О МОДЕЛИРОВАНИИ ГЕОМЕТРИИ ДВИЖЕНИЯ В СИСТЕМЕ “ВЕКТОР”

Болотов В.П., Коркишко С.В., Кузлякина В.В.

При компьютерном представлении движения геометрических тел, встречающихся в самых различных областях техники, науки и искусства, невозможно обойтись без систем машинной графики и знаний из теории механизмов и кинематики движения. В статье предпринята попытка синтезировать опыт, полученный одним из авторов статьи Валентиной Васильевной Кузлякиной при написании докторской диссертации “Методы и средства автоматизированного проектирования судовых механизмов и машин” (ДВГМА, 1997), и векторно-графический анализ, представляемый остальными авторами.

Основополагающей при определении кинематических параметров движения тел является теорема о замкнутых векторных контурах: **любой структурный модуль механизма можно представить в виде замкнутого векторного контура, при этом каждое звено представляется в виде вектора, модуль которого соответствует его длине, а угол, измеряемый против часовой стрелки от положительного направления оси OX до совпадения с направлением вектора длины звена определяет его положение.** В общем случае уравнение замкнутости контура имеет вид:

$$\Sigma \boldsymbol{l}_i = 0, \qquad (1)$$

где $\boldsymbol{l}_i$ - вектор, соответствующий i-ему звену, входящему в рассматриваемый контур. При составлении уравнения замкнутости направление обхода контура можно выбрать произвольно, но в уравнении (1) вектор $\boldsymbol{l}_i$ записывается со знаком “+”, если его направление совпадает с направлением обхода, и со знаком “-”, если оно противоположно ему.

Уравнение замкнутости векторов можно представить целевой функцией (ЦФ), которая стремится к нулю. Векторно-графическое представление ЦФ в системе “Вектор” (модуль Optim) и нахождение по нему минимума делает возможным поиск необходимых параметров, отвечающих условиям замкнутости контура и корректности задания исходных данных и ограничений. Рассмотрим пример.

Пример 1. Уравнение, удовлетворяющее условию замкнутости векторного контура, для определения параметров точки ***B*** обобщенного структурного модуля (рис. 1) будет иметь вид:

$$r_A + r_{AB} - r_B = 0.$$

Положение и закон движения входного звена определяют движение всех звеньев и точек механизма, поэтому угол поворота кривошипа (точка B) является обобщенной координатой. В проекциях на оси OX и OY координаты, определяющие положение точки *B* можно записать так:

$$x_B = r_B \cos\varphi,$$

$$y_B = r_B \sin\varphi.$$

Не останавливаясь на удобствах, предоставляемых уравнением замкнутости векторного контура, типа расчетов угловых скоростей и ускорений любого звена, кинематических параметров центров масс и т.п., рассмотрим расчет геометрических параметров примера 1.

Определим, что задано и что необходимо определить для того, чтобы построить движение заданного механизма.

Значения длин всех трех векторов должны быть известны.

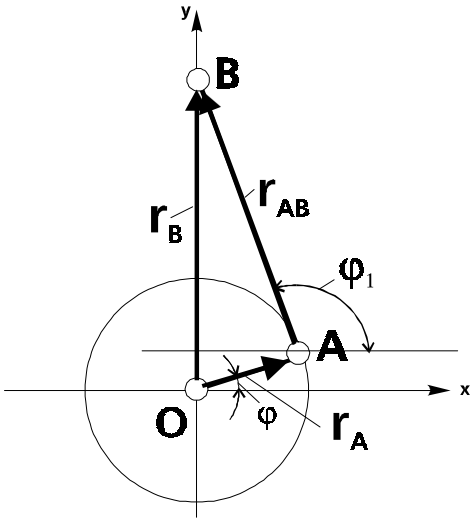

Рис. 1,а. Векторно-кинематическая схема вращательно-поступательного движения в обозначениях общепринятых

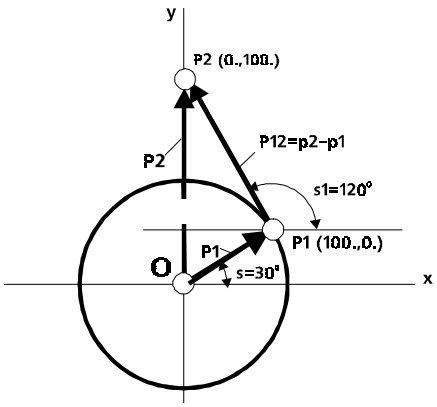

Рис. 1,б.. Векторно-кинематическая схема вращательно-поступательного движения в обозначениях системы “Вектор”

