УДК 681.2.088

ПРОГРАММНЫЕ СРЕДСТВА ОБРАБОТКИ НЕТОЧНЫХ ДАННЫХ, ВЫПОЛНЯЕМОЙ В СРЕДСТВАХ ИЗМЕРЕНИЙ

© 2016 К. К. Семенов

Санкт-Петербургский политехнический университет Петра Великого

Статья поступила в редакцию 16.12.2016

Данная работа посвящена описанию программных средств, с применением которых должна производиться разработка метрологически значимого программного обеспечения современных измерительных систем. В статье рассмотрены принципы дополнения расчетов, выполняемых с результатами измерений при их математической обработке, процедурами оценки погрешности конечных результатов вычислений, унаследованной от неточных исходных данных. На основе сделанных выводов предложено организовать программную библиотеку, позволяющую разработчику без затруднений дополнить создаваемое им программное обеспечение для средств измерений свойством собственного метрологического сопровождения. В работе представлен пример реализации элементов данной библиотеки и дано его обоснование. Использование программных средств, построенных на изложенных в статье принципах, позволяет добиться важных свойств для современных средств измерений, привлекающих для получения результатов измерений вычислительные устройства.
Ключевые слова: метрологически значимое программное обеспечение, метрологическое автосопровождение, наследственная погрешность.

ВВЕДЕНИЕ

Развитие вычислительной техники позволило автоматизировать расчеты, в том числе выполняемые в измерительных задачах. Современные приборы и информационно-измерительные системы широко используют программируемые вычислительные средства: производимая ими обработка данных уже немыслима в форме ручного труда. Выполняемые в средствах измерений расчеты ведутся со значениями, искаженными погрешностью, и вследствие этого всегда завершаются получением неточного результата. Для обеспечения возможности принять обоснованные решения на его основе важно сопроводить такой результат характеристиками его неопределенности, унаследованной от неточных исходных данных. Это обстоятельство играет существенную роль в современных средствах, выполняющих косвенные, совокупные или совместные измерения, поскольку в них в обязательном порядке в соответствии с утвержденной методикой производится обработка результатов прямых измерений. Важным примером таких устройств выступают счетчики потребляемой тепловой энергии, размещаемые в узлах учета в жилых домах. Результаты выполняемых ими измерений являются косвенными и формируются на основе вычислений, которые производятся по нормативным формулам и алгоритмам с данными, поступающими от входящих в состав теплосчетчиков датчиков. Необходимость сопровождения результатов этих расчетов оценкой их погрешности закреплена требованиями стандарта [1]. В работе [2] отмечено, что автоматическое выполнение такой оценки позволит преодолеть ряд сложностей и затруднений, имеющих место в современном метрологическом обеспечении результатов измерения количества потребляемого тепла.

Семенов Константин Константинович, кандидат технических наук, доцент кафедры измерительных информационных технологий. E-mail: semenov.k.k@iit.icc.spbstu.ru

Действия по оценке погрешности результатов вычислений с неточными данными поддаются алгоритмизации. Получить такие оценки можно автоматически и, что особенно важно, средствами самой программы в темпе производимых вычислений – то есть не после получения основного результата, а именно вместе и одновременно с ним [3]. Данное обстоятельство позволяет освободить пользователя от необходимости вести соответствующие метрологические расчеты.

В настоящей работе рассмотрены программные средства, которые позволяют выполнять оценку погрешности результатов обработки неточных данных, производимой в средствах измерения.

1. ПРИНЦИПЫ ОРГАНИЗАЦИИ ВЫЧИСЛЕНИЙ С НЕТОЧНЫМИ ДАННЫМИ

Программы расчетов с неточными исходными данными должны сопровождаться подпрограммами автоматической оценки неопределенности их результатов. Это позволит пользователю получить индивидуальную характеристику качества для каждого сообщенного ему результата вычислений. Данное утверждение является не только программным для современного состояния технологий обработки данных, но и поддержива-

ется конкретными тенденциями в этой области. Получило глубокую теоретическую проработку и в последующем важное прикладное значение специальное направление в вычислительной математике – интервальный анализ [4], – целью которого является автоматическая оценка пределов возможных значений результатов вычислений с данными, заданными интервалами. Его распространению способствует разработка программных библиотек, позволяющих переходить при разработке специализированных программ расчетов к вычислениям с интервалами [5]. Появились специализированные аппаратные процессоры [6], работающие с нечеткими числами и предоставляющие результаты вместе с характеристиками их неопределенности. Закрепление данных тенденций выполнено разработкой международными организациями нормативных документов, определяющих предписанные правила оценки неопределенности результатов расчетов с неточными данными, а также правила формализованного представления их в компьютерах и программных средствах математической обработки (новые типы данных, идущие на смену классическому представлению чисел с плавающей точкой). В качестве примера следует упомянуть стандарт [7].

Современные технологии разработки программного обеспечения позволяют переложить на него обязанности по оценке погрешности, наследуемой от исходных данных. Широко применяемое составление программ из отдельных модулей, решающих частные подзадачи, наводит на естественный путь – снабдить каждый такой модуль процедурой оценки погрешности выдаваемых им результатов. Это соображение было положено в основу первых пакетов линейной алгебры, сообщавших пользователю оценку возможной ошибки результата, вызванной несовершенством используемых вычислительных схем. В метрологических задачах модульная методология составления программ, выполняющих обработку результатов измерений и одновременно с тем оценку характеристик их погрешности, была описана в [8] (рис. 1).

