УДК 531;531.3;531.5;517.9; 519.6;519.8

# Математическое моделирование нелинейных обобщённо-механических систем в системе компьютерной математики MAPLE

**Ю. Г. Игнатьев, Х. Х. Абдулла**

*Математический факультет*
*Татарский государственный гуманитарно-педагогический университет*
*ул. Межлаук, д. 1, 420021, г. Казань, Татарстан*

Описаны алгоритмы и комплекс программ для математического моделирования в системах компьютерной математики нелинейных обобщённо-механических систем. Встроенные в пакет программные процедуры позволяют получать численные решения в форме сплайнов, $B$-сплайнов и кусочно-заданных функций. Описаны разработанные программные процедуры операций над сплайнами, позволяющие проводить аналитические вычисления с конвертированными численными решениями как с обычными функциями.

**Ключевые слова:** нелинейные динамические системы, математическое моделирование, комплексы программ, сплайны, системы компьютерной математики.

## 1. Введение

Уникальные графические возможности системы компьютерной математики (СКМ) Maple, в частности, возможности создания трёхмерных анимационных моделей, хорошо проработанные программные процедуры численного интегрирования систем обыкновенных дифференциальных уравнений (ОДУ), сплайновой и $B$-сплайновой интерполяции функций позволяют рассматривать СКМ Maple в качестве мощного современного инструмента математического моделирования [1,2]. В настоящее время методы математического моделирования в СКМ начали эффективно применяться в исследованиях математических моделей как фундаментальных физических явлений, так и прикладных задач. В частности, монография Д.П. Голоскокова [3] целиком посвящена проблемам математического моделирования в СКМ Maple объектов математической физики — физических полей, гидродинамических процессов, процессов теплопереноса и диффузии; в фундаментальной монографии В.П. Дьяконова [4] обширная глава посвящена применению СКМ Maple в математическом моделировании, в частности, моделированию электронных схем и измерительных систем на основе эффекта Допплера; монография М.Н. Кирсанова [2] содержит материалы по математическому моделированию сложных механических систем со связями и визуализации математических моделей этих систем. В [5–9] методы математического моделирования с помощью СКМ Maple успешно применяются для решения весьма сложных задач релятивистской кинетики, теории гравитации и космологии ранней вселенной. Важным преимуществом СКМ Maple является и возможность интеграции этой системы с СКМ MatLab, которая приспособлена к моделированию электронных систем и технологических процессов.

Объектом исследования этой статьи является математическое моделирование нелинейных обобщённо-механических систем, (НОМС), в среде компьютерной математики Maple. Такие системы в наиболее общем случае описываются системой нелинейных ОДУ, разрешённых относительно старших производных функций $y_i(t)$, вида:

$$y_i^{(n_i)} = F_i(y_1, \ldots, y_N, y_1', \ldots, y_N', y_1'', \ldots, y_N'', \ldots, y_1^{(n_i-1)}, t); \quad i = \overline{1, N}, \qquad (1)$$

где $y^{(n)} = \mathrm{d}^n f/\mathrm{d}t^n$ — обозначение $n$-той производной функции $f$ по независимой переменной $t$ времени, а $F_i$ — непрерывно-дифференцируемые функции своих

переменных. Будем в дальнейшем полагать выполненными начальные условия для системы (1):

$$y_i^{(k)}(t)\Big|_{t=t_0} = C_i^k; \quad k = \overline{1, n_i - 1}; \quad i = \overline{1, N}, \tag{2}$$

соответствующие стандартной задаче Коши, где $C_i^k$ — начальные значения производных $k$-го порядка функций $y_i(t)$.

Целью нашего исследования является разработка алгоритмов и пакетов (комплекса) программ в СКМ Maple для компьютерного исследования нелинейных обобщённо-механических систем и построения многопараметрических программных процедур визуализации математических моделей этих систем, в том числе и процедур динамической визуализации (анимации) математических моделей НОМС. Отметим, что важной идеей построения пакета программ явилась идея использования сплайновой и B-сплайновой интерполяции функций, позволившая создать автоматизированный вывод численных решений ОДУ в форме кусочно-заданных функций.

## 2. Блок-схема комплекса программ

Для обеспечения гибкой работы с численными решениями были созданы специальные внутренние программные процедуры, позволяющие проводить стандартные операции анализа функций одной переменной со сплайнами, а также конвертировать их в кусочно-заданные функции. В результате был получен программный аппарат аналитического (приближенного) исследования НОМС в СКМ Maple. Важным достоинством разработанного комплекса программ является его независимость от версии Maple, начиная с версии 1997 года Maple5.5 и кончая версией 2009 года Maple14. Представленный комплекс программ состоит из двух независимых пакетов программ.

Пакет программ `DifEq` содержит программы распознавания введённых системы дифференциальных уравнений и начальных условий `a`, программы автоматического преобразования введённых данных в задачу Коши для нормальной системы относительно унифицированных переменных, `b`, программы численного решения задачи Коши на заданном интервале с возможностью контроля и переключения метода численного интегрирования при заданном значении независимой переменной, `c`. Пакет программ `Splines` содержит программы генерации равномерных кубических сплайнов и $B$-сплайнов по заданной функции на заданном интервале, `d,f`, программы конвертирования сплайнов и $B$-сплайнов в кусочно-непрерывные функции и обратные операции, `e,g`, программы операций над сплайнами `h`. Операции указанных пакетов интегрируются в программе конвертирования численных решений в кусочно-заданные функции, `i`.Ниже мы опишем основные алгоритмы разработанного комплекса программ и продемонстрируем на примерах исполнение программных процедур.

