнением сторон узловыми точками, можно получить искомое распределение на криволинейном контуре. Однако, реализация такого обратного преобразования требует запоминания всех промежуточных значений степеней $P_k^{(m)}$ для каждой точки $z_k^{(m)}$ во время всех итерационных повторений m, применённых во время прямого преобразования, с тем, чтобы можно было выполнить их в обратном порядке. Практически это не реализуемо из-за очень большого числа шагов, $N^2 \cdot M$, где N это число узловых точек на контуре, а M – число итерационных повторений.

Поэтому выбран другой подход: построить ещё одну внешнюю итерационную петлю вокруг алгоритма IZ89 (который уже сам по себе является итерационным). Координаты узлов будущей сетки (x_k, y_k), лежащие на контуре, соответствуют последовательности координат этих точек вдоль периметра контура, s_k. Фактически зная s_k, координаты краевых точек будущей сетки (x_k, y_k) определяются интерполяцией при помощи сплайна, построенного ранее. Тогда, изменяя s_k, можно двигать точки, но так, чтобы они всё время оставались на контуре. Результатом преобразования IZ89 является распределение точек на прямоугольнике, которое отличается от равномерного (т.е. того, что мы хотим добиться). Зная это отличие (т.е. невязку), можно построить новую, скорректированную последовательность s_k, а по ней, в свою очередь, вычислить новые координаты (x_k, y_k) узловых точек на контуре. После этого весь процесс снова повторяется до тех пор, пока распределение точек

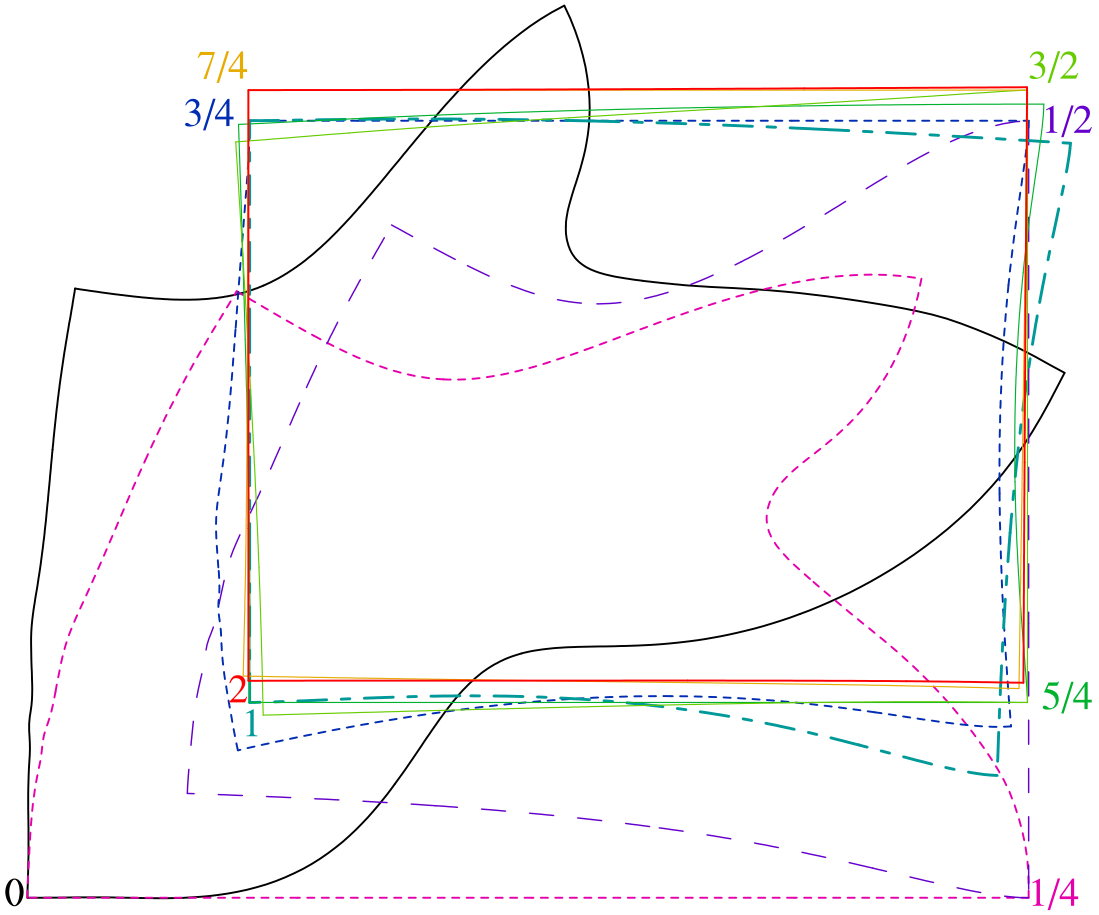


Рис. 9. Эволюция формы контура в процессе конформного преобразования в прямоугольник алгоритмом IZ89. Цвета контуров и дробных чисел, обозначающих последовательность шагов алгоритма, соответствуют друг другу. Начальный контур сетки Чёрного моря, сплошная чёрная линия, повёрнут, чтобы сориентировать его так, что касательная к западной стороне в юго-западной угловой точке стала вертикальной на этом рисунке. После этого, начиная от юго-западного угла и двигаясь на восток вдоль южной стороны, производится последовательное выпрямление всех углов южной стороны. Как только этот процесс достигнет юго-восточного угла, южная сторона становится прямой (пунктирный контур красно-фиолетового цвета, того же, что и 1/4, – это означает, что пройдена только одна сторона из четырёх). Однако, т.к. каждый раз элементарное преобразование IZ89 – возведение разностей комплексных чисел в степень – применяется сразу ко всему пространству (то есть ко всем точкам контура), то три остальные стороны при этом тоже деформируются. Далее, точно таким же образом спрямляется восточная сторона, контур цвета 1/2 (сине-фиолетовый), но при этом деформируются остальные, в том числе и только что спрямлённая южная. Далее спрямляется северная сторона, западная, и т.д. против часовой стрелки. Весь цикл повторяется снова и снова. Всего на этом рисунке показано два полных обхода контура. Последний (красный контур 2) является почти прямоугольным, а последовательность прямых сторон асимптотически приближается к нему по прямоугольной спирали. Для иллюстративных целей все контуры на этом рисунке нормированы на равную площадь, а для каждого элементарного преобразования спрямляемая точка остаётся неподвижной, что создаёт визуальный эффект «эвольвентного качения» (в фактическом алгоритме IZ89 в этой нормировке и неподвижности нет необходимости).

на прямоугольнике не сходится к желаемому равномерному. При этом, наряду с координатами узловых точек, можно так же изменить и их число с тем, чтобы соотношение интервалов между точками на смежных сторонах прямоугольника было как можно ближе к единице.

Для того, чтобы начать этот итерационный процесс, необходимо иметь началь-

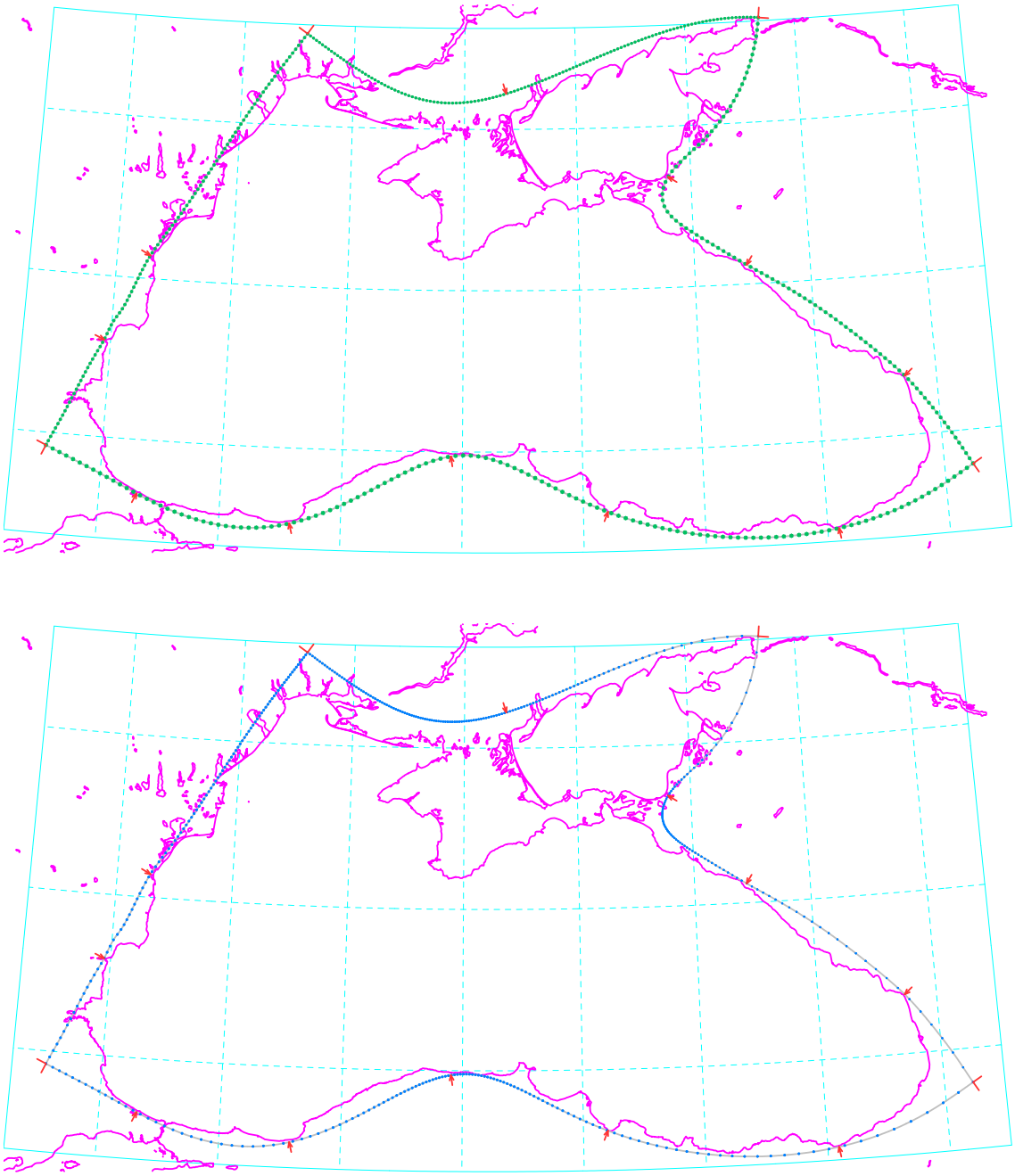


Рис. 10. *Сверху*: заполнение контура точками-узлами, равноудаленными на каждой из четырех сторон (это соответствует заданию `mode=2`). *Внизу*: неравномерное распределение узловых точек, подобранное таким образом, что конформное преобразование контура в прямоугольник превратит это распределение в равномерное на его сторонах (`mode=3`).

ное приближение, в качестве которого мы выбираем равномерное распределение точек на каждой стороне криволинейного контура (рис. 10, *верхняя панель*. Это соответствует `mode=2`. Само по себе такое распределение точек требует ещё одной итерационной процедуры, поскольку длины сторон криволинейного контура заранее неизвестны, в качестве оценки таковой используется сумма расстояний между узловыми точками, а для того, чтобы её вычислить, требуется наличие этих точек. Очевидным грубым приближением является сумма расстояний между опорными точками сплайна. По нему, зная число узлов будущей сетки в обоих направлениях, делается оценка шага равномерного заполнения для каждой из сторон контура. Каждая попытка заполнения приводит к более точной оценке длины кривой, а также к невязке – начав заполнение от одного угла, последняя точка должна точно попасть в следующий угол; недолет/перелет даёт коррекцию шага. При этом нужно отметить, что положение точек необходимо получить в терминах последовательности s_k с тем, чтобы их координаты можно было вычислить сплайн-интерполяцией. Так как используемые сплайны полиномами 3-го и 5-го порядков не допускают явного обращения, но применяется внутренний итерационный алгоритм на каждом шаге заполнения. Порядок сходимости алгоритма в целом (два вложенных итерационных цикла) такой же, как у метода Ньютона, поэтому сравнительно небольшое число повторений ($\sim 6...8$) может достичь ошибки округления при вычислении с двойной точностью.

После того как начальное приближение s_k и соответствующие ему (x_k, y_k) заданы, можно приступить к решению обратной задачи при помощи итерационного цикла вокруг алгоритма IZ89, описанного выше. Программно это соответствует `mode=3` и является самым сложным и вычислительно затратным этапом. На каждом этапе, зная s_k, вычисляются $z_k = x_k + iy_k$, которые затем подвергаются конформному преобразованию в прямоугольник, в результате чего становится известным распределение точек на сторонах прямоугольника. Это распределение мы хотим сделать равномерным, но оно не является таковым. По этому отличию вычисляются невязки и корректируются значения s_k. Итерационный процесс повторяется для достижения сходимости.

В части коррекции значений s_k на каждой внешней итерации, алгоритм аналогичен перераспределению узлов вдоль кривой (см. уравнения (23)–(26) а так же рис. 7 и описание к нему), если его применить к каждой стороне контура отдельно. Но есть два новых момента:

(i) помимо изменяющихся значений s_k, корректируется так же и число точек сетки в одном из направлений. Дело в том, что, как только соотношение сторон прямоугольника становится известным, для того, чтобы сделать ячейки декартовой сетки как можно ближе к квадратным, необходимо выбрать число ячеек так, чтобы `nx/ny` соответствовало соотношению сторон. Поэтому программа сама выбирает число ячеек вдоль первого направления, `nx`, чтобы этого добиться. При этом значение `nx`, заданное пользователем в во входном файле, является лишь его начальным приближением. Почему именно `nx` а не `ny`? Это связано с тем, что количество точек сетки во втором направлении, `ny`, желательно выбрать «хорошим» с точки

зрения распараллеливания ROMS – возможность деления нацело, на большое число процессоров. Поэтому его значение лучше не изменять относительно заданного пользователем. Для делимости `nx` требования не такие жёсткие, потому что деление на большие числа приводит к коротким внутренним циклам в коде ROMS и снижению скорости счёта. А для длин циклов порядка ~ 100 и более отличие на единицу не так существенно влияет на одинаковость загруженности процессоров.

(*ii*) следует обратить внимание, что алгоритм IZ89 является итерационным, таким образом, мы имеем двойной вложенный итерационный процесс. Его можно оптимизировать, уменьшив количество итераций внутренней петли для начальных итераций внешней, и постепенно увеличивать их – вначале достаточно лишь грубой прикидки. Эмпирически мы установили зависимость $M = \sqrt{2^{m+5}}$, где M – это число полных обходов контура в алгоритме IZ89, а $m = 1, 2, 3, ...$ – это номер итерации внешней петли. Число повторений внешней петли порядка $4...10$ устанавливается входным параметром `npass`. Практически это позволяет достичь ошибки, соизмеримой с округлением двойной точности, и при дальнейших итерациях ошибка не уменьшается.

Что же касается самого алгоритма IZ89, то он был полностью переработан нами. Целями являются предотвращение накопления ошибок округления и добиться улучшения скорости вычислений за счёт оптимизации арифметических вычислений, а также за счёт распараллеливание кода.

Во-первых, полный отказ от использования комплексных чисел.

Во-вторых, элементарный алгоритм пошагового выпрямления переформулирован в терминах угла поворота (внешнего угла, дополняющего $\angle z_{k-1}, z_k, z_{k+1}$ до π для всех точек z_k кроме углов будущей сетки, и до $\pi/2$ для угловых точек. Таким образом, единственной используемой обратной тригонометрической функцией является `ATAN2`, причём только для малых углов (стремящимся к бесконечно малым по мере сходимости к прямоугольнику). Следовательно, преобразование стремится к тождественному при повторных итерациях, и мы добились того, что решение при этом не «плывёт» за счёт ошибок округления.

В-третьих, алгоритм IZ89 использует слежение за тем, чтобы аргумент комплексных чисел изменялся непрерывно между точками контура, добавляя/вычитая 2π когда это необходимо. Эта часть заменена полностью: оригинальная версия несовместима с распараллеливанием и, кроме того, имеет тенденцию к накоплению ошибок округления. Распараллеливание по типу *coarse-grain* на общей памяти, делением контура на сегменты по числу CPU, с последующим обнаружением разрывов аргумента на стыках между процессорами добавлением нужного числа 2π при необходимости.

2.5. Заполнение сеткой области, ограниченной криволинейным контуром

Как только положение узлов сетки на контуре известно, становится возможным произвести заполнение криволинейной сетки. Для этого необходимо решить задачу Дирихле (4) для каждой из координат x и y в отдельности. Это последний этап,

соответствующий `mode=4`.

Поскольку в пространстве (ξ, η) сетка имеет близкие друг к другу шаги в каждом направлении, $\Delta\xi \approx \Delta\eta$ – неточность вызвана необходимостью округления отношения сторон прямоугольника к отношению чисел точек в соответствующих направлениях L/M – здесь выгодно применить дискретизацию «мерштеленферфарен»[2] (Collatz, 1960; Schaffer, 1984; Deriaz, 2020). Идея такого конечно-разностного оператора Лапласа заключается в том, что если нужно решить уравнение Пуассона,

$$\nabla^2 q = q_{\xi\xi} + q_{\eta\eta} = f\,, \tag{27}$$

то, если удаётся приблизить оператор ∇^2 конечно-разностным оператором второго порядка точности $\mathscr{L}$, устроенного таким образом, что его ошибку аппроксимации во втором порядке можно представить как лапласиан лапласиана,

$$\mathscr{L} q = \nabla^2 q + C \cdot \nabla^2 \left[\nabla^2 q\right] + \mathscr{O}\left(\Delta\xi^4, \Delta\xi^2\Delta\eta^2, \Delta\eta^4\right)\,, \tag{28}$$

где $C \sim \mathscr{O}\left(\Delta\xi^2, \Delta\xi\Delta\eta, \Delta\eta^2\right)$ – это коэффициент, квадратично уменьшающийся к нулю при $\Delta\xi \to 0, \quad \Delta\eta \to 0$, то, в силу самого уравнения (27), лапласиан в квадратных скобках $[\nabla^2 q]$ можно заменить на f и перенести в правую часть, т.к. он, по существу, известен. В этом случае вместо конечно-разностного уравнения $\mathscr{L} q = f$ можно решать уравнение с модифицированной правой частью,

$$\mathscr{L} q = f + C \cdot \nabla^2 f = \mathscr{R} f\,, \tag{29}$$

и его решение будет решением исходного уравнения (27), но уже не со вторым, а с *четвёртым* порядком точности. Заметим, что оператор $\mathscr{R} = 1 + C \cdot \nabla^2$ в правой части по сути является оператором сглаживания. Используя анализ Фурье, можно показать что применение сглаживания к правой части уравновешивает сглаживающий эффект замены «настоящего» лапласиана в левой на его конечно-разностный аналог, и именно за счет этого и происходит устранение ошибки во втором порядке точности. Этот принцип лежит в основе любой схемы *компактного дифференцирования*. При этом стоит подчеркнуть, что взятые по отдельности, операторы $\mathscr{L} q$ в левой и $\mathscr{R} f$ в правой частях четвёртым порядком точности не обладают.

Для вычисления $\nabla^2 f$ в правой части можно использовать любой аппроксимирующий ∇^2 оператор второго порядка точности, в частности и сам $\mathscr{L}$,

$$\mathscr{L} q = f + C \cdot \mathscr{L} f. \tag{30}$$

Наконец, если нас интересует уравнение Лапласа с нулевой правой частью, то конечно-разностное уравнение, построенное на этом операторе,

$$\mathscr{L} q = 0\,, \tag{31}$$

так же обладает четвёртым порядком точности.

[2] От немецкого *mehrstellenverfahren* – метод множественных позиций. По сложившейся традиции, в англоязычной литературе это слово никогда не переводится, а используется как есть.

Классическая схема «мерштеленферфарен» для уравнения Пуассона получена для строго одинаковых $\Delta\xi = \Delta\eta$, и теряет свои свойства (в т.ч. четвёртый порядок точности), если это равенство нарушается. Нас же интересует решение именно уравнения Лапласа, что несколько упрощает задачу, но на сетках, для которых $\Delta\xi \approx \Delta\eta$, хотя и близки друг к другу, но всё же не равны. Поэтому имеет смысл пересмотреть вывод этой схемы с целью изучения возможности адаптировать её под такую задачу.

