Доля П.Г.
Харьковский Национальный Университет
механико – математический факультет
2011 г.

Примеры решения математических задач в MAPLE.

Оглавление

1. Разложение функции в ряд Фурье

В Maple не существует специального пакета для построения разложения функций в ряды Фурье. Однако разработка процедуры, выполняющей такое построение, не вызывает осложнений.

Рядом Фурье периодической с периодом *2l* функции f(x) называется тригонометрический ряд вида

$$\frac{1}{2}a_0+\sum_{n=1}^{\infty}\left(a_n\cos\frac{n\pi x}{l}+b_n\sin\frac{n\pi x}{l}\right)$$

в котором коэффициенты вычисляются по следующим формулам

$$a_0=\frac{2}{l}\int_{-l}^{l}f(x)dx,\quad a_n=\frac{1}{l}\int_{-l}^{l}f(x)\cos\frac{n\pi x}{l}dx,\quad b_n=\frac{1}{l}\int_{-l}^{l}f(x)\sin\frac{n\pi x}{l}dx.$$

Создадим процедуру разложения в ряд Фурье алгебраического выражения. Ее параметрами являются само выражение, имя независимой переменной, по которой выражение раскладывается в ряд, значение полупериода *l* и количество удерживаемых членов.

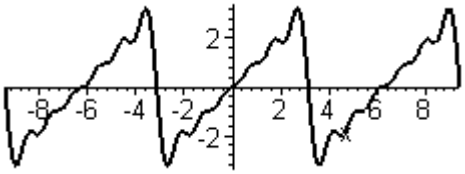
```
FourierSeries:=proc(f::algebraic,x::name,
                    l::anything,n::nonnegint)
   (int(f,x=-l..l)+
    sum(int(f*cos(k*Pi*x/l),x=-l..l)*cos(k*Pi*x/l),k=1..n)+
    sum(int(f*sin(k*Pi*x/l),x=-l..l)*sin(k*Pi*x/l),k=1..n))/l;
end proc:
```

Пример 1. Разложим на отрезке $[-\pi,\pi]$ в ряд Фурье функцию *f(x)=x*.

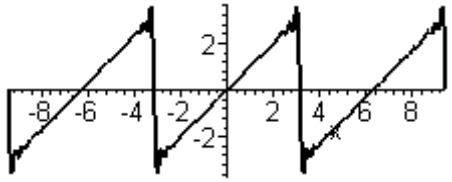
```
> ser1:=normal(FourierSeries(f,x,Pi,6));
```

$$2\sin(x)-\sin(2x)+\frac{2}{3}\sin(3x)-\frac{1}{2}\sin(4x)+\frac{2}{5}\sin(5x)-\frac{1}{3}\sin(6x)$$

```
> plot(ser1,x=-3*Pi..3*Pi,thickness=2,color=BLACK);
```



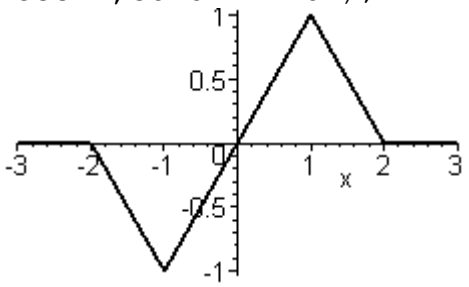
Поскольку функция нечетная, то в разложении нет косинусов. По графику ясно, что удерживаемых в ряде членов недостаточно для приемлемого приближения пилообразной функции (на отрезке $[-\pi,\pi]$ функция равна *x* и вне него периодически повторяет свои значения). Увеличение числа удерживаемых членов ряда позволяет повысить точность приближения.



Отметим, однако, что в окрестности точек разрыва ряд сильно осциллирует и его значения сильно отличаются от значений исходной функции.

Пример 2. Разложим с отрезка [-2,2] в ряд Фурье пилообразную функцию следующего вида

```
f:=-PR(x,-2,1)+2*PR(x,-1,2)-PR(x,1,1):
plot(f,x=-3..3,thickness=2,color=BLACK);
```



Ее уравнение и ряд имеют следующий вид

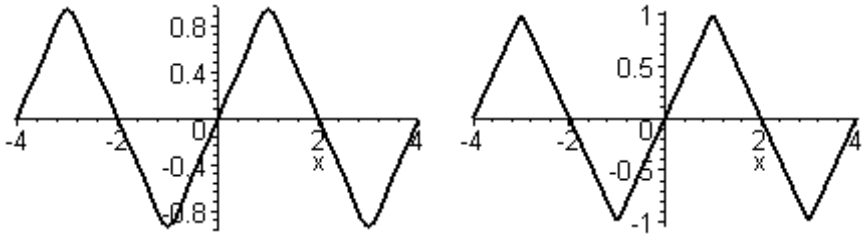
```
f;
ser2:=FourierSeries(f,x,2,6);
```

$$-\frac{1}{2}|x+2|+|x+1|-|x-1|+\frac{1}{2}|x-2|$$

$$ser2:=\frac{8\sin\left(\frac{\pi x}{2}\right)}{\pi^2}-\frac{8}{9}\frac{\sin\left(\frac{3\pi x}{2}\right)}{\pi^2}+\frac{8}{25}\frac{\sin\left(\frac{5\pi x}{2}\right)}{\pi^2}$$

Сама функция довольно хорошо приближается рядом Фурье. На следующем рисунке слева показан график ряда при 6 удерживаемых членах (предыдущая формула). На рисунке справа показан график ряда при 30-ти удерживаемых членах.

```
> plot(ser2,x=-4..4,thickness=2,color=BLACK);
```



Увеличения числа членов ряда делает приближение еще лучшим.