На рис. 1 представлена структура программы обработки результатов прямых измерений (образующих компоненты вектора $\vec{x}$), заключающейся в последовательном совершении с ними ряда отдельных операций. Каждый совершаемый шаг при штатных вычислениях дополнен оценкой погрешности $\Delta W(\vec{x}, \Delta\vec{x})$ текущих результатов, связывающих ее с характеристиками погрешности $\Delta\vec{x}$ значений $\vec{x}$ по заранее установленным для данной операции правилам. Возникающая последовательность оценок реализует метрологическое сопровождение основной программы обработки и сопоставляет ее конечному результату $\vec{W}$ его индивидуальную оценку погрешности ΔW, наследуемой от неточных исходных данных расчета. Будем в дальнейшем называть такую погрешность наследственной вслед за работой [9].

Составление программ для выполнения метрологически значимых расчетов из отдельных блоков имеет ряд несомненных преимуществ: обеспечивается возможность производить разили доработку программного обеспечения быстро, привлекая для этого уже готовые модули; нет необходимости в отладке отдельных операций, поскольку они были апробированы ранее; получаемый в результате программный код хорошо структурирован – как следствие, повышается его качество, что облегчает возможности его дальнейшей поддержки и развития.

Таким образом, идея сопровождать каждую операцию с неточными данными процедурной оценкой наследуемой погрешности оказывается удачной. При ее практической реализации возникает вопрос: как разбить последовательность совершаемых при обработке действий на отдельные шаги и что считать за таковые? Пусть, например, обработка заключается в решении системы алгебраических уравнений, коэффициентами которых выступают поступившие в программу результаты прямых многократных измерений. Разбиение на отдельные операции может быть произведено не единственным образом. Можно метрологически сопроводить оценкой погрешности операцию усреднения результатов многократных измерений, а затем – операцию решения системы уравнений, а можно – дополнительно разбить задачу поиска решений этой системы на этапы: на осуществление действий по улучшению обусловленности матрицы системы, на

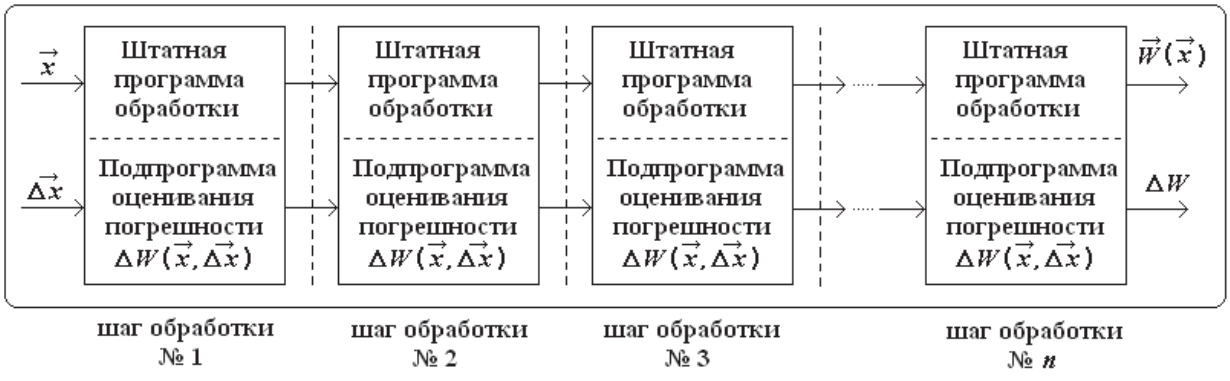

Рис. 1. Поэтапная оценка погрешностей вычислений [8]

выполнение операции взятия обратной к ней матрицы, на операцию перемножения матриц и тому подобное.

Современные объектно-ориентированные технологии разработки программного обеспечения позволяют выполнить предельное разбиение: считать отдельными шагами обработки далее неделимые элементарные операции, из которых состоит любая программа вычислений, – арифметические операции, логические операции, вызов математических примитивов (рис. 2) [10]. Тогда представленная идеология автоматической оценки наследственной погрешности будет состоять в составлении программ, которые вычисляют оценки погрешности каждого промежуточного результата.

Рис. 2 в качестве примера, поясняющего утверждение о составе элементарных операций, содержит фрагменты кода, реализующего вычисление значения функции $\operatorname{sinc} x = \dfrac{\sin x}{x}$, встречающейся в метрологических приложениях, при всех значениях аргумента *x*. Как видим, данная процедура привлекает для получения результата арифметические операции (деление), логические операции (сравнение аргумента *x* с нулем), вызов элементарных функций из стандартных библиотек (вычисление значения функции «синус»), работу с памятью данных. Данные операции не могут быть разделены на подоперации, выполняются атомарно и, значит, представляют собой предельную детализацию программного кода.

Рассмотрение подобной триады (арифметические действия – логические действия – математические примитивы) позволяет получить важное дополнительное свойство для метрологического программного обеспечения. Если мы дополним каждую операцию оценкой погрешности ее результата, унаследованной от входных операндов, то полученная программная конструкция будет носить универсальный характер: любая программа, ее использующая, автоматически будет метрологически сопровождена. При этом не потребуется изменять содержательную часть программного кода: он как состоял из заданной последовательности элементарных операций, так и будет из них состоять без необходимости вносить в них изменения.

Таким образом, представляется разумным составить подпрограммы оценивания погрешности для каждого элементарного вычислительного действия. Это позволит сопроводить оценкой погрешности любой промежуточный результат вычислений, а не только конечный, что имеет смысл и метрологическую ценность: становится возможным при необходимости автоматически проследить распространение погрешности исходных данных сквозь программу обработки [11]. С математической точки зрения составление указанных подпрограмм, реализующих метрологические сопровождение расчетов с неточными данными, сводится:

1) к использованию в качестве типа данных не просто чисел с плавающей точкой (`float`, `double`), а таких структур данных, которые несли бы также информацию о погрешности,

2) к определению правил действий с операндами такого нового типа данных и к определению значений элементарных математических функций от них.

