Библиотека Математических программ-решателей на языке Си

SADEL

Версия 2.1

Руководство пользователя

Андронов А.В., Жук Д.М., Кожевников Д.Ю., Маничев В.Б.

SADEL является частью проекта ПА10

http://pa10.ru, http://па10.рф

info@pa10.ru

Оглавление

Введение 3
1. Математические основы разработки 4
1.1 Принципы, лежащие в основе алгоритмов библиотеки 4
1.2 Математическая постановка задачи решения СЛАУ с гарантированной точностью 6
1.3 Математическая постановка задачи решения систем ОДУ с гарантированной корректностью и точностью 7
2. Системные требования 8
3. Структура библиотеки 9
3.1. Решатель СЛАУ методом Гаусса: LAE_Solver_01 10
3.2. Решатель СЛАУ с 3-х диагональными матрицами: LAE_Solver_3diag 11
3.3. Решатель СЛАУ методом LU разложения с полными матрицами: LAE_Solver_02 12
3.4. Решатель ДАУ с полными матрицами Якоби: DAE_Solver_01 13
4. Рекомендации по тестированию 19
4.1. Тестирование решателей СЛАУ 20
4.2. Тестирование решателей ДАУ 21
5. Список литературы 22

Введение

Библиотека **SADEL** (**S**ets of **A**lgebraic and **D**ifferential **E**quations solvers **L**ibrary – библиотека решателей для систем алгебраических и дифференциальных уравнений) предназначена для решения систем линейных алгебраических уравнений (СЛАУ) и систем обыкновенных дифференциальных уравнений (ОДУ) в форме дифференциально-алгебраических уравнений (ДАУ).

Программы библиотеки основаны на кардинально новых идеях, которые еще не отражены в учебниках по вычислительной математике. Профессор СПбГУ Петров Юрий Петрович в своих книгах [10,3] показал, что для вычислений, связанных с проектированием и созданием сложной и опасной техники (аэрокосмическая техника, атомные станции, и т.п.), корректность и точность вычислений очень важны. Юрий Петрович, например, показал, что ошибки и неточности в инженерных расчетах (в частности, при использовании общепризнанных эквивалентных (равносильных) преобразований [8] при решении «особых», «не корректных» (плохо обусловленных) СЛАУ), которые прошли все стадии утверждения и попали в конечную конструкторскую документацию, были причиной ряда авиа катастроф, а потери для госбюджета от таких катастроф только в военной авиации России превышают 3 млрд. рублей в год [11]. Юрий Петрович показал, что инженерные расчеты, как правило, выполняются для номинальных значений параметров самолетов так же, как и опытные испытания, которые показывают, что самолет работает отлично. Но для "особых" систем линейных уравнений малые изменения параметров приводят к чрезвычайно большим изменениям в решении этих уравнений. Вместе с тем, в процессе эксплуатации самолета малые изменения параметров от номинальных значений неизбежны (особенно при разнообразных, не стандартных внешних воздействиях), что иногда может приводить и приводит к авариям и катастрофам, поэтому корректность и точность инженерных расчетов при вариации параметров чрезвычайно важны. Заключения о причинах катастрофы со спутниками ГЛОНАСС, а также многих аварий российских и зарубежных самолетов подтверждают научные открытия и утверждения Юрия Петровича. Для сверхжестких и сверхколебательных классов задач получаемое даже для номинальных значений параметров решение с помощью известных решателей иногда сильно отличается от корректного, достоверного и точного решения. Сверхжесткими системами мы называем системы ОДУ со степенью жесткости более 10^6, которым соответствуют СЛАУ с числом обусловленности более 10^6, а сверхколебательными системами – системы ОДУ, которые моделируют колебательные системы с добротностью более 10^6. Теоретическое обоснование новых идей и методов и их экспериментальное подтверждение на соответствующих наборах тестовых задач рассматривается в лекциях Маничева В.Б. (см. http://pa10.ru)/.

Мы отказались от общепринятого [8] приведения систем ОДУ к нормальной форме Коши, разрешенной относительно производных, отказались от явных методов интегрирования систем ОДУ - применяем только неявные AL-устойчивые методы интегрирования систем ОДУ, отказались от численных эквивалентных (равносильных) преобразований - используем только символьные эквивалентные преобразования и перестановки строк и столбцов матриц коэффициентов при решении СЛАУ. В результате мы получаем только корректные, достоверные и точные результаты решения СЛАУ и систем ОДУ.

*Библиотека математических программ **SADEL** создается с целью предоставления сверхнадежных и сверхточных программ-решателей СЛАУ и систем ОДУ на языке Си (по сравнению с соответствующими известными программами-решателями на языке Си) **с гарантированной математической компьютерной машинной точностью.***

Свободно распространяется демонстрационная версия библиотеки, содержащая ограничения на размерность решаемых задач.

1. Математические основы разработки

1.1 Принципы, лежащие в основе алгоритмов библиотеки

SADEL (**S**ets of **A**lgebraic and **D**ifferential **E**quations solvers **L**ibrary – библиотека решателей для систем алгебраических и дифференциальных уравнений) – это математическая библиотека программ-решателей на алгоритмическом языке Си для решения СЛАУ и систем ОДУ с гарантированной математической компьютерной машинной точностью. Решение СЛАУ гарантируется получать с точностью в **15** верных значащих цифр (удвоенная точность вычислений *double precision* на языке Си, - это общепринятая точность элементарных вычислений в расчетных программах на языке Си) для **всех элементов вектора решений** СЛАУ, решение систем ОДУ гарантируется с задаваемой относительной точностью не менее **0.001** для всех переменных решаемой системы ОДУ.

