5.VAJA: LASTNE VREDNOSTI IN VEKTORJI, FOURIEREVA TRANSFORMACIJA

Dominik Primožič

15. APRIL 2025

1. Naloga

Izračunajte Fourierevo transformacijo nekaj gibanj dušenega nihala in Duffingovega nihala, ki jih vzbujamo.

Metode

Diskretno Fourierjevo transformacijo lahko zelo enostavno izračunamo na 2^N točkah. Preprost rekurziven algoritem je Cooley–Tukey algoritem. Za dan vektor podatkov x[n] je Fourierjeva transformacija:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}, \quad k = 0,1,\dots,N-1$$

Zahteva za število točk 2^N nam omogoča, da podatke razdelimo na sode in lihe:

$$x_e[n] = x[2n]$$

$$x_o[n] = x[2n+1]$$

Potem transformacija postane:

$$X[k] = \sum_{n=0}^{N/2-1} x_e[n] \cdot e^{-j\frac{2\pi}{N}(2n)k} + e^{-j\frac{2\pi}{N}k} \sum_{n=0}^{N/2-1} x_o[n] \cdot e^{-j\frac{2\pi}{N}(2n)k}$$

Kar se poenostavi v:

$$X[k] = X_e[k] + W_N^k X_o[k]$$

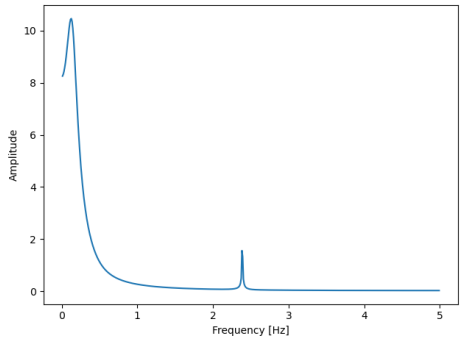
$$X[k+N/2] = X_e[k] - W_N^k X_o[k]$$

Lihe in sode dele se poišče z deljenjem prek rekurzije dokler ne dosežemo bazne rekurzije (eno členskih transformacij.

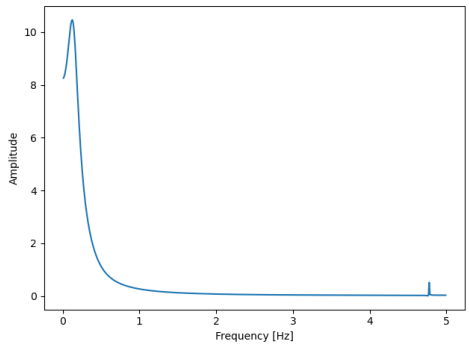
Nalogo sem rešil le za Duffingovo nihalo, ker sem imel program napisan že od prejšnjih nalog. Najprej sem generiral vektor podatkov o nihanju za dan vzorčevalni čas. Te sem nato transformiral s FFT algoritmom. Iz vzorčevalnega časa in števila točk sem izračunal frekvenčno domeno, da sem lahko grafično predstavil transformiran signal.

Rezultati

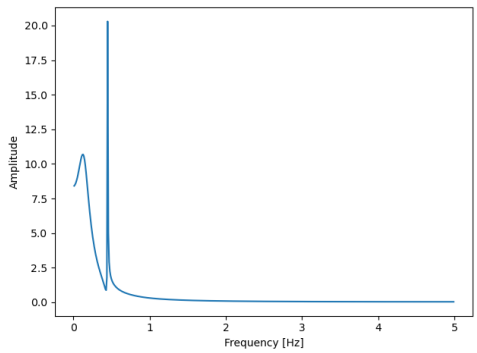
Najprej sem za Duffingovo nihalo vse parametre razen vzbujevalne frekvence nastavil na 1, korak je bil 0.1 in 1024 korakov. Vzbujevalno frekvenco sem postopno spreminjal in transformiral pozicije s FFT algoritmom.



