Курсовой проект: рекламный сервер

Общие правила

В конце второго модуля должен быть сдан отчёт и итоговая программа с реализованными численными методами. Каждое задание сдаётся до дедлайна ассистенту. За 1 просроченный дедлайн максимальная итоговая оценка за курсовой проект уменьшается на 10% (всего будет около 4-5 дедлайнов), а с невыполненным заданием 0 проект не принимается с оценкой 0.

Промежуточные дедлайны сделаны для того, чтобы можно было выполнять проект постепенно, не откладывая всё на последний момент.

Описание курсового проекта в файле Threshold_Dynamics.pdf .

Форма сдачи

Итоговая форма сдача проекта – отчёт (допускается в электронном виде, окончательный его план будет прислан позднее) и программный продукт. Сдавать каждое задание нужно М. Каледину, 4й курс 132ПМИ. Контакты (в порядке предпочтительности):

- VK: `https://vk.com/id61870705`
- Telegram: `https://telegram.me/XuMuK_MK`
- Email: `xumuk.unity.dev@gmail.com`

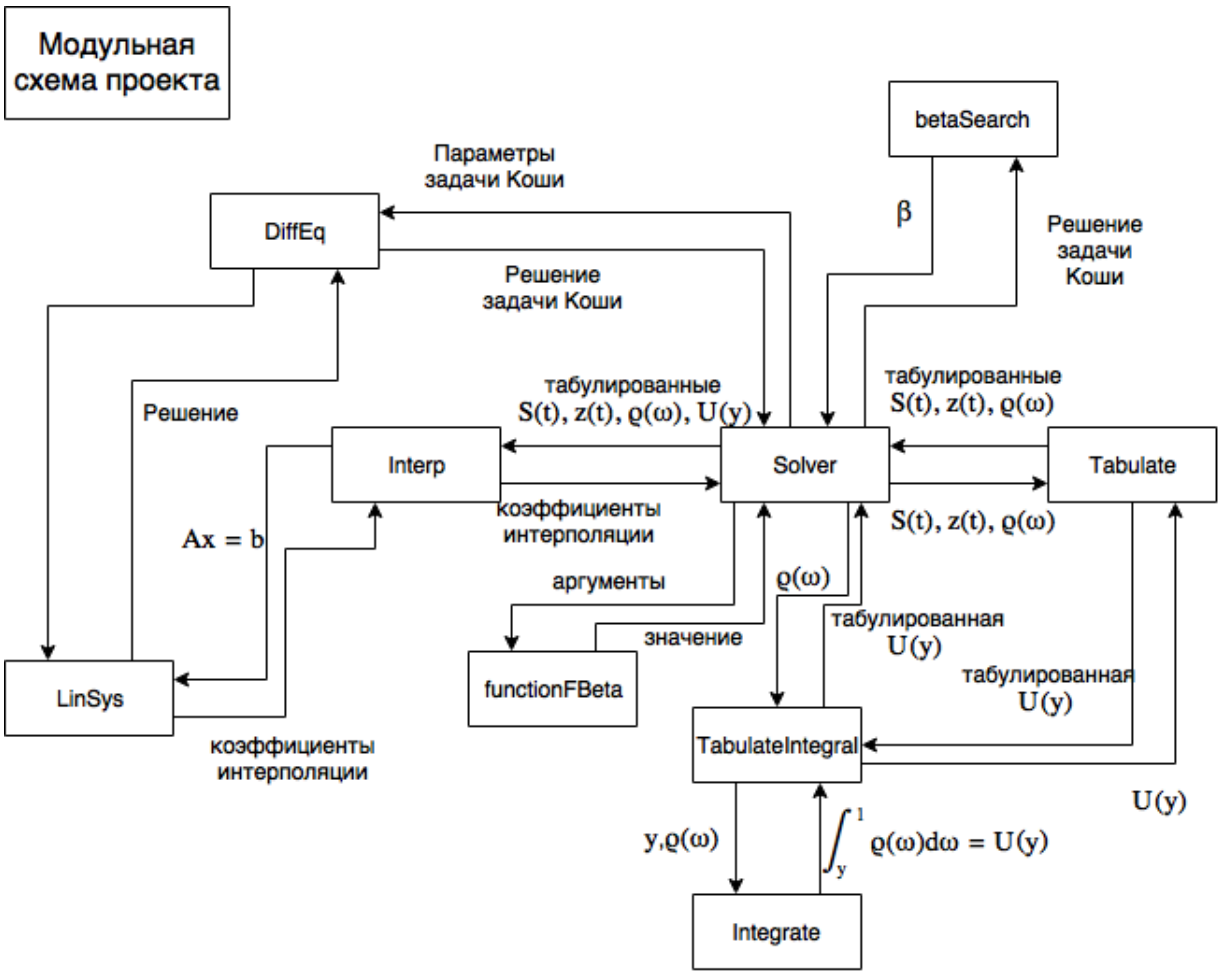
Задание 0

Дедлайн: 6 октября 2016

Прежде, чем приступать к работе над проектом нужно определиться с используемыми технологиями.

1. *Язык программирования* на ваш выбор;
2. *Графика*. Потребуется рисовать графики функций и, возможно, другую полезную графическую информацию;
3. *Пользовательский интерфейс*. Нужно сделать не только работающий, но ещё и приятный для использования продукт. Командная строка не подойдёт.

