УДК 624.04

ВЕРИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНОГО ВЫЧИСЛИТЕЛЬНОГО МОДЕЛИРОВАНИЯ

В.В. Надольский
Брестский государственный технический университет, г. Брест, Республика Беларусь
e-mail: Nadolski_V@mail.ru

В компьютерном моделировании основными алгоритмами решения являются метод конечных элементов и метод конечных разностей. Хотя теоретический порядок аппроксимации этих алгоритмов может быть известен из разложения степенных рядов дискретных уравнений, однако наблюдаемый (фактически реализуемый) порядок аппроксимации может отличаться. Многие факторы могут ухудшать наблюдаемый порядок аппроксимации по отношению к теоретическому, который характерен для математической особенности алгоритма. Эти факторы включают ошибки программирования, недостаточную регулярность сетки, сингулярности, разрывы, неадекватную итерационную сходимость, чрезмерно заданные граничные условия и т.д. При верификации все эти причины снижения порядка аппроксимации должны быть исследованы. Таким образом, важной частью верификации программного обеспечения является определение наблюдаемого порядка аппроксимации алгоритма решения. Верификация программного обеспечения компьютерного моделирования – процесс исключения ошибок и минимизации погрешностей в алгоритме численного решения и его компьютерном коде при любых обстоятельствах, при которых будет применяться программное обеспечение.

Ключевые слова: *верификация, валидация, точность, ошибка, неопределенность, компьютерная численная модель.*

VERIFICATION OF SOFTWARE FOR COMPUTER NUMERICAL MODELING

V. Nadolski
Brest State Technical University, Brest, Republic of Belarus
e-mail: Nadolski_V@mail.ru

In computer modeling, the main solution algorithms are the finite element method and the finite difference method. Although the theoretical approximation order of these algorithms may be known from the decomposition of power series of discrete equations, however, the observed (actually implemented) approximation order may differ. Many factors can worsen the observed approximation order in relation to the theoretical one, which is characteristic of the mathematical feature of the algorithm. These factors include programming errors, insufficient grid regularity, singularities, discontinuities, inadequate iterative convergence, overly specified boundary conditions, etc. During verification, all these reasons for the decrease in the approximation order should be investigated. Thus, an important part of software verification is to determine the observed order of approximation of the solution algorithm. Verification of computer modeling software is the process of eliminating errors and minimizing errors in the numerical solution algorithm and its computer code under any circumstances in which the software will be used.

Keywords: *verification, validation, accuracy, error, uncertainty, computer numerical model.*

Введение. Первое, с чего хочется начать – это с определения понятия «точность компьютерной вычислительной модели». Точность модели подразумевает отсутствие ошибок, уменьшение и контроль погрешности и неопределённости результатов моделирования [1]. Ошибка – это признанный недостаток моделирования, который не связан с недостатком знаний или с какими-то преднамеренными упрощениями. При проверке модели необходимо исключить появление ошибок. Неопределённость – это потенциальный недостаток моделирования, главным образом связанный с недостатком знаний и/или с природной изменчивостью случайных величин или процессов. При проверке модели задача состоит в уменьшении и оценки неопределённостей с последующим учётом этих неопределенностей при обеспечении надежности [2]. Погрешность – это признанный недостаток моделирования связанным с преднамеренными упрощениями. В процессе проверки необходимо уменьшить погрешность и определить её численно и, в конечном итоге, учесть в обеспечении надёжности. Проверка компьютерной модели включает верификацию и валидацию [1; 3]. Одним из этапов верификации является верификация программного обеспечения. *Верификация программного обеспечения компьютерного моделирования – процесс исключения ошибок и минимизации погрешностей в алгоритме численного решения и его компьютерном коде при любых обстоятельствах, при которых будет применяться программное обеспечение.*

Деятельность по верификации программного обеспечения фокусируется на математической корректности и конкретных реализациях дискретных алгоритмов решения в программном коде. Целью верификации программного обеспечения является подтверждение того, что алгоритмы численного решения правильно реализованы (запрограммированы) в программном коде и что эти алгоритмы функционируют так, как задумано. Верификация программного обеспечения основывается на тщательном изучении таких тем, как: скорости пространственной и временной сходимости, скорость итерационной сходимости, независимость численного решения от преобразования координат и надлежащее сохранение симметрии, связанное с различными типами начальных и граничных условий и т.д.

