УДК 519.6

И. В. Мельник, д-р техн. наук, **Н. В. Шинкаренко**, аспирантка
Национальный технический университет Украины
«Киевский политехнический ин-т»
(Украина, 03056, Киев, пр. Победы, 37, корпус 12, 2203,
тел. (044) 4068292, (044) 4549505, E-mail: imelnik@edd.ntu-kpi.kiev.ua)

Анализ алгоритмических особенностей вычисляемых матриц при решении задач программирования средствами матричных макроопераций

Рассмотрены возможности решения вычислительных задач среднего уровня сложности средствами матричного программирования. Для упрощения построения структур алгоритмов при написании программ средствами матричного программирования введено понятие матричных диаграмм, в которых выделены иерархические и рекуррентные связи. Проанализирована возможность использования контейнерных вычислительных схем для аппаратного ускорения процесса рекуррентных вычислений. В качестве примера приведена программа, реализующая известный метод Рунге—Кутта четвертого порядка, написанная средствами матричного программирования.

Розглянуто можливості розв'язування обчислювальних задач середнього рівня складності засобами матричного програмування. Для спрощення побудови структур алгоритмів при написанні програм засобами матричного програмування введено поняття матричних діаграм, в яких виділено ієрархічні та рекурентні зв'язки. Проаналізовано можливості використання контейнерних обчислювальних схем для апаратного прискорення процесу рекурентних обчислень. Як приклад наведено програму, в якій засобами матричного програмування реалізовано відомий метод Рунне—Кутта четвертого порядку.

К л ю ч е в ы е с л о в а: матричные макрооперации, матричные диаграммы, иерархические связи, рекуррентные связи, упорядоченные связи, произвольные связи, контейнеры.

В настоящее время в различных областях инженерной и научной деятельности при реализации аналитических расчетов и численных алгоритмов, необходимых для решения разнообразных по уровню сложности математических, инженерных и научных задач, все чаще применяется система научно-технических расчетов MatLab [1, 2]. Эта система позволяет, наряду с использованием обширных библиотек стандартных математических функций, эффективно решать стандартные и оригинальные задачи программирования, что делает ее особо привлекательной для инженерных и научных работников, решающих задачи математического моделирования

в процессе создания сложного программного обеспечения, и прежде всего для программистов. Во всем мире MatLab фактически стал стандартом универсальной математической САПР не только в научно-исследовательской, но и в университетской среде, о чем свидетельствует обширная учебная литература, изданная в последние годы [1 — 4, 6].

Основной концепцией разработчиков системы MatLab является ее ориентация на использование операторов и функций обработки матриц вместо стандартных средств структурного программирования, к которым традиционно относят операторы ветвления и цикла. Важным достоинством MatLab является наличие внутреннего языка программирования, в котором использованы все лучшие достижения таких современных языков, как BASIC, Pascal, C, C++, Java, Prolog. Особенностью языка MatLab является также то, что он позволяет реализовать различные современные подходы и парадигмы программирования, включая структурное, объектно-ориентированное, логическое, функциональное, модульное программирование [1—4].

Однако средства работы со структурами числовых данных, в частности с матрицами и векторами [1, 2, 5], а также особенности использования этих средств при реализации как традиционных, так и оригинальных вычислительных алгоритмов, пока изучены недостаточно. Поэтому программисты, пишущие на языке MatLab, часто вместо работы с матрицами предпочитают использовать хорошо изученные стандартные средства структурного программирования. Это подтверждает необходимость дальнейшего исследования дополнительных возможностей матричных макроопераций с целью их использования при написании программ.

Такие исследования описаны в работах [1, 2, 5], где введены новые понятия арифметико-логического выражения и вектора-функции и приведены программные коды некоторых функций, предназначенных для проведения рекуррентных вычислений в MatLab. В результате исследований удалось без ветвлений и циклов реализовать классические алгоритмы поиска и сортировки элементов вектора, а также подсчет числа элементов, лежащих в заданном интервале.