## 3. Пакет программ преобразования системы уравнений и решения задачи Коши (DifEq)

Рассмотрим ОДУ $n$-го порядка, разрешённого относительно старшей производной функции $y(x)$:

$$y^{(n)} = F\left(x, y, y', \ldots, y^{(n-1)}\right). \tag{3}$$

Для задания $n$-той производной функции $y(x)$ по переменной $x$ в СКМ Maple существуют две альтернативные процедуры: `diff(y(x),x$n)` и `(D@@n)(y)(x)`. Вторая из этих процедур предусматривает возможность вычисления производной в

заданной точке, что весьма удобно для задания начальных условий. Создадим программную процедуру `DifEq[DiffOp]`, позволяющую извлечь полную информацию о старшей производной в правой части уравнения (3)[1]:

```
> DifEq[DiffOp]:=proc(X) local var,s,i,ss,ff,tt:
  if op(0,X)=diff then: var:=op(X)[2]; s:=op(X)[1]; for i from 1 do
  if(nops(s)=1) then ff:=(op(0,s)); tt:=(op(1,s)); break: else
  s:=op(s)[1]; end if; end do: [diff,i,ff,tt]: elif
  op(1,op(0,op(0,X)))=D then [D,op(2,op(0,op(0,X))),
  op(-1,op(0,X)),op(-1,X)]: elif op(0,op(0,X))=D then
  [D,1,op(-1,op(1,op(0,X))),op(-1,X)]: else
  ff:= op(0,X): tt:=op(-1,X): [0,ff,tt]: end if:end proc:
```

Продемонстрируем исполнение этой процедуры на двух примерах:

```
> DifEq[DiffOp]((D@@4)(F)(tau));
```

$$[D,4,F,\tau]$$

```
> DifEq[DiffOp](diff(Sigma(zeta),zeta));
```

$$[diff,1,\Sigma,\zeta]$$

Таким образом, при действии на производную данная программная процедура создаёт упорядоченный список — тип оператора производной, порядок производной, имя неизвестной функции и имя независимой переменной. Это позволяет перейти в дальнейшем от обозначений пользователя к некоторым унифицированным внутренним именам зависимых и независимых переменных, что позволяет автоматизировать процессы работы с уравнениями.

Блок (b) включает три программные процедуры, осуществляющие поэтапное конвертирование системы ОДУ (1) с начальными условиями (2) к задаче Коши для нормальной системы ОДУ с унифицированными именами переменных. Знание информации о старшей производной уравнения (3) позволяет конвертировать это уравнение к нормальной системе ОДУ с унифицированными обозначениями, что обеспечивает автоматизацию манипуляций с этими уравнениями. Договоримся о следующем порядке присвоения унифицированных имён зависимых переменных. Для этого рассмотрим упорядоченный список `DifEq[MatAlf]` из восьми имён: `[X,Y,Z,U,V,W,Phi,Theta]`, который, конечно, можно продолжить, но для этого нет никаких причин практического характера. Обозначим переменную, введённую пользователем, посредством X, её первую производную — Y, вторую производную — Z и т.д., а независимую переменную посредством t. Полученная система и будет являться искомой нормальной системой ОДУ с унифицированными именами переменных:

$$\frac{\mathrm{d}X_i^1}{\mathrm{d}t} = X_i^2; \quad \frac{\mathrm{d}X_i^2}{\mathrm{d}t} = X_i^3; \ldots; \quad \frac{\mathrm{d}X_i^{n_i-1}}{\mathrm{d}t} = F_i, \quad i = \overline{1,N}, \tag{4}$$

где $F = F(t, X_1^1, \ldots, X_N^{n_N})$ и для краткости положено $X^k = \mathrm{DifEq}[\mathrm{MatAlf}]_k \to X^1 = X, X^2 = Y, \ldots$. и системой начальных условий для этих уравнений, соответствующих начальным условиям (2) :

$$X_i^k(t_0) = C_i^k; \quad k = \overline{1, n_i - 1}; \quad i = \overline{1,N}. \tag{5}$$

Первый из указанных алгоритмов — алгоритм приведения ОДУ произвольного порядка к нормальной системе ОДУ ОДУ произвольного порядка к нормальной системе ОДУ реализуется программной процедурой `DifEq[Oden_ConvNorm]`, которая, в свою очередь, использует две промежуточные процедуры, `DifEq[DiffOp]` и `DifEq[MatAlf]`. Рассмотрим исполнение этой процедуры на примере уравнения в обозначениях пользователя:

$$S^{IV}(\xi) = \left(S^{III}(\xi)\right)^2 S^3(\xi) + S'(\xi) + \sin\xi. \tag{6}$$

[1]Идею этого алгоритма подсказал А.В. Матросов, которому Ю.Г. Игнатьев выражает признательность.

Пользователь может ввести это уравнение в такой, например, форме:

```
>Eq1:=(D@@4)(S)(xi)=((D@@3)(S)(xi))\^{}2*S(xi)\^{}3+D(S)(xi)+sin(xi);
```

Результат применения нашей процедуры к уравнению (6) получается следующим:

```
>DifEq[Oden_ConvNorm](Eq1,2);
```

$$\Big[\Big[4, S, \xi, D\Big], \quad \Big[X_2, Y_2, Z_2, U_2\Big],$$
$$\Big[S(\xi) = X_2(t), \quad D(S)(\xi) = Y_2(t), \quad (D^{(2)})(S)(\xi) = U_2(t), \quad \xi = t\Big],$$
$$\Big[\frac{\mathrm{d}}{\mathrm{d}t}X_2(t) = Y_2(t),\ \frac{\mathrm{d}}{\mathrm{d}t}Y_2(t) = U_2(t),\ \frac{\mathrm{d}}{\mathrm{d}t}U_2(t) = U_2(t)^2 X_2(t)^3 + Y_2(t) + \sin(t)\Big]\Big].$$

Он состоит из упорядоченного списка четырёх упорядоченных списков: первый список содержит: [порядок уравнения, искомую функцию, независимую переменную, тип дифференциального оператора], введённые пользователем. Второй список содержит упорядоченную систему новых функций. Третий список содержит правила замены переменных в ОДУ, четвёртый список содержит упорядоченную нормальную систему ОДУ. Как видно, процедура `DifEq[Oden_ConvNorm]` содержит два обязательных параметра, первый — дифференциальное уравнение, а второй, $k$, — любое имя или число, необходимое для пометки уравнения и входящих в него переменных. Соответствующая метка помещается как нижний индекс у соответствующих величин. В рассмотренном примере $k = 2$. В частности, если мы хотим отказаться от индексирования функций, достаточно вместо индекса ввести ‘‘ — два апострофа.