В компьютерном моделировании основными алгоритмами решения являются метод конечных элементов и метод конечных разностей. Хотя теоретический (формальный) порядок аппроксимации этих алгоритмов может быть известен из разложения степенных рядов дискретных уравнений, однако наблюдаемый (фактически реализуемый) порядок аппроксимации может отличаться. В численном анализе скорость сходимости численной аппроксимации дифференциального уравнения к точному решению количественно определяет порядок аппроксимации. Наблюдаемый (фактически реализуемый) порядок аппроксимации является скоростью, с которой решение асимптотически приближается к точному решению по мере уточнения дискретизации. Многие факторы могут ухудшать наблюдаемый порядок аппроксимации по отношению к теоретическому (формальному) порядку аппроксимации, который характерен для математической особенности алгоритма. Эти факторы включают ошибки программирования, недостаточную регулярность сетки, сингулярности, разрывы, неадекватную итерационную сходимость, чрезмерно заданные граничные условия и т.д [4–6]. При верификации все эти причины снижения порядка аппроксимации должны быть исследованы. Таким образом, важной частью верификации программного обеспечения является определение наблюдаемого порядка аппроксимации алгоритма решения. Это может быть сделано путем сравнения двух или более результатов вычислений с различной дискретизацией и определением скорости сходимости (конвергенции).

Основными задачами при верификации программного обеспечения являются определение соответствующих тестовых задач для оценки точности численных алгоритмов и оценки функционирования этих алгоритмов в тестовых задачах. Верификация программного обеспе-

чения выполняется сравнением компьютерных решений с “тестовым решением”, которое обеспечивается аналитическими решениями или численными эталонными решениями для набора хорошо подобранных тестовых задач.

Учитывая нехватку надежных тестовых задач для сложных математических моделей, необходимо отметить два момента. Первый момент заключается в том, что некоторые тестовые задачи более подходят, чем другие, поэтому следует использовать тестовые задачи, относящиеся к рассматриваемой области применения. Эти тестовые задачи могут быть теми, для которых накоплен большой опыт, или они могут быть созданы для удовлетворения конкретных потребностей, возникающих при планировании верификации.

Второй момент заключается в том, что некоторые решения лучше других и, следовательно, следует признать иерархию достоверности тестовых задач. Существует следующая иерархия достоверности (от наивысшей к наименьшей) для тестирования алгоритмов:

(a) точные аналитические решения и предопределенные решения;

(b) полуаналитические решения;

(c) численные эталонные решения.

Далее представлена информация о видах тестовых задач и их решениях, используемых для верификации программного обеспечения.

Точные аналитические и полуаналитические решения. Аналитические решения “физически правдоподобной” задачи - это решения дифференциального уравнения в частных производных математической модели с начальными и граничными условиями, которые могут быть реально реализованы. Эти решения иногда являются точными (требующими только арифметических вычислений явных математических выражений), но часто являются полуаналитическими (представлены бесконечными рядами, комплексными интегралами, или асимптотическими разложениями). Трудности могут возникнуть при вычислении любого из этих полуаналитических решений, особенно представленных бесконечными рядами. При верификации требуется убедиться, что численная погрешность полуаналитических решений снижена до приемлемого уровня.

Обычно для задач, которые допускают аналитические решения, будь то точные или полуаналитические, критерии сравнения (т.е. критерии удовлетворительности или неудовлетворительности решением программного обеспечения) могут быть сформулированы в следующих терминах:

(a) соответствие между наблюдаемым порядком аппроксимации и теоретическим (формальным) порядком аппроксимации численного метода;

(b) соответствие между конвергентным численным решением и аналитическим решением с учетом определённых норм ошибок.

Компьютерные (вычислительные) решения сравниваются с аналитическими решениями либо в интересующих областях, либо по всей области решения с учетом определённых норм ошибок. Точность каждой из переменных или зависимости, представляющей интерес, должна быть определена как часть сравнения.

Предопределенные решения. Точные аналитические решения для физически значимой задачи известны только в относительно небольшом количестве простых случаев, в которых обычно используется и можно верифицировать лишь ограниченную часть программного кода. К счастью, метод предопределенных решений предлагает методику получения математически точного решения тесно связанной задачи, чтобы использовать все аспекты программного кода, которые могли бы быть активированы интересующими физическими явлениями. Метод предопределенных решений предоставляет общую процедуру генерации аналитического решения для верификации программного обеспечения.

Аналитические решения с предопределенным решением - это методика разработки особого типа аналитического решения. Чтобы применить его, инженер предопределяет (предписывает) функции решения для дифференциальных уравнений в частных производных и находит функции воздействия, которые согласуются с предопределенным решением. То есть предписанные функции решения вставляются в дифференциальные уравнения в частных производных, и уравнения перестраиваются (преобразуются) таким образом, что все оставшиеся члены, превышающие члены в исходных дифференциальных уравнениях, группируются в функции воздействия или исходные члены. Начальные условия и граничные условия получают аналогичным образом на основании предписанного решения для граничных условий. Для примера, для задачи с шарнирно-опертой пластинкой одно из предписанных решений можно составить для перемещений, которые требуют сосредоточенной нагрузки, или даже моменты. Если эта “функция воздействия” сосредоточенной нагрузки и момента может быть получена, то затем ее можно применить, используя компьютерную вычислительную модель пластины, и вычисленное поле перемещений можно сравнить с предписанным решением.

