DOI: 10.17725/rensit.2022.14.331

Вычислительный эксперимент – обезразмеривание уравнений, вычислительная устойчивость и тестирование программ

[1]Евтихов М.Г., [2]Евтихов В.Г.

[1]Институт радиотехники и электроники им. В.А.Котельникова РАН, Фрязинский филиал, http://fire.relarn.ru/
г. Фрязино 141190, Моск.обл, Российская Федерация
[2]Московский политехнический университет, https://mospolytech.ru/
Москва 107023, Российская Федерация
E-mail: emg20022002@mail.ru, evg2002@yandex.ru

Поступила 17.06.2022, рецензирована 24.06.2022, принята 01.07.2022

Представлена действительным членом РАЕН В.В. Колесовым

Аннотация: **Из этапов вычислительного эксперимента рассмотрен этап обезразмеривания исходного уравнения (системы уравнений) задачи – замена его переменных произведением соответствующих безразмерных величин на единицы их измерения с последующими преобразованиями. Такой переход от физической модели к математической (обезразмеренной) позволяет получить реализации программного обеспечения для проведения исследований. Проводится критическая оценка его сложности и оцениваются возможные ошибки результатов. При этом формируются новые версии программных средств. Средства объектно ориентированного программирования и системы управления версиями (например, git) позволяют формировать версии программных средств, адаптированных под различные условия их применения и для различных типов пользователей. Производится распараллеливание работы над версиями. При этом для дальнейшей программной реализации эффективен теоретико-множественный язык формул с частично-рекурсивными функциями. Для реализации версий с большими объемами вычислений и данных используются системы высокопроизводительных вычислений, основанные на программно-аппаратной акселерации, параллельной обработке информации и облачных архитектурах. Как правило, для версии программы строится разностная модель задачи и итерационные методы ее решения. Условия вычислительной устойчивости обычно оговариваются в современных инструкциях к стандартным библиотекам программ. Для новых алгоритмов необходим анализ устойчивости разностных схем на основе уточнения их спектральных свойств и использования методов функционального анализа. Для хранения и последующего применения результатов вычислительных экспериментов целесообразно использовать современные базы данных. Как разновидность вычислительного эксперимента рассмотрено также тестирование, альфа- и бета-версии программ, их релизов.**

Ключевые слова: **вычислительный эксперимент, обезразмеривание, частично-рекурсивная функция, вычислительная устойчивость, тестирование версий программ**
УДК 167.2, 004.004, 004.007

Благодарности: Авторы благодарны Е.П. Новичихину за внимание к настоящей работе.

Для цитирования: Евтихов М.Г., Евтихов В.Г. Вычислительный эксперимент – обезразмеривание уравнений, вычислительная устойчивость и тестирование программ. *РЭНСИТ: Радиоэлектроника. Наносистемы. Информационные технологии,* 2022, 14(3):331-340. DOI: 10.17725/rensit.2022.14.331.

Computational experiment – nondimensionalization of equations, computational stability and program testing

Mikhail G. Evtikhov
Kotelnikov Institute of Radioengineering and Electronics of RAS, Fryazinsky branch, http://fire.relarn.ru/
Fryazino 141190, Moscow region, Russian Federation
E-mail: emg20022002@mail.ru
Vladimir G. Evtikhov
Moscow Polytechnic University, https://mospolytech.ru/
Moscow 107023, Russian Federation
E-mail: evg2002@yandex.ru
Received 17 June 2022, peer-reviewed 24 Juni 2022, accepted 01 July 2022

Abstract: **From the stages of the computational experiment, the stage of non-dimensionalization of the initial equation (system of equations) of the problem is considered - the replacement of its variables by the product of the corresponding dimensionless quantities by their units of measurement with subsequent transformations. Such a transition from a physical model to a mathematical (dimensionless) one makes it possible to obtain software implementations for research. A critical evaluation of its complexity is carried out and possible errors in the results are evaluated. At the same time, new versions of software are formed. Object-oriented programming tools and version control systems (for example, git) allow you to create versions of software tools adapted to different conditions of their use and for different types of users. Parallelization of work on versions is carried out. At the same time, for further software implementation, the set-theoretic language of formulas with partially recursive functions is effective. To implement versions with large amounts of calculations and data, high-performance computing systems based on software and hardware acceleration, parallel information processing and cloud architectures are used. As a rule, a difference model of the problem and iterative methods for solving it are constructed for a program version. Computational stability conditions are usually stipulated in modern instructions for standard program libraries. For new algorithms, it is necessary to analyze the stability of difference schemes based on the refinement of their spectral properties and the use of functional analysis methods. For storage and subsequent application of the results of computational experiments, it is advisable to use modern databases. As a kind of computational experiment, testing of alpha and beta versions of programs and their releases is also considered.**

