РЕЦЕНЗИЯ №2

на статью **«Построение ортогональных криволинейных сеток для регионального моделирования океана: математический алгоритм и руководство пользователя»** автора**: Щепеткин А.Ф.**

В статье «Построение ортогональных криволинейных сеток для регионального моделирования океана: математический алгоритм и руководство пользователя» описывается алгоритм построения ортогональной сетки для моделей океана.

Достоинства этой статьи заключается в важности подобной задачи, и ее практической реализации. Важность данной работы хорошо аргументирована автором во введении, и рецензенту тут добавить нечего.

Рекомендация рецензента редакционной коллегии журнала: опубликовать данную статью.

Некоторые разделы можно рекомендовать включить в курс «Вычислительная математика» для студентов математических специальностей.

Комментарии рецензента

Тем не менее, у рецензента к автору есть несколько вопросов:

1) Несмотря на относительно большой объем (36 страниц) сама методология построения сеток занимает 19 страниц. По приблизительным подсчетам на изложение материала статьи необходимо 8 академических часов, что соответствует 2 авторским листам, т.е. 32 страницы текста учебно-методического пособия по техническим дисциплинам. Т.е. сама методика построение сеток изложена очень кратко и не доступна, к примеру, студентам или аспирантам. Но что бы материал был доступен, то статью будет необходимо еще увеличивать. Я бы рекомендовал авторам детальное описание алгоритма с листингом кода и конкретными примерами привести в дополнительных материалах к статье (supplementary).

2) С точки зрения геометрии – использование криволинейных координат приводит к экономии (иногда значительной) вычислительных ресурсов. Но нет ответа на такой важный вопрос – приводит ли использование криволинейной сетки к увеличению точности результатов моделирования.

Проблема *ROMS* в том, что жидкие границы бассейна установлены только в тех узлах сетки, которые находятся только на границах области моделирования. Разумеется, в *ROMS* есть возможность задавать внутренние точки как источники/стока массы, но эта опция используется только для задания речного стока. Как полноценные жидкие границы их использовать проблематично, поскольку для таких внутренних точек в *ROMS* используется только «жесткое» граничное условие. Поэтому если акватория, где необходимо провести моделирование, имеет сложную форму берега, то использование таких криволинейных сеток действительно спасает ситуацию.

Возникает двоякая ситуация. С одной стороны – использование криволинейной сетки упрощает задачу при выборе области моделирования – область можно взять произвольную, а алгоритм сам рассчитает все параметры сетки и при этом экономятся вычислительные ресурсы. Но с другой стороны – входные метеорологические данные (ветер, атмосферное давление, потоки радиации, осадки и пр.) и океанографические данные (течения, уровень океана, температура и соленость морских вод) как правило, берутся с каких либо мировых центров прогноза погоды и океанографических реанализов, которые поставляют данные на регулярной сетке. Т.е. если мы используем криволинейную сетку, то нам приходится с регулярной сетки интерполировать все входные данные на криволинейную сетку, а потом зачастую приходится для анализа результатов моделирования делать обратную задачу – интерполировать результаты моделирования с криволинейной на регулярную сетку. Таким

образом экономия времени при моделировании может быть соразмерна затратам на обработку данных.

Все вышесказанное приводит к закономерному вопросу:

3) не проще ли просто модифицировать код самого *ROMS* и позволить пользователю устанавливать жидкие границы произвольно? Тогда можно просто использовать для расчетов регулярную сетку.

Подпись: Рецензент №2 13.10.2020.

++++++++++++++++

Ответ рецензенту №2 на Рецензию от 13.10.2020 на статью автора: Щепеткин А.Ф.: «Построение ортогональных криволинейных сеток для регионального моделирования океана: математический алгоритм и руководство пользователя»

Во-первых, хочется выразить благодарность за внимательное прочтение статьи и комментарии. Ниже в ответе исходный текст рецензии выделен цветом и зауженной шириной абзацев. Все web-ссылки выделенные голубым цветом «живые», т.е. сразу обращаются к соответствующему сайту если на них кликнуть мышкой.

