Quantum Mechanics Bootstraps and the BFSS Conjecture

Peter Ripoll Kandylas

2026

UNIVERSITY
OF CRETE

Physics Department

Supervised by
Prof. Dr. Vasilis Niarchos

To my grandfather

Abstract

In this thesis, motivated by the BFSS conjecture and the corresponding matrix model, we illustrate the quantum mechanics bootstrap method using semidefinite programming and apply it to the harmonic and anharmonic oscillators for zero temperature and the harmonic and quartic oscillators for non-zero temperatures. We, also, briefly showcase the BFSS conjecture and attempts of using this method for probing the physics behind it.

Contents

1 Introduction

The Matrix Model defined by the Hamiltonian:

$$H = \mathrm{Tr}(\frac{1}{2}(P^i)^2 - \frac{1}{4}[X^i, X^j]^2 + \frac{1}{2}\psi_\alpha \gamma^i_{\alpha\beta}[X_i, \psi_\beta]) \tag{1.0.1}$$

allows for probing of M theory aspects according to the BFSS conjecture [1] such as graviton scattering. Recent work has been focused on examining this model, with some of it [2, 3, 4, 5] focusing on so-called Bootstrap methods.

The Bootstrap method leverages symmetries (depending on the system we are looking at, this might be SU(N) invariance for example) and attributes of a system (like its equation of motion) to arrive to rigorous upper and lower bounds for observables. Importantly, the bootstrap method is analytical, does not rely on weak coupling expansions and can work directly in large N limits, making it rather suitable for analysing this Matrix Model. Bootstrap techniques have been applied with notable success in other cases such as CFTs (with the Conformal Bootstrap Programme) [6, 7], in Matrix Quantum Mechanics [8, 4] and in lattice field theory (see [9] to name one example)

In the pages that follow we will present the non-negative Bootstrap (henceforth referred to simply as Bootstrap) method using Semidefinite Programming (SDP) and briefly delve into the BFSS conjecture and aspects of the Matrix Model as well as mention literature that uses the Bootstrap to approach this model.

In section 2 we will illustrate the method for both 0 temperature systems through the Harmonic and Anharmonic Oscillators and thermal cases following [10] with the harmonic and quartic oscillators.

In section 3 we illustrate a derivation of the bosonic part of 1.0.1 through the generalised DBI action as well as some arguments pertaining to the BFSS conjecture following [11]. We also list some properties of the matrix model.

In section 4 we list some similarities for approaching this problem as a bootstrap problem with the applications we showed in 2 as well as list some literature that actually implements the bootstrap.

In appendix A we briefly introduce the idea of Kaluza-Klein (KK) compactification and in appendix B we explain in detail the implementation code.

2 Quantum Mechanical Bootstrap

2.1 The Basics of QM Bootstraps

The main idea behind a non-negative quantum mechanical Bootstrap is that, using the non-negativity of the norm paired with more specialised constraints, we can

reject potential values for observables.

$$\langle \mathcal{O}^\dagger \mathcal{O} \rangle_\beta \geq 0 \ , \forall \ \mathcal{O} \in \mathcal{H} \tag{2.1.1}$$

While exact values are not always obtainable, one can find rigorous upper and lower bounds for observables.

The Setup Given a Hamiltonian H which defines a Hilbert space $\mathcal{H}$ and a set of operators $\mathcal{L}$, we can define the truncated operator basis with operator "words" of lengths L as:

$$\mathcal{B}_L = \{\text{all combinations of the operators in } \mathcal{L} \tag{2.1.2}$$

$$\text{with length} \leq L\} \tag{2.1.3}$$

We can use this truncated basis to set up the constraints used for any given problem. Central to all the examples we will see are the following constraints:

$$\langle [\hat{H}, \hat{\mathcal{O}}] \rangle_\beta = 0 \quad \forall \ \hat{\mathcal{O}} \in \mathcal{B}_L \tag{2.1.4}$$

$$\langle \hat{\mathcal{O}_1^\dagger} [\hat{x}, \hat{p}] \hat{\mathcal{O}}_2 \rangle_\beta = i \langle \hat{\mathcal{O}}_1^\dagger \hat{\mathcal{O}}_2 \rangle_\beta \quad \forall \ \hat{\mathcal{O}}_1, \hat{\mathcal{O}}_2 \in \mathcal{B}_L \tag{2.1.5}$$

