МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БАРАНОВИЧСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет педагогики и психологии БарГУ
Факультет славянских и германских языков БарГУ
Горловский институт иностранных языков

СОДРУЖЕСТВО НАУК. БАРАНОВИЧИ-2013

МАТЕРИАЛЫ IX МЕЖДУНАРОДНОЙ НАУЧНО-ПРАКТИЧЕСКОЙ КОНФЕРЕНЦИИ МОЛОДЫХ ИССЛЕДОВАТЕЛЕЙ

23—24 мая 2013 г.
г. Барановичи
Республика Беларусь

В 2 книгах
Книга 2

Барановичи
РИО БарГУ
2013

УДК 001
ББК 72
С57

Рекомендовано к печати редакционно-издательским советом учреждения образования
«Барановичский государственный университет»

Содружество наук. Барановичи-2013 [Текст] : материалы IX Междунар. науч.-практ. конф. молодых исследователей, 23—24 мая 2013 г., г. Барановичи, Респ. Беларусь : в 2 кн. / редкол.: А. В. Никишова (гл. ред.), А. В. Прадун, Ю. В. Башкирова (отв. ред.) [и др.] — Барановичи : РИО БарГУ, 2013. — Кн. 2. — 279 с. — 156 экз.

С57

ISBN 978-985-498-533-6
ISBN 978-985-498-534-3 (Книга 2)

Включены материалы докладов IX Международной научно-практической конференции молодых исследователей «Содружество наук. Барановичи-2013» по актуальным проблемам таких научных направлений, как экономические аспекты развития предприятия, региона; информационные технологии в образовании, науке и технике; современные тенденции развития производственных технологий машин и материалов; физика, математика; правоведение.

Адресовано преподавателям и студентам учреждений высшего образования, магистрантам, аспирантам.

УДК 001
ББК 72

ISBN 978-985-498-533-6
ISBN 978-985-498-534-3 (Книга 2)

УДК 004.021

А. А. Омельянович, О. И. Наранович
Учреждение образования «Барановичский государственный университет», Барановичи

РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ МОДИФИЦИРОВАННЫМИ АЛГОРИТМАМИ МЕТОДА ПРОГОНКИ

Метод прогонки наиболее удобен для решения разреженных трёх диагональных матриц, полученных в результате конечно-разностной или конечноэлементной аппроксимации дифференциальных уравнений в частных производных.

Sweep method is most suitable for solving sparse three diagonal matrices obtained by the finite difference or finite element approximation of partial differential equations.

Для решения систем линейных алгебраических уравнений (далее-СЛАУ) вида $Ax = b$ с трёхдиагональной матрицей A наиболее часто применяется метод прогонки, являющийся адаптацией метода Гаусса к этому случаю [1].

СЛАУ преобразовывается к виду

$$A_i x_i + B_i x_i + C_i x_{i+1} = F_i, i = 1,..., N-1$$

Исходный алгоритм приобретает следующий вид:

1. v4 = C[i] + A[i]*Alpha[i-1];
2. v4 = 1.0/v4;
3. v5 = -B[i];
4. v1 = v5*v4;
5. Alpha[i] = v1;
6. v2 = (f[i]- A[i]*Betha[i-1])*v4;
7. Betha[i] = v2; [1] .

Модифицированный алгоритм прогонки с использованием потоков. При использовании потоков в рамках программы достигается параллельность выполнения процессов, что приводит к увеличению производительности приложения, уменьшению использования системных ресурсов, и приложение становится более простым по структуре [2].

Для реализации данного алгоритма был создан класс **Progonka** с методом ***Run()*** (рисунок 1), который позволяет решать СЛАУ модифицированным методом.