Для реализации такого подхода требуется перейти от вычислений на основе алгебры действительных чисел к новой алгебре (интервалов, гистограмм, нечетких переменных или другой). Ее свойства должны быть близки к алгебраическим свойствам действительных чисел, чтобы переход мог быть осуществлен в полной мере и вызывал бы минимальные ограничения при использовании. Реализация такого алгебраического подхода позволяет удобно и автоматически получать оценки характеристик погрешности в темпе основных вычислений и тем самым реализует свойство, названное в работах [3, 8] метрологиче-

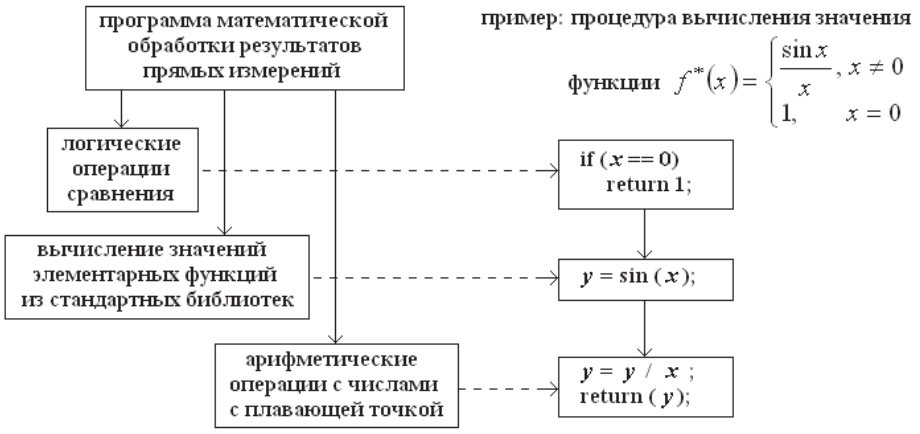

Рис. 2. Основные элементарные единицы, из которых строятся вычисления, выполняемые программами математической обработки [10]

ским автосопровождением программ обработки результатов измерений.

В [12] отмечается, что «методология метрологического автосопровождения, если оставить в стороне вопросы его технической реализации, практически совпадает с теорией расчета и оценки точности (характеристик погрешности)», что говорит о крайней целесообразности ее применения в качестве базового принципа при обработке неточных данных.

2. ОЦЕНКА ПОГРЕШНОСТИ РЕЗУЛЬТАТА ВЫЧИСЛЕНИЙ С НЕТОЧНЫМИ ДАННЫМИ

Для того, чтобы указать, каким образом метрологически сопроводить элементарные операции, из которых состоят программы обработки вычислений с результатами измерений, необходимо определить, как наипростейшим образом организовать оценку погрешности ее конечного результата.

Обозначим как $y = f(x_1, x_2, \ldots, x_n)$ функцию, которая реализует математическую обработку результатов прямых измерений $x_1, x_2, \ldots, x_n$, поступающих ей на вход из измерительных каналов средства измерений. Значение y есть конечный результат, предоставляемый пользователю. Пусть входные данные $x_1, x_2, \ldots, x_n$ приняли конкретные значения $x_{1\,0}, x_{2\,0}, \ldots, x_{n\,0}$, полученные в ходе соответствующих измерений. Пусть для значений $x_{10}, x_{20}, \ldots, x_{n0}$ заданы характеристики их погрешности, нормированные пределами $\Delta_1, \Delta_2, \ldots, \Delta_n$ допускаемых значений абсолютной погрешности $\Delta x_{10}, \Delta x_{20}, \ldots, \Delta x_{n0}$, такими, что $|\Delta x_{i0}| \le \Delta_i$ при $i = 1, 2, \ldots n$.

Тогда общепринятым подходом к оценке погрешности конечного результата расчетов y (по сути – результата косвенного измерения) является использование линеаризации, заключающейся в замене функции f в небольшой окрестности точки $(x_{1\,0}, x_{2\,0}, \ldots, x_{n\,0})$ ее линейным приближением. Оценка предела погрешности Δ_y, выражающего характеристику абсолютной погрешности величины y, унаследованной от исходных данных расчета, может быть получена следующим образом:

$$\Delta_y \approx \sum_{i=1}^{n} \left| \frac{\partial f\left(x_{1_0}, x_{2_0}, \ldots, x_{n_0}\right)}{\partial x_i} \right| \cdot \Delta_i \,. \quad (1)$$

Приближенный характер данной формулы не снижает ее практического значения. Действительно, в метрологии принято представлять результаты оценок погрешности в округленном виде, в котором сохраняют одну (максимум две) значащие цифры. В ситуации, когда пределы погрешностей $\Delta_1, \Delta_2, \ldots, \Delta_n$ относительно невелики, формула (1) предоставляет достоверные результаты.

Приведенное выражение задействует сведения о метрологических характеристиках $\Delta_1, \Delta_2, \ldots, \Delta_n$, известных из нормативной документации к используемым в составе средства измерений измерительным каналам, на выходе которых сформированы поступившие значения $x_{1\,0}, x_{2\,0}, \ldots, x_{n\,0}$. Неизвестными в формуле (1) являются только значения частных производных, которые обусловлены видом функции f, реализующей выполняемую математическую обработку. Таким образом, суть процедуры оценки погрешности может быть сведена к вычислению значений производных, которое необходимо организовать с предельно возможной точностью.

Для функции $f(x_1, x_2, \ldots, x_n)$ многих переменных, представленной программным кодом, давно известен метод автоматического дифференцирования [13], позволяющий так сопроводить вычисления, чтобы при заданных пользователем исходных данных $(x_1, x_2, \ldots, x_n) = (x_{1\,0}, x_{2\,0}, \ldots, x_{n\,0})$ могли бы быть получены не только значения самой функции $y_0 = f(x_{1\,0}, x_{2\,0}, \ldots, x_{n\,0})$, но и всех ее частных производных $\partial f(x_{1\,0}, x_{2\,0}, \ldots, x_{n\,0}) / \partial x_i$ при $i = 1, 2, \ldots n$. Для этого необходимо перейти от вычислений с обычными числами с плавающей точкой к вычислениям с так названными дуальными числами [3, 13]. Такой переход в полной мере согласуется с требованиями и соображениями, приведенными в предыдущем пункте статьи.