Where the expectation value is taken either with a density matrix that commutes with the Hamiltonian in the case of non-zero temperature or in an Energy Eigenstate. It is worth pointing out that the canonical relations are equivalent to simply ordering all operators such that all powers of either x or p are first in all operator expressions that are written which we used during implementation.

2.2 Bootstrap Examples

2.2.1 Zero Temperature

Harmonic Oscillator The simplest case we can use this tool in is the Harmonic Oscillator with Hamiltonian:

$$\hat{H} = \frac{\hat{p}^2}{2} + \frac{\hat{x}^2}{2} \tag{2.2.1}$$

The set of letters is $\mathcal{L} = \{\hat{x}, \hat{p}\}$ with the search space being the set containing the expectation values of all words of lengths L or less. Moreover, we have the constraints:

- $\langle \hat{\mathcal{O}^\dagger} \hat{\mathcal{O}} \rangle \geq 0 \ \forall \ \hat{\mathcal{O}} \in \mathcal{B}_L$
- $\langle [H, \hat{\mathcal{O}}] \rangle = 0 \ \forall \ \hat{\mathcal{O}} \in \mathcal{B}_L$
- $\langle \hat{\mathcal{O}^\dagger}_i [x, p] \hat{\mathcal{O}}_j \rangle = i \langle \hat{\mathcal{O}^\dagger}_i \hat{\mathcal{O}}_j \rangle \ \forall \ \hat{\mathcal{O}}_i, \hat{\mathcal{O}}_j \in \mathcal{B}_L$

- $\langle H\hat{\mathcal{O}}\rangle = E\langle\hat{\mathcal{O}}\rangle$

Using the canonical relations and the Schwinger-Dyson equation while working in an energy eigenstate at zero temperature, we can reduce any expectation value of the form $\langle\hat{x}^m\hat{p}^n\rangle$ to expressions containing only moments of $\hat{x}$ and the energy, giving us a recursion relation between all higher moments of $\hat{x}$ with the Energy and some moments required to initialise the recursion. Which those moments are depends on the potential. For more on this, refer to [12] section 3.1.1.

This means we are only interested in operators of the form $\hat{\mathcal{O}} = \sum_{n=0} c_n\hat{x^n}$ meaning that:

$$\langle\hat{\mathcal{O}}^\dagger\hat{\mathcal{O}}\rangle \geq 0 \implies \sum_{n,m} c_n^*\langle\hat{x}^{m+n}\rangle c_m = c^\dagger Mc \geq 0 \iff M \succeq 0 \tag{2.2.2}$$

Where the last implication can be taken as the definition of a positive semidefinite Hermitian matrix, i.e. a matrix whose eigenvalues are non-negative. Since the recursion implicates the energy in a non-linear fashion, we will have to scan over a search space of energies. For the harmonic oscillator we only need to scan over the energy. Since the eigenvalues must be positive, we are only concerned with the sign of the minimum eigenvalue of the moment matrix M and so to find a lower bound for the energy we must minimize $\min\mathrm{Spec}(M(E))$. In practice, we restrict ourselves to a truncated basis of operators up to length L which, after eliminating the moments of product operators, will correspond to a basis of only powers of $\hat{x}$. The truncation corresponds to a matrix size K. We will denote matrices in a truncated basis as $M^{(L)}$.

