6. Решение дифференциальных уравнений

6.1 Общие сведения

Для решения дифференциальных уравнений пользователю предлагается большой набор функций, основное количество которых расположено в библиотеке DEtools.

Решить дифференциальное уравнение или систему уравнений можно с помощью встроенной команды *dsolve.*

Формат команды:

dsolve(deqns, vars) или

dsolve(deqns, vars, eqns),

где **deqns** – одно дифференциальное уравнение или множество уравнений и/или начальных условий, **vars** – переменная или множество неизвестных переменных, **eqns** – необязательные выражения в форме *ключевое слово = значение*.

Команда *dsolve* позволяет определить решение многих дифференциальных уравнений. По умолчанию *dsolve* постарается найти точное решение дифференциального уравнения. Однако, если точное решение не может быть получено, то можно попытаться найти приближенное решение с помощью разложения его в ряд (параметр *type=series*) или численным методом (параметр *type=numeric*).

Примеры:

Решим дифференциальное уравнение без начальных условий:

```
>dsolve(t*diff(y(t),t)-t^2*y(t)=cos(t),y(t)  );
```

$$y(t) = \mathbf{e}^{\left(\frac{1}{2}t^2\right)}\left(\int \frac{\mathbf{e}^{\left(-\frac{1}{2}t^2\right)}\cos(t)}{t}\,dt + \_C1\right)$$

Найдем приближенное решение того же уравнения с начальными условиями при помощи параметра *type=series*. Число членов ряда можно изменить, присвоив параметру *Order* требуемое количество членов ряда:

```
>Order:=4;
```

$$Order := 4$$

```
>dsolve({t*diff(y(t),t)-t^2*y(t)=sin(t),
y(0)=1},y(t),type=series);
```

$$y(t) = 1 + t + \frac{1}{2}t^2 + \frac{5}{18}t^3 + O(t^4)$$

Важно отметить, что начальные условия, заданные производными, записываются в следующем виде, например, вторая производная от y в точке ноль: D(D(y))(0) или (D@@2)(y)(0).

Для решения дифференциальных уравнений, содержащих функции Дирака или Хевисайда, можно использовать опцию *method=laplace*. При этом при решении дифференциального уравнения будет использоваться преобразование Лапласа.

Пример:

```
>DifEqn1 := diff(y(t),t$2) + 5*diff(y(t),t) +
6*y(t) = Heaviside(t);
```

$$DifEqn1 := \left(\frac{\partial^2}{\partial t^2} y(t)\right) + 5\left(\frac{\partial}{\partial t} y(t)\right) + 6\, y(t) = \text{Heaviside}(t)$$

```
>dsolve({DifEqn1, y(0)=0, D(y)(0)=1}, y(t),
method=laplace);
```

$$y(t) = \frac{1}{6} - \frac{2}{3}\mathbf{e}^{(-3t)} + \frac{1}{2}\mathbf{e}^{(-2t)}$$

С помощью команды *dsolve* можно находить решение дифференциального уравнения численным методом. Пользователю предлагается 6 базовых методов. Следует отметить, что каждый метод имеет огромное количество опций, подробную информацию о которых можно получить в справочной системе Maple V Release 4 или в книге Прохоров Г.В., Леденёв М. А., Колбеев В.В. Пакет символьных вычислений Maple V. - М.: Компания "Петит", 1997.

Название опции	Описание метода
method=rkf45	Метод Рунге–Кутта четвертого–пятого порядка (параметр по умолчанию).
method=dverk78	Метод Рунге–Кутта седьмого–восьмого порядка.
method=classical[метод]	Содержит несколько основных методов: *foreuler* – прямой метод Эйлера; *heunform* – усовершенствованный метод Эйлера; *impoly* – модифицированный метод Эйлера *rk2* – классический метод Рунге–Кутта второго порядка; *rk3* – классический метод Рунге–Кутта третьего порядка; *rk4* – классический метод Рунге–Кутта четвертого порядка; *adambash* – метод Адамса–Башфорда (метод «прогноза»); *abmoulton* – метод Адамса-Моултона (метод «прогноза и коррекции»).
method=gear	Одношаговый метод Гира
method=mgear	Многошаговый метод Гира
method=lsode	Метод решения Ливермора Стиффа.