Keywords: **computational experiment, problem statement, nondimensionalization, partial recursive function, numerical stability, computer literacy, tester, software development, software versioning**

UDC 167.2, 004.004, 004.007

For citation: Mikhail G. Evtikhov, Vladimir G. Evtikhov. Computational experiment – nondimensionalization of equations, computational stability and program testing. *RENSIT: Radioelectronics. Nanosystems. Information technologies,* 2022, 14(3):331-340. DOI: 10.17725/rensit.2022.14.331.

Содержание

1. **Введение (333)**
2. **Этапы вычислительного эксперимента (334)**
   - **2.1. Постановка задачи (334)**
   - **2.2. Обезразмеривание уравнений (334)**
   - **2.3.Программирование модели (335)**
   - **2.4. Решение задачи (335)**
   - **2.5. Оценка решения задачи. Циклический возврат на уровень мысленных экспериментов (336)**
   - **2.6. Возврат в программирование модели (336)**
   - **2.7. Возврат на уровень натурных экспериментов (336)**
   - **2.8. Уровни моделей (336)**
3. **Вычислительная неустойчивость (336)**
4. **Версии программных продуктов (338)**
5. **Заключение (339)**

Литература (339)

1. ВВЕДЕНИЕ

Вычислительный эксперимент – эксперимент, проводимый не на исходном реальном объекте, а на математической (информационной, имитационной) модели объекта с использованием вычислительных и логических процедур, реализуемых соответствующими программными средствами на вычислительных комплексах. Близкими к вычислительному эксперименту понятиями являются численный, математический и имитационный эксперименты на ЭВМ [1].

В [2] рассмотрены различные подходы к формированию математических моделей, основанные на научных законах и принципах. Эти подходы можно рассматривать как способы постановок вычислительных экспериментов.

Особый взгляд на вычислительный эксперимент, тесно связанный с теорией вероятностей, развивается, например, в рамках направления "планирование экспериментов" [3]. При проектировании новых типов компьютеров в 70-е годы анализировались возможности заменять некоторые физические части вычислительной системы ее программными моделями [4]. В настоящее время этот эффективный подход проник в методы программирования в форме разделения интерфейсов и их реализаций (например *h*-файлов и *cpp*-файлов в языке C++).

Развитие методов вычислительного эксперимента и его центральной части – математического моделирования, сопровождалось прояснением ряда фундаментальных особенностей вычислительных методов. В [5] высказана глубокая мысль, что программы для компьютеров с математической точки зрения описываются частично-рекурсивными функциями. При строгих определениях алгоритмов подобное утверждение рассматривается как гипотеза и носит название "тезис Чёрча". С точки зрения прикладного программиста речь идет о более широко понимаемых математических конструкциях, чем привычные для многих исследователей формулы на основе радикалов. Например, численно решить уравнение 10 степени – стандартная задача, а общего решения в радикалах для уравнения уже 5-й степени не существует. Это не означает, что можно обойтись без формул. В формулах математики частично рекурсивных функций шире используется язык теории множеств и такие конструкции как рекурсия, подстановка, и перебор элементов множеств. Например, в [6] используется описание фракталов с помощью L-системы, т.е. рекуррентного выражения для множеств: если $Z^{(0)}$ – единичный черный квадрат на комплексной плоскости, то

$$Z^{(n+1)} = Z^{(n)} \bigcup \left(iZ^{(n)} + (1+i)2^n \right) \bigcup \left(-iZ^{(n)} + (1+i)2^n \right),$$

где $\bigcup$ – операция объединения множеств.

Такого рода обобщения формул адекватны предмету, когда необходимо точно и кратко записать результат уже полученного и исследованного решения задачи. Однако в таких математических конструкциях затруднено тестирование, поиск ошибок, внесение изменений. Соответственно, при разработке программ отладка занимает более 90% времени работы над новой программой и продолжается на протяжении всего ее жизненного цикла.

