Обратные задачи моделирования на основе регуляризации и распределенных вычислений в среде Everest

© А.П. Афанасьев © В.В. Волошинов © А.В. Соколов

Институт проблем передачи информации им. А.А. Харкевича РАН, Москва, Россия

alexander.afanasyev@gmail.com vladimir.voloshinov@gmail.com alexander.v.sokolov@gmail.com

Аннотация. Изложена методика оценки математических моделей физических явлений, происходящих в некоторой пространственной среде, на основе рядов экспериментальных данных. Целевая функция в обратной оптимизационной задаче идентификации параметров модели включает регуляризирующее слагаемое с неизвестными весовыми коэффициентами при вторых производных функций, описывающими исследуемое явление. Для выбора этих весовых коэффициентов применена процедура перекрестной (взаимной) верификации, когда часть исходных экспериментальных данных используется для «восстановления» остальных. Чем точнее результаты «взаимопроверки» для достаточно широкого набора проверочных тестов, тем «лучше» набор весовых коэффициентов. При вы-боре направления их улучшения требуется решить большой набор вспомогательных подзадач мате-матического программирования, для чего предложено использовать распределенную систему серви-сов оптимизации на платформе Everest.

Ключевые слова: обратные задачи, регуляризация, распределенные вычисления, REST-сервисы, платформа Everest.

Inverse Problem in the Modeling on the Basis of Regularization and Distributed Computing in the Everest Environment

© A.P. Afanasiev © V.V. Voloshinov © A.V. Sokolov

Institute for Information Transmission Problems RAS (Kharkevich institute), Moscow, Russia

alexander.afanasyev@gmail.com vladimir.voloshinov@gmail.com alexander.v.sokolov@gmail.com

Abstract. A method for estimating mathematical models of physical spatial phenomena is presented. Estimating is based on the series of experimental data. The objective function in the inverse optimization problem of identification of model parameters includes a regularizing term with unknown weight coefficients for the 2nd derivatives of the spatial function describing the phenomenon. Successive cross-validation procedure is used to choose values of weight coefficients. This cross-validation consists in approximation of one subset of experimental data by processing of a complementary subset. The better accuracy of the "cross-approximation", the better set of weight coefficients. Choosing direction of the possible improvement requires solving a number of subsidiary optimization problems. For that it is proposed to use distributed computing environment of optimization services deployed via Everest toolkit.

Keywords: inverse problems, regularized (ridge) approximation, distributed computing, Everest platform.

Введение

В работе предложен нестандартный подход к последовательному уточнению математических моделей, описывающих физические явления в некоторой пространственной среде, где для искомых зависимостей имеет смысл понятие «гладкости» по переменным, описывающим состояние модели. Речь может идти о физических процессах, которые моделируются дважды дифференцируемыми функциями на прямой, плоскости или в трехмерном пространстве.

Данный подход является развитием методов оптимизационной параметрической идентификации на основе экспериментальных данных с учетом возможных (и неизвестных) неточностей в этих данных. Чем шире доступный набор измерений (экспе-

Труды XIX Международной конференции «Аналитика и управление данными в областях с интенсивным использованием данных» (DAMDID/RCDL'2017), Москва, Россия, 10–13 октября 2017 года

риментальных данных) и выше их точность, тем более точную и подробную модель можно построить. Иногда недостаток количественной информации можно компенсировать дополнительными знаниями о закономерностях исследуемого процесса, уравнений его описывающих и т. д.

Аналогом является обработка статистических данных на основе сплайн-аппроксимации («сглаженных» кубических сплайнов), где коэффициент штрафа за интеграл квадрата 2-й производной аппроксимирующей зависимости играет роль параметра регуляризации [3, 4, 9, 12, 13]. Особенность предлагаемого подхода – минимизация суммы отклонения от экспериментальных данных и штрафа за «негладкость» при дополнительных ограничениях (соотношениях исследуемой модели).

По сути предлагается схема постепенного уточнения математической модели наблюдаемого физического явления с количественной оценкой качества такого уточнения. Если предсказательная точность новой модели повысилась, мы на верном пути.

1 Описание метода

Опишем общую схему метода на примере построения модели, описывающей исследуемое явление с помощью переменных *x*, *y*, *z*. Здесь *z* является измеряемой характеристикой, зависящей от *x* и *y*. Требуется построить модель в форме явной и неявной зависимостей между указанными переменными:

$$z = f(x, y),\ x \in X,\ y \in Y;\quad M(x, y, z) = 0. \quad (1)$$

Здесь *f(x,y)* – неизвестная функция, которую и требуется «восстановить» при дополнительных предположениях о «физике» наблюдаемого явления, *X, Y* – множества (интервалы) допустимых значений соответствующих переменных. Предполагаемая и подлежащая верификации физическая модель представлена вторым уравнением (1). Неявная зависимость *M* определяет дополнительные связи между переменными. Их может быть несколько, т. е. *M* – вектор-функция. Будем искать функцию *f(x,y)* либо в виде значений на достаточно «мелкой» сетке по переменным *x,y*, либо в классе функций, зависящих от параметров, значения которых и нужно определить, решив обратную задачу.

Пусть у исследователя имеется набор экспериментальных данных (измерений) следующего вида:

$$\{z_k, x_k, y_k\},\ k \in \mathrm{K},\ \mathrm{K} = 1,\ldots k_{\max}, \quad (2)$$

где значения z_k измерены с некоторыми неизвестными погрешностями. Задача восстановления функции *f* часто является некорректной и требует регуляризации. Будем искать зависимость *f(x,y)* (в параметризованном или «сеточном» виде) методом регуляризованной идентификации SvF, Smoothness-vs-Fitting, состоящем в поиске компромисса между точностью совпадения с экспериментальными данными и гладкостью искомой функции *f(x,y)*.

