Применение технологии параллельного программирования и сплайнов при решении уравнения Фредгольма второго рода

Бурова И. Г., профессор кафедры вычислительной математики СПбГУ,
burovaig@mail.ru,

Алцыбеев Г. О., аспирант кафедры вычислительной математики СПбГУ,
gleb.alcybeev@gmail.com

Аннотация

В работе рассматривается применение технологии параллельных вычислений OpenMP для решения интегрального уравнения Фредгольма второго рода с помощью локальных интерполяционных сплайнов второго порядка аппроксимации.

Введение

В настоящее время при разработке программного кода много внимания уделяется различного рода оптимизациям для повышения скорости вычислений. В качестве одного из основных способов повышения скорости работы программы можно отметить распараллеливание вычислений. Параллельные вычисления — это вид вычислений, при которых сразу несколько вычислительных процессов выполняются одновременно в течение одного и того же периода времени. Пионерами в области параллельных вычислений являются Эдсгер Дейкстра [1], Пер Бринч Хансен [2], [3] и К. А. Р. Хоар [4]. Большой вклад в области параллельных вычислений внесли В. В. Воеводин и В. В. Воводин [5], В. П. Гергель [6], А. С. Антонов [7], В. Д. Корнеев [8], и С. А. Немнюгин [9].

В качестве одного из инструментов параллельных вычислений можно выделить технологию OpenMP. OpenMP — это интерфейс прикладного программирования, который поддерживает многоплатформенное многопроцессорное программирование с общей памятью на языках C, C++ и Fortran.

Решение уравнения Фредгольма и сплайновые аппроксимации

Пусть $a, b \in \mathbb{R}$. Рассмотрим линейное интегральное уравнение Фредгольма второго рода

$$y\left(x\right)-\int_a^b K\left(x,s\right)y\left(s\right)ds=f\left(x\right), \quad x\in\left[a,b\right],$$

где $f\left(x\right)$ такая, что $f \in C\left[a,b\right]$ — правая часть, $K\left(x,s\right)$ — ядро, определенное в квадрате $\Pi = \left\{(x,s)\,|\,a\leqslant x\leqslant b, a\leqslant s\leqslant b\right\}$, полагаем, что ядро $K\left(x,s\right)$ непрерывно в квадрате Π, а $y\left(x\right)$ — искомая непрерывная функция, $x\in[a,b]$.

На промежутке $[a,b]$ задаем узлы сетки $a = x_1 < x_2 < \ldots < x_n = b$. Приближенные значения $\tilde{y}\left(s\right)$ функции $y\left(s\right)$ на промежутке $[x_j, x_{j+1}]$ определяем по правилу

$$\tilde{y}\left(s\right)=y\left(x_j\right)\omega_j\left(s\right)+y\left(x_{j+1}\right)\omega_{j+1}\left(s\right), \quad s\in\left[x_j,x_{j+1}\right], \tag{1}$$

где $\omega_j\left(s\right)$ и $\omega_{j+1}\left(s\right)$ — базисные полиномиальные сплайны

$$\omega_j\left(s\right)=\frac{s-x_{j+1}}{x_j-x_{j+1}}, \quad \omega_{j+1}\left(s\right)=\frac{s-x_j}{x_{j+1}-x_j}, \quad s\in\left[x_j,x_{j+1}\right],$$

отметим, что удобно представление $\omega_{j+1}\left(s\right)$: $\omega_{j+1}\left(s\right)=1-\omega_j\left(s\right)$.

Обозначим $\|y''\|_{[x_j,x_{j+1}]} = \max_{[x_j,x_{j+1}]}\left|y''\left(x\right)\right|$. Пусть $h = x_{j+1}-x_j$. Можно показать, что в случае полиномиальных сплайнов справедливо неравенство

$$|\tilde{y}(s)-y(s)|\leqslant 0{,}25h^2\|y''\|_{[x_j,x_{j+1}]}, \quad s\in[x_j,x_{j+1}].$$

Также можно использовать неполиномиальные базисные функции (см. [11], [12]) с погрешностью аппроксимации порядка $O(h^2)$.