Научно-исследовательские работы по численному решению систем ОДУ мы выполняем с 80-х годов. В наших работах было показано, что для гарантии получения качественно корректного и достоверного решения систем ОДУ численный метод решения систем ОДУ должен быть устойчивым для устойчивых систем ОДУ и должен быть неустойчивым для неустойчивых систем ОДУ, поэтому методы численного решения систем ОДУ должны быть *AL*-устойчивыми, т.е. абсолютно (***A****-)* устойчивыми строго в левой **(*L**eft*)** полуплоскости комплексной плоскости устойчивости методов численного решения систем ОДУ. Программная реализация этих методов в конечном итоге сводится к многократному решению соответствующих СЛАУ на каждом шаге численного интегрирования, что, как правило, приводит к нескольким тысячам и более обращений к программе-решателю СЛАУ на всем заданном отрезке численного интегрирования. *AL*-устойчивые методы 2-го и 4-го порядков точности были реализованы в программе *DMAN* [2]. Решение большого количества тестовых и практических задач моделирования технических систем и объектов во временной области с помощью этой программы показало, что для получения качественно корректного решения разнообразных систем ОДУ необходимо на всех шагах численного интегрирования обеспечить решение соответствующих тысяч разнообразных СЛАУ с гарантированной точностью в **15** верных значащих цифр (с удвоенной точностью выполнения простых арифметических операций и элементарных математических функций для типа ***double*** представления вещественных чисел алгоритмического языка Си, - именно с этой точностью вычисляются все элементы матрицы коэффициентов этих тысяч разнообразных СЛАУ) для всех элементов вектора решений СЛАУ. Тестирование показало, что в известных программах-решателях СЛАУ эта задача не решена. Итерационные численные методы решения СЛАУ не решают эту проблему, т.к. не могут гарантировать указанную выше точность получаемых решений для всех элементов вектора решений СЛАУ [4]. Нам удалось решить эту проблему с помощью точных, прямых численных методов решения СЛАУ и методов получения сверхточных (extra precision) решений, реализованных в математических пакетах программ Maple (метод Software Floating Point), MATLAB (метод Variable Precision Arithmetic), Mathematica (метод Arbitrary Precision Arithmetic) и др.

Впервые (по сравнению с другими известными нам программами-решателями СЛАУ на языке Си для персональных компьютеров) на обычном персональном компьютере (ОС Win32 XP; компилятор языка Си Visual Studio 2008, версия 9.0.210228) удалось решить плохо обусловленные СЛАУ с указанной выше точностью. Известные нам программы-решатели СЛАУ на языке Си на персональных компьютерах решают с такой точностью только хорошо обусловленные СЛАУ (число обусловленности матрицы коэффициентов близко к 1 или значения диагональных элементов этой матрицы преобладают по абсолютной

величине над недиагональными) и гарантируют не более 5-6 верных значащих цифр для всех элементов вектора решений плохо обусловленных СЛАУ.

Например, для «идеального теста» с матрицей Гильберта 10-го порядка (см. http://pa10.ru/?page_id=109#gilbert), только решатель СЛАУ **LAE-Solver-01** из библиотеки **SADEL** дает решение с 15-ю верными значащими цифрами для **всех** элементов вектора решения этой тестовой задачи.

Библиотека **SADEL** разрабатывается как **дополнительная** математическая библиотека на языке Си (для широко известных математических библиотек стандартных программ на языке Си: ***LINPACK, LAPACK, NAG, Intel MKL, IMSL, MAGMA*** и др.), а также как **дополнительные** программы-решатели (для соответствующих решателей СЛАУ и систем ОДУ из широко известных математических пакетов программ: ***MATLAB, Maple, Mathcad, Mathematica*** и др.) для решения плохо обусловленных СЛАУ, сверхжестких и сверхколебательных систем ОДУ. Например, при математическом моделировании объектов мехатроники, основной параметр в механической части - модуль упругости, который имеет порядок 10^{11}, а основной параметр в электронной части - постоянная времени p-n перехода, которая имеет порядок 10^{-12}, итоговая математическая модель таких объектов в форме системы ОДУ становится сверхжесткой. Соответствующие тестовые задачи в указанных выше математических пакетах не решаются или решаются неверно [6,7,9].

1.2 Математическая постановка задачи решения СЛАУ с гарантированной точностью

Постановка задачи - дана СЛАУ вида:

$$Ax = b, \qquad (1)$$

где $A = (a_{ij})$ – матрица коэффициентов размером $n \times n$, $det(A) \neq 0$, $b = (b1, \ldots, bn)^T$ – вектор-столбец правых частей, $x \in R^{n \times 1}$ – вектор решения СЛАУ (1) , $n \in N$, $A \in R^{n \times n}$ и $b \in R^{n \times 1}$, $rank\ A = n$, все исходные данные заданы с удвоенной точностью представления вещественных чисел языка Си (тип *double*) и нужно вычислить вектор x с гарантированной удвоенной точностью представления вещественных чисел языка Си, т.е. необходимо обеспечить решение СЛАУ вида (1) со **100 процентной гарантированной точностью** в **15** верных значащих цифр (с математической компьютерной машинной точностью выполнения простых арифметических операций и вычисления элементарных функций для стандартного типа ***double*** представления вещественных чисел алгоритмического языка Си) для всех элементов вектора решения СЛАУ (1).

В работе [4] и в лекциях Маничева В.Б. изложены методы и алгоритмы для решения этой задачи, которые реализованы в программах решения СЛАУ в библиотеке SADEL.

Все матрицы в программах библиотеки SADEL вводятся по строкам!