Для формализации такого подхода введем:

- характеристику совпадения с измерениями

$$\Delta_F\left(f(\cdot_{xy}), \mathrm{K}\right) = \frac{1}{|\mathrm{K}|}\sum_{k\in\mathrm{K}}\left(z_k - f(x_k, y_k)\right)^2; \quad (3)$$

- характеристику негладкости

$$\Delta_S\left(f(\cdot_{xy}), \beta_x, \beta_y\right) =$$
$$\beta_x^2 \iint\limits_{X\,Y}\left(\frac{\partial^2 f}{\partial x^2}\right)^2 \mathrm{d}x\mathrm{d}y + 2\beta_x\beta_y \iint\limits_{X\,Y}\left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 \mathrm{d}x\mathrm{d}y + \quad (4)$$
$$\beta_y^2 \iint\limits_{X\,Y}\left(\frac{\partial^2 f}{\partial y^2}\right)^2 \mathrm{d}x\mathrm{d}y$$

(для сеточной функции вторые производные заменяются разностными выражениями);

- компромиссный критерий (функционал Тихонова [10]), зависящий от параметров β_x, β_y и множества измерений, характеризуемого набором индексов K:

$$\Delta\left(f(\cdot_{xy}), \beta_x, \beta_y, \mathrm{K}\right) = \Delta_F\left(f(\cdot_{xy}), \mathrm{K}\right) + \Delta_S\left(f(\cdot_{xy}), \beta_x, \beta_y\right). \quad (5)$$

Будем искать функцию *f(x,y)* в классе дважды непрерывно-дифференцируемых функций, решая следующую задачу минимизации, где переменными являются либо значения функции «на сетке» по переменным *x,y*, либо параметры *f(x,y)*:

$$f^*(\cdot_{xy}) = \underset{f(\cdot_{xy})}{Arg\,\min}\left\{\begin{array}{l}\Delta\left(f(\cdot_{xy}), \beta_x, \beta_y, \mathrm{K}\right):\\ M\left(x, y, f(x, y)\right) = 0,\\ x \in X, y \in Y.\end{array}\right\} \quad (6)$$

Решение задачи (6) для различных значений β_x, β_y дает функции *f(x,y)*, соответствующие различным соотношениям между «гладкостью» и «точностью» (восстановления экспериментальных данных). Для поиска «разумного» баланса, выраженного в значениях β_x, β_y, воспользуемся процедурой перекрестной верификации [3, 4]. Для этого разобьем множество индексов экспериментальных данных на набор непересекающихся подмножеств

$$\mathrm{K} = \bigcup_{i\in I}\mathrm{K}_i,\ \mathrm{K_i} \cap \mathrm{K_j} = \varnothing,\ i \neq j. \quad (7)$$

Уберем из множества K одно из подмножеств K_i. Решим задачу минимизации (6) на оставшемся наборе данных $\mathrm{K}\setminus\mathrm{K}_i$. Пусть ее решение $f^*_{\mathrm{K}_i}(x, y)$:

$$f^*_{\mathrm{K}_i}(\cdot_{xy}) = \underset{f(\cdot_{xy})}{Arg\,\min}\left\{\begin{array}{l}\Delta\left(f(\cdot_{xy}), \beta_x, \beta_y, \mathrm{K}\setminus\mathrm{K}_i\right):\\ M\left(x, y, f(x, y)\right) = 0,\\ x \in X, y \in Y.\end{array}\right\} \quad (8)$$

Определим отклонение $f^*_{\mathrm{K}_i}(x, y)$ от измерений для $k \in \mathrm{K}_i$ по формуле $\sum_{k\in\mathrm{K_i}}\left(z_k - f^*_{\mathrm{K}_i}(x_k, y_k)\right)^2$. Повторив эту процедуру для всех подмножеств K_i, $i \in I$, получим «перекрестную» оценку точности модели для заданных β_x, β_y:

$$\Phi(\beta_x,\beta_y)=\left(\sigma_{\mathrm{SvF}}(\beta_x,\beta_y)\right)^2=\frac{1}{|\mathrm{K}|}\sum_{i\in\mathrm{I}}\sum_{k\in\mathrm{K}_i}\left(z_k-f^*_{\mathrm{K}_i}(x_k,y_k)\right)^2. \quad (9)$$

Для получения минимальной погрешности моделировании и выбора «оптимального» соотношения близость–сложность будем искать β_x, β_y, которые минимизируют величину $\sigma_{\mathrm{SvF}}(\beta_x,\beta_y)$. Для найденных в результате минимизации значений β_x, β_y искомая функция $f^*(x,y)$ определяется в результате решения основной задачи (6). Итоговая погрешность аппроксимации (невязка) определяется по формуле

$$\left(\sigma^*\right)^2=\frac{1}{|\mathrm{K}|}\sum_{k\in\mathrm{K}}\left(z_k-f^*(x_k,y_k)\right)^2. \quad (10)$$

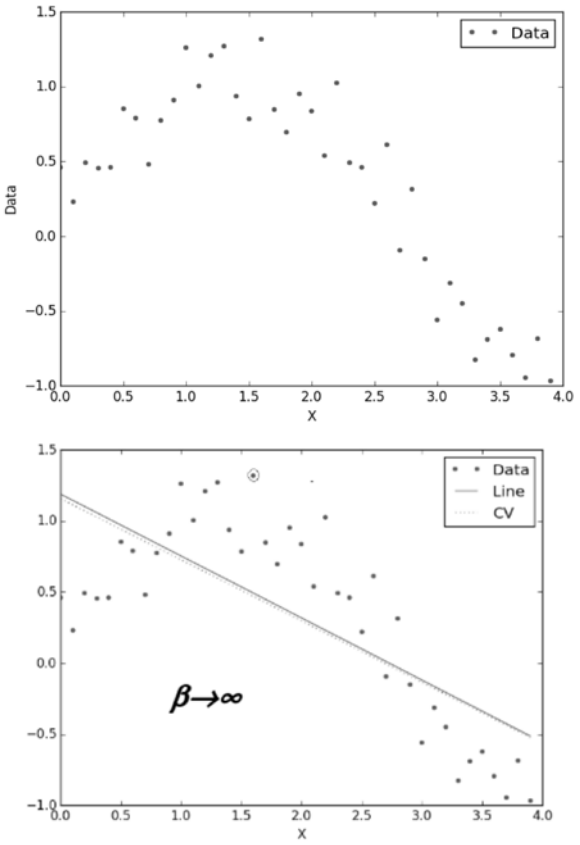

Рисунок 1a Исходные данные и результат метода наименьших квадратов $\beta\to\infty$