Now the SDP we want to carry out is:

$$\begin{aligned}&\text{Minimize: } \min\mathrm{Spec}(M^{(L)}(E))\\ &\text{Subject to: } M^{(L)}(E) \succeq 0\\ &\text{With Search-space: } \mathcal{E} = \{E\}\end{aligned}$$

However, since the eigenvalues are not affine in E, and as such we can't use the energy as an SDP variable, we want to transform the problem in a way where we have easier access to the minimum eigenvalue. This can be done by relaxing the constraint $M^{(L)} \succeq 0$ to $M^{(L)} - tI \succeq 0$ and adding the auxiliary variable t into our search space. Now, we only need to maximize t or, if we have to minimize the objective function as is often the case during implementation, minimize -t. This corresponds to finding a lower bound for $\min(\mathrm{Specc}(M^{(L)}(E))$. To see this, note that we can diagonalize M and the identity matrix simultaneously, effectively subtracting

t from each eigenvalue.This means that:

$$M^{(L)}(E) - tI \succeq 0 \iff \min \text{Spec}(M^{(L)}(E) - tI) \geq 0 \implies \min \text{Spec}(M^{(L)}(E)) \geq t$$

From this we see that to check if $M^{(L)}(E) \succeq 0$, and thus that the energy E is allowed, we need only check that $t_\star := \max t \geq 0$. With this, we can now run the final SDP defined as:

$$\begin{aligned} &\text{Minimize: } -t \\ &\text{Subject to: } M^{(L)}(E) - tI \succeq 0 \\ &\text{With Search-space: } \mathcal{E} = \{E, t\} \end{aligned}$$

Bellow, we show the allowed energies in figure 1, and the comparison between the minimum eigenvalues found through the SDP with those found by just checking the minimum eigenvalue for that energies in figure 2.

Anharmonic Oscillator The process is very similar to the harmonic oscillator, with only the initial search-space and recursive relation changing.
The Hamiltonian now is

$$\hat{H} = \hat{p}^2 + g\hat{x}^2 + \hat{x}^4 \tag{2.2.3}$$

with the search-space being

$$\mathcal{E} = \{E, \langle x^2 \rangle\} \tag{2.2.4}$$

The procedure continues as before:

$$\begin{aligned} &\text{Minimize: } -t \\ &\text{Subject to: } M^{(L)}(E, g, \langle x^2 \rangle) - tI \succeq 0 \\ &\text{With Search-space: } \mathcal{E} = \{E, t, \langle x^2 \rangle\} \end{aligned}$$

As before, we can check which energies are allowed (see figure 3). This time the search space (excluding t) is 2-dimensional, so we can also check the allowed region in this space and compare it to what we would find if we tried pairs of (E,$\langle x^2 \rangle$) by hand and checked if $\min \text{Spec}(M^{(L)}(E, g, \langle x^2 \rangle)) \geq 0$ (figure 4).

2.2.2 Thermal Case

This section closely follows [10]. Moving to the thermal case is done by introducing a new constraint, the KMS condition:

$$\langle \mathcal{O}_1 \mathcal{O}_2 \rangle_\beta = \langle \mathcal{O}_2 \, e^{-\beta H} \mathcal{O}_1 \, e^{\beta H} \rangle_\beta \tag{2.2.5}$$

