значительную частотную дисперсию ε в области низких частот (рис.3). При этом граничная частота получается достаточно низкой и требования к точностным параметрам аппаратуры снимаются. Это означает, что возможен двухпараметровый каротаж околоскважинной зоны в диапазоне частот от сотен герц до нескольких килогерц.

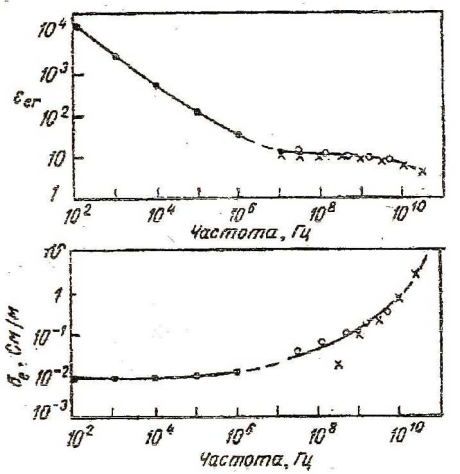

Рисунок 3.

Зависимость параметров σ и ε от частоты для суглинистой почвы.

Список использованных источников

1. Р.Кинг, Г.Смит. Антены в материальных средах. В 2-ух книгах. М., Мир. 1984. Кн.1, 355 с., кн.2. 813 с.
2. Черняк Г.Я. Электромагнитные методы в гидрогеологии и инженерной геологии. М: Недра. 1987. 212 с.

МЕТОД ОГРАНИЧЕНИЯ РАЗРЯДНОЙ СЕТКИ ЭВМ ДЛЯ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ КОМПЛЕКСОВ ПРОГРАММ

Жидченко В.В., Коварцев А.Н.

В условиях возрастающей сложности решаемых задач к алгоритмам, используемым в электронных устройствах, предъявляются все более высокие требования по быстродействию и надежности. В этих условиях одной из самых актуальных задач становится обеспечение высокого качества проектируемых программных продуктов (ПП).

Требуемое качество программного продукта достигается с помощью различных методов, которые можно объединить в две основные группы: методы безошибочного проектирования (пассивные методы) и методы обнаружения и устранения ошибок (активные методы) [1]. В данной работе рассмотрим проблему отладки вычислительных программ, основанных на использовании численных методов. Отладка вычислительных алгоритмов имеет следующие особенности.

Работа с множеством действительных чисел подразумевает широкий диапазон изменения входных параметров вычислительных подпрограмм. При испытаниях программного модуля, имеющего несколько вещественных параметров, не представляется возможным проверить его работоспособность при всех сочетаниях значений этих параметров. Возникает вопрос о получении наиболее достоверных результатов тестирования программных модулей при использовании ограниченного набора входных данных. Результатом тестирования вычислительного модуля является либо факт обнаружения программной ошибки, либо выдвижение гипотезы (с некоторой доверительной вероятностью) о том, что в программе отсутствуют ошибки вычислительного характера.

Для поиска вычислительных ошибок в программных модулях предлагается использовать метод, напоминающий схему независимых испытаний Бернулли. Суть метода заключается в генерации случайного вектора с координатами, равномерно распределенными в области определения входных параметров тестируемого модуля - Ω. Запуская модуль на выполнение с входными данными, определяемыми координатами сгенерированного вектора, будем фиксировать такие сочетания данных, которые приводят к вычислительным ошибкам. В результате повторения процедуры генерации и запуска не выполнение получим множество векторов, приводящих к ошибкам, которые образуют некоторую область Ω^E. Вероятность обнаружения ошибки при использовании предлагаемого алгоритма определяется отношением объемов ошибочной области (Ω^E) и области определения входных параметров тестируемого модуля (Ω):

$$P_E = V(\Omega^E) / V(\Omega), \qquad (1)$$

Размер области определения Ω определяет длительность полного просмотра пространства тестовых данных. Соотношение размеров области ошибок с размерами области определения может быть различным. В зависимости от его значения различают три типа ошибок:

1) грубые ошибки – проявляющиеся практически при любом запуске программы, вероятность обнаружения таких ошибок любым методом тестирования достаточно высока;

2) неточности – ошибки, появляющиеся не слишком часто и проявляющиеся в виде неустойчивой работы программы в зависимости от

значений исходных данных. Такие ошибки могут возникать, например, в результате неправильного применения математических функций (логарифм отрицательного числа или корень из отрицательного числа и т.п.);

3) редкие ошибки, которые практически не проявляются при работе программного модуля. Теоретически такая ошибочная ситуация может возникать в единственной точке, поэтому пользователь обычно считает, что работает с отлаженной программой.