Таким образом, процедура расчетов соответствует двухуровневой задаче оптимизации:

$$\Phi(\beta_x,\beta_y)=\frac{1}{|\mathrm{K}|}\sum_{i\in\mathrm{I}}\sum_{k\in\mathrm{K}_i}\left(z_k-f^*_{\mathrm{K}_i}(x_k,y_k)\right)^2\to\min_{\beta_x,\beta_y\ge 0}, \quad (11)$$

где функции $f^*_{\mathrm{K}_i}(x,y)$ определяются как решения независимых задач оптимизации (8).

Результат метода SvF находится, в некотором смысле, «между» результатами применения хорошо известных методов наименьших квадратов и кубической сплайн-интерполяции. На Рис. 1 метод SvF продемонстрирован на примере восстановления функции одного переменного по набору исходных данных (Рис. 1a) при отсутствии дополнительных модельных соотношений (формально можно положить $\boldsymbol{M}(x,y,z)$ тождественно равной нулю). Для функции одной переменной x требуется лишь один скалярный параметр β. При $\beta\to 0$ задача (6) становится задачей сплайн-интерполяции, решением которой является т. н. кубический сплайн, т. е. функция, имеющая минимальный (см. (4)) интеграл квадрата 2-ой производной и проходящая через все «экспериментальные» точки (2), Рис. 1b).

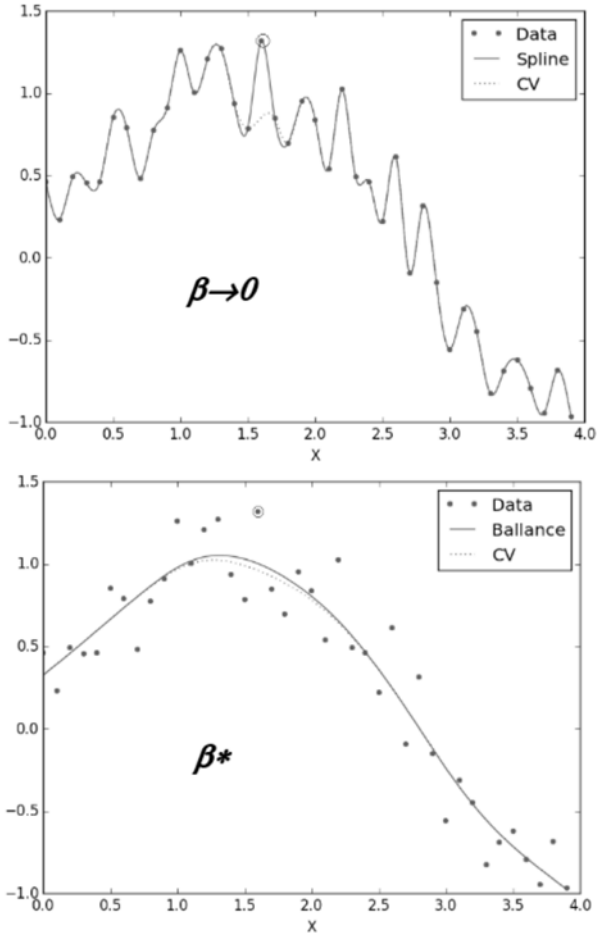

Рисунок 1b Кубический сплайн, $\beta\to 0$, и значение β^*, полученное методом SvF

При $\beta\to\infty$ мы получим линейную функцию, минимизирующую сумму квадратов отклонения от исходных данных, Рис. 1a. «Компромиссный» результат сплайн-аппроксимации со штрафом (за негладкость), определенным методом SvF, представлен на Рис. 1b снизу.

Опишем процедуру решения задачи верхнего уровня (11). Введем обозначение $P(\boldsymbol{a},\boldsymbol{\beta})$ для произвольного многочлена 2-го порядка вектора переменных $\boldsymbol{\beta}$, зависящего от вектора коэффициентов $\boldsymbol{a}$

$$P(\mathbf{a},\boldsymbol{\beta})=a_{xx}\beta_x^2+a_{xy}\beta_x\beta_y+a_{yy}\beta_y^2+a_x\beta_x+a_y\beta_y+a_0, \quad (12)$$

где $\mathbf{a}=(a_{xx},a_{xy},a_{yy},a_x,a_0),\boldsymbol{\beta}=(\beta_x,\beta_y)$.