Кроме того, средствами матричного программирования успешно решены более сложные математические задачи, в частности выполнен расчет степенных рядов, реализованы алгоритмы метода простой итерации для решения нелинейных уравнений и явного метода Эйлера для решения обыкновенных дифференциальных уравнений [1, 2, 5, 6—9]. Исследована также возможность использования матричных макроопераций для численного решении физических задач среднего уровня сложности с учетом их специфики и алгоритмических особенностей [2, 5, 7]. В качестве при-

мера подробно рассмотрена задача о движении электрона в магнитном поле и предложен метод ее решения, основанный на использовании средств матричного программирования, в частности понятия вектора-функции [5].

Основные недостатки методов матричного программирования [5] связаны, в первую очередь, с увеличением затрат процессорного времени и оперативной памяти. Однако следует учитывать увеличение вычислительных возможностей современных компьютеров и необходимость создания высококачественных программных продуктов в крайне сжатые сроки, что обусловлено жесткими экономическими требованиями современного рынка. Поэтому в ряде случаев простота написания и отладки программных средств является более существенным преимуществом, чем экономия вычислительных ресурсов.

В связи с этим представляет научный и практический интерес дальнейший анализ возможностей использования матричных макроопераций при реализации более сложных вычислительных алгоритмов, как рекуррентных, так и параллельных. При этом методом исследования является анализ связей между вычисляемыми элементами матрицы и их систематизация в соответствии с особенностями реализуемого алгоритма.

Иерархические и рекуррентные связи и их использование при вычислениях элементов матриц. Для расширения возможностей матричных вычислений получены соответствующие функции [1, 2, 5], позволяющие реализовать рекуррентные алгоритмы посредством вычисления элементов матрицы. Исследуем связи между элементами строк матрицы, их упорядочивание и построение на основе анализа этих связей вычислительного алгоритма.

Связи между элементами матрицы, используемые при построении вычислительных алгоритмов, можно разделить на иерархические и рекуррентные. Иерархические связи характеризуют взаимосвязь вычисляемого элемента с элементами предыдущих строк, а рекуррентные — с ранее вычисленными элементами текущей строки. Пример иерархических и рекуррентных связей для одного из вычисляемых элементов матрицы **M** приведен на рис. 1. Такие схемы далее будем называть матричными диаграммами.

Как иерархические, так и рекуррентные связи могут быть упорядоченными или произвольными. При упорядоченных связях функции для вычисления соседних элементов строки формируются по определенному закону, который может быть описан математически с указанием номера вычисляемого элемента. Произвольные связи имеют неупорядоченный характер и поэтому не поддаются строгому математическому описанию. Схемы упорядоченных и неупорядоченных иерархических связей между элементами строки матрицы приведены на рис. 2.

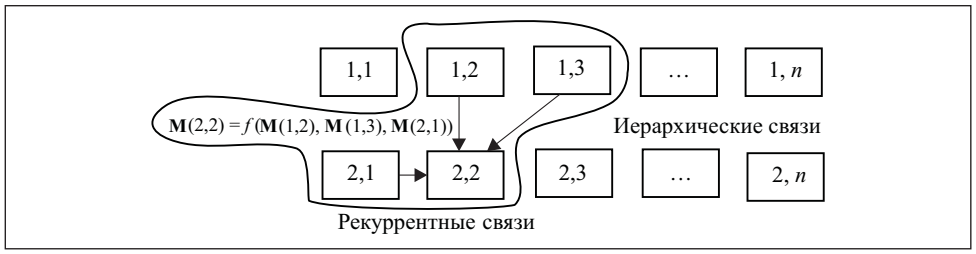

Рис. 1. Иерархические и рекуррентные связи для вычисляемого элемента матрицы **M** (2, 2)