При записи уже полученных теоретических решений задач язык формул с рекурсивными функциями эффективен для их дальнейшей программной реализации. Этот подход не только облегчает программирование, но и позволяет привлекать профессиональных математиков и конструкторов к решению сложных проблем оптимизации алгоритмов и хранения информации.

Существенная трудность вычислительного эксперимента – вычислительная неустойчивость [7]. В инструкциях к стандартным библиотекам программ (классов) обычно оговариваются условия вычислительной устойчивости и другие исследованные условия корректного использования программных средств. Использование библиотек для разреженных матриц общего вида [8] расширяет возможности разработки новых численных методов.

Существуют развитые программные средства и библиотеки программ для решения самых различных задач моделирования и

вычислительного эксперимента. Эти средства глубоко продуманы и исследованы. Казалось бы, есть все возможности далее развивать физику, другие науки и технологии, но реальная ситуация далеко не радужная. Методика разработки программ значительно усложнилась, языки программирования, а также сами программисты, стали специализированными.

В то же время, запас достижений в области функционального анализа еще далеко не исчерпан. Солидный задел в теории вероятностей [9,10] является основой многих успешных применений вычислительного эксперимента. Очевидно, программирование не может развиваться без экспериментальных исследований также и в области самого программирования.

В последние годы стал неизбежен outsourcing (привлечение внешних специалистов) в области тестирования программных продуктов. Появилась и распространяется новая профессия программистов – тестировщики. Работу тестировщика можно интерпретировать как экспериментальное исследование программного продукта, его деятельность предопределяет качество и конкурентоспособность программных продуктов. Несколько лет назад, от тестировщика еще не требовалось ни специальных навыков, ни образования. В настоящее время уже требуется специальная подготовка. Моделирование и вычислительный эксперимент становятся актуальными на всех уровнях современного программирования. Интересные и практически ценные идеи, связанные с тестированием, обсуждаются во многих публикациях в Интернете. Общий взгляд, "пирамида тестов", дается, например, в [11].

Цель данной работы – на примере простой задачи раскрыть понятие вычислительного эксперимента, учитывающее особенности точки зрения прикладных программистов, роль обезразмеривания уравнений, вычислительной устойчивости численных методов и развития программ версиями.

2. ЭТАПЫ ВЫЧИСЛИТЕЛЬНОГО ЭКСПЕРИМЕНТА

Представляется уместным изложить некоторые актуальные проблемы вычислительного эксперимента на примере решения относительно простой задачи.

2.1. Постановка задачи

Например: на крутом берегу моря пушка стреляет под заданным углом к горизонту. Как далеко улетит снаряд, если можно пренебречь сопротивлением воздуха и считать Землю плоской.

Полет снаряда должен удовлетворять второму закону Ньютона:

$$m\frac{d^2y}{dt^2} = -mg; \quad m\frac{d^2x}{dt^2} = 0,$$

m – масса снаряда, x – горизонтальная координата снаряда в полете, y – вертикальная координата, t – время, g – ускорение силы тяжести вблизи поверхности Земли.

Заметим, что массу можно сократить. Поэтому решение окажется независимым от массы снаряда. Если бы в уравнениях присутствовали еще и дополнительные члены, описывающие сопротивление воздуха, то они поделились бы на массу. Чем больше масса, тем их влияние было бы меньше. Мы временно пренебрегаем сопротивлением воздуха, поэтому уравнения, в целом, разумные, хотя и приспособленные для предельного случая массивных и малых по размеру тел:

$$\frac{d^2y}{dt^2} = -g; \Rightarrow \frac{dy}{dt} = -gt + C_1 = -gt +$$

$$+V_0 \sin\alpha; \Rightarrow y = -\frac{gt}{2} + tV_0 \sin\alpha + h,$$

$$\frac{d^2x}{dt^2} = 0; \Rightarrow \frac{dx}{dt} = C_2 = V_0 \cos\alpha; x = tV_0 \cos\alpha,$$

V_0 – скорость снаряда в момент его вылета из пушки, α – угол, под которым стреляет пушка, h – высота, на которой стоит пушка.

2.2. Обезразмеривание уравнений

Обезразмеривание уравнений – тонкий, не всем очевидный и, возможно, самый главный этап исследования. Он иногда недооценивается, что ведет к серьезным и даже катастрофическим последствиям. Этот этап требует консультаций

у специалистов, занимающихся натурными экспериментами.

