Белорусский государственный университет, Минск

ГЕНЕРАЦИЯ КИНЕМАТИЧЕСКИХ МОДЕЛЕЙ МЕХАНИЧЕСКИХ СИСТЕМ И ИССЛЕДОВАНИЕ ИХ ДВИЖЕНИЯ С ПРИМЕНЕНИЕМ ФУНКЦИЙ ПАКЕТА *MECHANICAL SYSTEMS*

Медведев Д.Г., Босяков С.М., Кохан Л.Л.

In the present paper development cycles of models of mechanical systems in package Mechanical Systems are submitted. Definitions of the functions forming mathematical system of the analytical equations on the basis of which calculation of kinematics characteristics of model is carried out are resulted. The example of calculation and modeling the three-dimensional crank-slider mechanism is submitted.

В настоящее время широкое применение находят системы компьютерной математики для персональных компьютеров, которые интегрируют в себе современный интерфейс пользователя, решатели математических задач в численном и символьном виде, а также мощные средства графики. Применение таких систем охватывает различные сферы деятельности прикладной математики, в частности, теоретическую и прикладную механику. В настоящей работе представлены функциональные возможностей пакета *Mechanical Systems* [1] компьютерной системы *Mathematica*, позволяющие выполнять моделирование и визуализацию движения механических систем различной сложности.

Внешний пакет *Mechanical Systems* представляет собой совокупность подпакетов, написанных на языке программирования системы *Mathematica* и предназначенных для анализа и проектирования плоских и пространственных твердых механизмов. Также как и остальные пакеты расширения, он имеет независимую платформу и может быть установлен совместно с любой версией системы *Mathematica*, начиная с версии 2.2. Пакет *Mechanical Systems* состоит из двух основных подпакетов – `Mech2D` и `Mech3D`, предназначенных для моделирования и анализа работы двумерных и трехмерных механических систем соответственно. Эти подпакеты могут быть инициализированы аналогично стандартным пакетам расширения системы, например, с помощью функций `Get["Mech`Mech2D`"]`, `Needs["Mech`Mech3D`"]`, а также конструкции `<<Mech`Mech2D`.` Отметим, что невозможно одновременно загрузить подпакеты `Mech2D` и `Mech3D`, поскольку они используют функции, имеющие одинаковые идентификаторы, но различный смысл. Для удаления определений, сделанных в ходе сессии, или отключения ядра пакета *Mechanical Systems* (подпакетов `Mech2D` и `Mech3D`) предназначены функции `ClearMech[]` (возвращает ядро в исходное состояние) и `KillMech[]` (полностью закрывает ядро *Mechanical Systems*).

Рассмотрим этапы генерации модели пространственного кривошипно-шатунного механизма, представленного на рис. 1, состоящего из двух движущихся тел – кривошипа и ползунка. Система приводится в движение вращающимся кривошипом, который вынуждает ползунок перемещаться по направляющим, параллельным оси кривошипа. Кривошип соединен с ползунком через шатун, который смоделирован ограничением расстояния.

