Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Прикладные задачи математического анализа

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему

АНИМАЦИОННЫЕ ВОЗМОЖНОСТИ MAPLE ДЛЯ ВИЗУАЛИЗАЦИИ РЕШЕНИЙ

БГУИР КП 1-40 04 01

Студент: гр.153502 Богданов А.С.

Руководитель: канд. ф.-м. н., доцент Калугина М.А.

Минск 2022

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Информационные технологии плотно вошли в нашу жизнь и сейчас активно применяются во многих науках, в том числе и в математике. Автоматизация процессов, новые идеи и алгоритмы позволяют математикам быстрее решать рутинные задачи, использовать вычислительные мощности для приближённых вычислений и, наконец, сверять своё решение с компьютером. В этих целях используются системы компьютерной алгебры (СКА). Одним из представителей таких систем является Maple.

Maple — это математическое программное обеспечение, которое сочетает в себе самый мощный в мире математический движок с интерфейсом, который позволяет чрезвычайно легко анализировать, исследовать, визуализировать и решать математические задачи. [1]

Проблема, которую было решено исследовать — использование Maple для визуализации решений, преимущественно в образовательных целях, ведь визуально материал воспринимается намного понятнее. Зачастую сложные математические формулы, явления, теоремы становятся более понятны и логичны, если их визуализировать. Но статическое изображение не может дать чёткое представление о процессе, который происходит. И тут на ум приходит логичная мысль: нужно анимировать, строить несколько состояний и рассматривать их последовательно, чтобы видеть все изменения.

Цель работы — изучить основные методы построения анимаций в Maple, а также инструменты (функции, объекты, процедуры), которые для этого используются.

Задача — рассмотреть применение анимаций для визуализаций математических объектов, теорем и некоторых математических и физических задач.

В первом разделе будет рассмотрена анимация с помощью встроенных функций *Maple*. Данный тип анимаций самый простой, однако он позволяет решать базовые задачи.

Во втором разделе будут рассмотрены покадрово построенные анимации. Данные анимации позволяют строить анимации кадр за кадром, что позволяет решать задачи любого типа.

В третьем разделе будет рассмотрено применение анимационных возможностей *Maple* в физических задачах.

1 ПРОСТЫЕ АНИМАЦИИ

СКА Maple имеет три встроенных процедуры для анимации функций одной или двух переменных — *animate*, *animatecurve*, *animate3d*. Стоит заметить, что функция *animate3d* является устаревшей, и в новых версиях Maple следует использовать просто *animate* вместо неё. Данные процедуры находятся в пакете *plots* и для их использования этот пакет нужно предварительно подключить.

Управление анимациями ведётся с помощью контекстного меню, которое появляется при нажатии на анимацию (Рис. 1).

Рисунок 1 — Контекстное меню

1.1 Анимации функций одной переменной

Функция *animate* пакета позволяет создавать анимированные изображения. Суть анимации заключается в построении серии изображений, причем каждое изображение (кадр) связано с изменяемой во времени переменной. Функция по стандарту создаёт 25 кадров. Первым аргументом *animate* принимает функцию двух переменных. Первая переменная, скажем, x – независимая переменная функции, график которой необходимо построить. Вторая (назовём её t) – так называемая переменная кадра. Это значение, которое должно изменяться от кадра к кадру. Синтаксис процедуры [1]:

$$animate(f(x,t), x = a..b, t = p..q, [options])$$

Некоторые из доступных *options [1]:*

- *labels* — задает метки для осей
- *view* — задаёт границы области координат, отображаемых в графике
- *numpoints* — задаёт минимальное число точек, по которым строится график. Этот параметр может быть полезен для лучшей «гладкости» графика во время анимаций.
- *color* — задаёт цвет графика
- *frames* — задаёт количество кадров, рисуемое функцией

Построим простую анимацию движения поворота прямой:

```
> restart; with(plots) :
  animate(tan(t)·x, x = -2 ..2, t = 0 ..4 Pi, frames = 60, scaling = constrained, view = [-2 ..2, -2 ..2])
```

На рис. 2 можно видеть результат выполнения функции.