Законы естественных наук обычно записывают в системе единиц измерений СИ. Преимуществом системы единиц измерения СИ является ее привязка к масштабам обычного житейского опыта.

Запишем физические переменные как произведение безразмерной величины (пометим ее волной) и единиц ее измерения (в квадратных скобках):

$$x = \tilde{x}[\mathrm{m}];\ y = \tilde{y}[\mathrm{m}];\ h = \tilde{h}[\mathrm{m}];\ t = \tilde{t}[\mathrm{s}].$$

Здесь все длины измеряются в метрах, время – в секундах,

$$g = \tilde{g}[m \cdot s^{-2}],\ \tilde{g} = 9{,}8;$$

g – ускорение силы тяжести вблизи поверхности Земли,

$$V_0 = \tilde{V}_0[m \cdot s^{-1}]$$

скорость измеряется в метрах в секунду.

В стандартных функциях библиотек подразумевается, что углы заданы в радианах, т.е. α уже является безразмерной величиной. Если есть необходимость задавать угол в градусах, то следует обезразмерить угол, заданный в градусах, с помощью соотношения $\alpha = \frac{\pi\tilde{\alpha}[\mathrm{deg}]}{180[\mathrm{deg}]}$, π – отношение длины окружности к диаметру, $\tilde{\alpha}$ – угол в градусах.

Процедура обезразмеривания уравнений состоит в том, что переменные вместе с их размерностями подставляются в исходные уравнения.

Оказывается, что размерности можно сокращать. Более того, все размерности должны сократиться. После обезразмеривания в уравнениях должны остаться исключительно безразмерные переменные:

$$\tilde{y} = -\frac{\tilde{g}\tilde{t}^2}{2} + \tilde{t}\tilde{V}_0 \sin\alpha + \tilde{h},$$

$$\tilde{x} = \tilde{t}\tilde{V}_0 \cos\alpha.$$

Вид обезразмеренного уравнения не обязательно должен в точности повторять вид исходного уравнения. Если окажется необходимым задавать, например, скорость не в метрах в секунду, а в километрах в час, то в обезразмеренных уравнениях появятся дополнительные множители. Иногда вводят специально для данной задачи искусственные единицы измерения.

При анализе сложных моделей возникают ситуации, когда размерности величин не сокращаются. Это однозначно говорит о том, что в уравнениях есть либо ошибка, либо какое-то серьезное недопонимание задачи. Часто с помощью обезразмеривания удается выявить несостоятельные претензии на новаторство в теориях.

2.3. Программирование модели

Процедура обезразмеривания есть переход от физической модели к математической. В памяти компьютера нет ни граммов, ни метров, ни рублей. Стандартные программы помогают перевести безразмерные величины модели в биты и байты компьютера. Используя математическую, обезразмеренную модель, программисты только и могут написать программы. В случае задачи о пушке, следует написать функции, вычисляющие $\tilde{x}$ и $\tilde{y}$ для заданного $\tilde{t}$. Хорошей манерой было бы оформить модель как класс и разместить ускорение $\tilde{g}$, высоту и угол в полях класса, а вычисление $\tilde{x}$ и $\tilde{y}$ оформить как методы класса.

Математическая модель полезна именно для той задачи, для которой она разработана. Ее расширенные интерпретации возможны, но сомнительны. Упрощенная модель может быть инструментом для решения более сложных задач и источником тестовых примеров для других усложненных моделей. Начинать следует с предельно упрощенной модели, а далее развивать версии программ. Идея Ньютона о предельно простых экспериментах особо актуальна в численном эксперименте и опирается на возможности применения наследуемых классов.

2.4. Решение задачи

Время полета снаряда есть решение уравнения $\tilde{y} = 0$. Метод класса, предназначенный для получения этого времени, можно реализовать полностью и самому, но лучше использовать стандартные программы. Дело не в том, легко или сложно решить квадратное уравнение. Важно осознать, что стандартными численными методами и программами можно численно решить даже такие уравнения, для которых нет решений в виде привычных

формул с радикалами. После вызова методов библиотечных классов, в нашем случае, следует предусмотреть отбор решения, в котором $\tilde{t} > 0$. Получив время полета снаряда, получим решение задачи с помощью метода класса для вычисления $\tilde{x}$. Остается оформить все решение задачи, как еще один метод класса.