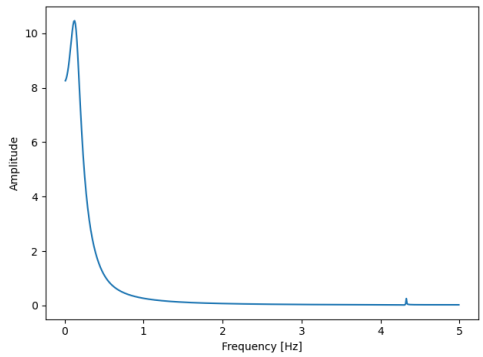
Graf 1: ω=15



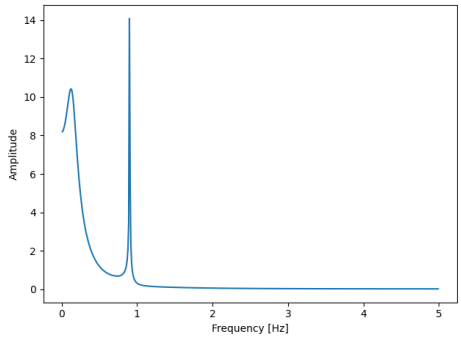
Graf 2: ω=30



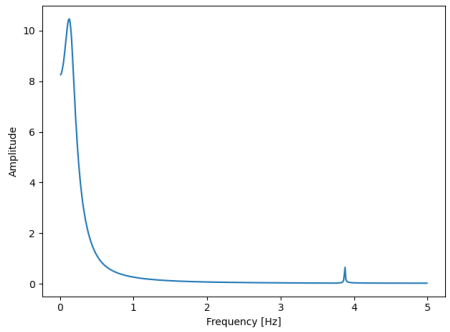
Graf 3: ω= 60



Graf 4: ω=90



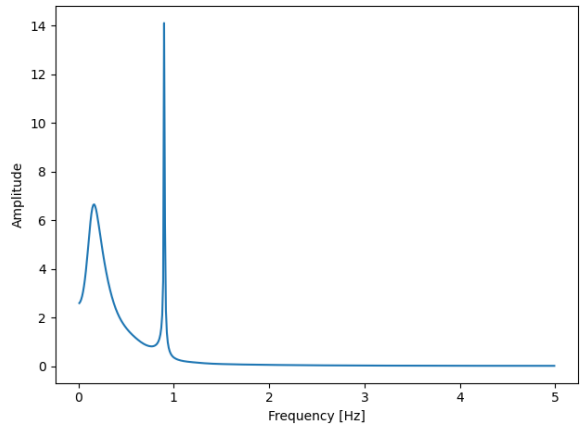
Graf 5: ω= 120



Graf 6: ω=150

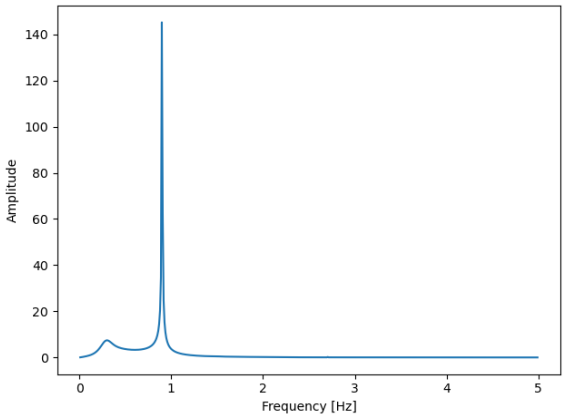
Grafov ne znam kritično ovrednotiti, ker Fourierjeve transformacije ne poznam dovolj dobro. Razvidno je, da se pri nekaterih krožnih frekvencah v frekvenčnem prostoru pojavijo vrhovi kasneje in en ali dva z večjo amplitudo. Začetni vrh je prisoten pri vseh.

Nato sem poskusil še s spremembo drugih parametrov.