Аналогично вводится и однопараметрическая процедура приведения системы ОДУ (1) к нормальной системе ОДУ `DifEq[SysOden_ConvNorm]`. Результат применения этой процедуры, который для краткости мы не приводим, представляется в виде трёх упорядоченных списков: в первом содержится один элемент $S$ — число уравнений нормальной системы (число степеней свободы НОМС); во втором — преобразования пользовательских функций к унифицированным функциям нормальной системы; третий список — есть упорядоченная нормальная система обыкновенных дифференциальных уравнений, причём первые $S-N$ уравнений представляют собой результат стандартной замены переменных вида $y' = z$.

На основе программ распознавания системы ОДУ (блок «а») и программ приведения ОДУ к нормальному виду строится двухпараметрическая программная процедура

```
DifEq[SysCauchy_ConvNorm](SystemODE,Inits Conditions)
```

приведения системы ОДУ произвольного порядка (1) с начальными условиями (2) к задаче Коши для нормальной системы ОДУ относительно унифицированных переменных $X_i^k(t)$ (5) с начальными условиями (5), т.е. к задаче Коши для нормальной системы ОДУ. Результат применения процедуры выводит упорядоченный список из шести упорядоченных списков: в первом содержится два числа: $S$ — число уравнений нормальной системы и $M = \max(n_1, .., n_N)$ — максимальный порядок уравнений в исходной системе уравнений, во втором — упорядоченные списки новых переменных, — число этих списков равно $M$, выбранных по следующему принципу — в первом списке содержатся новые переменные, $X[i]$, полученные из независимых функций пользователя, во втором — первые производные от этих переменных, $Y[i]$, если вторые производные от этих переменных содержатся в системе ОДУ, и т.д., — до $M$ − 1-го списка. Таким образом, количество внутренних списков независимых функций совпадает с максимальным порядком уравнений исходной системы, $M$. Третий список содержит преобразования пользовательских функций к унифицированным функциям нормальной системы, четвёртый список — есть упорядоченная нормальная система обыкновенных дифференциальных уравнений, причём первые $S-N$ уравнений представляют собой результат стандартной замены переменных вида $y' = z$, в пятом — начальные условия для нормальной системы ОДУ в форме (5), в шестом — один элемент — начальное значение независимой переменной. Созданная процедура

удобна для извлечения различной информации об исходной системе дифференциальных уравнений и начальных условий.

Блок «c» содержит две программные процедуры численного решения системы ОДУ (1) с начальными условиями (2), — трехпараметрическую программную процедуру `DifEq[NumDsolve](System ODE, Inits Conditions, Method)` и пятипараметрическую процедуру `DifEq[ReNumDsolve](SystemODE,InitsConditions, Method1,x1,Method2)`. Первая команда создаёт процедуру решения системы ОДУ с помощью метода Method, встроенного в пакет Maple; значение этого параметра 45 соответствует методу Рунге–Кутта 4–5 порядков, 78 — методу Рунге–Кутта 7–8 порядков, *rosenbrock* — методу Розенброка, *stiff* — методу stiff интегрирования жёстких уравнений, *classic* — классическому методу (по умолчанию методом Эйлера), *taylor* — методом разложения в ряды Тейлора. При этом Вывод решений осуществляется в виде упорядоченного списка в вложенными в него $M$ упорядоченными списками. При этом первый упорядоченный список содержит численные значения $N$ искомых функций: `[X[1](t),..X[N](t)]`, порядок записи значений функций в списке совпадает с порядком записи дифференциальных уравнений системы. Второй упорядоченный список содержит значения первых производных тех функций, производные которых не ниже второго порядка содержатся в системе ОДУ. Порядок записи значений первых производных функций в списке совпадает с порядком записи дифференциальных уравнений системы — при этом пропускаются значения производных тех функций, производные выше первого порядка которых не содержатся в системе ОДУ. Третий упорядоченный список содержит значения первых производных тех функций, производные которых не ниже третьего порядка содержатся в системе ОДУ. Порядок записи значений вторых производных функций в списке совпадает с порядком записи дифференциальных уравнений системы — при этом пропускаются значения вторых производных тех функций, производные выше второго порядка которых не содержатся в системе ОДУ, и т.д.

Для демонстрации формата ввода системы и вывода решений рассмотрим *пример интегрирования существенно нелинейной системы ОДУ* с максимальным 3-м порядком производных:

$$\frac{\mathrm{d}^2F}{\mathrm{d}x^2} = \Phi\frac{\mathrm{d}^2Z}{\mathrm{d}x^2}; \quad \frac{\mathrm{d}^3Z}{\mathrm{d}x^3} = -F^2; \quad \frac{\mathrm{d}\Phi}{\mathrm{d}x} = F^3 = \sin x \tag{7}$$

с начальными условиями:

$$F(\pi) = 1; \quad F'(\pi) = 0; \quad Z(\pi) = 1; \quad Z'(\pi) = -1; \quad Z''(\pi) = -1/3; \quad \Phi(\pi) = 0. \tag{8}$$

Систему (7) пользователь может ввести, например, таким способом[2]:

```
> ODE1:=[(D@@2)(F)(x)=Phi(x)*(D@@2)(Z)(x),(D@@3)(Z)(x)=-F(x)^2,
  D(Phi)(x)=F(x)^3+sin(x)];
```