1.3 Математическая постановка задачи решения систем ОДУ с гарантированной корректностью и точностью

Дана система ДАУ общего вида, не разрешенных относительно производных:

$$F(XP,X,Y,t)=0 \quad (2)$$

где *X* - вектор дифференцируемых переменных системы (2) размерностью m; *XP* - вектор производных дифференцируемых переменных по переменной *t* размерностью m, т.е. *XP = dX/dt; Y* - вектор алгебраических переменных системы (2) размерностью *l*; *t* - независимая переменная (например, время); *F* - вектор-функция системы (2) размерностью *n*, где *n=(m+l).* Заданы отрезок интегрирования и начальные условия для вектора дифференцируемых переменных: *X0=X(t0),* где *t0* - начальный момент времени интегрирования. Начальные значения остальных переменных, т.е. *XP0* и *Y0* рассчитываются перед началом интегрирования автоматически. Необходимо вычислить вектора *X(t), XP(t), Y(t)* как функции времени на заданном отрезке интегрирования с относительной точностью не менее 0.001.

В качестве примера рассмотрим систему ОДУ второго порядка в нормальной форме Коши (уравнения Дуффинга):

dx1/dt = x2

*dx2/dt = x1 - 0.2*x2 - x1*x1*x1 + sin(om*t)*

где x1,x2 - дифференцируемые переменные системы ОДУ; *om* - параметр.

Эта система может быть представлена как система ДАУ в общем виде по-разному (в зависимости от методов получения данной математической модели), например:

1-ый вариант:

f1(X) = xp1 - x2 = 0

*f2(X) = xp2 - x1 + 0.2*x2 + x1*x1*x1 - sin(om*t) = 0*

где *F = (f1,f2); X = (x1,x2); XP = (xp1,xp2); xp1 = dx1/dt; xp2 = dx2/dt; m = 2; l = 0; n = 2.*

2-ой вариант:

f1(X) = xp1 - x2 = 0

*f2(X) = xp2 - x1 + 0.2*x2 + y1 – sin (om*t) = 0*

*f3(X) = x1*x1*x - y1 = 0*

где *F = (f1,f2,f3); X = (x1,x2); XP = (xp1,xp2); xp1 = dx1/dt; xp2 = dx2/dt; Y = y1; m = 2; l = 1; n = 3.*

В процессе интегрирования система (1) на каждом шаге интегрирования автоматически дополняется системой:

$$G(XP,X,h)=0, \quad (3)$$

где *h* - шаг интегрирования; *G* - вектор-функция размерностью *m*, которая соответствует выбранному методу интегрирования. Например, для неявного метода Эйлера (метод M1 ниже) вектор-функция *G* имеет вид:

*G = XP - (1/h) * (X - XN) = 0,*

где *XN* - значение вектора дифференцируемых переменных в момент времени *tn*; *tn* - момент времени в начале шага интегрирования *h*; *X* - значение вектора дифференцируемых переменных в текущий момент времени *t; h=(t-tn).*

2. Системные требования

Минимальные системные требования:
Стандартный персональный компьютер, удовлетворяющий системным требованиям установленной на нем операционной системы.

Рекомендуемые системные требования:
Центральный процессор (*CPU*): не ниже *Intel Core 2 Duo*.
Оперативная память: не менее 1 Гб.
Графический процессор (*GPU*): Процессоры компании *NVIDIA* с поддержкой удвоенной точности вычислений с плавающей точкой (***DOUBLE PRECISION***): *GeForce GTX260/275/285/295, Quadro FX3800/4800/5800, Tesla, Fermi*

Сборка библиотеки существует для следующих программных конфигураций:

1. *Microsoft Windows* 32-разрядные ОС, компилятор языка Си *Microsoft Visual Studio 2008 и выше*.
2. *Microsoft Windows* 64-разрядные ОС, компилятор языка Си *Microsoft Visual Studio 2008 и выше*.
3. *Linux* x32, компилятор языка Си *GNU* gcc (примеры и задачи необходимо запускать с опцией –lm для подключения библиотеки math.h).*
4. *Linux* x64, компилятор языка Си *GNU* gcc (примеры и задачи необходимо запускать с опцией –lm для подключения библиотеки math.h).*

* сборки, помеченные звездочкой, доступны не для всех версий библиотеки.

Если Вы хотите использовать библиотеку с другим программным обеспечением, пожалуйста, обращайтесь на наш сайт http://pa10.ru (http://па10.рф) в раздел «Контакты» или пишите на info@pa10.ru.

3. Структура библиотеки

1. Решатель СЛАУ методом Гаусса с полными матрицами **LAE-Solver-0**1
2. Решатель СЛАУ методом прогонки с 3-х диагональными матрицами **LAE-Solver-3diag**
3. Решатель СЛАУ методом *LU* разложения с полными матрицами **LAE-Solver-02**
4. Решатель ДАУ с полными матрицами Якоби **DAE-Solver-01**

3.1. Решатель СЛАУ методом Гаусса: LAE_Solver_01

Описание функции:

```
void lae_solver_01
(
  int *n,      ///< определяет размер системы линейных уравнений
  double A[],  ///< квадратная матрица коэффициентов, размером nxn
  double B[],  ///< вектор правых частей системы
  double X[],  ///< вектор решения системы уравнений
  int *ier     ///< код ошибки:
///< ier=0 - точность в 15 верных значащих цифр для всех элементов вектора
///< решения системы получена
///< ier=1 - требуемая точность не получена или система вырождена
)
```

Обращение к функции:

lae_solver_01(&n, A, B, X, &ier);

Эта функция решает СЛАУ методом Гаусса с полными матрицами: уравнения вида:
$AX = B$,
где n – определяет размер СЛАУ, A - квадратная матрица коэффициентов, размером nxn, B - вектор правых частей системы, X - вектор решения СЛАУ, ier – код ошибки, если ier=0 – вектор решения X получен с точностью в **15** верных значащих цифр для всех элементов вектора решения СЛАУ, если ier=1 - точность в **15** верных значащих цифр для всех элементов вектора решения СЛАУ не получена или система вырождена.