Рецензент: В статье «Построение ортогональных криволинейных сеток для регионального моделирования океана: математический алгоритм и руководство пользователя» описывается алгоритм построения ортогональной сетки для моделей океана.

Достоинства этой статьи заключается в важности подобной задачи, и ее практической реализации. Важность данной работы хорошо аргументирована автором во введении, и рецензенту тут добавить нечего.

Рекомендация рецензента редакционной коллегии журнала: опубликовать данную статью.

Некоторые разделы можно рекомендовать включить в курс «Вычислительная математика» для студентов математических специальностей.

Комментарии рецензента

Тем не менее, у рецензента к автору есть несколько вопросов:

1) *Несмотря на относительно большой объем (36 страниц) сама методология построения сеток занимает 19 страниц. По приблизительным подсчетам на изложение материала статьи необходимо 8 академических часов, что соответствует 2 авторским листам, т.е. 32 страницы текста учебно-методического пособия по техническим дисциплинам. Т.е. сама методика построение сеток изложена очень кратко и не доступна, к примеру, студентам или аспирантам. Но что бы материал был доступен, то статью будет необходимо еще увеличивать. Я бы рекомендовал авторам детальное описание алгоритма с листингом кода и конкретными примерами привести в дополнительных материалах к статье (supplementary).*

Автор: Не очень понятно каком образом можно оценить число академических часов на преподавание этого материала, но на самом деле этот (или, даже, эти) вопрос(ы) о том какую глубину материала нужно включить в статью, и/или как его преподавать гораздо шире.

Во-первых, целевая аудитория: для кого писалась эта статья?

Пользователи *ROMS* (или других кодов, использующих регулярные структурированные сетки в ортогональных криволинейных координатах) которые уже имеют некоторый опыт, и несколько разочарованы простыми сетками в виде прямоугольника в повернутой проекции Меркатора, но ещё больше разочарованы отсутствием возможности построить сетку, которая была бы интуитивно более оптимальной для их конкретной задачи. Доходит и до такого: *Павел Саков* (не кто иной, как автор кода *gridgen-c* для построения ортогональных криволинейных сеток) в недавней работе

https://www.nature.com/articles/s41467-017-01595-0

довольствуется простейшей сеткой в обычных *lat-lon* координатах, даже не повернув её, чтобы сделать одну и из сторон вдоль побережья (см. их Fig. 1).

Во-вторых, с образовательной точки зрения: предполагается что любой человек конечно проходил курс ТФКП, и обязательно слышал про конформные преобразования,

голоморфные функции, условия *Коши-Римана*. Но это было некоторое время в прошлом, порой значительное, и про преобразование *Шварца-Кристоффеля* он, конечно, что-то слышал, но вот что именно это было он уже забыл. Некоторые конечно могут сделать вид: ах да, всё это можно найти в учебниках по ТФКП, это тривиально... Но это не решает проблему, потому что те теоремы, которые доказываются в этих учебниках, и те примеры, которые там рассматриваются, ещё не дают очевидного пути как сделать практический инструмент для создания сеток достаточно общей геометрии.

Во-третьих, современная культура, нацеленная на результат. Пользователя вполне устроит «черный ящик», коль скоро (1) четко оговорено что на входе и что должно получиться на выходе, и (2) достаточно надежен и даёт ожидаемый результат, не требуя никаких настроек со стороны пользователя. Например, алгоритм *Ives & Zakharias* вполне можно отнести к такому классу.

Здесь стоит отметить, что руководство пользователя (т.е. то что обычно называют *Manual*) для подавляющего большинства современных моделей океана *фактически отсутствует*. Например, то, что называется *MOM4p1 Manual*

https://www.gfdl.noaa.gov/wp-content/uploads/files/model_development/ocean/guide4p1.pdf

на деле оказывается «живой версией» книги *S. Griffies* (2004) "Fundamentals of ocean climate models постепенно переходящей в *MOM5*,

https://mom-ocean.github.io/assets/pdfs/MOM5_manual.pdf.