Если функцию, разлагаемую в ряд Фурье, надо определять в виде процедуры, то в нашу процедуру разложения в ряд надо внести поправки. В этом случае она принимает вид

```
FourierSeries1:=proc(f::procedure,x::name,
                     l::anything,n::nonnegint)
   (int(f(x),x=-l..l)+
    sum(int(f(x)*cos(k*Pi*x/l),x=-l..l)*cos(k*Pi*x/l),k=1..n)+
    sum(int(f(x)*sin(k*Pi*x/l),x=-
l..l)*sin(k*Pi*x/l),k=1..n))/l;
end proc:
```

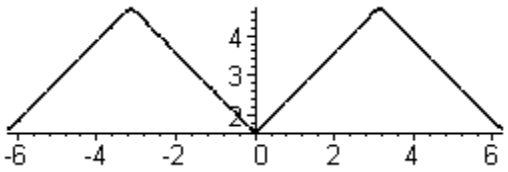
Здесь первый параметр должен быть типа процедуры и в теле вместо выражения *f* следует использовать *f(x)*.

Пример 3. Разложим в ряд Фурье с отрезка $[-\pi,\pi]$ функцию $f(x)=|x|$.

```
f:=proc(x) abs(x) end proc:
ser3:=collect(FourierSeries1(f,x,Pi,10),cos);
```

$$-\frac{4}{81}\frac{\cos(9x)}{\pi}-\frac{4}{49}\frac{\cos(7x)}{\pi}-\frac{4\cos(x)}{\pi}-\frac{4}{9}\frac{\cos(3x)}{\pi}-\frac{4}{25}\frac{\cos(5x)}{\pi}+\pi$$

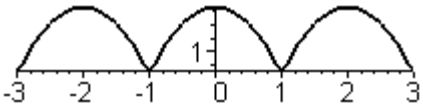
```
plot(ser3,x=-2*Pi..2*Pi,thickness=2,color=BLACK);
```



Как видим, уже при 10-ти удерживаемых членах график ряда хорошо аппроксимирует на отрезке $[-\pi,\pi]$ функцию $f(x)=|x|$. Отметим также, что в разложении в ряд четной функции $f(x)=|x|$ присутствуют только косинусы.

Пример 4. Разложим функцию $f(x)=1-x^2$ в ряд Фурье на отрезке [-1,1].

```
f:=x->1-x^2:
ser5:=collect(FourierSeries1(f,x,1,6),cos);
plot(ser5,x=-3..3,thickness=2,color=BLACK,scaling=CONSTRAINED);
```



Разложение четной функции и здесь дает ряд только по косинусам.

2. Решение задачи Коши разложением в степенной ряд.

Применение рядов для решения ДУ рассмотрим на примере нелинейного уравнения второго порядка вида $y'' = f(x, y, y')$ удовлетворяющего начальным условиям $y(x_0) = y_0$, $y'(x_0) = y_0'$.

Предположим, что решение поставленной задачи существует и будем его искать в форме ряда Тейлора функции *y(x)*

$$y(x) = y(x_0) + y'(x_0)(x - x_0) + \frac{1}{2} y''(x_0)(x - x_0)^2 + \ldots$$

Для построения решения нужно знать производные в точке x_0 порядка 1, 2, 3, …, но это можно сделать с помощью самого уравнения и его начальных условий. Действительно, из начальных условий мы сразу имеем значение функции и ее первой производной в точке x_0. Подставляя их в уравнение, сразу находим значение 2-й производной $y''(x_0) = f(x_0, y_0, y_0')$. Дифференцируя обе части исходного уравнения по аргументу *x*, получим выражение для 3-й производной

$$y''' = \frac{\partial f(x, y, y')}{\partial x} + \frac{\partial f(x, y, y')}{\partial y} y' + \frac{\partial f(x, y, y')}{\partial y'} y''$$

Подставляя в правую часть $x=x_0$ и значения предыдущих производных в этой точке, находим значение 3-й производной $y'''(x_0)$ в точке $x=x_0$. Дифференцируя последнее выражение еще раз, и выполняя подстановку известных производных, находим значение 4-й производной в точке $x=x_0$. Для определения производных функции *y(x)* в точке x_0 до требуемого порядка *n* дифференцирование и подстановку выполняем необходимое число раз. Затем найденные значения производных подставляем в представление решения в форме ряда Тейлора. Для тех значений *x* для которых этот ряд сходится, он будет представлять искомое решение задачи Коши.

Для решения задачи Коши $y'' = f(x, y, y')$, $y(x_0) = y_0$, $y'(x_0) = y_0'$ используем следующую процедуру (см. [1], стр. 450).

```
DiffSer:=proc(f::anything,x::name,y::name,y1::name,\
              x0::numeric,y0::numeric,y01::numeric,n::integer)
   local p,der,i,j,s:
   m0||1:=y01:
   m0||2:=subs(x=x0,y=y0,y1=y01,f);
   der:=subs(y1=m1,f);
   p:=y0+m0||1*(x-x0)+m0||2*(x-x0)^2/2:
   for i from 3 to n do
      s:=0:
      for j from 1 to i-2 do
         s:=s+diff(der,m||j)*m||(j+1);
      end do:
      der:=diff(der,x)+diff(der,y)*m||1+s;
      m0||i:=eval(der,{x=x0,y=y0,seq(m||k=m0||k,k=1..i-1)});
      p:=p+(m0||i)*(x-x0)^i/i!;
   end do;
end proc:
```

Аргументами процедуры являются выражение для правой части ДУ (*f*), имена независимой переменной (*x*), неизвестной функции (*y*) и ее производной (*y1*), а также параметры *x0, y0, y01*, определяющие условия задачи Коши. Порядок усечения ряда *n*, задаваемый последним параметром, указывает, что последний член ряда будет иметь вид $a_n(x-x_0)^n$.

Операторы расположенные до первого цикла *for* формируют первые 3 члена ряда. В цикле *for* вычисляются оставшиеся производные и их значения в точке $x=x_0$, а также формируется ряд. В процедуре для всех появляющихся при последовательном дифференцировании производных решения используются обозначения *mN*, где *N* – порядок производной, а для их значений в точке x_0 используются переменные *m0N*.