Таким образом, оценка погрешности результатов расчетов, унаследованной от неточных исходных данных, выполняется с применением автоматического дифференцирования. Если задать конкретный способ выражения характеристик неопределенности входных данных, то суть метрологического сопровождения оказывается сведена к простой комбинации, проиллюстрированной рис. 3.

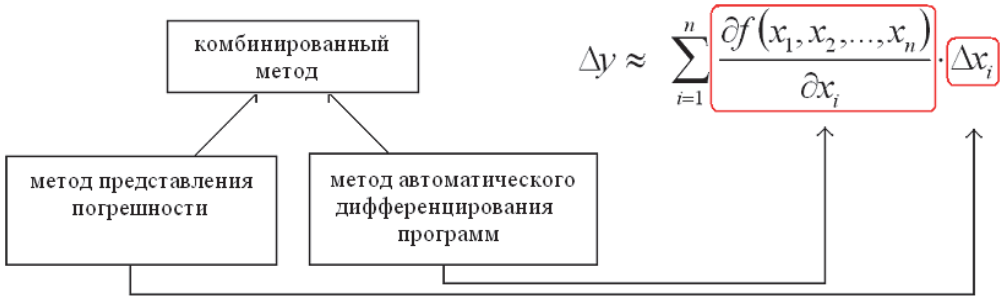

Рис. 3. Комбинация методов представления погрешности и автоматического дифференцирования для метрологического сопровождения вычислений с неточными данными

Представленная комбинация, по всей видимости, является наиболее целесообразной для использования в метрологических задачах: она согласуется с требованиями метрологии, не выходит за рамки устоявшихся в ней классических подходов к оценке погрешности и позволяет получать достоверные результаты, что подтверждается существенным количеством примеров ее применения [10, 14].

3. РЕАЛИЗАЦИЯ ПРОГРАММНЫХ СРЕДСТВ, ОБЕСПЕЧИВАЮЩИХ МЕТРОЛОГИЧЕСКОЕ СОПРОВОЖДЕНИЕ В ПРОГРАММАХ ВЫЧИСЛЕНИЙ С НЕТОЧНЫМИ ДАННЫМИ

Рассмотрим программную реализацию представленного подхода, заключающуюся, как было обсуждено выше, в составлении нового типа данных и задании правил элементарных операций с ним. Удобно организовать соответствующие программные средства в виде программной библиотеки – это является общепринятой практикой в технологиях разработки программного обеспечения. Такая библиотека может быть реализована на любом языке программирования. Поскольку наибольшее распространение в прикладных задачах метрологии и измерительного дела имеют языки программирования C/C++, то рассмотрим реализацию в их рамках.

Основными требованиями, выдвигаемыми к практической реализации, являются простота использования, оптимальная скорость вычислений и минимальный объем привлекаемой дополнительной памяти. Использование метрологического сопровождения расчетов с неточными данными не должно затруднять программисту разработку метрологически значимого программного обеспечения.

Как было отмечено, концепция реализации метрологического сопровождения заключается в очень простом с точки зрения современного программирования действии: в замене типа данных, с которым производятся вычисления, с чисел с плавающей точкой (`float, double`) на новый тип данных. Для последнего должны быть определены и реализованы в программной библиотеке все необходимые правила работы с ним: арифметические операции, операции сравнения, вычисление элементарных функций и математических примитивов. Тогда для реализации метрологического сопровождения пользователю необходимо будет выполнить минимальный набор действий в его программном коде: подключить библиотеку к своему проекту, заменить тип данных для всех исходных данных и связанных с ними переменных для промежуточных вычислений и ввести в код характеристики погрешности для входных аргументов. Данные действия не затрагивают содержательной части программы, что является несомненным преимуществом: новый код не потребует дополнительного видоизменения и последующей отладки.

Назовем новый тип данных, реализующий метрологическое сопровождение, как `mdouble` (сокращение от metrological double). Будем называть объект, реализуемый им, *метрологическим числом*. Его можно организовать в виде иерархии типов, построенной на структурах и классах языка C++, представленной в табл. 1. Данный тип должен содержать не только само значение числа (результата измерения x), но и сведения об его погрешности Δx, в качестве которых в простейшем случае следует выбрать пределы $\Delta_{сист}$ и $\sigma^2_{случ}$ допускаемых значений

Таблица 1. Введение структур данных, образующих тип `mdouble`

Описание структуры данных, задающих интервал возможных значений

```
struct interval
  { double left;          // левая граница интервала.
    double right; };      // правая граница интервала.
```

Описание типа данных, содержащего информацию о характеристиках погрешности

```
class uncertainty
  { public:
    interval systematic;   // предел значений сист. составляюшей погрешности.
    double  dispersion; }; // предел дисперсии случайной составляющей погрешности.
```

Описание типа данных mdouble

```
#define  n  20      /* число входных параметров программы расчета, возмущенных
погрешностью, доступное во всех частях программы при ее компиляции. */
class mdouble
  { public:
    double value;          // значение результата измерений.
    double der[n];                 // массив для значений частных производных.
    static  uncertainty* errors[n]; };      /* указатель на массив переменных, хранящих
                    сведения о характеристиках погрешности исходных данных. */
```

для систематической и для дисперсии случайной составляющих погрешности. Такая комбинация охватывает подавляющее большинство измерительных ситуаций и носит достаточно обобщенный характер. В случае, если в технической документации присутствуют сведения о характеристике только одной составляющей погрешности, вторую характеристику можно положить равной нулю. Если заданы пределы для полной погрешности (без разделения на случайную и систематическую составляющие), то следует считать, что погрешность носит систематический характер – это будет соответствовать получению достоверных результатов по итогам выполняемых вычислений.