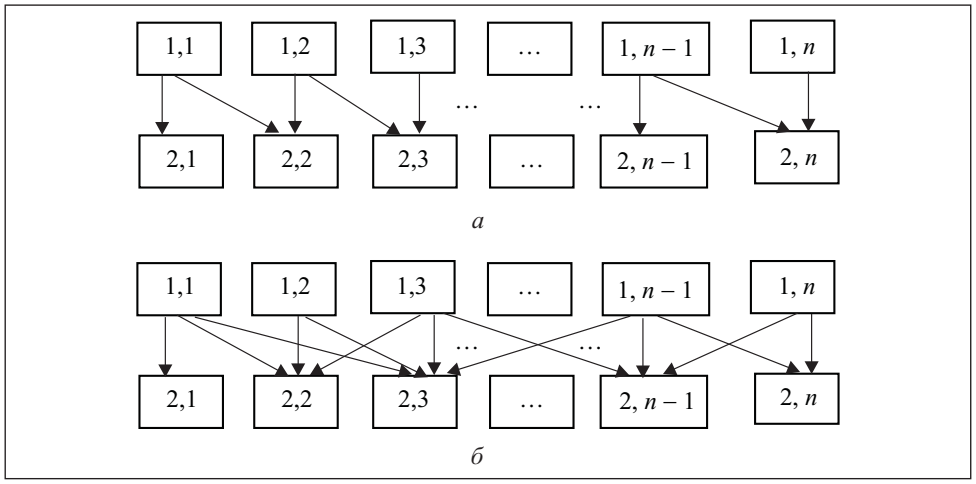

Рис. 2. Схемы упорядоченных (*а*) и неупорядоченных (*б*) связей между вычисляемыми элементами строк матрицы

В вычислительных алгоритмах могут быть описаны как упорядоченные, так и неупорядоченные связи между элементами строк матрицы. В частности, в [2, 5] показана возможность описания неупорядоченных иерархических связей между элементами матрицы через вектор-функцию, в которой каждый элемент вектора описывает функцию для вычисления соответствующего элемента строки матрицы.

Следует заметить, что неупорядоченными могут быть только связи между элементами одной строки, а связи между строками всегда носят закономерный характер, и именно это позволяет использовать рекуррентные вычисления для последовательного нахождения всех элементов результирующей матрицы. Примером решения реальной физической задачи с неупорядоченными связями между вычисляемыми элементами строк результирующей матрицы является расчет траекторий электронов в поле короткой фокусирующей магнитной линзы [2, 5].

Связи между элементами строк матрицы и их использование для оценки возможностей распараллеливания алгоритмов. Развитие вычислительной техники и переход современной полупроводниковой индустрии на производство многоядерных микропроцессоров стимулируют интерес к изучению возможностей использования параллельных вычислений для ускорения вычислительного процесса. Однако в классической теории алгоритмов вопрос о возможности их распараллеливания не исследуется и даже не существует аналитических средств, позволяющих четко выделить параллельные и последовательные ветви разработанного алгоритма. В связи с этим пока не разработана строгая теория параллельных вычислений, а исследования этой важной научно-технической проблемы проводятся полуэмпирическим способом, что в значительной степени снижает их эффективность.

Представление вычислительных алгоритмов в виде матричных диаграмм, аналогичных приведенным на рис. 1 и 2, позволяет четко выделить параллельные и последовательные ветви вычислительного процесса и оценивать не только возможность, но и степень его распараллеливания. Очевидно, что для таких оценок можно пользоваться следующим простым правилом.

Все вычислительные процессы, представленные на матричной диаграмме в виде иерархических связей, можно реализовать параллельно, в то время как процессы, представленные рекуррентными связями, основаны на последовательных вычислениях.

Таким образом, оценив число иерархических и рекуррентных связей на матричных диаграммах, можно не только сделать вывод о возможности распараллеливания алгоритма, но и определить степень распараллеливания с учетом числа иерархических и рекуррентных связей. При необходимости более строгих оценок степени распараллеливания алгоритма можно учесть также сложность рекуррентных и иерархических вычислительных процедур подсчетом числа элементарных операций процессора.