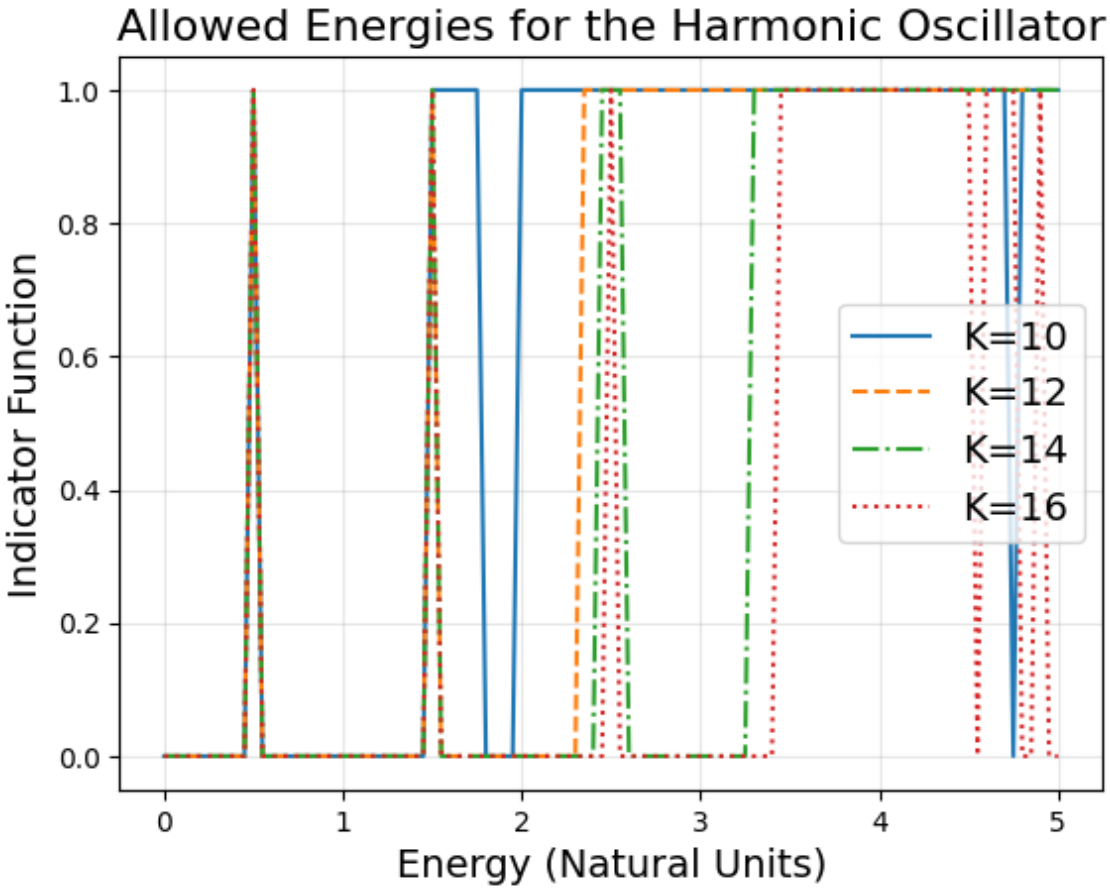

Figure 1: Here we see the allowed energies as found through the SDP by checking that $t_\star \geq 0$. We see that as K increases, the allowed regions narrow down to the allowed half integer values. The y-axis is the values of the indicator function, $\mathbb{1}_S(E)$ where $S = \{E|\ t_\star(E) \geq 0\}$ is the set of allowed energies for a given depth.

where:

$$\langle \mathcal{O} \rangle_\beta := \mathrm{Tr}_\mathcal{H}\left(\frac{e^{-\beta H}}{\mathrm{Tr}_\mathcal{H}(e^{-\beta H})}\mathcal{O}\right) \tag{2.2.6}$$

In the literature (see for example [13, 14, 15]), it has been rigorously shown that the KMS condition is equivalent to:

$$\beta C - A^{1/2}\log(A^{1/2}B^{-1}A^{1/2})A^{1/2} \succeq 0 \tag{2.2.7}$$

with:

$$A_{ij} := \langle \mathcal{O}_i^\dagger \mathcal{O}_j \rangle_\beta, \quad B_{ji} := \langle \mathcal{O}_j \mathcal{O}_i^\dagger \rangle_\beta, \quad C_{i,j} := \langle \mathcal{O}_i^\dagger [H, \mathcal{O}_j] \rangle_\beta \tag{2.2.8}$$

However 2.2.7 is not easy to implement as is. We can get around this by relaxing the constraint in a rigorous way such that the set of allowed points we find through