В соответствии с объектно-ориентированным подходом в программировании необходимо, чтобы все созданные классы были снабжены конструкторами создания, конструкторами копирования и по необходимости деструкторами. Также должны быть определены операторы приведения к другим типам данных.

Представленные в табл. 1 фрагменты кода программной библиотеки, реализующей метрологический тип данных, требуют комментария. Поскольку сведения о характеристиках погрешности в соответствии с формулой (1) используются только при оценке погрешности конечных результатов вычислений, а сами они сопоставляются только исходным данным расчета, то удобно хранить их в памяти отдельно от самих чисел типа `mdouble`. Последним необходимо лишь сообщить указатель на место их хранения в памяти.

Каждому входному аргументу расчета сопоставляется метрологическое число типа `mdouble` с n ячейками памяти, используемыми для хранения частных производных, отражающих чувствительность метрологического числа к погрешности входных данных. Каждая ячейка массива der сопоставлена конкретной ячейке массива errors. Для входных аргументов расчета при инициализации переменных в массив der должны быть записаны нули во все ячейки, кроме одной – той, что сопоставлена конкретному входному аргументу. В нее надлежит занести значение, равное единице. Действительно, погрешность значения входного аргумента x_i не зависит от погрешности других входных аргументов $x_1, x_2, \ldots x_{i-1}, x_{i+1}, \ldots x_n$ и ей сопоставлены частные производные

$$\frac{\partial x_i}{\partial x_j} = \begin{cases} 0, i \neq j \\ 1, i = j \end{cases},$$

все из которых равны нулю, кроме производной, занесенной в i-ую ячейку массива der.

При несложных вычислениях обычно имеет место ситуация, когда большинство элементов массива der для промежуточных результатов вычислений также будут заполнены нулями. Следовательно, непосредственное хранение в памяти частных производных по всем входным аргументам зачастую является избыточным решением, допустимым только тогда, когда n невелико. Если количество входных параметров программы расчета, являющихся источником наследственной погрешности конечных результатов вычислений, сравнительно большое, то удобнее хранить в памяти только те частные производные, значения которых отличны от нуля. С другой стороны такой подход усложняет процедуры выполнения арифметических операций с числами типа `mdouble` и может привести к удлинению времени выполнения программы обработки. Пример соответствующей реализации представлен в табл. 2.

После введения нового типа данных, соответствующего метрологическому числу, необходимо обеспечить для него верное выполнение арифметических действий и определить значения элементарных математических функций. Операции сложения, вычитания, умножения и деления могут быть введены так, как это представлено в табл. 3.

Как видим, все операции могут быть описаны единообразно, отличия имеют место только при вычислении частных производных, сопоставленных метрологическому числу – результату совершаемого арифметического действия.

В табл. 3 красными рамками выделены те строки кода, которые соответствуют определению значений частных производных. Это ключевые действия при формировании правил работы с метрологическими числами. Операции, совершаемые с элементами массива der при арифме-

Таблица 2. Пример введения типа данных `mdouble`, представляющего метрологические числа без хранения нулевых значений частных производных

Описание типа данных, соответствующих метрологическому числу

```
class mdouble
{ public:
  double value;    // значение результата измерений.
  int *der_ptr;   /* массив, содержащий индексы ненулевых частных производных. */
  double *der;                    /* указатель на массив со значениями ненулевых частных
                                     производных. */
  static uncertainty **errors; };   /* указатель на массив переменных, хранящих сведения о
                                характеристиках погрешности исходных данных. */
```

Таблица 3. Пример введения арифметики метрологических чисел

Операция сложения

```
mdouble mdouble::operator +(mdouble& x, mdouble& y)
 { int i; mdouble result;
   result.value = x. value + y. value;      // значение самого результата сложения.
   for (i = 0; i < n; i++)
       result.der[i] = x.der[i] + y.der[i];   /* значения частных производных, сопоставленных
                  метрологическому числу – результату сложения. */
   return (result); }
```

Операция вычитания

```
mdouble mdouble::operator -(mdouble& x, mdouble& y)
 { int i; mdouble result;
   result.value = x. value - y. value; // значение самого результата вычистания.
   for (i = 0; i < n; i++)
       result.der[i] = x.der[i] - y.der[i];          /* значения частных производных,
                  сопоставленных метрологическому числу – результату вычитания. */
   return (result); }
```

Операция умножения

```
mdouble mdouble::operator *(mdouble& x, mdouble& y)
  {
   int i; mdouble result;
   result.value = x.value * y.value; // значение самого результата умножения.
   for (i = 0; i < n; i++)
       result.der[i] = x.der[i] * y.value + y.der[i] * x.value;      /* значения частных
производных, сопоставленных метрологическому числу – результату умножения. */
   return (result); }
```

Операция деления

```
mdouble mdouble::operator /(mdouble& x, mdouble& y)
 { int i; mdouble result;
   if (!y.value)                              // проверка делителя на равенство нулю.
       printf("Divide by zero.");     // вывод ошибки о делении на ноль.
   result.value = x.value / y.value;          // значение самого результата деления.
   for (i = 0; i < n; i++)
       result.der[i] = (x.der[i] * y.value - y.der[i] * x.value) / (y.value* y.calue);
                  /* значения частных производных, сопоставленных метрологическому
                  числу – результату деления. */
   return (result); }
```

Операция умножения на коэффициент

```
mdouble mdouble::operator *(mdouble& x, double y)
 { int i; mdouble result;
   result.value = x.value * y;        // действительная часть результата умножения.
   for (i = 0; i < n; i++)
       result.der[i] = x.der[i] * y;  /* значения частных производных, сопоставленных
                  результату умножения на коэффициент. */
   return (result); }
```

тических действиях, повторяют правила взятия производных от суммы, разности, произведения и отношения двух дифференцируемых функций. Действительно, пусть, как и прежде, $x_1, x_2, \ldots, x_n$ – неточные входные данные, которые подвергаются математической обработке. Пусть y_1 и y_2 – два метрологических числа, полученных в результате промежуточных вычислений, и пусть y_3 – результат арифметической операции с ними. Тогда выполнены следующие общеизвестные соотношения между значениями частных производных $\partial y_3 / \partial x_i$, сопоставленных метрологическому числу y_3 и взятых по отношению к входным аргументам x_i (i = 1, 2, … n), и значениями соответствующих частных производных $\partial y_1 / \partial x_i$ и $\partial y_2 / \partial x_i$, сопоставленных операндам y_1 и y_2 (табл. 4).