Ограничение демо-версии**: не более 16 уравнений.**

3.2. Решатель СЛАУ с 3-х диагональными матрицами: LAE_Solver_3diag

Описание функции:

```
void lae_solver_3diag(
  int *n,      ///< размер системы уравнений
  double A[],  ///< трехдиагональная матрица коэффициентов, размером nx3
///1-й столбец - нижняя диагональ (A(1,1)=0.0),
///2-й столбец - основная диагональ,
///3-й столбец - верхняя диагональ(A(n,3)=0.0),
  double B[], ///< вектор правых частей системы
  double X[],  ///< вектор решения системы уравнений
   int *ier   ///< код ошибки:
///< ier=0 - точность в 15 верных значащих цифр для всех элементов вектора
///< решения системы получена
///< ier=1 - требуемая точность не получена или система вырождена
)
```

Обращение к функции:

lae_solver_3diag(&n, A, B, X, &ier);

Эта функция решает СЛАУ методом прогонки с 3-х диагональными матрицами: уравнения вида:

$AX = B$,

где n – определяет размер СЛАУ, A - матрица коэффициентов, размером $nx3$, 1-й столбец - нижняя диагональ (A(1,1)=0.0), 2-й столбец - основная диагональ, 3-й столбец - верхняя диагональ (A(n,3)=0.0), B - вектор правых частей системы, X - вектор решения СЛАУ, ier – код ошибки, если ier=0 – вектор решения X получен с точностью в **15** верных значащих цифр для всех элементов вектора решения СЛАУ, если ier=1 - точность в **15** верных значащих цифр для всех элементов вектора решения СЛАУ не получена или система вырождена.

Ограничение демо-версии**: не более 200 уравнений.**

Программа-решатель для СЛАУ с 3-х диагональными матрицами разработана по просьбе посетителей сайта. Эта программа решать с машинной гарантированной точностью СЛАУ с 3-х диагональными матрицами любой размерности, ограниченной только возможностями компьютера. Даже на ПК можно получать миллионы элементов вектора решения с удвоенной точностью (*double precision*) для всех элементов вектора решения СЛАУ, - такая точность недоступна для итерационных методов решения СЛАУ, используемых в настоящее время. Так как для 3-х диагональных матриц накопление погрешности для прямых, точных методов решения СЛАУ не происходит, то указанная машинная точность решения гарантируется независимо от размерности систем.

На этом же принципе построен решатель для произвольных разреженных СЛАУ, это снимет «проклятие огромной размерности решаемых СЛАУ», о которой нас предупреждают опытные расчетчики.

3.3. Решатель СЛАУ методом LU разложения с полными матрицами: LAE_Solver_02

Описание функции:

```
void lae_solver_02(
  int *n,      ///< определяет размер системы линейных уравнений
  double A[],  ///< квадратная матрица коэффициентов, размером nxn
  double B[],  ///< вектор правых частей системы
  double L[],  ///< нижнетреугольная L матрица разложения
  double U[],  ///< верхнетреугольная U матрица разложения
  double X[],  ///< вектор решения системы уравнений
  int *ier     ///< код ошибки: ier=0 - точность в 15 значащих цифр для всех ///< ier=0 - точность в 15 верных значащих цифр для всех элементов вектора
  ///< решения системы получена
  ///< ier=1 - требуемая точность не получена или система вырождена
  )
```

Обращение к функции:

lae_solver_02(&n, A, B, L, U, X,&ier);

Эта функция решает СЛАУ методом *LU* разложения с полными матрицами: уравнения вида:

$AX = B$,

где n – определяет размер СЛАУ, A - квадратная матрица коэффициентов, размером nxn, B - вектор правых частей системы, X - вектор решения СЛАУ, также пользователю выдается результат разложения – матрицы L и U, ier – код ошибки, если ier=0 – вектор решения X получен с точностью в **15** верных значащих цифр для всех элементов вектора решения СЛАУ, если ier=1 - точность в **15** верных значащих цифр для всех элементов вектора решения СЛАУ не получена или система вырождена.

Ограничение демо-версии**: не более 16 уравнений.**

3.4. Решатель ДАУ с полными матрицами Якоби: DAE_Solver_01

Описание функции:

```
void dae_solver_01(double z[],double xp[],double z1[],double xp1[],double f[],
                   double rj1[],double rj2[],
                   double t, double t0,double tk,double h,double hmn,
                 double hmx,double eps,double *tkv,
                   int n,int m,int nm,int ncon,int *nbad,int *ier,int ip[],
                   void fct(double z[],double xp[],double f[],
                            double rj1[],double rj2[],int n,int m,double t,double h, int
                     ncon,int *nbad,int ip[]),
                   void out(double z[],double xp[],int n,int m,double t,
                         double t0,double tk,double h,double *tkv,
                         int ncon,int ip[])
);
```

В решателе dae_solver_01 реализованы 3 метода интегрирования систем ДАУ:

M1 - A-устойчивый неявный метод Эйлера первого порядка точности;

M2 – AL-устойчивый (устойчивый строго в левой полуплоскости комплексной плоскости устойчивости методов интегрирования) неявный метод второго порядка точности из работы [1];

M3 - AL-устойчивый неявный метод четвертого порядка точности из работы [2];

Во всех методах на каждом шаге интегрирования оценивается относительная, локальная погрешность интегрирования для каждой дифференцируемой переменной (по отношению к максимальному абсолютному значению каждой переменной на заданном отрезке интегрирования). Вычисленная погрешность сравнивается с заданной погрешностью eps и если она ее превышает, то шаг интегрирования h уменьшается. Максимальное абсолютное значение каждой дифференцируемой переменной определяется в ходе интегрирования автоматически, эти значения можно также задать перед началом интегрирования.