2.5. Оценка решения задачи. Циклический возврат на уровень мысленных экспериментов

После того, как решение задачи, казалось бы, получено, приходит очередь самому неприятному, а иногда неожиданному вопросу: Какая ошибка у этого решения? Без оценки ошибки физические величины не имеют смысла. Оценки ошибок исходных параметров модели соответствуют этапам мысленного эксперимента и обезразмеривания. Численная модель дает возможность варьировать ее параметры и оценить ошибку результата. Иногда это проще, чем дифференцировать формулы решения.

На этапе критической оценки решения возникают идеи для следующей версии программы. Работа над текущей версией еще очень далека от завершения, но уже становится возможной и принципиально важной работа над новой версией. Технология распараллеливания работы над версиями возникла, видимо, в самолетостроении, но возможно и ранее. Все современные программы развиваются версиями. Существующие системы управления версиями позволяют это делать достаточно удобно.

2.6. Возврат в программирование модели

После того, как получено решение обезразмеренной математической задачи, встает ряд других вопросов, усложняющих математическую модель. Как преобразовать решение для его хранения и дальнейшего использования? Если данных совсем немного, то вопрос несущественен. Сбор, хранение и обработка больших данных становится непростой проблемой. В [12] обсуждаются различные виды современных баз данных и высокопроизводительные вычисления, основанные на программно-аппаратной акселерации, параллельной обработке информации и облачных архитектурах.

2.7. Возврат на уровень натурных экспериментов

Возможность разрабатывать веб-приложения (включая гаджеты) уже стала очень желанной компетенцией современного программиста [13]. Она пересекается с задачами вычислительного эксперимента и имеет многочисленные направления. Например, вики-проекты. Они позволяют пользователям использовать хорошо известный интерфейс Википедии. Некоторые фирмы создают свои собственные специализированные вики, независимые от основной Википедии. Они содержат сопутствующие материалы для технически сложных продуктов, включая программное обеспечение. Есть возможность организовать сложные автоматизированные поиски в основной Википедии и других Вики. В [14] приведен пример комплексного поиска, сравнения и визуализации исторических данных. Все эти возможности перспективны для применения в исследовательской экспериментальной работе.

2.8. Уровни моделей

В итоге вычислительный эксперимент можно представить как взаимосвязанные эксперименты с 4-мя слоями моделей:

1. Мысленный эксперимент на основе литературных данных, осознание способов оценки ошибок модели.
2. Натурный эксперимент с отработкой интерфейсов.
3. Математическая модель и эксперименты с базами данных.
4. Решение математических задач, анализ вычислительной устойчивости алгоритмов с использованием средств функционального анализа.

3. ВЫЧИСЛИТЕЛЬНАЯ НЕУСТОЙЧИВОСТЬ

Если учитывать сопротивление воздуха в предыдущей задаче, то в новой версии программного продукта придется строить разностную модель и итерационный метод ее решения. Итерационные алгоритмы эффективны и просты, отказаться от них практически невозможно. Обычно вычислительная неустойчивость появляется в итерационных алгоритмах.

Сущность вычислительной неустойчивости состоит в накоплении ошибок вычислений. Вычислительная неустойчивость является основной проблемой численных методов и экспериментов. Первые попытки использовать вычислительные модели физических процессов часто оказывались вычислительно неустойчивыми. Они приводили к результатам, далеким от истинных решений (например, схема Ричардсона [7]).

Простейшую модель осцилляций с частотой ω можно описать комплексным уравнением:

$$\frac{dX}{dt} = i\omega X;\ X(t=0) = A.$$

(например, уравнение Шредингера при нулевом импульсе в однородном потенциале).

Если A – действительное число, то решением этого уравнения будут колебания с амплитудой A, частотой ω и периодом $T = 2\pi/\omega$:

$$X = A\exp(i\omega t) = A\cos(\omega t) + iA\sin(\omega t),$$

$$\mathrm{Re}(X) = A\cos(\omega t);\ \mathrm{Im}(X) = A\sin(\omega t).$$

На плоскости (Re(X), Im(X)) должна получиться окружность с радиусом A.

Выберем шаг по времени τ << T и аппроксимируем производную как

$$\frac{dX}{dt} \approx \frac{X_{i+1} - X_i}{\tau}.$$

получим

$$\frac{X_{i+1} - X_i}{\tau} = iwX_i \Rightarrow X_{i+1} = X_i(1 + iw\tau);\ X_1 = A.$$

Пусть для простоты ω = 2π, тогда период колебаний должен быть 1. Пусть τ = 0.01. Такая аппроксимация уравнения кажется естественной, но эта видимость обманчива (**Рис. 1**).