При этом начальные условия (8) для этой системы:

```
>Inits1:=[F(Pi)=1,D(F)(Pi)=0,Z(Pi)=1,D(Z)(Pi)=-1,(D@@2)(Z)(Pi)=1/3,
  Phi(Pi)=0];
```

Процедура решения системы (7) с начальными условиями (8) методом Розенброка осуществляется командой:

```
>SS:=DifEq[NumDsolve](Eqs,Inits1,rosenbrock);
```

При этом решению произвольно присвоено имя `SS`. Тогда численное решение выводится функцией `SS(x)`. В данном случае список решений состоит из трёх внутренних списков ($M = 3$), в первом из них содержится 3 числа ($N = 3$) — это значения функций `[F(1),Z(1),Φ(1)]`, во втором списке содержится два числа, поскольку в системе (7) содержатся лишь две произ-водные не ниже 2-го порядка, — это значения первых производных `[F'(1),Z'(1)]`, наконец, третий список содержит лишь одно число — это значение второй производной функции

[2]Уравнения вводятся упорядоченным списком.

`Z''(1)`. Для вывода одного из этих списков, $i$-го, достаточно применить процедуру `SS(1)[i]`. График траектории системы в конфигурационном пространстве $E_3 : \{X_1 = F, X_2 = Z, X_3 = \Phi\}$ можно получить простой командой Maple, как и для трёхмерного графика обычной функции:

```
>plots[spacecurve](SS[1](t),t=0..10,color=black,axes=BOXED,
  labels=[‘X[1]‘,‘X[2]‘,‘X[3]‘]);
```

Программная процедура

```
DifEq[ReNumDsolve](Eqs,Inits,Method1,x1,Method2)
```

построена на основе рассмотренный выше процедуры `DifEq[NumDsolve]` и встроенной в Maple программной процедуры `piecewise(x>=x0 and x<x1,f1,x>=x1,f2)` кусочно-заданной функции, так что на интервале `[x0,x1]` при численном интегрировании системы ОДУ применяется метод `Method1`, а на интервале `(x1,..)` — метод `Method2`. При этом начальными условиями для численного интегрирования методом `Method2` являются результаты численного интегрирования системы ОДУ в точке `x1`, полученные методом `Method1`. Указанный метод можно назвать *методом интегрирования с перезагрузкой начальных условий*. Данный метод следует применять в тех случаях, когда на некотором отрезке стандартные методы интегрирования не дают хороших результатов. Следует отметить, что программные процедуры `DifEq[NumDsolve]` и `DifEq[ReNumDsolve]` не требуют выполнения пользователем предварительных операций по приведению системы ОДУ к нормальному виду, так как эти операции являются встроенными процедурами в указанные, — нормирование системы ОДУ и извлечение всей необходимой информации производится автоматически.

Для тестирования пакета программ `DifEq` на точность вычислений, а также выяснения немаловажного для задач компьютерного моделирования вопроса о скорости вычислений различными численными методами можно провести численное интегрирование различных точно решаемых нелинейных систем ОДУ различными методами с помощью программных процедур `DifEq[NumDsolve]` и `DifEq[ReNumDsolve]`. При этом реальное время, необходимое для интегрирования системы и представления численных результатов в графическом виде, можно вычислить с помощью встроенной в Maple функции `time()`, которая определяет точное время в секундах по встроенному в компьютер таймеру. Точность и скорость вычислений тестировались по целому ряду точно решаемых нелинейных систем ОДУ, примеры которых можно найти во многих учебниках по дифференциальным уравнениям. Рассмотрим следующий пример задачи Коши для нелинейной системы ОДУ:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{y}{x}; \quad \frac{\mathrm{d}z}{\mathrm{d}x} = -\frac{y}{z} \tag{9}$$

с начальными условиями:

$$y\left(\frac{1}{2}\right) = \frac{1}{2}; \quad z\left(\frac{1}{2}\right) = \frac{\sqrt{3}}{2}. \tag{10}$$

Указанная задача Коши точно решается:

$$y(x) = x; \quad z(x) = \sqrt{1 - x^2}; \quad z \geqslant 0, \tag{11}$$

так что $x^2 + y^2 = 1$, т.е. конфигурационной траекторией системы является окружность единичного радиуса. Хотя найденное точное решение задачи Коши имеет весьма простой вид, оонако с точки зрения применения численных методов интегрирования система (9) является достаточно неудобной: она не является автономной и имеет особые точки при $x = 0$, $z = 0$. Поэтому система ОДУ (9) и подобные ей удобны для тестирования численных методов решения ОДУ. Рассмотрим пример численного решения задачи Коши (9)–(10) и манипуляции с этими решениями. Сразу оговоримся, что для созданного комплекса программ исследование системы (9)–(10) является элементарной задачей, но мы продемонстрируем на этой задаче все особенности применения пакета программ, чтобы не

загромождать изложение громоздкими выражениями. Далее, хотя система (10) уже является нормальной, тем интереснее будет проверить для этого случая исполнение процедур преобразования её к задаче Коши для нормальной системы с унифицированными именами переменных:

```
> DifEq[SysCauchy_ConvNorm]([diff(y(x),x)=y(x)/x,
  diff(z(x),x)=-y(x)/z(x)],[y(1/2)=1/2,z(1/2)=sqrt(3)/2]);
```

$$[[2,1],\quad [X_1(t),X_2(t)],\quad [\,],\quad [y(x)=X_1(t),\quad z(x)=X_2(t)],$$
$$\left[\frac{\mathrm{d}}{\mathrm{d}t}X_1(t)=\frac{X_1(t)}{t},\ \frac{\mathrm{d}}{\mathrm{d}t}X_2(t)=-\frac{X_1(t)}{X_2(t)}\right],\quad \left[X_1\left(\frac{1}{2}\right)=\frac{1}{2},\ X_2\left(\frac{1}{2}\right)=\frac{\sqrt{3}}{2}\right],\ \frac{1}{2}\right].$$