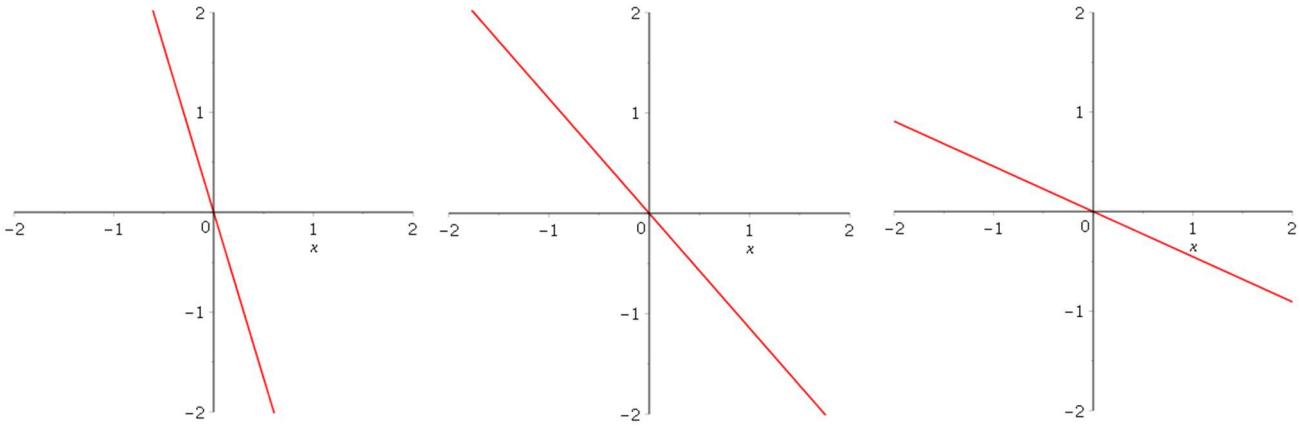

Рисунок 2 — Несколько кадров получившейся анимации.

Также присутствует возможность анимировать функции, заданные параметрически. Приведём в пример функцию, растягивающую эллипс в круг.

```
> restart; with(plots) :
  animate([t·cos(x), sin(x), x = 0 ..2Pi], t = 0.2 ..1, frames = 30, scaling = constrained, view = [-2 ..2, -2 ..2], )
```

Результат работы показан на рис. 3.

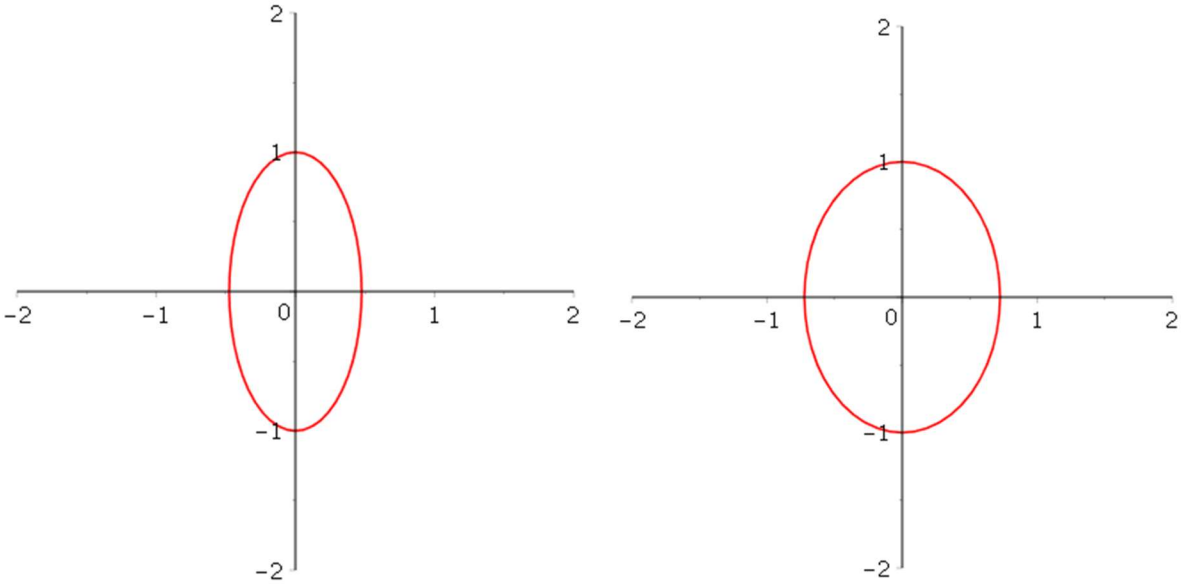

Рисунок 3 — Кадры анимации функции, заданной параметрически

1.2 Анимация с помощью процедур

СКА Maple предоставляет огромный функционал для написания собственных процедур. Это облегчает чтение инструкций и позволяет заново использовать написанную процедуру. Приведём пример использования такой процедуры для анимаций.

```
F := proc(t)
    display(line([ − 2, 0], [cos(t) − 2, sin(t)],
    color = blue), plottools[line]([cos(t) − 2,
    sin(t)], [t, sin(t)], color = blue), plot(sin(x), x
     = 0..t, view = [ − 3..7, − 5..5]))
end proc:
```

```
animate(F, [θ], θ = 0..2 π, background = plot([cos(t) − 2, sin(t), t = 0..2 π]), scaling = constrained, axes = none)
```

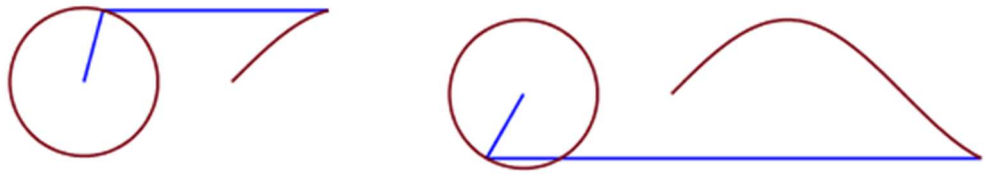

Рисунок 4 — Кадры анимации, полученные с помощью процедур

1.3 Анимация рисования кривой

СКА Maple так же предлагает готовое для поэтапного построения графика. В данном случае график будет формироваться покадрово. Функция *animatecurve* моделирует данное поведение. Она имеет следующий синтаксис [1]:

$$animatecurve(f(x), x = a..b, [options])$$

Команда имеет такие же опции, как и *animate,* и строит по умолчанию 25 кадров. Для демонстрации приведём следующую процедуру:

```
restart, with(plots):
animatecurve(cos(x), x = 0..2 Pi)
```