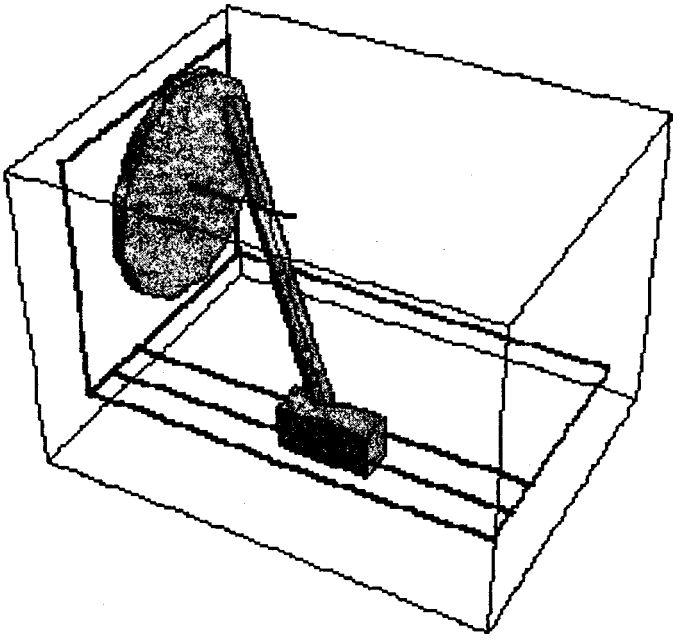

Рис. 1. Кривошипно-шатунный механизм

Каждому независимому элементу механизма необходимо присвоить уникальный целочисленный номер. Выбор номера для элемента произволен, за исключением номера 1, который обязательно должен быть присвоен соответствующему неподвижному элементу (основанию) механизма. Так, в нашем случае, имеем

```
ground=1;crank=2;slider=3;
```

Заметим, что реальный кривошипно-шатунный механизма имел бы четвертое тело - шатун между кривошипом и ползунком. Все элементы модели связаны с локальной системой координат, необходимой для описания характерных точек, линий и тех особенностей элементов модели, на которые накладываются механические связи. Для формирования элемента механической системы предназначена функция `Body[bnum,options]`. Здесь аргумент `bnum` – положительное целое число, определяющее номер тела, `options` – необязательные входные параметры, в качестве которых могут выступать опции `Mass`, `Inertia`, `Centroid`, `PointList`, `InitialGuess`. Функция `Body` в нашем случае, определяющая стержневую систему как основание мханизма приведена ниже

```
bd[ground] = Body[ground,
PointList→{{0,10,12},{10,10,12},{1,0,0},{0,1,0}}];
```

Таким образом, основание (`ground`) имеет четыре ключевых точки, определяющие данный объект, причем две из них определяют положение оси вращения кривошипа, еще две точки используются для формирования направляющих ползунка. Кривошип (`crank`) имеет две ключевые точки, определяющие данный объект, одна из которых используется для задания локальной оси координат и оси вращения, другая - точка приложения шатуна. Ползунок (`slider`) имеет три ключевые точки, определяющие данный объект: точка приложения шатуна на ползунке, точка, определяющая направление скольжения и точка, определяющая связи с основанием. С учетом опций, описывающих начальные условия, функции , определяющие кривошип и ползунок запишутся в следующем виде:

```
bd[crank]=Body[crank,PointList→{{8,0,0}, {0,8,0}},
InitialGuess→{{0,10,12},{1,0,0,0}}];
bd[slider]=Body[slider,PointList→{{1,0,0},{0,1,0}},
InitialGuess→{{20,0,0},{1,0,0,0}}];
```

Для объедения элементов механической системы в единую модель механизма применяется функция `SetBodies[bodies]` (`bodies` – список элементов механизма, построенных с помощью функции `Body`). В нашем случае следует объединить основание, кривошип и ползунок.

```
SetBodies[Array[bd,3]]
```

При дальнейшем построении модели необходимо совместить характерные точки в соответствии с теми связями, которые накладываются на систему. При формировании связей, пользователь задает уникальный номер `cnum`. Общая функция `Constraint[cnum,equation,{symbol,guess}]` используется для добавления связи в виде алгебраического уравнения (`symbol` – добавляется к вектору зависимых переменных в модели с начальными условиями `guess`). Функция `Constraint` имеет столько же степеней свободы, сколько неизвестных в уравнении. В подпакете также предусмотрены некоторые часто используемые виды ограничений, генерирующие соответствующие уравнения автоматически. При описании рассматриваемого кривошипно-шатунного механизма список функций, выполняющих наложение связей на элементы механизма, имеет вид

```
cs[1]=Revolute5[1,Line[ground,1,2],Line[crank,0,1]];
cs[2]=ProjectedAngle1[2,Line[crank,0,2],
      Line[ground,0,4],Line[ground,0,3],2 N[Pi] T];
cs[3]=RelativeDistance1[3,Point[crank,2],Point[slider,0],30.0];
cs[4]=Translate5[4,Axis[ground,0,3,4],Axis[slider,0,1,2]];
```

Здесь функция `Revolute5` задает параллельность линий и совпадение характерных точек, функция `ProjectedAngle1` - равенство угла `2 N[Pi] T` и угла между первой и второй линиями на плоскости, перпендикулярной третьей линии, функция `RelativeDistance1` моделирует соединение между элементами механизма и указывает на равенство расстояния между одной из точек ползунка и одной из точек кривошипа постоянной величине, функция `Translate5` задает совпадение двух осей.

