Проект

Моделирование падения сложной фигуры

Д. Курлов

2018 год

Содержание

1. Введение

1.1. Моделирование

Моделирование является важнейшим и неотъемлемым этапом процедуры проектирования современных механических устройств и систем. Термин "моделирование- весьма многогранен. Применнительно к техническим(в т.ч. механическим) системам под **моделированием**, будем понимать процесс, состоящий в выявлении основных свойств исследуемого объекта, построении моделей и их применении для прогнозирования поведения объекта. Математические модели в механики, как правило, представляются в виде систем дифференциальных уравнений

1.2. Методы решения задачи моделирования

Для решения задачи математического моделирования в общем случае могут использоваться следующие методы:

- Аналитический методы;
- Численные методы;

Аналитический метод. Для решения аналитическим методом необходимо явное интегрирование дифференциального уравнения, что на практике может быть весьма сложной или вовсе невыполнеимой задачей.

Численные методы. Существует множество алгоритмов численного решения дифференциальных уравнений, которые позволяют избавиться от проблемы сложного интегрирования и решают данную задачу с необходимой точностью.

В проекте я буду использовать численные методы, чтобы избежать сложного интегрирования.

2. Постановка задачи

Входные данные представляют собой множество точек на плоскости и матрицу связей.Точки представляют собо шарниры, которые соединяют ребра фигуры. Матрица связей представляет собой матрицу смежности, которая отражает, какие из точек соеденинены ребрами. По этим данным составляется некоторая геометрическая фигура. Для некоторого упрощения будем считать, что наша фигура представляет собой остовное дерево (т.е. в нашей фигуре нет циклов).

Необходимо смоделировать процесс падения нашей фигуры по законам механики.

3. Теория

3.1. Уравнения Лагранжа

Для моделирования падния фигуры я составил уравнения лагранжа второго рода. В общем случае уравнение выглядит так:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0, \qquad i = 1, \ldots, n; \tag{1}$$

q_i — обощенные координаты, а L — кинетический потенциал (или функция Лагранжа). $L = T - P$ где T — кинетическая энергия, P — потенцияальная энергия, g — ускорение свободного падения.

$$T = \sum_i \frac{1}{2} m_i V_i^2 + \sum_i \frac{1}{2} \omega_i^2 I_i. \tag{2}$$

$$P = \sum_i m_i g_i h_i. \tag{3}$$

В уравнениях (2) и (3) m_i — массы i-тых ребер, V_i — скорости i-тых точек, ω_i — угловые скорости точек, I_i — моменты инерции (которые вычисляются по формуле: $I_i = \frac{1}{1} m_i l_i^2$, где l — длина стержня между точками).

3.2. Конкретный случай

Для простоты рассказа будем использовать такой случай: на вход получено четыре точки, которые соединены тремя точками. На вход подаются координаты точек A, B,

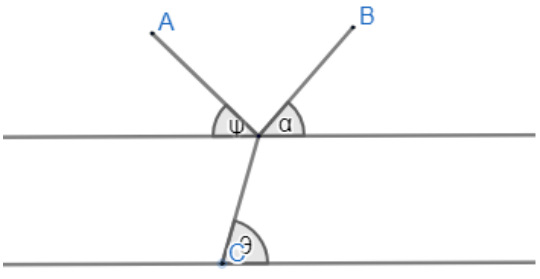

Рис. 1: Типовая фигура

C, O, а также матрица связей (она же матрица весов ребер). В данном случае матрица

выглядит таким образовм:

$$edges = \begin{array}{c} \\ A \\ B \\ C \\ O \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & O \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Уравнения Лагранжа в данном случае будут выглядеть так:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0 \tag{4}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right) - \frac{\partial L}{\partial \phi} = 0 \tag{5}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\psi}}\right) - \frac{\partial L}{\partial \psi} = 0 \tag{6}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = 0 \tag{7}$$

После подстановки подстановки всех значений в формулу, вычисления частных производных и упрощения, запишем систему (4) в матричном виде (левая часть в дальнейшем — **матрица A**, правая часть — **вектор b**):

$$\frac{d}{dt}\begin{pmatrix} l_c^2\left(m_a+m_b\right)+I_c & 0 & 0 & l_c\left(m_a+m_b\right) \\ 0 & I_B & 0 & 0 \\ 0 & 0 & I_A & 0 \\ l_c\left(m_a+m_b\right) & 0 & 0 & m_a+m_b+m_c \end{pmatrix} = -\begin{pmatrix} \cos\theta\left(m_a g l_c + m_b g l_c + m_c g \frac{l_c}{2}\right) \\ m_b g \frac{l_b}{2}\cos\phi \\ m_a g \frac{l_a}{2}\cos\psi \\ 0 \end{pmatrix}$$

3.3. Приближенное вычисление положения точек

В общем случае, за каждый шаг мы приближенно вычисляем приращения с помощью системы (4)-(7):

$$A^{-1}b = \Delta\dot{q} \tag{8}$$

Затем пересчитывается вектор скоростей:

$$\vec{V} = \vec{V} + \Delta\dot{q} \tag{9}$$

И, наконец, по вектору скоростей обновляются координаты точек. Осуществляются необходимые повороты и переносы точек.