Далее алгоритм строит последовательность значений $\boldsymbol{\beta}^{\nu}$, ν=1,2,… . Пусть, на N-ом шаге построены значения $\boldsymbol{\beta}^{\nu}$, ν=1:N, для которых вычислены значения $\Phi^{\nu}=\Phi(\boldsymbol{\beta}^{\nu})$. Без ограничения общности (возможно, после перенумерации) можно считать, что

Φ^N – минимальное из полученных значений. Будем трактовать $\{\Phi^\nu, \boldsymbol{\beta}^\nu\}_{\nu=1}^N$ как набор точек в R^3. Рассмотрим задачу аппроксимации этих точек графиком многочлена 2-го порядка (12), причем чем вектор $\boldsymbol{\beta}^\nu$ ближе к «наилучшему» $\boldsymbol{\beta}^N$, тем с большим весом он будет учитываться. Кроме того, будем штрафовать за кривизну построенной функции с коэффициентом μ (подлежащим выбору):

$$\sum_{\nu=1:N} e^{-\|\boldsymbol{\beta}^\nu - \boldsymbol{\beta}^N\|}\left(\Phi^\nu - P(\mathbf{a}, \boldsymbol{\beta}^\nu)\right)^2 + \mu\left(a_{xx}^2 + a_{xy}^2 + a_{yy}^2\right) \to \min_{\mathbf{a}}. \quad (13)$$

Пусть $\mathbf{a}^*(\mu)$ оптимальное значение вектора переменных в этой задаче выпуклого программирования.

Выберем значение μ, минимизируя отклонение значения аппроксимирующего полинома от наилучшего значения Φ^N. Тем самым для аппроксимации получим вновь двухуровневую задачу:

$$\left|P\left(\mathbf{a}^*(\mu), \boldsymbol{\beta}^N\right) - \Phi^N\right| \to \min_{\mu \geq 0}, \quad (14)$$

где $\mathbf{a}^*(\mu)$ – решение задачи «нижнего уровня» (13). Здесь в задаче верхнего уровня нужно выбрать единственную переменную μ, а задача нижнего уровня эффективно разрешима благодаря ее выпуклости. Поэтому для поиска оптимального штрафного коэффициента μ^* применим тот или иной алгоритм минимизации функции одного переменного.

После аппроксимации зависимости $\Phi(\boldsymbol{\beta})$ квадратичной функцией $P\left(\mathbf{a}^*(\mu^*), \boldsymbol{\beta}\right)$ новое значение вектора $\boldsymbol{\beta}$ находится в результате решения следующей вспомогательной задачи, которая напоминает метод линеаризации Пшеничного–Данилина [11], применяемый к минимизации функции $P\left(\mathbf{a}^*(\mu^*), \boldsymbol{\beta}\right)$ по $\boldsymbol{\beta}$:

$$\left(\nabla_{\boldsymbol{\beta}} P\left(\mathbf{a}^*(\mu^*), \boldsymbol{\beta}^N\right)\right)^{\mathrm{T}}\left(\boldsymbol{\beta} - \boldsymbol{\beta}^N\right) + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^N\right\|^2 \to \min_{\beta \geq 0},$$

где $\nabla_{\boldsymbol{\beta}} P(\ldots)$ обозначает градиент многочлена (12) по переменным β_x, β_y. Заметим, что решение этой выпуклой задачи квадратичного программирования существует и единственно. Вычислим значение $\Phi^{N+1} = \Phi\left(\boldsymbol{\beta}^{N+1}\right)$, решив набор задач нижнего уровня (8). Если величина $\left|\Phi\left(\boldsymbol{\beta}^{N+1}\right) - \Phi\left(\boldsymbol{\beta}^N\right)\right|$ меньше некоторого порогового значения, работа алгоритма прекращается. Если это не так, то схема расчетов повторяется для расширенного набора $\{\Phi^\nu, \boldsymbol{\beta}^\nu\}_{\nu=1}^{N+1}$.

3 Демонстрация метода при моделировании распространения тепла в высокотемпературной плазме

Физические явления в горячей плазме, удерживаемой сильным магнитным полем в тороидальных вакуумных камерах установок термоядерного синтеза, таких, как ТОКАМАКи и стеллараторы, важны для перспектив термоядерной энергетики. Например, в экспериментах наблюдается «быстрый нелокальный перенос тепла» – практически мгновенное (по сравнению с «классической» тепловой диффузией) повышение температуры в центре плазменного шнура после охлаждения его периферии или обратный процесс – мгновенное понижение температуры в центре при быстром нагреве периферии плазмы.

Здесь мы не будем обсуждать неясную физику этого явления. Ограничимся демонстрацией применения метода к предварительной обработке экспериментальных данных из статьи [8], где обсуждается явление «быстрого» охлаждения плазмы в стеллараторе LHD, http://www.lhd.nifs.ac.jp/en

Исходными данными являются графики зависимостей температуры плазмы от расстояния (ρ) до центра тороидальной камеры в различные моменты времени (t), Рис. 2.

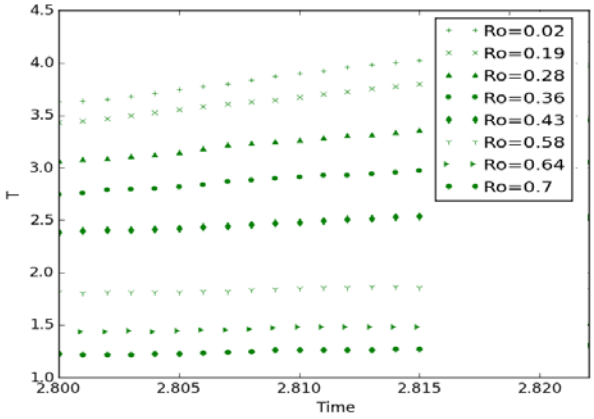

Рисунок 2 Температура на разных расстояниях от центра тороидальной камеры $T(\rho,t)$, [8]

Множеством $\mathbb{K}$ экспериментальных данных о температуре, см. (2), является набор пар индексов i_ρ, i_t, значений расстояния и моментов времени. Разделим множество измерений температуры на 8 частей, определяемых значениями ρ (горизонтальные группы точек на Рис. 2). Для перекрестной верификации будем использовать 6 множеств (см. (7)):

$$\mathrm{K_i} = \left\{\left(i_\rho, i_t\right): i_t \in I_t\right\},\ i_\rho = 2:7. \quad (15)$$

Крайние значения (ρ_1=0.02 и ρ_9=1) в перекрестном оценивании не учитывались, т. к. в этой задаче важна интерполяция значений температуры плазмы.