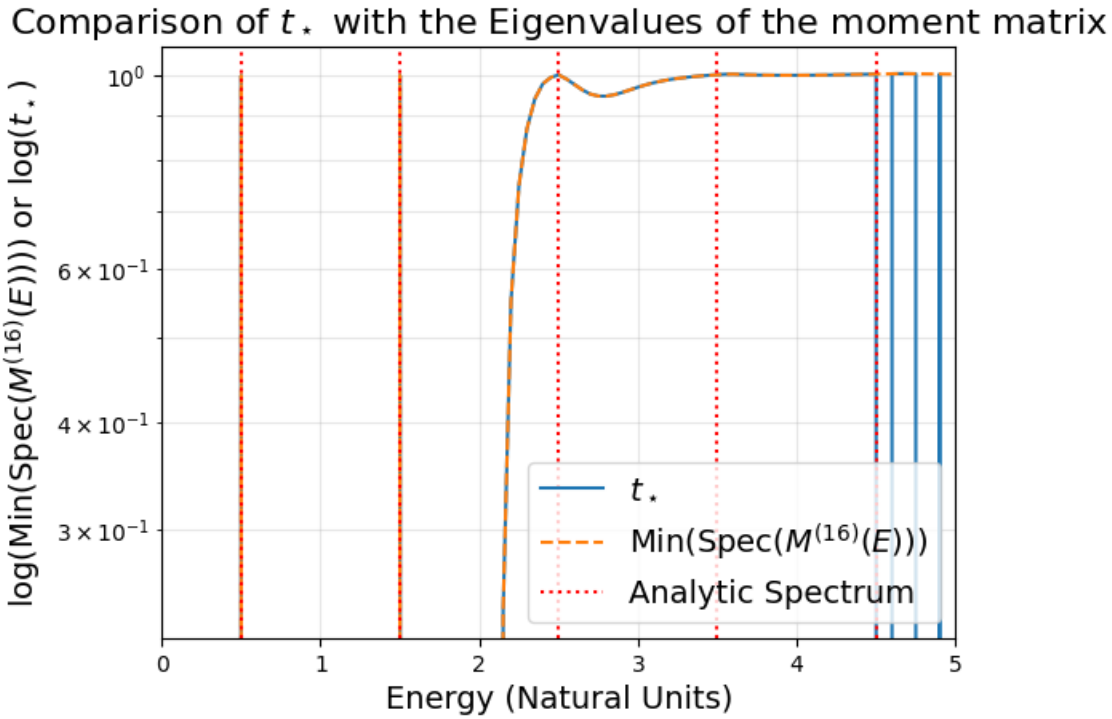

Figure 2: We see that the minimum eigenvalue found through the SDP always outlines the eigenvalue computed numerically for a given size of matrix. With dashed red lines, we show the analytic spectrum

the SDP always include the points we would find if we were to use 2.2.7 As in [10], we employed a formal relaxation of the logarithm as follows:

$$\log(x) \approx r_{k,m}(x) := 2^k r_m(x^{1/2^k}) \tag{2.2.9}$$

with $r_m(x) = \sum_{j=1}^{m} w_j f_{t_j}(x)$, $f_t(x) = \frac{x-1}{t(x-1)+1}$. The w_j are the weights of the Gauss-Radau quadrature[1], and the t_j are the abscissas of the quadrature with $t_1 = 0$. In practice, one needs to calculate the weights and abscissas only for implementing the KMS condition, as we will illustrate shortly. Using the approximated logarithm, the KMS condition is now equivalent to:

$$\beta C - A^{1/2} r_{k,m}(A^{1/2} B^{-1} A^{1/2}) A^{1/2} \succeq 0 \tag{2.2.10}$$

[1]During implementation, we consulted this website[16].

