Методическое пособие для первокурсников ФФПФ МФТИ
по курсу «Проектная деятельность»
для проекта «Моделирование движения заряженных частиц в магнитном и электрическом полях»

Методы решения систем ДУ с помощью языка программирования Matlab

Март 2019

Содержание

Вашим самым близким другом при работе с Matlab должен стать раздел Help, для поиска по которому выделите ваш запрос и нажмите кнопку F1. В хэлпе максимально ёмко описаны все команды и приведены примеры их использования. Также настоятельно рекомендую попутно с чтением методички держать открытой программу и вбивать туда всё, что вы видите в методичке, чтобы самостоятельно прочувствовать результат.

Часть I
Теоретическая часть

1. Дифференциальные уравнения

В этом проекте мы с вами будем учиться численно решать систему дифференциальных уравнений в рамках задачи о движении частицы в магнитном и электрическом полях (моделировать движение частицы). Самый первый вопрос здесь: что же такое дифференциальное уравнение?

Дифференциальное уравнение n-ного порядка – это уравнение вида

$$F(x,y,y',...,y^{(n)}) = 0.$$

Решением дифференциального уравнения называется функция $y = \phi(x)$ такая, что

$$\exists\ \phi'(x),...,\phi^{(n)} \text{ и при этом } F(x,\phi,\phi',...,\phi^{(n)}) = 0.$$

Зачем мы вообще говорим здесь о дифференциальных уравнениях? Всем известно, что движение тел описывается с помощью второго закона Ньютона, который как раз представляет из себя дифференциальное уравнение:

$$m\frac{d^2\mathbf{x}}{d\mathbf{x^2}} = F(t,\mathbf{x},\dot{\mathbf{x}}).$$

Надеюсь, что вам известно то, что дифференциальные уравнения, в принципе, трудно-решаемы на практике за исключением частных случаев. Именно поэтому их и решают численно.

2. Численные методы решения систем ДУ

Здесь я не буду распинаться на счёт всей глубокой теории вычислительной математики, у вас это будет на 3 курсе, а если не терпится узнать всё сейчас, то можно почитать методичку *Практические занятия по вычислительной математике : учебное пособие / Е.Н. Аристова, А.И. Лобанов*, которая лежит на сайте кафедры вычислительной математики. Я же постараюсь дать максимально полезную выжимку по аппарату, который нам понадобится освоить.

Для численного решения мы с вами будем применять метод Рунге-Кутты. Он применяется для решения задачи Коши (ДУ + начальное условие).

Но для начала, перед написанием и отладкой метода Рунге-Кутты, применим решатель ДУ, который уже вшит в MATLAB. Рассмотрим максимально простую модельную задачку, чтобы на её примере понять, с чем нам предстоит иметь дело.

Всем знакомое со школы уравнение движения математического маятника:

$$\ddot{x} + \omega^2 \sin x = 0.$$

Также со школы всем знаком факт того, что оно прекрасно решается аналитически в пределе $\sin x \approx x$. Дальше – беда. Вот именно для таких случаев и существуют численные методы решения различных уравнений. Естественно, что решение в таком случае мы получаем с какой-то точностью, которая зависит от метода и получаем мы всё же решение математической задачи, а не физической. Поэтому часто стоит делать проверку на физичность решений, но об этом можно долго вести различные разговоры, пока оставим это.

Что же касается решателя ДУ в языке MATLAB, то мы с вами посмотрим на функцию `ode45`, которая максимально часто используется для таких целей. Если заглянуть в прекрасный файл MATLABREF.PDF, можно заметить, что эта функция там описана как "решение нежесткой задачи Коши". Что такое задача Коши уже было упомянуто, что же такое нежесткая?