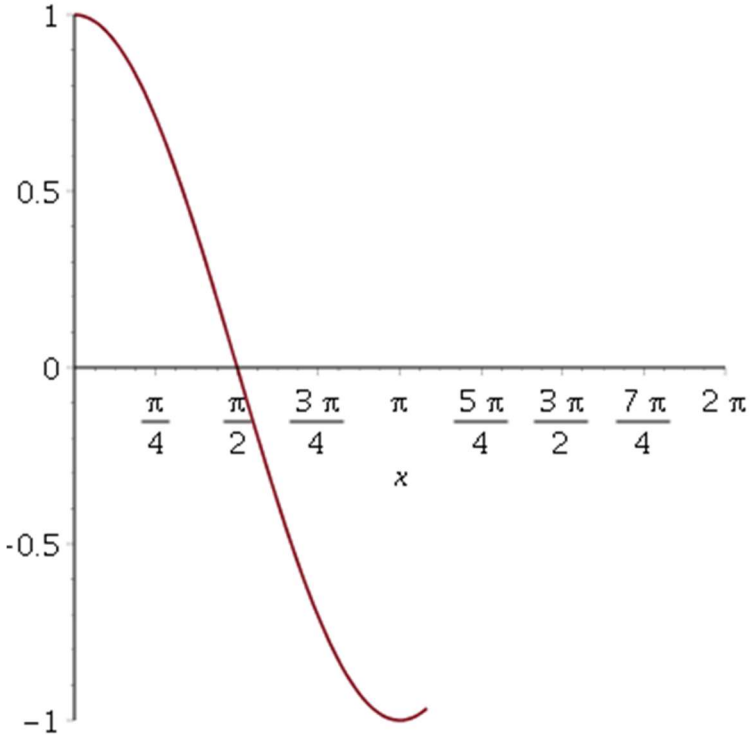

Рисунок 5 — Кадр с анимации кривой

В качестве альтернативы кривая может быть представлена в параметрической форме с помощью функции следующего вида.

```
restart; with(plots) :
animatecurve([16·sin(t)^3, 13·cos(t) − 5·cos(2·t) − 2·cos(3 t) − cos(4 t), t = 0 ..2·Pi], x = −17 ..17)
```

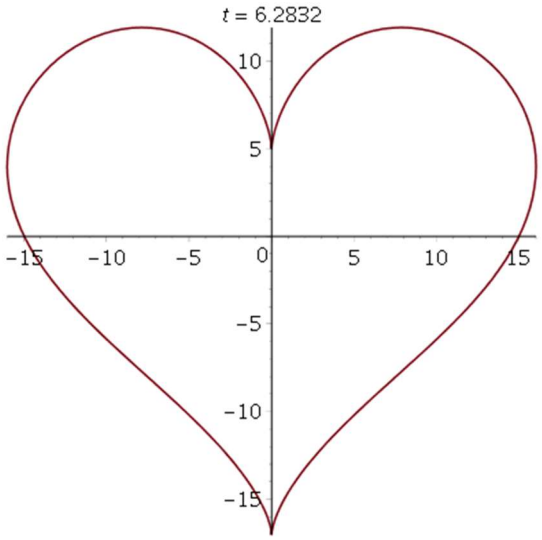

Рисунок 6 — Кадр с анимации кривой, заданной параметрически

1.4 Практический пример

Приведём пример анимации построения графиков сумм ряда Фурье, взяв в качестве параметра порядковый номер частичной суммы. [8]

```
f(x)
```

$$\begin{cases} -\frac{1}{2}x - 2 & 0 < x \text{ and } x < 3 \\ 3 & 3 \le x \text{ and } x \le 5 \end{cases}$$

```
FurieSum := proc(f, k, x1, x2)
    local a0, an, bn, n, l;
    l := 1 / 2 * x2 - 1 / 2 * x1;
    a0 := int(f(x), x = 0..2 * l) / l;
    assume(n::posint);
    an := int(f(x) * cos(π * n * x / l), x = 0..2 * l) / l;
    bn := int(f(x) * sin(π * n * x / l), x = 0..2 * l) / l;
    return 1 / 2 * a0 + add(an * cos(π * n * x / l) + bn * sin(π * n * x / l), n = 1..k)
end proc:
```