При неупорядоченных иерархических связях между элементами строк матрицы также возможно распараллеливание вычислительного процесса. В частности для задачи расчета траектории электрона в поле короткой магнитной линзы [2, 5] вычисление всех элементов результирующей матрицы может быть выполнено параллельно. Следует заметить, что возможность такого распараллеливания алгоритма существует для достаточно широкого круга физических задач среднего уровня сложности, связанных с анализом линейных или нежестких функций [7].

Параллельные вычисления взаимосвязанных физических величин допустимы в случаях, когда малый шаг пространственно-временной дискретизации допускает использование элементов предыдущей строки мат-

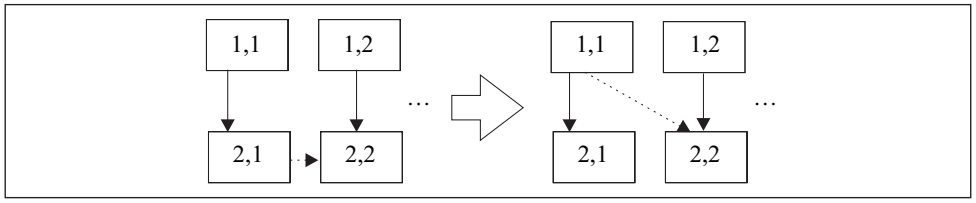

Рис. 3. Преобразование рекуррентных связей в иерархические

рицы вместо вычисленных элементов текущей строки. Тогда при построении вычислительной схемы необходимо, используя матричные диаграммы, провести модификацию алгоритма на основе преобразования рекуррентных связей в иерархические. Суть этого преобразования состоит в том, что связи между элементами текущей строки матрицы заменяются соответствующими связями с элементами того же столбца предыдущей строки. Преобразования связей между элементами вычисляемой матрицы показаны на рис. 3.

Однако следует заметить, что без существенных потерь точности расчета преобразования рекуррентных связей в иерархические возможны только для функций с малым и средним коэффициентом жесткости. Для жестких функций такое преобразование возможно только при очень малом шаге дискретизации, использование которого недопустимо с точки зрения рационального использования компьютерных ресурсов [8]. Именно поэтому все современные численные методы решения алгебраических и дифференциальных нелинейных уравнений, предназначенные для работы с жесткими функциями, основаны на рекуррентных процедурах, не допускающих замены в матричных диаграммах рекуррентных связей иерархическими. Такие замены при реализации итерационных процедур могут приводить к значительной потере точности расчетов.

Таким образом, матричные диаграммы — значительно более эффективный аналитический инструмент для исследования логических и информационных взаимосвязей между вычисляемыми переменными, чем широко используемые блок-схемы алгоритмов. При создании новых алгоритмов использование матричных диаграмм позволяет провести априорную оценку их сложности и основных вычислительных свойств, включая возможность распараллеливания. Используемые в настоящее время подходы, основанные на теории графов, по сравнению с матричными диаграммами являются слишком громоздкими и применимы только для анализа вычислительных особенностей очень сложных математических задач с разделением на подзадачи [10]. Свойственный матричному программированию декларативный способ описания сложных вычислительных процедур, для которого характерно отсутствие циклов и ветвящихся процессов, значительно упрощает программную реализацию параллельных вычислений.

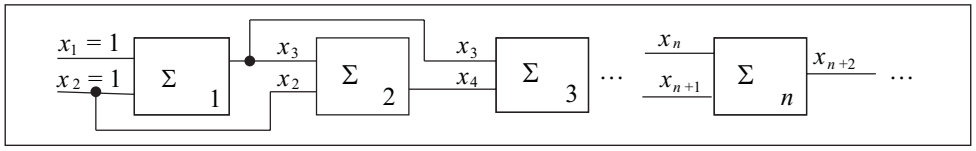

Рис. 4. Схема вычисления элементов последовательности Фибоначчи с помощью контейнера: *n* — номер сумматора