В вычислительных методах есть две проблемы: ~~дураки и дороги~~ корректность (существование и единственность решения задачи Коши) и обусловленность. Представим, что у нас есть некое дифференциальное уравнение + начальное условие. И вот мы получили решение нашей задачи Коши. А теперь возьмём и решим то же ДУ, но с немного другим начальным условием. Допустим, сделаем его больше на 0.1%. Может получиться так, что наше решение резко скаканёт на несколько порядков, чего, в хорошем случае, конечно, быть не должно. Это пример плохой обусловленности. То есть хорошая обусловленность задачи – это когда малые изменения начального условия приводят к малым изменениям в нашем решении (интегральных кривых). Такие задачи, к великой грусти, встречаются, и с ними пытаются совладать.

Так что же в итоге такое жесткая задача? Жесткая задача – это устойчивая корректно поставленная задача, но плохо обусловленная. Для решения таких задач явные методы решения не годятся, и в игру вступают неявны. Теорию, связанную с методами решения таких задач, вы можете посмотреть всё в той же методичке или постараться доучиться до третьего курса, чтобы пройти это там. Сейчас нас будет интересовать непосредственное применение этих методов для решения нашего ДУ. Нежесткая задача – это задача, у которой всё хорошо с обусловленностью, с такими задачками приятно работать, их можно бесконечно долго хвалить и так далее.

Движемся дальше – что за загадочные цифры `45`? Эти цифры означают порядок точности. В `ode45` зашит метод Рунге-Кутты 4 порядка (который мы дальше с вами напишем сами). Ошибка на одном шаге – это $\mathcal{O}(h^5)$, ошибка на конечном интервале интегрирования – $\mathcal{O}(h^4)$, где h – шаг по x.

Часть II
Практическая часть

3. Решение уравнения движения математического маятника (модельная задача)

Синтаксис функции `ode45` подразумевает, что под скобки нужно внести ссылку на соотсвествующую функцию, в которую забито уравнение, которое мы будем решать. Для создания таковой функции кликаем правой кнопкой в окне **Current Folder** и выбираем **New file** > **Function**. Создаётся такой же файл разрешения .M, как мы уже привыкли, но внутри он выглядит как:

```
function [ output_args ] = Untitled( input_args )
% UNTITLED Summary of this function goes here
% Detailed explanation goes here

end
```

Соотвественно мы понимаем, что нам здесь нужно задать не только саму функцию в теле кода, но и название, входные аргументы и выходные аргументы в шапке функции. Заполняем её:

```
function [ dr ] = ex_funct( t,r )
global w;
dr = zeros(2,1);
x = r(1); vx = r(2);
dr(1) = vx;
dr(2) = -w*w*sin(x);
end
```

Важно!! Файл с функцией должен называться так же, как функция!! Далее всё это решается достаточно просто: нужно всего лишь написать:

```
[T,R] = ode45(@ex_funct,t_span,r_start);
```

Теперь по порядку: `@ex_funct` – функция, с которой мы работаем; `t_span` – временной промежуток, на котором мы работаем, задаётся строчкой двух чисел; `r_start` – строчка начальных значений (здесь координаты и скорости). В результате выполнения команды получаем два массива: массив времени и массив координаты.

Далее рассмотрим реальную задачу с реальными физическими величинами и там уже разберём результаты более детально.

4. Решение уравнения движения математического маятника (реальная задача)

Рассмотрим математический маятник с длинной ниточки 5 м. Ускорение свободного падения будем считать 10 м/с^2. Для решения реальных задач обычно обезразмеривают уравнения, чтобы в рассчёте не оказалось слишком больших или слишком маленьких чисел (компьютер не любит ни те ни другие, и в результате вылезают никому не нужные ошибки). У нас с вами есть переменные координаты, времени и неявно скорости. Введём характерные масштабы угловой координаты, времени и угловой скорости скорости:

$$ch\_x = \pi/2; \quad ch\_t = 2\pi\sqrt{L/g}; \quad ch\_vx = ch\_x/ch\_t.$$

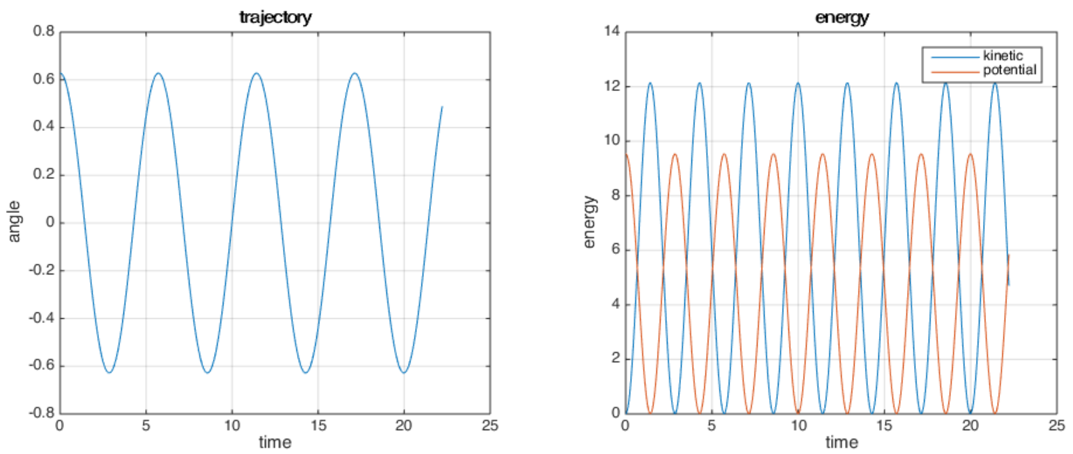
Тогда с учётом этого наше уравнение движения перепишется как:

$$\frac{(ch\_x)^2}{(ch\_t)^2}\frac{d^2\hat{x}}{d\hat{t}^2} = -\omega^2 \sin{(\hat{x} \cdot ch\_x)},$$

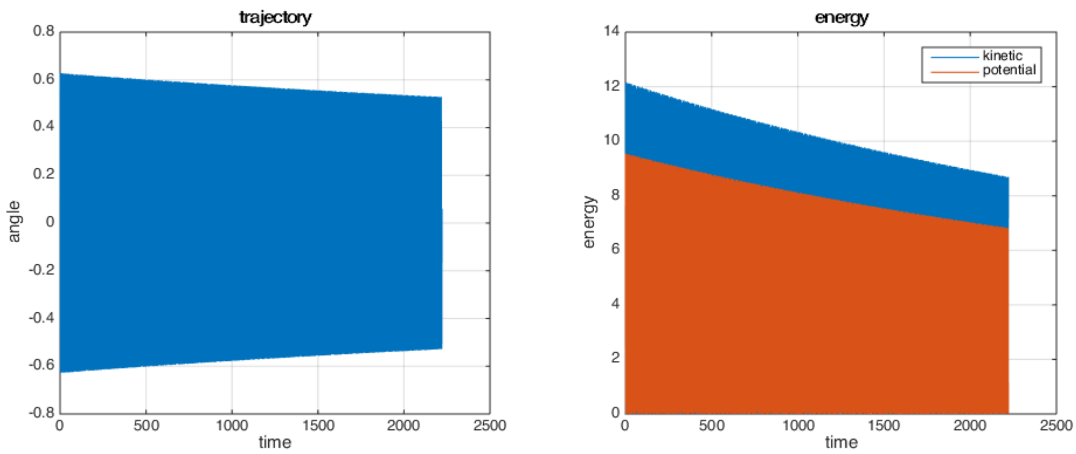
где $\hat{x}$ и $\hat{t}$ – обезразмеренные величины. Вытаскивать константу из-под синуса затруднительно (я пока не уверена, что знаю, как это хорошо сделать), поэтому пока не совсем принципиально, оставим её там. В итоге получаем уравнения

$$\frac{d\hat{x}}{d\hat{t}} = \hat{vx},$$
$$\frac{d^2\hat{x}}{d\hat{t}^2} = \frac{(ch\_t)^2}{(ch\_x)^2}\omega^2 \sin(\hat{x} \cdot ch\_x);$$