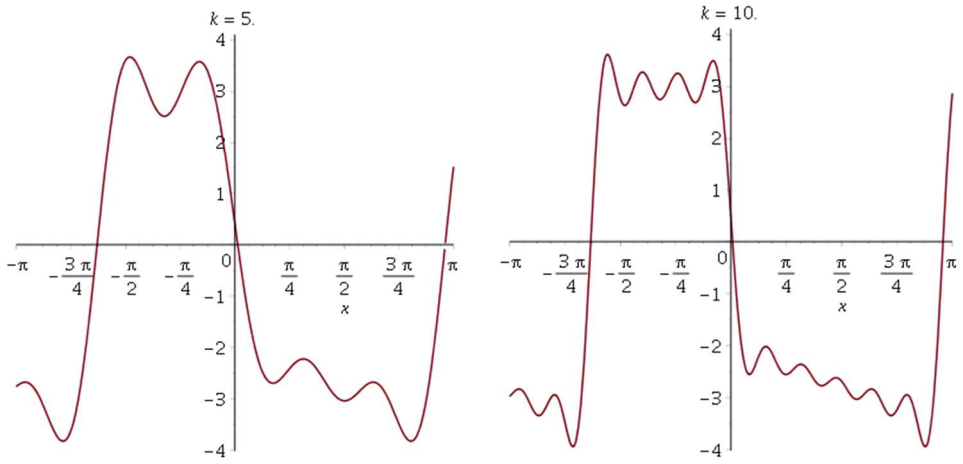

Рисунок 7 — Кадры с анимации построения графиков сумм ряда Фурье

1.5 Анимация функций двух переменных

Для анимации функций двух переменных используется процедура *animate.* По умолчанию создаётся 25 кадров. Синтаксис процедуры [1] для её использования с функцией двух переменных:

$$animate(f(x,y), x = a..b, y = c..d, t = p..q, [options])$$

Приведём небольшой пример [1] для демонстрации.

```
restart; with(plots) :
animate(spacecurve, [[cos(t), sin(t), (2 + sin(A)) t], t = 0 ..24, thickness = 5, numpoints = 200, color = black], A = 0 ..2 π)
```

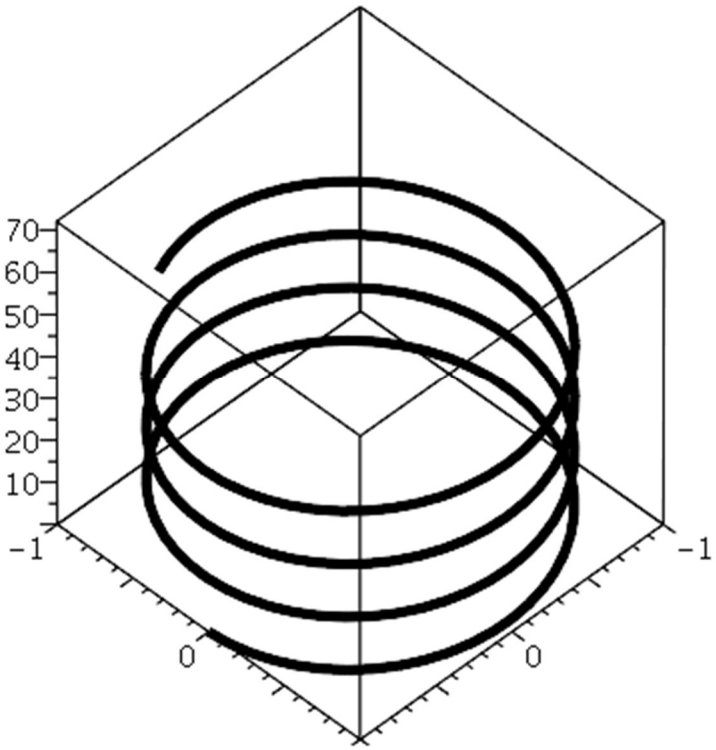

Рисунок 8 — Демонстрации анимации функции 2-х переменных

2 ПОСТРОЕНИЕ АНИМАЦИИ ИЗ ПОСЛЕДОВАТЕЛЬНЫХ КАДРОВ

Что же на самом деле такое анимация? Анимация – это последовательность кадров. Анимация в *Maple* – это последовательность изображений (полученных, например, с помощью процедуры *plot*). Чтобы показать их поочерёдно, как кадры анимации, можно использовать процедуру *display* с опцией *insequence=true*.

2.1 Понятие последовательности

Последовательность – один из фундаментальных объектов Maple. Задать последовательность можно через запятую.

```
a := 1, 2, 3, 4
```

Данной командой мы задали *a* как последовательность чисел от 1 до 4.

2.2 Практический пример. Аппроксимация определённого интеграла

Например, с помощью процедуры *rightbox* пакета *student* можно построить анимацию приближения определённого интеграла [2]. Для этого построим последовательность кадров и выведем её на экран (рис. 9):

```
restart;
with(student):
f := x→exp(-x^2):
rects := rightbox( f(x), x = 0..3, 6 ),
   rightbox( f(x), x = 0..3, 9 ),
   rightbox( f(x), x = 0..3, 12 ),
   rightbox( f(x), x = 0..3, 15 ),
   rightbox( f(x), x = 0..3, 18 ),
   rightbox( f(x), x = 0..3, 21 ),
   rightbox( f(x), x = 0..3, 24 ):

with( plots ):
display(rects, insequence = true);
```