Пример:

```
>syst:=diff(y(x),x)=f(x)-y(x)-x,diff(f(x),x)= y(x):
>func := {y(x), f(x)}:
>Fout := dsolve({syst,y(0)=0,f(0)=1},func,
type=numeric,method=classical[rk4]);
```

Fout := **proc**(*x_classical*) ... **end**

Далее можно распечатать значение решения в любой точке:

```
>Fout(1);
```

$$[x = 1, y(x) = .3437314050154296, f(x) = 1.258972078481968]$$

6.2 Решение дифференциальных уравнений с частными производными

Для решения дифференциальных уравнений с частными производными служит команда *pdesolve.*

Формат команды:

pdesolve(deqns, vars),

где **deqns** – дифференциальное уравнение с частными производными, **vars** – неизвестные переменные.

Команда *pdesolve* позволяет найти решения многих уравнений с частными производными. Произвольные функции выводятся в виде:_F1, _F2 и т. д.

Примеры:

```
>pdesolve(y*diff(F(x,y),x)+x*diff(F(x,y),y)=0,
F(x,y));
```

$$F(x,y) = \_F1(-x^2 + y^2)$$

```
>pdesolve(  3*diff(g(x,y),x,x)+7*diff(g(x,y),x
,y)=x*y, g(x,y) );
```

$$g(x,y) = -\frac{7}{216}x^4 + \frac{1}{18}x^3 y + \_F1(y) + \_F2(3y - 7x)$$

Для получения детальной информации о дифференциальном уравнении и этапах его решения следует выполнить команду:

```
>infolevel[pdesolve]:=5:
```

И затем решить дифференциальное уравнение.

6.3 Библиотека DEtools

Содержит 17 функций для работы с дифференциальными уравнениями и для построения решений дифференциальных уравнений

ФУНКЦИЯ	ДЕЙСТВИЕ
Denormal	Возвращает “нормализованную” форму дифференциального уравнения.
Deplot	Построение графика решения дифференциального уравнения или системы дифференциальных уравнений.
DEplot3d	Построение решения системы дифференциальных уравнений в виде трехмерного графика.
Dchangevar	Замена переменных в обыкновенном дифференциальном уравнении.
PDEchangecoords	Переход к другой системе координат в дифференциальном уравнении в частных производных.
PDEplot	Построение решения дифференциального уравнения с частными производными.
autonomous	Определяет, является ли система дифференциальных уравнений автономной.
convertAlg	Возвращает список коэффициентов дифференциального уравнения.
convertsys	Преобразует систему дифференциальных уравнений к системе первого порядка.
dfieldplot	Построение поля решения для системы дифференциальных уравнений.
phaseportrait	Построение решения дифференциального уравнения.
reduceOrder	Применят метод понижения порядка к дифференциальному уравнению.

Функция DEnormal.

Возвращает "нормализованную" форму дифференциального уравнения.

Формат команды:

DEnormal(des,ivar,dvar),

где **des** – дифференциальное уравнение или список из операторов, **ivar** – независимые переменные, **dvar** – зависимые переменные.

Под "нормализацией" понимается то, что коэффициенты дифференциального уравнения, списка дифференциальных операторов или результата выполнения команды DEtools[convertAlg] представляются в виде полиномов.

```
>with(DEtools):
DE := 2*(-2*x^3)/(x-5)*y(x) + 4*(x^2)/
(x+1)*x*D(y)(x)+5*x^3/(x-1)^2*(D@@2)(y)(x) =
x*;
```

$$DE := -4\frac{x^3\,\mathrm{y}(x)}{x-5} + 4\frac{x^3\,\mathrm{D}(y)(x)}{x+1} + 5\frac{x^3\,(D^{(2)})(y)(x)}{(x-1)^2} = x$$

```
>DEnormal(DE,x,y(x));
```

$$(4x^3 - 28x^2 + 44x - 20)\left(\frac{\partial}{\partial x}\mathrm{y}(x)\right) + (5x^2 - 20x - 25)\left(\frac{\partial^2}{\partial x^2}\mathrm{y}(x)\right) - 4\,\mathrm{y}(x)\,x^3$$
$$+ 4\,\mathrm{y}(x)\,x^2 + 4\,\mathrm{y}(x)\,x - 4\,\mathrm{y}(x) = x^2 - 6x + 4 + \frac{6}{x} - \frac{5}{x^2}$$

Функция DEplot

Построение графика решения дифференциального уравнения или системы дифференциальных уравнений.