Эти документы можно воспринимать как учебник по численному моделированию океана с хорошим изложением истории и современного статуса научной дискуссии, выявлением острых проблем, и вариантов их решений – всё что угодно, но только не только не руководство по использованию кода написанное такт чтобы его одного было достаточно для того чтобы можно было взять код и разобраться в нем в той степени чтобы самостоятельно начать его использовать.

Ровно аналогичная ситуация и с так называемой “*NEMO Book*”,

https://www.nemo-ocean.eu/wp-content/uploads/NEMO_book.pdf .

Для многих моделей, *MICOM/NYCOM, MITgcm/ECCO, POP, MOM6, MPAS* подобных документов не было написано никогда. *ROMS Manual* написанный *Kate Hedstrom* формально существует, но, фактически представляет собой обновленный вариант *SPEM/SCRUM Manual*, который безнадежно устарел и, уже долгое время едва ли полезен для того чтобы начать использовать код. Вместо этого, многие модели сопровождаются учебным материалом на web-сайтах, который полезен, но никогда не отражает всю полноту того, что имеет место в коде. Обучение пользованию моделями в основном происходит либо через личные контакты, либо на обучающих семинарах (*workshops*).

Сказанное выше не следует понимать, как оправдание существующего положения дел — это просто констатация факта, на который есть много причин, одна из которых это низкая (ниже плинтуса) мотивировка написания такого рода технических документов по сравнению с журнальными статьями по очевидным причинам. Но это действительность, в которой мы живём. Требования к статье другие – на первый план выходит новизна, а не полнота описания.

Вместе с тем, практика показывает, что недостаточно просто разместить код на каком-нибудь доступном сайте, и ожидать его широкого использования: даже хорошо написанный и задокументированный изнутри в комментариях код просто не привлечет должного внимания - здесь конкретным примером является *gridgen-c Павла Сакова* - на его сайте

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

есть описание, примеры построенных сеток, но алгоритмическим ядром всего пакета является заимствованная подпрограмма триангуляции *Делоне, triangle.c*, которая составляет более 15 тыс. строк кода, и которая по сути используется как черный ящик (описание алгоритма *CRDT* (*cross-ratious of Delaunay triangulation*), лежащего в его основе прилагается в виде статьи третьей стороны). Естественно, разобраться во всём этом потенциальному пользователю очень сложно. Кроме того, в этом пакете отсутствует

достаточно интуитивный и удобный для пользователя интерфейс (частично это компенсировано другими разработчиками кода), что так же не способствует широкому использованию этого пакета.

Поэтому, исходя из всего вышесказанного, нами специально выбран именно такой «смешанный» формат, который кому-то может показаться странным: это именно статья в которой описываются преимущественно новые черты математического алгоритма (относительно того что использовалось ранее, и с чем читатель скорее всего знаком), но при этом делается упор на то, чтобы дать понять что это именно цельный пакет готовый к применению, и что это не так уж сложно и стоит попробовать (т.е. раскрываются детали как его практически применить).

Естественно, описать всё в равной степени детализации потребует слишком большого объёма, и приходится чем-то жертвовать. Так, мы не рассчитываем на то, что, прочитав эту статью (и только её), читатель сможет полностью самостоятельно воспроизвести этот пакет. Детали построения сплайнов, решение эллиптической задачи-всё остаётся за кадром. Маску вода-суша тоже никто не отменял, и задача построения маски из данных береговой линии GSHHS тоже является достаточно сложным и по своему красивым алгоритмом.

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

Это тоже остаётся за кадром. Алгоритм *IZ* у нас в принципе описан полностью, и даже более детально чем в оригинальной статье *Ives & Zakharias* (1989): сначала рассматриваются свойства элементарное преобразования выпрямления угла, затем показывается что можно отобразить ломаную на прямую (точнее полупространство по одну стороны от ломаной с бесконечными лучами на краях на полуплоскость), и это преобразование обратимо, затем уже вводится итеративное отображение односвязной области на прямоугольник. Но при этом неизбежно возникает вопрос: а единственно ли полученное таким образом конформное преобразование (например, если мы используем другую последовательность обхода точек – будет ли, в конце концов, то же самое распределение точек на сторонах прямоугольника? Ответ на этот вопрос есть в учебниках ТФКП: да, единственно. Мы же это не доказываем, а в лучшем случае только ссылаемся на них. Скорость сходимости итерационных процедур: стараемся сделать так чтобы это ныла сходимость метода Ньютона, т.е. ошибка уменьшается как квадрат числа итераций. Опять же, это проверялось практически, но в статье это лишь упоминается, но не доказывается.