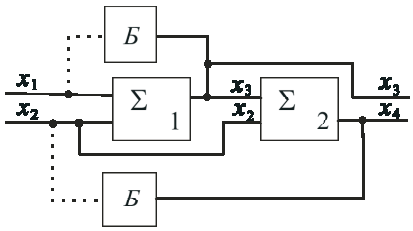

Рис. 5. Упрощенная схема вычисления элементов последовательности Фибоначчи, построенная на двух сумматорах Σ и двух буферных регистрах *Б*

Возможность аппаратного ускорения рекуррентных алгоритмов с использованием контейнеров. Схемы, в которых вместо параллельного соединения вычислительных устройств используется последовательное, называют контейнерами [11]. При этом для обеспечения непрерывного вычислительного процесса часто используют буферные регистры для хранения результатов вычислений.

Рассмотрим контейнерную схему для вычисления последовательности Фибоначчи. Соответствующая программа реализации вычисления элементов этой последовательности средствами матричных макроопераций системы MatLab приведена в [2, 5]. На рис. 4 представлена контейнерная схема с последовательным соединением сумматоров, для вычисления элементов последовательности ряда Фибоначчи, построенная без использования буферных регистров. Ее достоинством является высокое быстродействие, но она обладает аппаратной избыточностью, так как при увеличении числа итераций необходимо увеличивать число используемых сумматоров. Более простая схема, реализующая ту же функцию, но построенная с использованием буферных регистров *Б*, показана на рис. 5.

Рассмотрим возможность аппаратного ускорения для метода Рунге—Кутта четвертого порядка, широко используемого в вычислительной математике для численного решения обыкновенных дифференциальных уравнений. Известно, что для уравнения вида $\frac{dy}{dt} = f\,(t, y)$ этот метод описывается системой итерационных алгебраических уравнений [6—8]:

$$k_1 = f\,(t_n, y_n)\,;$$

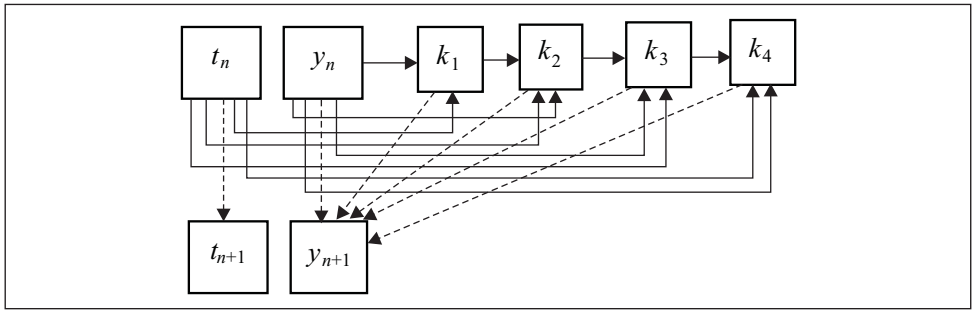

Рис. 6. Матричная диаграмма для метода Рунге—Кутта четвертого порядка: - - - — иерархические связи; —— — рекуррентные связи

$$k_2 = f\left(t_n + \frac{\tau}{2}, y_n + \frac{\tau k_1}{2}\right);$$

$$k_3 = f\left(t_n + \frac{\tau}{2}, y_n + \frac{\tau k_2}{2}\right); \qquad (1)$$

$$k_4 = f\left(t_n + \tau, y_n + \tau k_3\right);$$

$$y_{n+1} = y_n + \frac{\tau\,(k_1 + 2k_2 + 2k_3 + k_4)}{6},$$

где τ — шаг интегрирования; k_1, k_2, k_3, k_4 — вычисляемые коэффициенты; n — номер итерации. Матричная диаграмма для системы (1) приведена на рис. 6, из которого видно, что в данном случае рекуррентные связи меду элементами $y_n \to k_1 \to k_2 \to k_3 \to k_4$ не могут быть преобразованы в иерархические и поэтому аппаратное ускорение можно получить только при использовании контейнерных схем.