Во всех методах на каждом шаге интегрирования решается система нелинейных алгебраических уравнений относительно векторов переменных *X, XP, Y* методом Ньютона, для которого необходимо вычислять матрицу Якоби для системы (2) и системы (3). Матрица Якоби для системы (3) вычисляется в решателе dae_solver_01 автоматически. Матрица Якоби для системы (2) должна вычисляться в специальной подпрограмме пользователя fct, наряду с вычислением вектор-функции *F*. Эта матрица состоит из двух подматриц:

RJ = (RJ1, RJ2)

где *RJ* - матрица Якоби размером (*n*n1); n1=m+n. RJ1* - матрица частных производных вектор-функции *F* по переменным XP размером (*n*m),* т.е. матрица *RJ1 = [dF/dXP]*. *RJ2* - матрица частных производных вектор-функции *F* по переменным z размером *(n*n),* т.е. матрица *RJ2 = [dF/dz],* где z - вектор переменных системы ОДУ размерностью *n*, который включает дифференцируемые переменные и алгебраические переменные системы (2), т.е. z = *(x,y).*

Для приведенного выше примера (уравнения Дуффинга) матрица Якоби и соответствующие подматрицы имеют вид:

1-ый вариант:

1	0	0	-1
0	1	-1+3*x1*x1	0.2

2-ой вариант:

1	0	0	-1	0
0	1	-1	0.2	1
0	0	3*x1*x1	0	-1

Начальным приближением для метода Ньютона являются значения переменных в начале шага интегрирования, поэтому, если на некотором шаге нет сходимости итераций, то текущий шаг h уменьшается вплоть до минимального hmin, после чего выдается сообщение об ошибке (обычно из-за ошибок в формулах для определения элементов матрицы Якоби в подпрограмме пользователя или из-за расходимости самой системы ДАУ).

Обращение к функции (на примере теста test00):

dae_solver_01(z,xp,z1,xp1,f,rj1,rj2,t,t0,tk,h,hmn,hmx,eps,&tkv,n,m,nm,ncon,&nbad,&ier,ip,fct00, out00);

z - массив переменных системы ДАУ размерности n, в первых m элементах этого массива размещаются дифференцируемые переменные x системы ДАУ, в остальных элементах размещаются алгебраические переменные, поэтому всегда должно выполняться условие (n >= m). Перед началом интегрирования в первых m элементах этого массива размещаются начальные значения дифференцируемых переменных – *x0*.

xp - массив производных дифференцируемых переменных по времени размерностью m, т.е. xp=dx/dt.

z1 - рабочий массив размерностью n, перед началом интегрирования в первых m элементах этого массива размещаются максимальные абсолютные значения дифференцируемых переменных. Если они известны, то их считывают из файла, либо присваивают соответствующим значениям массива z1. Если они неизвестны, то в этот массив пересылаются абсолютные величины начальных значений дифференцируемых переменных – *abs(x0).*

xp1 - рабочий массив размерностью m.

f - массив размерностью n для вычисления вектор-функции *F* системы ДАУ в подпрограмме fct.

rj1 - матрица размером (n*m) для вычисления частных производных вектор-функции *F* по переменным xp в подпрограмме fct, т.е. rj1(1,1) = df(1)/dxp(1), rj1(1,2) = df(1)/dxp(2) и т.д.

rj2 - матрица размером (n*n) для вычисления частных производных вектор-функции *F* по переменным z в подпрограмме fct, т.е. rj2(1,1) = df(1)/dz(1), rj2(1,2) = df(1)/dz(2) и т. д. В программе предусмотрена возможность численного вычисления элементов этой матрицы. Если перед обращением к программе установить параметр ier=-1, то все элементы матрицы rj2 будут вычисляться автоматически методом приращений. Однако этот метод вычисления частных производных следует применять только на этапе отладки математических моделей. Общее правило – все, что можно вычислить аналитически (особенно линейные функции) следует вычислять аналитически. Для

сложных функциональных зависимостей следует использовать численное вычисление частных производных покомпонентно (см. лекции Маничева В.Б.).

t - текущее время интегрирования.

t0 - заданное время начала интегрирования.

tk - заданное время окончания интегрирования.

h - текущий шаг интегрирования.

hmn - заданный минимальный шаг интегрирования.

hmx - заданный максимальный шаг интегрирования.

eps - заданная относительная погрешность интегрирования, одинаковая для всех переменных (число верных значащих цифр в решении, обеспечиваемых для каждой дифференцируемой переменной).

