ПРОГРАММИРОВАНИЕ ОПОРНЫХ ЗАДАЧ ТЕОРИИ ПОВЕРХНОСТЕЙ В СРЕДЕ MATHEMATICA

Капустина Т.В.

E-mail: tv_kapustina@mail.ru

Елабуга, Елабужский государственный педагогический университет[1]

Аннотация. *В статье приведены программы основных задач классической теории поверхностей в трехмерном евклидовом пространстве, составленные в функциональном стиле и предназначенные для реализации в среде Mathematica. Данный материал может являться частью компьютерного учебника или компьютеризированного учебника; он приводится как дополнение к классическому изложению и позволяет автоматизировать основные вычисления с помощью среды Mathematica.*

В этой статье вниманию читателя предлагаются программы для решения нескольких опорных задач курса классической дифференциальной геометрии поверхностей трёхмерного евклидова пространства, составленные в среде Mathematica 5.0 (дальше: *Математика*).

Лучше всего оформить такой материал в виде пакета (стандартного дополнения *Математики*) [2]. Но в силу ограниченности объёма данной статьи нецелесообразно приводить здесь весь текст пакета и ещё комментарии к нему. Материал изложим в стиле "Notebook", условно полагая, что он помещён в одном документе. Входные ячейки печатаются полужирным шрифтом, а выходные - светлым, маркировка *In* и *Out* опущена.

Предварительно изложим краткие сведения о программировании в системе *Математика* (все справки можно найти в [2, 1]). Во всех примерах текст входных ячеек печатаем полужирным шрифтом, а выходных - светлым.

Функциональное программирование представляет собой стиль программирования в среде *Математика*, хорошо приспособленный для построения программы по шагам. Основная идея функционального программирования - составлять программу из функций, каждая из которых использует результаты предыдущих, т. е. сначала строится функция от аргументов, затем - функция от этой функции и т. д. Важно так писать программу, чтобы структура композиции функций была ясной. Функции, не содержащиеся в ядре системы, а задаваемые пользователем, называются *внешними*.

При определении внешней функции важно дать понять *Математике*, что её аргумент (аргументы) - это переменная (переменные). Для обозначения переменной можно использовать так называемые *именованные шаблоны*:

x_ обозначает любое выражение, представленное именем x ("_ символ подчеркивания).

x__ (с двумя символами подчеркивания) - последовательность из одного или более выражений, представленная именем x.

x___ (с тремя символами подчеркивания) - последовательность из нуля или более выражений, представленная именем x.

Если определяемая внешняя функция будет применяться в дальнейшем, целесообразно присвоить ей имя (лучше всего подойдёт заголовок, несущий информацию о её назначении).

Для примера определим пятую степень переменного x как именованную внешнюю функцию:

power5[x_] := x^5

Выходной строки нет, поскольку использовано отложенное присвоение SetDelayed (:=).

{power5[2], power5[11], power5[a]}

{32, 161051, a^5}

Вычислены значения этой функции от трёх значений аргумента.

С заголовками внешних функций можно обращаться так же, как с заголовками встроенных функций, например, применять их к элементам выражений, находящимся на определённом уровне:

Map[power5, {2, 11, a}]

{32, 161051, a^5}

Map применяет функцию (здесь - **power5**) к каждому элементу первого уровня выражения (в данном случае — списка).

Поверхность в трёхмерном пространстве, заданную относительно прямоугольной декартовой системы координат $(Oxyz)$ параметрическими уравнениями $x = x(u^1, u^2)$, $y = y(u^1, u^2)$, $z = z(u^1, u^2)$, будем обозначать именованным шаблоном **r_**; конкретную поверхность будем определять заданием списка правых частей этих уравнений.

Введём функции **r1** и **r2** для вычисления касательных векторов $\mathbf{r_1} = \partial\mathbf{r}/\partial u^1$ и $\mathbf{r_2} = \partial\mathbf{r}/\partial u^2$ к координатным линиям поверхности в произвольной точке:

[1] Аннотация на английском языке Автором не представлена.

r1[r_][u1_, u2_] := D[r[u1, u2], u1]
r2[r_][u1_, u2_] := D[r[u1, u2], u2]

Функция **normvs** будет вычислять нормальный вектор параметризованной поверхности в произвольной точке:

normvs[r_][u1_, u2_] := Cross[r1[r][u1, u2], r2[r][u1, u2]] // Simplify

Длина нормального вектора поверхности:

modnormvs[r_][u1_, u2_] :=
Sqrt[Simplify[Factor[normvs[r][u1, u2].normvs[r][u1, u2]]]]