Структура **Factor** содержит следующие поля:

a — массив, в который заносятся элементы главной матрицы, расположенные под главной диагональю;

Alpha — первый прогоночный коэффициент;

b — массив, в который заносятся элементы главной матрицы, расположенные над главной диагональю;

Betha — второй прогоночный коэффициент;

dim — размерность главной матрицы;

free_term — столбец свободных членов;

rez_v4 — поле, в которое заносятся значения **v4;**

rez_v5 — поле, в которое заносятся значения **v5=-b[i];**

v1 — поле, хранящее значения **v4*v5;**

v2 — поле, хранящее значения **(free_term[i]- Alpha[i]*Betha[i-1])*v4;**

v4 — поле, хранящее значения **c[i] + a[i]*Alpha[i-1];**

две WinAPI функции ***V_4_5*** и ***ALPHA_V2.***

Функция ***V_4_5*** получает значения переменных **v4** и **v5**. Затем она заносится в поток, благодаря чему переменные **v4** и **v5** вычисляются параллельно, это имеет смысл делать, так как операция чтения из памяти и деления происходит долго, и благодаря данной операции мы получаем ускорение выполнения программы.

Функция ***ALPHA_V2*** получает значение переменной **v2** и прогоночного коэффициента ***Alpha***. Затем она заносится в поток, благодаря чему запись элемента в память и его чтение из памяти происходит параллельно.

Описанный выше алгоритм реализован в среде Microsoft Visual Studio 2008 средствами языка C++:

```
//Значения переменных v4 и v5
void WINAPI V_4_5(PVOID pParam)
{
  Factor*localArgs=(Factor*)pParam;
  localArgs->rez_v4=1./localArgs->v4;
  localArgs->rez_v5=localArgs->b[localArgs->k]*(-1);
}

//Значения переменных Alpha[i] и v2
void WINAPI ALPHA_V2(PVOID pParam)
{
  Factor*localArgs=(Factor*)pParam;
  localArgs->Alpha[localArgs->k]=localArgs->v1;
  localArgs->v2=(localArgs->free_term[localArgs->k]-localArgs->a[localArgs->k]*localArgs-
  >Betha[localArgs->k-1])*localArgs->rez_v4;
```

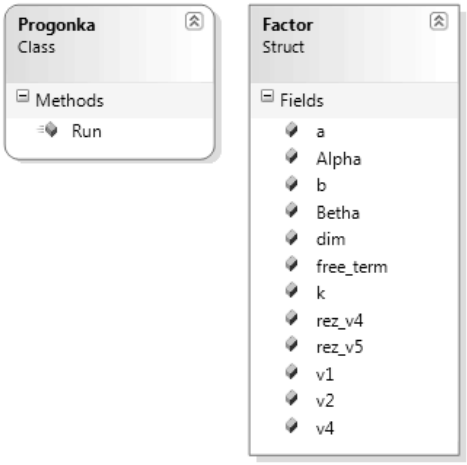

Рисунок 1 — Класс, реализующий метод прогонки с потоками

```
}
//Модифицированный алгоритм
  fact.Alpha[0]=-fact.b[0]/c[0];
  fact.v1=fact.Alpha[0];
  fact.Betha[0]=fact.free_term[0]/c[0];
  fact.k=1;
  for(int i=1;i<fact.dim-1;i++)
  {
  fact.v4=c[i]+fact.a[i]*fact.Alpha[i-1];
  Thr1=CreateThread(NULL,0,(LPTHREAD_START_ROUTINE)V_4_5,(PVOID) &fact,0,NULL);
  WaitForSingleObject(Thr1,INFINITE);

fact.v1=fact.rez_v5*fact.rez_v4;
  Thr2=CreateThread(NULL,0,(LPTHREAD_START_ROUTINE)ALPHA_V2,(PVOID)&fact,0,NULL);
  WaitForSingleObject(Thr2,INFINITE);

  fact.Betha[i]=fact.v2;
  fact.k++;
  }

  root[fact.dim-1]=(fact.free_term[fact.dim-1]-fact.a[fact.dim-1]*fact.Betha[fact.dim-2])/(c[fact.dim-
  1]+fact.a[fact.dim-1]*fact.Alpha[fact.dim-2] );

  for (int i=fact.dim-2;i>=0;i--)
        root[i]=fact.Alpha[i]*root[i+1]+fact.Betha[i];
```

Алгоритм прогонки с использованием стеков. Стек относится к области памяти, поддерживаемой процессором, в которой сохраняются локальные переменные. Доступ к стеку во много раз быстрее, чем к общей области памяти, поэтому использование стека для хранения данных ускоряет работу программы [3].

