SCILAB: Библиотеки

Глава 4. Дифференциальные уравнения

Содержание главы:

- Краткий русско-английский словарь употребляемых терминов дифференциального исчисления
- С помощью каких команд можно решить дифференциальное уравнение?
- Солвер ode для решения обыкновенного дифференциального уравнения
- Как установить режим вычислений с помощью для солвера дифференциальных уравнений ode?
- Как решить уравнение Риккати (Riccati)?

Приближенное определение производных функций должно всегда проводиться с особой осторожностью, так как известно, что дифференцирование, в противоположность интегрированию, сопровождается "разбалтыванием". Это означает, что у производной ухудшается свойство гладкости.

Краткий русско-английский словарь употребляемых терминов дифференциального исчисления

Обыкновенные дифференциальные уравнения (ОДУ)	**Ordinary Differential Equation (ODE)**
Правая часть дифференциального уравнения	**Right Hand Side (RHS)**
Уравнение дифференциальное жесткое	**stiff differential equation**
Жесткая задача	**stiff problem**
Дифференциальное алгебраическое уравнение	**Differential Algebraic System Solver = dassl**

С помощью каких команд можно решить дифференциальное уравнение?

Способ 1.
С помощью команды **ode**, которая является солвером для решения обыкновенного дифференциального уравнения.

Способ 2.
С помощью команды **odedc**, которая вычисляет решение смешанной дискретно-непрерывной системы. Смотри подробно **help odedc**.

Способ 3.
Команда **dassl**, которая дает решение неявно выраженного дифференциального уравнения:
g(t,y,ydot)=0
y(t0)=y0 и **ydot(t0)=ydot0**

Смотри подробно **help dassl**. Детальный пример дан в файле
`SCIDIR/tests/dassldasrt.tst`

Способ 4.
С помощью команды **impl**, которая дает решение неявно выраженного линейного дифференциального уравнения
A(t,y) dy/dt=g(t,y), **y(t0)=y0**

Смотри подробно **help impl**. Смотри также пример в файле
`SCIDIR/routines/default/Ex-impl.f .`

Солвер ode для решения обыкновенного дифференциального уравнения

Солвер служит для решения обыкновенного дифференциального уравнения (ОДУ)
dy/dt=f(t,y) , y(t0)=y0

Синтаксис
y=ode(y0,t0,t,f)
[y,w,iw]=ode([type],y0,t0,t [,rtol [,atol]],f [,jac] [,w,iw])
[y,rd,w,iw]=ode("root",y0,t0,t [,rtol [,atol]],f [,jac],ng,g [,w,iw])
y=ode("discrete",y0,k0,kvect,f)

Параметры
y0 : действительное число или матрица (условия инициализации)
t0 : действительный скаляр (время инициализации)
t : действительный вектор. Задает интервал времени для которого вычисляется решение уравнения.
f : внешний параметр (функция или строка или список).
type : строковая переменная принимающая одну из следующих значений:
"adams" "stiff" "rk" "rkf" "fix" "discrete" "roots"

rtol, **atol** : действительные константы и действительные константы того же размера, что и **y**.
jac : внешний параметр (функция или строка или список).
w, **iw** : действительные вектора.
ng : целое число.
g : внешний параметр (функция или строка или список).
k0 : целое число (начальное время).
kvect : целочисленный вектор.
Команды **ode** является стандартной функцией для решения явной "ODE" системы определенной следующим образом:

dy/dt=f(t,y), y(t0)=y0.
Имеется интерфейс для разнообразных солверов, в частности для **ODEPACK**. (ODEPACK=Differential Eguations PACKage. Это пакет для решения алгебраических дифференциальных уравнений.)

Способ решения проблемы и метод зависят от значения первого опционного аргумента, который может принимать значения из следующего списка:

: по умолчанию будет вызываться солвер пакета ODEPACK. Автоматически пакетом

производится выбор между методом прогноза и коррекции (nonstiff predictor-corrector) для нежестких задач и методом BDF (Backward Differentiation Formula) Адамса-Мултона для жестких задач. Изначально используется метод для нежесткой задачи и динамический контроль результатов для того, чтобы решить, какой из методов следует использовать.

