Л. Л. Голубева, А. Э. Малевич, Н. Л. Щеглова

Компьютерная математика.
АНАЛИТИЧЕСКАЯ ГЕОМЕТРИЯ
В КОМПЬЮТЕРНОЙ СРЕДЕ *Mathematica*

Минск: БГУ, 2021

Оглавление

ЛАБОРАТОРНАЯ РАБОТА
ВЕКТОР НА ПЛОСКОСТИ

Постановка задачи

Требуется спроектировать и построить математическую систему, позволяющую решать задачи в области аналитической и компьютерной геометрии. Среда моделирования – символьный математический пакет *Mathematica*.

3. Вектор

Прежде чем выполнять задания, перенесите в новый электронный документ все функции из предыдущей лабораторной работы, которые позволяют работать с объектом «точка на плоскости». А именно: конструкторы, функции для вычисления свойств, построения графического образа. Эти функции разместите в ОДНОЙ и той же ячейке типа `Input`.

3.1 Внутреннее представление вектора

Задание 3.1

Спроектируйте внутреннее представление объекта «вектор на плоскости».

Выполнение задания 3.1

Поставим в соответствие математическому понятию «вектор» объект «вектор на плоскости» в среде *Mathematica*.

Предположим, на плоскости зафиксирована прямоугольная декартова система координат. Следуем соглашениям, прописанным в (1.2).

Представлением объекта «вектор на плоскости» в *Mathematica* будем называть выражение вида

```
kmVector[km, "id", {x, y}]
```
(3.1)

где **x, y** – одноименные координаты вектора в прямоугольной декартовой системе координат.

Задание 3.2

Постройте булеву функцию **NamedQ**, позволяющую определить, имеет ли представленный объект «вектор на плоскости» имя.

Выполнение задания 3.2

Выполнить самостоятельно.

3.2 Конструкторы вектора

Задание 3.3

Спроектируйте функции-конструкторы объекта «вектор на плоскости».

Выполнение задания 3.3

Проектируем конструкторы вектора таким образом, чтобы при внесении изменений в систему изменялось как можно меньшее количество конструкторов. Для этого используем метод выделения базового конструктора.

Назовем конструктор объекта `базовым`, если соответствующее ему глобальное правило строит представление объекта, используя только входные данные, и при этом никакие другие конструкторы объекта не участвуют в построении представления объекта.

Такой подход позволит удобен тем, что если возникнет необходимость изменить/подправить внутреннее представление объекта, то достаточно переписать только базовые конструкторы. Остальные же глобальные определения не потребуют переделки.

Выявим способы, которыми мы можем задавать вектор. Задаваемой информацией может быть

1) имя и декартовы координаты вектора;
2) декартовы координаты вектора;
3) имя вектора и две различные точки: начало и конец направленного отрезка, соответствующего вектору;
4) две различные точки: начало и конец направленного отрезка, соответствующего вектору;
5) имя, орт (единичный вектор, со направленный с задаваемым вектором), и длина вектора.
6) орт и длина вектора.

Построим левые части глобальных определений 1) – 6), формирующих представление объекта «вектор на плоскости». Введем обозначения, которые будем использовать в образцах: **`coord`** – координаты вектора, **`id`** – имя вектора, **`P0`** и **`P1`** – экземпляры объекта «точка на плоскости», указывающие соответственно начало и конец направленного отрезка, соответствующего вектору.

Описания образцов, соответствующих способам задания 1) – 6), сведем в таблицу 3.1.

Таблица 3.1

Образцы для задания объекта «вектор на плоскости»

	Именованный	Без имени
Декартовы координаты вектора	1) **`kmVector[id_String, coord_List]`**	2) **`kmVector[coord_List]`**
Две точки: начало и конец отрезка	3)	4)

	kmVector[id_String, P0_kmPoint, P1_kmPoint]	**kmVector[P0_kmPoint, P1_kmPoint]**
Орт и длина вектора	5) Написать самостоятельно	6) Написать самостоятельно

Выберем в качестве базового способ задания 1), когда вектор задается именем и координатами, и левая часть глобального определения имеет вид **kmVector[id_String,coord_List]**.

Покажем далее, что каждый из конструкторов 2) – 4) можно свести к работе конструктора 1).

Рассмотрим способ 2) задания вектора, когда указываются только координаты вектора. При этом левая часть глобального определения имеет вид **kmVector[id_String,coord_List]**.