Алгоритм *IZ* требует ~ $(N_x + N_y)^3$ вычислений, поэтому для интересующих нас сеток $N_x \sim Ny$ 1000 это десятки минут, при условии, что код распараллелен. Если подобный код сделать на *Matlab* или *Python*, то он станет медленным сам по себе (т.е. *Fortran* vs. *Python* на одном процессоре), да и ещё потеряет многопроцессорность. Это сделает его если не совсем бесполезным, то по крайней мере не очень привлекательным. Но распараллеливание алгоритма *IZ* достаточно непростая задача, потому что оригинальный код содержит зависимость вычисления операций в цикле от одной точки к другой, т.к. необходимо гарантировать непрерывность аргумента комплексного числа. Описание того, как это преодолеть могло бы вызвать интерес, но тоже остается за кадром (в статье есть лишь упоминание). На самом деле, если сделать курс лекций на тему распараллеливания на общей памяти, начать с элементарных примеров, то до алгоритма подобной сложности можно дойти за несколько часов.

Т.е., если резюмировать коротко, то такой формат изложения, при всех его компромиссах, по-видимому, оптимален.

Что касается кода, то мы его планируем сделать открытым.

Сделать из него *supplementary material* можно, но не совсем целесообразно: во-первых, построение сеток является частью большего пакета программ со сложными связями и многократным повторным использованием компонент, и «выцарапать» из него именно ту часть что чисто относится к построению сеток можно, но проблематично. А во-вторых, как всякий код, он имеет свойство обновляться, поэтому то, что окажется на сайте журнала неизбежно будет статичным, и устареет.

Рецензент:

2) *С точки зрения геометрии - использование криволинейных координат приводит к экономии (иногда значительной) вычислительных ресурсов. Но нет ответа на такой важный вопрос – приводит ли использование криволинейной сетки к увеличению точности результатов моделирования.*

Автор: Ответ на этот вопрос «да», но это выходит за рамки статьи.

Стоит условно выделить три круга вопросов:

(1) способ построения сеток, т.е. имея контур, заданный последовательностью точек (например, опорных точек по которым проводится сплайн) как построит сетку;

(2) допустим, что проблема (1) полностью решена, и становится чисто технической. Тогда возникает другая проблема: как именно провести контур чтобы получить сетку с желаемыми свойствами (например, управлять разрешением);

(3) влияние сетки на решение, т.е. допустим, что (1) решена, и, (2) мы умеем управлять тем, где и как мы хотим получить более высокое разрешение, возникает вопрос, а к чему, собственно, стремиться? Понятно, что конечная цель это получить наиболее качественное решение в *интересующей нас части области*, но, в большинстве случаев, оно зависит от того что происходит вокруг, порой даже на значительном удалении, поэтому вычислительная область существенно больше чем интересующий нас район, а покрыть всё равномерной сеткой слишком затратно. Кроме того, есть ещё требование что разрешение сетки должно медленно меняться на масштабе $1\Delta x$, то же относится к скорости поворота координатных линий. Т.е. набор противоречивых требований и компромиссов.

Данная статья полностью решает проблему (1), частично углубляется в (2) (например, упоминается что простое следование береговой линии не всегда целесообразно); и совсем не касается (3) – последнее требует накопление большого опыта вычислений и анализа получившихся решений. Единственное что можно сказать априори, это то что, в нашем пример сетки Атлантического с высоким разрешением в Мексиканском заливе, если сетку заменить на равномерную, покрывающую весь океан с разрешением как в заливе, то вычислительная стоимость такой задачи (общее количество точек) возрастет более чем на порядок.