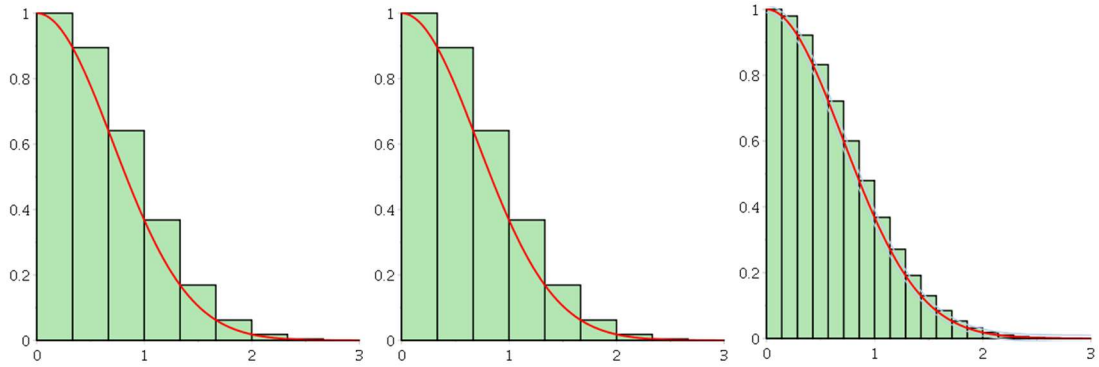

Рисунок 9 — Аппроксимация определённого интеграла

Однако построение последовательности кадров вручную занимает очень много времени, и эту проблему можно решить с помощью процедуры seq.

2.3 Процедура seq

Seq строит последовательность по заданным правилам. Синтаксис процедуры [1]:

$$seq(f(i), i = p..q)$$

Стоит отметить, что по стандарту шаг устанавливается равным единице. То есть, например,

$$seq(j, j = 2.1..8.3)$$

даст следующий результат:

$$2.1, 3.1, 4.1, 5.1, 6.1, 7.1, 8.1$$

Диапазон для шага последовательности не обязательно должен быть арифметической прогрессией. Можно, например, задать для этого свою собственную последовательность.

```
restart;
seq_ := 1, 2, 4, 8, 16 :
seq(A^3 + A^2 + A + 1, A = seq_)
```

Данный код выдаст следующий результат:

$$4, 15, 85, 585, 4369$$

2.4 Практический пример. Применение seq для автоматизации создания кадров

Теперь с помощью *seq* мы можем построить намного больше кадров, написав более короткую команду. Создадим последовательность кадров с помощью данной процедуры (результат – на рис. 10).

```
restart;
with(plots):
a := -3: b := 3: c := -.5: d := 2: n := 20:
f := x → exp(-x^2):

g := x → value(Int(f(t), t = a..x)):

an1 := plot(f(x), x = a..b, view = [a..b, c..d],
    color = blue):

for i from 1 to n do
  k := a + (b-a)*i/n:
  an2[i] := plot(f(x), x = a..k, filled = true, view
    = [a..b, c..d],
      color = turquoise):
  an3[i] := plot(g(x), x = a..k, view = [a..b, c
    ..d], color = red):
end do:

p := plots[display]([seq(an2[i], i = 1..n)],
   insequence = true):
q := plots[display]([seq(an3[i], i = 1..n)],
   insequence = true):
l := plot(sqrt(Pi), view = [a..b, c..d], color
    = green):

display(an1, p, q, l);
```

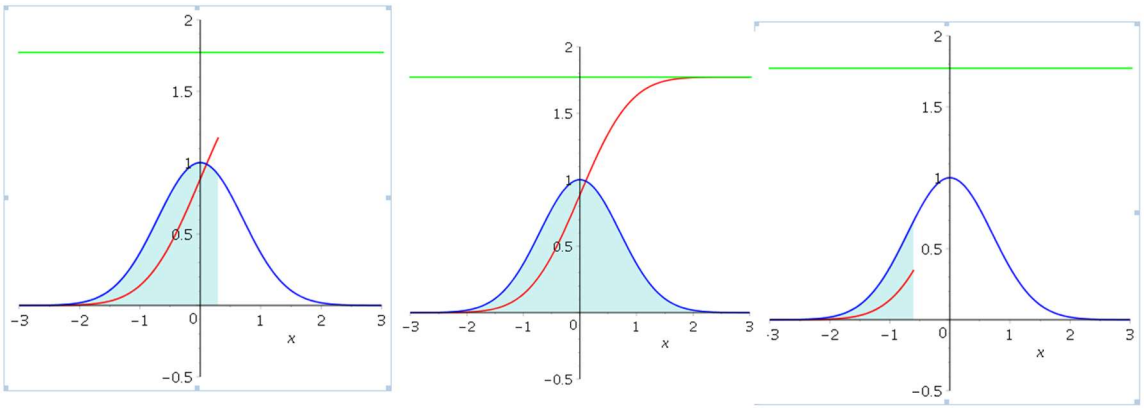

Рисунок 10 — Кадры анимации

2.5 Практический пример. Визуализация приближения функции с помощью ряда Тейлора

Рассмотрим функцию, разложение которой уже известно [7]:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Создадим функциональный оператор, который будет задавать n-ю частичную сумму ряда Тейлора:

$$ex := (x, k) \rightarrow evalf\left(\sum_{n=0}^{k}\left(\frac{x^n}{n!}\right)\right) :$$

Далее построим функцию e^x для сравнения:

$$explot := plot(\exp(x), x = -10..10, y = -1..10, color = grey) :$$

Теперь будем строить кадры, с каждым кадром увеличивая n.

```
> Frames := seq(plot(ex(x, t), x = -10..10, y = -1..10, thickness = 2), t = 0..15) :
> Animation := plots[display](Frames, insequence = true) :
> plots[display](Animation, explot)
```