Ниже приведен ряд моделей наблюдаемого явления вида (1). Они отличаются друг от друга предположениями о процессе распространения тепла в плазме в форме второй группы соотношений (1). Цель – выбрать модель, которая бы максимально точно описывала функцию $T(\rho,t)$ – зависимость температуры плазменного пучка от ρ и t. Можно ожидать, что при «правильных» уточнениях *предсказательная точность* моделирования (по отклонению функции $T(\rho,t)$ от значений на Рис. 2) должна улучшиться. Результаты расчетов по всем моделям приведены ниже в Таблице 1. Графики, иллюстрирующие расчеты по моделям, доступны на странице http://distcomp.ru/~vladimirv/damdid2017.

3.1 Модель 1 (сплайн-интерполяция без SvF)

Пусть у нас нет никаких предположений о физических причинах наблюдаемого явления. Будем искать функцию $T(\rho,t)$, проходящую через все точки используемого набора измерений и имеющую минимальную кривизну в смысле формулы (4), а точность определять процедурой перекрестной верификации на подмножествах (15). Формально этот прием соответствует случаю, $\beta \to 0$, Рис. 1b.

Получили задачу сплайн-интерполяции. При достаточно «необременительных» предположениях о расположении узлов интерполяции её решение существует и единственно [13].

3.2 Модель 2 (Сплайн-аппроксимация)

Не имея, как и выше, никаких предположений о «физике» явления, применим метод SvF, когда функция $M(x,y,z)$ тождественно равна нулю (см. (1) и Рис. 1b, снизу). Эта задача является задачей сплайн-аппроксимации с выбором штрафа за негладкость на основе перекрестного оценивания. Как и для Модели 1, её решение существует и единственно [13]. Несмотря на то, что построенная функция температуры уже не проходит через узлы, оценка точности заметно улучшилась (см. Табл. 1).

3.3 Модель 3 (Метод SvF и простое дифференциальное уравнение)

Предположим, что процесс описывается простейшим дифференциальным уравнением:

$$\frac{\partial T}{\partial t}(\rho,t) = \pi(\rho,t) \tag{16}$$

Здесь неизвестными являются две функции $T(\rho,t)$ и $\pi(\rho,t)$ от двух переменных. Поэтому в схеме **SvF** проводится оптимизация по четырем коэффициентам штрафа «за негладкость» обеих функций: $\beta_{T\rho}$, β_{Tt} и $\beta_{\pi\rho}$, $\beta_{\pi,t}$. Поскольку для неизвестной функции $\pi(\rho,t)$, правой части дифференциального уравнения (16), нет экспериментальных данных, то основной компромиссный «критерий» (5) имеет вид

$$\begin{aligned}\Delta\left(\left\{T(\cdot_{\rho t}), \pi(\cdot_{\rho t})\right\}, \beta_{T\rho}, \beta_{Tt}, \beta_{\pi\rho}, \beta_{\pi t}, \mathrm{K}\right) = \\ \Delta_F\left(T(\cdot_{\rho t}), \mathrm{K}\right) + \Delta_S\left(T(\cdot_{\rho t}), \beta_{T\rho}, \beta_{Tt}\right) + \\ \Delta_S\left(\pi(\cdot_{\rho t}), \beta_{\pi\rho}, \beta_{\pi t}\right)\end{aligned} \tag{17}$$

Сравнение с предыдущей моделью (см. Табл. 1) не выявило принципиальных изменений.

3.4 Модель 4 (Метод SvF и тепловая диффузия)

Здесь предполагается, что распространение тепла описывается классическим дифференциальным уравнением тепловой диффузии в цилиндрически симметричной среде (как приближении тороидальной камеры) с заранее неизвестным коэффициентом «теплопроводности» χ.

$$\frac{\partial T}{\partial t}(\rho,t) = \chi \frac{\partial}{\rho \partial \rho}\left(\rho \frac{\partial}{\partial \rho}\left(T(\rho,t) - T_0(\rho)\right)\right), \tag{18}$$

где $T_0(\rho)$ – функция температуры пучка в начальный момент времени предполагается известной. В такой постановке критерий (5) имеет вид (17), но, кроме коэффициентов $\beta_{T\rho}$, β_{Tt}, $\beta_{\pi\rho}$ и $\beta_{\pi t}$, минимизация проводится еще по переменной χ. Результаты моделирования показывают, что учет только диффузионного члена (без дополнительного «источника») не является удачным: точность моделирования падает.

3.5 Модель 5 (Метод SvF, тепловая диффузия и «неизвестный источник»)

Предположим, что наряду с «медленной» тепловой диффузией в плазме действует некоторый дополнительный механизм переноса тепла, который в следующей формуле обозначен $S(\rho,t)$:

$$\frac{\partial T}{\partial t}(\rho,t) = \chi \frac{\partial}{\rho \partial \rho}\left(\rho \frac{\partial}{\partial \rho}\left(T(\rho,t) - T_0(\rho)\right)\right) + S(\rho,t). \tag{19}$$

Здесь функция критерия (5) имеет вид, аналогичный (17), но зависит от четырех β-коэффициентов для функций $T(\rho,t)$ и $S(\rho,t)$:

$$\begin{aligned}\Delta\left(\left\{T(\cdot_{\rho t}), S(\cdot_{\rho t})\right\}, \beta_{T\rho}, \beta_{Tt}, \beta_{S\rho}, \beta_{St}, \mathrm{K}\right) = \\ = \Delta_F\left(T(\cdot_{\rho t}), \mathrm{K}\right) + \Delta_S\left(T(\cdot_{\rho t}), \beta_{T\rho}, \beta_{Tt}\right) + \\ + \Delta_S\left(S(\cdot_{\rho t}), \beta_{S\rho}, \beta_{St}\right).\end{aligned} \tag{20}$$

В итоге определяются коэффициенты $\beta_{T,\rho}$, $\beta_{T,t}$, $\beta_{S,\rho}$ и $\beta_{S,t}$, и коэффициент «теплопроводности» χ. Точность моделирования заметно возросла (Табл. 1).