Пример 1. Решить следующую задачу Коши: $\frac{\partial^2}{\partial x^2}y=-y\,x^2$, $y(0)=1$, $\frac{d}{dx}y(0)=0$.

```
s:=DiffSer(-y*x^2,x,y,y1,0,1,0,20);
```

$$s := 1-\frac{1}{12}x^4+\frac{1}{672}x^8-\frac{1}{88704}x^{12}+\frac{1}{21288960}x^{16}-\frac{1}{8089804800}x^{20}$$

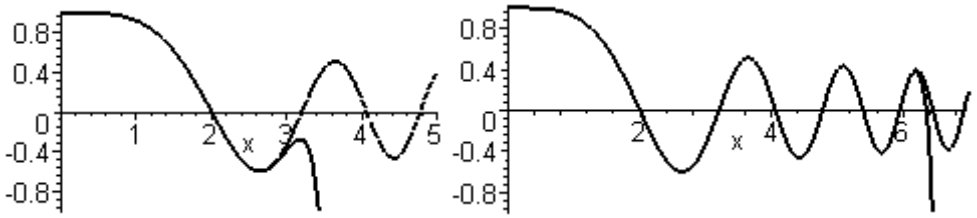
Выбранная задача имеет точное решение.

```
dsolve({diff(y(x),x$2)=-y(x)*x^2,y(0)=1,D(y)(0)=0},y(x));
s1:=rhs(%);
```

$$y(x)=\frac{1}{2}\Gamma\left(\frac{3}{4}\right)\sqrt{x}\ \mathrm{BesselJ}\left(\frac{1}{4},\frac{x^2}{2}\right)-\frac{1}{2}\Gamma\left(\frac{3}{4}\right)\sqrt{x}\ \mathrm{BesselY}\left(\frac{1}{4},\frac{x^2}{2}\right)$$

Для сравнения построим графики точного и приближенного решений. На следующем графике слева показано точное решение и приближенное до 20 – й степени, а справа до 100-й степени..

```
plot([s,s1],x=0..5,-
1..1,color=BLACK,thickness=2,linestyle=[1,4]);
```



Как видим, увеличение числа членов ряда расширяет интервал аппроксимации решения.

Пример 2. Исследуем колебания плоского математического маятника – точка массы *m* подвешена на конце нити длины *l* и находится в однородном поле тяжести. Колебания маятника описываются уравнением $ml^2\varphi''+mgl\sin\varphi=0$, где $\varphi(t)$ - угол отклонения маятника от положения равновесия. Перепишем это уравнение в виде $\varphi''_{tt}=-\frac{g}{l}\sin\varphi$. Это вид ДУ, который можно решать с помощью процедуры *DiffSer*.

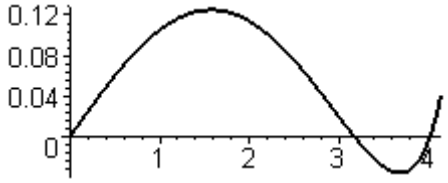
Для примера выберем *m=1, g=1*, а начальные условия возьмем следующего вида: $\varphi(0)=0, \varphi'(0)=\frac{1}{8}$ т.е. задана начальная скорость и в нулевой момент времени маятник находится в положении равновесия.

Решим эту задачу Коши с помощью процедуры `DiffSer`.

```
> s:=DiffSer(-sin(y),x,y,y1,0,0,1/8,14);
```

$$s := \frac{1}{8}x - \frac{1}{48}x^3 + \frac{13}{12288}x^5 - \frac{4801}{165150720}x^7 + \frac{19531}{21743271936}x^9$$
$$- \frac{38052343}{765363172147200}x^{11} + \frac{4421087563}{1528277182143528960}x^{13}$$

```
> plot(s,x=0..Pi+1,color=BLACK,thickness=2,
linestyle=[1,4],numpoints=1000);
```



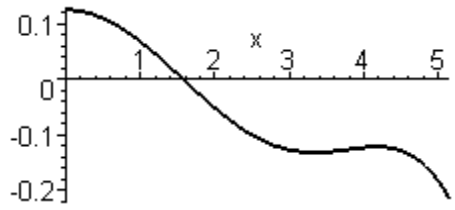
Если начальная скорость не слишком велика, то колебания должны быть периодическими. Мы видим, что наше решение через некоторый промежуток времени принимает нулевое значение, а значит, маятник возвращается в положение равновесия. Вероятно о дальнейшем поведении решения наш ряд ничего не говорит, поскольку начинает сказываться погрешность приближения.

Пример 3. Решим то же ДУ с другими начальными условиями $\varphi(0)=\frac{1}{8}, \varphi'(0)=0$, т.е. задана начальное отклонение относительно положения равновесия, а начальная скорость равна 0.

```
> s:=DiffSer(-sin(y),x,y,y1,0,1/8,0,9):
evalf[5](s);
```

$$0.12500 - 0.062335\,x^2 + 0.0051542\,x^4 - 0.00016239\,x^6 + 0.14468\,10^{-5}\,x^8$$

```
> plot(s,x=0..Pi+2,color=BLACK,thickness=2,
linestyle=[1,4],numpoints=1000);
```



Из графика видно, что материальная точка возвращается в положение равновесия, затем удаляется от него в противоположном направлении. Для больших моментов времени полученное решение применять нельзя.

Построить решение в форме ряда можно и другим способом. В этом способе решение *y(x)* представляется в виде степенного ряда с неопределенными коэффициентами

$$y(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)^2 + a_3(x-x_0)^3 + \ldots.$$

Этот ряд подставляется в левую и правую части уравнения $y'' = f(x, y, y')$, причем все функции правой части также раскладываются в степенные ряды. Далее выполняются необходимые вычисления с рядами правой части для представления ее в виде одного степенного ряда. Приравнивая в левой и правой части коэффициенты при одинаковых степенях $(x\text{-}x_0)^n$, получим систему линейных уравнений относительно неизвестных коэффициентов ряда a_n.