Формат команды:

DEplot(deqns, vars, trange, eqns)

DEplot(deqns, vars, trange, inits, eqns)

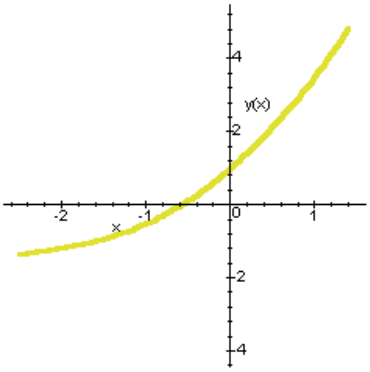
DEplot(deqns, vars, trange, yrange, xrange, eqns)

DEplot(deqns, vars, trange, inits, xrange, yrange, eqns),

где **deqns** – список или множество дифференциальных уравнений первого порядка или дифференциальное уравнение любого порядка, **vars** – зависимая переменная, список или множество зависимых переменных, **trange** – область изменения независимой переменной, **inits** – начальные условия для кривой решения, **yrange** – область изменения первой зависимой переменной, **xrange** – область изменения второй зависимой переменной, **eqns** – необязательное выражение вида *keyword=value*.

Пример:

```
>DEplot({diff(y(x),x$3)-diff(y(x),x$2)+diff
(y(x),x)=y(x)-x},{y(x)},x=-2.5..1.4,[[y(0)=1,
D(y)(0)=2,(D@@2)(y)(0)=1]],y=-4..5,stepsize
=.05);
```



Функция DEplot3d

Построение решения системы дифференциальных уравнений в виде трехмерного графика.

Формат команды:

DEplot3d(deqns, vars, trange, initset, options),

DEplot3d(deqns, vars, trange, yrange, xrange, initset, options),

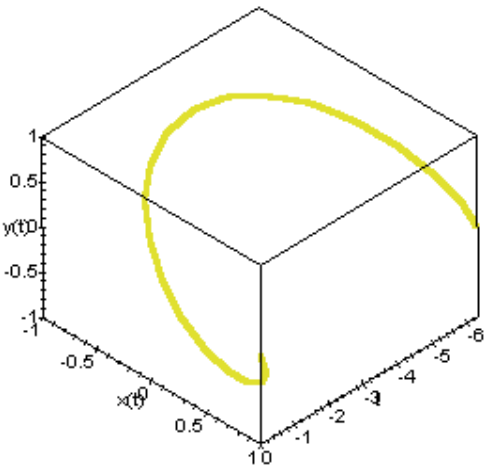
где **deqns** – список или множество обыкновенных дифференциальных уравнений, **vars** – зависимые переменные, **trange** – область изменения независимой переменной, **yrange** – область изменения первой зависимой перемен-

ной, **xrange** – область изменения второй зависимой переменной, **initset** – множество начальных условий для кривой решения, **options** – необязательное выражение вида *keyword=value*.

Для заданной системы обыкновенных дифференциальных уравнений численным методом строится кривая решения. По умолчанию используется метод Рунге – Кутта 4-го порядка. Следует отметить что в системе должна быть только одна независимая переменная.

Пример:

```
>DEplot3d({diff(x(t),t)=-sin(t),diff(y(t),t)=
cos(t)},{y(t),x(t)},t=-2*Pi..0,[[y(0)=0,x(0)=1]
]);
```



Функция Dchangevar

Замена переменных в обыкновенном дифференциальном уравнении.

Формат команды:

Dchangevar(trans, deqns, c_ivar, n_ivar),

Dchangevar(tran1, tran2, ..., tranN, deqns, c_ivar, n_ivar),

где **trans** – список или множество выражений, подставляемых в уравнение, **tran1, tran2, ..., tranN** – выражения, подставляемые в уравнение, **deqns** – дифференциальное уравнение, **c_ivar** – имя существующей независимой переменной, **n_ivar** – имя новой независимой переменной (необязательный параметр).

С помощью данной команды можно провести замену переменных в диф-

ференциальном уравнении или системе дифференциальных уравнений.

Возможны три ситуации:

1. Изменение имени зависимой переменной. Это означает изменение зависимой переменной. Однако независимая переменная не изменяется.
2. Замена независимой переменной. При этом старая переменная будет заменена новым выражением.
3. Изменение зависимой переменной на переменную, зависящую от других аргументов.