Вопреки ожиданиям осцилляций с постоянной амплитудой получаются колебания с нарастающей амплитудой. Причина этого паразитного нарастания

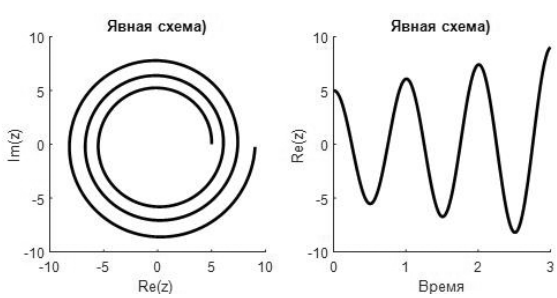

Рис. 1. *Явная разностная схема.*

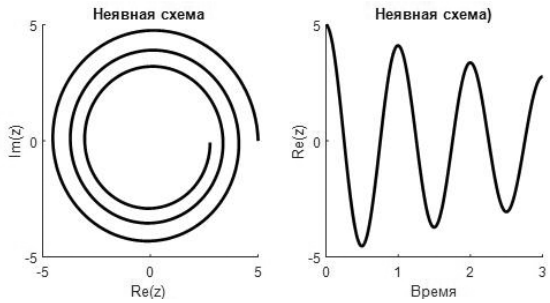

Рис. 2. *Неявная разностная схема*

амплитуды в том, что небольшие ошибки на каждом шаге итераций накапливаются, а в итоге решение все более и более отклоняется от истинного. Это явление называется вычислительной неустойчивостью. При уменьшении шага τ паразитное нарастание амплитуды уменьшается, но не исчезает.

Разностная схема, когда значение источника берется на текущем слое времени, называется явной разностной схемой. Возможна неявная разностная схема, когда функция источника берется не на текущем, а на новом слое времени (**Рис. 2**):

$$\frac{X_{i+1} - X_i}{\tau} = iwX_{i+1} \Rightarrow X_{i+1} = X_i / (1 - iw\tau);\ X_1 = A.$$

Эта разностная схема считается более удачной. Но теперь появляется паразитное затухание амплитуды. Амплитуда также не сохраняется, что также далеко не всегда допустимо. Вычислительная устойчивость разностной схемы не гарантирует отсутствия подобного вычислительного затухания. В [15] можно познакомиться с методами спектрального анализа разностных схем.

Примеры на Рис. 1 и Рис. 2 раскрывают противоположность понятий "вычислительной устойчивости" и "робастности" (устойчивости к выбросам). Предпочтительная разностная схема для вычислительного эксперимента должна быть минимально робастной из устойчивых. Возникает вопрос: "Можно ли, в конце-то концов, посчитать осцилляции так, чтобы амплитуда сохранялась?". Достаточно, например, воспользоваться разностной схемой Кранка-Николсона [15] (**Рис. 3**):

$$\frac{X_{i+1} - X_{i-1}}{2\tau} = iwX_i \Rightarrow X_{i+1} = X_{i-1} +$$

$$+2iw\tau X_i;\ X_1 = A;\ X_2 = (1 + iw\tau)A.$$

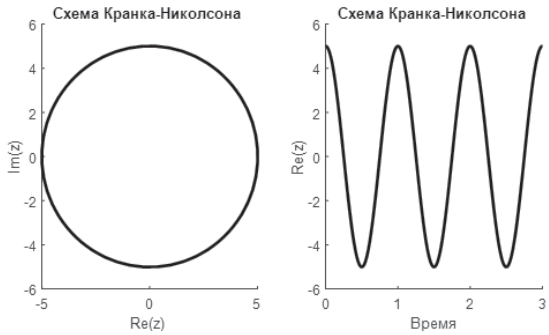

Рис. 3. *Постоянная амплитуда.*

Эффекты вычислительной неустойчивости проявляются не только в моделях осцилляций, но и тогда, когда осцилляций не видно, например, в трехслойной явной разностной схеме Ричардсона для задачи теплопроводности [7].