Грубые ошибки и неточности достаточно легко обнаружить методом статистических испытаний. Вопрос об их возникновении при тестировании модуля определяется лишь числом тестовых точек. Редкие ошибки этим способом выявить практически невозможно. Для их нахождения следует использовать другие методы, например, доказательство корректности программ, однако авторам удалось реализовать метод поиска редких ошибок, основанный на статистических испытаниях.

Каков бы ни был период генератора тестовых данных, если размеры ошибочной области ничтожно малы по сравнению с областью изменения входных параметров модуля, то за экономически целесообразное время тестирования можно не попасть в точку ошибки. С другой стороны, дискретная природа представления действительных чисел в ЭВМ, ограниченность ресурсов ЭВМ и особенности реализации арифметических операций позволяют упростить процедуру локализации редких ошибок.

В работе [2] был предложен метод, позволяющий значительно увеличить размеры области ошибок путем искусственного снижения точности вычислений, производимых ЭВМ. Чтобы пояснить суть метода, рассмотрим пример. Пусть требуется найти значение аргумента, приводящего к ошибке при вычислении функции $f(x) = 1/(x-a)^2$. При $x=a$ попытка вычисления этой функции приведет к ошибке деления на ноль. Во всех остальных точках теоретически функция должна возвращать значимый результат.

Рассмотрим задачу нахождения корней уравнения $(x-a)^2 = 0$ на ЭВМ с t+1 значащими цифрами.

Пусть $x = x_t 10^t + x_{t-1} 10^{t-1} + \ldots + x_0$ и $a = a_t 10^t + a_{t-1} 10^{t-1} + \ldots + a_0$ – представление десятичных чисел в (t+1)-значной арифметике. Выражение $(x-a)^2$ можно записать в виде:

$$(x-a)^2 = (x_t-a_t)^2 10^{2t} + 2(x_t-a_t)(x_{t-1}-a_{t-1}) 10^{2t-1} + [(x_{t-1}-a_{t-1})^2 + 2(x_t-a_t)(x_{t-2}-a_{t-2})] 10^{2t-2} + 2[(x_t-a_t)(x_{t-3}-a_{t-3}) + (x_{t-1}-a_{t-1})(x_{t-2}-a_{t-2})] 10^{2t-3} + \ldots + 2(x_1-a_1)(x_0-a_0)10 + (x_0-a_0)^2$$

Тогда исходное уравнение, с учетом ограниченности разрядной сетки (при условии, что округление результата производится путем отбрасывания младших разрядов), можно заменить системой уравнений:

$$\begin{cases} (x_t-a_t)^2 = 0 \\ (x_t-a_t)(x_{t-1}-a_{t-1}) = 0 \\ (x_{t-1}-a_{t-1})^2 + 2(x_t-a_t)(x_{t-2}-a_{t-2}) = 0 \\ \dots \\ (x_t-a_t)\,(x_0-a_0) + (x_{t-1}-a_{t-1})(x_1-a_1) = 0 \end{cases} \quad (2)$$

Например, при t=4 система (2) примет вид:

$$\begin{cases} (x_4-a_4)^2 = 0 \\ (x_4-a_4)(x_3-a_3) = 0 \\ (x_3-a_3)^2 + 2(x_4-a_4)(x_2-a_2) = 0 \\ (x_1-a_4)\,(x_1-a_1) + (x_3-a_3)(x_2-a_2) = 0 \\ (x_2-a_2)^2 + 2(x_4-a_4)(x_0-a_0) = 0 \end{cases} \quad (3)$$

Корни системы уравнений (3): $x_4=a_4, x_3=a_3, x_2=a_2, \forall x_1, \forall x_0$. (4)

Таким образом, из (4) видно, что в реальных ЭВМ уравнение $(x-a)^2 = 0$ имеет корень в целом ряде точек, отличающихся несколькими младшими разрядами. Поэтому каждая ошибочная точка обладает некоторой окрестностью, в которой также происходят сбои в работе программы. Размер этой окрестности, очевидно, зависит от количества значащих разрядов, используемых для представления чисел в ЭВМ (в нашем примере оно равно t+1). Вычислительные эксперименты подтверждают выдвинутую гипотезу. На рис. 1 представлены результаты вычисления функции $f(x,y) = 1 / ((1-x)^4+(1-y)^4)$ на 8-ми разрядной тестовой ЭВМ.