В работе [1] стр.453 приведена процедура решения задачи Коши для ДУ 2-го порядка, имеющей вид $y'' = f(x, y, y')$, $y(x_0) = y_0$, $y'(x_0) = y_0'$, реализующая описанный алгоритм. Вот ее код.

```
intDiff:=proc(f::anything,x::name,y::name,y1::name,\
              x0::numeric,y0::numeric,y01::numeric,n::integer)
   local p,pL,i,s,a,result:
   p:=y0+y01*(x-x0):
   result:=p+sum(a[i]*(x-x0)^i,i=2..n);
   p:=p+sum(a[i]*(x-x0)^i,i=2..n+2);
   pL:=series(subs(y=p,y1=diff(p,x),f),x=x0,n+1);
   for i from 0 to n do
      eq||i:=coeff(diff(p,x$2)-convert(pL,polynom),x,i)=0;
   end do:
   s:=solve({seq(eq||i,i=0..n)},{seq(a[i],i=2..n+2)});
   assign(s);
   eval(result);
 end proc:
```

Аргументы этой процедуры имеют тот же смысл, что и аргументы предыдущей процедуры. В теле процедуры введен промежуточный ряд, имеющий слагаемых на 2 больше, чем требуемое число членов, т.к. его приходится дважды дифференцировать. Отметим, что эта процедура работает быстрее предыдущей.

Пример 4. Решим задачу Коши $xy'' + y' + xy = 0$, $y(0) = 1, y'(0) = 0$. Для решения это уравнение с помощью разработанной процедуры, преобразуем его к виду $y'' = -\frac{y'}{x} - y$. Отметим, что функция правой части при $x=0$ не определена. Поэтому применение процедуры реализующей первый способ построения решения в виде ряда не представляется возможным, т.к. ее алгоритм предполагает вычисление правой части в точке $x=0$. Но в методе поиска неопределенных коэффициентов, реализованном в последней процедуре, значение правой части в этой точке не вычисляется.

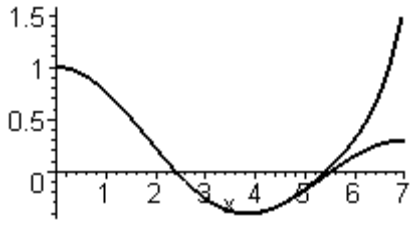
```
> s:=intDiff(-y1/x-y,x,y,y1,0,1,0,12);
```

$$1 - \frac{1}{4}x^2 + \frac{1}{64}x^4 - \frac{1}{2304}x^6 + \frac{1}{147456}x^8 - \frac{1}{14745600}x^{10} + \frac{1}{2123366400}x^{12}$$

```
> s1:=dsolve({x*diff(y(x),x$2)+diff(y(x),x)+x*y(x)=0,
y(0)=1,D(y)(0)=0},y(x));
```

$$s1 := \mathrm{y}(x) = \mathrm{BesselJ}(0, x)$$

```
> plot([s,rhs(s1)],x=0..7,color=BLACK,thickness=2,
linestyle=[1,4],numpoints=1000);
```



На графике приведено точное решение задачи и приближенное в виде ряда.

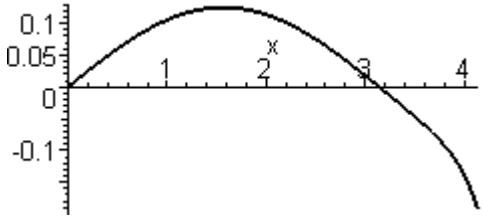
Расхождение решения в форме ряда с точным решением может происходить по двум причинам. Первая состоит в том, что мы не можем выбрать достаточное количество членов ряда для получения удовлетворительной точности на широком интервале изменения аргумента. Это ограничение связано с ограничением по времени выполнения программы и по объему используемой ею памяти. Вторая причина состоит в том, что большинство рядов имеют конечный радиус сходимости и использование ряда для приближения решения на более широком интервале принципиально невозможно. Выход из создавшегося положения состоит в разбиении интервала поиска решения на отрезки такой длины, в пределах которых степенные ряды будут сходиться и давать приемлемое приближение решения. Вначале строится решение $s_1(x)$ в форме ряда Тейлора на отрезке $[x_0,x_1]$, где мы знаем, что ряд хорошо аппроксимирует решение и сходится. Затем находим значение этого ряда и его производной в точке x_1 и используем эти значения в качестве начальных значений для построения нового ряда $s_2(x)$ разложения решения в окрестности точки x_1 по степеням $(x-x_1)^n$. Построенный ряд используем для определения начальных условий Коши в точке $x=x_2$ и определяем третий ряд аппроксимирующий решение на следующем отрезка поиска решения и т.д.

Для иллюстрации этого подхода рассмотрим ДУ описывающее колебания математического маятника.

Пример 5. Для нашей задачи возьмем начальные условия вида $y(0)=\frac{1}{8}, y'(0)=0$ и для построения решения в форме ряда используем процедуру *intDiff*.

```
s1:=intDiff(-sin(y),x,'y','y1',0,0,1/8,20):
evalf[3](s1);
plot(s1,x=0..Pi+1,color=BLACK,thickness=2,
linestyle=[1,4],numpoints=1000);
```

$$0.125\,x - 0.0208\,x^3 + 0.00106\,x^5 - 0.0000291\,x^7 + 0.898\,10^{-6}\,x^9$$
$$- 0.497\,10^{-7}\,x^{11} + 0.289\,10^{-8}\,x^{13} - 0.146\,10^{-9}\,x^{15} + 0.697\,10^{-11}\,x^{17}$$
$$- 0.351\,10^{-12}\,x^{19}$$