Graf 7: α=1 β=10 δ=1 γ=1 ω=300

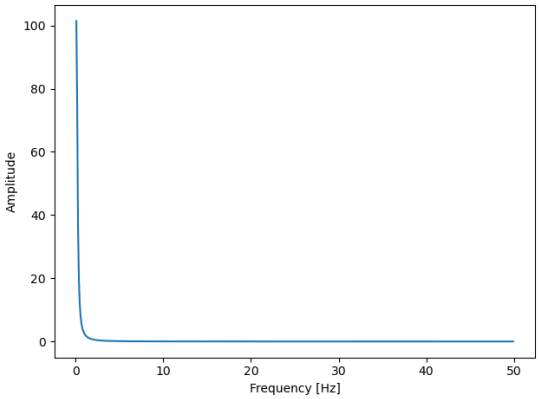
Pri tem je prvi vrh šibkejši in vrh pri 1 Hz zelo intenziven. To je lahko zaradi β ali pa, ker je ω=300



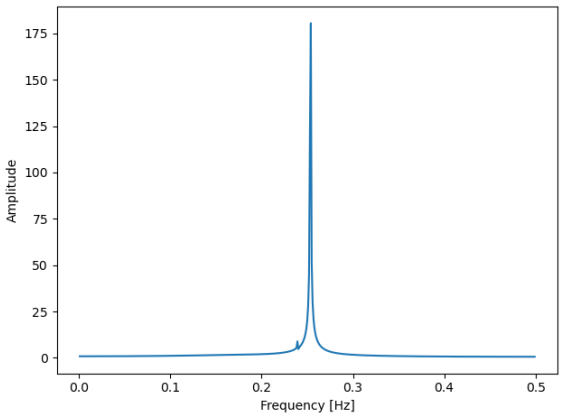
Graf 8: α=1 β=10 δ=1 γ=10 ω=120

Enak vrh pri 1Hz se pojavi tudi pri tem setu parametrov, torej je najverjetneje to posledica nihanja, ki ga narekuje β.

Preveril sem tudi kako vpliva vzorčevalni čas.



Graf 9: α=1 β=10 δ=1 γ=10 ω=300, dt=0.01



Graf 10: α=1 β=10 δ=1 γ=10 ω=300, dt=1

Pri majhnem vzorčevalnem času je v frekvenčnem prostoru le en vrh na začetku, pri večjem vzorčevalnem času pa je vrh bolj oblikovan na sredini intervala.

1. Naloga

Za harmonski oscilator z motnjo tretjega ali četrtega reda poišči neka najnižjih energij in lastnih stanj, za različne močne motnje.

$$\hat{H} = \hat{H}_0 + \alpha \hat{q}^3$$

Hamiltonov operator v brezdimenzijski obliki zapišemo kot

$$\hat{H}_0 = \frac{1}{2}(\hat{p}^2 + q^2)$$

Metode

Za harmonski oscilator v kvantni mehaniki rešujemo Schrödingerjevo enačbo:

$$\hat{H}\psi(x) = E\psi(x)$$

Reševanja se lotimo z uvedbo dveh novih operatorjev, tako imenovanih »creation« in »annihilation« operatorjev:

$$\hat{a} = \frac{1}{\sqrt{2}}(\hat{q} + i\hat{p}),$$

$$\hat{a}^\dagger = \frac{1}{\sqrt{2}}(\hat{q} - i\hat{p})$$

Ta operatorja delujeta na stanja n prek relacij:

$$\hat{a}|n\rangle = \sqrt{n}|n-1\rangle,$$

$$\hat{a}^\dagger|n\rangle = \sqrt{n+1}|n+1\rangle$$

Z njima zapišemo Hamiltonov operator:

$$\hat{H} = \hat{a}^\dagger\hat{a} + \frac{1}{2}$$

Potem lahko Schrödingerjevo enačbo zapišemo v obliki:

$$\hat{H}|n\rangle = \left(n + \frac{1}{2}\right)|n\rangle$$

kjer so $|n\rangle$ lastni vektorji Hamiltonovega operatorja (valovne funkcije), $n + \frac{1}{2}$ pa lastne vrednosti.

Ob uvedbi matrične formulacije kvantne mehanike lahko operatorje zapišemo kot matrične elemente:

$$H_{i,j} = \langle i|\hat{H}|j\rangle$$

Podobno storimo za druge operatorje.