tkv - переменная, с помощью которой можно точно табулировать задаваемые пользователем моменты времени в ходе интегрирования. Эта переменная устанавливается в подпрограмме пользователя out и обычно используется для табуляции результатов. Например, если надо получить значения переменных в целочисленные моменты времени 0, 1, 2 и т.д., то в подпрограмму out следует вставить следующие операторы, обнулив предварительно вспомогательную переменную tp (test05):

```
if(t==tp)  fprintf(f01,"   %e  %e  %e\n",t,z[1],z[2]);
 if(t>=tp) tp=tp+1e0;
 if (tp< *tkv) {if (tp>=t) *tkv=tp;}
```

ar - рабочий массив размером не менее, чем (3*n*m+5*n*n+9*n+25*m+3), в первых трех элементах этого массива размещаются максимальные (среди всех дифференцируемых переменных) значения относительных погрешностей интегрирования для методов M1, M2 и M3 соответственно. В последующих m элементах этого массива размещаются максимальные абсолютные значения дифференцируемых переменных.

n - общее число уравнений в системе ДАУ.

m - количество дифференцируемых переменных в системе ДАУ.

nm - заданный номер метода интегрирования:

nm = 1 - метод M1;

nm = 2 - метод M2;

nm = 3 - метод M3;

ncon - ключ расчета начальных значений переменных:

ncon = 0 - выполняется расчет начальных значений переменных *xp0* и *y0* при заданном *x0* для момента времени t0 от нулевых значений *xp0* и *y0*. Значения *x0* соответствуют исходным значениям первых m элементов массива z. До начала интегрирования с этим значением ключа ncon выполняется одно обращение к подпрограммам fct и out для выполнения однократных вычислений параметров, которые в ходе интегрирования не изменяются. После выполнения однократных вычислений следует установить ncon=1 в подпрограмме fct.

ncon = 1 - расчет начальных значений переменных xp и y0 не производится. Этот признак следует использовать в случаях повторных обращений к решателю dae_solver_01, если надо продолжить интегрирование решаемой системы ДАУ. До этого должно быть хотя бы одно обращение к решателю dae_solver_01 с ncon=0 или с ncon=2.

При нормальном выходе из решателя dae_solver_01устанавливается ncon=1.

ncon = 2 - выполняется расчет начальных значений переменных xp0 и y0 при заданном x0 для момента времени t0 от вводимых начальных значений xp0 и y0. Эти значения либо вводятся из заранее сформированного файла исходных данных, либо присваиваются вычисленным заранее значениям этих переменных. Значения x0 соответствуют исходным значениям первых m элементов массива z.

nbad – (принципиально новый, важнейший параметр) - ключ уменьшения шага

интегрирования, устанавливаемый в подпрограммах вычислений элементов вектор-функции *F* ДАУ. Перед обращением к моделям устанавливается nbad=0 на каждой итерации. Если отдельные переменные в моделях принимают значения, которые могут привести к останову вычислений (корень из отрицательного числа, превышение допустимого порядка в степенных функциях (test06) и т.п.), то необходимо установить nbad=1, тогда итерации будут прерваны, произойдет уменьшение шага интегрирования до тех пор, пока можно будет продолжать вычисления, либо до значения минимального шага с сообщением о не сходимости итераций. Если в моделях установить nbad=2, то произойдет уменьшение шага до учетверенного минимального с дальнейшим продолжением итераций с этим шагом (фактически расчет с новыми начальными условиями). nbad=2 можно установить только в том случае, если nbad не равно 1. nbad=2 следует использовать для идентификации точек разрыва производных в кусочно-нелинейных функциях (test07). Подробнее см.лекции Маничева В.Б.

ier - код ошибки:

ier = 0 - нет ошибок;
ier = 1 - m отрицательно или больше n;
ier = 2 - hmn отрицательно или больше hmx;
ier = 3 – t0 больше tk;
ier = 4 - eps меньше 1e-12;
ier = 5 - nm меньше 0, или больше 3;
ier = 6 - ncon меньше 0, или больше 2;
ier = 7 - нет сходимости итераций при решении нелинейных алгебраических уравнений на минимальном шаге интегрирования hmin. Следует проверить правильность формул для вычисления элементов матриц rj1 и rj2, или уменьшить заданный минимальный шаг интегрирования, или проверить не расходится ли система ДАУ .
ier = 8 - Вырожденность системы линейных алгебраических уравнений при выполнении Ньютоновских итераций.
ier=-1 – Это значение параметра используется для установки численного вычисления элементов матрицы rj2 при первом обращении к решателю.

ip - рабочий массив размером не менее, чем (20+2*n+6*m), в первых 20-ти элементах этого массива размещаются специальные счетчики и ключи для методов интегрирования:

ip(1) - номер итерации (начиная с 0) при решении системы нелинейных алгебраических уравнений на текущем шаге интегрирования (не более 10-ти).
ip (2) - суммарное количество итераций при решении нелинейных алгебраических уравнений на всех выполненных и аннулированных шагах интегрирования.
ip (3) - суммарное количество выполненных шагов интегрирования.
ip (4) - суммарное количество делений шага интегрирования с отменой выполненного предыдущего шага, т.е. суммарное количество фактически аннулированных шагов.