Таблица 4. Правила заполнения элементов массива der при выполнении арифметических операций

операция	значение производной $\partial y_3/\partial x_i$
$y_3 = y_1 + y_2$	$\partial y_3/\partial x_i = \partial y_1/\partial x_i + \partial y_2/\partial x_i$
$y_3 = y_1 - y_2$	$\partial y_3/\partial x_i = \partial y_1/\partial x_i - \partial y_2/\partial x_i$
$y_3 = y_1 \cdot y_2$	$\partial y_3/\partial x_i = (\partial y_1/\partial x_i)\cdot y_2 + y_1\cdot(\partial y_2/\partial x_i)$
$y_3 = y_1 / y_2$	$\partial y_3/\partial x_i = ((\partial y_1/\partial x_i)\cdot y_2 - y_1\cdot(\partial y_2/\partial x_i)) / (y_2)^2$

Элементарные математические функции, представленные в стандартных библиотеках языков C/C++, также могут быть единообразно модифицированы для метрологических чисел, что иллюстрирует табл. 5 на нескольких примерах.

Представленный вариант реализации соответствует так называемой «прямой форме» метода автоматического дифференцирования. Кроме него, существует также «реверсивная форма», позволяющая получать значения производных быстрее, но с привлечением дополнительной памяти.

Реализованные правила позволяют надежно и достоверно (с точностью вплоть до ошибок округления в цифровых вычислительных машинах) получать в темпе основных штатных вычислений значения тех частных производных, что согласно формуле (1) необходимы для метрологической оценки погрешности результатов расчетов, унаследованной от неточных входных данных. Пример применения данных правил, иллюстрирующий пошаговое вычисление производных, представлен, например, в [13].

Приведенные в табл. 5 процедуры вычисления экспоненты, натурального логарифма и тригонометрического тангенса от метрологического числа соответствуют общеизвестным правилам вычисления производных от этих функций и правилу дифференцирования сложной функции:

$$\partial\exp(y_3)/\partial x_i = \exp(y_3)\times\partial y_3/\partial x_i,$$
$$\partial\ln(y_3)/\partial x_i = (1/y_3)\times\partial y_3/\partial x_i,$$
$$\partial\mathrm{tg}(y_3)/\partial x_i = (1+y_3^2)\times\partial y_3/\partial x_i.$$

Пусть, как и прежде, величины $x_1, x_2, \ldots, x_n$, представляющие входные аргументы расчетов, приняли в ходе измерений значения $x_{1\,0}, x_{2\,0}, \ldots, x_{n\,0}$. Пусть им сопоставлены пределы $\Delta_{\text{сист}\,i}$ и $\sigma^2_{\text{случ}\,i}$ допускаемых значений для систематической и и дисперсии случайной составляющих. Пусть $y=f(x_1, x_2, \ldots, x_n)$ – конечный результат расчетов. Тогда, получив по итогам вычислений с метрологическими числами значения произ-

Таблица 5. Пример элементарных математических функций от метрологических чисел

Функция вычисления экспоненты

```
mdouble exp(mdouble& x)
 { int i; mdouble result;
   result.value = exp(x.value); // вычисление экспоненты от результата измерения.
   for (i = 0; i < n; i++)
       result.der[i] = y.der[i] * result.value;      /* значения частных производных,
          сопоставленных результату взятию экспоненты от метрологического числа. */
   return (result); }
```

Функция вычисления натурального логарифма

```
mdouble log(mdouble& x)
 { int i; mdouble result;
   result.value = log(x.value); // вычисление логаримфма от результата измерения.
   for (i = 0; i < n; i++)
       result.der[i] = y.der[i] / x.value;    /* значения частных производных, сопоставленных
          результату взятию логарифма от метрологического числа. */
   return (result); }
```

Функция вычисления тригонометрического тангенса

```
mdouble tan(mdouble& x)
 { int i; mdouble result;
   result.value = tan(x.value); // вычисление тангенса от результата измерения.
   double derivative = 1.0f + result.value * result.value;
   for (i = 0; i < n; i++)
       result.der[i] = y.der[i] * derivative; /* значения частных производных, сопоставленных
     результату вычисления тангенса от метрологического числа. */
   return (result); }
```

водных $\partial f(x_{1\,0}, x_{2\,0}, \ldots, x_{n\,0}) / \partial x_i$, можем получить значения пределов $\Delta_{\text{сист } y}$ и $\sigma^2_{\text{случ } y}$ допускаемых значений систематической и дисперсии случайной составляющих погрешности величины y по выражениям, аналогичным формуле (1):

$$\Delta_{\text{сист } y} \approx \sum_{i=1}^{n} \left| \frac{\partial f\left(x_{1_0}, x_{2_0}, \ldots, x_{n_0}\right)}{\partial x_i} \right| \cdot \Delta_{\text{сист } i},$$

$$\sigma^2_{\text{случ } y} \approx \sum_{i=1}^{n} \left(\frac{\partial f\left(x_{1_0}, x_{2_0}, \ldots, x_{n_0}\right)}{\partial x_i} \right)^2 \cdot \sigma^2_{\text{случ } i}.$$

Видим, что все операции в приведенных соотношениях для характеристик погрешности величины y являются линейными. Следовательно, для типа данных `uncertainty`, введенного выше, необходимо определить только операции сложения, вычитания и умножения на коэффициент.