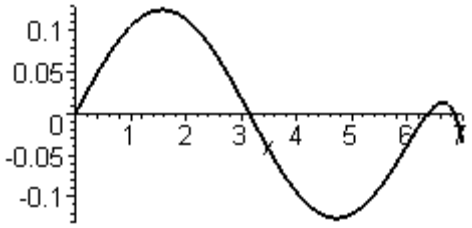
Полученный ряд дает удовлетворительное решение на отрезке [0,3]. Используем *x=3* в качестве начальной точки для решения новой задачи Коши. Подставим *x=3* в найденный ряд и его производную по *x* для получения новых значений $y(0)=y_0, y'(0)=y_0'$. Опять решаем задачу Коши используя процедуру *intDiff*.

```
x0:=3:
y0:=subs(x=x0,s1):
y1:=subs(x=x0,diff(s1,x$1)):
s2:=intDiff(-sin(y),x,'y','y1',x0,y0,y1,14):
evalf[3](s2);
```

$$0.392 - 0.125\,x - 0.00895\,(x-3.)^2 + 0.0208\,(x-3.)^3 + 0.000758\,(x-3.)^4$$
$$- 0.00106\,(x-3.)^5 - 0.0000291\,(x-3.)^6 + 0.0000287\,(x-3.)^7$$
$$+ 0.115\,10^{-5}\,(x-3.)^8 - 0.857\,10^{-6}\,(x-3.)^9 - 0.771\,10^{-7}\,(x-3.)^{10}$$
$$+ 0.461\,10^{-7}\,(x-3.)^{11} + 0.530\,10^{-8}\,(x-3.)^{12} - 0.265\,10^{-8}\,(x-3.)^{13}$$
$$- 0.306\,10^{-9}\,(x-3.)^{14}$$

Из двух рядов создаем кусочно-непрерывную функцию и строим ее график.

```
Sser:=x->piecewise(x<=x0,s1,x>x0,s2):
plot(Sser(x),x=0..4,color=BLACK,thickness=2,
linestyle=[1,4],numpoints=1000);
```



Полученная функция уже хорошо приближает решение на отрезке [0,6], где она похожа на синусоиду. Мы это знаем т.к. решение должно быть нечетным и периодическим (задача о колебании маятника). Интересно отметить, что построенная функция непрерывна вместе со своей 1-й производной. Действительно, 1-я производная первого ряда в точке *x=3* использована в качестве первой производной второго ряда как начальное значение. Вторая производная в точке стыковки для обеих функций, как правило, будет различной. Для первого ряда она вычисляется подстановкой $x=x_1$ во 2-ю производную первого ряда, а для второй вычисляется по формуле $y''(x_1)=f(x_1,y_1,y_1')$.

Для реализации метода решения задачи Коши ДУ 2-го порядка разложением в степенной ряд с разбиением на отрезки мы создали специальную процедуру.

```
StepSeries:=proc(f::anything,x::name,y::name,y1::name,\
                 x0::numeric, y0::numeric, y01::numeric,\
                 n::integer,xend::numeric, dlt::numeric)
   local i,nn,DiffInner;
   # Внутренняя процедура разложения решения в ряд
   # в окрестности 0
   DiffInner:=proc(f::anything,x::name,y::name,y1::name,\
              y0::Or(numeric,realcons),\
              y01::Or(numeric,realcons),n::integer)
```

```
      local p,pL,i,s,a,result:
      p:=y0+y01*x:
      result:=p+sum(a[i]*x^i,i=2..n);
      p:=p+sum(a[i]*x^i,i=2..n+2);
      pL:=series(subs(y=p,y1=diff(p,x),f),x=0,n+1);
      for i from 0 to n do
         eq||i:=coeff(diff(p,x$2)-convert(pL,polynom),x,i)=0;
      end do:
      s:=solve({seq(eq||i,i=0..n)},{seq(a[i],i=2..n+2)});
      assign(s);
      eval(result);
   end proc:
   nn:=floor((xend-x0)/dlt): # число точек стыковки
   # Второе число X0i,Y0i,Y1i указывает номер шага i
   X0||0:=x0: Y0||0:=y0: Y1||0:=y01:
   s||1:=DiffInner(f,x,y,y1,Y00,Y10,n):
   for i from 1 to nn-1 do
      X0||i:=x0+dlt*i:
      Y0||i:=evalf(subs(x=dlt,s||i));
      Y1||i:=evalf(subs(x=dlt,diff(s||i,x$1)));
      s||(i+1):=DiffInner(f,x,y,y1,Y0||i,Y1||i,n):
   end do:
   piecewise(seq(op([x<=X0||i,subs(x=x-X0||(i-
1),s||i)]),i=1..nn-1),\
                     subs(x=x-X0||(nn-1),s||nn));
end proc:
```

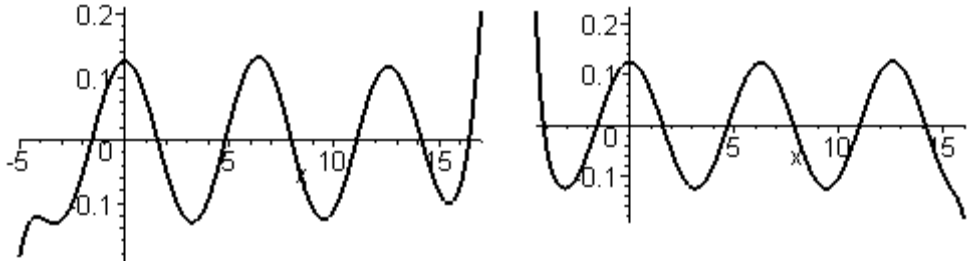
Аргументами процедуры являются выражение для правой части ДУ (*f*), имена независимой переменной (*x*), неизвестной функции (*y*) и ее производной (*y1*), а также параметры *x0, y0, y01*, определяющие условия задачи Коши. Порядок усечения рядов задаваемый параметром *n*, указывает, что их последние члены будут иметь вид $a_n(x-x_0)^n$. Эти аргументы в точности те же, что и у процедуры *DiffSer*. Последние два параметра определяют конец интервала поиска решения *xend* (оно ищется на интервале [*x0, xend*]) и длину отрезков на которые разбивается этот интервал – *dlt*. Это означает, что будут построено $N = \left[\frac{xend - x0}{dlt}\right]$ степенных рядов, которые будут стыковаться в точках $x_k = x0 + k \cdot dlt, k = 1,2,\ldots,N$. Внутренняя процедура *DiffInner* идентична процедуре *DiffSer* и всегда вызывается со значением *x0=0*, поэтому *DiffInner* не имеет этого аргумента. Результатом работы процедуры является *piecewise* функция, составленная из построенных рядов для которых выполняется преобразование к общей системе координат (координатная система первого отрезка).