Условно аннулированными называются шаги h = hmin, которые выполняются несмотря на то, что текущая погрешность превышает заданную при nbad=2.

ip (5) - суммарное количество аннулированных шагов из-за превышения заданной погрешности eps, включая условно аннулированные.
ip (6) - суммарное количество аннулированных шагов из-за не сходимости итераций при решении алгебраических уравнений на каждом шаге интегрирования, включая условно аннулированные.
ip (7) - суммарное количество выполненных итераций по методу M2.
ip (8) - суммарное количество выполненных шагов интегрирования по методу M2.
ip (9) - суммарное количество выполненных итераций по методу M3.
ip (10) - суммарное количество выполненных шагов интегрирования по методу M3.
ip (11) – плохая обусловленность системы..
ip (12) – количество коррекций производных для методов 2 и 4 порядка точности.

ip (13) – уменьшение шага из-за prwar и acr.
ip (14) – уменьшение шага из-за nbad=1.
ip (15) - уменьшение шага из-за превышения числа итераций больше 10.
ip (16) -уменьшение шага из-за того, что невязка и приращения не уменьшаются.
ip (17)}
... } - резерв счетчиков и ключей.
ip (20)}
ip (21) }
... } - рабочий массив для подпрограммы решения
... } системы линейных алгебраических уравнений.
ip (20+2*n+6*m)}

fct - имя внешней подпрограммы пользователя. В этой подпрограмме пользователь вычисляет вектор-функцию *F* решаемой системы ДАУ и матрицы rj1 и rj2 частных производных df/dxp и df/dz. Нулевые значения элементов матриц rj1 и rj2 в подпрограмме fct присваивать не нужно. К подпрограмме fct происходит обращение на каждой итерации. При первом обращении к решателю (ключ ncon=0 или ncon=2) организовано одно обращение к подпрограмме fct при t=t0 и h=hmin для выполнения однократных вычислений параметров, не изменяющихся в ходе интегрирования. После выполнения однократных вычислений следует установить ключ ncon=1. При каждом обращении в массив f заносится рекомендуемый вектор приращений dz для переменных z с целью возможности численного вычисления частных производных df(i)/dz(j) = (fi(zj+dzj)-fi(dzj))/dzj; Эту возможность следует использовать только при невозможности аналитического вычисления частных производных. Подробнее в лекциях Маничева В.Б.

Заголовок функции fct:

void fct(double z[],double xp[],double f[],double rj1[],double rj2[],int n,int m,double t,double h,int ncon,int *nbad,int ip[]);

out - имя внешней подпрограммы пользователя для вывода и обработки результатов интегрирования. Обращение к этой подпрограмме происходит:

а) после расчета начальных значений переменных xp0 и y0 для момента времени t0. При этом перед обращением к подпрограмме out устанавливается ключ ncon=0. Это можно использовать для однократных вычислений параметров подпрограммы out, которые не изменяются в ходе интегрирования.

б) после каждого успешно выполненного шага интегрирования;

в) в случае ошибки (ier не равно 0).

Заголовок функции out:

void out(double z[],double xp[],int n,int m,double t,double t0,double tk,double h,double tkv,double ar[],int ncon,int ip[]);

Задаваемые параметры интегрирования

Перед обращением к программе должны быть заданы значения следующих параметров, ключей и переменных:

n - общее число уравнений в системе ДАУ;
m - количество дифференцируемых переменных в системе ДАУ;
t0 - заданное время начала интегрирования;
tk - заданное время окончания интегрирования;
hmn - заданный минимальный шаг интегрирования;
hmx - заданный максимальный шаг интегрирования;
eps - заданная относительная погрешность интегрирования, одинаковая для всех дифференцируемых переменных;

nm - заданный номер метода интегрирования;
ncon - ключ расчета начальных значений переменных;
ier = -1 – ключ для установки автоматического численного вычисления элементов матрицы rj2.
z(1)...z(m) - начальные значения дифференцируемых переменных.
z1(1)...z1(m) - либо максимальные абсолютные значения дифференцируемых переменных, либо z1(i)=abs(z(i)).

Ограничение демо-версии**: не более 16 уравнений.**

4. Рекомендации по тестированию

Библиотека SADEL распространяется до полного завершения разработки всех решателей. Мы предлагаем Вам проверить нашу библиотеку на своих задачах, чтобы к моменту выхода полной версии представлять себе возможности SADEL. Мы ввели ограничения на размерность задач, чтобы примеры оставались тестовыми. Если Вы хотите проверить наши решатели на более сложных задачах, пожалуйста, обращайтесь по адресу info@pa10.ru.

4.1. Тестирование решателей СЛАУ

На нашем сайте (http://pa10.ru/?page_id=109) приведены результаты сравнения программ решения СЛАУ данной библиотеки с другими решателями СЛАУ при решении 3-х плохо обусловленных СЛАУ (класс “некорректных” задач [3]). Эти и другие задачи (всего 12 тестовых задач) будут включены в файлы библиотеки для скачивания.

Рекомендации для сравнения SADEL с используемыми у Вас решателями СЛАУ:

1. Рассчитать все тестовые задачи (12 тестовых задач, описанных в файле LAE-examples.pdf) с помощью программ решения СЛАУ, которые Вы используете. Сравнить полученные решения с абсолютно точными решениями для этих задач, особенно для тестовой задачи 02 Маничева для вырожденных СЛАУ (СЛАУ с почти вырожденной матрицей - абсолютно точное решение этой задачи, полученное методом Крамера: x1=0; x2=1.5; решатели СЛАУ из математических пакетов программ дали следующие результаты решения этой тестовой задачи: MATLAB: x1=0.090; x2=1.450; Mathcad: x1=0.043; x2=1.478; Maple: решение получить не удалось; Mathematica с предупреждением о возможном неправильном решении: x1=0.043; x2=1.478;).

2. Рассчитать с помощью программ библиотеки Ваши тестовые задачи СЛАУ - особенно интересуют задачи, для которых будет выдан код ошибки ier=1.

3. Формировать Ваши тестовые задачи СЛАУ с известным, абсолютно точным решением по следующему алгоритму Уилкинсона:

   А. Задать размер, например, 3x3 и коэффициенты матрицы *A* по строкам, например, (3.0 5.0 7.0, 2.0 4.0 6.0, 1.0 -2.0 4.0).

   Б. Задать вектор неизвестных *X,* например, (1.0 2.0 3.0).

