Лабораторная работа №2. Интерполяция в MatLab

1. Запускаем приложение *splinetool*	`>> splinetool`

2а. Интерполяция табличных функций

Создаем массивы со значениями табличной функции	`>> x = [-4 -1.7 0 0.2 0.7 0.9 1.1];` `>> y=[-1.66 -1.36 0 0.42 1.26 1.97 2.24];`

Кнопка *Import your own data*, в окне *Please provide data* указываем `x` и `y`.

2б. Интерполяция непрерывных функций

Создаем массив равномерно распределенных точек на **СВОЕМ** отрезке `[a,b]`	`>> x = linspace(a, b, 11)`

Создаем m-функцию в текущей папке	`function y=f(x)`
	`  y=sin(x); % здесь нужна СВОЯ функция`
	`end`

Из среды *Spline Tool: File → Import Data…*, в окне *Please provide data* указываем `x` (или без массива, сразу `linspace..`) и функцию `f`

3. Окно *Spline Tool*

окно *List of approximations* (список приближений)	по одним данным можно построить приближения разным способом
окно *Approximation method* (способ приближения)	*Cubic Spline Interpolant* (интерполяционный кубический сплайн)
окно *Data, breaks/knots, weights* (Данные, точки разрыва веса	Изменение данных в отдельных точках
меню *Tools*	Установка (скрытие) сетки, легенды
меню *View*	Вывод графика первой, или второй производных сплайна, или ошибки (по умолчанию)

4. Выбор граничных условий (`A` и `B` – это числовые значения производных, `i` и `j` – это числа 1 или 2)

not-a-knot (отсутствие узла)	`>> csape(x, y, 'not-a-knot')`
Почему третья производная сплайна непрерывна во второй и предпоследней точках?	
natural или *variational* (естественные)	`>> csape(x, y, 'variational')` `>> csape(x,[0 y 0],[2,2])`
Lagrange (Лагранжа) – по умолчанию	`>> csape(x,y)`
periodic (периодические)	`>> csape(x,y,'periodic'])`
Что означают эти условия?	
clamped или *complete* (условия на первую производную)	`>> csape(x,[A y B],'complete')` `>> csape(x,[A y B],[1,1])`
second (условия на вторую производную)	`>> csape(x,[A y B],'second')` `>> csape(x,[A y B],[2,2])`
custom (произвольные)	`>> csape(x,[A y B],[i,j])`

5. Экспорт сплайна в рабочую среду. *File → Export Spline...*, в окне *Copy spline to workspace* задаем имя `s`. Теперь `s` – это структура с полями `form`, `breaks`, `coefs`, `pieces`, `order`, `dim`).

`form: 'pp'` – форма представления (кусочно-полиномиальная, pp – сокращение от *piecewise-polynomial*)
`breaks: [1x10 double]` – точки разрыва или все узлы
`coefs: [9x4 double]` – массив коэффициентов
`pieces: 9` – число кубических полиномов
`order: 4` – порядок сплайна (степень плюс 1)
`dim: 1` – размерность (одномерный, т.к. данные одномерны)

После этого можем работать со сплайном в Командном окне

Смотрим на коэффициенты	`>> s.coefs`

Вычисление значения сплайна в заданных точках	`>> f = fnval(s,pi/2) % в одной точке`
	`>> x = [-pi/2 0 pi/2];`
	`>> f = fnval(s, x) % в нескольких точках`

Построение графика сплайна	`>> fnplt(s)`
Вычисление производной от сплайна Почему coefs 9x3 и order = 3?	`>> dsdx = fnder(s)`
Вычисление неопределенного интеграла от сплайна Почему coefs 9 на 5 и order = 5?	`>> ints = fnint(s)`

6. **Построить** графики сплайна, его первой, второй и третьей производных и объяснить, почему они такие. Проверить значения первой и второй производной сплайна на границах отрезка интерполирования.

7. **Построить** на одних осях графики кубических полиномов, из которых состоит сплайн различными цветами.

координаты точек разрыва
коэффициенты всех кубических полиномов
точки на 1ом интервале
значения 1ого полинома в них
 	
рисование 1ого полинома красной линией
	`>> br = s.breaks`
	`>> C = s.coefs`
	`>> x = linspace(br(1), br(2), 30);`
	`>> y = C(1,1)*(x-br(1)).^3+C(1,2)*...`
	`(x-br(1)).^2+C(1,3)*(x-br(1))+C(1,4);`
	`>>plot(x, y, 'r')`

8. **Построить** графики зависимости ошибки (модуля максимального уклонения) **естественного кубического сплайна** для своей функции на своем отрезке и для функции sin *x* на отрезке [0, π] в зависимости от шага *h* равномерной сетки. Почему для функции sin *x* ошибка убывает как $O(h^4)$? Как и почему убывает ошибка для своей функции?

1. создать равномерную сетку из n+1 узла	`linspace`
2. вычислить значения функции в этих узлах	`y=`
3. создать структуру кубического сплайна по этим данным	`csape`
4. создать равномерную сетку из 10n+1 узла	`linspace`
5. вычислить максимальный модуль разности между функцией и сплайном по новой сетке	`max, abs`
6. записать шаг и ошибку в массив	`h, error`
7. шаги с 1 по 6 объединить в цикл по числу узлов. На каждом шаге число узлов увеличивать в 2 раза	`for…end`
8. после цикла построить в двойных логарифмических координатах зависимость ошибки от шага и линию h^4	`loglog`

9. Для вычисления корней табличных функций при помощи интерполяции
   1) интерполировать функцию сплайном при помощи функций csape
   2) найти корни сплайна при помощи функции fnzeros, в ее входном аргументе задается сплайн.

В общем случае функция fnzeros выводит матрицу из двух строк

$$\begin{bmatrix} a_1 & a_2 & \dots & a_k \\ b_1 & b_2 & \dots & b_k \end{bmatrix}.$$

Если $a_j \approx b_j$, то это корень, а если $a_j \neq b_j$, то на всем $[a_j, b_j]$ сплайн тождественно равен нулю.

Пример: Вводим данные и строим сплайн (какой?) и отображаем их графически.

Массивы данных	`>> x = [-4.4 -3.8 -2.7 -1.3 0.4 1.8 2.9];` `>> y = [-2.2 -1.8 -0.3 0.6 0.7 1.2 1.7];`
Создание структуры сплайна	`>> s = csape(x, y)`
Построение графика сплайна	`>> fnplt(s); hold on`
Отмечаем узлы	`>> plot(x, y, 'og')`
Находим корни и получаем	`>> r = fnzeros(s)`
Наносим корень маркером на график	`>> plot(r(1), 0, 'or')`

Вычислить корни кубического сплайна построенного по полиному (из своего варианта)