Результат можно наблюдать на рисунке 11:

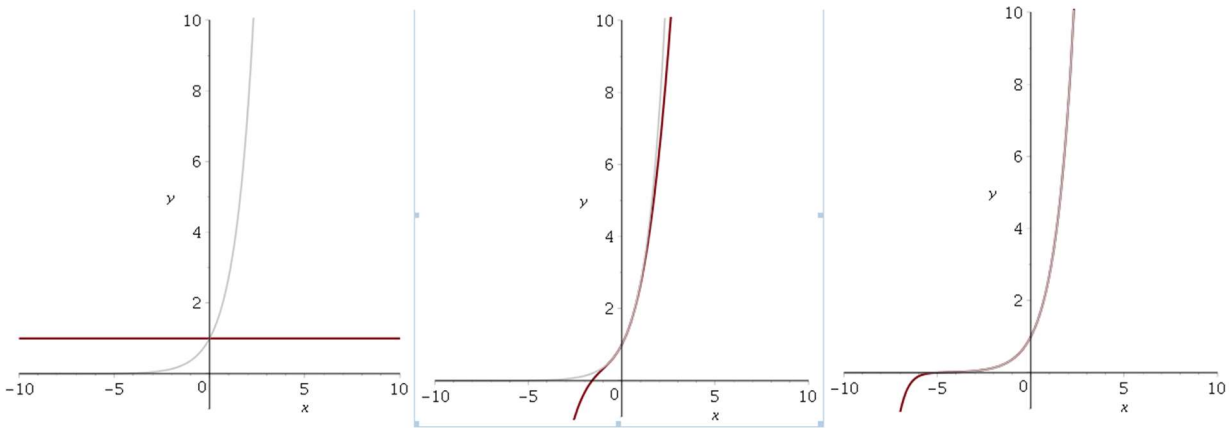

Рисунок 11 — Приближение экспоненты рядом Тейлора

2.6 Построение анимаций движения текста

СКА *Maple* обладает большим функционалом для анимации не только функций, но и текста. Для этого можно воспользоваться функцией *textplot*. Результат можно видеть на рис. 12.

```
restart:
with( plots ):
MovingText := seq( textplot( [cos(2*Pi/50*i),
sin(2*Pi/50*i), "(cos t, sin t)" ], font = [TIMES, ROMAN, 20] ), i = 0..50 ):
display( MovingText, insequence = true, view = [-3..3,-3..3] );
```

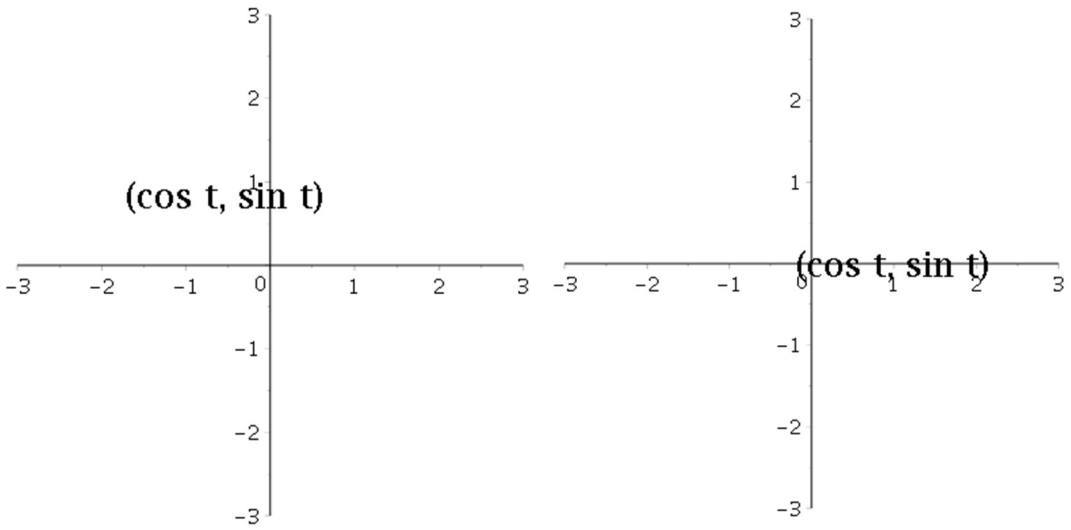

Рисунок 12 – Анимация движения текста

3 ПОСТРОЕНИЕ АНИМАЦИЙ ФИЗИЧЕСКОГО ВЗАИМОДЕЙСТВИЯ ОБЪЕКТОВ

СКА *Maple* можно использовать для анимации взаимодействия физических объектов. Для этого нужно составить закон зависимости координат от времени каждого из тел физической системы и потом последовательно вывести кадры в анимацию.

3.1 Построение анимации движения точки.

Для построения движения материальной точки напишем собственную процедуру *dot*:

```
dot:=proc(x, y)
    plots[plots:-pointplot]([[x, y]], color = blue, symbol = solidcircle, symbolsize
    = 40)
end proc:
```

Построим анимацию падения материальной точки с опцией *trace* для просмотра траектории падении точки в поле силы тяжести.

