Лабораторная работа 5 по курсу «Нелинейная динамика и её приложения». Отчёт.

Владислав Соврасов
381503м4

1 Оценка качества нахождения Флоке-базиса для уравнения Шрёдингера с периодическим по времени гамильтонианом

Рассматриваается уравнение Шрёдингера с периодической правой частью:

$$i|\dot{\psi}\rangle = H(t)\,|\psi\rangle\,, H(t) = H_0 + F\sin\left(\frac{2\pi}{T}t\right)H_1, \tag{1}$$

в котором H_0, H_1 — GUE матрицы.

Для нахождения Флоке-базиса в момент времени T используется матрица-пропагатор $U(T)$. Как было выяснено, её собственные векторы являются Флоке-базисом в моменты kT, а собственнные значения определяют квазиэнергии системы.

При нахождении каждого из столбцов матрицы $U(T)$ применялось численное интегрирование методом Рунге-Кутты из стандартных базисных ортов евклидова пространства.

Если взять в качестве начального условия один из векторов Флоке-базиса и подействовать на него пропагатором, то $|\psi(T)\rangle = U(T)\,|\varphi(0)\rangle = \mu\,|\varphi(0)\rangle$. Сравнивая это с теоремой Флоке ($|\psi(T)\rangle = e^{i\Xi T}\,|\varphi(T)\rangle$), получим что, $\mu = e^{i\Xi T}$; откуда следует, что в качестве оценки ошибки нахождения собственных числел $U(T)$ можно рассматривать величину $\varepsilon_\mu = \max_{k=\overline{1,N}} ||\mu_k| - 1|$.

На рис. 1 показан график зависимости ε_μ от шага интегрирования для системы размерности 100 при $F = 0.1, T = 1$. Как видно из графика, ошибка при уменьшении шага убывает с четвёртым порядком.

В качестве оценки ошибки нахождения собственных векторов использовалась величина $\varepsilon_\varphi = \max_{k=\overline{1,N}} \|U_{\frac{h}{2}}(T)\,|\varphi_{k,h}\rangle - \mu_{k,h}\,|\varphi_{k,h}\rangle\,\|$, где h — шаг интегрирования, с которым получены пропагатор и соответствующие векторы и собственные числа. Фактически, эта запись эквивалентна численному интегрированию системы с половинным шагом из начальных условий $|\varphi_{k,h}\rangle$ и последующему сравнению с результатом пропагации $U_h(T)\,|\varphi_{k,h}\rangle = \mu_{k,h}\,|\varphi_{k,h}\rangle$, полученным с одинарным шагом. На рис. 2 показана зависимость ε_φ от шага интегрирования для той же системы размерности 100, что рассматривалась ранее. Величниа ε_φ, как и ε_μ, при уменьшении шага убывает с четвёртым порядком.