Этот случай можно свести к способу задания 1). Конструктор (3.3), сводит вычисление представления вектора к способу задания 1), указывая в правой части правила в качестве имени вектора пустую строку

```
kmVector[coord_List] := kmVector["", coord]                (3.3)
kmVector[{2, -1}]
kmVector[, {2, -1}]
```

Далее построим глобальное определение для способа задания 3), когда вектор задается именем и двумя различными точками: началом и концом направленного отрезка, соответствующего этому вектору.

При построении представления вектора в этом случае мы вызовем конструктор вида 1), где в качестве второго аргумента укажем выражение, которое вычислит координаты вектора.

```
kmVector[id_String,P0_kmPoint,P1_kmPoint]:=
     kmVector[id, выражение для вычисления
                                  координат вектора]
```

Рассмотрим теперь случай, когда вектор задается способом 4).

В отличие от способа задания 3), здесь имя вектора не указывается. Тогда мы можем вычислить имя вектора, если именованы начало и конец направленного отрезка, соответствующего вектору. В противном случае зададим экземпляру объекта нарицательное имя «Вектор».

Таким образом, при способе задания 4) мы вызываем конструктор вида 3), который, в свою очередь, передаст вычисление базовому конструктору 1).

```
kmVector[P0_kmPoint,P1_kmPoint]:=
     kmVector[выражение для вычисления имени вектора,
```

```
P0,P1]
```

Конструкторы 5) и 6) спроектируйте самостоятельно, по аналогии с проектированием конструкторов 2), 3) и 4).

Задание 3.4

Создайте функции-конструкторы объекта «вектор на плоскости» для случаев, когда вектор задается двумя различными точками: началом и концом направленного отрезка, соответствующего вектору, с указанием имени либо нет. Функции должны сводить вычисления к базовому конструктору 1).

Выполнение задания 3.4

Рассмотрим случай 3), когда вектор задается именем и двумя различными точками: началом и концом направленного отрезка, соответствующего вектору.

Зададим точки, имеющие имена `A` и `B` соответственно, ассоциируем их с символами `P0` и `P1`

```
{P0, P1} = MapThread[kmPoint[#1, #2] &,
  {{"A", "B"}, {{3, 4}, {-2, 3}}}]
{A(3, 4), B(-2, 3)}
```

Чтобы использовать базовый конструктор 1), следует вычислить координаты вектора

```
P1@"coord" - P0@"coord"
{-5, -1}
```

и указать выражение для вычисления координат вектора в качестве второго аргумента вызываемого конструктора 1)

```
kmVector[id_String, P0_kmPoint, P1_kmPoint] :=
 kmVector[id, P1@"coord" - P0@"coord"]
```
(3.4)

Таким образом, в случае задания вектора по имени и координатам начала и конца направленного отрезка, соответствующего вектору, мы готовим информацию для работы базового конструктора 1)

```
kmVector["CD", P0, P1]
kmVector[CD, {-5, -1}]
```

Базовый конструктор, строящий представление вектора при его способе задания 1), мы напишем позже.

Следует отметить также принцип формирования имени вектора. В случае 3), когда мы задаем имя вектора непосредственно в аргументах конструктора, игнорируются имена точек, посредством которых этот вектор определяется.

Рассмотрим теперь случай 4), когда вектор задается двумя различными точками: началом и концом направленного отрезка, соответствующего вектору. Этот способ задания отличается от способа 3) тем, что имя вектора не указано явно.

Тогда формирование имени может взять на себя создаваемое глобальное определение. Как только имя сформировано, можно вызвать конструктор 3).

Для формирования имени примем следующее, общепринятое в математике, соглашение: если обе заданные точки имеют имя, то вектор именуем, подряд записывая имя точки-начала и имя точки-конца направленного отрезка.

```
StringJoin[P0@"id", P1@"id"]
AB
```

Узнать, именованы ли обе точки, поможет функция **NamedQ**, которая должна быть определена и для объекта **kmPoint**.

```
And@@NamedQ/@{P0, P1}
True
```

Тогда возможность 4) задания вектора определяется конструктором

```
kmVector[P0_kmPoint, P1_kmPoint] :=                    (3.5)
 kmVector[
  If[And@@NamedQ/@{P0, P1},
   StringJoin[P0@"id", P1@"id"], ""], P0, P1]
```

При выполнении правила (3.5) процесс вычисления передается правилу (3.4), которое, в свою очередь, вызывает базовый конструктор 1)

```
kmVector[P0, P1]
kmVector[AB, {-5, -1}]
```

Задание 3.5

Постройте конструктор объекта «вектор на плоскости» в случае, когда он именован либо нет и задается направлением (вектором-ортом,

объектом!) и длиной. Функция должны сводить вычисления к базовому конструктору.