4. Реализация

4.1. Инициализация

В основном файле — **MechanicProject.cpp** реализован ввод данных. Пользователь вводит набор координат точек в формате (x,y) и заполняет матрицу весов, создаются необходимые объекты классов:

- Объект класса *Scene* (грубо говоря, уровень земли в окне `OpenGl`)
- Объект класса *RigidBody*, который хранит в себе данные о нашей фигуре - количество точек, их координаты, матрицу связей (весов)
- Объект класса *Physical*, который отвечает за пересчет положения точек тела

Для хранения данных об отдельной точке был создан класс *Point*. Также в данном классе реализована необходимая арфметика (перегружены операторы *, +, - и =, а также реализована функция поворота одной точки вокруг другой `Point::Rotate`) Особым, в смысле инициализации, является объект класса *ModelSimulator*. Данный класс отвечает за активацию отрисовки объектов на сцене и на его метод *render* должна ссылаться функция отрисовки *Draw* OpenGl. Но *Draw* не может принимать что-либо в качестве аргумента, поэтому создается глобальный указатель на объект этого класса.
После инициализаци данных производится инициализация окна **OpenGl**

4.2. Иерархия классов и описание методов

Для отрисовки я создал абстрактный базовый класс *SceneObject*, от которого отнасл-довались два класса

- *Scene*
- *RigidBody*

SceneObject содержит только виртуальную функцию *render*. Класс *Scene* отвечает только за отрисовку "уровня земли"в окне. В перспективе можно добавить другие объекты, с которыми будет взаимодействовать наша фигура.

Класс *RigidBody*, как было уже сказано ранее, содержит в себе данные о фигуре. А также составляет массивы весов и длин ребер (по координатам точек и матрице связей), отрисывавает наше тело по положению точек.

Особый интерес представляют классы *ModelSimulator* и *Physical*. Класс *ModelSimulator* отвечает за обновление среды. То есть вызывает пересчет координат в методе `ModelSimulat`
а после вызывает функцию отрисовки тела `RigidBody::render` и сцены `Scene::render`.

Класс *Physical* отвечает за математическую модель нашего физического процесса (падения фигуры), а если проще, то за пересчет координат точек нашей фигуры. На каждом шаге после проведения всех необходимых вычислений указанных в **разделе 3** производятся повороты и смещения точек относительно друг друга с помощью перегруженных операторов арифметических операций класса *Point*, а также его метода `RigidBody::rotate`.

5. Анализ

5.1. Тестирование

Рассматривались различные начальные положения системы (смещение влево, смещение вправо, вниз и т.д). Падение моделируется достаточно приближенно к реальности. Так как изначально не было никаких внешних сил, связи предполагались идеальные. При таком подходе выбранный метод приближенного решения системы приводил к слишком большому накапливанию ошибок. То есть, если программа была запущена в течение длительного времени, накапливалась большая погрешность при пересчете координат и модель начинала выглядеть не естественной.

5.2. Анализ результатов

Вышеуказанные неточности я решил с помощбю добавления внешних сил — сил трения. В таком случае фигура, в конечном итоге, принимала положение покоя(равновесия).

5.3. Вывод

Данная модель является очень гибкой. Ее можно использовать для моделирования при разных внешних воздействиях или, например, для моделирования падения на луне (изменив силу тяжести).

Для модели с идеальными связями необходимо использовать более точные методы

5.4. Что можно добавить/изменить

Как было сказано, можно изменить метод приближенного вычисления на более точный.

Доконца реализовать различное управление (например принятие заданного положения и его поддержание с помощью регулируемых сил)

6. Ссылки на источники

1. https://compgraphics.info — общее руководство по **OpenGl**
2. https://triplepointfive.github.io — руководство по анимации в **OpenGl**
3. https://ru.wikipedia.org — статья об уравнениях Лагранжа
4. link.pdf — полезна статья о применениях уравнения Лагранжа при решении прикладных задач
5. Михаил — основной куратор и консультант.