На основе данных операций для метрологического числа может быть задана процедура получения сопоставленной ему интервальной характеристики полной погрешности (процедура GetErr, представленная в табл. 7). Для ее реализации необходимо задать ряд арифметических правил для работы с типом данных `uncertainty`, выражающим характеристики погрешности, занесенные в метрологическое число (табл. 6).

В табл. 6 красными рамками выделены фрагменты кода, определяющие характеристику погрешности результата выполняемой арифметической операции.

Процедура GetErr позволяет вычислить оценку предельных значений погрешности результата совершенных вычислений, унаследованную им от неточных исходных данных. Для того, чтобы преобразовать предельное значение дисперсии случайной погрешности в границы интервала ее возможных значений, необходимо задать коэффициент охвата (обычно составляет 2÷3 для доверительной вероятности P = 0,90÷0,95) [15].

При помощи процедуры getUncertainty может быть получена характеристика погрешности интересующего пользователя результата вычислений.

Сформированный набор определений типов и основных процедур для метрологических чисел необходимо вынести в отдельную би-

Таблица 6. Пример введения арифметических операций с характеристиками погрешности в составе метрологических чисел

Операция сложения

```
uncertainty uncertainty::operator +(uncertainty& x, uncertainty& y)
 { uncertainty result;
   result.dispersion = x.dispersion + y.dispersion;
          // дисперсия случайной составляющей погрешности.
   result.systematic.a = x.systematic.left + y.sistematic.left;
   result.systematic.b = x.systematic.right + y.sistematic.right
          // пределы возможой систематической погрешности.
   return (result); }
```

Операция вычитания

```
uncertainty uncertainty::operator -(uncertainty& x, uncertainty& y)
 { uncertainty result;
   result.dispersion = x.dispersion - y.dispersion;
          /* дисперсия случайной составляющей погрешности. */
   result.systematic.left = x.systematic.left - y.sistematic.right;
   result.systematic.right = x.systematic.right - y.sistematic.left;
          /* пределы возможой систематической погрешности. */
   return (result); }
```

Операция умножения на коэффициент

```
uncertainty uncertainty::operator *(uncertainty& x, double y)
 { uncertainty result;
   result.dispersion = x.dispersion * (y * y);
          /* дисперсия случайной составляющей погрешности */
   double xa = x.systematic.left * y;                    // пределы возможой
   double xb = x.systematic.right * y;                   // систематической погрешности.
   result.systematic.left = (xa < xb) ? xa : xb;
   result.systematic.right = (xa < xb) ? xb : xa;
   return (result); }
```

Таблица 7. Пример введения процедуры GetErr оценки интервала полной погрешности результата вычислений с неточными исходными данными

Операция оценки интервала возможных значений полной погрешности

```
interval uncetainty::getInterval (double coeff)
  { /* coeff – коэффициент охвата, задаваемый пользователем. */
    interval result;
    if (coeff <= 0)
        printf("Коэффициент охвата должен быть положительным.");
                    // вывод ошибки о значении коэффициента охвата.
    double random = sqrt(this->dispersion) * coeff;
    result.left = this->systematic.left - random;   // левая граница интервала.
    result.right = this->systematic.right + random; // правая граница интервала.
    return (result);  }
```

Операция получения характеристики погрешности из обобщенного числа

```
uncertainty mdouble::getUncertainty (void)
 { int i; uncertainty Sum = 0;        /* Sum – переменная для накопления наследственной
погрешности результата вычислений. */
    for (i = 0; i < N; i++)          // оценка предела погрешности, унаследованной от
        Sum = Sum + this->der[i] * (*this->errors[i]);   // неточных исходных данных.
    return (Sum);  }
```

Операция GetErr

```
interval mdouble::GetErr (double coeff)
  {   /* coeff – коэффициент охвата, задаваемый пользователем. */
    uncertainty Error = this->getUncertainty();
            // получение характеристик погрешности из метрологического числа.
    interval result = Error.getInterval(coeff);
            // получение требуемой интервальной характеристики полной погрешности.
    return (result); }
```

блиотеку (например «mdouble.h»), включаемую пользователем в состав разрабатываемого им программного продукта для задач обработки результатов измерений. В этом случае включение метода метрологического сопровождения в состав штатной программы обработки результатов измерений оказывается совершенно простым [3], что может быть проиллюстрировано примером реализации простой программы в отсутствие метрологического сопровождения и при его применении. Такой пример представлен в табл. 8.

В правой части табл. 8 красным шрифтом выделены фрагменты кода, которые подверглись модификации при включении метрологического сопровождения. Изменения содержат: включение в программу библиотеки «mdouble.h», реализующей метрологическое сопровождение вычислений, изменение типа для исходных данных расчета и введение информации об их характеристиках

Таблица 8. Пример подключения библиотеки работы с метрологическими числами в код метрологически значимой программы

Исходный код	Модифицированный код

`#include<math.h>`
`#include<stdlib.h>`
 	
`void main (void)`
`{ double x, y;`
`  x = GetMeasuredResult();`
`  y = sin(x) / x;`
`  printf('%f ", y); }`
	`#include<math.h>`
	`#include<stdlib.h>`
	`#include "mdouble.h"`
	
	`void main (void)`
	`{ mdouble x(GetMeasuredResult(), 0.1, 0.2);`
	`  mdouble y;`
	`  y = sin(x) / x;`
	`  printf('%f ", z.value);`
	`  interval error = z.GetErr(3.0);`
	`  printf('[ %f, %f]', error.left, error.right); }`

погрешности. Содержательная часть программы, реализующая собственно саму математическую обработку (выделена в табл. 8 красными рамками) не претерпевает никаких изменений.