Один из возможных вариантов такой вычислительной схемы для метода Рунге—Кутта приведен на рис. 7, из которого видно, что это схема с достаточно сложными структурными связями, содержащая семь сумматоров $\sum$, четыре умножителя *П*, четыре блока умножения × или деления (/) на число, четыре функциональных блока *F* и два буферных регистра *Б* для хранения результатов вычислений.

Функциональные блоки, реализующие на аппаратном уровне вычисление функции $f(t, y)$, самые сложные при том, что характер этой функции изначально неизвестен. Однако ни в схемах параллельных, ни в схемах контейнерных вычислений нельзя использовать один вычислительный блок вместо нескольких повторяющихся, поскольку каждый из них выполняет в схеме свою функцию, и работают они всегда не последовательно, а одновременно.

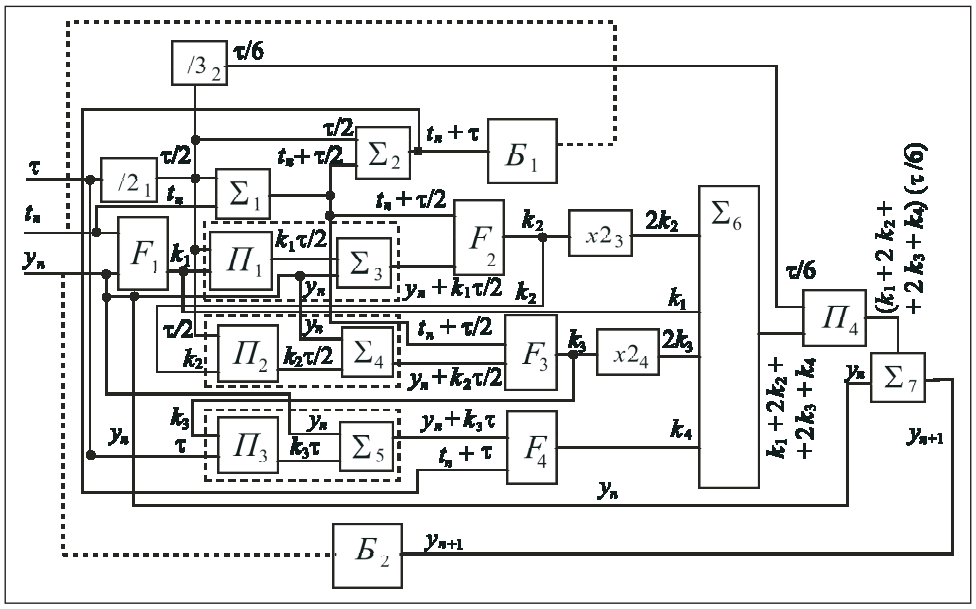

Рис. 7. Схема реализации контейнерных вычислений для рекуррентного метода Рунге—Кутта четвертого порядка: штриховой линией обведены блоки, выполняющие однотипные операции; пунктирные линии — обратная связь

Программа на языке MatLab, в которой алгоритм метода Рунге—Кутта реализован средствами матричного программирования, следующая:

```
function outxy=runge4(xst,xend,dx,yst,ff)
if (xst>=xend)error…
('Error! Start point is greater, then end point, verify input data'),…
end;
if (xend-xst<=dx) error …
('Error! Integration interval is smaller, then step, verify input data'), …
end;
if (dx<=0) error…
('Error! Step can not be negative or zero, verify input data'),… end;
strdx=num2str(dx/2); strdx2=num2str(dx);
str0=strcat('(ii==2)*(v(2)+',strdx2,')+');
str1=strcat('(ii==3)*',ff,'(v(2),v(7))+');
str2=strcat('(ii==4)*',ff,'(v(2)','+',strdx,',v(7)+',…
    strdx,'*v(3))+');
str3=strcat('(ii==5)*',ff,'(v(2)','+',strdx,',v(7)+',…
    strdx,'*v(4))+');
str4=strcat('(ii==6)*',ff,'(v(2)','+',strdx2,',v(7)+',…
```