Выполнение задания 3.5

Выполнить самостоятельно, аналогично выполнению задания 3.4.

Задание 3.6

Создайте возможность задания объекта «вектор на плоскости» посредством указания имени вектора и его декартовых координат.

Выполнение задания 3.6

Выполнить самостоятельно.

3.3 Свойства вектора

Задание 3.7

Напишите оператор, возвращающий имя объекта «вектор на плоскости», если оно существует, и нарицательное имя **"Вектор"** в противном случае.

Выполнение задания 3.7

Выполнить самостоятельно.

Задание 3.8

Напишите оператор, возвращающий координаты заданного объекта «вектор на плоскости».

Выполнение задания 3.8

Выполнить самостоятельно.

Задание 3.9

Напишите оператор, возвращающий длину заданного объекта «вектор на плоскости».

Выполнение задания 3.9

Построим сначала компьютерную модель для вычисления длины вектора по его заданным координатам.

Пусть вектор *V* имеет координаты *(Vx, Vy)*. В *Mathematica* координаты вектора представлены списком, содержащим два элемента `{Vx, Vy}`. Тогда операция извлечения корня квадратного из скалярного произведения списка самого на себя (**Dot**), представленная в виде

```
√{Vx, Vy}.{Vx, Vy}
```

$$\sqrt{Vx^2 + Vy^2}$$

может служить компьютерной моделью для вычисления длины вектора.

Завершите выполнение задания самостоятельно.

Задание 3.10

Напишите оператор, возвращающий орт заданного объекта «вектор на плоскости».

Выполнение задания 3.10

Ортом вектора $V \neq 0$ называют вектор единичной длины, со направленный с вектором V.

Орт вектора V будем обозначать Ve и вычислять по формуле

$$Ve = \frac{V}{|V|}, \text{ где } |V| \text{ – длина вектора } V,\ V \neq 0.$$

Завершите выполнение задания самостоятельно.

3.4 Компьютерные модели в задачах о точках и векторах

Задание 3.12

Постройте в *Mathematica* выражение, которое определяет, являются ли два вектора, заданные на плоскости, коллинеарными.

Выполнение задания 3.12

Рассмотрим два ненулевых вектора a и b, заданные своими координатами

$$a = \begin{pmatrix} a_x \\ a_y \end{pmatrix},\ b = \begin{pmatrix} b_x \\ b_y \end{pmatrix}.$$

Известно, что векторы a и b являются коллинеарными тогда и только тогда, когда существует $\lambda \in \square$, $\lambda \neq 0$ такое, что $a = \lambda b$. Формально это условие можно представить в координатном виде $\frac{a_x}{b_x} = \frac{a_y}{b_y}$, откуда следует $a_x b_y - a_y b_x = 0$, или, используя понятие определителя матрицы

$$a_x b_y - a_y b_x = \begin{vmatrix} a_x & b_x \\ a_y & b_y \end{vmatrix} = 0.$$

Утверждение 3.1 Векторы a и b коллинеарны тогда и только тогда, когда определитель матрицы, содержащей координаты этих векторов, равен нулю, т. е.

$$|a \quad b| = 0, \tag{3.9}$$

где a - столбец координат вектора a (a_x, a_y) и b - столбец координат вектора b (b_x, b_y).

Построим в *Mathematica* выражение вида (3.9). Рассмотрим два вектора, ассоциируем их с символами **V1** и **V2**

```
V1 = kmVector["a", {ax, ay}]
V2 = kmVector["b", {bx, by}]
kmVector[km, a, {ax, ay}]
```

```
kmVector[km, b, {bx, by}]
```

Тогда правую часть равенства (3.9) представим в *Mathematica*

```
Det[{V1["coord"], V2["coord"]}]
-ay bx + ax by
```

Булево выражение, возвращающее **`True`** в случае коллинеарности векторов **`V1`** и **`V2`**, и **`False`** в противном случае, запишем в виде

```
Det[{V1["coord"], V2["coord"]}] === 0
```
(3.10)

Постройте тест-функцию, которая для двух заданных векторов плоскости возвращает **`True`** в случае их коллинеарности и **`False`** в противном случае. Имя функции должно нести смысловую нагрузку. Напишите спецификацию функции. Тестируйте работу функции.

Задание 3.13

Постройте булеву функцию **`isVectorsCollinear`**, определяющую взаимную направленность двух заданных векторов плоскости. Функция возвращает одно из значений, соответствующих следующему расположению векторов:

1	коллинеарны и сонаправлены,
-1	коллинеарны и противонаправлены,
2	не коллинеарны.