Рассмотрим девятиточечный конечно-разностный оператор,

$$\mathscr{L}_* = \begin{bmatrix} \gamma & \beta & \gamma \\ \alpha & \delta & \alpha \\ \gamma & \beta & \gamma \end{bmatrix}, \qquad \delta = -2\alpha - 2\beta - 4\gamma\,, \tag{32}$$

где коэффициенты α, β, γ пока ещё не определены. Используя разложения в ряд Тейлора,

$$\begin{gathered} q_{i+1,j} = q_{i,j} + ... + \frac{\Delta\xi^2}{2}\cdot\frac{\partial^2 q}{\partial\xi^2} + ... + \frac{\Delta\xi^4}{24}\cdot\frac{\partial^4 q}{\partial\xi^4} + ... \\ q_{i,j+1} = q_{i,j} + ... + \frac{\Delta\eta^2}{2}\cdot\frac{\partial^2 q}{\partial\eta^2} + ... + \frac{\Delta\eta^4}{24}\cdot\frac{\partial^4 q}{\partial\eta^4} + ... \\ q_{i+1,j+1} = q_{i,j} + \sum_{n=1}^{\infty}\frac{1}{n!}\left(\Delta\xi\frac{\partial}{\partial\xi} + \Delta\eta\frac{\partial}{\partial\eta}\right)^n q = ... + \frac{\Delta\xi^2}{2}\cdot\frac{\partial^2 q}{\partial\xi^2} + \frac{\Delta\eta^2}{2}\cdot\frac{\partial^2 q}{\partial\eta^2} \\ +... + \frac{1}{24}\left[\Delta\xi^4\frac{\partial^4 q}{\partial\xi^4} + 6\Delta\xi^2\Delta\eta^2\frac{\partial^4 q}{\partial\xi^2\partial\eta^2} + \Delta\eta^4\frac{\partial^4 q}{\partial\eta^4}\right] + ... \end{gathered}$$

где нас интересуют только четные степени $\Delta\xi$ и $\Delta\eta$ (члены с нечетными сократятся в силу симметрии (32)), получаем

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

Чтобы обеспечить соответствие членов второго порядка, необходимо выполнить два условия,

$$\alpha = 1/\Delta\xi^2 - 2\gamma \quad \text{и} \quad \beta = 1/\Delta\eta^2 - 2\gamma\,, \tag{33}$$

где коэффициент γ пока остаётся произвольным. Исключив α и β при помощи этих соотношений и перегруппировав члены четвёртого порядка, можно получить что

$$\gamma = 1/(6\Delta\xi\Delta\eta) \tag{34}$$

приводит к

$$\begin{gathered} \mathscr{L}_* q = \frac{\partial^2 q}{\partial\xi^2} + \frac{\partial^2 q}{\partial\eta^2} + \frac{1}{12}\cdot\left[\Delta\xi\cdot\frac{\partial^2}{\partial\xi^2} + \Delta\eta\cdot\frac{\partial^2}{\partial\eta^2}\right]^2 q \\ +\mathscr{O}\left(\Delta\xi^4\right) + \mathscr{O}\left(\Delta\xi^2\Delta\eta^2\right) + \mathscr{O}\left(\Delta\eta^4\right) \end{gathered} \tag{35}$$

и это единственно возможное значение γ, при котором члены четвёртого порядка факторизуются в произведение операторов.

Если $\Delta\xi = \Delta\eta = \Delta h$, то получаем классический оператор,

$$\mathscr{L}_{*} = \frac{1}{6} \cdot \frac{1}{\Delta h^2} \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}, \tag{36}$$

который обладает искомым свойством,

$$\mathscr{L}_{*}\, q = \nabla^2 q + \frac{1}{12} \cdot \Delta h^2 \cdot \nabla^4 q + \mathscr{O}\left(\Delta h^4\right). \tag{37}$$

Его так же можно представить как взвешенную сумму обычного, прямого 5-точечного лапласиана, взятого с весом 2/3 и диагонального, взятого с весом 1/3,

$$\mathscr{L}_{*} = \frac{2}{3}\mathscr{L}_{+} + \frac{1}{3}\mathscr{L}_{\times} = \frac{2}{3} \cdot \frac{1}{\Delta h^2} \begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} + \frac{1}{3} \cdot \frac{1}{\Delta h^2} \begin{bmatrix} 1/2 & & 1/2 \\ & -2 & \\ 1/2 & & 1/2 \end{bmatrix} \tag{38}$$

Если же $\Delta\xi \neq \Delta\eta$, то оператор (35) свести к лапласиану и повторному лапласиану нельзя, однако если представить

$$\Delta\xi = \Delta h(1+\epsilon) \quad \text{и} \quad \Delta\eta = \Delta h(1-\epsilon), \quad \text{где} \quad \epsilon \ll 1, \tag{39}$$

то (35) приводится к виду

$$\begin{aligned} \mathscr{L}_{*}\, q &= \nabla^2 q + \frac{\Delta h^2}{12} \cdot \left[\frac{\partial^2}{\partial\xi^2} + \frac{\partial^2}{\partial\eta^2} + \epsilon \cdot \left(\frac{\partial^2}{\partial\xi^2} - \frac{\partial^2}{\partial\eta^2}\right)\right]^2 q + \mathscr{O}\left(\Delta h^4\right) \\ &= \nabla^2 q + \frac{\Delta h^2}{12} \cdot \left[\nabla^2 + \epsilon \cdot \mathscr{H}\right]^2 q + \mathscr{O}\left(\Delta h^4\right) \\ &= \nabla^2 q + \frac{\Delta h^2}{12} \cdot \nabla^4 q + \frac{\Delta h^2}{6} \cdot \epsilon \cdot \mathscr{H}\nabla^2 q + \frac{\Delta h^2}{12} \cdot \epsilon^2 \cdot \mathscr{H}^2 q + \mathscr{O}\left(\Delta h^4\right) \end{aligned} \tag{40}$$

где $\mathscr{H}$ это «оператор перекоса». Первые два члена в правой части совпадают с (37). Поскольку подразумевается, что q является решением конечно-разностного уравнения $\mathscr{L}_{*}\, q = 0$, можно утверждать что, по крайней мере *не хуже чем со вторым порядком* точности, $\nabla^2 q = 0 + \mathscr{O}\left(\Delta h^2\right)$ Тогда для третьего и четвёртого членов справедливы оценки

$$\begin{aligned} \frac{\Delta h^2}{6} \cdot \epsilon \cdot \mathscr{H}\nabla^2 q \to \frac{\Delta h^2}{6} \cdot \epsilon \cdot \mathscr{H}\mathscr{O}\left(\Delta h^2\right) \sim \mathscr{O}\left(\epsilon\Delta h^4\right) \\ \frac{\Delta h^2}{12} \cdot \epsilon^2 \cdot \mathscr{H}^2 q \sim \mathscr{O}\left(\epsilon^2\Delta h^2\right) \end{aligned} \tag{41}$$

Для оператора (37) решение уравнения $\mathscr{L}_{*}\, q = 0$ даёт $\nabla^2 q = 0 + \mathscr{O}\left(\Delta h^4\right)$ с четвёртым порядком точности. Аналогичная оценка для (40) даёт $\nabla^2 q = 0 + \mathscr{O}\left(\epsilon^2\Delta h^2\right) + \mathscr{O}\left(\Delta h^6\right)$, т.е. порядок уменьшился до второго, однако если обеспечить стремление $\epsilon \to 0$ совместно с $\Delta h \to 0$, то четвёртый порядок можно сохранить. В нашем случае

это достигается тем, что число точек сетки L в одном из направлений, направлении ξ, выбирается автоматически из соотношения сторон прямоугольника, получившегося в результате конформного преобразования алгоритмом IZ89, с тем, чтобы сделать отношение $\Delta\xi/\Delta\eta$ как можно ближе к единице. При этим, с увеличением числа точек сетки $\Delta\xi \sim 1/L$, а так же $\epsilon \sim 1/L$.

Важно отметить, что ни (37), ни тем более (40), сами по себе не являются конечно-разностными схемами, аппроксимирующими оператор Лапласа с четвёртым порядком точности. Речь идёт именно об аппроксимации уравнения Пуассона *в целом*, т.е. когда правая часть подвергается сглаживанию с тем, чтобы вносимые таким образом предыскажения уравновесили ошибку аппроксимации разностного оператора в левой части, а уравнение Лапласа, аппроксимированное таким способом является лишь частным случаем уравнения Пуассона с тривиальной правой частью. Так же заметим, что в первой строке уравнения (41) переход члена $\frac{\Delta h^2}{6}\cdot\epsilon\cdot\mathscr{H}\nabla^2 q$ в более высокий порядок малости (т.е. фактически его устранение) возможнен благодаря тому, что решается именно уравнение Лапласа.

Оператор $\mathscr{L}_*$, определяемый уравнением (35), используется для построения задач Дирихле в прямоугольной области для каждой из координат $x = x(\xi,\eta)$ и $y = y(\xi,\eta)$ по отдельности. Несмотря на геометрическую простоту, дискретная задача не является сепарабельной (в отличие от 5-точечного лапласиана), и распространённые прямые методы, такие, как FACR[3], оказываются неприменимыми. Тем не менее, задача решается методом сверх-релаксации с Чебышевским ускорением (Press et al., 1992, в особенности Sec. 19.5), но при этом используется не обычный, красно-черный «шахматный», а четырёхцветный релаксатор Гаусса-Сейделя, и, соответственно, последовательность оптимальных значений параметра сверхрелаксации ω получается несколько иной: зададим радиус Якоби r_{Jacobi} и начальное значение ω,

$$r_{\text{Jacobi}} = [\cos(\pi/L) + \cos(\pi/M)]/2\,, \qquad \omega_1 = 1\,, \tag{42}$$

где L и M это числа точек сетки в направлениях ξ и η, далее рекуррентно,

$$\omega_m = 1/\left[1 - (1/4)\cdot\omega_{m-1}\cdot r^2_{\text{Jacobi}}\right] \qquad \forall m = 2, \ldots \tag{43}$$

где m – это номер итерации. В отличие от Press et al. (1992), эта последовательность монотонно возрастает, а подстановкой $\omega_m \to \omega_\infty$ и $\omega_{m-1} \to \omega_\infty$ в (43) можно показать, что ω_m асимптотически стремится к $\omega_\infty = 2\Big/\left[1+\sqrt{1-r^2_{\text{Jacobi}}}\right]$, что теоретически является оптимальным значением параметра сверхрелаксации.

В принципе, геометрическая простота данной задачи даёт возможность применения и более совершенного алгоритма, например, многосеточного, однако вычислительные затраты даже для очень больших сеток, порядка 1500×1000 точек, оказываются вполне приемлемыми – десятки минут на современных CPU, используя все ядра, и, учитывая то, что это приходится делать только один раз, а все свойства создаваемой сетки можно проинспектировать и «обкатать» на меньшем разрешении, это делает излишнюю сложность менее привлекательной.

[3] Fourier Analysis Cyclic Reduction (Swarztrauber, 1977)

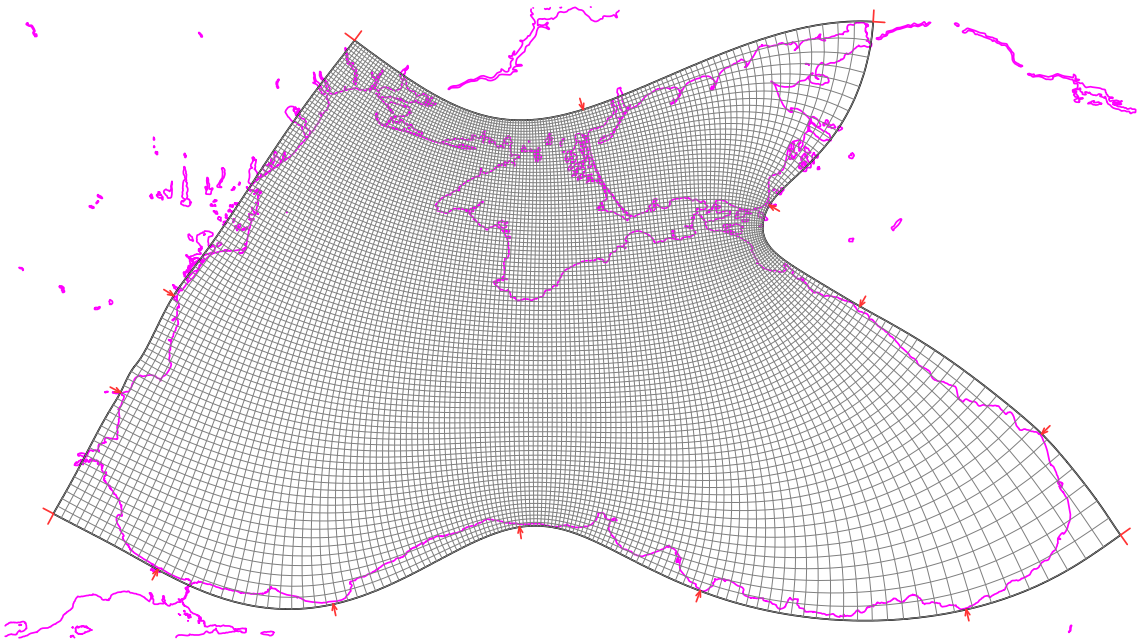


Рис. 11. Готовая ортогональная криволинейная сетка, разрешение 131×101 точек. Для практических целей нас интересуют сетки геометрически подобные этой, но с примерно в 8 раз более тонким разрешением в обоих направлениях ($\sim 1050 \times 800$). Очевидно, их нельзя отобразить графически из-за слияния линий.

Результат решения эллиптической задачи для построения сетки Чёрного моря показан на рис. 11. После того как координаты всех узлов сетки $(x_{i,j}, y_{i,j})$ в плоской системе координат картографической проекции известны, остаётся только вернуться в сферические координаты, вычислить метрические коэффициенты (т.е. обратные расстояния между узлами сферической сетки), вычислить в каждой точке азимутальный угол между направлением на восток и локальным направлением вдоль первой криволинейной координатной линии. Эти преобразования хорошо известны (например, Snyder, 1987). Кроме того, на этом этапе учитывается тот факт, что ROMS (NEMO и т.д.) использует смещённые сетки для разных переменных (т.н. Arakawa C-grid, Arakawa & Lamb (1977). Далее следует построить земляную маску при помощи одного из алгоритмов, используемых в сообществе ROMS, например,

https://www.myroms.org/forum/viewtopic.php?f=23&t=3878&p=14839.

Завершающей операцией является построение топографии дна, после чего файл сетки готов к использованию.

2.6. Проверка ошибок ортогональности и изотропии

Несмотря на то, что понятие ортогональности достаточно интуитивно, поскольку мы имеем дело с дискретной системой, само определение ортогональности неоднозначно и допускает некоторую вариативность в рамках порядка точности дискретизации. Это проиллюстрировано на рис. 12.

В данном случае сетки заданы аналитически и являются ортогональными в том смысле, что непрерывные функции, определяющие конформное отображение, обла-

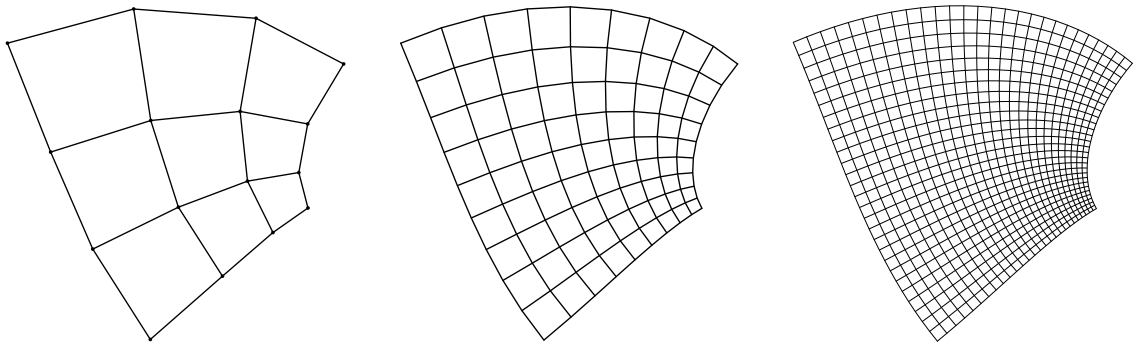


Рис. 12. Пример сеток, построенных аналитическим конформным отображением для трёх разрешений, 3×3, 9×9 и 27×27 ячеек.

дают всеми необходимыми свойствами. Очевидно, что имеет место сходимость, и нет сомнений в том, что при увеличении числа точек ошибка ортогональности (как бы она ни была определена) стремится к нулю. Однако, если посмотреть на левую панель рис. 12, то никакой угол там не является визуально прямым, и критерий, при помощи которого можно судить об ошибках ортогональности этой сетки, совершенно не очевиден. В данной части мы рассмотрим такие критерии. Это особенно необходимо, если приходится иметь дело с сетками, построенными численными методами, а не аналитически. Наличие файла с координатами всех узлов сетки $(x_{i,j}, y_{i,j})$ в декартовой системе координат даёт возможность вычислить ошибку ортогональности.

Конечно-разностное определение ошибок ортогональности является наиболее простым. Рассмотрим элементарную ячейку сетки, ограниченную точками $(x_{i,j}, y_{i,j})$, $(x_{i+1,j}, y_{i+1,j})$, $(x_{i+1,j+1}, y_{i+1,j+1})$ и $(x_{i,j+1}, y_{i,j+1})$. Нам нужно вычислить вектора, касательные к координатным линиям криволинейной сетки в обоих направлениях,

$$\boldsymbol{\ell}_\xi = (\partial_\xi x, \partial_\xi y) \quad \text{и} \quad \boldsymbol{\ell}_\eta = (\partial_\eta x, \partial_\eta y)\,, \tag{44}$$

с тем, чтобы можно было проверить условие их взаимной ортогональности,

$$(\boldsymbol{\ell}_\xi \cdot \boldsymbol{\ell}_\eta) = \partial_\xi x \cdot \partial_\eta x + \partial_\xi y \cdot \partial_\eta y = 0, \tag{45}$$

и получить меру её ошибки. Конечно-разностные приближения для производных относительно середины ячейки, т.е. для $i + {}^1\!/_2, j + {}^1\!/_2$,

$$\begin{array}{rcl} \partial_\xi x &=& (x_{i+1,j+1} - x_{i,j+1} \;+\; x_{i+1,j} - x_{i,j})\,/(2\Delta\xi) \\ \partial_\xi y &=& (y_{i+1,j+1} - y_{i,j+1} \;+\; y_{i+1,j} - y_{i,j})\,/(2\Delta\xi) \\ \partial_\eta x &=& (x_{i+1,j+1} - x_{i+1,j} \;+\; x_{i,j+1} - x_{i,j})\,/(2\Delta\eta) \\ \partial_\eta y &=& (y_{i+1,j+1} - y_{i+1,j} \;+\; y_{i,j+1} - y_{i,j})\,/(2\Delta\eta)\,, \end{array} \tag{46}$$

представляют собой комбинацию дифференцирования и осреднения с равными весами в направлении, перпендикулярном к направлению дифференцирования. Подставив эти выражения в условие (45), легко увидеть, что

$$\begin{array}{rcl} (\boldsymbol{\ell}_\xi \cdot \boldsymbol{\ell}_\eta) = \big[(x_{i+1,j+1} - x_{i,j})^2 \;+\; (y_{i+1,j+1} - y_{i,j})^2\big] \\ - \big[(x_{i,j+1} - x_{i+1,j})^2 \;+\; (y_{i,j+1} - y_{i+1,j})^2\big] = 0\,, \end{array} \tag{47}$$

т.е. ортогональность обеспечивается равенством диагоналей ячейки. Заметим, что это справедливо при любом соотношении $\Delta\xi/\Delta\eta$. Меру же ошибки ортогональности естественно определить как,

$$\epsilon = \sin\left(\angle\boldsymbol{\ell}_\xi\boldsymbol{\ell}_\eta - \frac{\pi}{2}\right) = \frac{(\boldsymbol{\ell}_\xi \cdot \boldsymbol{\ell}_\eta)}{|\boldsymbol{\ell}_\xi| \cdot |\boldsymbol{\ell}_\eta|} = \frac{\partial_\xi x \cdot \partial_\eta x + \partial_\xi y \cdot \partial_\eta y}{\sqrt{(\partial_\xi x^2 + \partial_\xi y^2) \cdot (\partial_\eta x^2 + \partial_\eta y^2)}}\,, \tag{48}$$

что есть ни что иное, как синус отклонения угла между векторами $\boldsymbol{\ell}_\xi$ и $\boldsymbol{\ell}_\eta$ от $\pi/2$.