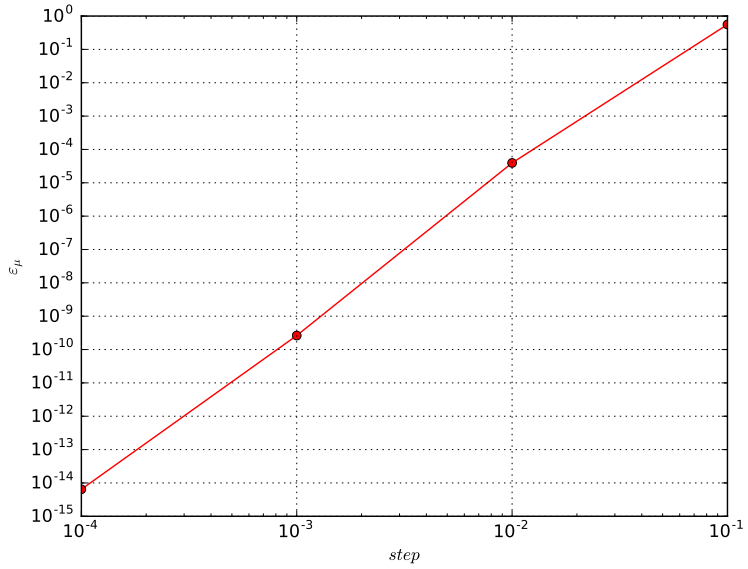

Рис. 1: Отклонение модуля собственных чисел матрицы-пропагатора от 1

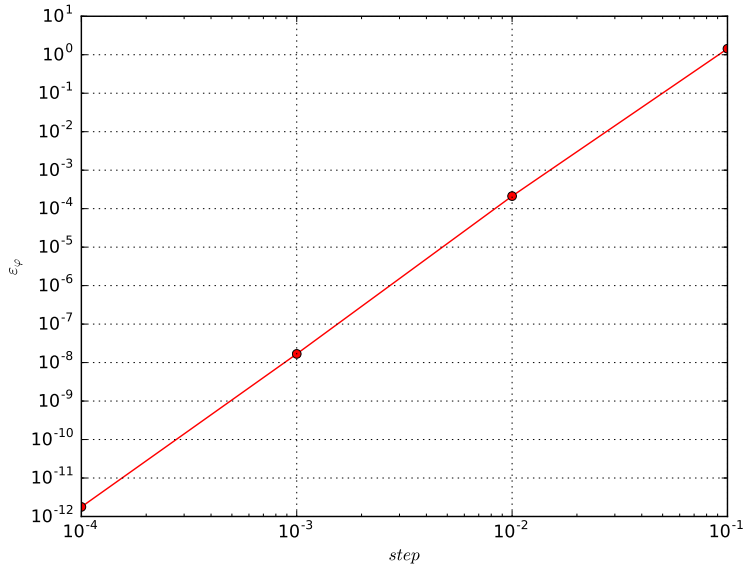

Рис. 2: Оценка ошибки нахождения собственных векторов матрицы-пропагатора

2 Количественные характеристики решений уравнения Шрёдингера с периодическим по времени гамильтонианом

Для периодического по времени гамильтониана из (1) можно ввести аналог энергии стационарной системы: $E_\alpha(t) = \langle \varphi_\alpha(t) | H(t) | \varphi_\alpha(t) \rangle$, где α — индекс Флоке-вектора. Будем рассматривать величниу $E_\alpha(0)$, тогда $\varphi_\alpha(0)$ является собственным вектором пропагатора $U(T)$.

Для изучения распределения величины $E_\alpha(0)$ (квазиэнергии нестационарной системы) при значениях $F \in \{0.01, 0.1, 1\}$ были сгенерировынны 120 реализаций пар матриц (H_0, H_1) из (1) размерности 100. Перед построением нормированных гистограмм, как и в случае стационарной системы, квазиэнергии были умножены на коэффициент $\frac{1}{\sqrt{N}}$, чтобы их спектр не зависел от размерности системы.

На рис. 3, 4, 5 можно увидеть, что с ростом F спектр начинает сосредотачивается в окрестности 0 (то есть при нарастании отличия гамильтониана от стационарного): при $F = 0.01$ спектр квазиэнергий почти не отличается от спектра энергий стационарной системы, а при $F = 1$ имеется явный пик распределения вблизи 0.