Нетрудно видеть, что

$$\int_a^b K\left(x,s\right)y\left(s\right)ds\approx\sum_{k=1}^{n-1}\int_{x_j}^{x_{j+1}}K\left(x,s\right)\tilde{y}\left(s\right)ds, \quad x\in\left[a,b\right],$$

где $\tilde{y}\left(s\right)$ имеет вид (1). В результате применения сплайновых аппроксимаций, получаем систему линейных алгебраичеких уравнений (СЛАУ)

$$\tilde{y}\left(x_k\right)+\sum_{j=1}^{n-1}W_j\left(x_k\right)=f\left(x_k\right), \quad k=1,2,\ldots,n, \tag{2}$$

где

$$W_j(x_k) = \tilde{y}(x_j)\int_{x_j}^{x_{j+1}} K(x_k, s)\,\omega_j(s)\,ds + \\ +\tilde{y}(x_{j+1})\int_{x_j}^{x_{j+1}} K(x_k, s)\,(1-\omega_j(s))\,ds. \tag{3}$$

Равенство (3) можно упростить, таким образом оно принимает следующий вид

$$W_j(x_k) = (\tilde{y}(x_j) - \tilde{y}(x_{j+1}))\int_{x_j}^{x_{j+1}} K(x_k, s)\,\omega_j(s)\,ds + \\ +\tilde{y}(x_{j+1})\int_{x_j}^{x_{j+1}} K(x_k, s)\,ds,$$

кроме этого, в случае, если интеграл трудно вычислять, можно использовать следующую форму записи

$$W_j(x_k) = \left(\frac{\tilde{y}(x_j) - \tilde{y}(x_j+1)}{x_j - x_{j+1}}\right)\int_{x_j}^{x_{j+1}} K(x_k, s)\,s ds + \\ + \left(\tilde{y}(x_{j+1}) - \frac{x_{j+1}}{x_j - x_{j+1}}\right)\int_{x_j}^{x_{j+1}} K(x_k, s)\,ds.$$

В большинстве случаев интегралы вычисляются в конечном виде, либо можно применить соотвующие квадратурные формулы.

Приведем СЛАУ (2) к виду $A\tilde{y} = b$. Выпишем случай, когда $n = 3$, тогда $\tilde{y} = (\tilde{y}(x_1), \tilde{y}(x_2), \tilde{y}(x_3))^T$, а $b = (f(x_1), f(x_2),\ f(x_3))^T$.

С учетом того, что $n = 3$ имеем систему

$$\begin{cases} \tilde{y}(x_1) - \tilde{y}(x_1)K_{00} - \tilde{y}(x_2)K_{10} - \tilde{y}(x_2)K_{20} - \tilde{y}(x_3)K_{30} = f(x_1), \\ \tilde{y}(x_2) - \tilde{y}(x_1)K_{01} - \tilde{y}(x_2)K_{11} - \tilde{y}(x_2)K_{21} - \tilde{y}(x_3)K_{31} = f(x_2), \\ \tilde{y}(x_3) - \tilde{y}(x_1)K_{02} - \tilde{y}(x_2)K_{12} - \tilde{y}(x_2)K_{22} - \tilde{y}(x_3)K_{32} = f(x_3), \end{cases} \tag{4}$$

в которой приняты следующие обозначения

$$K_{00} = \int_{x_1}^{x_2} K(x_1, s)\,\omega_1(s)\,ds, \quad K_{10} = \int_{x_2}^{x_3} K(x_1, s)\,\omega_2(s)\,ds,$$

$$K_{20} = \int_{x_1}^{x_2} K(x_1, s)\,\omega_2(s)\,ds, \quad K_{30} = \int_{x_2}^{x_3} K(x_1, s)\,\omega_3(s)\,ds,$$

$$K_{01} = \int_{x_1}^{x_2} K(x_2, s)\,\omega_1(s)\,ds, \quad K_{11} = \int_{x_2}^{x_3} K(x_2, s)\,\omega_2(s)\,ds,$$

$$K_{21} = \int_{x_1}^{x_2} K(x_2, s)\,\omega_2(s)\,ds, \quad K_{31} = \int_{x_2}^{x_3} K(x_2, s)\,\omega_3(s)\,ds,$$

$$K_{02} = \int_{x_1}^{x_2} K(x_3, s)\,\omega_1(s)\,ds, \quad K_{12} = \int_{x_2}^{x_3} K(x_3, s)\,\omega_2(s)\,ds,$$

$$K_{22} = \int_{x_1}^{x_2} K(x_3, s)\,\omega_2(s)\,ds, \quad K_{32} = \int_{x_2}^{x_3} K(x_3, s)\,\omega_3(s)\,ds.$$

Нетрудно заметить, что система (4) преобразуется к системе вида

$$\begin{cases} \tilde{y}(x_1)\underbrace{(1 - K_{00})}_{a_{11}} - \tilde{y}(x_2)\underbrace{(K_{10} + K_{20})}_{a_{12}} - \tilde{y}(x_3)\underbrace{K_{30}}_{a_{13}} = f(x_1), \\ -\tilde{y}(x_1)\underbrace{K_{01}}_{a_{21}} + \tilde{y}(x_2)\underbrace{(1 - K_{11} - K_{21})}_{a_{22}} - \tilde{y}(x_3)\underbrace{K_{31}}_{a_{23}} = f(x_2), \\ -\tilde{y}(x_1)\underbrace{K_{02}}_{a_{31}} - \tilde{y}(x_2)\underbrace{(K_{12} + K_{22})}_{a_{32}} + \tilde{y}(x_3)\underbrace{(1 - K_{32})}_{a_{33}} = f(x_3), \end{cases}$$

которую можно записать в матричном виде

$$\begin{pmatrix} a_{11} & -a_{12} & -a_{13} \\ -a_{21} & a_{22} & -a_{23} \\ -a_{31} & -a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} \tilde{y}(x_1) \\ \tilde{y}(x_2) \\ \tilde{y}(x_3) \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \end{pmatrix}.$$

Можно показать, что матрица системы уравнений имеет диагональное преобладание при произвольном n, поэтому она — неособенная. Полученную СЛАУ можно решать численными методоми, например методом Гаусса.