"adams": Используется для нежесткой задачи. Вызывается **lsode** солвер пакета ODEPACK и используется метод Адамса.

"stiff": Используется для жесткой задачи (stiff problems). Вызывается lsode solver пакета ODEPACK и используется метод BDF (Backward Differentiation Formula). Этот метод схож с методом Адамса-Мултона (Adams-Moulton).

"rk": Используется адаптивный метод Рунге-Кутта (Runge-Kutta) 4-го порядка (RK4).

"rkf": Используется программа Shampine и Watts, основанная на методе Рунге-Кутта-Фельберга (Fehlberg's Runge-Kutta) 4-5-го порядка точности (RKF45) с автоматической оценкой ошибки. Этот опцион для нежестких (non-stiff) и умеренно-жестких (mildly stiff) задач, если оценка производной не важна. Этот метод не следует применять , если пользователю требуется высокая точность.

"fix": Используется тот же солвер, что и с значением параметра "rkf", но с очень простым интерфейсом для пользователя, т.е. необходимо задавать солверу только значение параметров. Это самый простой метод для начала.

"root": Используется ODE солвер со способностью нахождения корней. Используется **lsodar** солвер пакета ODEPACK. Это вариант **lsoda** солвера, в котором он находит корни заданного вектора функции. Подробно смотри **help ode_root**.

"discrete": Моделирование дискретного времени (Discrete time simulation). Подробно смотри **help de_discrete**.

Здесь мы описываем только использование команды **ode** для стандартных явных ODE систем. Простейщий вариант: **y=ode(y0,t0,t,f)**, где **y0** является вектором начальных условий; **t0** - начальное время; **t** -вектор времени, за которое вычисляется решение **y** и **y**-вектор решений **y=[y(t(1)),y(t(2)),...]**. Функция **f** для команды **ode** определяется как внешняя, т.е. определена как scilab-функция или является именем Fortran или C функции, вызываемая соответствующим образом или задана списком (**list**). Если **f** является функцией, то синтаксис должен быть следующий:
ydot = f(t,y)
где **t** действительный скаляр (время) и **y** -действительный вектор (положение). Эта функция является правой частью (RHS=right hand side) дифференциального уравнения **dy/dt=f(t,y)**. Если **f** задано символьной переменной, она отсылает к имени Fortran процедуры или C функции. Например, для команды `ode(y0,t0,t,"fex")`, `fex` будет именем вызываемой подпрограммы. Эту подпрограмму следует вызвать с помощью конструкции **f(n,t,y,ydot)**. Внешняя подпрограмма может быть динамически линкована в Scilab. (Смотри главу 6: Интерфейс между программами, написанными на языках C и Fortran и пакетом Scilab).

Примеры таких программ находятся в файлах `SCIDIR/routines/default/README` и `SCIDIR/routines/default/Ex-ode.f`.
Аргумент **f** может быть также задан списком. Для конструкции `ode(y0,t0,t,lst)`

параметр `lst` должен быть списком со следующей структурой:
lst=list(f,u1,u2,...un)
ydot = f(t,y,u1,u2,...,un)
Это позволяет использовать параметры в качестве аргументов функции **f**. Функция **f** может возвращать вместо вектора матрицу размера **p** на **q**. Подробно смотрите **help ode**.

