УДК 629.78

СВОБОДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ МОДЕЛИРОВАНИЯ ДИНАМИКИ СИСТЕМ ТВЕРДЫХ ТЕЛ

к .т. н., с. н. с. Белоножко П. П., к .т. н., с. н. с. Храмов Д. А.
Институт технической механики НАН Украины и ГКА Украины, г. Днепропетровск

Возрастающие требования к качеству моделирования динамики систем твердых тел заставляют создавать все более подробные математические модели. Такие модели, как правило, громоздки, а их разработка, компьютерная реализация и верификация представляют собой серьезную проблему. С другой стороны, в настоящее время существует множество программных средств и подходов, позволяющих упростить эту работу, сделать ее более эффективной. В связи с этим возникает необходимость в анализе имеющегося арсенала программных средств моделирования динамики систем твердых тел и выработке рекомендаций по их рациональному использованию.

В настоящей работе мы ограничимся анализом свободного программного обеспечения с открытым исходным кодом. Использование подобного программного обеспечения представляется особенно важным в научных исследованиях, так как делает его код, являющийся составной частью работы, доступным для независимой проверки.

Программные средства моделирования динамики систем твердых тел можно разделить на четыре группы: языки программирования и математические библиотеки, системы символьного моделирования, пакеты физического моделирования, библиотеки интерактивного моделирования. Рассмотрим их более подробно.

Совместное использование языков программирования и математических библиотек дает максимальную гибкость, возможность точной настройки на решаемую задачу: в языках программирования нет специальных средств для описания динамики систем твердых тел — их нужно создавать самостоятельно. Вместе с тем, такая удаленность от языка предметной области затрудняет создание расчетных программ, и требует высокой квалификации разработчика.

В этой связи может оказаться удобным использование интерпретируемых языков, в частности, Python, Scilab и GNU Octave.

Они проще в освоении по сравнению с компилируемыми языками, что частично снимает трудности разработки. Что же касается скорости выполнения программы, то здесь на первый план выступает качество кода специализированной библиотеки и возможность ее настройки на аппаратуру конкретного компьютера. Например, обращение матриц, реализованное на «чистом» C может выполняться медленнее, чем на Python с помощью библиотеки NumPy. Список доступных математических библиотек приведен в [1].

Такой подход удобно использовать для выполнения разовых расчетов, когда необходима быстрая оценка состояния системы, либо для создания узкоспециализированных пакетов.

Системы символьного моделирования (computer algebra systems) предназначены, в первую очередь, для преобразования формул. Работа с такими системами имеет ряд особенностей.

Так, применение систем символьного моделирования (Axiom, Maxima, Giac, Sage) может начаться раньше, чем станет возможным традиционное программирование, — еще на этапе записи уравнений движения в векторном виде. При традиционном программировании, последовательность выполнения команд, реализующих математические выражения, диктуется логикой вычислений: чтобы получить численное значение выражения, необходимо вычислить значения всех входящих в него величин. В системе символьного моделирования можно реализовать другой подход, близкий к работе исследователя «с карандашом и бумагой»: сначала записываются наиболее общие уравнения, затем входящие в них величины расписываются подробно, делаются оценки и упрощения, результаты которых подставляются в исходные уравнения. Можно сказать, что системы символьного моделирования позволяют выполнять расчеты «от общего к частному», тогда как численное моделирование выполняется «от частного к общему».

Вывод уравнений движения в матричном виде и приведение их к форме, удобной для численного интегрирования, осуществляются с помощью процедур на языке системы символьного моделирования. Время, затраченное на разработку этих процедур, окупается при проведении многократных расчетов, требующих внесения изменений в модель.

Некоторые системы символьного моделирования позволяют конвертировать полученные с их помощью уравнения в программный код на языках программирования C, или Fortran. Это делает привлекательным совместное использование систем символьного и численного моделирования.

В **пакетах физического моделирования** (MBDyn, Scicos, OpenModelica, Jmodelica) пользователь работает не с уравнениями, а с механической схемой системы, составляя ее из объектов, представляющих тела, шарниры, силы и т. п.

Значительная часть этих пакетов основана на языке программирования Modelica. Modelica представляет собой объектно-ориентированный язык, призванный стать стандартом описания мультидоменных систем, т. е. систем, элементы которых относятся к разным областям науки. Это позволяет исследовать «целиком» такие объекты как, например, электродвигатель, состоящий из электрических и механических элементов.