Несколько более точные выражения можно получить, заменив осреднение с равными весами в (46) на *интерполяцию с весами обратно пропорциональными расстоянию между точками сетки*. Такая интерполяция более естественна для криволинейных сеток, так так сгущение сетки наблюдается сразу в обоих направлениях. Для этого определим

$$\begin{aligned} \delta_\xi x_{i+1/2,j} &= x_{i+1,j} - x_{i,j} \qquad & |\boldsymbol{\ell}_\xi|_{i+1/2,j} &= \sqrt{\delta_\xi x^2_{i+1/2,j} + \delta_\xi y^2_{i+1/2,j}} \\ \delta_\xi y_{i+1/2,j} &= y_{i+1,j} - y_{i,j} & & \\ \delta_\eta x_{i,j+1/2} &= x_{i,j+1} - x_{i,j} & |\boldsymbol{\ell}_\eta|_{i,j+1/2} &= \sqrt{\delta_\eta x^2_{i,j+1/2} + \delta_\eta y^2_{i,j+1/2}}\,, \\ \delta_\eta y_{i,j+1/2} &= y_{i,j+1} - y_{i,j} & & \end{aligned} \tag{49}$$

и заменим (46) на

$$\begin{aligned} \partial_\xi x &= \frac{|\boldsymbol{\ell}_\xi|_{i+1/2,j} \cdot \delta_\xi x_{i+1/2,j+1} + |\boldsymbol{\ell}_\xi|_{i+1/2,j+1} \cdot \delta_\xi x_{i+1/2,j}}{|\boldsymbol{\ell}_\xi|_{i+1/2,j} + |\boldsymbol{\ell}_\xi|_{i+1/2,j+1}} \\ \partial_\xi y &= \frac{|\boldsymbol{\ell}_\xi|_{i+1/2,j} \cdot \delta_\xi y_{i+1/2,j+1} + |\boldsymbol{\ell}_\xi|_{i+1/2,j+1} \cdot \delta_\xi y_{i+1/2,j}}{|\boldsymbol{\ell}_\xi|_{i+1/2,j} + |\boldsymbol{\ell}_\xi|_{i+1/2,j+1}} \\ \partial_\eta x &= \frac{|\boldsymbol{\ell}_\eta|_{i,j+1/2} \cdot \delta_\eta x_{i+1,j+1/2} + |\boldsymbol{\ell}_\eta|_{i+1,j+1/2} \cdot \delta_\eta x_{i,j+1/2}}{|\boldsymbol{\ell}_\eta|_{i,j+1/2} + |\boldsymbol{\ell}_\eta|_{i+1,j+1/2}} \\ \partial_\eta y &= \frac{|\boldsymbol{\ell}_\eta|_{i,j+1/2} \cdot \delta_\eta y_{i+1,j+1/2} + |\boldsymbol{\ell}_\eta|_{i+1,j+1/2} \cdot \delta_\eta y_{i,j+1/2}}{|\boldsymbol{\ell}_\eta|_{i,j+1/2} + |\boldsymbol{\ell}_\eta|_{i+1,j+1/2}}\,. \end{aligned} \tag{50}$$

При этом (48) остаётся без изменений, а (47) становится неприменимым.

Точно такая же интерполяция для самих расстояний, $|\boldsymbol{\ell}_\xi|$ и $|\boldsymbol{\ell}_\eta|$, эквивалентна их гармоническому осреднению,

$$\begin{aligned} |\boldsymbol{\ell}_\xi|_{i+1/2,j+1/2} &= \frac{2\,|\boldsymbol{\ell}_\xi|_{i+1/2,j} \cdot |\boldsymbol{\ell}_\xi|_{i+1/2,j+1}}{|\boldsymbol{\ell}_\xi|_{i+1/2,j} + |\boldsymbol{\ell}_\xi|_{i+1/2,j+1}} \\ |\boldsymbol{\ell}_\eta|_{i+1/2,j+1/2} &= \frac{2\,|\boldsymbol{\ell}_\eta|_{i,j+1/2} \cdot |\boldsymbol{\ell}_\eta|_{i+1,j+1/2}}{|\boldsymbol{\ell}_\eta|_{i,j+1/2} + |\boldsymbol{\ell}_\eta|_{i+1,j+1/2}}\,. \end{aligned} \tag{51}$$

Их отношение, $|\boldsymbol{\ell}_\xi|_{i+1/2,j+1/2} \,/\, |\boldsymbol{\ell}_\eta|_{i+1/2,j+1/2}$, в идеале должно равняться отношению $\Delta\xi/\Delta\eta$, и поэтому может служить мерой ошибки изотропии разрешения сетки (применимо только для $\Delta\xi = \Delta\eta$).

Рис. 13 даёт геометрическую иллюстрацию к уравнениям (46) и (50). Построенные в каждой ячейке тонкие линии (левая и средняя панели) визуально пересекаются под углами гораздо ближе к $90°$ – а именно эти углы и принимаются в качестве критерия ошибки. При этом, поскольку коэффициенты интерполяции в (50) вычисляются

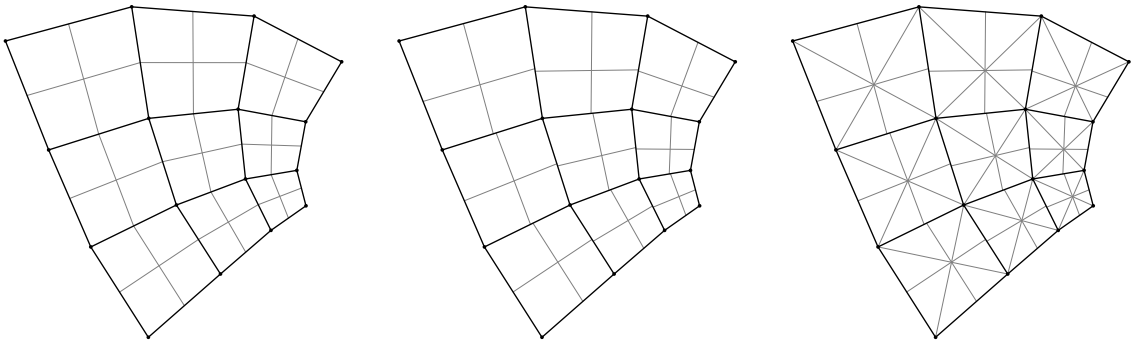


Рис. 13. Критерии оценки ошибок ортогональности. *Слева*: по углам пересечения прямых (показаны тонкими линиями), соединяющих середины сторон ячеек – уравнения (46) и (48). *В центре*: используя интерполяцию (50) и (48). *Справа*: то же что и в центре, но в дополнение построены диагонали в каждой ячейке.

индивидуально для каждой ячейки исходя из соотношения длин её сторон, точки в которых эти прямые пересекаются с общей стороной двух примыкающих ячеек не совпадают. Как и ожидалось, (50), *в центре*, даёт смещение точек пересечения в сторону меньших сторон в обоих направлениях, по сравнению с (46), *слева*. Так же интересно отметить, что точки пересечения этих прямых почти расположены близко к точкам пересечения диагоналей (*справа*), но на самом деле точного совпадения здесь нет.

Определение ошибок ортогональности при помощи *вычисления производных через построение сплайна* является альтернативой конечным разностям. Разностные схемы, рассмотренные выше, обладают вторым порядком точности. Кубический сплайн даёт возможность вычислить производные с четвёртым непосредственно в точках i, j в обоих направлениях, и, таким образом, избежать необходимости осреднения или интерполяции результата.

Для того, чтобы вычислить $\partial_\xi x\big|_{i,j}$ и $\partial_\xi y\big|_{i,j}$ зафиксируем индекс j и рассмотрим последовательность $i = 1, ..., L$, где крайние точки, $i = 1$ и $i = L$ лежат на периметре сетки. Предполагается, что x и y для каждой линии j описывается кубическим полиномом, и для всех точек, кроме крайних, $i = 2, ..., L-1$ ставятся условия на непрерывность второй производной по ξ.

Значения производных в крайних точках определяются заранее. Для этого используется тот же самый алгоритм, что и на рис. 4: все значения $x_{i,j}, y_{i,j}$ на периметре сетки известны, контур выпрямляется в линию, и строится сплайн с периодическими граничными условиями. В результате производные вдоль контура становятся известными. Производные же в перпендикулярном направлении для точек, лежащих на контуре, получаются из условий ортогональности,

$$\partial_\xi x\big|_{i,j\in\partial\mathscr{D}} = \partial_\eta y\big|_{i,j\in\partial\mathscr{D}} \qquad \text{и} \qquad \partial_\xi y\big|_{i,j\in\partial\mathscr{D}} = -\partial_\eta x\big|_{i,j\in\partial\mathscr{D}}\,. \tag{52}$$

Это означает, что проверить таким алгоритмом ортогональность сетки в самих контурных точках нельзя, поскольку она постулируется.

После того, как производные в крайних точках известны, внутренние производные вычисляются сплайном. Точно так же поступаем в направлении η: фиксируем индекс i и ставим условия непрерывности на вторую производную по η во всех точках, кроме крайних. В результате все необходимые производные вычислены, но в отличие от (46), не в серединах ячеек, $i + 1/2, j + 1/2$, а в самих точках сетки i, j. Применяя (48), вычисляем ошибку ортогональности. Пример сетки, построенной таким способом при помощи сплайнов, показан на рис. 14, слева. Единственные данные, которые использовались для построения, это декартовы координаты $(x_{i,j}, y_{i,j})$ 16-ти точек. Визуально, все кривые пересекаются под углами неотличимыми от $90°$, однако заметим, что все четыре угла между смежными сторонами внешнего периметра, а также углы примыкания линий посередине к внешнему периметру в точности прямые именно из-за свойств алгоритма, поэтому в данном случае, проверке на ортогональность подлежат только четыре угла центральной ячейки.

Справа на рис. 14 показана скорость сходимости ошибки ортогональности измеренной тремя методами. Поскольку в данном случае заранее известно, что сетка ортогональная (т.к. задана аналитически), этот пример является тестированием методов измерения ошибки, а не сетки – собственно, в этой ситуации метод должен «распознать» что сетка ортогональная. Как и ожидалось, локальные методы, в которых ошибка ортогональности каждой ячейки сетки вычисляется отдельно, исходя лишь из лишь координат её угловых точек, ограничены вторым порядком точности. По-видимому, ничего существенно более точного, чем (50)–(48), здесь предложить нельзя. Кубический сплайн даёт более точный результат, что не удивительно, потому что сравнивая левую панель рис. 14 с рис. 12, становится очевидным, что сплайн может воспроизвести кривизну линий близкую к сеткам более высокого разрешения – а это типичная ситуация, потому что нашей целью является построение именно *достаточно гладких сеток*, поскольку численные алгоритмы, заложенные в модель океана, рассчитаны именно на такие сетки, чтобы не терять точность. Однако, существенным ограничением сплайнового метода является невозможность оценки

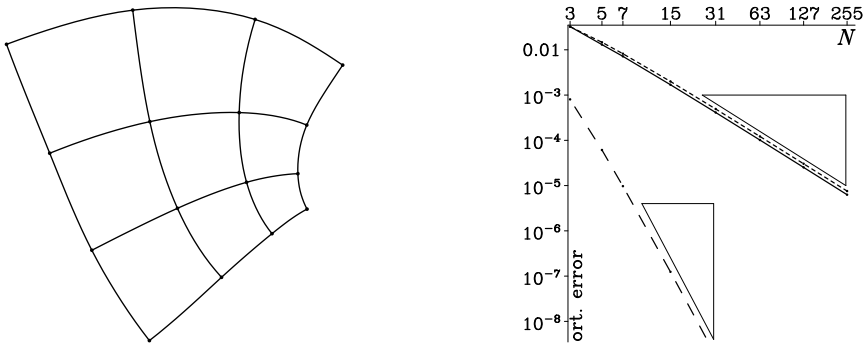


Рис. 14. *Левая панель*: сетка, размером 3×3, состоящая из криволинейных ячеек, построенных кубическими сплайнами. *Справа*: сравнение сходимости ошибки ортогональности вычисленной тремя методами: (46)– верхняя пунктирная линия; (50) – сплошная чуть ниже; сплайном – нижняя пунктирная линия. Треугольниками показаны скорости сходимости соответствующие второму (верхний) и шестому (нижний) порядку точности.

ошибки ортогональности в точках примыкания линий сетки к её внешней границе, а так же искусственное (кажущееся) проникновение вычисляемой ошибки от границы внутрь – становится не очевидным, в каком именно месте ошибка наибольшая, а в нашем случае это происходит чаще всего как раз именно в точках примыкания линий сетки к внешнему контуру – например, из-за его избыточной местной кривизны, вызванной желанием пользователя слишком детально воспроизвести местные особенности береговой линии. Поэтому, несмотря на худшую точность, локальные конечно-разностные методы не теряют своей актуальности.

Для контроля ошибок ортогональности разработана специальная программа `checkort`, в которой реализованы все три алгоритма, описанные выше. Результатом её работы является файл в формате `netCDF`, в котором отклонения от ортогональности, изотропии, а так же остаточная ошибка решения эллиптической задачи, вычисленные в каждой точке сетки, сохраняются как двумерные массивы, которые затем можно проинспектировать.

3. Обсуждение. Особенности применения ортогональных криволинейных сеток для моделирования океана

Пакет программ, созданный в данной работе, даёт возможность строить сетки, наиболее отвечающие требованиям конкретной задачи, однако достижение этой цели не является автоматическим само по себе: в зависимости от конкретной конфигурации приходится прибегать к особым приёмам. В этой части мы рассмотрим некоторые из них.

Об актуальности настоящей работы – востребованности построения ортогональных криволинейных сеток, с одной стороны, и отсутствии программных продуктов надлежащего качества, разработанных для этой цели, с другой – можно судить по следующему примеру: в только что опубликованной статье Androsov et al. (2020), авторы предприняли попытку построить такую сетку с целью моделирования генерации внутренних волн в Ломбокском проливе (это пролив в акватории Малайского архипелага между индонезийскими островами Бали и Ломбок), используя «эллиптический метод с ортогонализацией на границах», со ссылкой на Thompson (1982) (см. так же хорошо известную книгу Thompson et al., 1985). Полученная ими сетка показана на стр. 6 (Figure 3) в их статье. Совершенно очевидно, что по уровню отклонения углов пересечения координатных линий от $90°$, их сетка сопоставима с нашим рис. 6, но при этом не наблюдается «выворачивания» сетки вблизи мысов. Для сравнения, мы построили сетку для данной конфигурации, используя наш алгоритм (рис. 15). При этом, мы старались как можно ближе следовать береговой линии, стараясь воспроизвести все мысы и бухты, чтобы имитировать то, как это сделано в статье Androsov et al. (2020). Отличие получилось весьма значительным. В нашем случае визуальных нарушений ортогональности не наблюдается совсем, есть тенденция к сгущению узлов сетки вблизи мысов и к разряжению в бухтах, а соотношение сторон ячеек сетки постоянно по всей её площади. В сетке из статьи Androsov et al. (2020), такое сгущение/разряжение практически не наблюдается, а ячейки становятся вы-

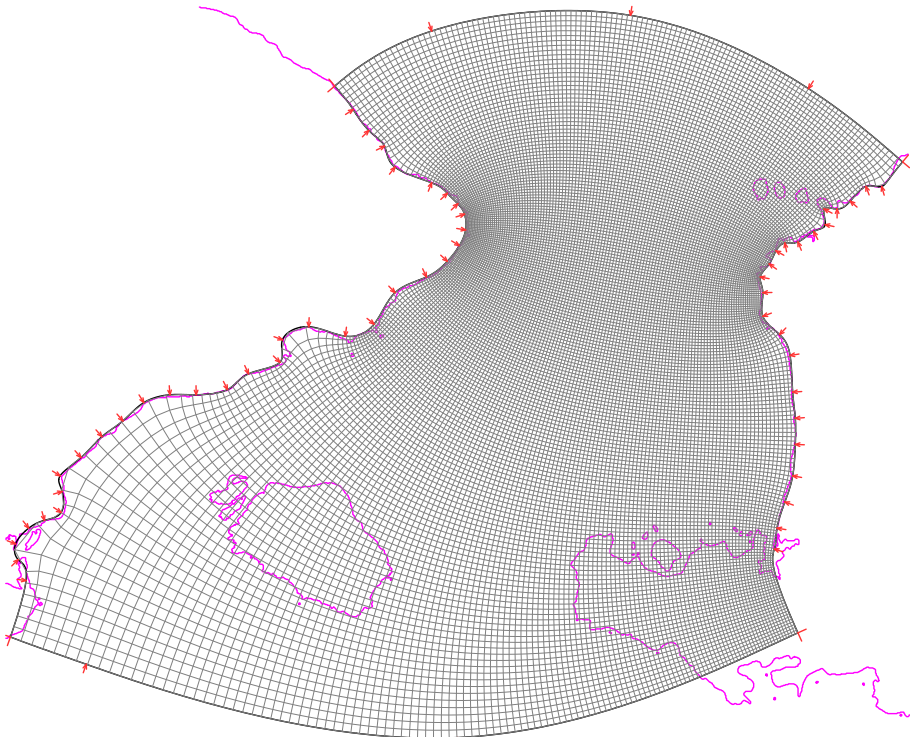


Рис. 15. Сетка для конфигурации Ломбокского пролива, подобная используемой в статье Androsov et al. (2020, их Figure 3), но построенная нашими средствами.

тянутыми в том или другом направлении вблизи береговой линии, в зависимости от её кривизны.

Изучение книги Thompson et al. (1985) даёт понять, что «эллиптический метод» по сути связан с решением уравнения Лапласа релаксационным методом с одновременным перемещением узлов сетки вдоль контура с целью минимизировать отклонение от ортогональности вблизи границы. Теоретически, этот процесс может сойтись и дать ортогональную сетку, однако Figure 3 в Androsov et al. (2020) выглядит так, что до сходимости ещё очень далеко. Справедливости ради отметим, что методы описываемые в Thompson et al. (1985), разрабатывались для задач аэродинамики, где основным приоритетом является обязательное точное следование координатной линией поверхности обтекаемого тела, а некоторое отклонение от ортогональности хоть и нежелательно, но допустимо, т.к. соответствующие коды это учитывают. Большинство моделей океана (в частности все упоминаемые в настоящей работе) подразумевают ортогональность горизонтальных координат. Так же, судя по форме эллиптического оператора для уравнения Пуассона для негидростатической части давления, это же относится и к модели, разработанной и использовавшейся в статье Androsov et al. (2020). Поэтому заимствование методов построения сеток из аэродинамики и использования их для океана может быть не всегда оправдано.

Ещё одно свойство сетки на рис. 15, на которое стоит обратить внимание, это частые сгущения/разрежения сетки из-за изрезанности береговой линии. Это может

быть нежелательным из-за локальных ограничений по числу Куранта, а так же и по физическим соображениям: целью работы Androsov et al. (2020) является генерации и распространение внутренних волн. Растяжение сетки вблизи открытых границ на севере и юге здесь выгодно, так как позволяет поставить искусственные граничные условия как можно дальше от зоны интереса – в самой узкой части пролива. Но там же было бы лучше обеспечить более равномерное разрешение, а этого можно добиться только более плавным контуром, т.е. отказаться от строгого следования береговой линии.