Результат выполнения можно наблюдать на рис. 13.

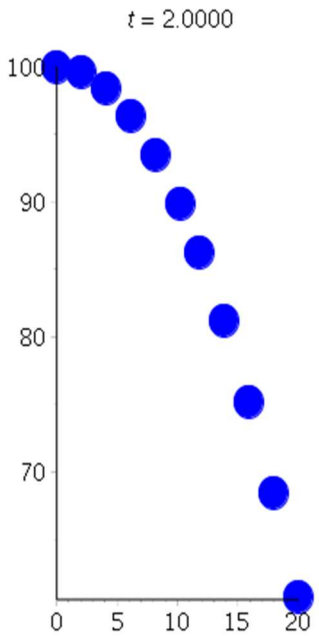

Рисунок 13 — Анимация падения материальной точки

Следующая процедура демонстрирует колебания точки по закону синуса:

```
animate( ball, [0, sin(t)], t = 0..4 π, scaling
    = constrained, frames = 100)
```

Результат можно наблюдать на рисунке 14:

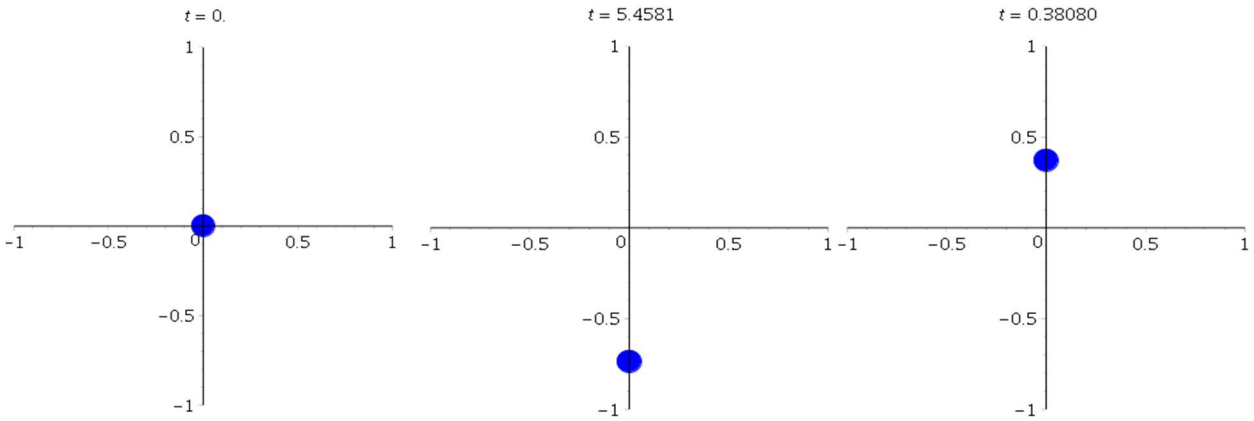

Рисунок 14 — Анимация движения материальной точки по закону синуса

3.2 Построение сложной анимации взаимодействия физических объектов.

С помощью СКА *Maple* можно строить и более сложные анимации взаимодействия. Их построение сопровождается более сложными законами движения, и, соответственно, более сложными зависимостями. Приведём пример [2].

```
restart: with(plots): with(plottools):
dx := 1:
dy := 0.3:
d12 := 4:
v1 := 1:
t1 := (d12 - 2·dx)/v1:
ω := 1:
r := 0.2:
a := v1/(2·ω):
x2 := (t) → piecewise(t < t1, d12, d12 + a·sin(ω·(t - t1))):
x1 := (t) → piecewise(t < t1, v1·t, d12 - 2·dx + a·sin(ω·(t - t1))):
dxk := t → (5 - x2(t))/10:
F := proc(t)
    plots[plots:-display](plottools:-disk([x1(t) + (-1)*0.5*dx, r], r, color
     = yellow), plottools:-disk([x1(t) + 0.5*dx, r], r, color = yellow), plottools:-
    disk([x2(t) + (-1)*0.5*dx, r], r, color = yellow), plottools:-disk([x2(t)
     + 0.5*dx, r], r, color = yellow), plottools:-translate(plottools:-rectangle([
     - dx, 2*r - dy], [dx, 2*r + dy], color = green), x1(t), 0), plottools:-
    rectangle([-1, -1], [9, 0], color = khaki), plottools:-translate(plottools:-
    rectangle([-dx, 2*r - dy], [dx, 2*r + dy], color = green), x2(t), 0),
    plottools:-translate(plottools:-rectangle([-dx, 0], [dx, 2*dy], color
     = cyan), 8, 0), plottools:-curve([[7, dy], [6.5, dy], [6.5 + (-1)*0.5
    *dxk(t), dy + 0.1], [6.5 + (-1)*1.5*dxk(t), dy - 0.1], [6.5 + (-1)*2.5
    *dxk(t), dy + 0.1], [6.5 + (-1)*3.5*dxk(t), dy - 0.1], [6.5 + (-1)*4.5
    *dxk(t), dy + 0.1], [6.5 + (-1)*5.5*dxk(t), dy - 0.1], [6.5 + (-1)*6.5
    *dxk(t), dy + 0.1], [6.5 + (-1)*7.5*dxk(t), dy - 0.1], [6.5 + (-1)*8.5
    *dxk(t), dy + 0.1], [6.5 + (-1)*9.5*dxk(t), dy - 0.1], [x2(t) + dx + 0.5,
    dy], [x2(t) + dx, dy]]))
end proc:
animate(F, [t], t = 0..5, frames = 30, axes = none, scaling = constrained)
```