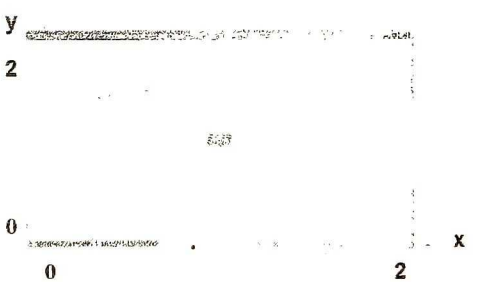

Рисунок 1.
Область ошибок при вычислениях функции $f(x,y) = 1 / ((1-x)^4+(1-y)^4)$ на 8-ми разрядной ЭВМ

Несмотря на то, что указанная функция не имеет значения в единственной точке (1;1), попытка ее вычисления на тестовой ЭВМ привела к ошибкам деления на ноль в целом ряде точек, изображенных на рис.1. Воспользуемся данным фактом для оптимизации процесса тестирования вычислительных программных модулей. Из формулы (1) видно, что для увеличения вероятности обнаружения ошибки необходимо увеличить размер области Ω^E, который зависит от числа выполняемых при вычислениях операций m (приводящих к ошибкам округления) и от точности вы-

числений t. Если искусственно вводить дополнительные арифметические операции, возрастет время тестирования, следовательно, целесообразнее оперировать точностью вычислений. Для этого введем два параметра [2]:

t – размер разрядной сетки гипотетической ЭВМ;

r – область «нечувствительности» к операциям «+» и «-» ($r \in [0..t]$).

Выполняя вычисления, будем преобразовывать реальные числа, участвующие в арифметических операциях (обозначим их a и b), к разрядности t. Если переопределить сами арифметические операции, то мы получим модель t-разрядной ЭВМ.

$$a \pm b = \begin{cases} 0, \text{ если } |[a]_t \pm [b]_t| < (\max\{ |[a]_t|, |[b]_t| \}) / 10^{t-r} \\ [[a]_t \pm [b]_t]_t, \text{ иначе} \end{cases}$$

$$a * b = [[a]_t * [b]_t]_t \qquad (5)$$

$$a / b = [[a]_t / [b]_t]_t$$

Для демонстрации метода искусственного ограничения разрядной сетки ЭВМ приведем следующий пример. Рассмотрим модуль, вычисляющий значение функции:

$$f(x,y) = 1 / ((x^4 - 4*x^3 + 6*x^2 - 4*x + 1) + (y^4 - 4*y^3 + 6*y^2 - 4*y + 1)) \qquad (6)$$

Эта функция эквивалентна рассмотренной выше, но для ее вычисления используется больше арифметических операций. На рис.2 изображены результаты тестирования программы вычисления функции (6) при различных моделируемых значениях точности вычислений (параметр t в формуле 5). Из рис.2 видно увеличение размеров ошибочной области при снижении точности представления величин, участвующих в вычислениях.

t = 8 t = 5 t = 2

Рисунок 2.

Результаты тестирования программы вычисления функции
$f(x,y) = 1 / ((x^4 - 4*x^3 + 6*x^2 - 4*x + 1) + (y^4 - 4*y^3 + 6*y^2 - 4*y + 1))$
при различных значениях t

На рис.3 приведен график зависимости диаметра области ошибок от размера разрядной сетки, построенный по результатам исследований.