Как видно из предыдущего описания, построенные нами алгоритмы изобилуют итерационными подходами, причём зачастую в сложном, вложенном варианте. Возникает естественный вопрос об их надёжности и устойчивости. Приведённый выше пример для Чёрного моря является случаем построения сетки с довольно умеренной кривизной (отчасти это и является целью), так что сомнений в устойчивости и сходимости не возникает. Чтобы продемонстрировать устойчивость алгоритма, мы модифицируем задачу и построим сетку с быстро меняющийся кривизной контура (рис. 16). Мотивировкой построения такой сетки является желание получить локально высокое разрешение в нужном районе – в данном случае, вокруг Крыма.

Необходимо отметить, что конформное преобразование, отображающее данный контур на прямоугольник, единственно с точностью до размера прямоугольника (но не соотношения его сторон(!)), поэтому, если поставить условие изотропности сетки, то *единственным способом изменять разрешение в нужном месте остаётся манипуляция с кривизной контура* – использование того факта, что узлы сетки стремятся концентрироваться в местах вогнутости контура внутрь сетки (т.е. выпуклости береговой линии вблизи мысов, если контур идёт вдоль неё) и наоборот, разрежаться в бухтах. Данная сетка построена с применением именно этого свойства. Поскольку в местах экстремальной разреженности мы видим значительную кривизну на масштабе ячеек сетки, естественно, не стоит ожидать хорошей точности ортогональности, равно как и точности от модели ROMS (или другого кода), использующего такую сетку. Тем не менее, данный пример показывает, что разработанный нами метод построения конформных сеток обладает значительным «запасом прочности».

В качестве примера сетки с более тонким, но вместе с тем как можно более равномерным разрешением в нужной её части, и с плавным переходом в разреженную область, на рис. 17 показана конфигурация сетки Атлантического океана с фокусом в области Мексиканского залива и Гольфстрима. Частично этот пример мотивирован работой Ezer et al. (2000), где использовался более простой вариант подобной идеи; а так же работой Barkan et al. (2017), целью которой являлось уже именно моделирование с высоким разрешением течений в Мексиканском заливе, для чего использовалась четырёхступенчатая последовательность вложенных сеток:

$$15\,\mathrm{km} \;\to\; 5\,\mathrm{km} \;\to\; 1.5\,\mathrm{km} \;\to\; 0.5\,\mathrm{km}\,,$$

где $15\,\mathrm{km}$ сетка покрывала весь Атлантический океан; $5\,\mathrm{km}$ Карибское море, Мексиканский залив и Гольфстрим; $1.5\,\mathrm{km}$ только Мексиканский залив (целиком), а $0.5\,\mathrm{km}$

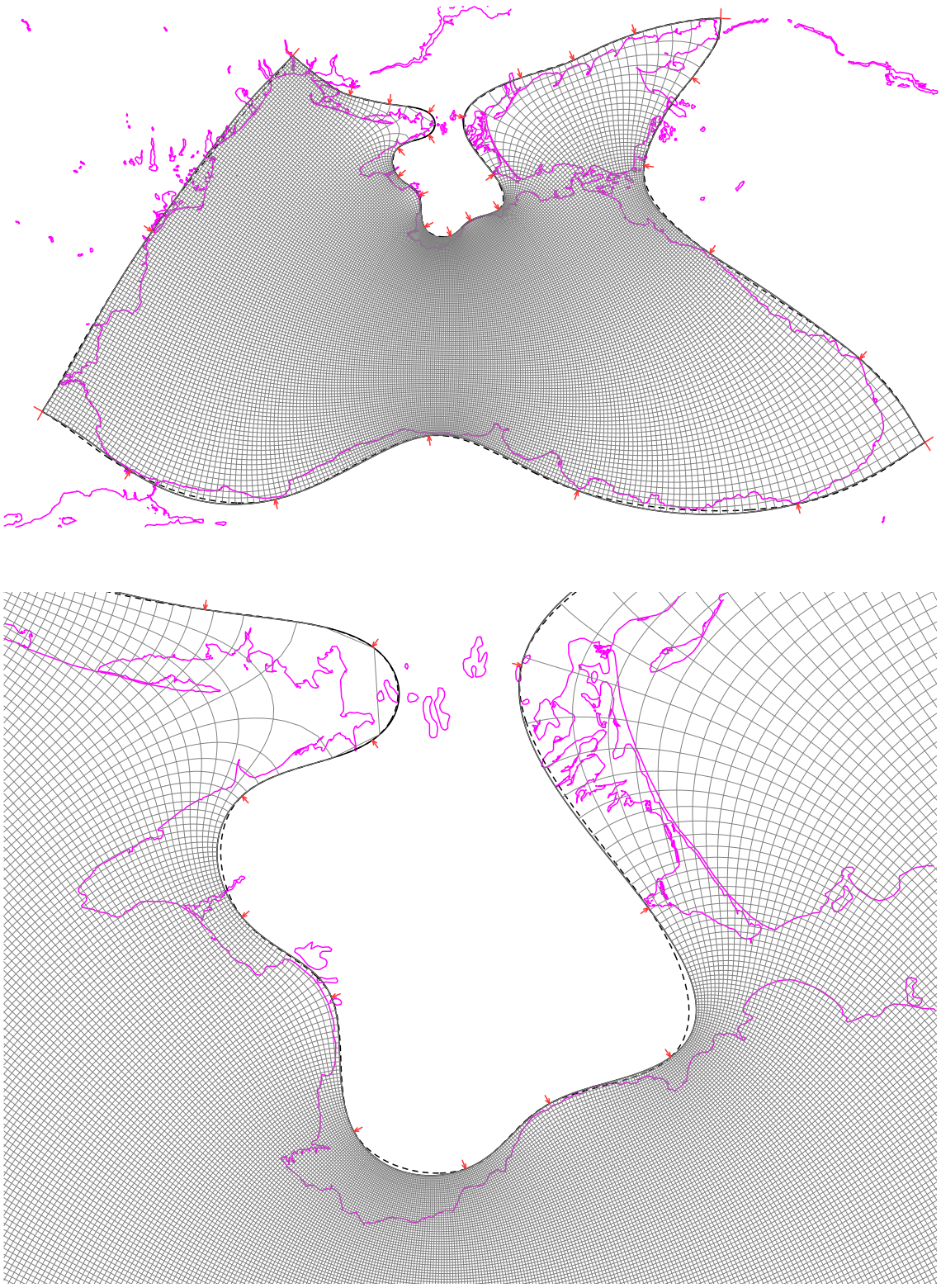


Рис. 16. Ортогональная криволинейная сетка, построенная по контуру, выбранному специально, чтобы получить сгущение точек вокруг Крыма. *Верхняя панель:* разрешение 449×241 точек (показана каждая вторая координатная линия сетки в обоих направлениях). *Нижняя панель:* увеличенный фрагмент части сетки более высокого разрешения, 1703×913, (показана каждая четвёртая линия) геометрически подобной сетке на верхней панели (т.е. обе сетки заполняют один и тот же контур).

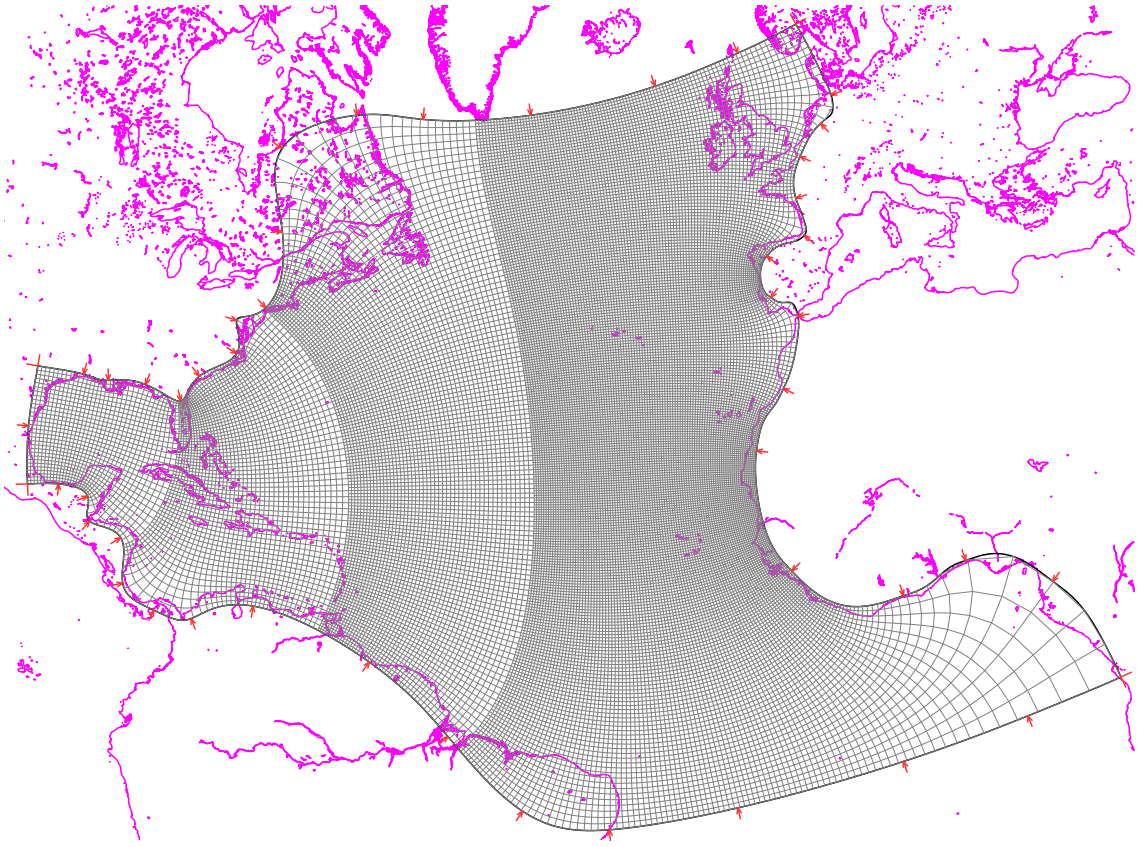


Рис. 17. Сетка модели Атлантического океана с высоким локальным разрешением в области Мексиканского залива и Гольфстрима, сконструированная специально для того, чтобы получить как можно более равномерное разрешение в Мексиканском заливе. Северная и южная открытые границы этой сетки расположены вдоль параллелей $60°$с.ш. и $10°$ю.ш. Для иллюстративных целей, чтобы избежать слияния линий, сетка поделена на 4 зоны, так что в крайней слева (Мексиканский залив) показана только каждая восьмая линия сетки в обоих направлениях, в следующей зоне каждая четвёртая линия, далее каждая вторая, далее все линии. При этом первая зона содержит 45% всех точек сетки; первая и вторая вместе – 70%, первые три вместе – 85%. Полная размерность этой сетки 637×241 точек.

только часть его (см. Fig. 1 в Barkan et al., 2017). При этом на каждом этапе происходило нетривиальное воздействие внешнего решения через открытые границы – собственно для внутренней сетки Гольфстрим задавался как граничное условие. Использование сетки на рис. 17 позволяет получить решение, по крайней мере, на $1.5\,\mathrm{km}$ сетке в один этап, причем с более высоким качеством, т.к. полностью исключаются численные эффекты на границах вложенных сеток, а процедура из односторонней становится фактически двухсторонней[4]. Но при этом сконструированная криволинейная сетка должна обладать новым, по сравнению с Ezer et al. (2000), качеством: её разрешение внутри Мексиканского залива должно быть как можно ближе к равно-

[4] В работе Barkan et al. (2017) решение, полученное на грубой сетке, покрывающей бóльшую область, используется для задания граничных условий на следующей, более мелкой, покрывающей меньшую область. При этом взаимодействие идёт только в одну сторону – решение на мелкой сетке никак не влияет на решение на грубой.

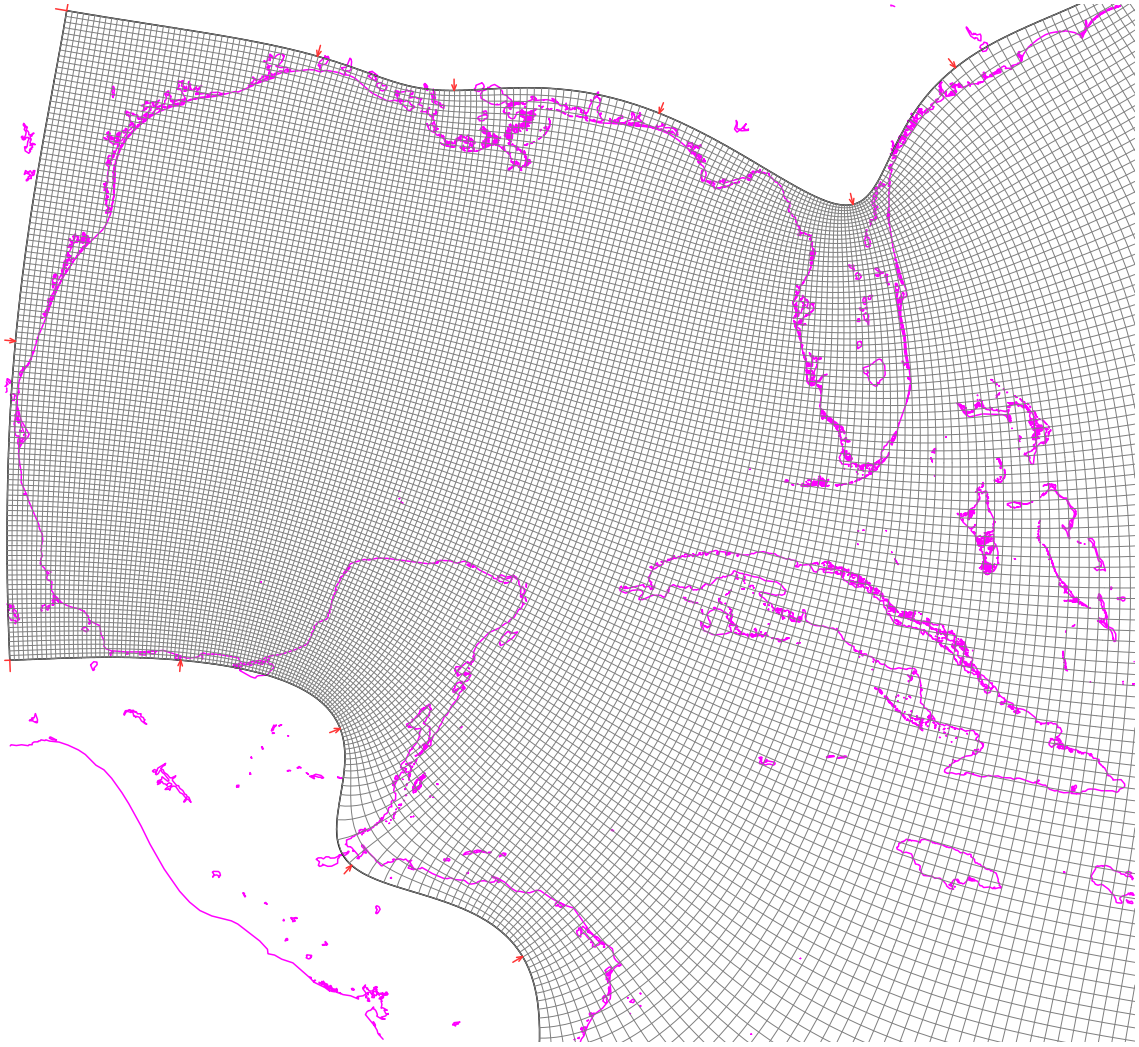


Рис. 18. Часть рис. 17, соответствующая Мексиканскому заливу, но уже с равномерным разрешением (показана каждая вторая линия сетки в обоих направлениях). Заметим, что примерно ∼60% всех точек сетки находятся в этой области.

мерному. Это показано на рис. 18. Отметим несколько приёмов, позволяющих этого добиться. В целом, контур на рис. 18 отслеживает береговую линию, однако не заходит полностью на мысы Флорида и Юкотан – глубина проникновения контура как раз подобрана так, чтобы сконцентрировать разрешение вблизи выпуклости контура ровно настолько, чтобы сделать его близким к равномерному, при этом область непосредственно вблизи выпуклости контура, где разрешение избыточно высокое, оказывается закрыта земляной маской. Подобный же приём используется вблизи мыса Гаттерас – выпуклость контура притягивает разрешение, но ровно настолько, чтобы сделать его более-менее равномерным вдоль побережья от Флориды до мыса Гаттерас. Далее оно неизбежно становится более грубым. Наоборот, большие области разрежения, связанные с необходимостью повернуть направление контура к параллели, расположены так, чтобы закрыть их земляной маской (Ньюфаундленд на

севере и побережье Бразилии на юге на рис. 17).

Таким образом, несмотря на то, что идея применения конформных преобразований для построения сеток известна давно, специфика построения сеток для моделирования океана отличается от, например, задач обтекания гладкого тела в аэродинамике, где основной целью было сделать так, чтобы одна из координатных линий проходила точно по поверхности тела (это необходимо для хорошего разрешения поверхностного слоя). В случае океана, береговые линии сильно изрезаны, и заставить гладкий контур проходить точно вдоль них и невозможно, и не нужно. Вместо этого, приоритетной становится задача – добиться получения более высокого разрешения в местах, где это целесообразно, при одновременном стремлении минимизировать использование земляной маски (но не исключая её совсем), и необходимости избежать избыточной локальной неравномерности разрешения. Т.е. окончательная конфигурация сетки всегда является результатом компромисса, полученного методом проб и ошибок, и неизбежно связана с принятием взвешенных решений человеком, оставаясь, в этом смысле, «предметом искусства». Соответственно, разработанный здесь пакет программ, хотя и существенно облегчает задачу, но по сути является лишь инструментом.

4. Заключение: чего удалось добиться?

В результате этой работы создан пакет программного обеспечения для построения ортогональных криволинейных сеток для модели океана ROMS (потенциально совместим и с другими, в частности NEMO), который выгодно отличается от ранее существующих:

(*i*) удалось добиться математически точной ортогональности углов сопряжения криволинейных сторон периметра сетки – проблема, оставшаяся без удовлетворительного решения за последние 25 лет. Это даёт возможность избавиться от неконтролируемого сгущения или разбега узлов сетки вблизи её углов, а также уменьшить ошибки ортогональности сетки до уровня, недостижимого ранее и сравнимого с ошибками аналитически построенных сеток;

(*ii*) степень гладкости контура: построение сплайнами 3-го или 5-го порядков; непрерывность производных вплоть до 2-го и 4-го порядков в опорных точках;

(*iii*) непрерывность тех же производных на углах контура: поворот одной из криволинейных сторон на 90 градусов вокруг угловой точки, который превращает угол в плавную кривую такой же степени гладкости, как и для «прямых» опорных точек;

(*iv*) распределение узлов сетки на контуре получается решением обратной задачи: требуется расположить точки на контуре таким образом, что конформное преобразование контура в прямоугольник сделает это распределение равномерным;

(*v*) решается итерационным методом, построенным вокруг алгоритма выпрямления Ives & Zakharias, при этом сам алгоритм полностью переработан;

(*vi*) последующее заполнение сетки решением задачи Дирихле, используя дискретизацию типа «мерштеленферфарен», обеспечивающую компактный четвёртый порядок точности;

(*vii*) алгоритмы, связанные с затратными вычислениями – Ives & Zakharias и решение эллиптической задачи – распараллелены для многоядерных CPU;

(*viii*) пакет построен по принципу *compile once – use forever* для использования наподобие «чёрного ящика». Там, где необходимо ПО третьей стороны, – для визуализации и иных целей – применяется только открытое (*open source*) ПО.

Автор выражает благодарность Российскому Фонду Фундаментальных Исследований за поддержку данной работы в рамках гранта РФФИ 19-05-00459 «Короткопериодная изменчивость в аэробной зоне и ее влияние на запас кислорода в присклоновых водах Чёрного моря» и, частично, гранта 18-05-80089 «Моделирование последствий техногенных катастроф в океане». Эта работа так же получила частичное финансирования в рамках темы госзадания № 0149-2019-0004 Института Океанологии им. П.П. Ширшова. Автор признателен трём рецензентам за внимательное прочтение статьи, их замечания и неформальное отношение, литературному редактору журнала Юлии Воробьевой и техническому редактору Наталье Ляховой. Хочется отметить многочисленные и полезные обсуждения данной темы на форуме сообщества пользователей модели ROMS, в ходе которых и был выявлен достаточный общественный интерес к разработке данного пакета программ, именно, как открытого кода, который будет доступен для всех, кому он может быть полезен.