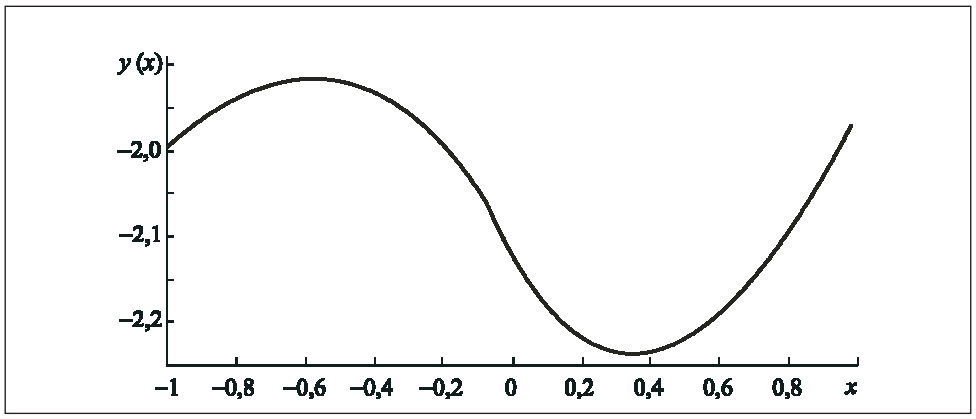

Рис. 8. График решения уравнения (2) на отрезке [– 1; 1]

```
    strdx2,'*v(5))+');
str5=strcat('(ii==7)*(v(7)+(v(3)+2*v(4)+2*v(5)+v(6))*',…
    strdx2,'/6)');
strfun=strcat(str0,str1,str2,str3,str4,str5);
strin=strcat('recvectm(1,7,length(M(:,1)),…
M(length(M(:,1)),:),''',strfun,''')');
nn=floor((xend-xst)/dx); vin=[xst,xst,0,0,0,0,yst];
outm=recmat(1,nn,vin,strin,1e-10,1e10);
outy=outm(:,7); outx=outm(:,2);
hpc=plot(outx,outy);
grid on
set (hpc,'Color',[0,0,0],'LineWidth',[2]);
outxy=[outx; outy];
return
```

Анализ полученных результатов. Матричные диаграммы отражают свойства вычислительных алгоритмов и характеризуют возможность распараллеливания вычислительного процесса. В соответствии с матричной диаграммой может быть построена вычислительная схема для аппаратного ускорения процесса вычислений, в которой иерархическим связям соответствует параллельное подключение элементарных вычислительных блоков, а рекуррентным связям — последовательное подключение, или контейнеры. При организации вычислительного процесса через контейнеры вычислительные схемы могут быть значительно упрощены при использовании буферных регистров. Обычные блок-схемы алгоритмов, так же как и графы, используемые для анализа возможностей распараллеливания алгоритмов сложных задач [10], не могут быть непосредст-

венно использованы для построения вычислительных схем, поэтому преимущества матричных диаграмм в данном случае очевидны.

Для реализации предложенного подхода наиболее подходящим является язык программирования системы MatLab, поскольку в нем наиболее развито декларативное описание матричных макроопераций. Аппаратные возможности использования параллельных и контейнерных вычислений непосредственно зависят от структуры и архитектуры используемого компьютера и в первую очередь от его центрального процессора.

На рис. 8 представлен результат работы приведенной выше программы для уравнения

$$\frac{dy}{dx} = \sqrt{|xy|} + \frac{x - y^2}{x^2 + y^2}, \quad y(-1) = -2. \tag{2}$$

Программа написана так, что начальные условия и функция, описывающая правую часть уравнения, заданы как внешние параметры.

Аналогичные исследования были проведены для полиномов Чебышева и для аппроксимации сплайнами [6, 8, 9].

Выводы

Полученные результаты позволяют сделать вывод о возможности реализации вычислительных алгоритмов среднего уровня сложности с помощью матричных макроопераций со структурами данных без использования стандартных средств структурного программирования, в частности операторов ветвления и циклов.