Работу процедуры проиллюстрируем на примере уравнения колебания математического маятника. Мы не будем приводить формулы полученной *piecewise* функции из-за их длины.

Пример 6. Для маятника задано начальное отклонение, а начальная скорость равна 0.

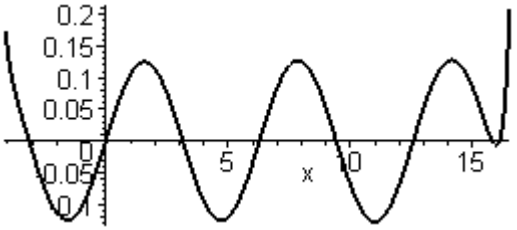
```
st:=time():
x0:=0: y0:=1/8: y1:=0:
r1:=StepSeries(-sin(y),x,'y','y1',x0,y0,y1,8,15,3):
time()-st;
plot(r1,x=-5..11.5,color=BLACK,thickness=2,numpoints=500);
```

На следующем рисунке слева приведен график решения, составленного из многочленов до 8-й степени, а справа –14-й. Оба решения составлены из 5-ти рядов построенных на отрезках [*0,3*], [*3,6*], [*6,9*], [*9,12*], [*12,15*]. На левом рисунке заметно некоторое отличие 3-й волны графика от 2-х предыдущих, поэтому для улучшения решения пришлось повысить степень многочленов до 14-ти.



Выбрав другое начальное условие (начальное отклонение 0, начальная скорость не 0) получим следующее решение

```
x0:=0: y0:=0: y1:=1/8:
r1:=StepSeries(-sin(y),x,'y','y1',x0,y0,y1,20,15,3):
plot(r1,x=-4.1..16.5,color=BLACK,thickness=2,numpoints=500);
```

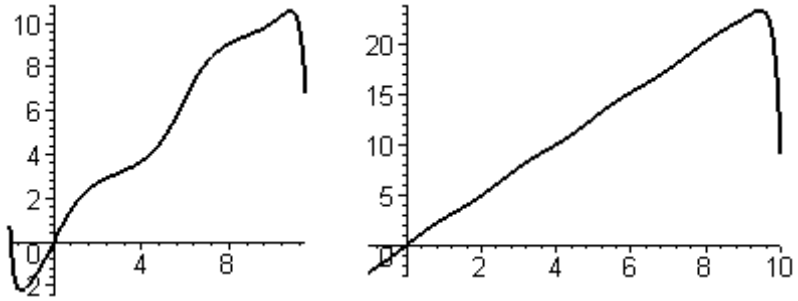


Здесь для построения приемлемого решения пришлось строить многочлены 20-й степени. Полученное решение можно применять на интервале несколько более широком, чем отрезок заданный в аргументе процедуры, поскольку первый ряд применим левее начальной точки *x0*, а правый может давать приближение на отрезке несколько более широком, чем *dlt*. Если мы правильно выбрали параметр *dlt*, то решение в виде составного ряда будет всегда приемлемым на отрезке [*x0-dlt, xend*].

Если начальная скорость достаточно высока (в нашем примере больше 2), то движение маятника перестает быть периодическим (маятник доходит до верхней точки и переваливает через нее). В этом случае угол отклонения маятника возрастает монотонно.

```
st:=time():
x0:=0: y0:=0: y1:=2.05:
r1:=StepSeries(-sin(y),x,'y','y1',x0,y0,y1,12,9,1):
time()-st;
plot(r1,x=-1..10,color=BLACK,thickness=2,numpoints=500);
```

На следующем рисунке слева показан график решения при нулевом начальном смещении и начальной скорости *y1=2.05*, а на правом – при начальной скорости равной *y1=3*. Если начальная скорость достаточно велика, то кривизны графика решения почти незаметно. Легко видеть, что вне отрезка [0,9] построенные решения неприменимы.



Отметим, что для построения последних решений длину отрезков, на которых построены ряды, пришлось брать равной единице, т.е. наши решения составлены из 9 многочленов 12-й степени. Если бы мы не использовали процедуру сшивания рядов, то не смогли бы построить хорошее решение на таком широком интервале изменения аргумента. Отметим также, что периодическое решение, соответствующее колебательному процессу достаточно построить на отрезке равном периоду колебаний, а на более широкий интервал времени решение мы можем продолжить с помощью нашей методики периодического продолжения функций.

3. Метод Галеркина решения краевых задач ОДУ

Общая краевая задача для ОДУ 2-го порядка формулируется следующим образом - требуется найти решение ДУ вида

$$Lu \equiv u'' + p(x)u' + q(x)u = f(x), \quad a \le x \le b,$$

удовлетворяющее краевым условиям

$$l_0 u \equiv \alpha_0 u(a) + \beta_0 u'(a) = \gamma_0,$$
$$l_1 u \equiv \alpha_1 u(a) + \beta_1 u'(a) = \gamma_1.$$

где *p, q* и *f* являются непрерывными функциями переменой *x* на интервале [*a,b*], а α_i, β_i, γ_i – некоторые числа.