Дополнение 1: Описание программы `izogrid`

Программа `izogrid` (Isotropic-resolution, Ives & Zakharias grid), разработанная для построения ортогональных криволинейных сеток, является частью общего пакета для обработки данных модели ROMS (в совокупности примерно 50 выполнимых программ, 170 файлов, более 50 тыс. линий на ФОРТРАНе), и тесно интегрирована в него. Для её работы необходимы: операционная система Linux, с установленным компилятором ФОРТРАНА, совместимыми библиотеками `NetCDF` и `NCAR Graphics`, а также данными береговых линий `GSHHS` в исходном двоичном формате. Так же используются стандартные для Linux средства для работы с графическими файлами – мы использовали `ps2epsi` и `epstopdf` для конвертирования `PostScript` файлов в PDF формат, а так же `okular` для визуализации, но могут использоваться и другие утилиты. Заметим, что пакет `NCAR Graphics` может создать PDF файл напрямую (давая при этом несколько худшее его качество), и, кроме того, имеет свой собственный интерфейс для визуализации, который так же может быть использован при работе с `izogrid`. Сама же программа `Izogrid` построена по принципу *compile-once – use-forever*, т.е., фактически может использоваться как чёрный ящик.

`Izogrid` требует один аргумент – имя файла входных параметров, пример которого приведен на рис. 3. Файл состоит из преамбулы (шапки) и последовательности координат опорных точек сплайна, количество которых может быть произвольным.

Ниже описано назначение каждого параметра.

`mode` – выбор режима, возможные значения `0,1,2,3,4,5` определяют этап на котором прекращается выполнение программы. Результатом является графический файл для `mode=0,...,4`, а так же `NetCDF` файл сетки в декартовых координатах для `mode=4` и `5`.

`mode=0` – построение географической карты (рис. 2). Целью данного шага является подбор параметров (см. ниже) проекции карты, наиболее подходящий для выбранной области.

`mode=1` – построение контура, рис. 3. На этом этапе пользователь изменяет количество и координаты опорных точек контура во входном файле, запускает программу каждый раз и визуально проверяет результат, с целью добиться желаемой конфигурации контура. Этот этап наиболее затратный по времени, из-за большого числа повторений, однако время выполнения программы для каждой попытки составляет от долей до единиц секунд. Нужно подчеркнуть, что если контур окажется самопересекающимся – а это в принципе возможно при соответствующем задании координат опорных точек контурного сплайна – то задача нахождения конформного преобразования окажется математически некорректной (сингулярной), а алгоритм IZ89 неустойчивым. Поэтому, перед тем как перейти к следующим этапам, пользователь должен убедиться, что самопересечений нет.

`mode=2` и `mode=3` – проверочные режимы: программа останавливается после заполнения сторон контура равноудаленными точками и после их перераспределения конформным преобразованием (рис. 10, *верхняя* и *нижняя* панели соответственно).

`mode=4` и `mode=5` – прохождение всех этапов построения сетки с созданием `NetCDF` файла. Отличие `mode=4` от `mode=5` в том, что в последнем случае графический файл не создается.

`gshhs_data=filename.b` – задаёт разрешение данных береговой линии. Допустимые значения: полное имя файла данных GSHHS, либо одна из букв `c, l, i, h, f` (по начальным буквам "coarse "low "intermediate "high "full") соответствующих выбранному разрешению (детализации) береговой линии. Значение по умолчанию соответствует среднему разрешению (intermediate).

`proj=`*type* – тип картографической проекции (Меркатор, Ламберт, Стереографическая – соответствует двухбуквенной кодировке библиотеки `NCAR Graphics`. Допустимы только конформные проекции.

`rlat=`*lat*, `rlon=`*lat*, `rota=`*angle* – географические координаты (широта, долгота) центра и азимут угла поворота картографической проекции (например, повернутой проекции Меркатора).

`south_edge=`*value*, `north_edge=`*value*, `west_edge=`*value*, `east_edge=`*value* – границы выбранной географической области (влияет только на графическое изображение, но не на саму будущую сетку). Так же, задавая эти параметры, можно построить изображение только части области, покрываемой сеткой, но в увеличен-

ном размере, давая возможность более точно подобрать координаты опорных точек сплайна в конкретном месте (например, рис. 16, *нижняя панель*, был получен именно таким образом).

`latlongrid=`*value* – размер ячеек географической сетки (широта–долгота) на изображении карты, например, на рис. 2 это 2 градуса, соответственно `latlongrid=2`.

`uscale=`*value* – масштабирующий множитель декартовых координат. Выбирается на этапе `mode=1` для того, чтобы привести декартову систему координат к более удобному численному диапазону. Этим параметром можно изменять шаг «миллиметровки» (рис. 3).

`spline_type=`*value* – тип сплайна для построения контура. Допустимые значения `3,5,4` для кубического, сплайна полиномами пятого порядка и обоих сразу соответственно. В последнем случае для построения сетки используется сплайн пятого порядка (т.е. построенная сетка полностью идентична для значений `4` и `5`), но контур, образованный кубическим сплайном, так же показывается на графике.

`nx=`*value*, `ny=`*value* – размер (число точек в каждом направлении) будущей сетки. При этом `nx` является лишь начальным приближением – окончательное значение определяется из условия изотропности разрешения сетки и подбирается автоматически в ходе внешнего итерационного цикла вокруг алгоритма IZ89.

`npass=`*value* – множитель, задающий количество итераций алгоритма IZ89 (как внешнего, так и внутреннего циклов, поскольку они связаны между собой). Эмпирический диапазон допустимых значений 4...10. Ме́ньшая величина приводит к увеличению ошибок алгоритма из-за раннего прерывания итерационного процесса. Бо́льшая – к более длительным вычислениям. Практически всегда можно добиться сходимости на уровне ошибок округления двойной точности. Использование быстрого (но менее точного) режима полезно, если необходимо сделать грубую прикидку, чтобы впоследствии, улучшив подбор опорных точек, сделать бо́лее точный окончательный расчет.

`lwidth=`*value* – множитель толщины линий на графике. Значение по умолчанию `lwidth=1`. Возможность выбора толщины позволяет избежать слияния линий, а так же обеспечить более точное задание точек контура относительно береговой линии.

`xygrid=`*finename* имя создаваемого файла (в NetCDF формате), содержащего декартовы координаты узлов будущей сетки. Эта сетка впоследствии конвертируется в сферическую. По умолчанию `xygrid=xygrid.nc`. Такая двухступенчатая процедура нужна для того, чтобы можно было легко диагностировать уровень ошибок ортогональности и изотропии будущей сетки.

Дополнение 2: Эрмитовы сплайны третьего и пятого порядка

Несмотря на то, что сплайн-интерполяция довольно широко представлена в литературе, нам не удалось найти источник, наиболее подходящий для наших целей, достаточно полный, но, вместе с тем, удачно сочетающий в себе математическую простоту и алгоритмическую эффективность, поэтому здесь ниже приводится наше собственное описание.

Задача сплайн-интерполяции заключается в том, чтобы для последовательности значений f_k, определенной в точках s_k,

$$\{f_k = f(s_k), \quad k = 1, ..., N\}\,, \qquad \Delta s_{k+1/2} = s_{k+1} - s_k > 0\,, \quad \Delta s_{k+1/2} \neq \text{const} \tag{53}$$

построить непрерывную функцию $f = f(s)$, представляющую собой полином третьей (пятой) степени от независимой переменной s в пределах каждого сегмента $[s_k, s_{k+1}]$, которая принимает значения f_k во всех точках s_k, а её производные вплоть до второго, (четвёртого для сплайнов пятого порядка) непрерывны. Как мы скоро увидим, задача сводится к нахождению значений производных в тех же точках,

$$d_k = {}^{\partial f}\!/_{\partial s}\big|_{s=s_k}\,, \qquad \delta''_k = {}^{\partial^2 f}\!/_{\partial s^2}\big|_{s=s_k}\,, \tag{54}$$

через которые затем вычисляются значения кубического (пятой степени) полинома в произвольном месте внутри каждого интервала, $s \in [s_k, s_{k+1}]$.

Любой ***полином третьей степени***, определенный на интервале $-1/2 \le \xi \le +1/2$, можно представить в виде

$$f(\xi) = f^L \cdot h^L(\xi) + f^R \cdot h^R(\xi) + \widehat{d}^L \cdot g^L(\xi) + \widehat{d}^R \cdot g^R(\xi)\,, \tag{55}$$

где f^L и f^R значения самой функции, а $\widehat{d}^L$ и $\widehat{d}^R$ значения её первых производных на левом и правом концах интервала,

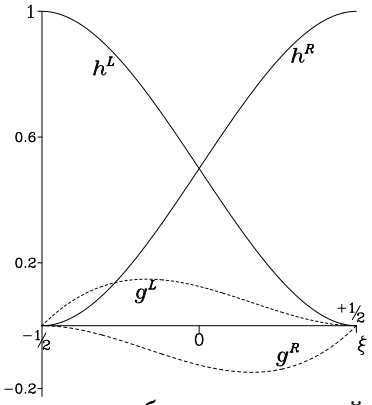
$$f(\xi)\begin{cases}\nearrow & f^R, \quad \xi \to +1/2 \\ \searrow & f^L, \quad \xi \to -1/2\end{cases} \qquad \frac{\partial f}{\partial \xi}\begin{cases}\nearrow & \widehat{d}^R, \quad \xi \to +1/2 \\ \searrow & \widehat{d}^L, \quad \xi \to -1/2\end{cases}\,. \tag{56}$$

Функции $h^L = h^L(\xi)$, $h^R = h^R(\xi)$, $g^L = g^L(\xi)$ и $g^R = g^R(\xi)$ определены так, чтобы они сами, либо их первые производные, обращались в единицу на одном из концов и давали ноль на другом, в соответствие с таблицей:

	значение функции		$\partial/\partial\xi$	
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

Поскольку эти функции сами являются кубическими полиномами, и для каждой из них задано четыре условия в этой таблице, их коэффициенты определены однозначно раз и навсегда,

$$
\begin{aligned}
h^R(\xi) &= \frac{1}{2} + \xi\left(\frac{3}{2} - 2\xi^2\right) \\
h^L(\xi) &= \frac{1}{2} - \xi\left(\frac{3}{2} - 2\xi^2\right) \\
g^R(\xi) &= \left(\xi^2 - \frac{1}{4}\right)\left(\xi + \frac{1}{2}\right) \\
g^L(\xi) &= \left(\xi^2 - \frac{1}{4}\right)\left(\xi - \frac{1}{2}\right).
\end{aligned}
\tag{57}
$$



Это функции составляют *базис Эрмита*. При построении кубического сплайна значения функции в опорных точках известны, значения первых производных в тех же точках получаются из условия непрерывности вторых, поэтому дополним таблицу значениями вторых производных на концах,

	значение функции		$\partial/\partial\xi$		$\partial^2/\partial\xi^2$	
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

Непрерывность первой производной в любой точке s_k получается по построению,

$$
\frac{1}{\Delta s_{k-1/2}} \cdot \underbrace{\lim_{\xi\to+1/2} \frac{\partial f}{\partial \xi}}_{s\in[s_{k-1},s_k]} = \lim_{s\to s_k} \frac{\partial f}{\partial s} = d_k = \lim_{s_k\leftarrow s} \frac{\partial f}{\partial s} = \frac{1}{\Delta s_{k+1/2}} \cdot \underbrace{\lim_{-1/2\leftarrow\xi} \frac{\partial f}{\partial \xi}}_{s\in[s_k,s_{k+1}]},
\tag{58}
$$

коль скоро,

$$
\frac{\widehat{d}^R_{s\in[s_{k-1},s_k]}}{\Delta s_{k-1/2}} = d_k = \frac{\widehat{d}^L_{s\in[s_k,s_{k+1}]}}{\Delta s_{k+1/2}},
\tag{59}
$$

где d_k, $\widehat{d}^R$ и $\widehat{d}^L$ пока неизвестны. Так же отметим, что координата ξ определена индивидуально в пределах каждого сегмента $s \in [s_k, s_{k+1}]$, соответственно

$$
s_k \leftarrow s \quad \Leftrightarrow \quad -1/2 \leftarrow \xi \qquad \text{и} \qquad \xi \to +1/2 \quad \Leftrightarrow \quad s \to s_{k+1},
\tag{60}
$$

в то время как координата s изменяется непрерывно при прохождении стыков между последовательными сегментами

Используя таблицу, условие непрерывности для второй производной,

$$
\frac{1}{\Delta s^2_{k-1/2}} \cdot \underbrace{\lim_{\xi\to+1/2} \frac{\partial^2 f}{\partial \xi^2}}_{s\in[s_{k-1},s_k]} = \lim_{s\to s_k} \frac{\partial^2 f}{\partial s^2} = \lim_{s_k\leftarrow s} \frac{\partial^2 f}{\partial s^2} = \frac{1}{\Delta s^2_{k+1/2}} \cdot \underbrace{\lim_{-1/2\leftarrow\xi} \frac{\partial^2 f}{\partial \xi^2}}_{s\in[s_k,s_{k+1}]},
\tag{61}
$$

получается мгновенно: представляя f в виде (55), дважды дифференцируя и заменяя вторые производные h^L, h^R, g^L, g^R на их значения из таблицы для $\xi = +1/2$ в левом

интервале $s \in [s_{k-1}, s_k]$, и для $\xi = -1/2$ в правом, $s \in [s_k, s_{k+1}]$,

$$\overbrace{\frac{6f^L - 6f^R + 2\widehat{d}^L + 4\widehat{d}^R}{\Delta s^2_{k-1/2}}}^{\xi=+1/2,\ s\in[s_{k-1},s_k],\ s\to s_k} = \overbrace{\frac{-6f^L + 6f^R - 4\widehat{d}^L - 2\widehat{d}^R}{\Delta s^2_{k+1/2}}}^{s_k\leftarrow s,\ s\in[s_k,s_{k+1}],\ \xi=-1/2} .$$

Используя соотношения (59), заменяем $f^L, f^R, \widehat{d}^L, \widehat{d}^R$ на фактические переменные

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

после чего разделяем известные и неизвестные переменные,

$$\frac{d_{k-1}}{\Delta s_{k-1/2}} + \left(\frac{2}{\Delta s_{k-1/2}} + \frac{2}{\Delta s_{k+1/2}}\right)\cdot d_k + \frac{d_{k+1}}{\Delta s_{k+1/2}} = 3\cdot\left(\frac{f_k - f_{k-1}}{\Delta s^2_{k-1/2}} + \frac{f_{k+1} - f_k}{\Delta s^2_{k+1/2}}\right) . \quad (62)$$

Получившиеся уравнения для вычисления d_k и применимы к индексам $k = 2, ..., N-1$ (кроме первой и последней точки), однако, поскольку в данной работе нас интересует построение сплайнов только с периодическими граничным условиями, уравнения (62) можно считать так же применимыми и к $k = 1$, и $k = N$, при условии циклической замены индексов, выходящих за пределы $1 \le k \le N$,

$$k = 1 \ \begin{cases} d_{k-1} \to d_{N-1} \\ f_{k-1} \to f_{N-1} \end{cases} \qquad \text{и} \qquad k = N \ \begin{cases} d_{k+1} \to d_1 \\ f_{k+1} \to f_1 \,, \end{cases} \quad (63)$$

кроме того, входящие в крайние уравнения $\Delta s_{N+1/2} \equiv \Delta s_{1/2}$ так же нуждаются в доопределении, что сделано согласно ур. (7) в основном тексте.

Для построение сплайнов ***полиномами пятой степени*** вначале определим базисные функции Эрмита. По аналогии с (55), представим

$$\begin{aligned} f(\xi) = f^L\cdot H^L(\xi) + f^R\cdot H^R(\xi) + \left.\frac{\partial f}{\partial\xi}\right|^L\cdot G^L(\xi) + \left.\frac{\partial f}{\partial\xi}\right|^R\cdot G^R(\xi) \\ + \left.\frac{\partial^2 f}{\partial\xi^2}\right|^L\cdot D^L(\xi) + \left.\frac{\partial^2 f}{\partial\xi^2}\right|^R\cdot D^R(\xi) \end{aligned} \quad (64)$$

где шесть функций, $H^L(\xi)$, $H^R(\xi)$, $G^L(\xi)$, $G^R(\xi)$, $D^L(\xi)$ и $D^R(\xi)$, определены таким образом, что они сами, их первые и вторые производные, обращаются в 0 или 1 на левой и правой границах интервала $\xi \in [-1/2; +1/2]$ согласно таблице:

	значение функции		$\partial/\partial\xi$		$\partial^2/\partial\xi^2$	
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

из чего следует, что на левой и правой границах интервала, сама функция (64), а так же её первая и вторая производные, соответственно обращаются в f^L, f^R, $\partial f/\partial\xi|^L$, $\partial f/\partial\xi|^R$, $\partial^2 f/\partial\xi^2|^L$ и $\partial^2 f/\partial\xi^2|^R$, т.е. в коэффициенты перед $H^L(\xi)$, ..., $D^R(\xi)$.