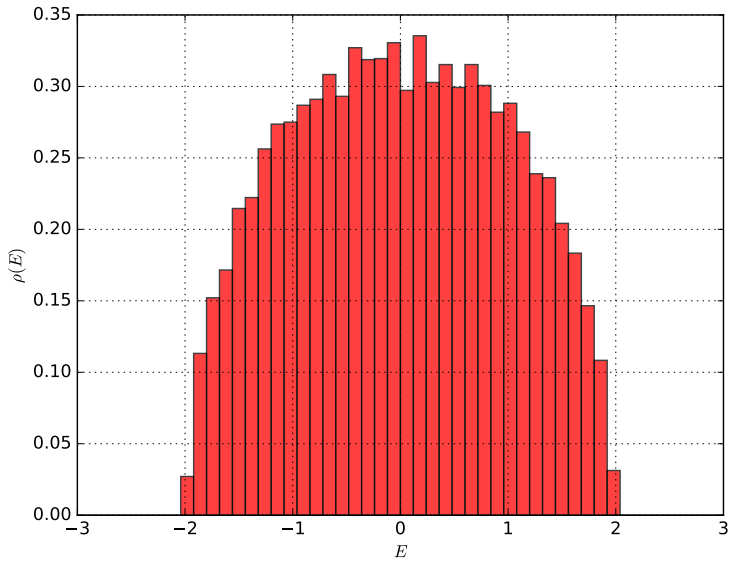

Рис. 3: Распределение энергий Флоке-векторов гамильтониана при $F = 0.01$

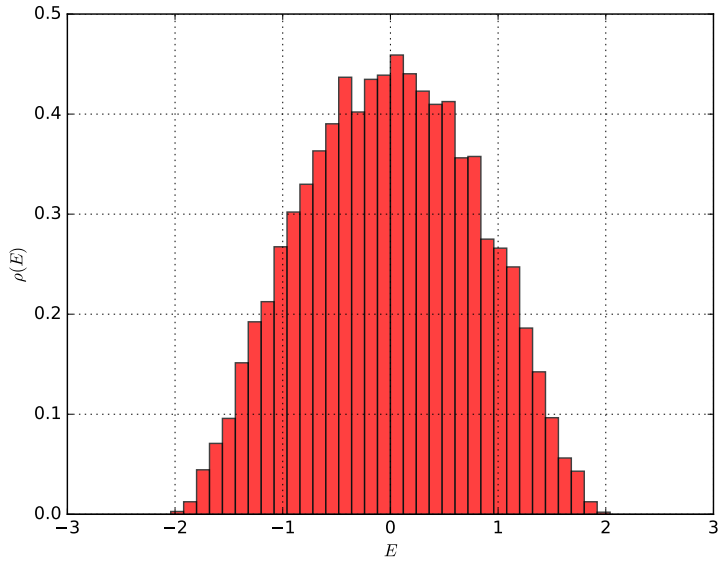

Рис. 4: Распределение энергий Флоке-векторов гамильтониана при $F = 0.1$

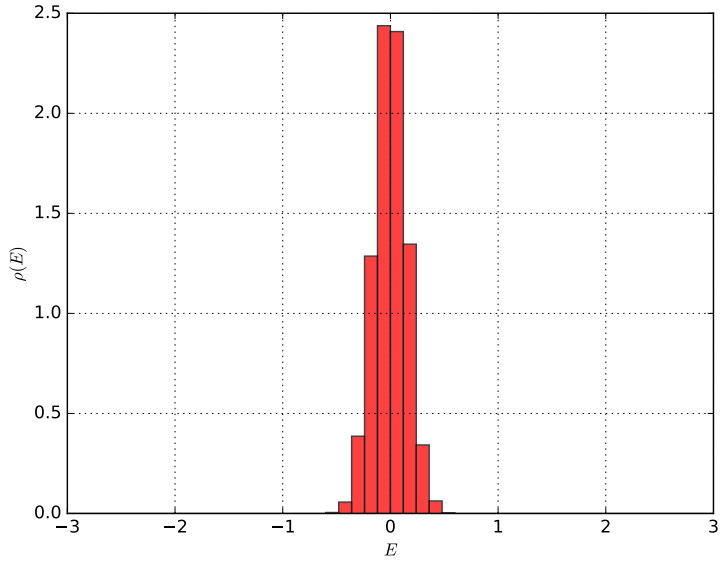

Рис. 5: Распределение энергий Флоке-векторов гамильтониана при $F = 1$

Как и в стационарном случае, наряду с самими квазиэнергиями рассматривается и их расщепление, определяемое по формуле: $s_\alpha = E(0)_{\alpha+1} - E(0)_\alpha, i = \overline{1, N-1}$. По полученным ранее реализациям были построены гистограммы нормированных расщеплений: $\bar{s}_\alpha = \frac{s_\alpha}{\langle s_\alpha \rangle}$,

где $\langle s_\alpha \rangle$ — средняя величина расщепления по всем реализациям. Из рис. 6, 7, 8 видно, что для гамильтониана, близкого к стационарному, распределения имеет удалённый от нуля пик, но при увеличении F этот пик смещается влево, что говорит о росте плотности спектра квазиэнегий.