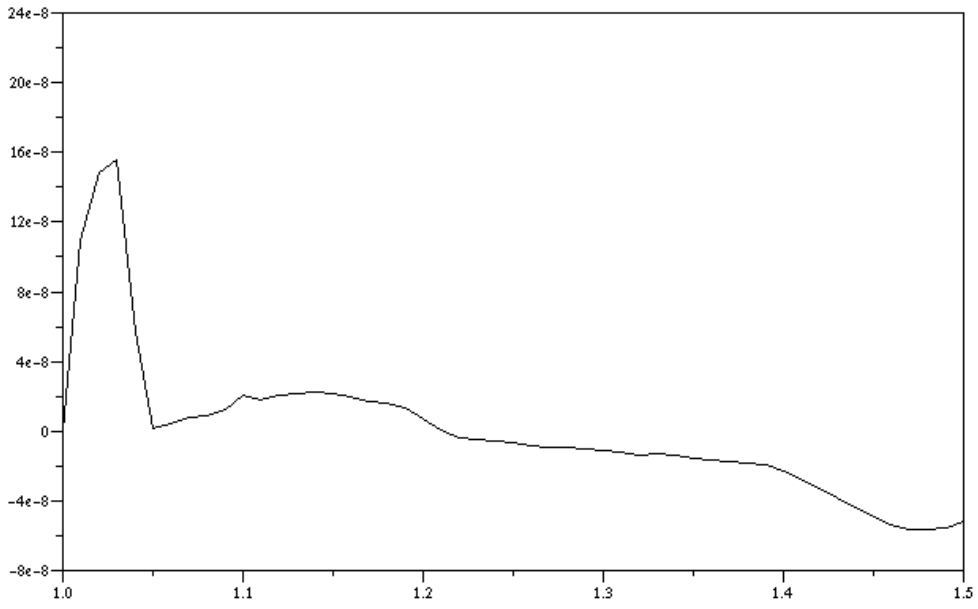
Замечание: Используя переменную **%ODEOPTIONS**, можно задать еще больше опционов для ODEPACK солверов. Подробно смотрите **help odeoptions**.

Пример 1.
Получим численное решение дифференциального уравнения `dy/dt= f(t,y)`, где $f=t*y^{(1/3)}$ с начальные условия `t0=1` и `y0=1`. Мы хотим получить численное решение этого уравнения в интервале значений `t` от `tmin=1` до `tmax=5` с шагом `dt=0.01`.

Поскольку известно, что это уравнение имеет точное аналитическое решение $y(t)=((t^2+2)/3)^{(3/2)}$, мы сможем проверить полученный с помощью пакета Scilab результат.

```
y0=1;
t0=1;
t=1:0.01:1.5;
deff("[ydot]=f(t,y)","ydot=y^(1/3)*t")
y=ode(y0,t0,t,f);
//
y_exact=((t^2+2)/3)^(1.5); // это функция точного решения для сравнения
my_er=y-y_exact;
plot(t,y-y_exact) // это график ошибки вычисления от аргумента t
```

Результат:



Важное замечание:
Начальные условия должны быть заданы в точке слева от минимальной координаты в интервале для которого мы находим численное решение дифференциального уравнения, то есть `t0<=tmin`. В противном случае (например, при начальных условиях `t0=1` и `y0=1` хотим найти решение в интервале `t=0.8:0.1:5`), вычисления не производится и поступает

сообщение об ошибке, например, такого рода:

```
intdy-- t (=r1) illegal
where r1 is : .1000000000000E+00
t n est pas entre tcur - hu (= r1) et tcur (=r2)
where r1 is : .6989745878701E-01 and r2 : -.2841502812730E-01
lsoda-- problems due to intdy. itask=i1,tout=r1
where i1 is : 1
where r1 is : .1000000000000E+00
!--error 9999
illegal input
```

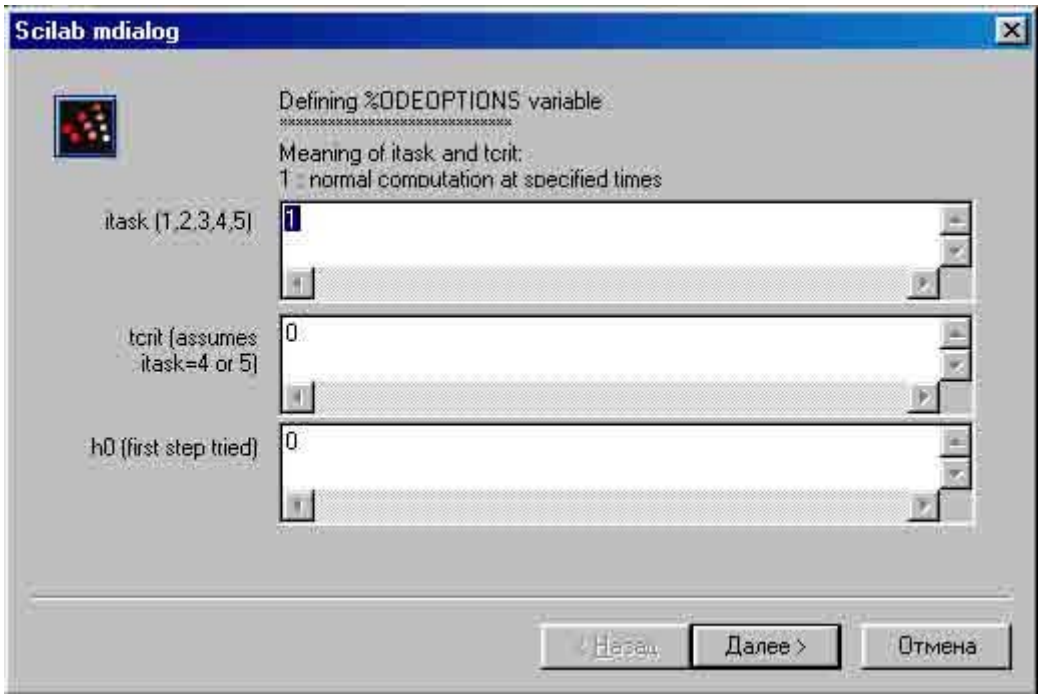
Замечание: Команда **ode** может иметь очень много разнообразных параметров. Часто целесообразно устанавливать режим солвера дифференциальных уравнений с помощью команды **odeoptions**.