Примеры:

Преобразования, описанные в пункте 1:

```
>Dchangevar(m(x)=l(x)*sin(x),n(x)=k(x),[D(m)(x)=m(x),
(D@@2)(n)(x)=n(x)^2],x);
```

$$[\mathrm{D}(l)(x)\sin(x)+l(x)\cos(x)=l(x)\sin(x),(D^{(2)})(k)(x)=\mathrm{k}(x)^2]$$

Преобразования, описанные в пункте 2:

```
>Dchangevar(t=arctan(tau),diff(x(t),t)=sin(t),t,tau);
```

$$\mathrm{D}(x)(\arctan(\tau))=\sin(\arctan(\tau))$$

Преобразования, описанные в пункте 3:

```
>Dchangevar(x(t)=L*y(phi),diff(x(t),t$3)  =
tan(t),t,phi);
```

$$\frac{\partial^3}{\partial\phi^3}L\,\mathrm{y}(\phi)=\tan(\phi)$$

Функция PDEchangecoords

Переход к другой системе координат в дифференциальном уравнении в частных производных.

Формат команды:

PDEchangecoords(pdes, c_ivar, c_name, n_ivar),

где **pdes** – дифференциальное уравнение с частными производными или система дифференциальных уравнений, **c_ivar** – список существующих независимых переменных, **c_name** – имя новой системы координат, **n_ivar** – список новых независимых переменных (необязательный параметр).

Осуществляется переход к новой системе координат в терминах частного дифференцирования. Для двухмерных систем возможен переход к 14 новым системам. Вот некоторые из них:

bipolar – биполярная;
hyperbolic – гиперболическая;
logarithmic – логарифмическая;
parabolic – параболическая;
polar – полярная.

Для трехмерных систем возможна трансформация в 29 новых систем координат. Вот лишь некоторые из них:

conical – коническая;
cylindrical – цилиндрическая;
spherical – сферическая;
toroidal – тороидальная.

Для получения большей информации о системах координат следует обратиться к справочной системе Maple V по ключевому слову “coords”.

Примеры:

```
>PDEchangecoords(diff(z(x,y),x),[x,y],polar);
```

$$\frac{\cos(y)\left(\frac{\partial}{\partial x}z(x,y)\right)x-\sin(y)\left(\frac{\partial}{\partial y}z(x,y)\right)}{x}$$

Функция PDEplot

Построение решения дифференциального уравнения в частных производных.

Формат команды:

PDEplot(pdiffeq, vars, i_curve, srange, options),

PDEplot(pdiffeq, vars, i_curve, srange, xrange, yrange, urange, options),

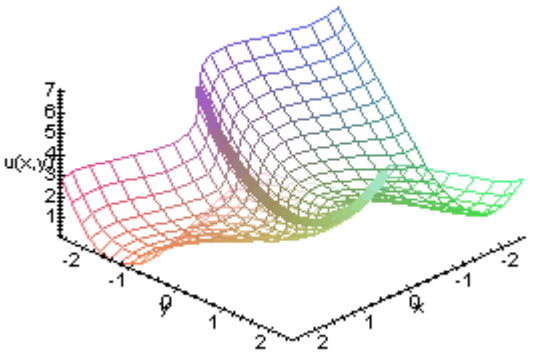
где **pdiffeq** – дифференциальное уравнение с частными производными первого порядка, **vars** – зависимые переменные, **i_curve** – начальные условия, **srange** – область изменения параметров начальных условий , **xrange** – область изменения первой независимой переменной, **yrange** – область изменения второй независимой переменной, **urange** – область изменения зависимой переменной.

Для уравнения с частными производными строится кривая реше-

ния. Необходимо помнить, что начальные условия задаются в виде списка из трех элементов, которые означают параметрическую форму кривой в трехмерном пространстве.

Пример:

```
>PDEplot(D[1](u)(x,y)+cos(2*x)*D[2](u)(x,y)=-
sin(y)*u(x,y),[0,s,1+s^2],s=-2..2);
```



Функция autonomous

Определяет, является ли множество дифференциальных уравнений автономным.

Формат вызова:

autonomous(des,vars,ivar),

где **des** – дифференциальное выражение (или множество), **vars** – список или множество зависимых переменных, **ivar** – список или множество независимых переменных.