В ходе представленной работы был разработан программный комплекс на языке C++ для работы с интегральными уравнениями, а также библиотеки **AS::LinearAlgebra** и **AS::MathAnalysis** для выполнения сопутствующих операций. Информация об этом в следующем разделе.

Распараллеливание вычислений в методе Гаусса

Полученную СЛАУ можно решать различными численными методами, например в данном случае хорошо подходит метод Гаусса. Много внимания распараллеливанию метода Гаусса было уделено в работах В. П. Гергеля (см. например [6]). В ходе работы была разработана высокоуровневая библиотека **AS::LinearAlgebra** на языке C++ для работы с матрицами и векторами. В библиотеку вошло большинство стандартных операций для решения задач линейной алгебры, в том числе различные методы для решения СЛАУ. Рассмотрим реализацию процедуры **AS::LinearSolve** на примере метода Гаусса. Параметры процедуры: A — экземпляр класса **AS::Matrix** из библиотеки **AS::LinearAlgebra**, b — экземпляр класса **AS::Vector** (может использоваться также класс **AS::Matrix**) из библиотеки **AS::LinearAlgebra**, столбец свободных членов.

Рассмотрим процедуру метода Гаусса в общем виде и внесем в нее некоторые модификации с помощью технологии OpenMP. Обозначим $y_i = \tilde{y}\left(x_i\right)$ и $b_i = f\left(x_i\right)$. Дана СЛАУ порядка n

$$\begin{cases} a_{11}^{(0)} y_1 + a_{12}^{(0)} y_2 + \ldots + a_{1n}^{(0)} y_n = b_1^{(0)} \\ a_{21}^{(0)} y_1 + a_{22}^{(0)} y_2 + \ldots + a_{2n}^{(0)} y_n = b_2^{(0)} \\ \ldots \\ a_{n1}^{(0)} y_1 + a_{n2}^{(0)} y_2 + \ldots + a_{nn}^{(0)} y_n = b_n^{(0)} \end{cases}.$$

Разделим первое уравнение системы на $a_{11}^{(0)}$, тогда получим

$$y_1 + a_{12}^{(1)} y_2 + a_{13}^{(1)} y_3 + \ldots + a_{1n}^{(1)} y_n = b_1^{(1)}, \tag{5}$$

где $a_{1j}^{(1)} = a_{1j}^{(0)} / a_{11}^{(0)}$, $j = 2, 3, \ldots, n$, $b_1^{(1)} = b_1^{(0)} / a_{11}^{(0)}$. Предположим, что система уравнений такова, что $n > 3000$. Вычисления в цикле деления элементов можно распараллелить используя директивы OpenMP **parallel** и **for**. В результате имеем конструкцию представленную на Листинге 1.

```
#pragma omp parallel
{
  #pragma omp for
  for (int i = 0; i < Ab.GetColSize(); i++)
  {
    Matrix_Ab_c[k][i] = Matrix_Ab_c[k][i] / Ab[k][k];
  }
}
```

Листинг 1. Участок кода с циклом деления элементов в процедуре **LinearSolve** с использованием директив OpenMP

Далее исключаем неизвестную y_1 из каждого уравнения системы, начиная со второго. Это делается вычитанием уравнения (5), умноженного на коэффициент при переменной y_1 в соответствующем уравнении. Преобразованные уравнения имеют вид

$$\begin{cases} y_1 + a_{12}^{(1)} y_2 + a_{13}^{(1)} y_3 + \ldots + a_{1n}^{(1)} y_n = b_1^{(1)} \\ \qquad a_{22}^{(1)} y_2 + a_{23}^{(1)} y_3 + \ldots + a_{2n}^{(1)} y_n = b_2^{(1)} \\ \qquad \ldots \\ \qquad a_{n2}^{(1)} y_2 + a_{n3}^{(1)} y_3 + \ldots + a_{nn}^{(1)} y_n = b_n^{(1)} \end{cases}, \tag{6}$$