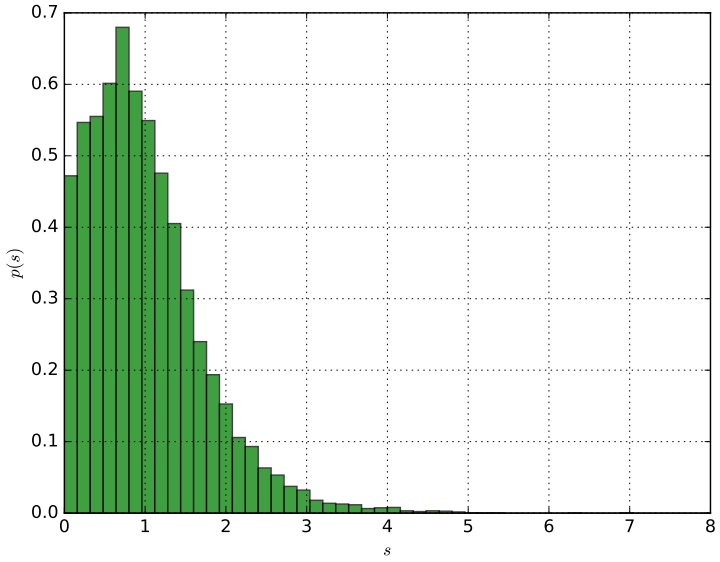

Рис. 6: Распределение расщепления энергий Флоке-векторов гамильтониана при $F = 0.01$

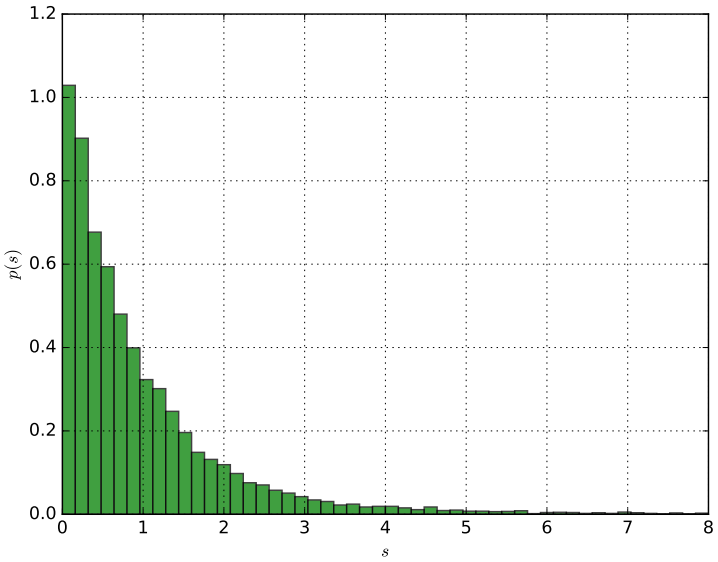

Рис. 7: Распределение расщепления энергий Флоке-векторов гамильтониана при $F = 0.1$

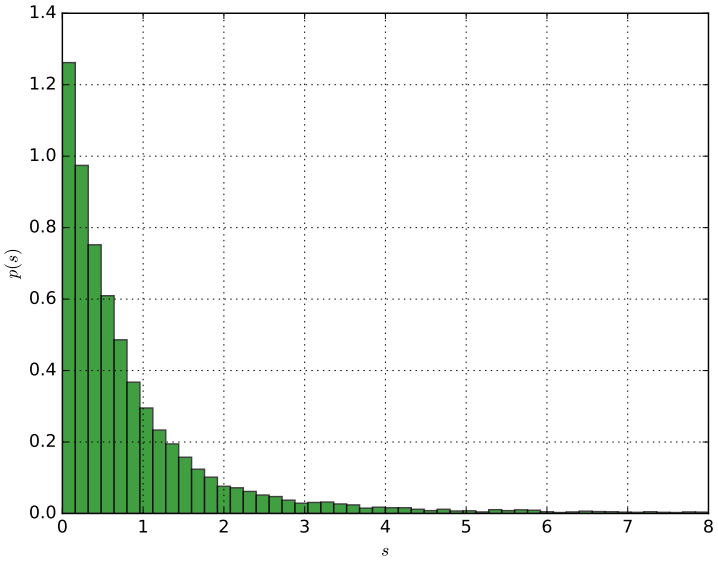

Рис. 8: Распределение расщепления энергий Флоке-векторов гамильтониана при $F = 1$