3.6 Сравнение результатов расчетов

Как уже было объявлено, результаты расчетов сведены в Таблицу 1. Графические изображения построенных зависимостей от ρ и t приведены на рисунках в http://distcomp.ru/~vladimirv/damdid2017.

Таблица 1 Результаты моделирования

Модель	Погрешность	
	Кросс-валидации, %	Аппроксимации, СКО[1], %
1[2]	11.94	0
2	9.25	3.55
3	9.19	3.55
4	10.00	8.53
5 (χ=0.21)	1.85	0.59

Заметим, что переход от «простых» сплайнов к более содержательным моделям 2–4 дал незначительное улучшение точности перекрестной проверки. Но расчет по Модели 5 дал многократное улучшение показателей точности, т. е. выделение неизвестного «источника» (переносчика тепловой энергии) является, по-видимому, верным уточнением модели. Приведенный пример демонстрирует применение метода для количественной проверки «качества» различных математических моделей на имеющемся наборе экспериментальных данных.

[1] Среднеквадратичное отклонение

[2] Сплайн-интерполяция

4 Возможности программной реализации в среде Everest

Предлагаемая методика основана на решении задач математического программирования. Для ее практического применения нужен набор программных инструментов для: 1) описания указанных задач; 2) формирования структур данных, соответствующих отдельным экземплярам таких задач; 3) отправки этих данных пакетам численных методов (решателям), для поиска решения; 4) обработки результатов работы решателей, например, для изменения метода расчетов. Сложившаяся практика применения оптимизационных моделей предусматривает два способа организации расчетов.

Первый, «низкоуровневый», на основе открытого программного интерфейса (API) решателя для некоторого языка программирования (C/C++, C#, Java, Python и т. п.). Подготовка данных для отправки решателю и обработка результатов производятся обычно на языке API решателя. Такой подход, хотя и может привести к созданию высокопроизводительной системы расчетов, является достаточно трудоемким и требует привлечения высококвалифицированных программистов. Кроме того, изменения в схеме расчетов или структуре применяемой оптимизационной модели требуют переписывания значительных фрагментов программного кода.

Второй, «высокоуровневый», подход использует алгебраические (декларативные) языки оптимизационного моделирования, что гораздо удобнее, особенно для поисковых исследований. Развитие таких языков (AML, Algebraic Modelling Language – в англоязычной литературе) ведется уже более 30 лет. До настоящего времени наиболее популярными являются AMPL, GAMS. Основными составляющими AML-систем являются: собственно язык для описания оптимизационных моделей, средства автоматического дифференцирования и унифицированный интерфейс взаимодействия с пакетами.

AML-языки позволяют записать соотношения оптимизационных задач (параметры, переменные, целевую функцию, ограничения, индексы параметров, переменных и ограничений и т. п.), разделив «символьное описание» задачи (т. н. *модельное представление*) и «конкретные данные» (наборы индексов, значения числовых параметров и т. п.). Символьная модель и конкретные данные, представленные обычно текстовыми файлами, обрабатываются специальным транслятором. На выходе – специальная структура данных в виде т. н. стаб-файла (stub, в терминологии AMPL), готового для передачи решателям, совместимым с языком моделирования. Для нелинейных задач стаб также содержит правила вычисления первых и вторых производных всех функций задачи математического программирования. Если численный метод находит решение, то AML-совместимый решатель возвращает файл, содержащий значения всех «прямых» и двойственных переменных задачи (множителей Лагранжа при ограничениях). Формат этого файла соответствуют стандарту применяемого AML, и AML-транслятор может его импортировать.

Также эти языки позволяют описывать сложносоставные сценарии расчетов по оптимизационным моделям: содержащие условные переходы, циклы, динамическое формирование новых задач на основе результатов решения предыдущих, создание наборов задач для различных наборов значений параметров и т. п. Являясь по назначению языками программирования высокого уровня, AMPL и GAMS не отвечают требованиям, предъявляемым даже к процедурным языкам (надо иметь ввиду «почтенный возраст» языков, AMPL и GAMS появились в конце 1970-х годов). Например, в них нет понятия процедуры-функции, все переменные (кроме внутренних индексов циклов или операторов «итерирования») являются глобальными и т. п.

В связи с этим большой интерес вызывает система оптимизационного моделирования Pyomo (Python Optimization Modeling Objects) [2] http://pyomo.org, основанная на популярном объектно-ориентированном языке программирования Python (Pyomo представляет собой специализированный Python-пакет). Четыре года лет назад Pyomo стал совместимым со стандартом AMPL. Это произошло благодаря тому, что авторы AMPL 12 лет назад «раскрыли» внутренний формат AMPL-стаба.

Принцип расчетов в системе Pyomo повторяет схему применения языка AMPL: модель (в форме набора Python-объектов) вместе с исходными данными (либо в виде Python-объектов, либо в формате файлов с данными AMPL-формате) преобразуются в AMPL-стаб, передаваемый AMPL-решателю. Файл с решением можно считать специальной процедурой пакета Pyomo, для оформления результата и/или подготовки исходных данных новых задач математического программирования.

4.1 Сведения о платформе Everest

Разработка программного обеспечения для создания систем на основе REST-сервисов ведутся в нашем коллективе около шести лет. Первоначальная и последующая стабильная версии ПО имели название MathCloud, mathcloud.org. Последние три года велась активная работа по переходу на новую версию программного инструментария, т. н. Everest [6, 7], http://everest.distcomp.org.