Системы “Вектор” накладывает следующие ограничения: движение точки B должно быть только вертикальным, а точки A - по окружности радиуса, равного длине вектора r_A. Переменным параметром является угол φ вращения точки A вокруг центра O. Требуется определить положение точки B при изменении параметра φ, если известно, что длины векторов r_A и r_{AB} - постоянные величины и для каждого состояния системы выполняется условие замкнутости векторного контура:

$$r_A + r_{AB} - r_B = 0, \qquad (2)$$

Или, в проекциях на оси OX и OY:

$$l_{OA}\cos\varphi + (l_{AB}\cos\varphi_1) - x_B = 0, \qquad (3)$$
$$l_{OA}\sin\varphi + (l_{AB}\sin\varphi_1) - y_B = 0 \qquad (4)$$

Проверим частный случай (рис. 1,б) данного утверждения в системе “Вектор”. Известны координаты точек p1 и p2 (рис. 1,б), углы s и s_1 (s=30., s1=60.) и длина вектора p12 (s12=86) известны. На языке “Калькулятор” проверка выражается следующим образом:

```
: p1=43 , 25.5  p2=0.,100.5 s12=86.0189
 p12=p2-p1
 p=p1+p12-p2
x=(50.
cos(30.*3.14159/180.)+(s12*cos(120.*3.14159
/180.))-x2) =0.029
y=(50.
sin(30.*3.14159/180.)+(s12*sin(120.*3.14159/
180.))-y2) 1.0054
p=0.291958 , -1.0054 , 0.
```

Координаты x и y получились равными нулю с некоторыми погрешностями.

Точка A (рис. 1) определяется в зависимости от параметра φ. Точка B может быть определена, если известен угол φ_1 наклона отрезка AB к оси OX.

ЦФ стремится к нулю

$$F = l_{OA}\cos\varphi + (l_{AB}\cos\varphi_1) - x_B \to 0 \qquad (5)$$

При фиксированном значении φ неизвестным будет параметр φ_1, а x_B=0. Таким образом, при вышеуказанном условии ЦФ будет зависеть от одного параметра φ_1 и будет одномерной. Параметр φ_1 изменяется в пределах от 0 до 180 (угол измеряется против часовой стрелки). Ниже дана МК расчета ЦФ и ее график (рис. 2, а). На рис.2 приведен график ЦФ, а под ним текст макрокоманды (МК) расчета ЦФ. На рис. 2 отражен момент ($s=120^0$), когда достигнут минимум ЦФ, а значения y_B ($l-y_2$) можно вычислить по величине угла s.

При этом y2=99,89. Тоже самое получаем, подставляя значение φ_1=s в уравнение (4):

$$y_B = l_{OA}\sin\varphi + (l_{AB}\sin\varphi_1) = 99{,}89,$$

что совпадает с ожидаемым результатом в пределах погрешностей.

Рис. 2

```
<tkuz0>
: p1= 43.0,25.5  s12=86.0189
y=50.* cos(0.*3.14159/180.)+(s12*
cos(s*3.14159/180.))
tkuz : s=0. p99=0.,y   y99=10000.
y2=50.*$sin(30.*3.14159/180.)+(s12*$sin(s*3.1415
9/180.))

<tkuz>
s > 180. ? exit
y=50.* cos(30.*3.14159/180.)+(s12*
cos(s*3.14159/180.))
y < 0.? y=-y
y < y99 ?  y99=y
tprin : p101=p105 p102=s/2.,y/2.
n9=y99*1000000.
n9 <> 291954 ? exit
tkuz: s=s+5.  p105=p102
```

В более общем случае, когда x_B неизвестно, ЦФ будет двухмерной. Минимум этой функции также даст значение угла φ_1, по которому можно определить координаты точки В. Применяя для каждого фиксированного значения φ предложенную методику, можно вычислять значения φ_1 и таким образом строить динамику движения тел, а также производить все сопутствующие вычисления.

Предложенная методика не является единственной. Если рассматривать только геометрию движения тел, можно использовать и другие подходы ориентированные, на интерфейсе графических систем.