Следует заметить, что исполнение данной команды не является обязательным для решения системы уравнений, но оно хорошо демонстрирует структуру внутренних программных процедур. Само численное решение, например, методом Розенброка сразу достигается командой:

```
> SS1:=DifEq[NumDsolve]([diff(y(x),x)=y(x)/x,diff(z(x),x)=-y(x)/z(x)],
  [y(1/2)=1/2,z(1/2)=sqrt(3)/2],rosenbrock);
```

и выводится в функциональном списочном виде, соответствующем векторной функции скалярного аргумента. Продемонстрируем построение *конфигурационной траектории* системы (9)–(10) $\mathbf{r} = [y(x), z(x)] = [X_1(t), X_2(t)]$ при интегрировании методом Розенброка. Построение траектории достигается простой стандартной командой Maple, в которой мы указали две необязательные опции: scaling = CONSTRAINED – для сохранения масштаба, и color=black, для определения цвета кривой:

```
> plot([SS4(t)[1,1],SS4(t)[1,2],t=-1..1],scaling=CONSTRAINED,
  color=black);
```

Заметим, что визуально изменение фазовых траектории в зависимости от применяемых из перечисленных выше методов численного интегрирования обнаружить не удаётся. Для выяснения вопроса о точности методов интегрирования будем вычислять логарифм модуля разности численного, `Z(t)`, и точного, `Z0(t)`, решений (11):

$$L(t) = \lg|Z(t) - Z0(t)|. \tag{12}$$

Вычисления проводились на компьютере AMD Athlon 64×2 Dual Core Processor 4200+ 2.2 ГГц 2,00 ГБ ОЗУ. Так, например, время, затрачиваемое на вычисления с помощью метода Рунге–Кутта составляет для рассмотренного примера порядка 0,062 сек., в то время как эта же операция, выполненная с помощью метода Тейлора, занимает 66,2 сек., т.е. в 1000 раз больше. Классический метод дает слишком плохие результаты для нелинейных систем ОДУ. Оптимальными методами вычислений (достаточно хорошая точность при высокой скорости вычислений) можно признать методы Розенброка и stiff-метод, а методом, дающим наибольшую точность при средней скорости вычислений можно признать метод Рунге–Кутта 7–8 порядков.

## 4. Программные процедуры сплайновой интерполяции функций

Опишем теперь алгоритмы и программные процедуры пакета программ Splines, позволяющие автоматически получать решение задачи Коши для нелинейной системы ОДУ произвольного порядка в форме равномерных кубических сплайнов [10, 11] и $B$-сплайнов. Для создания аппарата аналитического исследования решений нелинейной системы ОДУ и проведения компьютерного моделирования нелинейных обобщённо-механических систем были созданы специализированные программные процедуры, позволяющие выполнять над сплайнами алгебраические и интегро-дифференциальные операции, а также операции взаимообратного конвертирования сплайнов и $B$-сплайнов в кусочно- и кусочно-параметрически

заданные функции. Таким образом удалось добиться важного результата — создать комплекс программ, позволяющий проводить аналитическое компьютерное исследование нелинейных обобщённо-механических систем.

В пакете `CurveFitting` системы Maple имеются встроенные процедуры построения сплайновой интерполяции `Spline`, которые могут иметь формат

`Spline(xydata,v,dgr,endpts) или Spline(xdata,ydata,v,dgr,endpts),`

где `xydata` — двумерный массив вида $[\ldots,[x_i,y_i],\ldots]$; `xdata`, `ydata` — одномерные массивы вида $[\ldots,x_i,\ldots],[\ldots,y_i,\ldots]$; `v` — имя независимой переменной, `dgr` — необязательный параметр, задание которого осуществляется в форме `degree=p`, где $p \in \mathrm{Z}$ — наивысшая степень интерполяционных полиномов. По умолчанию $p = 3$, и получается кубический сплайн. Наконец, `endpts` — необязательный параметр, управляющий типом сплайна, например `endpoints='natural'`, дает натуральный сплайн. Применение процедуры `Spline` к двумерному или двум одномерным массивам генерирует сплайн, который в СКМ Maple представляется, фактически, кусочно-заданной функций с вертикальным форматом вывода её кусков. Несмотря на то, что программные процедуры сплайновой интерполяции функций системы Maple весьма достоверно и надёжно осуществляют построение сплайновой интерполяции числовых баз данных указанных массивов, программный аппарат действий со сплайнами в СКМ отсутствует, что не позволяет непосредственно применять результаты сплайновой интерполяции численных решений для аналитических исследований моделей. Поэтому для создания эффективного аппарата компьютерного моделирования нелинейных обобщённо-механических систем в СКМ необходимо разработать программное обеспечение операций над сплайнами.

Используя процедуру `Spline`, создадим необходимую в дальнейшем промежуточную 6-параметрическую обратную к `Spline` простую процедуру генерации равномерных $n$-кусочных кубических сплайнов относительно функции $f(x)$, заданной на отрезке $[a,b]$, `Splines[SplineF](f,x,a,b,n,z)`, с передачей имени `z` её независимому аргументу:

```
> Splines[SplineF]:=proc(f,x,a,b,n,z) local F,X,Basa:
  F:=(X)->subs(x=X,f):
  Basa:=[seq([a+i*(b-a)/n,eval(F(a+i*(b-a)/n))],i=0..n)]:
  CurveFitting[Spline](Basa,z): end proc:
```