Как установить режим вычислений с помощью для солвера дифференциальных уравнений ode?

С помощью команды **odeoptions**.
Синтаксис
odeoptions()

Замечание: Команда **odeoptions()** должна быть набрана обязательно строчными буквами, а не заглавными, в то время как название переменной **%ODEOPTIONS** состоит из заглавных букв.

После выполнения команды **odeoptions()** нам будет предложена в диалоговом режиме последовательность форм, заполнив которые, мы сможем ввести параметры вычисления для решения дифференциального уравнения. Содержание формы показывают предварительно установленные параметры.



В результате после заполнения форм мы получим в главном окне Scilab текст эквивалентной команды Scilab, например:
%ODEOPTIONS=[4,4,0,%inf,0,2,500,12,5,0,-1,-1]

Если это Вам кажется более удобным, то можно выполнять эту команду и в виде командной строки.
Замечание:
Глобальная переменная **%ODEOPTIONS** является опциональной (необязательной). Если Scilab обнаружит наличие этой переменной, то будет использовать ее без предупреждения, поэтому для использования значений по умолчанию ее следует уничтожить. Чтобы ее создать - следует выполнить команду **odeoptions**. Переменная **%ODEOPTIONS** является вектором следующего вида:
[itask,tcrit,h0,hmax,hmin,jactyp,mxstep,maxordn,maxords,ixpr,ml,mu]
Значение **%ODEOPTIONS** по умолчанию:
[1,0,0,%inf,0,2,500,12,5,0,-1,-1]
Расшифровку смысла значений элементов вектора **%ODEOPTIONS** можно узнать с помощью команды **help %ODEOPTIONS**.

Как решить уравнение Риккати (Riccati)?

Способ 1.
С помощью команды **riccati.**
Синтаксис **X=riccati(A,B,C,dom,[typ]) [X1,X2]=riccati(A,B,C,dom,[typ])**

Параметры
A, **B**, **C** : действительные матрицы размером
n на **n**, **B** и **C** симметричны.
dom : **'c'** или **'d'** для домена времени (непрерывного и дискретного)
typ : string : **'eigen'** для блока диагонализации или **'schur'** для метода Шура.
X1, **X2**, **X** : квадратные действительные матрицы (**X2** обратимая), **X** симметричная

Команда **X=riccati(A,B,C,dom,[typ])** является солвером для уравнения Риккати:
A'*X+X*A-X*B*X+C=0

в случае для непрерывного времени, или: **A'*X*A-**
(A'*X*B1/(B2+B1'*X*B1))*(B1'*X*A)+C-X
где **B=B1/B2*B1'** в случае дискретного времени. Если команды задается с помощью двух выходных аргументов, команда **riccati** возвращает **X1**, **X2**, которые представляют **X=X1/X2**.

Способ 2.
С помощью команды **ricc**. Подробно смотри **help ricc**.

Способ 3.
С помощью команды **ric_desc**.Это солвер уравнения Риккати с Гамильтонианово матрицей на входе. Смотри подробно с помощью команды **help ric_desc**.