Рецензент: Проблема ROMS в том, что жидкие границы бассейна установлены только в тех узлах сетки, которые находятся только на границах области моделирования. Разумеется, в ROMS есть возможность задавать внутренние точки как источники/стока массы, но эта опция используется только для задания речного стока. Как полноценные жидкие границы их использовать проблематично, поскольку для таких внутренних точек в ROMS используется только «жесткое» граничное условие.

Автор: Насчет стоков, конечно, нет, допускаются только источники (подразумеваются впадение рек, но может быть и экзотическая ситуация в воде трубы со сбросом примеси *из трубы в море*), и математически это понятно: граничные условия для источников жестко предписывают скорость и *T*, S-свойства приходящей воды. Для уходящей воды нужны другие граничные условия, и такая постановка в *ROMS* никогда не ставилась (кроме как на краю сетки).

Рецензент: Поэтому если акватория, где необходимо провести моделирование, имеет сложную форму берега, то использование таких криволинейных сеток действительно спасает ситуацию. Возникает двоякая ситуация. С одной стороны – использование криволинейной сетки упрощает задачу при выборе области моделирования – область можно взять произвольную, а алгоритм сам рассчитает все параметры сетки и при этом экономятся вычислительные ресурсы.

Автор: На самом деле произвол здесь ограничен: форма внешнего периметра ортогональной криволинейной сетки не всегда следует береговой линии (возможно срезая острые мысы чтобы избежать резких изгибов). Если попытаться это сделать, то будут нежелательные последствия, сгущение сетки там, где это не нужно и даже вредно из-за ограничений на число *Куранта*. Неструктурированные сетки дают гораздо большую свободу в этом смысле, но также имеют свои недостатки: трудно добиться порядка точности выше второго, коды получаются существенно медленнее, если сравнивать

вычислительные затраты на одну степень свободы.

Рецензент: *Но с другой стороны – входные метеорологические данные (ветер, атмосферное давление, потоки радиации, осадки и пр.) и океанографические данные (течения, уровень океана, температура и соленость морских вод) как правило, берутся с каких-либо мировых центров прогноза погоды и океанографических реанализов, которые поставляют данные на регулярной сетке. Т.е. если мы используем криволинейную сетку, то нам приходится с регулярной сетки интерполировать все входные данные на криволинейную сетку, а потом зачастую приходится для анализа результатов моделирования делать обратную задачу – интерполировать результаты моделирования с криволинейной на регулярную сетку. Таким образом экономия времени при моделировании может быть соразмерна затратам на обработку данных.*

Автор: Ответ на этот вопрос четкое «нет».

Процедуры интерполяции данных с одной криволинейной сетки на другую достаточно хорошо отработаны в *ROMS*, причём с достойным качеством: двумерная бикубическая эрмитова сплайн интерполяция, если речь идет о горизонтальных направлениях.

Причем это применимо к сеткам, которые повернуты на произвольные углы относительно друг друга (т.е. для случая, когда задачу принципиально нельзя свести к повторной одномерной интерполяции, сначала вдоль оси у потом вдоль у, или наоборот).

Более того, это применимо как к скалярным величинам, например T, S так и к векторам, u, v - в последнем случае происходит поворот компонент, и, поскольку компоненты вектора расположены в разных местах из-за использования смещённых сеток (e.g., *Arakawa C-grid*), это тоже приходится учитывать и учтено (оба компонента вектора (u, v) расположенные в своих точках на материнской сетке напрямую интерполируются и на точки u, и на v в дочерней сетки, после чего происходит поворот на необходимый угол – пересчет компонент).

Наконец, всё это проходится проделывать в присутствии маски земля-суша, т.е. данные существуют не во всех точках исходной сетки, а кое-где «заземлены». Чтобы сделать интерполяцию необходимо заполнить эти точки данными. Маски материнской и дочерней сеток не всегда согласованы друг с другом, и фактически, речь идет об экстраполяции имеющихся «живых» данных в ближе в сторону береговой линии.