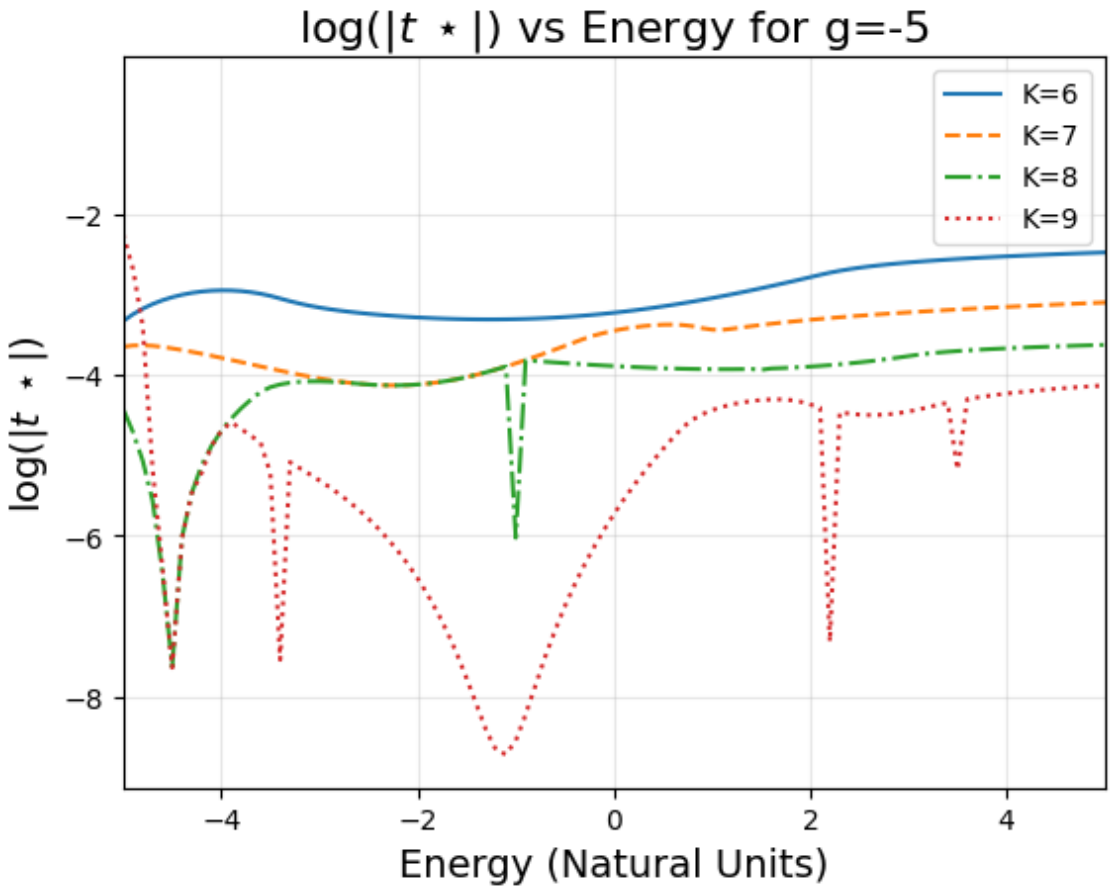

Figure 3: Here we are plotting the energy range versus $\log|t_\star|$. Each peak corresponds to a 0 crossing meaning that two peaks mark the start and end of an allowed region. As K increases those peaks get closer and closer until they converge to the allowed energy.

Which in turn is equivalent to the constraints [10]:

$$Z_0 = B,\ \sum_{j=1}^{m} w_j T_j = \beta C,\ \begin{bmatrix} Z_i & Z_{i+1} \\ Z_{i+1} & A \end{bmatrix} \succeq 0 \text{ and } \begin{bmatrix} Z_k - A - T_j & -\sqrt{t_j}T_j \\ -\sqrt{t_j}T_j & A - t_j T_j \end{bmatrix} \succeq 0 \tag{2.2.11}$$

With $Z_i, T_j \in H^n$ and the indices running from 0 to k-1 for i and from 1 to m for j meaning there are k+1 Z matrices and m T matrices. The canonical relations constraining the expectation values of the operator-words, as mentioned before, is equivalent to rewriting all operators such that $\hat{x}$ and it's powers always come before $\hat{p}$ and it's powers or inversely. Meaning that the entries of M are now a linear