Многие задачи вычислительного эксперимента давно решены, а решения многократно опробованы. Для того чтобы оценить чужие нетривиальные решения, в новой версии программы стоило бы внимательнее отнестись к рекомендациям библиотек стандартных программ. Теоремы о свойствах разностных схем [7] носят достаточно общий характер. В [17], например, для уравнений математической физики рассматривалась возможность комбинировать разностные схемы с быстрым преобразованием Фурье и вложением решений в пространство Соболева W_2^6. С помощью этого подхода удавалось получать численными методами не только самодостройку диссипативных структур и магнитных доменов, но и солитоны.

В случае нелинейных задач доказать их вычислительную устойчивость бывает сложно, но можно включать дополнительные проверки в алгоритмы вычислений. Появление программ для работы с разреженными матрицами общего вида (sparse matrix) [8] расширяет возможности реализации численных методов и делает еще более актуальными применения основных определений и подходов теории разностных схем [7]. Математический подход к анализу и обоснованию синтеза разностных схем подразумевает знакомство с функциональным анализом и умение погружать разностные схемы в пополненные пространства [18].

4. ВЕРСИИ ПРОГРАММНЫХ ПРОДУКТОВ

В 90 годы сложилась определенная классификация версий программных продуктов. Альфа-версия – первые варианты разработанной программы. Они работоспособны, но реально их могут использовать только сами разработчики. Бета-версия – более дружественная к пользователю версия программы. Предполагается, что пользователь использует ее на свой страх и риск, что он сам будет разбираться со своими вопросами.

Первоначально бета-версии предназначались для ограниченного круга потребителей, достаточно квалифицированных, чтобы бесплатно и без претензий использовать программу, несмотря на возможные ошибки и сбои. В настоящее время к бета-тестированию привлекают более широкий круг лиц. Формирование этого круга лиц обеспечивает дальнейшее продвижение продукта на рынке.

Облачный сервис GitHub позволяет программистам достаточно удобно обмениваться своими наработками и материалами сопровождения программ. Релизы – дальнейшее совершенствование программного продукта. Нумерация релизов включает в себя номер версии и номер ее модификации. Например, ОС Windows 3.1 стала первой широко распространенной версией Windows. Иногда версии именуются по году выпуска, или же им присваивается специальные названия. За развитием версий программных продуктов скрывается не только работа программистов: разработка и программирование.

При разработке и применении программных продуктов большая часть затрат рабочего времени уходит на обнаружение и устранения багов, т.е. ошибок, несоответствий спецификациям программы, несоответствий ожиданиям пользователя. Эти несоответствия только в самых простых случаях устраняется формулировкой и проверкой гипотез. На проблемы тестирования целесообразно смотреть как на разновидность вычислительного эксперимента с версией программы. В современных браузерах

имеются режимы, позволяющие отслеживать структуры документов и связанные с ними форматы отображения данных, сообщения, пересылаемые между элементами формы. Известны программные средства, записывающие и воспроизводящие действия тестировщика при тестировании. Результаты тестирования должны накапливаться в базе данных с возможностью отображения в форме, удобной для дальнейшего использования. Объем необходимого ручного тестирования можно и нужно сокращать за счет автоматизации синтеза тестов и осуществления процесса тестирования [11]. Для параллельного выполнения многих технологических операций при разработке и сопровождении программных продуктов некоторые фирмы используют политику привлечения сторонних специалистов (outserting). Появляются фирмы, специализирующиеся на тестировании программных продуктов и новые специализации программистов.

Нередко устранение бага отражается на работе других частей программы, тогда его устранение разумно отложить до новой версии. Другим упрощенным способом избавления от сложных багов является создание специализированной версии с более узкими возможностями. Иногда оказывается полезной демонстрационная версия. Эти способы борьбы с багами соответствуют уровням математической модели, натурного эксперимента, программированию. Но есть и более глубокий уровень, на котором программы не исследуются и не улучшаются, а используются как средство решения задач.

5. ЗАКЛЮЧЕНИЕ

В работе рассмотрены три аспекта вычислительного эксперимента: обезразмеривание исходной модели, вычислительная неустойчивость решений и тестирование как разновидность вычислительного эксперимента. Показано, что на этапе обезразмеривания физическая модель преобразуется в математическую, что позволяет обоснованно переходить к следующему этапу вычислительного эксперимента – программированию. По результатам оценки ошибок и сложности решений проводится распараллеливание работы над версиями программных средств для вычислительного эксперимента. Применяются средства объектно-ориентированного программирования и системы управления версиями.