где $a_{ij}^{(1)} = a_{ij}^{(0)} - a_{1j}^{(1)} \cdot a_{i1}^{(0)}, j = 2, 3, \ldots, n, b_i^{(1)} = b_i^{(1)} \cdot a_{i1}^{(0)}, i = 2, 3, \ldots, n.$ Поступаем аналогично со следующим уравнением из преобразованной системы. В конечном итоге приводим исходную систему к эквивалентной системе с треугольной матрицей

$$\begin{cases} y_1 + a_{12}^{(1)} y_2 + a_{13}^{(1)} y_3 + \ldots + a_{1n}^{(1)} y_n = b_1^{(1)} \\ \qquad y_2 + a_{23}^{(2)} y_3 + \ldots + a_{2n}^{(2)} y_n = b_2^{(2)} \\ \qquad \ldots \\ \qquad\qquad a_{nn}^{(n)} y_n = b_n^{(n)} \end{cases}. \tag{7}$$

Цикл с исключением неизвестной y_i из каждого уравнения системы также распараллеливается с применением директив **parallel** и **for**. В результате имеем конструкцию представленную на Листинге 2.

```
#pragma omp parallel
{
  #pragma omp for
  for (int i = k + 1; i < Ab.GetRowSize(); i++)
  {
    double K = Matrix_Ab_c[i][k] / Matrix_Ab_c[k][k];
    for (int j = 0; j < Ab.GetColSize(); j++)
    {
      Matrix_Ab_c[i][j] = Matrix_Ab_c[i][j]
        - Matrix_Ab_c[k][j] * K;
    }
  }
}
```

Листинг 2. Участок кода с циклом исключения неизвестной y_i в процедуре **LinearSolve** с использованием директив OpenMP

Далее обратным ходом из системы (7) находим неизвестные y_1, y_2,…, y_n. Соответствующие циклы распараллеливаются аналогично. Кроме этого, аналогичная конструкция была построена для цикла записи ответа, однако данные действия привели к незначительным ускорениям программы.

Были проведены численные эксперименты. Численные эксперименты проводились при решении СЛАУ, в случае 3000–4000 уравнений. В частности, выбиралось ядро $K(x, s) = e^{-s \cdot x}$, а правая часть $f(x)$ строилась по решению $y \equiv 1$. Характеристики ЭВМ, на которой проводились эксперименты: процессор — Inter Core i7-7700HQ CPU @ 2.80 ГГц, оперативная память — DDR4 8 Гб. Компилятор C++: MinGW w64 6.0. Среднее время рассчитывалось на основании 15 вызовов процедуры. В случае $n = 3000$ ускорение при распараллеливании вычислений оказалось равным примерно 1,5529.

Заключение

В работе было рассмотрено применение технологий параллельный вычислений для решения интегрального уравнения Фредгольма второго рода с помощью локальных интерполяционных сплайнов второго порядка аппроксимации.

Список литературы

[1] Edsger W. Dijkstra. Selected Writings on Computing: A Personal Perspective. Monographs in Computer Science. New York: Springer Science & Business Media, 2012. 362 p.

[2] Per Brinch Hansen. Design Principles. New Jersey: Prentice Hall, Englewood Cliffs, 1977. 314 p.

[3] Per Brinch Hansen. Experience with Modular Concurrent Programming // IEEE Transactions on Software Engineering. 1977. No. 3 (2). P. 156–159.

[4] Hoare C. A. R., Communicating Sequential Processes. New Jersey: Prentice Hall, 1985. 260 p.

[5] Воеводин В. В., Воеводин В. В. Параллельные вычисления. СПб.: BHV, 2002. 608 с.

[6] Гергель В. П. Высокопроизводительные вычисления для многоядерных многопроцессорных систем. Нижний Новгород: Изд-во Нижегородского госуниверситета, 2010. 421 с.

[7] Антонов А. С. Технологии параллельного программирования MPI и OpenMP. М.: Изд-во МГУ, 2012. 344 с.

[8] Корнеев В. Д. Параллельное программирование в MPI. Новосибирск: Изд-во ИВМиМГ СО РАН, 2002. 215 с.

[9] Немнюгин С. А., Стесик О. Л. Параллельное программирование для многопроцессорных вычислительных систем. СПб.: БХВ-Петербург, 2002. 396 с.

[10] Алцыбеев Г. О., Бурова И. Г. Газотурбинный двигатель и сплайновые приближения // Процессы управления и устойчивость. 2021. Т. 8 (24). С. 101–107.

[11] Burova I. G. On left integro-differential splines and Cauchy problem // International Journal of Mathematical Models and Methods in Applied Sciences. 2015. Vol. 9. P. 683–690.

[12] Burova I. G., Alcybeev G. O. Application of Splines of the Second Order Approximation to Volterra Integral Equations of the Second Kind. Applications in Systems Theory and Dynamical Systems // International Journal of Circuits, Systems and Signal Processing. 2021. Vol. 15. P. 63–71.