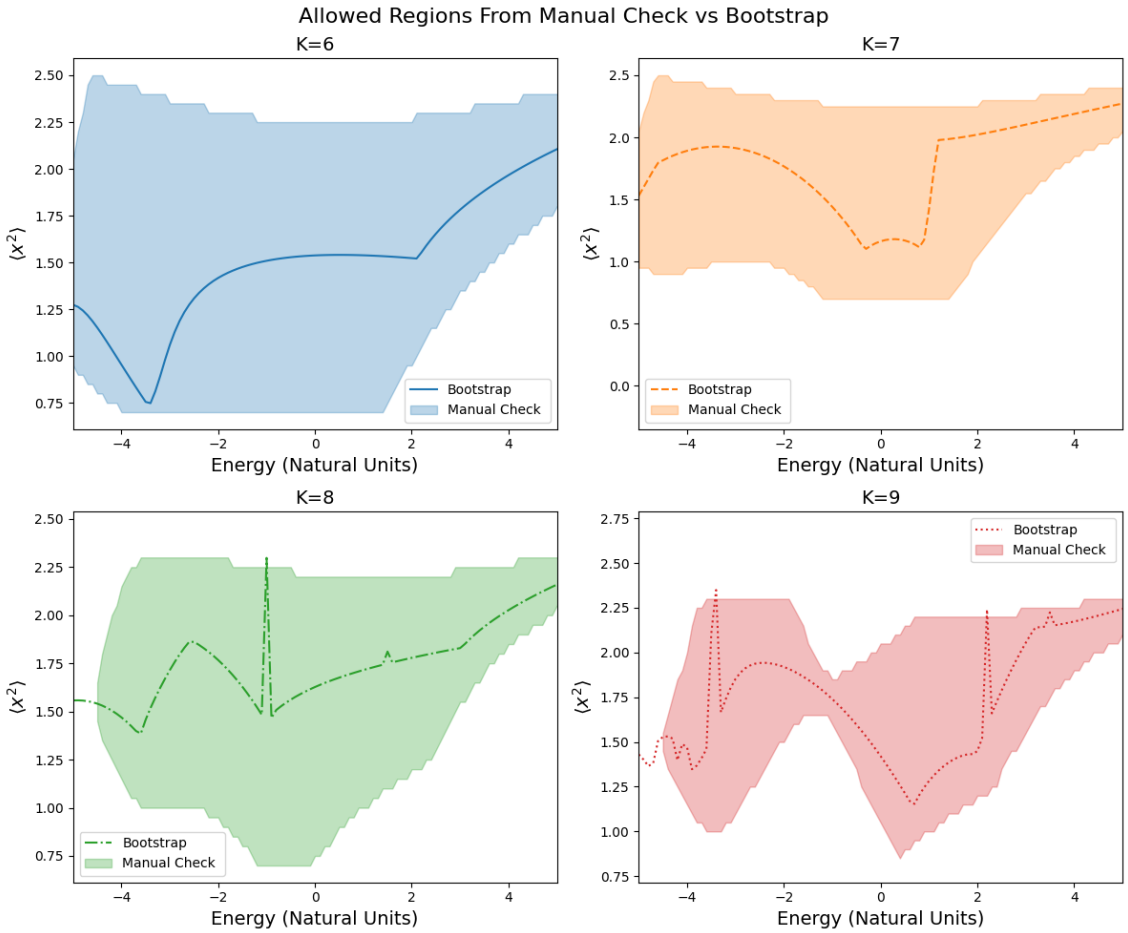

Figure 4: The lower bound for x2 computed using the bootstrap is always within the allowed region found by simply checking the space by hand for the same matrix size.