Для нахождения приближенного решения краевой задачи на отрезке [*a,b*] задают некоторую линейно независимую систему дважды непрерывно дифференцируемых функций $\varphi_0, \varphi_1, ..., \varphi_n, ...$ таких, что функция φ_0 удовлетворяет заданным краевым условиям $l_0\varphi_0 = \gamma_0$, $l_1\varphi_0 = \gamma_1$, а остальные функции системы φ_i при *i=1,2,…* удовлетворяют однородным краевым условиям $l_0\varphi_i = 0$, $l_1\varphi_i = 0$. Эта система функций называется базисной.

Приближенное решение краевой задачи ищется в виде

$$y_n(x) = \varphi_0(x) + a_1\varphi_1(x) + ... + a_n\varphi_n(x)$$

с неизвестными коэффициентами a_i, $i=1,2,\ldots,n.$, которые определяются из выполнения некоторого критерия. Отметим, что в силу выбора базисных функций функция $y_n(x)$ будет точно удовлетворять краевым условиям.

В методе Галеркина в качестве критерия для определения неопределенных коэффициентов выбирается требование ортогональности базисных функций $\varphi_1, \varphi_2, .., \varphi_n$ к функции невязки $\Psi(x, a_1, ..., a_n)$ являющейся разностью левой и правой части ОДУ при подстановке в него приближенного решения $y_n(x)$.

$$\Psi(x, a_1, ..., a_n) = Ly_n - f(x) = L\varphi_0(x) - f(x) + \sum_{k=1}^{n} a_k L\varphi_k(x)$$

Под ортогональностью понимается равенство нулю скалярного произведения невязки и базисных функций $\int_a^b \Psi(x, a_1, ..., a_n)\varphi_i(x)dx = 0$. Это требование приводит к линейной системе алгебраических уравнений для определения неизвестных коэффициентов a_i приближенного решения $y_n(x)$.

$$a_1(L\varphi_1, \varphi_1) + ... + a_n(L\varphi_n, \varphi_1) = (f - L\varphi_0, \varphi_1)$$
$$a_1(L\varphi_1, \varphi_2) + ... + a_n(L\varphi_n, \varphi_2) = (f - L\varphi_0, \varphi_2)$$
$$\ldots\ldots\ldots\ldots \quad \ldots\ldots\ldots\ldots \quad \ldots\ldots\ldots\ldots$$
$$a_1(L\varphi_1, \varphi_n) + ... + a_n(L\varphi_n, \varphi_n) = (f - L\varphi_0, \varphi_n)$$

Решение этой системы всегда существует и единственно в силу линейной независимости базисных функций.

Вот процедура построения приближенного решения краевой задачи ОДУ второго порядка, реализующая метод Галеркина (см. [1] стр.474).

```
Galerkin:=proc(L::procedure,f::algebraic,phi::procedure,
               x::name,a::numeric,b::numeric,n::numeric)
   local y,i,eq,A:
   y:=phi(0,x)+sum(A[i]*phi(i,x),i=1..n):
   for i to n do
   eq||i:=sum(A[k]*int(L(phi(k,x),x)*phi(i,x),x=a..b),k=1..n)=\
        int((f-L(phi(0,x),x))*phi(i,x),x=a..b);
   end do:
   eq:=solve({seq(eq||i,i=1..n)},{seq(A[i],i=1..n)});
   assign(eq);
   simplify(eval(y));
end proc:
```

Параметрами этой процедуры является дифференциальный оператор *L*, правая часть уравнения *f*, процедура вычисления базисных функций *phi*. Параметр *x* определяет имя независимой переменной, *a,b* – координаты граничных точек, *n* – число удерживаемых членов в приближенном решении. Дифференциальный оператор *L* передается в виде процедуры с двумя параметрами – алгебраическим выражением и именем независимой переменной, по которой выполняется дифференцирование. Параметр-процедура *phi* вычисляет *n*-ю базовую функцию φ_n, и ее параметрами являются номер базовой функции и имя независимой переменной.

Пример1. Решим методом Галеркина краевую задачу

$$Lu \equiv u'' + u' = -x,\ \ 0 \le x \le 1,\ u(0) = 0,\ u(1) = 0$$

Выбираем базисные функции $\varphi_0(x) = 0,\ \varphi_i(x) = x^i(1-x),\ i = 1,2,\ldots$. Функция $\varphi_0(x)$ удовлетворяет граничным условиям краевой задачи, а функции $\varphi_i(x)$ однородным граничным условиям.

Создадим необходимые процедуры вычисления дифференциального оператора, базовых функций и выражения правой части.

```
> L1:=proc(y,x)
    diff(y,x$2)+y;
  end proc:
> f:=-x:
> phi:=proc(n,x)
    if n=0 then 0 else x^n*(1-x) end if;
  end proc:
```

Построим приближенное решение

```
> res1:=Galerkin(L1,f,phi,x,0,1,1);
  res2:=Galerkin(L1,f,phi,x,0,1,2);
  res3:=Galerkin(L1,f,phi,x,0,1,3);
```

$$res1 := -\frac{5\,x\,(-1+x)}{18}$$

$$res2 := \frac{71}{369}x - \frac{8}{369}x^2 - \frac{7}{41}x^3$$

$$res3 := \frac{13811}{73554}x + \frac{469}{73554}x^2 - \frac{2667}{12259}x^3 + \frac{7}{299}x^4$$

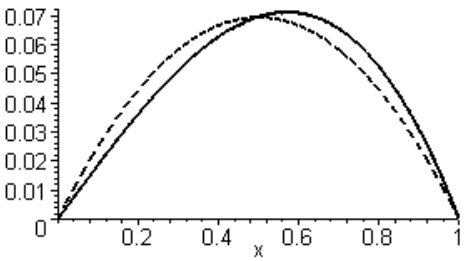
Найдем точное решение этой краевой задачи

```
> exact:=dsolve({diff(y(x),x$2)+y(x)=-x,y(0)=0,y(1)=0},y(x));
```

$$exact := \mathrm{y}(x) = \frac{\sin(x)}{\sin(1)} - x$$

Для сравнения построим графики полученных решений

```
> plot([res1,res2,res3,rhs(exact)],x=0..1,color=black,
thickness=2,linestyle=[4,7,1]);
```



Видно, что приближенные решения, в которых указаны две базовые функции (и три) на графике сливаются с точным.