При использовании матричных макроопераций алгоритм вычислений удобно представлять в виде матричных диаграмм, выделяя на них рекуррентные и иерархические связи. Матричные диаграммы наиболее наглядно отображают вычислительные свойства анализируемого алгоритма, позволяющие рассматривать возможности его распараллеливания и строить схему всего вычислительного процесса. При построении схемы вычислений иерархические связи необходимо заменять параллельными блоками, а рекуррентные связи — последовательными блоками, или контейнерами.

Полученные результаты могут быть использованы для развития теории алгоритмов, а также в практике программирования при анализе вычислительных свойств новых алгоритмов и возможностей их распараллеливания.

Possibilities of solving a computational problems of middle level of complicity using matrix programming means are considered in the article. To simplify the creation of algorithm structures during writing the programs by means of matrix programming a conception of matrix diagrams has been introduced. Hierarchic and recurrent relations are distinguished in matrix diagrams. A

possibility of using the container calculation procedures for the schemes of hardware acceleration of the process of recurrent calculations is also analyzed. As an example, the program realizing calculations for well-known Runge—Kutt fourth order algorithm and written using the proposed matrix programming conception is presented.

1. *Мельник І. В.* Система науково-технічних розрахунків MatLab та її використання для розв'язання задач із електроніки: навчальний посібник. Т. 1. Основи роботи та функції системи. — К. : Ун-т «Україна», 2009. — 507 с.
2. *Мельник І. В*. Система науково-технічних розрахунків MatLab та її використання для розв'язання задач із електроніки: навчальний посібник. Т. 2. Основи програмування та розв'язання прикладних задач. — К. : Ун-т «Україна», 2009. — 327 с.
3. *Дьяконов В. П.* MatLab 6/6.1/6.5 + Simulink 4/5. Полное руководство пользователя. — М. : Салон-Пресс, 2002. — 768 с.
4. *Мартынов Н. Н.* Введение в MatLab 6. — М. : Кудиц-Образ, 2002. — 352 с.
5. *Мельник И. В.* Анализ возможностей использования матричных макроопераций системы MatLab при решении прикладных задач // Электрон. моделирование.— 2009. — **31**, № 3. — С. 37—51.
6. *Мэтьюз Д., Куртис Д.* Численные методы. Использование MatLab. — М. : Изд. дом «Вильямс», 2001. — 720 с.
7. *Ильина В. А., Силаев П. К.* Численные методы для физиков-теоретиков. — Москва-Ижевск: Институт компьютерных исследований, 2003. — 132 с.
8. *Самарский А. А., Гулин А. В.* Численные методы: Учеб. пособие для вузов. — М.: Наука, 1989. — 432 с.
9. *Мельник И. В, Шинкаренко Н. В.* Анализ взаимосвязей вычисляемых элементов матриц при реализации рекуррентных алгоритмов с помощью матричных макроопераций / Моделювання та інформаційні технології. Матеріали міжнародної наук. конф. «Моделювання — 2010», 12—14 травня 2010 р. — Київ : ІПМЕ НАН України, 2010. — С. 253—261.
10. *Воеводин В. В.* Вычислительная математика и структура алгоритмов. — М. : Изд. МГУ, 2006. — 112 с.
11. *Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.* Приемы объектно-ориентированного проектирования. — СПб: «Питер», 2007. — 366 с.

Поступила 30.08.10

МЕЛЬНИК Игорь Витальевич, д-р техн. наук, доцент кафедры электронных приборов и устройств Национального технического университета Украины «Киевский политехнический ин-т», который окончил в 1989 г. Область научных исследований — моделирование электронно-лучевых технологических устройств, теория газового разряда, программирование и теория алгоритмов.

ШИНКАРЕНКО Наталья Викторовна, аспирантка кафедры электронных приборов и устройств Национального технического университета Украины «Киевский политехнический ин-т», который окончила в 2009 г. Область научных исследований — математическое моделирование и теория алгоритмов.