Последние две строки кода в правой части табл. 8 содержат вызов процедур, вычисляющих наследственную погрешность результата вычислений.

ЗАКЛЮЧЕНИЕ

Представленная работа посвящена программным средствам, которые позволяют обеспечить метрологическое сопровождение программ математической обработки, выполняемой в составе современных средств измерений. Это позволяет наделить такие программы важным свойством, заключающимся в том, что в темпе совершаемых ими расчетов вместе с каждым результатом вычислений пользователю сообщается надежная и достоверная характеристика погрешности данного результата. Подобное метрологическое сопровождение крайне необходимо для измерительных ситуаций, когда при метрологических процедурах аттестации затруднительно или вовсе невозможно при метрологических процедурах аттестации обеспечить поверку всего средства измерений целиком, не испытывая по отдельности его составные узлы, в особенности программное обеспечение. Такая ситуация в частности имеет место при измерениях количества потребленного тепла в жилищно-коммунальном хозяйстве, поскольку данная измеряемая величина может быть измерена только косвенно. В работе [2] представлен пример, показывающий, как использование метрологического сопровождения позволяет разрешить существенные затруднения в метрологическом обеспечении измерений количества теплоты, затраченной на отопление жилых домов.

В работе [16] отмечается, что только программы, дополненные метрологическим сопровождением, могут претендовать на то, чтобы быть метрологически аттестованы в полном смысле этого слова: предоставление каждому результату вычислений индивидуальной характеристики его погрешности позволяет создать объект для поверочных операций в ходе аттестации программного обеспечения.

Автор надеется, что данная статья поспособствует распространению чрезвычайно простого подхода, позволяющего совсем незначительными усилиями добиться столь существенных преимуществ и прогресса в задачах метрологического обеспечения, требующих привлечения программного обеспечения.

СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ 8.728-2010. ГСИ. Оценивание погрешностей измерений тепловой энергии и массы теплоносителя в водяных системах теплоснабжения. М.: Стандартинформ, 2010.
2. *Семенов К. К., Солопченко Г. Н.* Современное состояние метрологического обеспечения коммерческого учета тепловой энергии, потребляемой в жилом фонде // Приборы. 2016. № 6. С. 13-21
3. *Семенов К. К., Солопченко Г. Н.* Теоретические предпосылки метрологического автосопровождения программ обработки результатов измерений // Измерительная техника. 2010. № 6. С. 9-14.
4. *Moore R. E.* Interval arithmetic and automatic error analysis in digital computing: Ph. D. dissertation. Stanford University, 1962.
5. *Gustafson J. L.* The End of Error: Unum Computing. CRC Computational Science. Vol. 24. Chapman & Hall, 2015. 416 p.
6. *Reznik L. K.* Fuzzy controllers handbook. Newnes, 1997. 240p.
7. IEEE SA – 1788-2015. IEEE Standard for Interval Arithmetic. Enacted in 2015.
8. *Солопченко Г. Н.* Принципы метрологического автосопровождения вычислений в компьютерных измерительных информационных технологиях // Сборник докладов Международной конференции по мягким вычислениям и измерениям «SCM'99». Т. 1. СПб: СПбГЭТУ, 1999.
9. *Желнов Ю. А.* Точностные характеристики управляющих вычислительных машин. М.: Энергоатомиздат, 1983. 135 с.
10. *Семенов К. К.* Метрологическое автосопровождение программ вычислений в информационно-измерительных системах: дисс. … канд. техн. наук. СПб: СПбГПУ, 2011.
11. *Mari L.* A computational system for uncertainty propagation of measurement results // Measurement. 2009. Vol. 42. No 6. P. 844–855.
12. *Соболев В. С.* Метрологическое автосопровождение результатов измерений в интеллектуальных измерительных системах: дис. … докт. техн. наук. СПб: СПбГЭТУ, 1999.
13. *Семенов К. К.* Автоматическое дифференцирование функций, выраженных программным кодом // Известия вузов. Приборостроение. 2011. Т. 54. № 12. С. 34-40.
14. *Семенов К. К., Солопченко Г. Н.* Исследование комбинированного метода метрологического автосопровождения программ обработки результатов измерений // Измерительная техника. 2011. № 4. С. 14-19.
15. Why two sigma? A theoretical justification / *H. T. Nguen, V. Kreinovich, C.-W. Tao, G. N. Solopchenko* // Soft Computing in Measurement and Information Acquisition. (Ed. L. Reznik, V. Kreinovich). Berlin–Heidelberg: Springer-Verlag, 2003. P. 10–22.

16. *Semenov K. K., Reznik L. K., Solopchenko G. N.* Fuzzy Intervals Application for Software Metrological Certification in Measurement and Information Systems // International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 2015. Vol. 23. Suppl. 1. P. 95-104.

SOFTWARE TOOLS FOR INACCURATE DATA PROCESSING IN MEASURING INSTRUMENTS

Peter the Great St. Petersburg Polytechnic University

This work is dedicated to the presenting of the software tools that should be used during metrological software developing for the modern measuring systems and instruments. In the paper, the main principles are considered that allow supplementing the calculations with processed measurement results with the procedures for the estimation of the final computed results' error that is inherited from the inaccurate input data. On the base of the drawn conclusions, it is proposed to organize a software library that allows the developer to support his created software for the measuring instruments with the property of its metrological self-tracking. The paper contains the example of such library elements realization and its justification. The program tools that are based on the principles presented in this work bring to the achieving of the important properties for modern measuring instruments that use the computational devices for the measurement results obtaining.
Keywords: metrological software, metrological self-tracking, inherited error, transformed error.

Konstantin Semenov, Ph. D., Associate Professor at the Measuring Informational Technologies Department.
E-mail: semenov.k.k@iit.icc.spbstu.ru