Пример 2. Решим то же ДУ с другими краевыми условиями $u(0) = 1,\ u(1) = 0$. Для этого надо выбрать другую функцию $\varphi_0(x) = 1 - x$ и, следовательно, изменить процедуру *phi*.

```
> phi:=proc(n,x)
    x^n*(1-x);
```

```
 end proc:
> res1:=Galerkin(L1,f,phi,x,0,1,1);
 res2:=Galerkin(L1,f,phi,x,0,1,2);
 res3:=Galerkin(L1,f,phi,x,0,1,3);
 exact:=dsolve({diff(y(x),x$2)+y(x)=-x,y(0)=1,y(1)=0},y(x));
```

$$res1 := 1 - \frac{4}{9}x - \frac{5}{9}x^2$$

$$res2 := 1 - \frac{4}{9}x - \frac{5}{9}x^2$$

$$res3 := 1 - \frac{407}{897}x - \frac{448}{897}x^2 - \frac{28}{299}x^3 + \frac{14}{299}x^4$$

$$exact := y(x) = -\frac{\sin(x)\,(\cos(1)-1)}{\sin(1)} + \cos(x) - x$$

Построим графики полученных решений

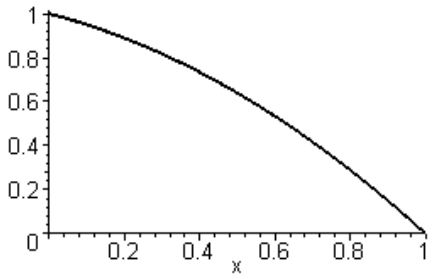


График уже первого приближения сливается с графиком точного решения.

Пример 3. Метод Галеркина имеет смысл применять для задач, не имеющих точного решения. Пусть требуется решить следующую краевую задачу

$$Lu \equiv u'' + (1+x^2)u' = -1,\ -1 \le x \le 1,\ u(-1) = 0,\ u(1) = 0$$

Выбираем базисные функции $\varphi_0(x) = 0,\ \varphi_i(x) = x^{2i-2}(1-x^2),\ i = 1,2,\ldots$. Функция $\varphi_0(x)$ удовлетворяет граничным условиям краевой задачи, а функции $\varphi_i(x)$ однородным граничным условиям.

Создадим необходимые процедуры вычисления дифференциального оператора, базовых функций и выражения правой части

```
> L1:=proc(y,x)
    diff(y,x$2)+(1+x^2)*y;
 end proc:
> f:=-1:
> phi:=proc(n,x)
    if n=0 then 0 else x^(2*n-2)*(1-x^2) end if:
 end proc:
```

Решение

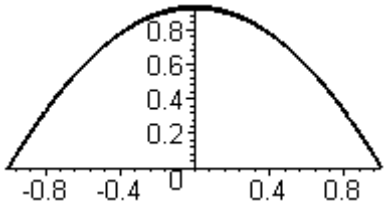
Находим приближенные решения при *n=1, 2, 3.*

```
> res1:=Galerkin(L1,f,phi,x,-1,1,1);
 res2:=Galerkin(L1,f,phi,x,-1,1,2);
 res3:=Galerkin(L1,f,phi,x,-1,1,3);
> plot([res1,res2,res3],x=-1..1,color=black,
thickness=2,linestyle=[4,7,1]);
```

$$res1 := \frac{35}{38} - \frac{35\,x^2}{38}$$

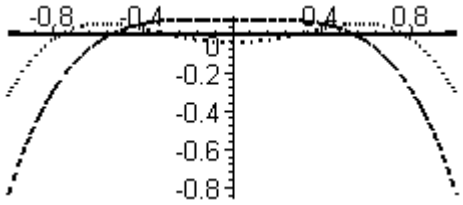
$$res2 := \frac{3969}{4252} - \frac{1050}{1063}x^2 + \frac{231}{4252}x^4$$

$$res3 := \frac{5148231}{5523436} - \frac{62073}{64226}x^2 + \frac{26169}{5523436}x^4 + \frac{81939}{2761718}x^6$$

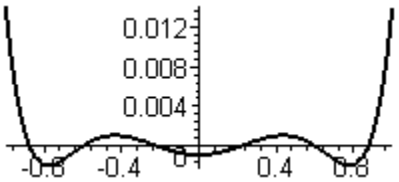


Поскольку точного решения не существует, то для оценки погрешности приближенных решений мы построим график невязок.

```
> plot([L1(res1,x)-f,L1(res2,x)-f,L1(res3,x)-f],x=-
1..1,color=black,thickness=2,linestyle=[4,7,1]);
```



Невязка третьего приближения отображена сплошной линией и практически полностью сливается с осью *x*.Для оценки точности приведем ее график отдельно.



Максимальная невязка для третьего приближения составляет 0.014

Метод Галеркина весьма эффективен для решения краевых задач ОДУ. Однако при его применении возникает трудность выбора линейно независимой системы базисных функций, удовлетворяющей однородным краевым условиям. В связи с этим отметим, что получивший в последнее время широкое распространение метод конечных элементов является дальнейшим развитием метода Галеркина. Фактически МКЭ имеет специфические особенности выбора базисной системы функций, а в остальном совпадает с методом Галеркина.

Литература.

1. А.. Матросов. “Maple 6. Решение задач высшей математики и механики”, БХВ, Петербург, 2001г.