Введём предварительно 3-параметрическую процедуру `Splines[Conv_List](sp,t,z)` конвертирования сплайна `sp` функции $f$ на заданном отрезке в упорядоченный список, состоящий из чётного числа элементов, в котором каждая пара представляет собой упорядоченный набор элементов, первый из которых — неравенство, устанавливающее верхнюю границу интервала, а второй — сплайн на данном интервале:

```
> Splines[Conv_List]:=proc(sp,t,z)
  local LSS0,LSS,T,ddd,NN0,NN1,t1,DDD:
  LSS0:=convert(sp,list):LSS:=(T)->
  subs(t=T,LSS0):NN0:=nops(LSS0):NN1:=(NN0+1)/2:
  ddd:=rhs(op(3,LSS(t)))-rhs(op(1,LSS(t))):
  t1:=rhs(op(1,LSS(t)))-ddd:DDD:=t1+NN1*ddd:
  [seq(op(i,LSS(z)),i=1..NN0-1),(z<DDD,op(-1,LSS(z)))]: end proc:
```

Заметим, что процедура автоматически распознает параметры сплайна. Создадим также и обратную процедуру конвертирования списка в кусочно-заданную функцию, piecewise:

```
> Splines[Conv_Piece]:=proc(Ls,t,z) local T,LS0,LSS0:
  LS0:=(T)->subs(t=T,Ls):
  LSS0:=(t)->subs(op(-2,LS0(t))=NULL,LS0(t)):
  piecewise(op(LSS0(z))): end proc:
```

Здесь применена встроенная процедура Maple `NULL` — команда пустого множества $\varnothing$.

Операция вычисления производной n-го порядка в точке `t0` от сплайна можно ввести простой программной процедурой с помощью встроенной в Maple программной процедурой `convert(DSP(T),piecewise,T)`, конвертирующей сплайн в

кусочно-заданную функцию. Вычислим, например, с помощью построенной процедуры первую производную сплайна PPS6 в точке $\pi/4$:

```
> Splines[SplineDiff](PPS6(t),t,Pi/4,1);
```

$$\frac{207\sqrt{3}}{160\pi} \approx 0,7133.$$

Прямое вычисление значения $\sin(\pi/4) =$ дает $\sqrt{2}/2 \approx 0,7071$, т.е. отличается от предыдущего в третьем знаке. Однако уже при увеличении числа интервалов в два раза (сплайн PPS12) точность интерполяции возрастает на порядок: $\sin(\pi/4) \approx 0,7072$. Для использования производной от заданного сплайна как функции необходимо теперь просто определить эту функцию:

```
> F:=(x)->Splines[SplineDiff](PPS(t),t,x,1):
```

Процедуры нахождения функции $f(S,t)$ от сплайна $S$можно создать двумя путями. Первый, самый простой, основан на первоначальном конвертировании исходной аппроксимации в список и построении на основе его новой базы для сплайна. Второй, более сложный, состоит в начальном конвертировании исходного сплайна в список, преобразование функций списка и конвертирование этого списка в кусочно-заданную функцию. Полученная вторым способом функция уже не является, строго говоря, сплайном $n$-го порядка, так как на каждом интервале представляется уже не многочленом $n$-го порядка, а функцией от этого многочлена. Однако при этом сохраняется принадлежность кусочно-заданной функции$f(S)$ классу $C^n$, если сама функция $f(x)$ принадлежит этому классу. Первая из указанных процедур 4-параметрическая процедура `Splines[SplineFunctionPoint]`. Вторая из указанных процедур с такими же параметрами имеет вид `Splines[SplineFunction](sp,t,f,z)`. Применение созданных процедур к различным функциям показывает практически полное совпадение результатов интерполяции функции сплайнов с помощью процедуры `Splines[SplineFunction]` с функцией-оригиналом даже при небольшом числе интервалов и значительное расхождение результатов при применении процедуры `Splines[SplineFunctionPoint]`[3].

В дальнейшем понадобятся функции от двух сплайнов `S1` и `S2`, — `f(S1,S2,t)`. В частности, такие функции будут полезными при вычислении определённых интегралов, когда по формуле Ньютона необходимо вычислить разность двух сплайнов. Создадим процедуру вычисления таких функций. При этом важно, чтобы сплайны совпадали как внешними, так и внутренними интервалами. Будем предполагать сплайны равномерными. Для ввода процедуры вычисления функции от пары сплайнов `Splines[BiSplineFunction]`, используем рассмотренный выше второй метод построения программной процедуры вычисления функции от сплайна. Созданная программная процедура `Splines[BiSplineFunction](S1,S2,t)` автоматически распознает вводимые пользователем сплайны `S1, S2` и при их несоответствии другу выдаёт пользователю сообщение об этом и прекращает выполнение операций. Создадим также процедуру вычисления определённого интеграла от функции $f$, заданной в сплайновой интерполяции, вида $\int_a^b f(S,t)\mathrm{d}t$, где `S(t)` — равномерный $N$-кусочный сплайн, определённый на некотором отрезке `[t1,tn]`, и `[a,b]` — произвольный вещественный отрезок. Созданная процедура автоматически распознает параметры сплайна и в случаях, когда границы отрезка `[a,b]` выходят за границы определения сплайна, выдаёт пользователю сообщение и прекращает выполнение операции. Созданная процедура использует промежуточную процедуру приближенного интегрирования методом прямоугольников, `Splines[Integral]`.

[3]Заметим, что с увеличением числа интервалов процедура `Splines[SplineFunctionPoint]` также начинает давать неплохие результаты.

$B$-сплайном (базисным сплайном) называют сплайн-функцию, имеющую наименьший носитель для заданной степени, порядка гладкости и разбиения области определения. Фундаментальная теорема устанавливает, что любая сплайн-функция для заданной степени, гладкости и области определения может быть представлена как линейная комбинация $B$-сплайнов той же степени и гладкости на той же области определения. В СКМ Maple функция вычисления $B$-сплайнов, `BSplineCurve`, содержится, как и функция `Spline`, в пакете `CurveFitting`. Программная процедура `BSplineCurve` имеет формат ввода, аналогичный формату ввода процедуры `Spline`:

`BSplineCurve(xydata,v,opts) или BSplineCurve(xdata, ydata,v,opts)`,

где теперь, необязательные параметры имеют формат `order=k` или `knots=knotlist`, а `knowlist` — список узлов. Однако, формат вывода процедуры `BSplineCurve` существенно отличается от формата вывода процедуры `Spline` — вместо кусочно-заданной на отрезке `[a,b]` функции $S(x)$ `BSplineCurv` выводит два списка кусочно-заданных функций $S(t)$ и $x(t)$ в вертикальном формате, где $t$ — новый параметр, который принимает значения на некотором интервале $[k, n]$, $n > k$, где $k, n \in \mathbb{Z}$, при этом отрезок $[k, n]$ разбивается на целочисленные отрезки: $[i, i + 1]$, $i = \overline{k, n}$.