В качестве примера на рис. 1 показано описание плоского математического маятника на языке Modelica.

```
class Pendulum
  constant Real PI=3.141592653589793;
  parameter Real m=1, g=9.81, L=0.5;
  Real F;
  output Real x(start=0.5),y(start=0);
  output Real vx,vy;
equation
  m*der(vx)=-(x/L)*F;
  m*der(vy)=-(y/L)*F-m*g;
  der(x)=vx;
  der(y)=vy;
  x^2+y^2=L^2;
end Pendulum;
```

Рис. 1. Модель математического маятника, записанная на языке Modelica [2]

Наряду с текстовой записью базовых классов Modelica, стандартизировано и их графическое представление (рис. 2). Поэтому, разобравшись с одним пакетом, использующим Modelica, другой можно освоить по аналогии.

Особое внимание при использовании пакетов физического моделировании следует уделять составлению механической схемы исследуемой системы, поскольку вывод уравнений движения и все дальнейшее изучение системы осуществляется на ее основе. Цена ошибки здесь высока, и хотя пакет отсеивает случаи, когда уравнения движения составить невозможно (проверяет синтаксические ошибки, если воспользоваться аналогией с трансляторами), но можно получить уравнения, описывающие поведение системы, отличной от задуманной (т. е. совершить логическую ошибку). Снизить риск появления

подобных ошибок можно, если формировать систему постепенно, небольшими обозримыми блоками, каждый из которых будет доступен непосредственной проверке.

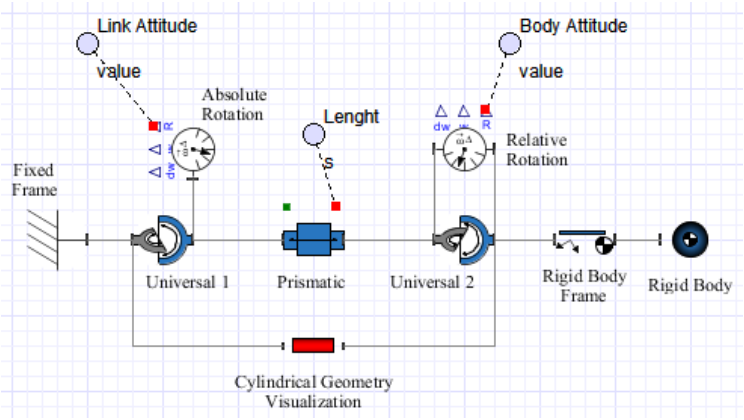

Рис. 2. Графическое представление в Modelica схемы системы, состоящей из твердого тела, подвешенного на невесомой упругой нити

За приближение к языку предметной области пакетам физического моделирования приходится расплачиваться утратой некоторых возможностей. В частности, с их помощью затруднительно моделировать системы, структура которых изменяется в ходе расчета.

Библиотеки интерактивного моделирования динамики (Bullet, Open Dynamics Engine, Newton Game Dynamics, Siconos), называемые иначе «физическими движками» (physics engines), дают пользователю возможность создавать такие объекты как «частица», «тело», «связь» и управлять их характеристиками. В отличие от пакетов физического моделирования, физические движки позволяют изменять характеристики движения, добавлять и удалять тела непосредственно в ходе расчета [3].

Моделирование с помощью физического движка начинается с создания виртуального «мира», в который помещаются тела, с заданными характеристиками движения, массой и формой. Далее указываются действующие силы и моменты, а также связи между телами. По этой информации физический движок строит уравнения движения рассматриваемой системы.

Расчет движения начинается с обнаружения возможных контактов

(столкновений) между телами. На основе этой информации вычисляются скорости тел после столкновения, и корректируются их положения во избежание проникновения друг в друга. Вычисление новых координат и скоростей тел выполняется с помощью численного интегрирования уравнений движения, для чего, как правило, используются методы Эйлера-Кромера и Верле.

Полученные данные о состоянии тел передаются средству отображения (рендереру), которое обычно не входит в состав движка.

Физические движки применяются для создания программ, позволяющих взаимодействовать с пользователем, изменять структуру системы и получать результаты расчетов в реальном времени. При этом обычно жертвуют точностью расчетов ради скорости их выполнения. В частности, поэтому во многих движках используются методы численного интегрирования не выше 2-го порядка. Тем не менее, для свободного программного обеспечения этот недостаток исправляется подключением дополнительных математических библиотек.

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

1. List of numerical libraries [Электронный ресурс]. — Режим доступа : en.wikipedia.org/wiki/List_of_numerical_libraries.
2. Fritzson P. Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica / P. Fritzson. — New York : Wiley, 2011. — 211 p.
3. Millington I. Game Physics Engine Development / I. Millington. — New York : Morgan Kaufmann Publishers, 2010. — 553 p.