Energije so potem diagonalni elementi Hamiltonovega operatorja, saj so bazni vektorji neodvisni in tvorijo enotsko matriko.

V primeru, da želimo valovne funkcije izraziti v prostorski bazi, jih razširimo:

$$\psi_n(x) = \langle x|n\rangle = \left(\frac{1}{\sqrt{2^n n!\,\pi^{1/2}}}\right) H_n(x)\, e^{-\frac{x^2}{2}}$$

Motnjo uvedemo, tako da jo izrazimo z operatorjema številske baze. Popravimo Hamiltonov operator in zopet izračunamo lastne vrednosti in vektorje.

$$\hat{H}|k\rangle = E_k|k\rangle$$

Te lastne vektorje izrazimo s prejšnjimi:

$$|k\rangle = \sum_n c_n^{(k)}|n\rangle$$

Potem dobimo:

$$\hat{H}|k\rangle = \hat{H}\sum_n c_n^{(k)}|n\rangle = E_k\sum_n c_n^{(k)}|n\rangle$$

V matrični obliki pa:

$$HC = E_k C$$

V tem primeru matrika lastnih vektorjev predstavlja matriko koeficientov za linearno kombinacijo številskih lastnih vektorjev.

Za reševanje tega problema sem nadgradil svoj matrični objekt iz prejšnjih vaj, dodal sem tudi hitre matrične operacije na SIMD in paralelno množenje matrik.

```
matrix matrix::add(const matrix& other) const {
    matrix result(rows_, cols_);

    size_t total_size = rows_ * cols_;

    #pragma omp simd
    for (size_t i = 0; i < total_size; ++i) {
        result.data[i] = data[i] + other.data[i];
    }

    return result;
}

matrix operator+(const matrix& a, const matrix& b) {
    return a.add(b);
}

matrix operator-(const matrix& a, const matrix& b) {
    matrix result(a.rows(), a.cols());
    size_t total_size = a.rows() * a.cols();

    #pragma omp simd
    for (size_t i = 0; i < total_size; ++i) {
        result.get_data()[i] = a.data[i] - b.data[i];
    }

    return result;
}
```

```
matrix operator*(const matrix& a, const matrix& b) {
    matrix result(a.rows(), b.cols());
    #pragma omp parallel for
    for (int i = 0; i < a.rows(); ++i) {
        for (int j = 0; j < b.cols(); ++j) {
            double sum = 0.0;
            for (int k = 0; k < a.cols(); ++k) {
                sum += a(i, k) * b(k, j);
            }
            result(i, j) = sum;
        }
    }

    return result;
}

matrix operator*(const matrix& mat, double scalar) {
    matrix result(mat.rows(), mat.cols());
    size_t total_size = mat.rows() * mat.cols();

    #pragma omp simd
    for (size_t i = 0; i < total_size; ++i) {
        result.get_data()[i] = mat.data[i] * scalar;
    }

    return result;
}

matrix operator*(double scalar, const matrix& mat) {
    return mat * scalar;
}
```

Zapisal sem svoj eigensolver, ki temelji na Jacobijevi metodi, saj so vse matrike simetrične.

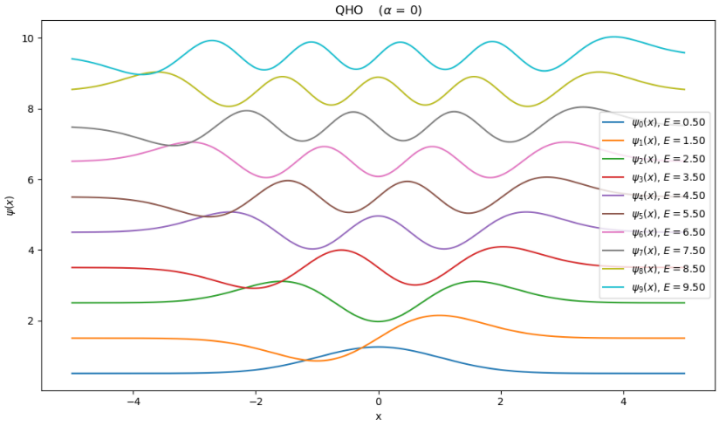
Rezultati

Izračunal sem prvih 10 energij za različne motnje, rezultati so v spodnji tabeli.