Выполнение задания 3.13

Выполнить самостоятельно.

Задание 3.14

Постройте булеву функцию **`isVectorsOrthog`**, определяющую, являются ли два заданных вектора плоскости перпендикулярными.

Выполнение задания 3.14

Задание 3.14 выполните самостоятельно, используя материал, представленный ниже.

Утверждение 3.4 Ненулевые векторы a и b взаимно перпендикулярны тогда и только тогда, когда выполняется условие

$$a \circ b = 0 \qquad (3.15)$$

Следует помнить, что при представлении выражения (3.15) в *Mathematica* знак равенства соответствует встроенной булевой функции **`SameQ`**.

Задание 3.15

Постройте функцию *VectorsDirection,* определяющую взаимное направление двух векторов a и b плоскости. Функция возвращает одно из

значений, соответствующих следующему взаимному направлению векторов *a* и *b*:

0	перпендикулярны,
1	коллинеарны и со направлены,
-1	коллинеарны и против направлены,
2	не коллинеарны и не перпендикулярны.

Выполнение задания 3.15

Выполнить самостоятельно.

Задание 3.16

Создайте функцию *VectorBisector,* вычисляющую единичный вектор, со направленный с биссектрисой угла между векторами *a* и *b*.

Выполнение задания 3.16

Направление биссектрисы угла между ненулевыми векторами *a* и *b* можно вычислить, если выполнить операцию сложения двух векторов, со направленных с заданными векторами *a* и *b* и имеющих одинаковую длину.

Уравнять длину и сохранить направление можно, умножая каждый из векторов на длину другого. После выполнения операции сложения векторов достаточно запросить орт результирующего вектора.

Для реализации модели в *Mathematica* введем два глобальных правила: правило сложения двух векторов и правило умножения вектора на число. Каждое из них ассоциируем с символом **kmVector**.

```
V1 = kmVector["a", {ax, ay}];
V2 = kmVector["b", {bx, by}];

VectsSum[V1_kmVector, V2_kmVector] ^:=
 kmVector[V1["coord"] + V2["coord"]]
VectsSum[V1, V2]
kmVector[km, , {ax + bx, ay + by}]

VectScale[V_kmVector, λ_] ^:= kmVector[λ V["coord"]]
VectScale[V1 , k]
kmVector[km, , {ax k, ay k}]
```

Теперь зададим векторы *a (2, 0)* и *b (0, 5)*, ассоциируя их с символами **V1** и **V2**

```
V1 = kmVector["a", {2, 0}];
V2 = kmVector["b", {0, 5}];
```

Получим векторы одинаковой длины, со направленные с заданными

```
VectScale[V2 , V1["length"]]
VectScale[V1 , V2["length"]]
kmVector[km, , {0, 10}]
kmVector[km, , {10, 0}]
```

Прежде чем выполнить операцию сложения двух векторов, запишем их списком

```
MapThread[VectScale[#1, #2["length"]] &,
 {{V1, V2}, {V2, V1}}]
{kmVector[km, , {10, 0}], kmVector[km, , {0, 10}]}
```

Тогда операция сложения векторов выполняется посредством функции **Apply**

```
VectsSum @@ MapThread[VectScale[#1, #2["length"]] &,
  {{V1, V2}, {V2, V1}}]
kmVector[km, , {10, 10}]
```

Таким образом, выражение, вычисляющее единичный вектор, со направленный с биссектрисой угла между векторами a и b, можно представить в виде

```
(VectsSum @@                                          (3.16)
   MapThread[VectScale[#1, #2["length"]] &,
    {{V1, V2}, {V2, V1}}])["ort"]
```

Постройте функцию **VectorBisector**, описанную в задании 3.16. Напишите спецификацию функции. Тестируйте работу функции.

Задание 3.17

Создайте функцию *VectorProjection,* вычисляющую вектор-проекцию вектора a на вектор b.

Выполнение задания 3.17

Векторной проекцией вектора *V* на вектор *W* называют вектор, коллинеарный вектору W и имеющий длину $\left\|V\right|\cos\alpha\left|\right.$

$$pr_W V = \left|V\right|\cos\alpha W_e, \qquad (3.17)$$

где α - угол между векторами V и W, W_e - единичный вектор, сонаправленный с вектором W.

Вектор-проекцию вектора V на вектор W называют также **ортогональной составляющей вектора V вдоль вектора W**.