Фактически формат вывода $B$-сплайна эквивалентен выводу кусочно-параметрически заданной функции:

$$x = x(t); \quad S = S(t) \Rightarrow \mathbf{r} = [x(t), S(t)], \tag{13}$$

хотя и не совпадает с ней, что создаёт большие неудобства для пользователя.

Для устранения указанного недостатка СКМ Maple введём 6-параметрическую процедуру `Splines[BSplineF(f,x,a,b,n,z)` генерации $n$-кусочного $B$-сплайна на основе заданной на интервале $[a, b]$ функции $f(x)$ и последующего конвертирования полученного $B$-сплайна в кусочно-параметрически заданную функцию вида (13). Также создадим 3-параметрическую процедуру `Splines[ConvBSpline_Piece] (BS,x,z)` конвертирования $B$-сплайна `BS(x)` в кусочно-параметрически заданную функцию с именем $z$ параметра. Созданные процедуры конвертирования устанавливают связь между $B$-сплайновым и сплайновым представлениями интерполяции и позволяют использовать мощный аппарат $B$-сплайнов *аналитизации* процесса численного интегрирования нелинейных систем ОДУ.

Интегрируем, наконец, программные процедуры описанного здесь пакета программ Splines с программными процедурами пакета `DifEq` в блоке `(i)` конвертирования численных решений нелинейной системы ОДУ в кусочно-заданные функции. Этот блок содержит четыре программные процедуры: `DifEq[ODESpline], DifEq[ODEBSpline], DifEq[ReODESpline], DifEq[ReODEBSpline]`. Процедура автоматизированной сплайновой обработки численного решения задачи Коши для нелинейной системы произвольного числа ОДУ произвольного порядка `DifEq[ODESpline](Eqs,Inits,Method,x1,n,i,k,x)` содержит 8 параметров: `Eqs` — упорядоченный список системы ОДУ, `Inits` — упорядоченный список начальных условий, `Method` — метод интегрирования, `x1` — конечная точка интервала интегрирования, `n` — число интервалов сплайна, `i` — порядковый номер выводимой функции[4], $k$ — порядок производной этой функции, $x$ — присваиваемое имя независимой переменной. Аналогично строится и 8-параметрическая процедура `DifEq[ODEBSpline]`, но в в ней использована процедура `Splines[ConvBSpline_Piece]`, позволяющая автоматически осуществлять вывод решений в виде кусочно-параметрически заданных функций (3). Аналогично строятся и 10-параметрические программные процедуры `DifEq[ReODESpline] и DifEq [ReODEBSpline]` с перезагрузкой метода интегрирования в точке `x2`. Заметим, что все рассмотренные в этом разделе процедуры являются конечными и не требуют от пользователя применения каких-либо других процедур для получения решений в системы

[4]Совпадает с порядковым номером уравнения в системе ОДУ.

нелинейных ОДУ в форме сплайнов или $B$-сплайнов. Кроме того, решения выводятся в формате аналитической функции или упорядоченной пары функций в случае $B$-сплайна.

## 5. Пример компьютерного исследования системы нелинейных ОДУ

Для демонстрации комплекса программ рассмотрим простой пример получения решения системы ОДУ в виде сплайнов. В качестве системы ОДУ рассмотрим нелинейную систему оду 1-го порядка (9)–(10). Отметим, что выбор простой системы, а также малое количество интервалов сплайна, которое мы собираемся ввести, продиктованы лишь необходимостью краткости изложения. Тестирование показывает весьма хорошие результаты и при большом числе уравнений системы (десятки), высокого порядка уравнений системы (до 6-го) и большом количестве интервалов сплайнов (десятки). Однако вывод таких результатов занимает весьма большое место, не совместимое с форматом статьи. Вся операция численного решения системы с его сплайновым представлением производится простой командой:

```
> Sp6:=(xi)->Splines[ODESpline]([diff(y(x),x)=y(x)/x,
  diff(z(x),x)=-y(x)/z(x)],
  [y(-1/2)=-1/2,z(-1/2)=sqrt(3/2)],45,1,6,1,2,xi):
```

Вычислим, например, вторую производную от полученного решения в точке $\xi = 1/8$:

```
> Splines[SplineDiff](Sp6(t),t,1/8,2);
```

-1.210034572

Вычислим, например, определённый интеграл от полученного решения $\int_0^1 \sin z(\xi)\mathrm{d}\xi$:

```
> Splines[SplineDefInt](Sp6(xi),xi,sin(xi),0,1);
```

0.4596988779;

Графическое представление решения на интервале $[-1/2, 1]$ чёрным цветом в нерастяжимом формате производится стандартной командой Maple (рис. 1, 2):

```
> plot(Sp6(t),t=-1/2..1,color=black,scaling=CONSTRAINED);
```

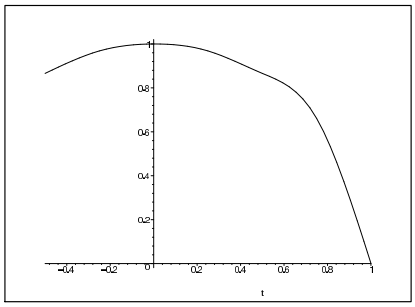

Рис. 1. **Решение уравнений (9) с начальными условиями (10) в формате 6-кусочного сплайна**

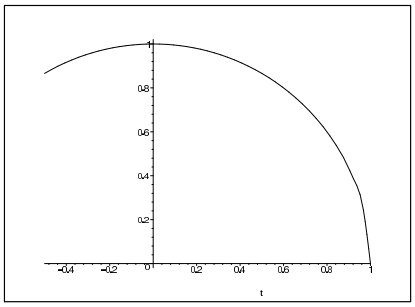

Рис. 2. **Решение уравнений (9) с начальными условиями (10) в формате 40-кусочного сплайна**

В работе авторов [12] приведены примеры исследования существенно нелинейной релятивистской электродинамической системы с помощью разработанного программного комплекса.