3 Исходный код

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt
from lab4 import getGUE
from lab3 import rungeKuttaMethod

def getPropagator(f, T, size, step):
    U = [[]]*size

    for i in range(size):
        psi0 = np.zeros((size, 1))
        psi0[i] = 1.
        uCol = rungeKuttaMethod(f, 0., T, psi0, step)[1][-1]
        U[i] = uCol

    return np.matrix(np.array(U).reshape(size, size)).getT()

def plotError(steps, errValues, yLabel, outputName):
    colors = ['r', 'g', 'b', 'c']

    plt.xlabel('$step$')
    plt.ylabel(yLabel)
    plt.yscale('log')
    plt.xscale('log')

    plt.plot(steps, errValues, colors[0] + '-o')

    plt.grid()
    plt.savefig(outputName, format = 'pdf')
    plt.clf()

def plotHist(data, xLabel, yLabel, fileName, color, valRange):
    colors = ['blue', 'green', 'red']
    plt.xlabel(xLabel)
    plt.ylabel(yLabel)

    plt.hist(data, 50, normed = True,
        facecolor = color, alpha = 0.75, range = valRange)

    plt.grid()
    plt.savefig(fileName, format = 'pdf')
    plt.clf()

def main():

    np.random.seed(10)

```

```
50      size = 100
51      H0 = getGUE(size)
52      H1 = getGUE(size)
53      f = lambda x, t: -(H0 + 0.1*np.sin(2.*np.pi*t)*H1)*x * 1j
54
55      epsPhi = []
56      epsMu = []
57
58      steps = np.logspace(-4, -1, 4, endpoint=True)
59      for step in steps:
60          U = getPropagator(f, 1., size, step)
61          U2 = getPropagator(f, 1., size, step / 2.)
62
63          values, vectors = np.linalg.eig(U)
64
65          epsPhi.append(0.)
66          for i in range(len(values)):
67              phi2 = U2*vectors[:,i]
68              epsPhi[-1] = max(epsPhi[-1], \
69                  np.linalg.norm(phi2 - np.matrix(vectors[:,i])*values[i]))
70
71          epsMu.append( \
72              np.linalg.norm(np.abs(values) - np.ones(len(values)), np.inf))
73      plotError(steps, epsMu, \
74                  r'$\varepsilon_{\mu}$', '../pictures/lab5_eigvals_error.pdf')
75      plotError(steps, epsPhi, \
76              r'$\varepsilon_{\varphi}$', '../pictures/lab5_eigvecs_error.pdf')
77
78      size = 100
79      nImpls = 120
80      fValues = [0.01, 0.1, 1]
81
82      for F in fValues:
83          allEnergies = []
84          allSplits = []
85          for i in range(nImpls):
86              H0 = getGUE(size)
87              H1 = getGUE(size)
88              f = lambda x, t: -(H0 + H1*F*np.sin(2.*np.pi*t))*x * 1j
89
90              U = getPropagator(f, 1., size, 0.005)
91              _, vectors = np.linalg.eig(U)
92              energysSet = []
93              for j in range(len(vectors)):
94                  vector = np.matrix(vectors[:,j])
95                  energysSet.append(\
96                      float(np.real(np.conj(vector.getT())*H0*vector)))
97
98              energysSet = sorted(energysSet)
```

```
 99                 allEnergies.append(energysSet)
100                 allSplits.append( \
101                     [energysSet[k+1] - energysSet[k] \
102                         for k in range(len(energysSet) - 1)])
103 
104                 print('f = {}, #impl: {} '.format(F, i))
105 
106             allEnergies = np.array(allEnergies).reshape(nImpls*len(allEnergies[0]))
107             allSplits = np.array(allSplits).reshape(nImpls*len(allSplits[0]))
108             allSplits /= np.mean(allSplits)
109             allEnergies /= np.sqrt(size)
110 
111             energySetsToPlot.append(allEnergies)
112             spacingSetsToPlot.append(allSplits)
113 
114             plotHist(allEnergies, r'$E$', r'$\rho(E)$',
115                 '../pictures/lab5_eigens_hist' + 'F=' \
116                             + str(F) +'.pdf', 'red', (-3, 3))
117 
118             plotHist(allSplits, r'$s$', r'$p(s)$',
119                 '../pictures/lab5_eigens_spacings_hist' + 'F=' + str(F) + '.pdf', \
120                     'green', (0, 8))
121 
122 if __name__ == '__main__':
123     main()
```