Сами же функции $H^L(\xi)$, ..., $D^R(\xi)$ пока неизвестны, но однозначно находятся из их свойств, согласно таблице. Пусть

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

тогда три правые функции, H^R, G^R и D^R, можно искать в виде полиномов пятой степени,

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

что обеспечивает обращение в ноль их величины и значений производных на левом конце. На правом конце, очевидно,

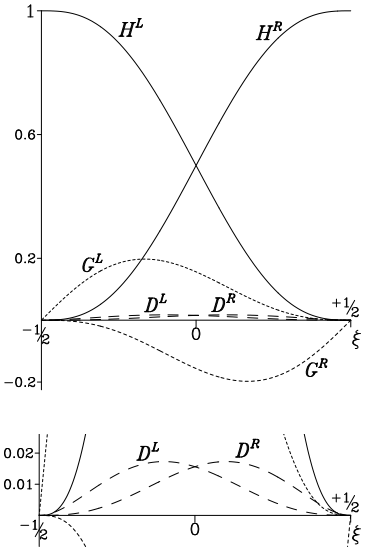
$p=1$							H^R	G^R	D^R
$\mathscr{P}(p)$	$=$	A	$+$	B	$+$	C	$=$	1	0	0
$\partial\mathscr{P}/\partial p$	$=$	$5A$	$+$	$4B$	$+$	$3C$	$=$	0	1	0
$\partial^2\mathscr{P}/\partial p^2$	$=$	$20A$	$+$	$12B$	$+$	$6C$	$=$	0	0	$1\,,$

что и определяет коэффициенты A, B, C для всех трёх функций. Левые функции H^L, G^L, D^L получаются из условий симметрии,

$$\begin{array}{rcrrr} H^R(p) & = & 6p^5 & -15p^4 & +10p^3 \\ G^R(p) & = & -3p^5 & +7p^4 & -4p^3 \\ D^R(p) & = & p^5/2 & -p^4 & +p^3/2 \end{array} \qquad \begin{array}{l} H^L(p) = H^R(1-p) \\ G^L(p) = -G^R(1-p) \\ D^L(p) = D^R(1-p) \end{array} \tag{65}$$

подставив $\xi = p - 1/2$, получаем,

$$H^R(\xi) = \frac{1}{2} + \xi\left(\frac{15}{8} - 5\xi^2 + 6\xi^4\right)$$
$$H^L(\xi) = \frac{1}{2} - \xi\left(\frac{15}{8} - 5\xi^2 + 6\xi^4\right)$$
$$G^R(\xi) = \left(\frac{1}{4} - \xi^2\right)\left[\xi\left(3\xi^2 - \frac{7}{4}\right) + \left(\frac{\xi^2}{2} - \frac{5}{8}\right)\right]$$
$$G^L(\xi) = \left(\frac{1}{4} - \xi^2\right)\left[\xi\left(3\xi^2 - \frac{7}{4}\right) - \left(\frac{\xi^2}{2} - \frac{5}{8}\right)\right]$$
$$D^R(\xi) = \frac{1}{2}\left(\frac{1}{4} - \xi^2\right)^2\left(\frac{1}{2} + \xi\right)$$
$$D^L(\xi) = \frac{1}{2}\left(\frac{1}{4} - \xi^2\right)^2\left(\frac{1}{2} - \xi\right)$$



либо, выразив их через p и $q = 1 - p$, получаем симметричную форму, наиболее

удобную для вычислений,

$$\begin{aligned} H^R &= p^3(1+3q+6q^2)\,, \qquad G^R = -p^3\,(q+3q^2)\;, \qquad D^R = p^3q^2/2\,, \\ H^L &= q^3(1+3p+6p^2)\,, \qquad G^L = +q^3\,(p+3p^2)\;, \qquad D^L = q^3p^2/2\,. \end{aligned} \tag{66}$$

После того как базисные функции стали известны, найдем

$$\begin{aligned} \partial^3 H^R/\partial p^3 &= 360p^2 - 360p + 60 & \partial^4 H^R/\partial p^4 &= 720p - 360 \\ \partial^3 G^R/\partial p^3 &= -180p^2 + 168p - 24 & \partial^4 G^R/\partial p^4 &= -360p + 168 \\ \partial^3 D^R/\partial p^3 &= 30p^2 - 24p + 3 & \partial^4 D^R/\partial p^4 &= 60p - 24 \end{aligned}$$

и дополним таблицу значениями высших производных на концах отрезка,

	значение функции		$\partial/\partial\xi$		$\partial^2/\partial\xi^2$		$\partial^3/\partial\xi^3$		$\partial^4/\partial\xi^4$	
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

Построение сплайна пятого порядка сводится к вычислению производных первого, $d_k = {}^{\partial f}/_{\partial s}\big|_{s=s_k}$, и второго порядка, $\delta''_k = {}^{\partial^2 f}/_{\partial s^2}\big|_{s=s_k}$, в точках s_k, $k = 1, ..., N$ таким образом, чтобы интерполяционные полиномы (64). определенные в каждом интервале $s \in [s_k, s_{k+1}]$ по значениям функции, f_k, f_{k+1}, и этих производных, d_k, d_{k+1}, δ''_k и δ''_{k+1}, обеспечивали непрерывность всех производных, вплоть до четвёртого порядка включительно, во всех точках s_k. Таким образом, $\{d_k, \delta''_k \;\; k = 1...., N\}$ являются неизвестными. Прежде всего заметим, что, аналогично (59), независимо от значений d_k, δ''_k, сама функциональная форма (64) обеспечивает непрерывность первой и второй производной если выполнены условия

$$\underbrace{\frac{1}{\Delta s_{k-1/2}}\cdot\frac{\partial f}{\partial\xi}\bigg|^R}_{s\in[s_{k-1},s_k]} = d_k = \frac{\partial f}{\partial s}\bigg|_{s=s_k} = \underbrace{\frac{1}{\Delta s_{k+1/2}}\cdot\frac{\partial f}{\partial\xi}\bigg|^L}_{s\in[s_k,s_{k+1}]}, \tag{67}$$

и

$$\underbrace{\frac{1}{\Delta s^2_{k-1/2}}\cdot\frac{\partial^2 f}{\partial\xi^2}\bigg|^R}_{s\in[s_{k-1},s_k]} = \delta''_k = \frac{\partial^2 f}{\partial s^2}\bigg|_{s=s_k} = \underbrace{\frac{1}{\Delta s^2_{k+1/2}}\cdot\frac{\partial^2 f}{\partial\xi^2}\bigg|^L}_{s\in[s_k,s_{k+1}]}, \tag{68}$$

что, в свою очередь, связывает коэффициенты (64) с d_k, δ''_k, d_{k+1} и δ''_{k+1}, если речь идет об интервале $s \in [s_k, s_{k+1}]$.

Условие непрерывности третьей производной,

$$\underbrace{\frac{1}{\Delta s_{k-1/2}^3}\cdot\left.\frac{\partial^3 f}{\partial\xi^3}\right|^R}_{s\in[s_{k-1},s_k]}=\left.\frac{\partial^3 f}{\partial s^3}\right|_{s=s_k}=\underbrace{\frac{1}{\Delta s_{k+1/2}^3}\cdot\left.\frac{\partial^3 f}{\partial\xi^3}\right|^L}_{s\in[s_k,s_{k+1}]}, \tag{69}$$

после постановки f через (64), замены коэффициентов при $H^R,...,D^L$ на d_k и δ_k'' с соответствующими индексами, используя (67) и (68), и замены третьих производных $H^R,...,D^L$ на соответствующие значения $\partial^3/\partial\xi^3\big|^{R,L}$ из таблицы, приводит к

$$\overbrace{\frac{60f_k-60f_{k-1}-36\Delta s_{k-1/2}d_k-24\Delta s_{k-1/2}d_{k-1}+9\Delta s_{k-1/2}^2\delta_k''-3\Delta s_{k-1/2}^2\delta_{k-1}''}{\Delta s_{k-1/2}^3}}^{\text{подставим }\xi=+1/2,\ \ k-1\text{ вместо }{}^L,\ \ k\text{ вместо }{}^R,\ \ s\in[s_{k-1},s_k],\ \ s\to s_k}$$
$$=\underbrace{\frac{60f_{k+1}-60f_k-24\Delta s_{k+1/2}d_{k+1}-36\Delta s_{k+1/2}d_k+3\Delta s_{k+1/2}^2\delta_{k+1}''-9\Delta s_{k+1/2}^2\delta_k''}{\Delta s_{k+1/2}^3}}_{s_k\leftarrow s,\ \ s\in[s_k,s_{k+1}],\ \ \text{подставим }\xi=-1/2,\ \ k\text{ вместо }{}^L,\ \ k+1\text{ вместо }{}^R},$$

что после разделения переменных приводится к виду,

$$\begin{gathered}-\frac{2}{5}\cdot\frac{d_{k-1}}{\Delta s_{k-1/2}^2}-\frac{3}{5}\cdot\left(\frac{1}{\Delta s_{k-1/2}^2}-\frac{1}{\Delta s_{k+1/2}^2}\right)\cdot d_k+\frac{2}{5}\cdot\frac{d_{k+1}}{\Delta s_{k+1/2}^2}\\ -\frac{1}{20}\cdot\frac{\delta_{k-1}''}{\Delta s_{k-1/2}}+\frac{3}{20}\cdot\left(\frac{1}{\Delta s_{k-1/2}}+\frac{1}{\Delta s_{k+1/2}}\right)\cdot\delta_k''-\frac{1}{20}\cdot\frac{\delta_{k+1}''}{\Delta s_{k+1/2}}\\ =\frac{f_{k+1}-f_k}{\Delta s_{k+1/2}^3}-\frac{f_k-f_{k-1}}{\Delta s_{k-1/2}^3}.\end{gathered} \tag{70}$$

Аналогичным образом, непрерывность четвёртой производной,

$$\underbrace{\frac{1}{\Delta s_{k-1/2}^4}\cdot\left.\frac{\partial^4 f}{\partial\xi^4}\right|^R}_{s\in[s_{k-1},s_k]}=\left.\frac{\partial^4 f}{\partial s^4}\right|_{s=s_k}=\underbrace{\frac{1}{\Delta s_{k+1/2}^4}\cdot\left.\frac{\partial^4 f}{\partial\xi^4}\right|^L}_{s\in[s_k,s_{k+1}]} \tag{71}$$

после постановки f через (64) и замены четвёртых производных $H^R,...,D^L$ на их значения из таблицы, даёт

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

соответственно,

$$\frac{7}{15}\cdot\frac{d_{k-1}}{\Delta s^3_{k-1/2}}+\frac{8}{15}\cdot\left(\frac{1}{\Delta s^3_{k-1/2}}+\frac{1}{\Delta s^3_{k+1/2}}\right)\cdot d_k+\frac{7}{15}\cdot\frac{d_{k+1}}{\Delta s^3_{k+1/2}}$$
$$+\frac{1}{15}\cdot\frac{\delta''_{k-1}}{\Delta s^2_{k-1/2}}+\frac{1}{10}\cdot\left(\frac{1}{\Delta s^2_{k-1/2}}-\frac{1}{\Delta s^2_{k+1/2}}\right)\cdot\delta''_k-\frac{1}{15}\cdot\frac{\delta''_{k+1}}{\Delta s^2_{k+1/2}} \tag{72}$$
$$=\frac{f_{k+1}-f_k}{\Delta s^4_{k-1/2}}+\frac{f_k-f_{k-1}}{\Delta s^4_{k+1/2}}\,.$$

Взятые вместе, уравнения (70) и (72) можно представить в матричном виде,

$$\boldsymbol{a}_k\cdot\boldsymbol{d}_{k-1}+\boldsymbol{b}_k\cdot\boldsymbol{d}_k+\boldsymbol{c}_k\cdot\boldsymbol{d}_{k+1}=\boldsymbol{f}_k \tag{73}$$

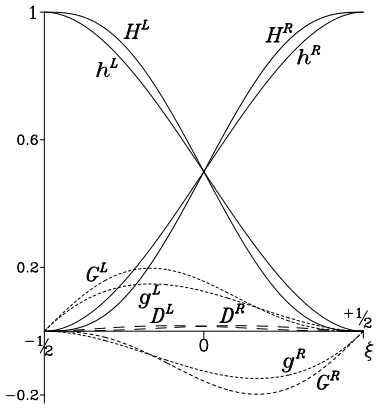
где $\boldsymbol{d}_k$ – вектор неизвестных, состоящий из двух элементов, $\boldsymbol{f}_k$ – вектор правой части, $\boldsymbol{a}_k$, $\boldsymbol{b}_k$ и $\boldsymbol{c}_k$ – матрицы размером 2×2,

$$\boldsymbol{a}_k=\begin{pmatrix}\dfrac{7}{15\Delta s^3_{k-1/2}} & \dfrac{1}{15\Delta s^2_{k-1/2}}\\ -\dfrac{2}{5\Delta s^2_{k-1/2}} & -\dfrac{1}{20\Delta s_{k-1/2}}\end{pmatrix}\qquad \boldsymbol{c}_k=\begin{pmatrix}\dfrac{7}{15\Delta s^3_{k+1/2}} & -\dfrac{1}{15\Delta s^2_{k+1/2}}\\ \dfrac{2}{5\Delta s^2_{k+1/2}} & -\dfrac{1}{20\Delta s_{k+1/2}}\end{pmatrix}$$
$$\boldsymbol{b}_k=\begin{pmatrix}\dfrac{8}{15}\cdot\left(\dfrac{1}{\Delta s^3_{k-1/2}}+\dfrac{1}{\Delta s^3_{k+1/2}}\right) & \dfrac{1}{10}\cdot\left(\dfrac{1}{\Delta s^2_{k-1/2}}-\dfrac{1}{\Delta s^2_{k+1/2}}\right)\\ -\dfrac{3}{5}\cdot\left(\dfrac{1}{\Delta s^2_{k-1/2}}-\dfrac{1}{\Delta s^2_{k+1/2}}\right) & \dfrac{3}{20}\cdot\left(\dfrac{1}{\Delta s_{k-1/2}}+\dfrac{1}{\Delta s_{k+1/2}}\right)\end{pmatrix}$$
$$\boldsymbol{d}_k=\begin{pmatrix}d_k\\ \delta''_k\end{pmatrix}\qquad \boldsymbol{f}_k=\begin{pmatrix}\dfrac{f_{k+1}-f_k}{\Delta s^4_{k-1/2}}+\dfrac{f_k-f_{k-1}}{\Delta s^4_{k+1/2}}\\ \dfrac{f_{k+1}-f_k}{\Delta s^3_{k+1/2}}-\dfrac{f_k-f_{k-1}}{\Delta s^3_{k-1/2}}\end{pmatrix},$$

причём здесь мы поменяли порядок уравнений – (72) становится верхним, а (70) нижним – так, чтобы сделать матрицу $\boldsymbol{b}_k$ диагонально доминантной (если $\Delta s_{k-1/2}=\Delta s_{k+1/2}$ то она становится диагональной). Как и в случае кубического сплайна (62), уравнения (73), так же как и (70) и (72), применимы для всех $k=1,...,N$ при условии циклической замены индексов, выходящих за пределы $1\le k\le N$, (63). Никакие другие варианты краевых условий для $k=1$ и $k=N$ в данной работе не нужны, и поэтому не рассматриваются. Задача (73) является блочной трёхдиагональной системой уравнений с условиями циклического замыкания и решается методом матричной прогонки, который в целом аналогичен обычной (не матричной) прогонке для задачи (62)–(63), с той лишь разницей, что вместо умножения чисел появляется умножение матрицы на вектор, а там где в обычной прогонке имеет место деление коэффициентов, в матричной используется нахождение обратной матрицы и перемножение получившейся матрицы на другую матрицу или вектор.

На этом описание алгоритмов построения сплайнов закончено, однако стоит обратить внимание на следующие моменты:

(*i*) Кубический сплайн (62), построенный выше, математически эквивалентен описанному в литературе, например, Press et al. (1992, Sec. 3.3), несмотря на то, что последний выражен в других переменных – через вторые производные, которые находятся из условия непрерывности первых.



(*ii*) Несмотря на визуальное сходство Эрмитовых функций для кубических, h^L, h^R, g^L, g^R, и полиномов пятой степени, H^L, H^R, G^L, G^R, они различны. Это показано на рисунке справа.

(*iii*) В случае построения сплайна для интерполяции функции одной переменной $f = f(s)$, независимая координата s определена *a priori*. Однако, нас интересует построение кривой на плоскости, $(x, y) = \big(x(\sigma), y(\sigma)\big)$, где σ – это *некая* координата вдоль кривой, а входными данными являются лишь декартовы координаты упорядоченного набора точек $\big(x_k, y_k\big)$. Таким образом, возникает неоднозначность как определить координату σ и сопоставить этим точкам значения σ_k. Казалось бы, наиболее естественным является определить σ как длину пути вдоль кривой, s, приняв за начало какую-нибудь из точек, скажем $\big(x_1, y_1\big)$. Но для того, чтобы знать длину пути, нужно знать саму кривую, т.е. сплайн должен быть *уже построен*, а чтобы построить сплайн, нужно *сначала задать* s_k.

Дилемма (*iii*) в целом напоминает деликатную ситуацию, в которую попал известный рассказчик правдивых историй (нижняя иллюстрация[5] справа), и для которой он быстро нашёл элегантное решение, вертикально подняв себя из болота за собственную косичку, причём вместе с лошадью. Поскольку в настоящее время сплайны на плоскости очень широко применяются в компьютерной графике, то естественным следующим шагом будет обратиться к тому, как эта проблема решается там и, по возможности, адаптировать подходящий алгоритм. Однако, даже беглое изучение материалов на эту тему (например Marschner, 2013) показывает, что прак-

[5] Иллюстрация Гюстава Доре́ (Paul Gustave Doré), 1832-1883.

тически всегда речь идет о параметрическом задании кривой, $(x,y) = \big(x(t), y(t)\big)$, где параметр t задаётся произвольным образом, никак не связанным ни с длиной пути, ни с дифференцированием вдоль неё. Свойства же гладкости определенного порядка в точках сопряжения сегментов (т.е. непрерывность самих функций и их производных вплоть до определенного порядка) получаются за счёт соответствующего выбора базисных функций (кривые Безьé, B-сплайны, и т.д.), и в этом смысле происходит отход от методологии построения классических одномерных сплайнов, основанный исключительно на стремлении получить непрерывность как можно большего порядка производных, оставаясь в рамках полиномов определенной степени. В частности, за кубическим сплайном стоят такие свойства, как определение первых производных методом компактного дифференцирования (более точного, чем конечные разности того же порядка), а так же вариационный принцип минимизации функционала интеграла квадрата второй производной. Как правило, эти свойства игнорируются в компьютерной графике. Поэтому рассмотрим подробнее.

Заметим, что нас интересует не непрерывность первых и вторых производных как таковых, а плавность построенной кривой, т.е. отсутствие излома при переходе от одного сегмента к другому, а так же непрерывность кривизны. Пусть σ – произвольно определенная координата вдоль кривой на плоскости, $(x,y) = \big(x(\sigma), y(\sigma)\big)$. Тогда единичный вектор касательной к кривой можно записать в виде,

$$\boldsymbol{\ell} = \frac{\big(\partial x/\partial\sigma, \partial y/\partial\sigma\big)}{\sqrt{(\partial x/\partial\sigma)^2 + (\partial y/\partial\sigma)^2}}\,, \qquad |\boldsymbol{\ell}| \equiv 1. \tag{74}$$

Если в качестве σ выбрать *настоящую длину пути вдоль кривой*, s, то очевидно

$$\sqrt{(\partial x/\partial s)^2 + (\partial y/\partial s)^2} \equiv 1 \qquad \text{и} \qquad \boldsymbol{\ell} = \big(\partial x/\partial s, \partial y/\partial s\big)\,, \tag{75}$$

а для произвольной координаты σ, приращения длины пути и координаты связаны как

$$ds = d\sigma \cdot \sqrt{(\partial x/\partial\sigma)^2 + (\partial y/\partial\sigma)^2}\,. \tag{76}$$

Если производные $\partial x/\partial\sigma$ и $\partial y/\partial\sigma$ вычисляются через уравнения (62), в случае построения кубического сплайна, где x и y выступают в роли f, а σ в роли s (т.е. заданы последовательные $\Delta\sigma_{k+1/2}$, причем уравнения для x и y имеют *одинаковые коэффициенты в левой части*), либо через уравнения (70) и (72) с аналогичной подстановкой, в случае сплайна пятого порядка, то

$$\frac{\partial x/\partial\sigma}{\partial x/\partial s} = \frac{\partial y/\partial\sigma}{\partial y/\partial s} = \frac{\sqrt{(\partial x/\partial\sigma)^2 + (\partial y/\partial\sigma)^2}}{\sqrt{(\partial x/\partial s)^2 + (\partial y/\partial s)^2}} = \frac{ds}{d\sigma}\,, \tag{77}$$

поэтому единичный вектор $\boldsymbol{\ell} = \boldsymbol{\ell}(\sigma)$ ведет себя непрерывно при переходе через σ_k от одного сегмента к другому,

$$\lim_{\sigma\to\sigma_k} \boldsymbol{\ell}(\sigma) = \lim_{\sigma_k\leftarrow\sigma} \boldsymbol{\ell}(\sigma)\,, \tag{78}$$

даже если

$$\lim_{\sigma\to\sigma_k}\frac{ds}{d\sigma}\neq\lim_{\sigma_k\leftarrow\sigma}\frac{ds}{d\sigma}\,, \tag{79}$$

терпит разрыв. Чтобы представить себе такую ситуацию, допустим, что на двух последовательных сегментах кривой, $[\sigma_{k-1},\sigma_k]$ и $[\sigma_k,\sigma_{k+1}]$, введено разбиение отрезка $[\sigma_{k-1},\sigma_{k+1}]$ на равные, бесконечно малые интервалы, $\delta\sigma = const$, $\delta\sigma \ll \sigma_k-\sigma_{k-1},\ \sigma_{k+1}-\sigma_k$. Соответствующие им приращения расстояний вдоль кривой δs уже не будут равномерными, но плавно изменяющимися в пределах своего сегмента. Однако при переходе через точку σ_k, т.е. из одного сегмента в другой, возможно скачкообразное изменение последовательны расстояний δs. Тем не менее, излома кривой, т.е. резкого изменения направления касательного вектора $\boldsymbol{\ell}$ в точке σ_k при этом не происходит.