Введём параметрические уравнения прямого геликоида $x = u^1 \cos u^2$, $y = u^1 \sin u^2$, $z = au^2$, чтобы проверять на нём действие вводимых функций:

helicoid[a_][u1_, u2_] := {u1 Cos[u2], u1 Sin[u2], a u2}

Найдём для прямого геликоида касательные векторы к координатным линиям, нормальный вектор и его длину:

r1[helicoid[a]][u1, u2]
{Cos[u2], Sin[u2], 0}
r2[helicoid[a]][u1, u2]
{-u1 Sin[u2], u1 Cos[u2], a}
normvs[helicoid[a]][u1, u2]
{a Sin[u2], –a Cos[u2], u1}
modnormvs[helicoid[a]][u1, u2]
$\sqrt{a^2 + u1^2}$

Определим функцию **nn** для вычисления *единичного вектора нормали* параметризованной поверхности в её произвольной точке; таким образом, у нас будет полностью определён сопровождающий трёхгранник поверхности $\{\mathbf{r_1}, \mathbf{r_2}, \mathbf{n}\}$.

nn[r_][u1_, u2_] := normvs[r][u1, u2] / modnormvs[r][u1, u2]

Единичный нормальный вектор прямого геликоида:

nn[helicoid[a]][u1, u2]
$\left\{\frac{a\,\mathrm{Sin}[u2]}{\sqrt{a^2+u1^2}}, -\frac{a\,\mathrm{Cos}[u2]}{\sqrt{a^2+u1^2}}, \frac{u1}{\sqrt{a^2+u1^2}}\right\}$

Для составления неявного уравнения касательной плоскости к поверхности (точнее, выражения его левой части) понадобится радиус - вектор произвольной точки этой плоскости **R**: **R := {x, y, z}**

Функция **tangplane** даёт *неявное уравнение касательной плоскости* параметризованной поверхности:

tangplane[r_][u1_, u2_] := Simplify[Expand[(R-r[u1,u2]). normvs[r][u1, u2]]] ==0

Касательная плоскость прямого геликоида в произвольной точке:

tangplane[helicoid[a]][u1, u2]
u1 (–a u2+z)–a y Cos[u2]+ a x Sin[u2]==0

Касательная плоскость прямого геликоида в точке $M_1(u^1 = 1,\ u^2 = \frac{\pi}{2})$:

% /. {u1–>1, u2–>Pi/2}
$-\frac{\mathrm{a}\,\pi}{2} + \mathrm{a\,x} + \mathrm{z} == 0$

Если потребуется строить на одном чертеже изображение поверхности и её касательной плоскости в какой-либо точке, то желательно иметь *параметрические уравнения касательной плоскости*. Для этого введём функцию **tangentplane** (здесь (u_0^1, u_0^2) — криволинейные координаты точки касания):

tangentplane[r_][u10_, u20_][u1_, u2_] := r[u10, u20] +
u1∗r1[r][u10, u20] + u2∗r2[r][u10, u20]

Список правых частей параметрических уравнений касательной плоскости к прямому геликоиду в произвольной точке:

tangentplane[helicoid[a]][u10, u20][u1, u2]
{u10 Cos[u20]+u1 Cos[u20]–u10 u2 Sin[u20], u10 u2 Cos[u20]+ u10 Sin[u20]+u1 Sin[u20],
a u20+a u2}

То же в точке $M_1(u_0^1 = 1, u_0^2 = \frac{\pi}{2})$:

%/.{u10–>1, u20–>Pi/2}
{-u2, 1 + u1, $\frac{\mathrm{a}\pi}{2}$ + a u2}

Введём функции для вычисления *коэффициентов первой квадратичной формы* (координат метрического тензора) поверхности; их обозначения близки к стандартным (g_{11}, g_{12}, g_{21}, g_{22}):

g11[r_][u1_, u2_] := Simplify[r1[r][u1, u2].r1[r][u1, u2]]
g12[r_][u1_, u2_] := Simplify[r1[r][u1, u2].r2[r][u1, u2]]
g21[r_][u1_, u2_] := g12[r][u1, u2]
g22[r_][u1_, u2_] := Simplify[r2[r][u1, u2].r2[r][u1, u2]]

Убедимся на примере прямого геликоида, что эти функции работают:

g11[helicoid[a]][u1, u2]

1
g12[helicoid[a]][u1, u2]
0
g22[helicoid[a]][u1, u2]
$a^2 + u1^2$
g21[helicoid[a]][u1, u2]
0