Для реализации данного алгоритма был использован класс **Progonka_St** и структура данных стек, которая реализована посредством контейнерного класса **stack**. Использованы методы класса **stack *push(), pop(), top(),*** а также метод класса **Progonka_St *Run()***, который решает СЛАУ модифицированным методом.

Метод ***push()*** используется для добавления значения в вершину стека.

Метод ***pop()*** используется для изменения значения в вершине.

Метод ***top()*** используется для извлечения значения из стека.

Все переменные (**v1, v2, v4, v5**) заносятся в стек. Благодаря этому обеспечивается быстрая запись в память и чтение из памяти данных значений, что позволяет значительно сократить время решения СЛАУ.

```
//Модифицированный алгоритм
  Alpha[0]=-b[0]/c[0];
  s.push(v1=Alpha[0]);
  Betha[0]=free_term[0]/c[0];
  for(int i=1;i<dim-1;i++)
  {
        s.push(v4=c[i]+a[i]*Alpha[i-1]);
        s.push(v4=1./s.top());
        s.push(v5=-b[i]);
        v1=s.top();
        s.pop();
        s.push(v1=v1*s.top());

        Alpha[i]=s.top();
        s.pop();
        s.push(v2=(free_term[i]-a[i]*Betha[i-1])*s.top());
        Betha[i]=s.top();
  }
  root[dim-1]=(free_term[dim-1]-a[dim-1]*Betha[dim-2])/(c[dim-1]+a[dim-1]*Alpha[dim-2] );

  for (int i=dim-2;i>=0;i--)
        root[i]=Alpha[i]*root[i+1]+Betha[i] ;
```

Результаты тестирования. В таблицах 1 и 2 приведены результаты тестирования предложенных алгоритмов с использованием ПЭВМ двух различных конфигураций.

Т а б л и ц а 1 — Результаты тестирования алгоритмов для ПЭВМ с конфигурацией 1

OS	**Windows XP SP3 x 32**		
CPU	**Pentium(R) Dual-Core CPU E5700 3.00 GHz**		
RAM	**2.00 Гб**		
Размерность матрицы	**10^2**	**10^3**	**10^4**
Время выполнения при использовании потоков	**0,062 с**	**0,797 с**	**20,094 с**
Время выполнения при использовании стеков	**≈0 с**	**0,016 с**	**0,156 с**

Т а б л и ц а 2 — Результаты тестирования алгоритмов для ПЭВМ с конфигурацией 2

OS	**Windows 7 x64**		
CPU	**Intel Core I7 – 3610QM, 2.3 GHz**		
RAM	**6.00 Гб**		
Размерность матрицы	**x10^2**	**x10^3**	**x10^4**
Время выполнения при использовании потоков	**0,07 с**	**0,281**	**2,761 с**
Время выполнения при использовании стеков	**≈0с**	**≈0с**	**0,125 с**

Таким образом, полученные результаты показывают, что алгоритм наиболее эффективно работает при его реализации с помощью стеков, так как структура данных стек обеспечивает более быструю запись информации в память и её чтение из памяти, в отличие от параллельных вычислений с помощью потоков. Максимальный размер матрицы, вычисляемый с помощью данного алгоритма, — $10^4 \times 10^4$.

Список цитируемых источников

1. Метод прогонки [Электронный ресурс] — Режим доступа:: http://ssd.sscc.ru/school/2006s/files/Progonka.doc — Дата доступа :12.07.2012. — Загл. с экрана.
2. Метод параллельного программирования [Электронный ресурс]. — Режим доступа : URL: http://www.xaviersarrate.com/parallelnoe-prog/preimushchestva.html — Дата доступа: 12.07.2012. — Загл. с экрана.
3. : [Электронный ресурс] . Режим доступа: http://simple-cs.ru/store/csharp/4 — Дата доступа : 12.07.2012 . — Загл. с экрана.

Материал поступил вредакцию 15.03.2013 г.