Рассмотрим, как изменяется кривизна при перемещении вдоль кривой. Для компактности записи обозначим,

$$\dot{x}=dx/d\sigma\,,\qquad \dot{y}=dy/d\sigma\,,\qquad \ddot{x}=d^2x/d\sigma^2\,,\qquad \ddot{y}=d^2y/d\sigma^2\,, \tag{80}$$

тогда приращение единичного вектора вдоль кривой,

$$\begin{aligned} d\boldsymbol{\ell}=\frac{d}{d\sigma}\left(\frac{(\dot{x},\dot{y})}{\sqrt{\dot{x}^2+\dot{y}^2}}\right)\cdot d\sigma &= \left(\frac{(\ddot{x},\ddot{y})}{\sqrt{\dot{x}^2+\dot{y}^2}}-(\dot{x},\dot{y})\cdot\frac{\dot{x}\ddot{x}+\dot{y}\ddot{y}}{\left(\sqrt{\dot{x}^2+\dot{y}^2}\right)^3}\right)\cdot d\sigma \\ &= \frac{(\ddot{x}\dot{y}-\dot{x}\ddot{y})\cdot(\dot{y},\,-\dot{x})}{\left(\sqrt{\dot{x}^2+\dot{y}^2}\right)^3}\cdot d\sigma=\frac{\ddot{x}\dot{y}-\dot{x}\ddot{y}}{\dot{x}^2+\dot{y}^2}\cdot\boldsymbol{n}\cdot d\sigma\,, \end{aligned} \tag{81}$$

где $\boldsymbol{n}=(\dot{y},\,-\dot{x})/\sqrt{\dot{x}^2+\dot{y}^2}$ это единичный вектор нормали к кривой, $\boldsymbol{n}\perp\boldsymbol{\ell}$, направленный влево перпендикулярно $\boldsymbol{\ell}$. Поскольку касательный вектор $\boldsymbol{\ell}$ всё время остается единичным, единственная степень свободы, которой он обладает, – это изменение направления, поэтому он сам и его приращение всегда взаимно перпендикулярны, $d\boldsymbol{\ell}\perp\boldsymbol{\ell}$. Вектор кривизны,

$$\frac{d\boldsymbol{\ell}}{ds}=\frac{1}{\sqrt{\dot{x}^2+\dot{y}^2}}\cdot\frac{d\boldsymbol{\ell}}{d\sigma}=\frac{(\ddot{x}\dot{y}-\dot{x}\ddot{y})\cdot(\dot{y},-\dot{x})}{\left(\sqrt{\dot{x}^2+\dot{y}^2}\right)^4}=\frac{\ddot{x}\dot{y}-\dot{x}\ddot{y}}{\left(\sqrt{\dot{x}^2+\dot{y}^2}\right)^3}\cdot\boldsymbol{n}=\frac{1}{r}\cdot\boldsymbol{n}\,, \tag{82}$$

является производной угла направления касательной при перемещении вдоль кривой, а введенный таким образом радиус кривизны r считается положительным, если кривая поворачивает влево по ходу движения в положительном направлении s, и отрицательным, если вправо. Точка, находящаяся на расстоянии r в направлении $\boldsymbol{n}$ от точки на кривой, является центром окружности, касающейся кривой. Заметим,

что

$$\frac{\ddot{x}\dot{y}-\dot{x}\ddot{y}}{\left(\sqrt{\dot{x}^2+\dot{y}^2}\right)^3}=\frac{1}{(ds/d\sigma)^3}\cdot\left\{\left[\frac{d}{ds}\left(\frac{dx}{ds}\cdot\frac{ds}{d\sigma}\right)\right]\cdot\frac{ds}{d\sigma}\cdot\left(\frac{dy}{ds}\cdot\frac{ds}{d\sigma}\right)\right.$$
$$\left.-\left(\frac{dx}{ds}\cdot\frac{ds}{d\sigma}\right)\cdot\left[\frac{d}{ds}\left(\frac{dy}{ds}\cdot\frac{ds}{d\sigma}\right)\right]\cdot\frac{ds}{d\sigma}\right\} \tag{83}$$
$$=\frac{d^2x}{ds^2}\cdot\frac{dy}{ds}-\frac{d^2y}{ds^2}\cdot\frac{dx}{ds}$$

т.е. кривизну, вычисленную через произвольную координату σ, можно полностью выразить через производные по настоящей длине вдоль кривой s,

$$\frac{d\boldsymbol{\ell}}{ds}=\frac{d^2x}{ds^2}\cdot\frac{dy}{ds}-\frac{d^2y}{ds^2}\cdot\frac{dx}{ds}\,. \tag{84}$$

Соображения, приведённые выше, дают некоторую гибкость в выборе координаты σ, поскольку основные свойства сплайна – гладкость двумерной кривой, понимаемая как отсутствие изломов (непрерывность направления) и отсутствие резких изменений кривизны (её непрерывность вдоль кривой) – возможно обеспечить и при произвольном её определении. Например, если в качестве σ_k использовать индексы точек (x_k,y_k), через которые должна пройти кривая, $\sigma_k=k$, подразумевая, что внутри каждого интервала, $\sigma_k<\sigma<\sigma_{k+1}$, координаты точек кривой $(x,y)=\big(x(\sigma),y(\sigma)\big)$ вычисляются интерполяцией, то уравнения для построения кубического сплайна (62) принимают простой вид,

$$d_{k-1}+4d_k+d_{k+1}=3\left(f_{k+1}-f_{k-1}\right)\,, \tag{85}$$

а уравнения (70) и (72) для сплайна пятого порядка становятся соответственно,

$$-\frac{2}{5}d_{k-1}+\frac{2}{5}d_{k+1}-\frac{1}{20}\delta''_{k-1}+\frac{3}{10}\delta''_{k}-\frac{1}{20}\delta''_{k+1}=f_{k+1}-2f_k+f_{k-1}\,, \tag{86}$$

и

$$\frac{7}{15}d_{k-1}+\frac{16}{15}d_k+\frac{7}{15}d_{k+1}+\frac{1}{15}\delta''_{k-1}-\frac{1}{15}\delta''_{k+1}=f_{k+1}-f_{k-1}\,, \tag{87}$$

где, и в том и в другом случае, в качестве известных f_k в правой части выступают координаты заданных точек, x_k и y_k, через которые должна пройти кривая, а в результате решения двух систем уравнений (85), отдельно для x и y, получаются значения производных $d_k^{(x)}=dx/d\sigma|_{\sigma=\sigma_k}$ и $d_k^{(y)}=dy/d\sigma|_{\sigma=\sigma_k}$ в тех же точках. В результате решения (86)–(87), кроме того, получаются и вторые производные $\delta''^{(x)}_k=d^2x/d\sigma^2|_{\sigma=\sigma_k}$ и $\delta''^{(y)}_k=d^2y/d\sigma^2|_{\sigma=\sigma_k}$.

Поскольку связь вычисленных таким образом производных с «настоящими» dx/ds и dy/ds заранее неизвестна, в частности, если в каждом интервале $\sigma\in[\sigma_k,\sigma_{k+1}]$ задать последовательность M равноудаленных точек $\sigma_{k,m}=k+{}^m\!/_M$, где $m=1,...,M>1$, то соответствующие им точки $\big(x(\sigma_{k,m}),y(\sigma_{k,m})\big)$ уже не будут равноудаленными в пространстве (x,y). Более того, при переходе от одного интервала в другой, $[\sigma_{k-1},\sigma_k]\to[\sigma_k,\sigma_{k+1}]$ возможно скачкообразное изменение расстояния

между последовательными точками, соответствующими $\sigma = k-{}^{1}/_{M},\ k,\ k+{}^{1}/_{M}$, принадлежащими разным интервалам. Так же нет никаких оснований утверждать, что направления касательных векторов, $\boldsymbol{\ell}_k = \left(\dfrac{d_k^{(x)}}{\sqrt{\left(d_k^{(x)}\right)^2 + \left(d_k^{(y)}\right)^2}}, \dfrac{d_k^{(y)}}{\sqrt{\left(d_k^{(x)}\right)^2 + \left(d_k^{(y)}\right)^2}} \right)$, вычисленных через производные, полученные решением (85) или (86)–(87), эквивалентны $\boldsymbol{\ell}_k = \left(\left.\dfrac{dx/ds}{\sqrt{(dx/ds)^2 + (dy/ds)^2}}\right|_{s=s_k}, \left.\dfrac{dy/ds}{\sqrt{(dx/ds)^2 + (dy/ds)^2}}\right|_{s=s_k} \right)$, где производные $dx/ds|_{s=s_k}$ и $dy/ds|_{s=s_k}$ получены решением (62) или (70)–(72), в которых в качестве координаты s используется либо настоящая длина пути вдоль кривой, либо её приближённая версия, но в любом случае последовательные $\Delta s_{k+1/2} \neq const$ не равны друг другу.

Таким образом, уравнения (85) или (86)–(87), довольно привлекательны благодаря своей простоте и хорошей обусловленности (с последней точки зрения даже более привлекательны чем (70)–(72)), но стоит убедиться в том, что кривизна построенных таким образом кривых в пространстве (x, y) действительно ведёт себя непрерывно при переходе из одного сегмента в другой. Что же касается (62) или (70)–(72), то для того, чтобы их применять, необходимо сначала сделать оценку всех $\Delta s_{k+1/2}$, которая была бы как можно ближе к истинной длине сегмента будущей кривой. Для решения и той, и другой задачи заметим, что через каждые три последовательные точки ломаной или кривой можно провести окружность, и в качестве оценки длины сегмента кривой взять длину дуги, соединяющей соседние точки, что является альтернативой измерения расстояния между точками по прямой. Так же можно использовать радиус окружности в качестве оценки кривизны. Обратимся к рис. 19, *левая нижняя панель*. Обозначим,

$$\Delta x_{k+1/2} = x_{k+1} - x_k\,, \qquad \Delta y_{k+1/2} = y_{k+1} - y_k\,, \qquad \Delta s^2_{k+1/2} = \Delta x^2_{k+1/2} + \Delta y^2_{k+1/2}\,,$$

тогда, после несложных вычислений, находим радиус окружности r_k, проходящей через точки (x_{k-1}, y_{k-1}), (x_k, y_k) и (x_{k+1}, y_{k+1}),

$$\frac{1}{r_k} = \frac{2 \cdot \left(\Delta x_{k-1/2} \cdot \Delta y_{k+1/2} - \Delta x_{k+1/2} \cdot \Delta y_{k-1/2}\right)}{\sqrt{\begin{array}{r}\left(\Delta y_{k+1/2} \cdot \Delta s^2_{k-1/2} + \Delta y_{k-1/2} \cdot \Delta s^2_{k+1/2}\right)^2 \\ + \left(\Delta x_{k+1/2} \cdot \Delta s^2_{k-1/2} + \Delta x_{k-1/2} \cdot \Delta s^2_{k+1/2}\right)^2\end{array}}}\,, \tag{88}$$

а также вектор направления из точки (x_k, y_k) в её центр, $\boldsymbol{n}_k = \left(n_k^{(x)}, n_k^{(y)}\right)$,

$$\begin{aligned} n_k^{(x)} &= -\frac{\Delta y_{k+1/2} \cdot \Delta s^2_{k-1/2} + \Delta y_{k-1/2} \cdot \Delta s^2_{k+1/2}}{\sqrt{(.)^2 + (.)^2}} \\ n_k^{(y)} &= +\frac{\Delta x_{k+1/2} \cdot \Delta s^2_{k-1/2} + \Delta x_{k-1/2} \cdot \Delta s^2_{k+1/2}}{\sqrt{(.)^2 + (.)^2}} \end{aligned} \tag{89}$$

где знаменатели точно такие же как и в (88). Заметим, что в выражениях (88) и (89) не возникает деления на ноль, коль скоро $\Delta s^2_{k-1/2} > 0$ и $\Delta s^2_{k+1/2}$. Кривизна $1/r_k$,

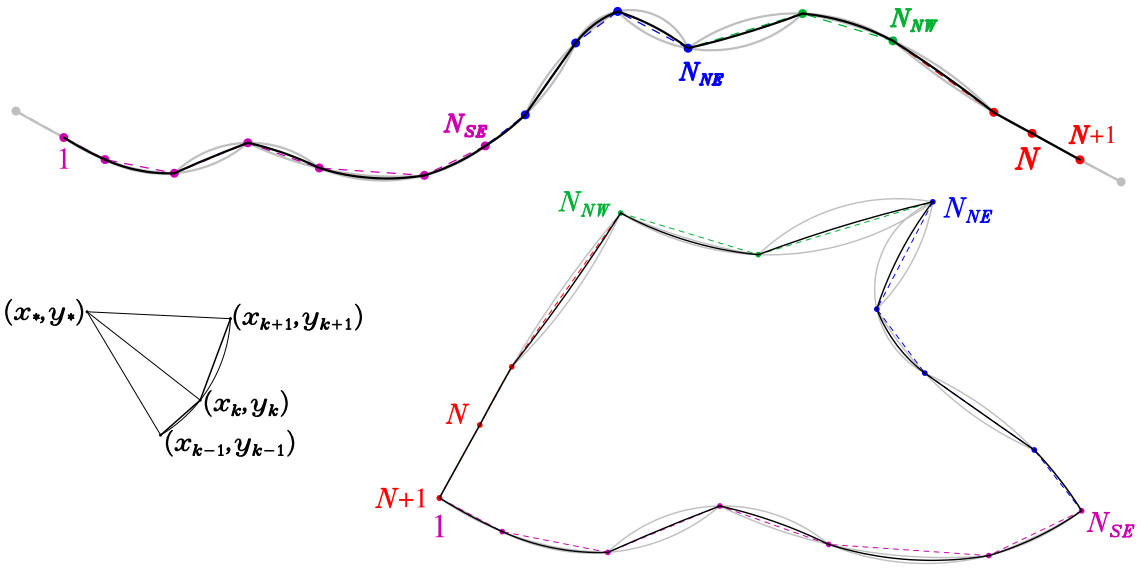


Рис. 19. Приближение контура сегментами окружности как альтернатива ломаной линии. *Внизу слева*: пояснение к алгоритму построения дуги окружности, проходящей через три точки. *Сверху*: выпрямленный контр сетки Чёрного моря. Цвета соответствуют разным сторонам контура, как на рис. 4. Серым цветом справа и слева показаны две дополнительные точки, полученные из условия периодичности. Так же серым цветом показаны дуги окружностей проведенные через каждые три последовательные точки (по две дуги на каждый сегмент ломаной, потому что каждый сегмент участвует в построении окружности для точек на обоих концах). Чёрным цветом показаны дуги окружностей средней кривизны (по одной на каждый сегмент). *Внизу справа*: этот же контур, свернутый обратным преобразованием на рис. 4.

определяемая в (88), может быть положительной или отрицательной, в зависимости от направления поворота сегмента ломаной при движении в сторону увеличения индекса k, а вектор кривизны, аналог (82), определяемый как $d\boldsymbol{\ell}/ds = \boldsymbol{n}_k/r_k$, всегда направлен в сторону центра окружности.

Сверху на рис. 19 показано приближение выпрямленного контура сетки Чёрного моря (рис. 4) дугами окружностей, построенных по трём точкам (*серые линии*) и дугами окружностей средней кривизны,

$$\frac{1}{r_{k+1/2}} = \frac{1}{2}\cdot\left(\frac{1}{r_k} + \frac{1}{r_{k+1}}\right), \tag{90}$$

для каждого сегмента (*чёрные линии*). *Справа внизу* показан тот же контур, но уже преобразованный к нормальному виду. Очевидно, что опорные точки сплайна на рис. 19 расставлены слишком редко, чтобы контур, построенный из сегментов окружностей (*чёрные линии*), был существенно ближе к построенному сплайнами (рис. 4), по сравнению с исходной ломаной, состоящей из прямых сегментов. За исключением, пожалуй, южной стороны, где кривизна меняется более плавно и уже наметилась некоторая тенденция к сходимости. Однако отметим, что приближение основанное на средней кривизне (90), естественным образом получается устойчивым и безопасным, поскольку если трёхточечная кривизна (*серые линии*) в соседних сегментах получается слишком разной по величине, и, тем более, разных знаков, то конечный

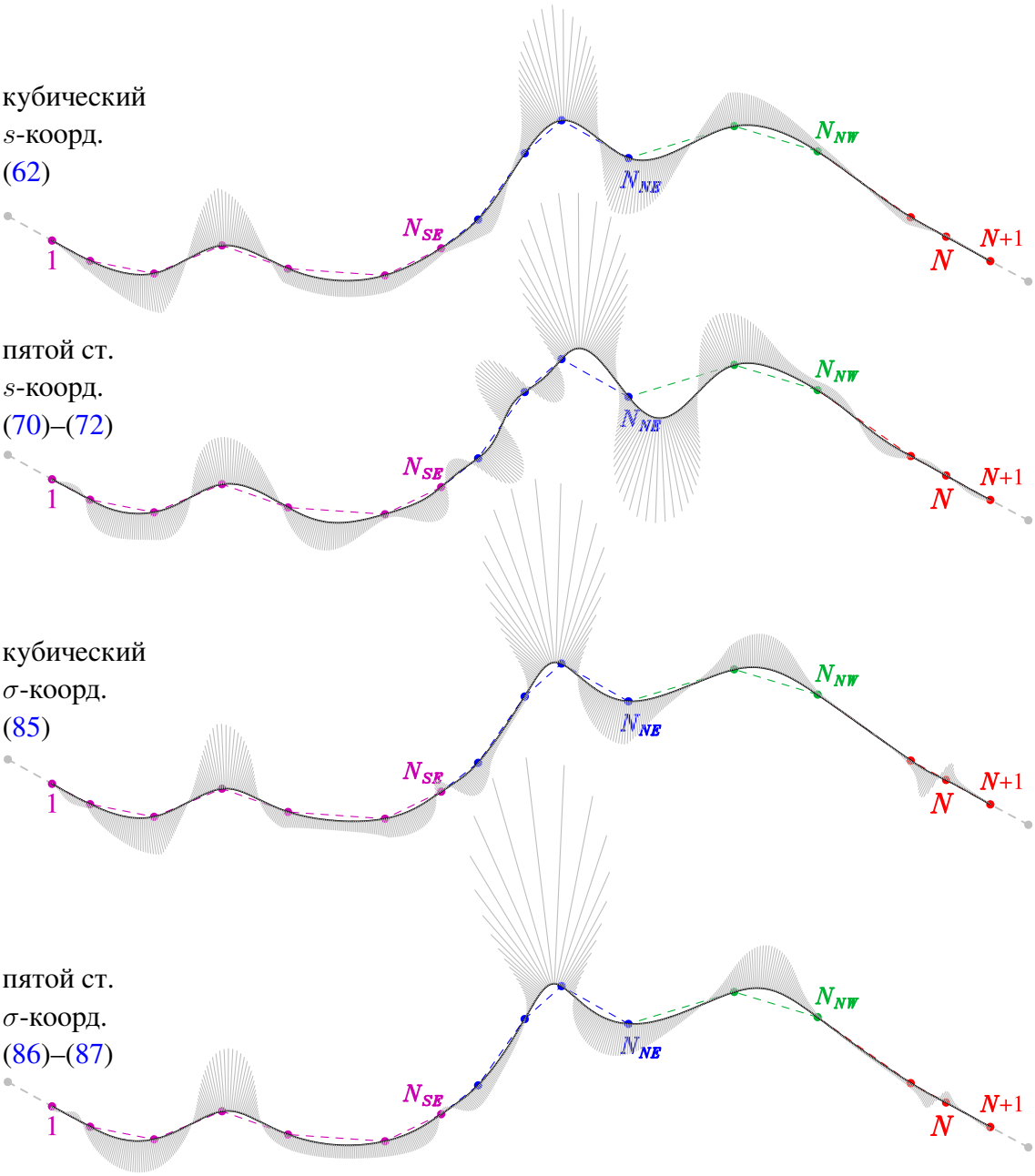


Рис. 20. Серыми линиями показаны векторы кривизны (см. (82), а также (88)–(89)), умноженные для иллюстративных целей на отрицательный масштабирующий множитель для того, чтобы они были направлены не к центру, а в сторону выпуклости кривой, выпущенные из каждой точки выпрямленного контура сетки Чёрного моря построенного алгоритмами обозначенными слева на каждой панели.