Пример:

```
>autonomous(sin(z(t)-z(t)^2)*(D@@4)(z)(t)-cos
(z(t))-5,z,t);
```

true

Функция convertAlg

Возвращает список коэффициентов дифференциального уравнения.

Формат вызова:

convertAlg(des,dvar),

где **des** – дифференциальное выражение, **dvar** – зависимая переменная.

Пример:

```
>convertAlg(diff(x(t),t)*sin(t)+5*x(t)=cos(t),x(t));
```

$$[[5, \sin(t)], \cos(t)]$$

Функция convertsys

Преобразует систему дифференциальных уравнений к системе первого порядка.

Формат команды:

convertsys(deqns, inits, vars, ivar, yvec, ypvec),

где **deqns** – обыкновенное дифференциальное уравнение, система или множество дифференциальных уравнений, **inits** – множество или список начальных условий, **vars** – зависимая переменная или переменные в системе, **ivar** – независимые переменные, **yvec** – имя, которое будет использоваться для вектора решения в системе первого порядка, **ypvec** – тоже самое для его производной.

Функция преобразует одно уравнение или систему дифференциальных уравнений к системе первого порядка. Соответствующие начальные условия также преобразуются.

Выходной список имеет вид:

[eqnlist, Ydefs, x0, Y0], где **eqnlist** – список выражений вида Y'(x) = f(x,Y(x)), **Ydefs** – вектор неизвестных параметров в системе, **x0** – точкав которой определены начальные условия, **Y0** – значения начальных условий.

Пример:

```
> deq1:=diff(y(t),t$2)=y(t)-x(t), diff(x(t) ,
t) = x(t):
>  init1 := y(0) = 1, D(y)(0) = 2, x(0) = 3:
>  convertsys({deq1}, {init1}, {x(t), y(t)}, t,
y, y_p);
```

$$\left[[Y\_p_1 = Y_1, Y\_p_2 = Y_3, Y\_p_3 = Y_2 + Y_1], \left[Y_1 = \mathrm{x}(t), Y_2 = \mathrm{y}(t), Y_3 = \frac{\partial}{\partial t}\mathrm{y}(t) \right], 0, [1, 0, 1] \right]$$

Функция dfieldplot

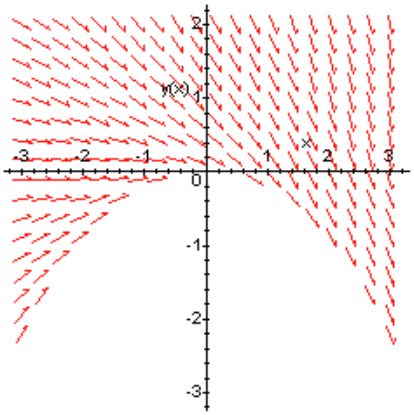
Построение поля решения для системы дифференциальных уравнений.

Формат команды:

аналогичен команде DEplot.

Пример:

```
>dfieldplot(diff(y(x),x)=1/2*(-x-(x^2+4*y(x))
^(1/2)),y(x),x=-3..3,y=-3..2);
```



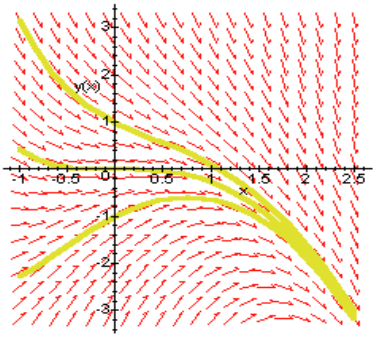
Функция phaseportrait

Построение решения дифференциального уравнения.

Формат команды:

аналогичен команде DEplot.

Пример:



Функция reduceOrder

Применят метод понижения порядка к дифференциальному уравнению. Формат вызова:

reduceOrder(des,dvar,partsol, solutionForm),

где **des** – обыкновенное дифференциальное уравнение, **dvar** – зависимая переменная, **partsol** – одно из решений дифференциального уравнения или список из нескольких решений, **solutionForm** – флаг, означающий, что будет выведено точное решение.

Если не используется флаг *solutionForm*, то будет найдено дифференциальное уравнение более низкого порядка, если флаг *solutionForm* установить в значение basis, то будет применен метод понижения порядка для решения дифференциального уравнения.