Проблема не нова: много лет назад пришлось столкнуться с необходимостью отбраковки данных *QuikSCAT* спутникового ветра. Если пиксель частично касается земли, то данные оказываются слишком неточными и пиксель приходится выбрасывать целиком, заменяя его значения на экстраполированные из ближайших точек. Фанатики *Matlab* не придумали ничего другого как использовать *nearest-neighbor interpolation* для этой цели. Просто, потому что эта процедура готовая к применению.

В современном варианте (с 2017 г.) применяется специально разработанный алгоритм “*etchhing*” т.е. травление. Допустим у нас имеется двумерная пластинка из металла очень сложной формы, и её растворяют в кислоте, т.е. металл постепенно исчезает, и происходит заполнение освобождаемой площади жидкостью, которая приносит с собой свои свойства. Естественно, если есть узкие длинные проходы (например, фьорды), то жидкость проникает прежде всего через них, и соответственно, заполнение значений в точках, изначально закрытых маской происходит преимущественно из тех ячеек, которые ближе всего достигаются *по воде* (в противоположность просто геометрически ближайшим соседям, как в *Matlab nearest-neighbor*).

При этом попутно происходит *слабая экстраполяция*, т.е. если значения в паре ближайших «живых» точках указывают что есть градиент, то это учитывается, но как как если бы производная в точке куда производится экстраполяция обращается в ноль (в простейшем одномерном случае, если есть две живые точки по одну сторону, это эквивалентно тому что строится парабола которая проходит через эти две точки, а её производная обращается в ноль в точке где вычисляется интерполированное значение; если же, наоборот, имеется ситуация просто одной изолированной точки окруженной живыми со всех сторон, то процедура эквивалентна би-кубической интерполяции восстановления потерянного значения в которой участвуют все точки вокруг.

Коды для всех этих алгоритмов в принципе доступны, например
https://www.myroms.org/forum/viewtopic.php?f=23&t=5050
где внизу нужно найти ссылку на источник tools.tar, и после распаковки, изучить файлы *r2r_init.F, r2r_interp_init.F, r2r_subs.F, etch_into_land.F*. [Содержание дискуссии на этой web-странице к вопросу интерполяции между регулярными сетками отношения не имеет.]

...и, в соответствие с прочно установившимися традициями, интерполяционные алгоритмы нигде не опубликованы, так как научного интереса они не представляют.

Рецензент: Все вышесказанное приводит к закономерному вопросу:

3) *не проще ли просто модифицировать код самого ROMS и позволить пользователю устанавливать жидкие границы произвольно? Тогда можно просто использовать для расчетов регулярную сетку.*

Автор: Нет, не проще. Проблема интерполяции решена выше, а сама идея отказа от криволинейных сеток возвращает нас к избыточному использованию земляной маски.

А, кроме того, реанализы для океана на самом деле считаются на криволинейных сетках с целью избежать проблемы полюсов: северный полюс смещается вглубь Евразийского континента, либо расщепляется на два, один из которых оказывается в Евразии, другой в Северной Америке. Получившаяся при этом сетка не имеет сингулярности в акватории северного Ледовитого океана (да и вообще нигде, где есть вода).

https://nomads.gfdl.noaa.gov/CM2.X/oceangrid.html
https://nomads.gfdl.noaa.gov/CM2.X/tripolargrid.html
https://nomads.gfdl.noaa.gov/CM2.X/tripolargrid.html

Геофизические поля на регулярной *lat-lon* сетке (доступные на соответствующих *web*-сайтах) являются вторичными продуктами, переинтерполированными с исходных сеток. Кроме того, на этом этапе часто подавляют смещённые сетки: *u, v, T, S* - все оказываются в одних и тех же точках. Это, с одной стороны, удобно для конечного пользователя, но оказывается, что вычислить некоторые производные поля (например, завихренность) из этих данных на самом деле труднее, а результат получается заметно хуже.

С уважением, Автор. 15.10.2020.

++++++++++++++++++

Рекомендация Рецензента №2 редакционной коллегии журнала: опубликовать данную статью.

Подтверждение Рецензента №2, Подпись, 15.10.2020