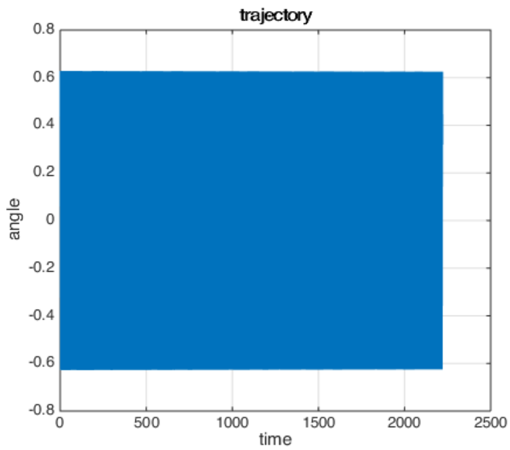
описывающие движение нашего маятника. Ниже приведёны графики угловой координаты (уже размерной) и кинетической энергии (квадрата размерной обычной не угловой скорости) с потенциальной энергией (произведение ускорения свободного падения на координату по игреку):

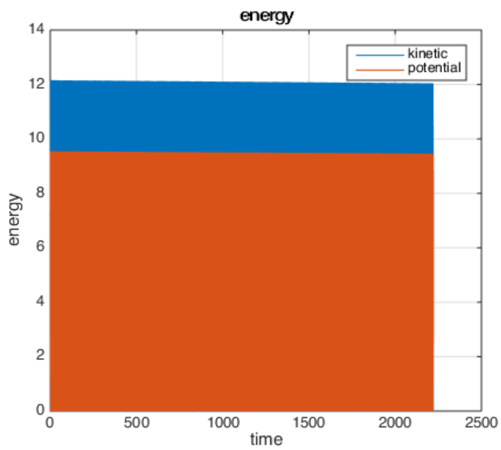


Если увеличить время, картина будет такой:



Как видно, энергия на самом деле падает и движение затухает. Это связано с потерями как алгоритма, так и ошибками вычислений. Но мы можем прописать другую точность вычислений нашего решения в опциях решателя `ode45` и получить картиночку, которая находится на следющей странице.





Код этой штуки лежит в папке на гитхабе (функция + скрипт), там написаны комментарии, с кодом можно поиграться. Там же есть отдельный блок с анимацией движения маятника, но не запускайте этот блок, если у вас большое время вычисления (если всё же запустили, можно прервать выполнение скрипта из **Command Window** сочетанием клавиш **Ctrl** + **C**). Постарайтесь максимально разобраться в этом коде, потому что примерно такая же штука от вас требуется далее.

Часть III

Задание для самостоятельной работы

Самостоятельная работа заключается в написании програмы по расчёту уравнений движения частиц в магнитном и электических полях в объёмном случае.
Уравнение движения:

$$M\frac{d^2\mathbf{X}}{dt^2} = \frac{e}{c}[\mathbf{V} \times \mathbf{B}] + \mathbf{eE},$$

где $\mathbf{X} = [x,y,z]$, $\mathbf{V} = [v_x,v_y,v_z]$, $\mathbf{B} = [B_x,B_y,B_z]$, $\mathbf{E} = [E_x,E_y,E_z]$ и $M = qm$, m – масса протона.

Ввести безразмерные величины, где возможно в зависимости от задачи:

1. Тестовая задачка: движение протона в электрическом поле (вдоль и поперёк);

2. Движение частиц в скрещенных магнитном и электрическом (слабом) постоянных полях, рассмотреть несколько случаев движения. Частицы: электрон, протон и альфа-частицы (программа должна быть для всех частиц одновременно);

3. Случай движения частицы в магнитном поле диполя (*по желанию*).

Для каждого случая нарисоват (и чут-чут подписат) график кинетической энергии частицы.