Функция **discrim1** будет предназначаться для вычисления *дискриминанта* первой квадратичной формы $g_{11}g_{22} - g_{12}^2$:

discrim1[r_][u1_, u2_] :=
Simplify[g11[r][u1, u2]*g22[r][u1,u2]−g12[r][u1, u2]^2]

Дискриминант первой квадратичной формы прямого геликоида:

discrim1[helicoid[a]][u1, u2]
$a^2 + u1^2$

Линию на поверхности, задаваемую внутренними параметрическими уравнениями $u^1 = u^1(t)$, $u^2 = u^2(t)$, будем обозначать именованным шаблоном **alpha_**; его значением будет список правых частей внутренних параметрических уравнений линии. Функцию **dst** введём для вычисления подынтегральной функции в формуле длины дуги линии, принадлежащей поверхности (фактически это есть значение $\dfrac{ds}{dt}$, где s — длина дуги):

dst[r_][u1_, u2_][alpha_] := Sqrt[Simplify[
Factor[g11[r][u1, u2]*D[First[alpha], t]^2+
2 g12[r][u1, u2]*D[First[alpha], t]*D[Last[alpha], t]+
g22[r][u1, u2]*D[Last[alpha], t]^2]]] /. {u1−>First[alpha], u2−>Last[alpha]}

Зададим кривую внутренними параметрическими уравнениями $u^1 = t$, $u^2 = t$

alph1 := {t, t}

и найдём значение функции **dst** (то есть $\dfrac{ds}{dt}$) для такой кривой на прямом геликоиде:

dst[helicoid[a]][u1, u2][alph1]
$\sqrt{1 + a^2 + t^2}$

Далее введём функцию **s** для вычисления *длины дуги кривой на поверхности* (именованные шаблоны **aa_** и **bb_** обозначают пределы интегрирования, т. е. значения параметра t концов дуги):

s[r_][u1_, u2_][alpha_][aa_, bb_] :=
Integrate[dst[r][u1, u2][alpha], {t, aa, bb}]

Вычислим длину дуги кривой $u^1 = t$, $u^2 = t$ между точками $t = 0$ и $t = 1$ на прямом геликоиде:

s[helicoid[a]][u1, u2][alph1][0, 1]
$-\dfrac{1}{4}\,(1 + a^2)\,\mathrm{Log}[1 + a^2] + \dfrac{1}{2}(\sqrt{2 + a^2} + (1 + a^2)\,\mathrm{Log}[1 + \sqrt{2 + a^2}])$

Введём параметрические уравнения координатной линии $u^2 = \dfrac{\pi}{2}$ и вычислим длину её дуги между координатными линиями $u^1 = 0$ и $u^1 = 1$ на прямом геликоиде:

alph2 := {t, Pi/2}
s[helicoid[a]][u1, u2][alph2][0, 1]
1

Для координатной линии $u^1 = 2$ на прямом геликоиде найдём длину её дуги между координатными линиями $u^2 = 0$ и $u^2 = \pi$:

alph3 := {2, t}
s[helicoid[a]][u1, u2][alph3][0, Pi]
$\sqrt{4 + a^2}\,\pi$

Наконец, найдём длину дуги кривой $u^1 = t$, $u^2 = \ln(t + \sqrt{t^2 + a^2})$ в пределах $t = 0$, $t = \sqrt{2}$ на прямом геликоиде:

alph4 := {t, Log[t+Sqrt[t^2+a^2]]}
s[helicoid[a]][u1, u2][alph4][0, Sqrt[2]]
2

С помощью функции **areasurf** можно будет находить *площадь области на поверхности*:

areasurf[r_][u1_, u2_][aa_, bb_, cc_, dd_] :=
Integrate[Sqrt[discrim1[r][u1, u2]], {u1, aa, bb}, {u2, cc, dd}]

Найдём площадь криволинейного черырехугольника на прямом геликоиде, ограниченного координатными линиями $u^1 = 0$, $u^1 = a$, $u^2 = 0$, $u^2 = 1$:

areasurf[helicoid[a]][u1, u2][0, a, 0, 1]
$-\dfrac{1}{4}\,a^2\,\mathrm{Log}[a^2] + \dfrac{1}{2}\,a\,(\sqrt{2}\,\sqrt{a^2} + a\,\mathrm{Log}[a + \sqrt{2}\,\sqrt{a^2}])$