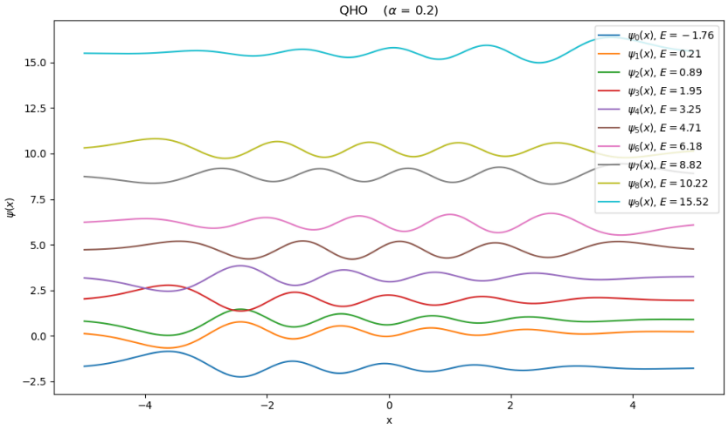
Stanje \ α	0	0.2	0.4	0.6	0.8	1
0	0.50	-1.76	-9.85	-18.11	-26.41	-34.72
1	1.50	0.21	-2.31	-5.60	-9.00	-12.44
2	2.50	0.89	0.11	-0.77	-1.92	-3.22
3	3.50	1.95	1.31	0.91	0.54	0.10
4	4.50	3.25	2.89	2.56	2.15	1.73
5	5.50	4.71	4.43	4.10	3.92	3.89
6	6.50	6.18	6.51	7.14	7.87	8.65
7	7.50	8.82	10.37	11.67	13.03	14.45
8	8.50	10.22	12.81	16.09	19.49	22.93
9	9.50	15.52	23.74	32.02	40.32	48.63

Tabela 1: Prvih 10 energij za različne motnje

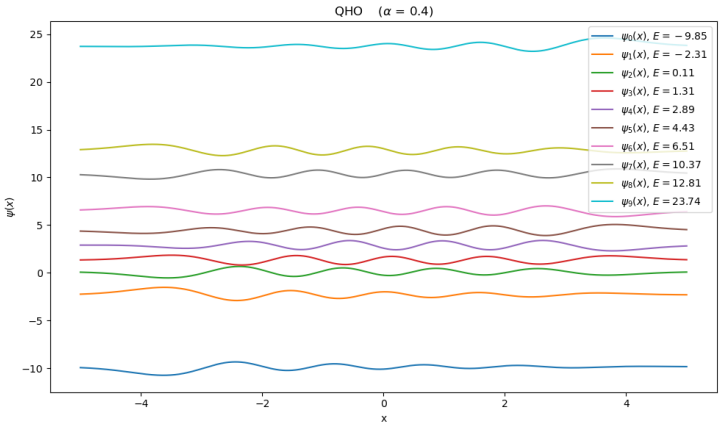
Nato pa sem še grafično predstavil valovne funkcije za vse te energije.