Опишите все модули и сигнатуры их функций. Приблизительная схема проекта:



Спроектируйте интерфейс всего продукта и реализуйте его при выбранных технологиях. Должны быть доступны все модули, чтобы можно было проверить работоспособность интерфейсной части без смысловой нагрузки, т.е. без модулей с численными методами. При этом вычислительные функции должны быть, но с пустым наполнением. Например, функция численного интегрирования вместо результата должна печатать сообщение, что она сработала. Что обязательно должно присутствовать в итоговой версии (пока лишь в черновой):

- *Графики*: $\rho(\omega), x(t), S(t), z(t), x(t)-S(t), y(t)$, траектория $S(x)$, а также значения критериев $C_1(\beta), C_2(\beta)$ (в виде изолиний или изоповерхностей, если вы используете несколько параметров β)
- *Формы ввода*: для ввода параметров задачи, пути к файлам и т.д.
- *Два режима запуска*: ручной и автоматический, подробнее см. описание в Threshold_Dynamics.pdf

Реализуйте несколько *use-case* для быстрой проверки работоспособности (формат для чтение и записи в файл продумайте сами):

1. *Задание функций* $\rho(\omega), z(t), S(t)$.
   - Ввод нескольких параметров, определяющих вид функций $\rho(\omega), z(t), S(t)$. Например: $\rho(\omega) = a\omega(b-\omega)$ требует ввода параметров a, b.
   - Запись табулированных функций в файл
2. *Табулирование интеграла* $\int_y^1 \rho(\omega) = U(y)$.
   - Чтение из файла $\rho(\omega)$ (результат use-case 1)
   - Вычислить интеграл в точках сетки (ЗДЕСЬ НЕ НАДО ВЫЧИСЛЯТЬ, просто имитируйте процесс, возвращая значения интеграла)
   - Вычислить коэффициенты интерполяции функции $U(y)$ (ТОЖЕ ПОКА лишь имитация вычислений, реальная интерполяция в задании 1), записать их в файл

3. *Постановка и решение задачи Коши*.

   - Ввод параметров задачи (x_0, y_0, β, T), а также функций $U(y), S(t), z(t)$ (в виде интерполяции с коэфициентами, здесь пока можете лишь считать (как бы) коэффициенты из файла, а сами функции пока что прописать явно в коде),
   - Организация цикла по сетке времени $t_k{}_{k=1}^{N}$ для решения задачи Коши, для примера внутри цикла вычисляйте x_k, y_k согласно явной схеме Эйлера (простейший приближённый метод решения задачи Коши, в оригинальном уравнении заменяете $dy/dt, dx/dt$ на соответственно $(y_k - y_{k-1})/\tau_k$ и $(x_k - x_{k-1})/\tau_k$, где $\tau_k = t_k - t_{k-1}$). В расчётах используйте интерполяции функций $U(y), S(t), z(t)$ (КАК БЫ, можете на текущий момент железно забить $U(y), S(t), z(t)$ в код).
   - Решение задачи Коши запишите в файл.

В функциях, где возможен некорректный ввод данных (скажем, пользователь вводит текст вместо числа), необходима проверка некорректного ввода. Если в вашей задумке предполагается работа с файлами, то обязательно проверяйте наличие ошибок (не найден файл и пр.).

По ходу курса, когда будут выдаваться задания реализовать конкретный модуль, вместо бессмысленных пустых функций будут появляться полновесные численные методы. Это первое домашнее задание необходимо, чтобы исключить ошибки с интерфейсной частью в дальнейшем.

На текущей стадии должно быть готово: список используемых технологий и программных пакетов, подробное описание программных модулей, реализованный интерфейс. Возможно, начатый отчёт. Сдаётся устно ассистенту, он же будет тестировать интерфейс.

Задание 1

Пока возможны небольшие изменения в формулировке относительно тестирования.

Дедлайн: 23 октября 2016

Напишите программу для модуля табуляции функции. Вход: сетка аргументов; выход: значения функции в узлах.

Составьте модуль интерполяции. Подробно опишите метод, протестируйте его для интерполяции по узлам трёх различных функций (гладких, разрывных, осциллирующих), привести сравнительные графики с изображенной истинной функцией и интерполяцией по узлам. Получите графики экспериментальной ошибки $E(x) = L[f](x) - f(x)$ и сравните её (с помощью графика) с теоретической оценкой для выбранного метода интерполяции.

Запрограммируйте модуль численного интегрирования (любым доступным методом: формулы Ньютона-Котеса, Гаусса и т.д.). В отчёте укажите простую формулу, сетку узлов и составную формулу. Оцените аналитически порядок точности.

Проведите тестирование на трёх различных функциях (гладких, разрывных, осциллирующих), сравните численно полученный результат с точным (либо посчитанным численно с помощью стороннего мат. пакета и более точной формулой, чем у вас, стоит посмотреть Matlab или Python sci-kit).

Проверьте порядок точности экспериментально, вычислив численно интеграл от одной из функций при уменьшающемся шаге сетки h (и, соответственно, большем числе узлов) и нарисовав график погрешности (при сравнении с более точным решением) в осях X-LogY (просто возьмите логарифм от Y). С помощью графиков и таблиц прокомментируйте результаты.

Графики и вычисления нужно делать в отдельном тестирующем модуле, который включается в главный проект в качестве *use-case* и использует функции из проекта (за исключением сторонних для тестирования точности, их можно не подключать к проекту, просто используйте полученный оттуда ответ).

На текущей стадии должно быть готово: описание алгоритмов, требуемые выше обоснования и иллюстрации, модуль для тестирования задания 1, программа, совмещённая с уже написанным ранее интерфейсом. Готовое приписывается к отчёту.