Программный инструментарий Everest является системой с открытым кодом, свободно доступным на популярном портале gitlab.com. Семантика Everest базируется на следующей иерархии понятий:

- *Приложение Everest* – REST-сервис с REST-интерфейсом в формате JSON; приложение Everest, вообще говоря, является абстракцией, для которой не выделено никакого реального вычислительного ресурса (его выбор и подключение происходят непосредственно перед вызовом);
- *Вычислительный ресурс Everest* – реальное вычислительное устройство или инфраструктура

(сервер, кластер, грид, облако), где производится обработка данных Everest-приложениями;

- *Агент доступа к вычислительным ресурсам Everest* – программный модуль (на Python), обеспечивающий подключение ресурса к системе Everest (некоторые основные типы ресурсов представлены на Рис. 3);
- *сервер Everest* (контейнер приложений) – центральный сервер для: сохранения дескрипторов всех приложений; регистрации пользователей; управления правами доступа к созданным приложениям; управление очередями заданий;
- веб-интерфейс работы с сервером Everest, включающий средства создания приложений, запуска и контроля за ходом выполнения заданий.

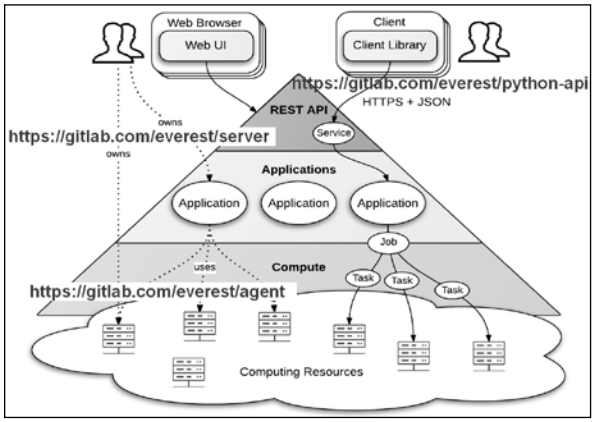

Рисунок 3 Архитектура программного инструментария Everest

Перечислим ряд особенностей Everest, важных с точки зрения практического развертывания и применения систем оптимизационного моделирования в распределенной вычислительной инфраструктуре.

1. Автор приложения может «незаметно» для пользователей повышать/понижать вычислительную «мощность» приложений (сервисов), изменив список ресурсов (фактически агентов доступа к ресурсам), ассоциированных с данным приложением. Например, если комплект решателей будет установлен на новом вычислительном сервере вместе с агентом Everest, то производительность Everest-приложения для решения задач оптимизации повысится.

2. Платформа Everest предлагает унифицированный программный интерфейс (Everest Python API), gitlab.com/everest/python-api. Он позволяет клиентским модулям на Python взаимодействовать с сервисами Everest по модели асинхронных вызовов удаленных объектов и программировать вычислительные сценарии координированной обработки данных несколькими приложениями Everest. При этом независимые задания будут выполняться одновременно несколькими приложениями (или одним приложением, но на разных вычислительных ресурсах, подключенных к этому приложению). Задача балансировки вычислительной нагрузки между ресурсами возложена на сервер Everest.

3. Подсистема балансировки вычислительной нагрузки управляет выполнением заданий, распределяя их между вычислительными ресурсами, подключенными к одному приложению. Пользователь может не знать, какие именно ресурсы обрабатывают его данные. Эта подсистема постоянно совершенствуется разработчиками Everest, в частности, ожидается возможность выбора различных политик распределения заданий между ресурсами.

4. Система контроля доступа к приложениям и защиты данных в Everest использует две технологии: защищенный обмен данными между Everest-сервером и агентами по протоколу HTTPS/SSH; специальные «временные» ключи, т. н. токены, выдаваемые зарегистрированным пользователям Everest с ограниченным сроком действия (7 дней). Предъявление токенов обязательно и для работы с веб-интерфейсом сервера, и при вызове приложений через Everest API.

4.2 Сервис оптимизации

Базовым сервисом решения задач оптимизации в Everest является сервис solve-ampl-stub решения задач математического программирования, представленных в виде AMPL-стаб-файла [1], пакетом численных методов (решателем), указанным при обращении к сервису. Сейчас сервис обеспечивает унифицированный доступ к следующим пакетам, позволяющим решать основные типы задач математического программирования (LP/MILP/NLP/MINLP):

- **Ipopt** (Coin-OR **I**nterior **P**oint **Opt**imizer, NLP), https://projects.coin-or.org/Ipopt;
- **CBC** (**C**oin-OR **B**ranch-and-**C**ut, LP, MILP), https://projects.coin-or.org/Cbc;
- **SCIP** (**S**olving **C**onstraint **I**nteger **P**rograms, LP, MILP, MINLP (билинейные невыпуклые)), http://scip.zib.de
- **Bonmin** (COIN-OR **B**asic **O**pen-source **N**onlinear (convex) **M**ixed **In**teger programming, MINLP), https://projects.coin-or.org/Bonmin

Данным приложением можно пользоваться как через его веб-интерфейс, так и через программный интерфейс Everest Python API.

4.3 Сведения о программной архитектуре Pyomo-Everest

Основным требованием к системе было: обеспечить возможность выполнения любых программ (сценариев расчетов) на языке Python с использованием пакета Pyomo и AMPL-совместимых решателей в среде сервисов оптимизации Everest, возможно, после некоторой модификации самой программы согласно определенному набору правил. Общий принцип работы PyomoEverest (https://github.com/distcomp/pyomo-everest) аналогичен разработанной нами ранее системе AMPLx [5].

PyomoEverest состоит из двух элементов: 1) модуля на Python, который посредством Everest Python API обеспечивает взаимодействие с сервисом solve-

ampl-stub (см. выше); 2) пула вычислительных ресурсов, подключенных к сервису solve-ampl-stub посредством агентов доступа Everest.