   В. Умножить матрицу *A* на вектор *X* и получить вектор *B*, в рассматриваемом примере вектор *B* будет (34.0 28.0 9.0).

   Г. Решить сформированную задачу с помощью программ библиотеки и проверить получение 15 верных значащих цифр для всех элементов вектора *X*.

   Примечание:

   Не задавать такие численные значения коэффициентов матрицы *A* и вектора *X* в тестовых задачах, для которых элементы вектора *B* будут содержать в общем количестве более, чем 15-ть десятичных значащих цифр! Программы библиотеки не гарантируют точность в 15 верных значащих цифр после десятичной точки (accuracy), а гарантируют точность общего количества значащих цифр, независимо от положения десятичной точки (precision)!

Результаты тестирования, вопросы, замечания, пожелания и предложения присылайте на адрес info@pa10.ru.

Если у Вас что-то не получилось или, напротив, Ваши расчеты показали некорректность работы наших алгоритмов, пожалуйста, напишите нам об этом!

4.2. Тестирование решателей ДАУ

В решателе dae_solver_01 реализованы принципиально новые методы и алгоритмы, изложенные в работах [1,2] и в лекциях Маничева В.Б. Результаты тестирования (тестовые задачи test01-test04) и сравнение с лучшими решателями для решения жестких систем ОДУ из пакетов **MATLAB, Maple, Mathcad** приведены в работе [7]. Эти и другие задачи (всего 10 тестовых задач) включены в файлы библиотеки для скачивания. Рассчитайте все эти тестовые задачи с помощью программ решения систем ОДУ, которые Вы используете, и сравните полученные решения с корректными, достоверными и точными решениями полученными для этих задач с помощью программ библиотеки SADEL (все тестовые задачи были решены за 6сек. на Intel i5, 2Gb, Win7-x64&VS2010). Результаты сравнения будут с благодарностью приняты. Графики различных зависимостей для рассчитанных переменных можно получать с помощью программы pa10_view.exe, входящей в архив с библиотекой. Эта программа разработана только для построения графиков функций, полученных нашим решателем dae_solver_01:

Формат файла результатов для этой программы:

1-я строка файла результатов - заголовок задачи, (запятая) время, (запятая) и далее через запятую имена ровно 4-х переменных.

Следующие строки - это результаты решения ДАУ на каждом шаге интегрирования.

Примеры вывода результатов расчета переменных ДАУ в этом формате приведены в тестах: файлы grtest01, grtest02 …

В данной версии решателя dae_solver_01 реализован классический, устаревший алгоритм выбора шага интегрирования, поэтому заданная точность не гарантируется за однократный расчет. Для гарантии надо выполнять многократные расчеты (желательно и с решателями из других библиотек) с разными параметрами точности eps *(вплоть до максимально допустимой) разными методами интегрирования и добиваться совпадения результатов расчета. Новые алгоритмы будут реализованы позднее.*

Описания тестовых задач со скриншотами графиков результатов решения тестовых задач и рекомендации по использованию решателя dae_solver_01 будут доступны позднее.

5. Список литературы

1 Д.М. Жук, В.Б. Маничев, А.О. Ильницкий Методы и алгоритмы решения дифференциально-алгебраических уравнений для моделирования систем и объектов во временной области. // Информационные технологии. - 2010. – часть1 - №7, часть 2 - №8.

2. Жук Д.М., Маничев В.Б. Программа *DMAN* для решения дифференциально-алгебраических уравнений, номер государственной регистрации 2009612666 от 27 мая 2009.

3. Ю.П. Петров Обеспечение достоверности и надежности компьютерных расчетов. – СПб.: БХВ-Петербург, 2008. - 160 с.

4. В.Б.Маничев, В.Н.Глазкова, Д.Ю.Кожевников, Д.А.Кирьянов, М.К.Сахаров Решение систем линейных алгебраических уравнений с удвоенной точностью вычислений на языке Си. 10 марта 2010 г. статья была сдана в редакцию журнала «Вестник МГТУ», в октябре 2010 г. рекомендована к опубликованию.

5. Тыртышников Е.Е. Методы численного анализа : учеб. пособие для студ. вузов / — М.: Издательский центр «Академия», 2007. — 320 с. — (Университетский учебник. Сер. Прикладная математика и информатика).

6. Маничев В.Б. Метод тестирования программ, реализующих неявные методы интегрирования ОДУ. Тезисы докладов международной конференции «Современные проблемы вычислительной математики и математической физики», Самарский-2009, Москва, МГУ имени Ломоносова, июнь 2009.

7. Д.М. Жук, В.Б. Маничев, А.В. Андронов Платформа математического моделирования во временной области разнородных технических систем и объектов FMS PA10. В сб. научных трудов Всероссийской научно-технической конференции "ПРОБЛЕМЫ РАЗРАБОТКИ ПЕРСПЕКТИВНЫХ МИКРО- и НАНОЭЛЕКТРОННЫХ СИСТЕМ – 2010 (МЭС - 2010) - М.: ИППМ РАН, 2010.

8. Вержбицкий В.М. Основы численных методов. Учебник для вузов. – М.: Высш. шк., 2002.-840 с.

9. Андронов А.В. Математическое моделирование систем, процессов и явлений во временной области. CAD/CAM/CAE Observer # 7(59)/2010.

10. Ю. П. Петров, Л. Ю. Петров Неожиданное в математике и его связь с авариями и катастрофами.- Издательство: БХВ-Петербург, 2005 г. 234 стр.

11. Ю. П. Петров Записки профессора. - Издательство: БХВ-Петербург, 2010 г. 176 стр.