Для развития версий со сложными вычислениями и большими объёмами данных применяются средства высокопроизводительных вычислений, основанные на программно-аппаратной акселерации, параллельной обработке информации и облачных архитектурах.

Как правило, для новой версии программы строится разностная модель задачи и итерационные методы ее решения. Условия вычислительной устойчивости обычно оговариваются в современных инструкциях к стандартным библиотекам программ. Для новых алгоритмов необходим анализ устойчивости разностных схем на основе уточнения их спектральных свойств и использования методов функционального анализа.

Полученное решение преобразуется для его хранения и последующего использования в современных базах данных. При необходимости возможно применять распределенные базы данных и облачные хранилища.

Как разновидность вычислительного эксперимента рассмотрено также тестирование и использование альфа- и бета-версии, релизов.

ЛИТЕРАТУРА

1. Михайлов АП, Четверушкин БН. ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ. *Большая российская энциклопедия.* Том 6, стр. 161. Москва, 2006, Электронная версия (2016); https://bigenc.ru/mathematics/text/2379725.
2. Самарский АА, Михайлов АП. *Математическое моделирование: Идеи. Методы. Примеры.* М., Физматлит, 2001, 320 с.
3. Красовский ГИ, Филаретов ГФ. *Планирование эксперимента.* Минск, Изд. БГУ, 1982, 302 с.
4. Байцер Б. *Архитектура вычислительных комплексов.* М., Мир, 1974, 498 с.
5. Колмогоров АН. *Теория информации и теория алгоритмов.* М., Наука, 1987, 304 с.

6. Арзамасцева ГВ, Евтихов МГ, Лисовский ФВ, Мансветова ЕГ. Дифракция света на фракталах: сопоставление данных экспериментов с получаемыми численными методами фурье-образами изображений объектов. *РЭНСИТ: Радиоэлектроника. Наносистемы. Информационные технологии,* 2017, 9(2):221-228. DOI: 10.17725/rensit.2017.9.221.
7. Самарский АА. *Введение в теорию разностных схем.* М., Наука, 1971, 552 с.
8. Тьюарсон Р. *Разреженные матрицы.* М., Мир, 1977, 191 с.
9. Крамер Г. *Математические методы статистики.* М., Мир, 1975, 647 с.
10. Колмогоров АН. *Теория вероятностей и математическая статистика.* М., Наука, 1986, 535 с.
11. Фокке Г. *Практическая тестовая пирамида.* Ham Vocke. The Practical Test Pyramid [Электронный ресурс] martinfowler.com. URL: https://martinfowler.com/articles/practical-test-pyramid.html (дата обращения: 12.04.2022).
12. Евтихов ВГ, Евтихова НВ, Евтихов МГ, Евтихов МВ. *Высокопроизводительные вычисления.* Казань, Бук, 2020, 150 с.
13. Евтихов ВГ, Евтихова НВ. *WEB-разработка в образовательных и википедийных проектах.* Казань, Бук, 2018, 112 с.
14. Евтихов МВ, Евтихов ВГ. Визуализация исторических данных. *Молодой ученый,* 2019, 1(239):1-5.
15. Годунов СК, Рябенький ВС. *Разностные схемы.* М., Наука, 1973, 440 с.
16. Марчук ГИ. *Методы вычислительной математики.* М., Наука, 1973, 346 с.
17. Евтихов МГ. О численном решении уравнений математической физики с нелинейными источниками. *Радиотехника и электроника*, 2007, 52(8):1-4.
18. Анучина НН, Бабенко КИ, Годунов СК, Дмитриев НА, Дмитриева ЛВ, Дьяченко ВФ, Забродин АВ, Локуциевский ОВ, Малиновская ЕВ, Подливаев ИФ, Прокопов ГП, Софронов ИД, Федоренко РП. *Теоретические основы и конструирование численных алгоритмов задач математической физики.* М., Наука, 1979, 295 с.

Евтихов Михаил Георгиевич
к.ф.-м.н., с.н.с.
ФИРЭ им. В.А. Котельникова РАН
Фрязино 141190, Московская область, Россия
E-mail: emg20022002@mail.ru

Евтихов Владимир Георгиевич
к.т.н.
Московский политехнический университет
Москва 107023, Россия
E-mail: evg2002@yandex.ru.