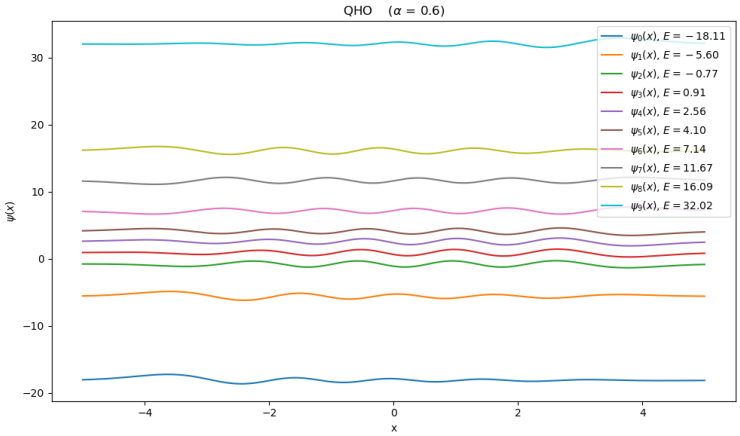
Graf 11: Valovne funkcije za α=0



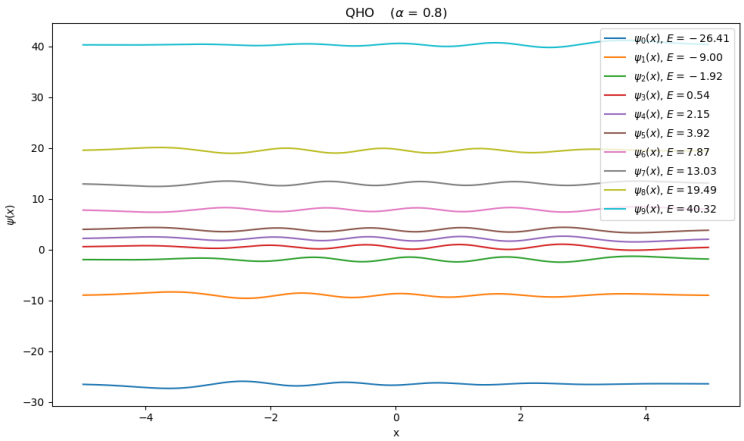
Graf 12: Valovne funkcije za α=0.2



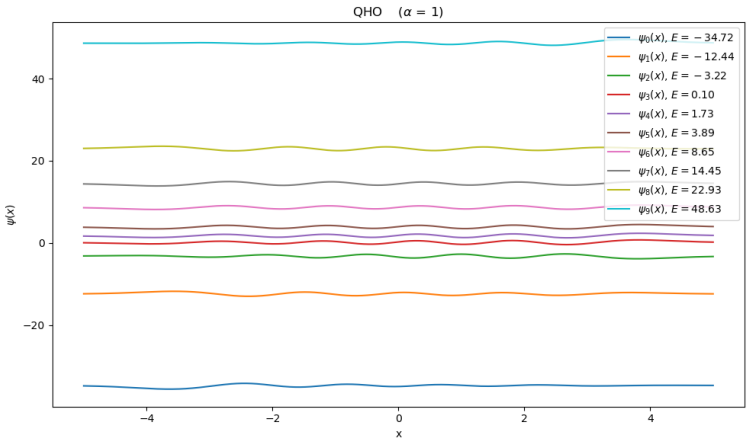
Graf 13: Valovne funkcije za α=0.4



Graf 14: Valovne funkcije za α=0.6

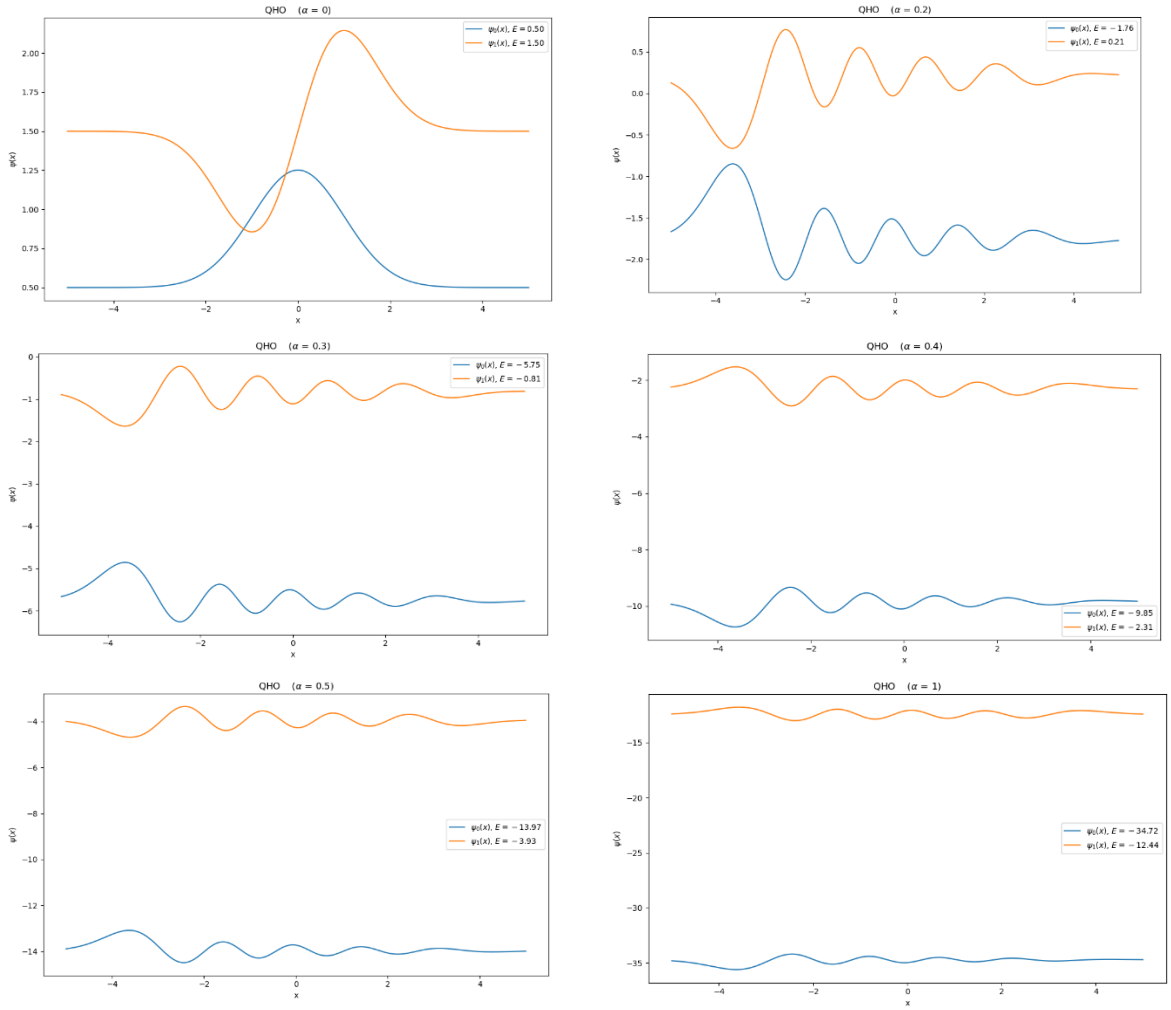


Graf 15: Valovne funkcije za α=0.8



Graf 16: Valovne funkcije za α=1

Razvidno je, da večja kot je motnja, manjša je energija osnovnega stanja in večje so energijske razlike med stanji. Nihanja se zdijo bolj dušena, a to je le zaradi raztegnjene osi, zato sem za prvi dve stanji preveril nekaj različnih moten in kako vplivajo na nihanja. Za izračun matrike sem uporabil 10 vrednosti, ker pri samo dveh sploh ni prišlo do sprememb, to se zgodi, ker so stanja sklopljena in so rešitve odvisne od števila sistemu dostopnih stanj.



Graf 17: Vpliv velikosti motnje na prvi dve stanji (n=10)

Jasno se vidi vpliv motnje, ki počasi duši nihanje in zmanjšuje energije.

Nazadnje sem preveril še kako število vseh stanj v sistemu vpliva na energijo osnovnega stanja. Za α sem izbral, kar 0.1.

Število stanj	E_0
1	0.5
10	0.444157
100	-148.277
1000	-6821.4

Tabela 2: Vpliv števila stanj na energijo osnovnega stanja

Več kot je v sistemu na voljo stanj manjša je energija osnovnega stanja. To je logično, ker več kot je stanj več je možnega mešanja in nastalo bo vedno bolj ugodno stanje, seveda pa tudi vedno več manj ugodnih stanj.