Введём следующие функции для вычисления *коэффициентов второй квадратичной формы* поверхности (координат тензора h_{ij}):

```
h11[r_][u1_, u2_] := Simplify[Det[{r1[r][u1, u2], r2[r][u1, u2],
    D[r1[r][u1, u2], u1]}] / Sqrt[discrim1[r][u1, u2]]]
h12[r_][u1_, u2_] := Simplify[Det[{r1[r][u1, u2], r2[r][u1, u2],
    D[r1[r][u1, u2], u2]}] / Sqrt[discrim1[r][u1, u2]]]
h21[r_][u1_, u2_] := h12[r][u1, u2]
h22[r_][u1_, u2_] := Simplify[Det[{r1[r][u1, u2], r2[r][u1, u2],
    D[r2[r][u1, u2], u2]}] / Sqrt[discrim1[r][u1, u2]]]
```

Подсчитаем коэффициенты второй квадратичной формы прямого геликоида:

h11[helicoid[a]][u1, u2]

0

h12[helicoid[a]][u1, u2]

$-\frac{\mathrm{a}}{\sqrt{\mathrm{a}^2+\mathrm{u1}^2}}$

h22[helicoid[a]][u1, u2]

0

Функцию **discrim2** предназначим для вычисления дискриминанта второй квадратичной формы $h_{11}h_{22}-h_{12}^2$:

```
discrim2[r_][u1_, u2_] :=
    Simplify[h11[r][u1, u2]*h22[r][u1, u2]–h12[r][u1, u2]^2]
```

Функция **gcurvature** будет вычислять *полную (гауссову) кривизну* поверхности в произвольной точке (общепринятое в математической литературе обозначение гауссовой кривизны K):

```
gcurvature[r_][u1_, u2_] :=
    Simplify[discrim2[r][u1, u2]/discrim1[r][u1, u2]]
```

Получим выражение для гауссовой кривизны прямого геликоида:

gcurvature[helicoid[a]][u1, u2]

$-\frac{\mathrm{a}^2}{(\mathrm{a}^2+\mathrm{u1}^2)^3}$

Среднюю кривизну поверхности (математическое обозначение H) будем находить при помощи функции **meancurvature**:

```
meancurvature[r_][u1_, u2_] := Simplify[Factor[(
   g11[r][u1, u2]*h22[r][u1, u2]–2*g12[r][u1, u2]*h12[r][u1, u2]+
    g22[r][u1, u2]*h11[r][u1, u2])/(2 discrim1[r][u1, u2])]]
```

Подсчитаем среднюю кривизну прямого геликоида в его произвольной точке:

meancurvature[helicoid[a]][u1, u2]

0

Введём внешние функции **k1** и **k2** для вычисления *главных кривизн* поверхности k_1 и k_2:

```
k1[r_][u1_, u2_] := meancurvature[r][u1, u2] +
    Sqrt[Simplify[ meancurvature[r][u1, u2]^2–gcurvature[r][u1, u2]]]
k2[r_][u1_, u2_] := meancurvature[r][u1, u2] –
    Sqrt[Simplify[ meancurvature[r][u1, u2]^2–gcurvature[r][u1, u2]]]
```

Найдём главные кривизны прямого геликоида:

k1[helicoid[a]][u1, u2]

$\sqrt{\frac{\mathrm{a}^2}{(\mathrm{a}^2+\mathrm{u1}^2)^3}}$

k2[helicoid[a]][u1, u2]

$-\sqrt{\frac{\mathrm{a}^2}{(\mathrm{a}^2+\mathrm{u1}^2)^3}}$

В заключение приведём программу для подсчета коэффициентов связности и для интегрирования уравнений геодезических линий параметризованной поверхности.

```
G[r_][u1_, u2_] := {{g11[r][u1, u2], g12[r][u1, u2]}, {g21[r][u1, u2],
    g22[r][u1, u2]}}
GG[r_][u1_, u2_] := Inverse[G[r][u1, u2]]
```

Функция **G** даёт тензор первой квадратичной формы поверхности, а функция **GG** — сопряжённый ему тензор.

```
uv := {u1, u2}
cristoffel[i_, j_, k_][r_][u1_, u2_] := 1/2*Simplify[
    GG[r][u1, u2][[k, 1]]*(D[G[r][u1, u2][[1, j]], uv[[i]]]+
    D[G[r][u1, u2][[i, 1]], uv[[j]]] – D[G[r][u1, u2][[i, j]], u1])+
    GG[r][u1, u2][[k, 2]]*(D[G[r][u1, u2][[2, j]], uv[[i]]]+
    D[G[r][u1, u2][[i, 2]], uv[[j]]] – D[G[r][u1, u2][[i,j]], u2])]
```