В кинематике движения тел наиболее часто встречаются задачи на пересечение дуг и отрезков (при вращательном и поступательном движении), сопряжения двух, трех и более вращающихся тел, имеющих форму круга, и т.п. Для решения таких задач оптимальным является представления их на комплексной плоскости, основные принципы которого реализованы в модуле Vecline. Реализация этого инструмента в задачах моделирования кинематических движений не только интересна в ракурсе ее применения в вышеуказанном направлении но и неизвестна авторам. На рис. 3 представлен фрагмент меню модуля Vecline системы "Вектор", в котором многие задачи, за исключением состояния "**Линии общего назначения"** и, частично, состояние "**Технологические линии: эволюта, эквидистанта",** реализованы на основе представления геометрическох объектов (окружности, отрезка) на комплексной плоскости. Задание дуг и отрезков прямых на комплексной плоскости, производится через начальную и конечную точки, центральный угол по следующей параметрической формуле:

$$z=(h\nu z_1+(1-h)z_2)/(h\nu+(1-h)), \qquad (6)$$

где

z_1 - начальная точка дуги;

z_2 - конечная точка дуги;

$\nu=e^{i\varphi/2}$ (или $\nu=\cos\varphi/2+i\sin\varphi/2$), где φ - центральный угол дуги.

h - безразмерный параметр, изменяющийся в пределах $0 \le h \le 1$.

Тройка чисел (z_1,z_2, ν) однозначно определяет дугу. Для отрезка центральный угол φ равен 0, $\nu=1$.

При $\nu=1$, уравнение (6) имеет вид

$$z=hz_1+(1-h)z_2, \qquad (7)$$

известный как уравнение отрезка прямой с весовыми функциями в начале и конце отрезка.

Реализация описанного подхода основана на использовании комплексных чисел Как и 2-мерный вектор, комплексное число представляет собой упорядоченную пару чисел, операции с которыми отличаются от векторного исчисления при вычислениях суммы, разности произведения, деления двух комплексных чисел, нахождения модуля комплексного числа и т.п. Ниже приведены программы вычислений над комплексными числами, реализованных в системе "Вектор". Для сравнения даны программы вычислений над векторами.

*** 1 - Линии общего назначения**
1 - Отрезок прямой
2 - Квадратично-рациональная
3 - Кубично-рациональная Безье
4 - Сплайн-Фергюсона
5 - Лагранж 3 точки
6 - Лагранж 4 точки
7 - Параметрический сплайн через четыре точки
8 - B-сплайн
9 - Составная линия из _0] (2-5) отрезков прямых
10 - Составная Линия: прямая - Безье - прямая
11 - ЛИния: прямая - квадратичная - прямая
12 - КВадратичная - прямая - квадратичная
13 - ПРямая - КВадратичная - квадратичная
14 - КВадратично-кубичная
15 - Окружность в плоскости ОП
16 - Прямоугольник в плоскости ОП
17 - Шестиугольник замкнутый
18 - ОКружность в плоскости XY
*20 - Задание линий по проекциям на пл. xy и xz
*21 - Задание Линий по проекциям на пл. xz и yz
22 - Линия из файла F - редактора (линейная интерполяция)
23 - Линия файла F - редактора (сплайновая интерполяция)

*** 3 - Дополнительные построения (дуг, отрезков на комплексной плоскости)**
* 1 - ПРямая через точку, ее длину и угол к ранее заданной прямой
* 2 - Прямая точку, ее длину и направл. вектор
* 3 - Дуга через две точки и центр
* 4 - Дуга Через центр,точку и угол
* 5 - Дуга/отрезок две точки и центр.-й угол
* 7 - Дуга окружность через три точки
* 8 - Дуга Окружность через две точки и радиус

***15 - Задачи на сопряжение дуг и отрезков**
* 1 - Касание прямой дуги окружности
* 2 - Касание отрезком прямой двух дуг окружности
* 3 - Касание дугой окружности отрезка или дуги окружности
* 4 - КАсание дугой окружности (зад. Радиусом) двух об"ектов
* 5 - Касание Дугой окружности (зад. Точкой) двух об"ектов
* 6 - КАсание Дугой окружности (Точка на Объекте) двух об"ектов
* 7 - Касание окружности (зад. центром) прямой

***14 - Пересечение дуг и отрезков**
1 - Задание первого отрезка-дуги
2 - Задание второго отрезка-дуги
3 - Искомая точка пересечения линий (в p190 или p191)