Утверждение 3.5 Векторная проекция вектора V на вектор W может быть вычислена по формуле

$$pr_W V = (V, W_e) W_e \qquad (3.18)$$

где W_e - орт вектора W.

Покажем, что из выражения (3.17) следует (3.18). Используем определение скалярного произведения (V, W) векторов V и W, откуда получаем $|V|\cos\alpha = \frac{(V,W)}{|W|}$. С учетом того, что единичный вектор можно представить в виде $W_e = \frac{W}{|W|}$, получим $pr_W V = \frac{(V,W)}{|W|} W_e = (V, W_e) W_e$.

Представим в *Mathematica* выражение, правую часть равенства (3.18), которое вычисляет вектор-проекцию вектора V на вектор W. Для наглядности рассмотрим векторы

```
V = kmVector[{4, 4}];
W = kmVector[{0, 5}];
```

Получим координаты орта, единичного вектора, со направленного с вектором W

```
W@"ort"
kmVector[km, , {0, 1}]

(W@"ort")@"coord"
{0, 1}
```

Вычислим скалярное произведение векторов V и единичного вектора, со направленного с вектором W

```
Dot[V@"coord", #] &[(W@"ort")@"coord"]
```

Тогда выражение, вычисляющее вектор-проекцию вектора V на вектор W, можно представить в виде

```
VectScale[# , Dot[V@"coord",
     (#)@"coord"]] &[W@"ort"]
```
(3.19)

Постройте функцию **VectorProjection**, описанную в задании 3.17. Напишите спецификацию функции. Тестируйте работу функции.

3.5 Задачи о точках и векторах

Задание 3.21

Решите предложенные ниже задачи, используя построенную ранее систему «точка и вектор на плоскости»:
а) напишите функции пользователя;
б) в текстовых ячейках запишите спецификации[1] построенных функций;
в) тестируйте функции на конкретных экземплярах объектов;
г) изобразите в графической области геометрические объекты, фигурирующие в задаче.

1. Напишите функцию, которая вычисляет длину медианы AD треугольника ABC, если даны координаты его вершин $A(x_A, y_A)$, $B(x_B, y_B)$ и $C(x_C, y_C)$. Использовать понятие вектора.
2. Напишите функцию, которая вычисляет точку пересечения медиан треугольника ABC, если даны координаты его вершин $A(x_A, y_A)$, $B(x_B, y_B)$ и $C(x_C, y_C)$. Использовать понятие вектора.
3. Заданы вершины $A(x_A, y_A)$, $B(x_B, y_B)$ и точка $M(x_M, y_M)$ пересечения медиан треугольника ABC. Напишите функцию, которая вычисляет координаты вершины C. Использовать понятие вектора.
4. Треугольник ABC задан координатами своих вершин $A(x_A, y_A)$, $B(x_B, y_B)$, $C(x_C, y_C)$. Напишите функцию, которая вычисляет расстояние от начала координат до точки пересечения медиан этого треугольника. Использовать понятие вектора.
5. Треугольник ABC задан координатами своих вершин $A(x_A, y_A)$, $B(x_B, y_B)$, $C(x_C, y_C)$. Напишите функцию, которая вычисляет расстояние от вершины A до основания D высоты, опущенной из вершины B на сторону AC. Использовать понятие проекции вектора на вектор.
6. Заданы три последовательных вершины параллелограмма ABCD координатами $A(x_A, y_A)$, $B(x_B, y_B)$, $C(x_C, y_C)$. Напишите функцию, которая вычисляет координаты четвертой вершины D.

Литература

1. Голубева Л.Л., Малевич А.Э., Щеглова Н.Л. Компьютерная математика. Символьный пакет *Mathematica*. Лаб. практикум в 2 ч. Ч 1.- Минск: БГУ, 2012. – 235 с.

---

[1] **Спецификация функции** состоит из описания имени функции, аргументов, назначения функции и выхода. При описании аргументов функции указывается, что они означают и их область допустимых значений. Если число аргументов может быть различно, указывают обязательные и необязательные аргументы и описывают каждый вариант их использования. Выход – значение функции – описывается вплоть до структуры выражения-результата.
Пример спецификации функции. Функция **PolyCoeffSolve[poly, x, vars]** для заданного многочлена **poly** от переменной **x** вычисляет те значения параметров, содержащихся в коэффициентах многочлена, при которых многочлен **poly** тождественно равен нулю (если они существуют). Искомые параметры указываются в списке **vars**. Функция **PolyCoeffSolve** возвращает список вычисленных значений параметров в виде локальных подстановок, выражений с головой **Rule**.