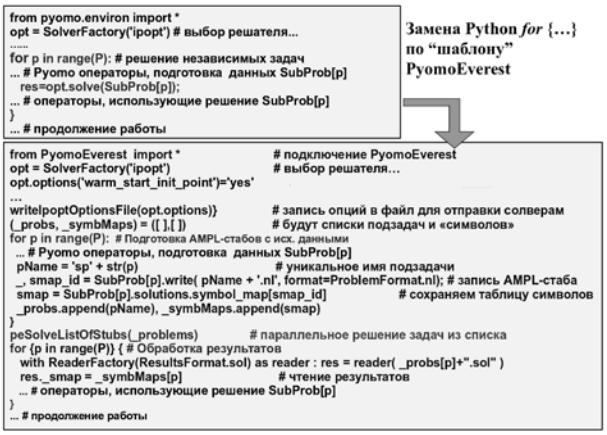

Рисунок 4 Модификация Python/Pyomo кода по «шаблону» PyomoEverest

Типовой прием «распараллеливания» фрагмента алгоритма расчетов, записанного на Pyomo, представлен на Рис. 4. Вверху, в рамке, приведены фрагменты кода для выбора решателя и цикла `for`, где решается набор независимых подзадач. Внизу находится фрагмент модифицированного кода. Правило модификации в том, чтобы *каждый цикл* `for` или `while` нужно заменить тремя группами операторов:

1. цикл формирования набора подзадач в форме AMPL-стабов;
2. параллельное решение задач, представленных своими стаб-файлами, приложением Everest (сейчас – solve-ampl-stub) с подключенным к нему пулом AMPL-совместимых решателей;
3. цикл обработки результатов, доставленных в виде набора файлов *.sol с решениями подзадач.

Модифицированный фрагмент приведен в нижней рамке. Здесь важно использование Python-класса pyomo.core.base.SymbolMap. Экземпляр этого класса (структура symbol_map) создается при записи стаб-файла подзадачи в первом цикле, сохраняется (здесь – в массиве _symbMaps) и применяется во втором цикле при чтении решений из *.sol-файлов для корректного сопоставления их содержимого структуре соответствующей оптимизационной подзадачи.

Заключение

Приведенные результаты моделирования динамики температуры плазмы подтверждают эффективность предложенного метода «гладкой» регуляризации для обработки экспериментальных данных.

Практическое применение метода основано на «перекрестной» (взаимной) верификации – процедуре определения «пропущенной» части экспериментальных данных по остальным измерениям. Это известный байесовский подход к «настройке» параметров некоторой регрессии по статистическим данным [3, 4]. Поскольку здесь требуется решать наборы независимых задач математического программирования, то работа алгоритма может быть ускорена за счет одновременного решения указанных задач пулом решателей, установленных в распределенной вычислительной среде.

Эта вычислительная схема характеризуется периодическим решением относительно небольшого набора (десятки) независимых и относительно сложных задач математического программирования (время решения каждой современным решателем на современном сервере – не менее пары минут). Для ее реализации в режиме распределенных вычислений предлагается использовать систему Pyomo-Everest. Решение всех задач выполняется сервисом оптимизации Everest, причем независимые подзадачи могут решаться параллельно под управлением службы балансировки вычислительной нагрузки Everest на ресурсы, подключенных к сервису.

Предложенный подход указывает перспективное направление применения распределенных систем на основе высокоуровневых средств оптимизационного моделирования.

Благодарности

Работа поддержана Российским научным фондом (грант № 16-11-10352).

Литература

[1] Fourer, R., Gay, D. M., Kernighan, B. W.: AMPL: A Modeling Language for Mathematical Programming, 2nd edition.: Duxbury Press (2002)

[2] Hart, W. E., Laird, C., Watson, J.-P., Woodruff, D. L.: Pyomo-optimization modeling in python, 67, 238 p. Springer (2012)

[3] Hastie, T. J., Tibshirani, R. J.: Generalized additive models, 43, CRC Press (1990)

[4] Hastie, T. J., Tibshirani, R. J., Friedman, J.: Unsupervised Learning. The Elements of Statistical Learning: Springer, pp. 485-585 (2009)

[5] Smirnov, S., Voloshinov, V., Sukhosroslov, O.: Distributed Optimization on the Base of AMPL Modeling Language and Everest Platform. Procedia Computer Science, 101, pp. 313-322 (2016)

[6] Sukhoroslov, O., Rubtsov, A., Volkov, S.: Development of Distributed Computing Applications and Services with Everest Cloud Platform. Computer Research and Modeling, 7 (3), pp. 593-599 (2015)

[7] Sukhoroslov, O., Volkov, S., Afanasiev, A. A.: Web-Based Platform for Publication and Distributed Execution of Computing Applications. Parallel and Distributed Computing, 14th Int. Symposium on IEEE, pp. 175-184 (2015)

[8] Tamura, N., et al.: Impact of Nonlocal Electron Heat Transport on the High Temperature Plasmas of LHD. Nuclear Fusion, 47 (5), pp. 449 (2007)

[9] Линник, В. Г., Соколов, А. В., Мироненко, И. В.: Паттерны 137CS и их трансформация в ландшафтах ополья Брянской области. Современные тенденции развития биогеохимии. М.: ГЕОХИ РАН, с. 423-434 (2016)

[10] Морозов, В. А.: Регулярные методы решения некорректно поставленных задач. М.: Наука (1987)

[11] Пшеничный, Б. Н.: Метод линеаризации. М.: Наука (1983)

[12] Роженко, А. И.: Теория и алгоритмы вариационной сплайн-аппроксимации. Новосибирск: Изд-во ИВМиМГ СО РАН (2005)

[13] Тихонов, А. И.: О математических методах автоматизации обработки наблюдений. Проблемы вычислительной математики. М.: МГУ, с. 3-17 (1980)