Пример:

```
>de := diff(y(x),x$3) - 6*diff(y(x),x$2) + 11*
diff(y(x),x) - 6*y(x);
```

$$de := \left(\frac{\partial^3}{\partial x^3}\,y(x)\right) - 6\left(\frac{\partial^2}{\partial x^2}\,y(x)\right) + 11\left(\frac{\partial}{\partial x}\,y(x)\right) - 6\,y(x)$$

```
>sol := exp(x);
```

$$sol := \mathbf{e}^{x}$$

```
>reduceOrder( de, y(x), sol);
```

$$\left(\frac{\partial^2}{\partial x^2}\,y(x)\right) - 3\left(\frac{\partial}{\partial x}\,y(x)\right) + 2\,y(x)$$

Для решения уравнения воспользуемся параметром basis:

```
>reduceOrder( de, y(x), sol, basis);
```

$$\left[\mathbf{e}^{x}, \mathbf{e}^{(2\,x)}, \frac{1}{2}\,\mathbf{e}^{(3\,x)}\right]$$

6.4 Пример из теории автоматического управления

В теории автоматического управления (ТАУ) широко используются временные характеристики исследуемой системы управления. Одна из таких характеристик – импульсная переходная функция (ИПФ), которая показывает реакцию системы на импульсное воздействие (т.е. входной сигнал – дельта-функция).

В качестве примера построим ИПФ линейной нестационарной системы управления.

Рассмотрим простейшую систему – колебательное звено с изменяющимися во времени параметрами:

```
> restart:
> xi:=sin(t): T1:=0.6*(1.22^t+0.01):
```

Опишем систему управления дифференциальным уравнением:

```
> eq:=T1*diff(x(t),t$2)+xi*diff(x(t),t)+x(t)=
10*Heaviside(t-tau);
```

$$eq := (.6\ 1.22^{t} + .006)\left(\frac{\partial^2}{\partial t^2}\mathrm{x}(t)\right) + \sin(t)\left(\frac{\partial}{\partial t}\mathrm{x}(t)\right) + \mathrm{x}(t) = 10\ \mathrm{Heaviside}(t-\tau)$$

где t – текущее время;

τ – время поступления на вход системы воздействия.

Примечание: из-за особенностей функции dsolve в качестве входного воздействия используется единичная функция. После решения дифференциального уравнения в качестве результата надо брать не функцию x(t), а ее первую производную по времени.

Т.к. коэффициенты дифференциального уравнения зависят от времени, то решить такое уравнение можно только численным методом.

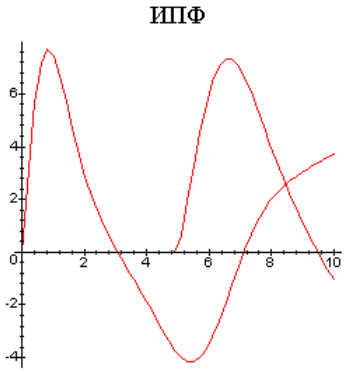
```
> r:=dsolve({eq,x(0)=0,D(x)(0)=0},x(t),
type=numeric);
```

$$r := \mathrm{proc}(rkf45\_x)\ \ldots\ \mathrm{end}$$

Построим две ИПФ для разных значений τ:

```
> tau:=0:
> pp1:=plots[odeplot](r,[t,diff(x(t),t)],
0..10):
```

```
> tau:=5:
> pp2:=plots[odeplot](r,[t,diff(x(t),t)],0..10):
> plots[display]({pp1,pp2},title=`ИПФ`);
```



А теперь построим всю ИПФ для заданной системы. Рассматриваемая система нестационарная, поэтому ИПФ зависит от двух переменных: t и τ. Следовательно, график ИПФ получится трехмерным, причем, в следствие линейности системы поверхность будет гладкой.

```
> a:=array(0..20,0..20);
```

$$a := \mathrm{array}(0 .. 20, 0 .. 20, [\ ])$$

```
> for x from 0 to 20 do
> for y from 0 to 20 do
> tau:=x/2:
>  a[x,y]:=rhs(op(3,r(y/2))):
> od:
> od:

> lst:= [seq([seq( [i/2,j/2,a[i,j]], i=0..20)],
j=0..20)]:
> with(plots):
> surfdata(lst,axes = framed,labels =
```

```
[`Tau`,`t`,`k(t,Tau)`],orientation=[55,55],
style=patch,title=`ИПФ`);
```

ИПФ