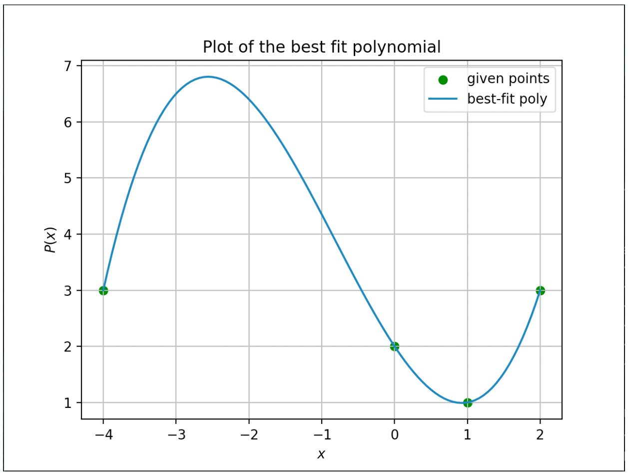
CS5016 : Computational Methods and Applications
Assignment 5 : Least Square Function Approximations

112001051
VM Sreeram

Question 1

The Polynomial class created in previous coding assignment was used to solve this problem.

This problem was solved by solving for a in Sa = b equation, where b is the RHS of Normal Equation for best-fit polynomial, and S_{jk} is $\Sigma_{i=1}^{m} x_i^{j+k}$ (slide 5). The matrix equation was solved using linalg.solve. The function also returns the best fit polynomial, apart from displaying the plot.

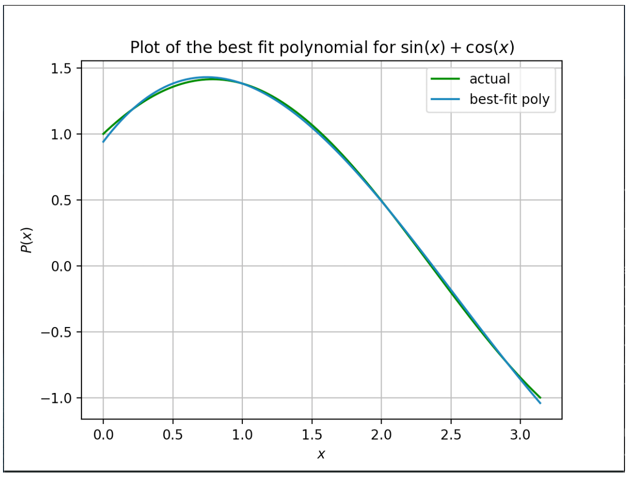


The above plot is shown when bestFitPoly([(2,3),(1,1),(0,2),(-4,3)],3) is run.

Question 2

The Polynomial class created in previous coding assignment was used to solve this problem.

This problem was also solved using the Sa=b solution method, but for normal equation of least square approx of fn using monomial polynomial. The required integrals were computed using quad() from scipy.integrate. The function also returns the best fit polynomial, apart from displaying the plot.



The above plot is shown when bestFitPoly_sinxPLUScosx(3) is run.

Question 3

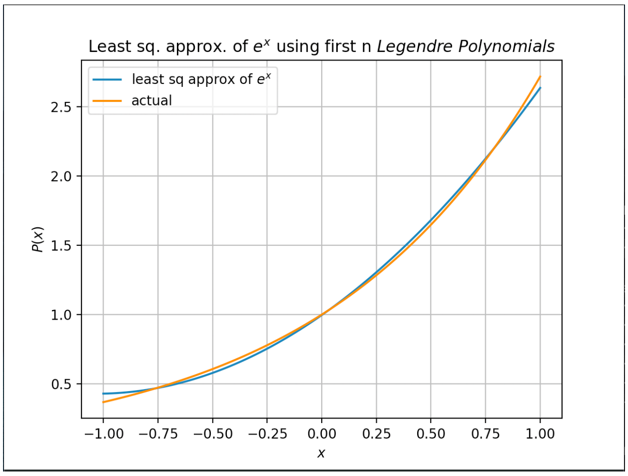
The problem was solved using the formula

$$L_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n$$

Polynomial class's functions were exploited to compute derivative. The function returns object of Polynomial class.

Question 4

The function computes the least square approximation of e^x using first n Legendre Polynomials (computed using the solution for Q3).



The above plot is shown when bestFitLegendre(2) is run.

Question 5

This problem was solved using the recursive formula to compute chebyshev polynomials:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

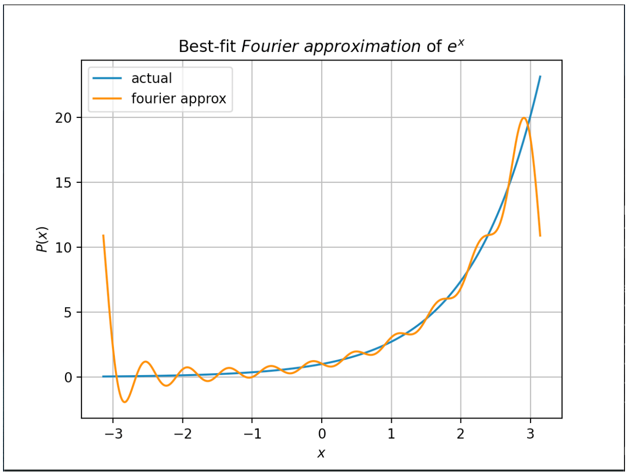
and the base cases $T_0(x) = 1$ *and* $T_1(x) = x$.

Question 6

The function prints a *rounded to 2 decimal places* 5 x 5 square matrix. We can verify that non-diagonal entries are 0 (approx) and diagonal entries are not. This demonstrates that the first 5 chebyshev polynomials are orthogonal to the given weight function.

Question 7

The function shows best fit Fourier approximation of e^x. The function also prints the coefficients of $S_n(x)$.



The above plot is shown when bestFitFourier(10) is run.

Question 8

The program, apart from performing the fft-multiplication also performs the naïve O(n^2) school-multiplication. Then a heavy computation is performed 10000 times and time taken is printed for the comparison.

The output for 223153122414 * 31231231325523:

```
time taken for 100000 iterations of fastft_product =  3.54435396194458 s.
time taken for 100000 iterations of school_product =  4.88260412216186 s.
time taken for 100000 iterations of python product =  0.00769710540771 s.
fastft_product = 6969346787124385501572522
school_product = 6969346787124385501572522
python product = 6969346787124385501572522
```

It was noted that fft-product is faster by significant amount for large multiplications.