4 - Технологические линии: эволюта, эквидистанта
1 - Изабражение эволюты (и окружности кривизны после 2)
2 - Окружность кривизны (R - в s190, 1/R - в s191) в т. от u =_00.00
3 - Визуализация эквидистантной кривой, по нормали длины =_00000.00
4 - Движение по плоск. контуру фрезы R =_000.00] со скоростью =_00.0
5 - 3-х гр-к Френе (p191,p192,p193 кас.вектор,норм.,круч.) u =_00.00
6 - Движение По простр. контуру фрезы R =_000.00] со скор-ю =_00.00
14 - Востановить визуализацию заданной кривой

Рис. 3. Диаграмма состояний (фрагмент) модуля Vecline системы “Вектор”

Комплексные числа

```
rcompl( r, p )
    p_out.x = r * p.x;
    p_out.y = r * p.y;
return( p_out );
acompl( p1, p2 )
    p_out.x = p1.x + p2.x;

rvec( r, p )
    p_out.x = r * p.x;
    p_out.y = r * p.y;
    p_out.z = r * p.z;
return( p_out );

drvec( r, p )
    p_out.x = p.x/r;
    p_out.y = p.y/r;
    p_out.z = p.z/r;
return( p_out );

avec( p1, p2 )

    p_out.x = p1.x + p2.x;
    p_out.y = p1.y + p2.y;
    p_out.z = p1.z + p2.z;
return( p_out );

svec( p1, p2 )
p_out.x = p1.x - p2.x;
    p_out.y = p1.y - p2.y;
    p_out.z = p1.z - p2.z;
return( p_out );

mvec( p1, p2 )

    p_out.x = p1.y * p2.z  -  p1.z * p2.y;
    p_out.y = p1.z * p2.x  -  p1.x * p2.z;
    p_out.z = p1.x * p2.y  -  p1.y * p2.x;
return( p_out );

  edvec( p1 )
    p_out.y = p1.y + p2.y;
return( p_out );

scompl( p1, p2 )
     p_out.x = p1.x - p2.x;
    p_out.y = p1.y - p2.y;
return( p_out );

mcompl( p1, p2 )
    p_out.x = p1.x * p2.x  -  p1.y * p2.y;
    p_out.y = p1.x * p2.y  +  p2.x * p1.y;
return( p_out );
}

dcompl( p1, p2 )
mod2 = p2.x * p2.x  +  p2.y * p2.y;
if (mod2 > 0.0) {
    p_out.x = (p1.x * p2.x  +  p1.y * p2.y) /
mod2;
    p_out.y = (p2.x * p1.y  -  p1.x * p2.y) /
mod2;
}
else {
p_out.x=0.0; p_out.y=0.0; }
return( p_out );

modcompl( p )
return( sqrt( p.x * p.x + p.y * p.y ) );
```

Векторное исчисление

```
    p_out.x = p1.x/sqrt( p1.x * p1.x + p1.y *
p1.y + p1.z * p1.z );
    p_out.y = p1.y/sqrt( p1.x * p1.x + p1.y *
p1.y + p1.z * p1.z );
    p_out.z = p1.z/sqrt( p1.x * p1.x + p1.y *
p1.y + p1.z * p1.z );
return( p_out );

mskvec( p1, p2 )
return(  p1.x * p2.x + p1.y * p2.y + p1.z *
p2.z  );

modvec( p )
return( sqrt( p.x * p.x + p.y * p.y + p.z * p.z
) );
```

Координаты точки на дуге вычисляются по формуле:

$$x=(h\nu z_1+(1-h)z_2)/(h\nu+(1-h)) \qquad (8)$$

С учетом правил работы с комплексными числами вычисления координаты точки можно проводить покомпанентно (см. ниже фрагмент программы на языке СИ):

```
tp12.x=(s1*(pfi.x*z1.x-pfi.y*z1.y)+(((i1.x-is1.x)*z2.x))-
(((i1.y-is1.y)*z2.y)));
tp12.y=(s1*(pfi.x*z1.y+pfi.y*z1.x)+(((i1.x-
is1.x)*z2.y))+(((i1.y-is1.y)*z2.x)));
tp14.x=(s1*pfi.x+(i1.x-is1.x));
p14.y=(s1*pfi.y+(i1.y-is1.y));
mod2 = tp14.x * tp14.x  +  tp14.y * tp14.y;
if (mod2 > 0.0) {
z.x = (tp12.x * tp14.x  +  tp12.y * tp14.y) / mod2;
```

```
z.y = (tp14.x * tp12.y  -  tp12.x * tp14.y) / mod2;
}
else {
tp2.x=0.0; tp2.y=0.0; }
```