Результат выполнения можно видеть на рис.15

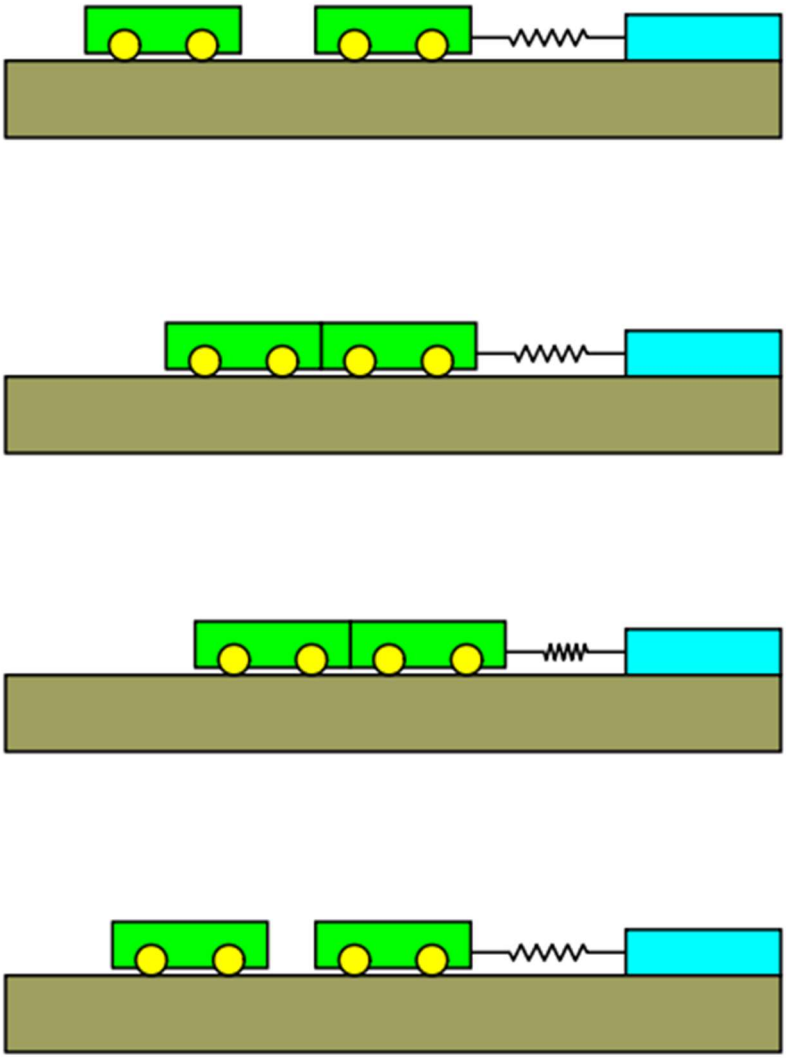

Рисунок 15 — Кадры анимации столкновения физических объектов.

ЗАКЛЮЧЕНИЕ

В результате работы были изучены основные приемы построения анимаций в СКА *Maple*. На практике рассмотрено множество задач и различных применений анимаций для визуализации и проверки решений. Было обнаружено, что есть два способа создания анимации: использование встроенных функций *Maple* и с помощью функции *display* пакета *plots* с параметром *insequence=true*. Таким образом, практически нет ограничений на создание анимации в *Maple*, поскольку все, что вам нужно сделать, это отрисовать кадры. Таким образом, мы можем сделать вывод, что все, что можно нарисовать в Maple, можно и анимировать.

Также *Maple* умеет анимировать не только графики, но и точки, процессы построения графиков. Такие возможности значительно расширяют круг задач, к которым можно применить визуализацию решения. Это было подтверждено

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Maple Documentation [Электронный ресурс]. – Режим доступа: https://www.maplesoft.com/support/help/

[2] Maple animation / John F. Putz - Chapman & Hall/CRC, 2003.

[3] Maple 6. Решение задач высшей математики и механики / Матросов А.В., BHV-Санкт-Петербург, 2001.

[4] Математический анализ. Сборник задач и решений с применением системы Maple / Кузнецова О.С., Кирсанов М.Н. – Инфра-М, 2021.

[5] Высшая математика. Часть 1. / Жевняк Р. М., Карпук М. А. – Минск, Вышэйшая школа, 1984.

[6] Высшая математика. Часть 2. / Жевняк Р. М., Карпук М. А. – Минск, Вышэйшая школа, 1985.

[7] Высшая математика. Часть 3. / Жевняк Р. М., Карпук М. А. – Минск, Вышэйшая школа, 1985.

[8] Калугина, М. А. Математический анализ. Лабораторный практикум в системе Maple: учеб.-метод. пособие / М. А. Калугина. – Минск, БГУИР, 2018.