Преимуществ метода предопределенных решений много. Его можно применять к широкому спектру задач. Он может тестировать большое количество численных характеристик в коде, таких как численный метод, метод пространственного преобразования для генерации сетки, корректность кодирования алгоритма и т.д. Метод предопределенных решений обеспечивает четкую оценку, поскольку, если нет программных ошибок, результаты вычислений должны совпадать с решением, используемым для получения функции воздействия. Однако метод предопределенных решений не лишен недостатков. При любом нетривиальном применении этого метода алгебра и математический анализ, необходимые для получения функции воздействия, могут стать очень сложными. Метод предопределенных решений может эффективно выявлять наличие ошибок, однако он не может указывать на источники этих ошибок и не может идентифицировать ошибки в эффективности алгоритма.

Численное эталонные решения. Когда аналитические решения не могут быть найдены, единственной опцией остается численное эталонное решение. Существует две категории численных эталонных решений. Одна категория состоит из решений, в которых дифференциальное уравнение в частных производных было сокращено преобразованиями подобия или другими средствами до одного или нескольких обыкновенных дифференциальных уравнений, которые могут быть проинтегрированы численно. Для численного интегрирования обыкновенных дифференциальных уравнений доступны хорошо зарекомендовавшие себя стандартные методы оценки порядка аппроксимации. Другая категория состоит из решений, в которых дифференциальные уравнения в частных производных были решены непосредственно численными методами. Точность таких численных эталонных решений должна быть критически оценена, чтобы их можно было использовать при верификации программного обеспечения. В случае численного интегрирования дифференциального уравнения в частных производных ни одно решение не может считаться эталоном до тех пор, пока код, используемый при создании этого решения, не был тщательно проверен и документально подтвержден. Кроме того, должна быть представлена оценка ошибки численного интегрирования. Достоверность повысится, если независимые исследователи, предпочтительно используя отличающиеся численные подходы и программное обеспечение, произведут несколько решений. Использование нескольких независимых источников снизит риск ошибок тестовых решений.

Заключение. В статье представлено описание процедуры верификации программного обеспечения, которая направлена на проверку математической корректности и точности конкретных реализаций дискретных алгоритмов, т.е. того, что программная реализация (исполнение) численных алгоритмов не содержит ошибок программирования и ошибок реализации.

Однако, кроме верификации численной точности алгоритмов, заложенных в программном комплексе на этапе его разработки, также важно поддержание качества программного обеспечения, т.е. поддержание точности и стабильности решения независимо от компьютерного оборудования и программной среды (компиляторы, библиотеки). Поддержание качества программного обеспечения затрагивает такие вопросы, как контроль версий, конфигураций, архитектуры программного кода, и направлено на исключение ошибок в процессе выпуска и установки программного обеспечения.

ЛИТЕРАТУРА

1. Надольский, В. В. Верификация и валидация компьютерной вычислительной модели для проектирования строительных конструкций / В. В. Надольский // Вестник Полоцкого государственного университета. Серия F. Строительство. Прикладные науки. – 2024. – № 2 (37). – С. 42-50. – DOI: 10.52928/2070-1683-2024-37-2-42-50.
2. Тур, В. В. Концепция проектирования строительных конструкций на основе численных моделей сопротивления / В.В. Тур, В.В. Надольский // Строительство и реконструкция. – 2022. – №6 (104) – С.78-90. DOI: 10.33979/2073-7416-2022-104-6-78-90
3. Надольский, В. В. Оценка расчетного значения несущей способности стальных элементов, проектируемых на основе численных моделей /В. В. Надольский // Вестник МГСУ. – 2023. – Т. 18. – Вып. 3. – С. 367–378. – DOI: 10.22227/1997-0935.2023.3.367-378
4. Kövesdi, B. Finite Element Model-based Design of Stiffened Welded Plated Structures Subjected to Combined Loading // Periodica Polytechnica Civil Engineering. – 2021. – Vol. 65(2). – pp. 677–689. – DOI: 10.3311/PPci.17229.
5. Надольский, В. В. Оценка несущей способности стальной балки методом конечных элементов при совместном действии локальных и сдвиговых усилий / В. В. Надольский, В.И. Подымако // Строительство и реконструкция. – 2022. – №2 (100) – С. 26-43. – DOI: 10.33979/2073-7416-2022-100-2-26-43
6. Надольский, В. В. Оценка несущей способности балок с гофрированной стенкой методом конечных элементов при действии локальной нагрузки / В. В. Надольский, А.И. Вихляев // Вестник МГСУ. – 2022. – Т. 17. – Вып. 6. – С. 693–706. – DOI: 10.22227/1997-0935.202.