или непосредственно с использованием программ вычислений комлексных чисел

```
z=dcompl( acompl(
rcompl(s1,mcompl(pfi,z1)),
mcompl(scompl(i1,is1),z2)),
acompl( rcompl(s1,pfi),scompl(i1,is1)));
```

где pfi (в формуле (8) это параметр ν) вычисляется по формулам:

```
pfi.x=cos(fi/2.);
pfi.y=sin(fi/2.);
```

На рис. 4 приведен пример реализации геометрии движения, в котором используются методика работы с объектами на комплексной плоскости и методика теории кинематики движения тел. Программы-макрокоманды необходимые, для получения геометрии движения (рис 4.) приведены ниже и состоят из МК задания исходных данных и блоков вычислений задач на пересечение при вращательно-поступательном движении. Программа реализации примера легко модернизируется для любого числа цилиндров с помощью добавления МК типа kuzl2 или организации цикла. Расчеты, выполненные для двухмерного представления геометрии движения, можно использовать и для трехмерного представления реальных сцен с передачей их в систему CG твердотельного моделирования (каждый раз моделируя одно состояние). Тексты макрокоманд, реализующих пример (рис.4).

```
<kuzl0>
: p100=0.,0. s100=20. s110=70.
        x=s100*cos(s)
        y=s100*$sin(s)
: p110=0.,s110 s111=20. s112=10. s114=80.
s115=-80.
p1=0.,s110
s1=$sqrt((x1-x)*(x1-x)+(y1-y)*(y1-y))
kuzl : s=0.
<kuzl>
 s > 2.*3.14 ? exit
okr: p100=0.,0. s100=20.
        x=s100*cos(s)
        y=s100*$sin(s)
p1=0.,s110
kuzl2   $ пересечения верхнего цилиндра
s110=y191
kvadrat: p110=0.,s110 s111=20. s112=10.
col : p115=p110 n9=3
lprin : p101=p p102=p191
$  goto met
kuzl3  $ пересечения правого цилиндра
s114=x191
kvadrat: p110=s114,0.0 s111=10. s112=20.
col : p115=p110 n9=12
lprin : p101=p p102=p191
$ goto met
kuzl4    $  пересечения левого цилиндра
s115=x191
kvadrat: p110=s115,0.0 s111=10. s112=20.
kuzl : s=s+0.2

<kuzl2>
_Пересечение_дуг_и_отрезков
_Задание_первого_отрезка-дуги
_Начало_=_( x+s1    y       00000.0
_Конец__=_( x-s1    y       00000.0
_Задание_центрального_угла_= 0180.0
        _Задание_второго_отрезка-дуги
        _Начало_=_(     0.0   125.0     0.0
        _Конец__=_(     0.0     0.0     0.0
        _Задание_центрального_угла_= 0000.0
_Искомая_точка_пересечения_линий_(в p190)
_Выход
```

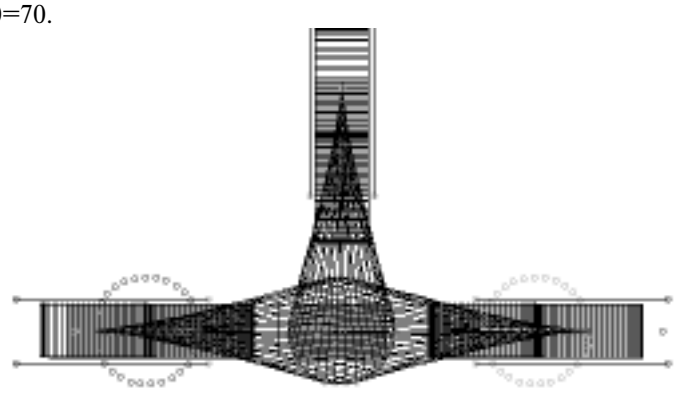

Рис. 4. Пример реализации геометрии движения 3-цилиндрового двигателя внутреннего сгорания в системе “Вектор” (движение без удаления промежуточных состояний)