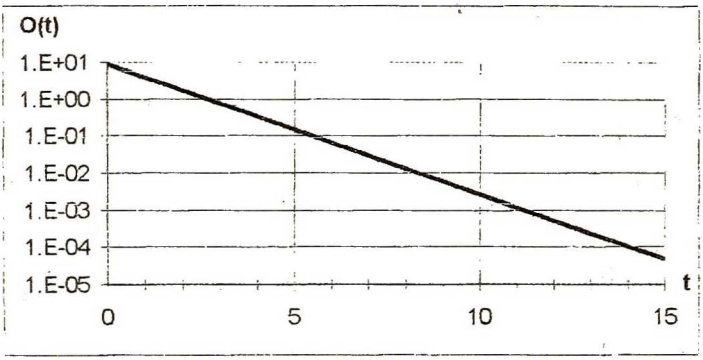

Рисунок 3.
Зависимость диаметра области ошибок от разрядной сетки ЭВМ

Опишем общий алгоритм тестирования с учетом использования метода ограничения разрядной сетки ЭВМ. На первом шаге число значащих разрядов t устанавливается равным 1 или 2, и производится тестирование программного модуля с указанной точностью. Если ошибок на этом шаге не будет выявлено, то это означает, что, либо они отсутствуют, либо размеры ошибочной области настолько малы, что ее не удастся обнаружить методом статистических испытаний, и для поиска ошибок следует воспользоваться другими методами.

Если ошибки обнаружены, то точность вычислений повышается за счет увеличения параметра t, а тестирование производится в новом подпространстве, непосредственно обрамляющем область ошибок. При повышении точности вычислений размер ошибочной области будет постепенно уменьшаться. Процесс повторяется до тех пор, пока параметр t не станет равным действительной размерности разрядной сетки ЭВМ, на которой производится тестирование.

Повышение производительности тестирования достигается за счет постепенного сужения пространства поиска ошибок, а значит, увеличения вероятности обнаружения ошибки за определенное число тестов на каждом шаге. На рис. 4 изображен график зависимости числа тестов, выполняемых до обнаружения первой ошибки, от параметра t.

Из рисунка видно, что ошибка, которую практически невозможно обнаружить методом статистических испытаний при тестировании с реальной точностью ЭВМ, проявляется очень быстро при искусственном ограничении точности вычислений. Так, уже при t=12 (в реальной ЭВМ действительные числа представлены с точностью до 16 знаков), количество тестов до появления первой ошибки превышает 10^6. Однако уменьше-

ние параметра t до 3 позволяет выявить ошибку в самом начале испытаний.

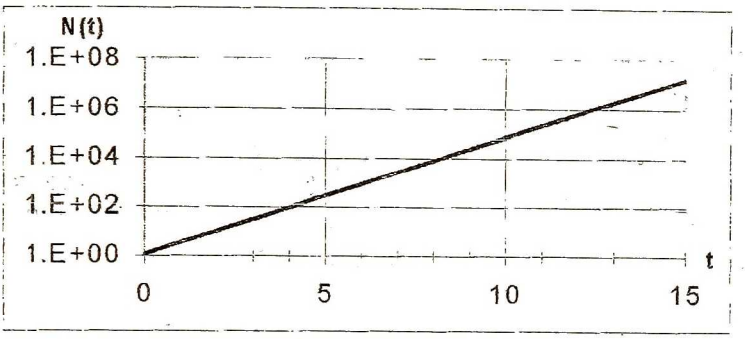

Рисунок 4.
Зависимость числа тестов, выполняемых до обнаружения первой ошибки, от точности вычислений

Рассмотренный метод был с успехом применен в подсистеме автоматизированного тестирования объектов технологии графосимволического программирования [2]. Созданное на его основе программное обеспечение было внедрено в учебный процесс кафедры информационных систем и технологий СГАУ

Список использованных источников

1. Липаев В.В. Тестирование программных средств. Методическое руководство. – М.: МГТУ «Станкин». 1999. – 118с.
2. Коварцев А.Н. Автоматизация разработки и тестирования программных средств на основе технологии графосимволического программирования: Дис. на соиск. учен. ст. докт. тех. наук. — Самара. 1999. — 284 с.

ТАКСОНОМИЧЕСКИЕ МОДЕЛИ УПРАВЛЕНИЯ ДОРОЖНЫМ ДВИЖЕНИЕМ

Михеев С.В.

Усложнение дорожно-транспортных условий требует непрерывного совершенствования методов и средств управления движением. В комплекс мероприятий, направленных на решение задачи обеспечения нормального функционирования современного города в условиях повышенной автомобилизации. автоматизация управления дорожным движением занимает одно из ведущих мест [1]. При этом современная автоматизированная система управления дорожным движением - это комплекс