результат (*чёрные линии*) всегда ближе прямой, и, в этом смысле, алгоритм обладает некоторыми ограничительными свойствами, препятствующими возникновению ложных колебаний. Длина сегмента дуги, соединяющей две последовательные точки,

$$\Delta l_{k+1/2} = 2r_{k+1/2} \cdot \operatorname{acrsin}\left(\frac{\Delta s_{k+1/2}}{2r_{k+1/2}}\right) \approx \Delta s_{k+1/2}\left[1 + \frac{1}{6}\left(\frac{\Delta s_{k+1/2}}{2r_{k+1/2}}\right)^2 + ...\right], \quad (91)$$

так же не имеет особенности при $r_{k+1/2} \to \infty$, и фактически может рассматриваться как коррекция высоких порядков для расстояния по прямой. Это свойство полезно тем, что при помощи него довольно просто улучшить скорость сходимости итерационного алгоритма определения настоящей длины кривой при помощи самого сплайна (часть 2.2. параграф содержащий уравнение (7)), если ур. (88) и (91) применить так же и для расчёта расстояний между интерполированными точками, вместо того, чтобы считать эти расстояния по прямой.

Чтобы убедиться в том, что кривизна построенного контура изменяется непрерывно при переходе от одного сегмента к другому, обратимся к рис. 20, где распрямлённый контур сетки Чёрного моря построен всеми четырьмя возможными алгоритмами сплайнов: кубическими или полиномами пятой степени, каждый из которых в качестве координаты может использовать либо длину пути вдоль кривой s, либо «индексную» координаты сигма. Во всех случаях кривая была заполнена равноудалёнными точками с малым шагом, используя итерационный алгоритм (23)–(26). Из каждой точки проведен вектор кривизны, вычисленный по трёхточечной схеме (88)–(89), используя две соседние точки. Умножение вектора кривизны на отрицательный множитель даёт возможность избежать сгущения и пересечения линий вблизи центра кривизны. Концы этих векторов должны образовать непрерывную плавную огибающую, без скачков при переходе через границы сегментов, обозначенных цветными точками. Собственно, это и наблюдается. Отметим, что использование истинной длины пути вдоль кривой s требует итераций при построении сплайна (ур. (7 и следующий за ним параграф), и в случае сплайна пятого порядка, при определенных условиях, эти итерации могут быть неустойчивыми, поэтому все $\Delta s_{k+1/2}$ были вычислены кубическим сплайном, а далее по ним уже построен сплайн пятого порядка. Алгоритмы (85) и (86)–(87) не требуют итераций, но тем не менее дают непрерывность кривизны, и по совокупности свойств очень привлекательны.

Литература

Androsov A., Voltzinger N., Kuznetsov I., Fofonova V. Modeling of nonhydrostatic dynamics and hydrology of the Lombok Strait // Water. 2020. Vol. 12. No. 11. Art. 3092. DOI:10.3390/w12113092.

Arakawa A., Lamb V.R. Computational design of the basic dynamical processes of the UCLA general circulation model // Meth. Comput. Phys. 1977. Vol. 17. P. 174–267.

Barkan R., McWilliams J.C., Shchepetkin A.F., Molemaker M.J., Renault L., Bracco A., Choi J. Submesoscale dynamics in the Northern Gulf of Mexico. Part I: Regional and seasonal characterization and the role of river outflow // J. Phys. Oceanogr. 2017. Vol. 47. No. 9. P. 2325–2346. DOI:10.1175/JPO-D-17-0035.1.

Bruciaferri D., Shapiro G., Stanichny S., Zatsepin A., Ezer T., Wobus F., Francis X., Hilton D. The development of a 3D computational mesh to improve the representation of dynamic processes: The Black Sea test case // Ocean Modelling. 2020. Vol. 146. Art. 101534. DOI:10.1016/j.ocemod.2019.

Collatz L. The Numerical Treatment of Differential Equations // Göttingen, Berlin and Heidelberg. Springer-Verlag. 1960. 568 pp. DOI:10.1007/978-3-662-05500-7.

Deriaz E. Compact finite difference schemes of arbitrary order for the Poisson equation in arbitrary dimensions // BIT Numerical Mathematics, Springer Verlag. 2020. Vol. 60. P. 199–233. DOI:10.1007/s10543-019-00772-5. hal-01998201.

Driscoll T.A., Trefethen L.N. Schwartz–Christoffel Mapping // Cambridge University Press. 2002. 132 p. ISBN 0521 807263.

Driscoll T. A., Vavasis S. A. Numerical conformal mapping using cross-ratios and Delaunay triangulation // SIAM J. Sci. Comp. 1998. Vol. 19. No. 6, P. 1783–1803. http://www.math.udel.edu/d̃riscoll/papers/1998-DriscollVavasis-1783.pdf.

Ezer T., Mellor G.L. Sensitivity studies with the North Atlantic sigma-coordinate Princeton Ocean Model // Dyn. Atmos. Oceans. 2000. Vol. 32. P. 185–208. DOI:10.1016/S0377-0265(00)00047-6.

Gan J., Allen J.S. On open boundary conditions for a limited-area coastal model off Oregon. Part 1: Response to idealized wind forcing // Ocean Modelling. 2005. Vol. 8. P. 115–133. DOI:10.1016/j.ocemod.2003.12.006.

Haidvogel D.B., Beckmann A., Hedstrom K.S. Dynamical simulations of filament formation and evolution in the coastal transition zone // J. Geophys. Res. Oceans. 1991. Vol. 96. P. 15,017–15,040. DOI:10.1029/91JC00943.

Hedstrom K. Curvilinear orthogonal grid generation // `Gridpak` web page. 2005 https://marine.rutgers.edu/po/gridpak.html.

Hetland R. `Pygridgen` source code web page // https://phobson.github.io/pygridgen/.

Ives D.C., Zacharias R.M. Conformal mapping and orthogonal grid generation // J. Propulsion and Power. 1989. Vol. 5. No. 3. P. 327–333. DOI:10.2514/3.23156 (AIAA-87-2057).

Marchesiello P., McWilliams J.C., Shchepetkin A.F. Equilibrium structure and dynamics of the California Current System // J. Phys. Oceanogr. 2003. Vol. 33. P. 753–783. DOI:10.1175/1520-0485(2003)33<753:ESADOT>2.0.CO;2.

Marschner S. 2D Spline Curves // Lecture course CS4620. Cornell university. 2013. http://www.cs.cornell.edu/courses/cs4620/2013fa/lectures/16spline-curves.pdf.

Moretti G. Grid generation using classical techniques // Numerical Grid Generation Techniques. NASA Conference publication 2166 / R. E. Smith, editor. Nasa Langley Research Center. Hampton, VA. October 6–7. 1980. 563 p. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19810006176.pdf.

POMWEB/GRID-DATA http://www.ccpo.odu.edu/POMWEB/GRID-DATA/GRID.f.

Press, W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P. Numerical Recipes in Fortran 77. The Art of Scientific Computing, 2nd Edition // Cambridge University Press. 1992. ISBN 0-521-43064-X, 818 p.

Sakov, P. `Gridgen-c` source code web page // https://github.com/sakov/gridgen-c.

Schaffer S. Higher order multi-grid methods // Math. Computation. 1984. Vol. 43. No. 167. P. 89–115. DOI:10.2307/2007401.

Schwartz H.A. Gesammelte mathematische abhandlungen // Berlin. Springer verlag. 1890. 370 p. https://archive.org/details/gesammeltemathem01schwuoft/page/48/mode/2up.

Shchepetkin A.F., McWilliams J.C. The regional oceanic modeling system (ROMS): A split-explicit, free-surface, topography-following-coordinate oceanic model // Ocean Modelling. 2005. Vol. 9. No. 4. P. 347–404. DOI:10.1016/j.ocemod.2004.08.002.

Shchepetkin A. F. Generating ROMS land mask from GSHHS Global Coastline Database. // 2015.

https://www.myroms.org/forum/viewtopic.php?f=23&t=3878&p=14839.

Snyder J. Map Projections: A Working Manual (USGS Professional Paper 1395) // Washignton D.C. US Government printing office. 1987. 385 p. DOI:10.3133/pp1395. http://pubs.er.usgs.gov/publication/pp1395.

Swarztrauber P. N. The methods of Cyclic Reduction, Fourier Analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle // SIAM Review. 1977. Vol. 19. No. 3. P. 490–501. DOI:10.1137/1019071.

Thompson J. F. Numerical Grid Generation // Amsterdam, The Netherlands. Elsevier. 1982.

Thompson J. F., Warsi Z. U. A., Mastin C. Numerical grid generation: Foundations and applications // North-Holland. 1985. 336 p. http://read.pudn.com/downloads218/ebook/1024406/GridGen_by_Thompson.pdf.

Vorobieff P. Schwartz-Christoffel transformation. Lecture notes for University of New Mexico // http://www.me.unm.edu/ kalmoth/ME530-ch4.pdf.

ON THE CONSTRUCTION OF ORTHOGONAL CURVILINEAR GRIDS FOR REGIONAL OCEANIC MODELING: MATHEMATICAL ALGORITHM AND USER GUIDE

A.F. Shchepetkin

Shirshov Institute of Oceanology, Russian Academy of Sciences,
36 Nakhimovskiy prospect, Moscow, 117997, Russia,
and *Moscow Institute of Physics and Technology,*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia,
e-mail: old_galaxy@yahoo.com
Submitted 12.06.2020, accepted 30.11.2020

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. It is based on the numerical solution of the inverse problem by an iterative procedure – finding such distribution of grid points along its perimeter, so that the conformal transformation of the perimeter into a rectangle turns this distribution into uniform one. The iterative procedure itself turns out to be multilevel - i.e. an iterative loop built around another, internal iterative procedure. Thereafter, knowing this distribution, the grid nodes inside the region are obtained solving an elliptic problem. It is shown that it was possible to obtain the exact orthogonality of the perimeter at the corners of the grid, to achieve very small, previously unattainable level of orthogonality errors, as well as make it isotropic – local distances between grid nodes in both directions are equal to each other.

Keywords: orthogonal curvilinear grids on a sphere, conformal mapping, discrete Schwartz–Christoffel transform, multilevel nested iterative methods for inverse problem, regional oceanic modeling

Outline: The article consists of four sections, one of which is subdivided into six subsections, and two additions.

Sec. 1 is an introduction which exposes the current situation of the need and, at the same time, the lack of availability of a reliable tool suitable for building orthogonal

curvilinear grids with sufficiency small orthogonality errors. This section also introduces the basic principles of conformal mapping, and formulates the approach as the combination of direct mapping of the curvilinear region onto a rectangle, within which a Cartesian grid is set up and then transformed back onto curvilinear region by the inverse mapping. It also discussed inherent difficulties of this approach, and proposes modifications to makes it practical.

Sec. 2 describes a practical, step-by-step procedure for building such grids.

Subsec. 2.1 makes conformal projection of geographical region onto flat plane, setting the stage for subsequent specification of curvilinear contour – perimeter of the future grid – which encloses the desired region. This contour is set up by spline going through user-specified reference points.

Subsec. 2.2 introduces a new algorithm to ensure that the corners of curvilinear rectangular contour are exactly orthogonal, regardless of how user specifies the reference points.

Subsec. 2.3 demonstrates the necessity to distribute points of the future grid along the perimeter (defined by spline in Sec. 2.2) in a proper unique way, such that conformal mapping of the contour (now defined as a polygon of spline-interpolated points) onto a rectangle results in uniform distribution of the points along the sides of rectangle. Not doing so results in non-orthogonal grid, even if the subsequent solution of discrete Dirichlet elliptic problem to fill the grid is done exactly. It then introduces principles and algorithms of discrete conformal mapping, and shows the pathway to build orthogonal curvilinear grids via inverse conformal mapping of the rectangle (filled with Cartesian grid) back to curvilinear contour.

Subsec. 2.4 shows that to achieve the goal of properly distributing points along the perimeter, an inverse problem needs to be solved, and a method to do so, based on an iterative procedure build around the algorithm of Ives & Zacharias (1989, hereafter IZ89) is introduced and described in detail. The iterative procedure is needed because algorithm IZ89 is an iterative by itself, and is not directly reversible. As the result, the overall procedure is a double-layer iterative loop, which is potentially computationally expensive. To mitigate the computational cost, the number of iterative repeats of the internal (within IZ89 algorithm itself) loop is initially set to be small, but increases according to a specially optimized rule, and the external loop progresses toward more and more accurate solution. This allows to achieve roundoff-level accuracy at practically acceptable computer run time.

Subsec. 2.5 deals with solution of Dirichlet elliptic problem to fill the grid interior using compact finite-difference method with 9-point discretization of Laplacian operator.

Subsec. 2.6 deals with method of evaluation accuracy – orthogonality errors – of the constructed grids.

Sec. 3 discusses the strategies and provides examples of practical applications for constructing orthogonal curvilinear grids for oceanic modeling.

Sec. 4 is summary.

Addition 1 describes the usage and control parameters of program `izogrid` – a practical tool to build grids developed by this study.

Addition 2 describes algorithms to build cubic and quintic splines, and discussed

the associated theoretical questions relevant to this study.

Acknowledgments: The author expresses gratitude to Russian Fund of Fundamental Research for supporting this work in part via RFFI grants № 19-05-00459 "Short-period variability in the aerobic zone and its influence on the oxygen inventory in the Black Sea slope water"and № 18-05-80089 "Modeling of consequences of technogenic accidents in the ocean". This study also received some partial support from Shirshov Institute of Oceanology state contract № 0149-2019-0004. Author is thankful to three reviewers for their careful reading this rather lengthy manuscript, and their thoughtful comments, which turned out to be very useful. Very tedious and careful work of editorial team of the *Journal of Oceanological Research*, especially Yulia Vorob'yova and Natalya Lyakhova is greatly appreciated. It is worth to mention numerous discussions of this subject – building orthogonal curvilinear grids – on ROMS user community forum web suite, which was both encouraging and stimulating, and which exposed sufficient interest in developing a comprehensive software package, openly available to anybody to whom it may be useful.

References

Androsov, A., N. Voltzinger, I. Kuznetsov, and V. Fofonova. Modeling of nonhydrostatic dynamics and hydrology of the Lombok Strait, *Water*, 2020, Vol. 12, No. 11, art# 3092, doi:10.3390/w12113092.

Arakawa, A. and V. R. Lamb. Computational design of the basic dynamical processes of the UCLA general circulation model, *Meth. Comput. Phys.*, 1977, Vol. 17, pp. 174–267.

Barkan, R., J. C. McWilliams, A. F. Shchepetkin, M. J. Molemaker, L. Renault, A. Bracco, and J. Choi. Submesoscale dynamics in the Northern Gulf of Mexico. Part I: Regional and seasonal characterization and the role of river outflow. *J. Phys. Oceanogr.*, 2017, Vol. 47, No. 9, pp. 2325–2346, doi:10.1175/JPO-D-17-0035.1.

Bruciaferri, D., G. Shapiro, S. Stanichny, A. Zatsepin, T. Ezer, F. Wobus, X. Francis, and D. Hilton. The development of a 3D computational mesh to improve the representation of dynamic processes: The Black Sea test case. *Ocean Modelling*, 2020, Vol. 146, art# 101534, doi:10.1016/j.ocemod.2019.

Collatz, L. The Numerical Treatment of Differential Equations. *Springer-Verlag*, Göttingen, Berlin, and Heidelberg. 1960, 568 p., doi:10.1007/978-3-662-05500-7.

Deriaz, E. Compact finite difference schemes of arbitrary order for the Poisson equation in arbitrary dimensions. *BIT Numerical Mathematics, Springer Verlag*, 2020, Vol. 60, pp. 199–233, doi:10. 1007/s10543-019-00772-5. hal-01998201.

Driscoll, T.A. and L.N. Trefethen. chwartz–Christoffel Mapping. *Cambridge University Press.*. 2002, 132 p., ISBN 0521 807263.

Driscoll, T.A. and S.A. Vavasis. Numerical conformal mapping using cross-ratios and Delaunay triangulation. *SIAM J. Sci. Comp.*, 1998, Vol. 19, No. 6, pp. 1783–1803, http://www.math.udel.edu/đriscoll/papers/1998-DriscollVavasis-1783.pdf.

Ezer, T. and G.L. Mellor. Sensitivity studies with the North Atlantic sigma-coordinate Princeton Ocean Model. *Dyn. Atmos. Oceans.*, 2000, Vol. 32, pp. 185–208, doi:10.1016/S0377-0265(00)00047-6.

Gan, J. and J.S. Allen. On open boundary conditions for a limited-area coastal model off Oregon. Part 1: Response to idealized wind forcing. *Ocean Modelling*, 2005, Vol. 8, pp. 115–133,

doi:10.1016/j.ocemod.2003.12.006.

Haidvogel, D.B., A. Beckmann, and K.S. Hedstrom. Dynamical simulations of filament formation and evolution in the coastal transition zone. *J. Geophys. Res. Oceans*, 1991, Vol. 96, pp. 15,017–15,040, doi:10.1029/91JC00943.

Hedstrom, K. Curvilinear orthogonal grid generation. `Gridpak` web page, 2005, https://marine.rutgers.edu/po/gridpak.html.

Hetland, R. `Pygridgen` source code web page. https://phobson.github.io/pygridgen/.

Ives, D.C. and R.M. Zacharias. Conformal mapping and orthogonal grid generation. *J. Propulsion & Power*, 1989, Vol. 5, No. 3, pp. 327–333, doi:10.2514/3.23156(AIAA-87-2057).

Marchesiello, P., J. C. McWilliams, and A.F. Shchepetkin. Equilibrium structure and dynamics of the California Current System. *J. Phys. Oceanogr.*, 2003, Vol. 33, pp. 753–783, doi:10.1175/1520-0485(2003)33<753:ESADOT>2.0.CO;2.

Marschner, S. 2D Spline Curves. Lecture course CS4620. *Cornell university*, 2013, http://www.cs.cornell.edu/courses/cs4620/2013fa/lectures/16spline-curves.pdf.

Moretti, G.: Grid generation using classical techniques. *Numerical Grid Generation Techniques. NASA Conference publication 2166*, R. E. Smith, editor. Nasa Langley Research Center, Hampton, VA, October 6-7 1980, 563 p., https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19810006176.pdf.

POMWEB/GRID-DATA http://www.ccpo.odu.edu/POMWEB/GRID-DATA/GRID.f.

Press, W.H., S. A. Teukolsky, W.T. Vetterling, and B.P. Flannery. Numerical Recipes in Fortran 77. The Art of Scientific Computing, 2nd Edition. *Cambridge University Press.*, 1992, ISBN 0-521-43064-X, 818 p.

Sakov, P., `Gridgen-c` source code web page. https://github.com/sakov/gridgen-c.

Schaffer, S. Higher order multi-grid methods *Math. Computation.*, 1984, Vol. 43, No. 167, pp. 89–115, doi:10.2307/2007401.

Schwartz, H. A. Gesammelte mathematische abhandlungen. *Springer verlag*, Berlin, 1890, 370 p. https://archive.org/details/gesammeltemathem01schwuoft/page/48/mode/2up.

Shchepetkin, A.F. and J.C. McWilliams, The regional oceanic modeling system (ROMS): A split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean Modelling*, 2005, Vol. 9, No. 4, pp. 347–404, doi:10.1016/j.ocemod.2004.08.002.

Shchepetkin A.F. Generating ROMS land mask from GSHHS Global Coastline Database, 2015, https://www.myroms.org/forum/viewtopic.php?f=23&t=3878&p=14839.

Snyder, J. Map Projections: A Working Manual (USGS Professional Paper 1395), *US Government printing office*, Washignton D.C., 1987, 385p., doi:10.3133/pp1395, http://pubs.er.usgs.gov/publication/pp1395.

Swarztrauber, P.N., 1977: The methods of Cyclic Reduction, Fourier Analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. *SIAM Review*, 1977, Vol. 19, No. 3, pp. 490–501, doi:10.1137/1019071.

Thompson, J.F. Numerical Grid Generation. *Elsevier*, Amsterdam, The Netherlands, 1982.

Thompson, J.F., Z.U.A. Warsi, and C. Mastin. Numerical grid generation: Foundations and applications. *North-Holland*, 1985, 336 p., http://read.pudn.com/downloads218/ebook/1024406/GridGen_by_Thompson.pdf.

Vorobieff, P. Schwartz-Christoffel transformation. *Lecture notes for University of New Mexico*, http://www.me.unm.edu/ kalmoth/ME530-ch4.pdf.