Заметим, что еще одной возможностью наложения связей является составное ограничение, которое задается функцией `StageSwitch[cnum,testi,constrainti]` передает управление `constrainti` при выполнении `testi` и содержит несколько обыкновенных ограничений `Mech`, позволяющих пользователю указывать, какое из них является активным. Здесь `constrainti` – стандартные ограничения `Mech` (или список ограничений), каждое из которых должно ограничивать одинаковое число степеней свободы.

Провести учет установленных ограничений степеней свободы и начальных условий движения механизма, также наложить реакции связей позволяет функция `SetConstraints[constraints,options]`, которая выполняет генерацию математической системы, описывающую модель механизма. Аргументы `constraints` могут быть списком или последовательностью ограничений, каждое из которых должно быть переменной или функцией. Опции `options` позволяют манипулировать начальными условиями и видом расчета. Приведем запись функции для нашего случая кривошипно-шатунного механизма.

```
SetConstraints[Array[cs,4]]
```

На следующем этапе построения модели используется функция SetLoads[loads,options], которая применяет заданные SysLoad объекты (результаты функций нагрузок) к модели. Все возможные виды нагружения могут быть представлены в виде вложенного списка или последовательности нагрузок. При успешном выполнении функция SetLoads возвращает Null.

Поскольку пакет *Mech* использует метод множителей Лагранжа для статического и динамического расчета модели, нет необходимости задавать начальные условия при отсутствии сил трения. Если начальные условия необходимы, они могут быть заданы LambdaGuess опцией. Модели механизма часто содержат множество определенных пользователем переменных, но они все должны быть инициализированы прежде, чем будет осуществлен запуск функции SolveMech. Для этого применяется функция CheckSystem[], которая проверяет модель на наличие неинициализированных переменных и избыточность ограничений, возвращает False, если найдены ошибки и выводит сообщение, описывающее ошибку.

Решатель ядра пакета *Mechanical Systems* может численно находить результат подобно встроенной функции FindRoot пакета *Mathematica*. Здесь основной функцией является SolveMech[], которая осуществляет поиск единственного числового решения, а также определяет положение и ориентацию каждого из тел. Заметим, что большинство моделей характеризуется временной зависимостью (в уравнениях по умолчанию используется символ T). Если положение и ориентация элементов механизма не зависят от времени необходимо задать начальный момент времени. В этом случае аргумент функции SolveMech может быть записан в виде SolveMech[time]. Рассчитаем положение и ориентацию элементов рассматриваемого кривошипно-шатунного механизма для момента времени равного 0,05 с.

```
SolveMech[0.05]//Chop
```

```
{T→0.05, X2→0, Y2→10., Z2→12., Eo2→0.987688, Ei2→0.156434,
Ej2→0, Ek2→0, X3→19.5064, Y3→0, Z3→0, Eo3→1., Ei3→0,
Ej3→0,Ek3→0}
```

Здесь Eo, Ei, Ej, Ek – обобщенные углы Эйлера, Xn, Yn, Zn – координаты центра тяжести элементов механизма.

В случае, если интерес представляют числовые характеристики отдельных элементов механизма, целесообразно использовать функцию SetCouple[symbol,equation, options]. Эта функция возвращает объект CoupleSystem, который определяет значение symbol, удовлетворяющее equation и всем ограничениям степеней свободы. При этом генерация всех необходимых начальных условий и моментов времени осуществляется функцией LastSolve[]. На заключительном этапе используется функция SolveCouple[cpsys], которая рассчитывает решение для объекта cpsys, являющегося результатом функции CoupleSystem, и возвращает список, содержащий решение.

ЛИТЕРАТУРА

1. Дьяконов, В. П. *Mathematica* 4 с пакетами расширений. – М.: Нолидж, 2000. – 605 с.
2. Beretta R. Mechanical Systems Pack. – Vancouver: Wolfram Research, 1995. – Pp. 525.