Внешняя функция **cristoffel** служит для подсчёта коэффициентов связности поверхности $\Gamma_{ij}^k = \dfrac{1}{2} g^{ks}(\partial_i g_{sj} + \partial_j g_{is} - \partial_s g_{ij})$. Как видим, программа весьма компактна. Её действие проверим на прмере геликоида:

cristoffel[1,2,2][helicoid[1]][u1, u2]

$\dfrac{\mathrm{u1}}{1+\mathrm{u1}^2}$

Функции **eq1** и **eq2** дадут систему дифференциальных уравнений для определения геодезических линий поверхности:

```
eq1[r_] := u1''[s]+Simplify[cristoffel[1,1,1][r][u1[s], u2[s]]*
    u1'[s]*u1'[s]+2 cristoffel[1,2,1][r][u1[s], u2[s]]*
    u1'[s]*u2'[s]+cristoffel[2,2,1][r][u1[s], u2[s]]*
    u2'[s]*u2'[s], Trig->True] = = 0
eq2[r_] := u2''[s]+Simplify[cristoffel[1,1,2][r][u1[s], u2[s]]*
    u1'[s]*u1'[s]+2 cristoffel[1,2,2][r][u1[s], u2[s]]*
    u1'[s]*u2'[s]+cristoffel[2,2,2][r][u1[s], u2[s]]*
    u2'[s]*u2'[s], Trig->True] = = 0
```

Функции **geodline** и **geod** предназначены соответственно для получения приближенного и точного решений системы дифференциальных уравнений геодезических:

```
geodline[r_][u10_, u20_, p1_, p2_, sb_, se_] :=
    NDSolve[eq1[r], eq2[r], u1[0]= =u10, u2[0]= =u20,
    u1'[0]= =p1, u2'[0]==p2, {u1, u2}, {s, sb, se}]
geod[r_][u10_, u20_, p1_, p2_] := DSolve[eq1[r], eq2[r],
    u1[0]==u10, u2[0]==u20, u1'[0]==p1, u2'[0]==p2, {u1[s], u2[s]}, s]
```

Продемонстрируем действие этих функций на примере цилиндра:

cilinder[u1_, u2_] := {Cos[u1], Sin[u1], u2}

geod[cilinder][0,0,1,2]

{{u1[s]−> s, u2[s]−>2 s}}

После визуализации получаем изображение:

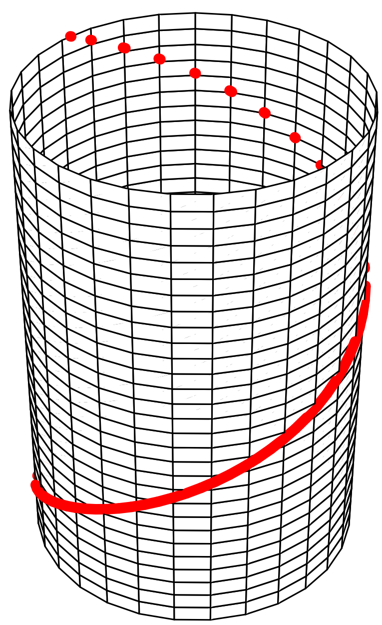

Рис.1. *Геодезическая на цилиндре*

geodline[cilinder][0,0,1,1,0,2]

{{u1−>InterpolatingFunction[0.,2.,<>], u2−>InterpolatingFunction [0.,2.,<>]}}

С интерполяционными функциями можно работать практически так же, как и с обычными: подставлять их в параметрические уравнения поверхности для получения параметрических уравнений геодезической, определяемой начальными условиями, использованными в процессе приближённого решения системы, и применять эти параметрические уравнения для визуализации геодезической.

Примерно в таком же ключе можно разработать приёмы решения и других опорных задач теории поверхностей: нахождения нормальной кривизны, составления уравнений линий кривизны поверхности и т. п.

Литература

[1] Воробьёв Е. М. *Введение в систему символьных, графических и численных вычислений "Математика–5"*: Учеб. пособие / Е. М. Воробьёв. – М.: Диалог-МИФИ, 2005. – 368 с.: ил.

[2] Капустина Т. В. *Компьютерная система Mathematica 3.0 для пользователей*: Справ. пособие / Т. В. Капустина. – М.: СОЛОН-Р, 1999. – 240 с.: ил.

[3] Gray A. *Modern Differential Geometry of Curves and Surfaces.* – Cambridge University Press. – 1997.

[4] Wolfram S. *The Mathematica Book.* Fourth Edition. Mathematica Version 4. – Wolfram Media / Cambridge University Press, 1999.