## Литература


1. *Матросов А. В.* Maple 6. Решение задач высшей математики и механики. — СПб.: БХВ-Петербург, 2001. — 528 с. [*Matrosov A. V.* Maple 6. Reshenie zadach vihssheyj matematiki i mekhaniki. — SPb.: BKhV-Peterburg, 2001. — 528 s.]
2. *Кирсанов М. Н.* Maple 13 и Maplet. Решение задач механики. — М.: Физматлит, 2010. — 349 с. [*Kirsanov M. N.* Maple 13 i Maplet. Reshenie zadach mekhaniki. — M.: Fizmatlit, 2010. — 349 s.]
3. *Голоскоков Д. П.* Уравнения математической физики. Решение задач в системе Maple. — СПб.: Питер, 2004. — 539 с. [*Goloskokov D. P.* Uravneniya matematicheskoyj fiziki. Reshenie zadach v sisteme Maple. — SPb.: Piter, 2004. — 539 s.]
4. *Дьяконов В. П.* Maple 9.5/10 в математике, физике и образовании. — М.: Солон-Пресс, 2006. — 720 с. [*Djyakonov V. P.* Maple 9.5/10 v matematike, fizike i obrazovanii. — M.: Solon-Press, 2006. — 720 s.]
5. *Ignatyev Y. G., Alsmadi K.* A Complect Relativistic Kinetic Model of Symmetry Violation in an Anisotropic Expanding Plasma. II. X-boson Distribution Function // Gravitation and Cosmology. — 2005. — No 11. — Pp. 363–372.
6. *Ignatyev Y. G., Alsmadi K.* A Complect Relativistic Kinetic Model of Symmetry Violation in an Anisotropic Expanding Plasma. III. Specific Entropy Calculation // Gravitation and Cosmology. — 2007. — No 13. — Pp. 114–120.
7. *Игнатьев Ю. Г., Зиатдинов Р. А.* Асимптотическое приближение модели Фоккера-Планка космологической эволюции сверхтепловых ультрарелятивистских частиц при наличии скейлинга взаимодействий // Известия Вузов, сер. Физика. — 2008. — № 42. — С. 87–92. [*Ignatjev Yu. G., Ziatdinov R. A.* Asimptoticheskoe priblizhenie modeli Fokkera-Planka kosmologicheskoyj ehvolyucii sverkhteplovihkh uljtrarelyativistskikh chastic pri nalichii skeyjlinga vzaimodeyjstviyj // Izvestiya Vuzov, ser. Fizika. — 2008. — No 42. — S. 87–92.]
8. *Игнатьев Ю. Г., Эльмахи Н.* Динамическая модель сферических возмущений во вселенной Фридмана. III. Автомодельные решения // Известия Вузов, сер. Физика. — 2008. — № 42. — С. 69–75. [*Ignatjev Yu. G., Ehljmakhi N.* Dinamicheskaya modelj sfericheskikh vozmutheniyj vo vselennoyj Fridmana. III. Avtomodeljnihe resheniya // Izvestiya Vuzov, ser. Fizika. — 2008. — No 42. — S. 69–75.]
9. *Ignatyev Y. G., Ignatyev D. Y.* Kinetics of the Nonequilibrium Universe. III. Stability of Nonequilibrium Scenario // Gravitation and Cosmology. — 2008. — No 14. — Pp. 286–292.
10. *Бахвалов Н. С., Жидков Н. П., Кобельков Г. М.* Численные методы. — М.: Бином, 2001. — 525 с. [*Bakhvalov N. S., Zhidkov N. P., Kobeljkov G. M.* Chislennihe metodih. — M.: Binom, 2001. — 525 s.]
11. *Fox L., Mayers D. F.* Numerical Solution of Ordinary Differential Equations for Scientists and Engineers. — New-York: Springer, 1987. — 624 p.
12. *Игнатьев Ю. Г., Абдулла Х. Х.* Математическое моделирование нелинейных электродинамических систем в системе компьютерной математики Maple // Вестник ТГГПУ. — 2010. — № 2 (20). — С. 22–27. [*Ignatjev Yu. G., Abdulla Kh. Kh.* Matematicheskoe modelirovanie nelineyjnihkh ehlektrodinamicheskikh sistem v sisteme kompjyuternoyj matematiki Maple // Vestnik TGGPU. — 2010. — No 2 (20). — S. 22–27.]

UDC 531;531.3;531.5;517.9; 519.6;519.8

# Mathematics Modeling of Nonlinear Generic Mechanics System in Computers Mathematic Maple

**Yu. G. Ignatyev, K. H. Abdulla**

*Mathematics Department*
*Tatar State Humanitary-Pedagogical University*
*Mezhlauk 1, 420021, Kazan, Tatarstan*

The algorithms and set of programs for mathematical modeling, within the frame of computer mathematics, of nonlinear generalized mechanical systems are presented. The built-in-system of programming procedures prove to obtain numerical solutions in the form of splines, splines-B and piecewise-defined functions. We define the spline operations as programming procedures permitting to perform the analytic calculations over the converted numerical solutions like over the ordinary functions.

**Key words and phrases:** nonlinear dynamic systems, mathematics simulation, programm complexes, splines, computer mathematic systems.