combination of expectation values of $\langle x^l p^{l'} \rangle$ for some integers l and l'. The final SDP problem then, for a given word length L greater than the length of operators in H, is:

$$\text{minimize/maximize:} \quad \langle H \rangle \tag{2.2.12}$$

$$\text{Subject to:} \tag{2.2.13}$$

$$M^{(L)} \succeq 0 \tag{2.2.14}$$

$$\langle [H, \mathcal{O}] \rangle_\beta = 0 \; \forall \mathcal{O} \in \mathcal{B}_{L-2} \tag{2.2.15}$$

$$\text{relation 2.2.11 with } A = A_{ij}^{(L)} := \langle \mathcal{O}_i^\dagger \mathcal{O}_j \rangle_\beta,$$
$$B = B_{ji}^{(L)} := \langle \mathcal{O}_j \mathcal{O}_i^\dagger \rangle_\beta, \quad C = C_{i,j}^{(L)} := \langle \mathcal{O}_i^\dagger [H, \mathcal{O}_j] \rangle_\beta \;\; \forall \mathcal{O}_i, \mathcal{O}_j \in \mathcal{B}_{L/2-2} \tag{2.2.16}$$

The final search-space consists of the independent $\langle \mathcal{O} \rangle_\beta$ and the T and Z matrices. Below, we can see the above SDP implemented for the Harmonic Oscillator (figure 5) and the Quartic Oscillator with Hamiltonian: $\hat{H} = \hat{p}^2 + \hat{x}^4$ (figure 6)

Here we would also like to mention that the thermal case can be treated using the Quantum Information Conic Solver (QICS)[17][2]

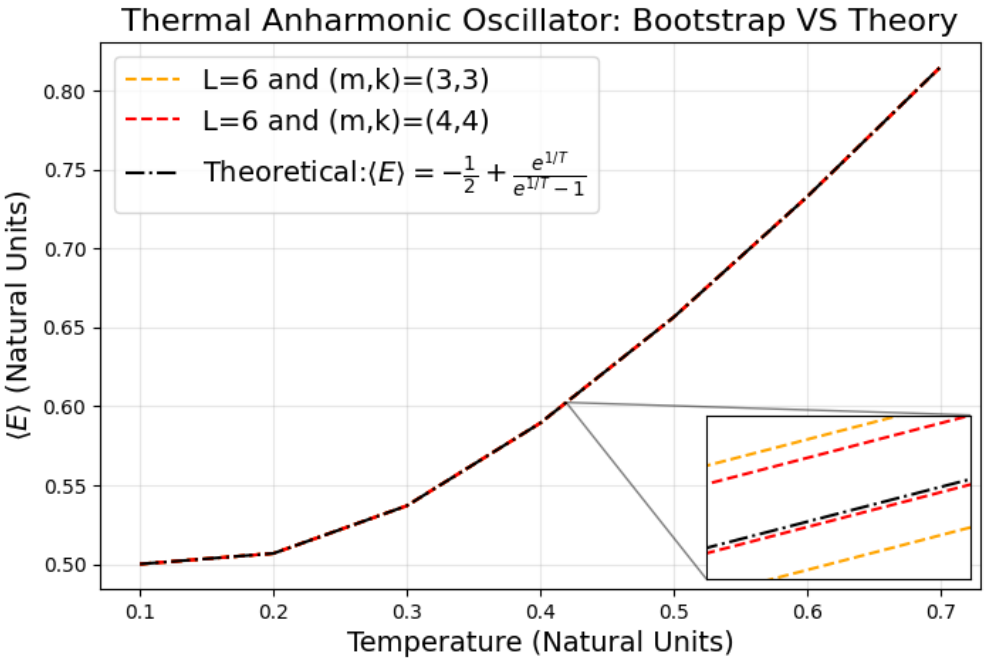

Figure 5: We have plotted the SDP vs the theoretical value for the expectation value of the energy for the harmonic oscillator $\langle E \rangle = -\frac{1}{2} + \frac{e^{1/T}}{e^{1/T}-1}$ we see that even for as low as L=6 the convergence to the theoretical value is very good with more convergence as the relaxation parameters increase. The bootstrap values were computed using `CVXPY`[18] with the `CLARABEL` solver [19].

2.3 Implementation

For the 0 temperature cases, we implemented the harmonic oscillator in Mathematica, then exported the results into JSON files and plotted the final plots there, while for the anharmonic case we implemented everything in python (although a version of the code in Mathematica is also available).

[2]We thank Minjae Cho for their correspondence on this matter

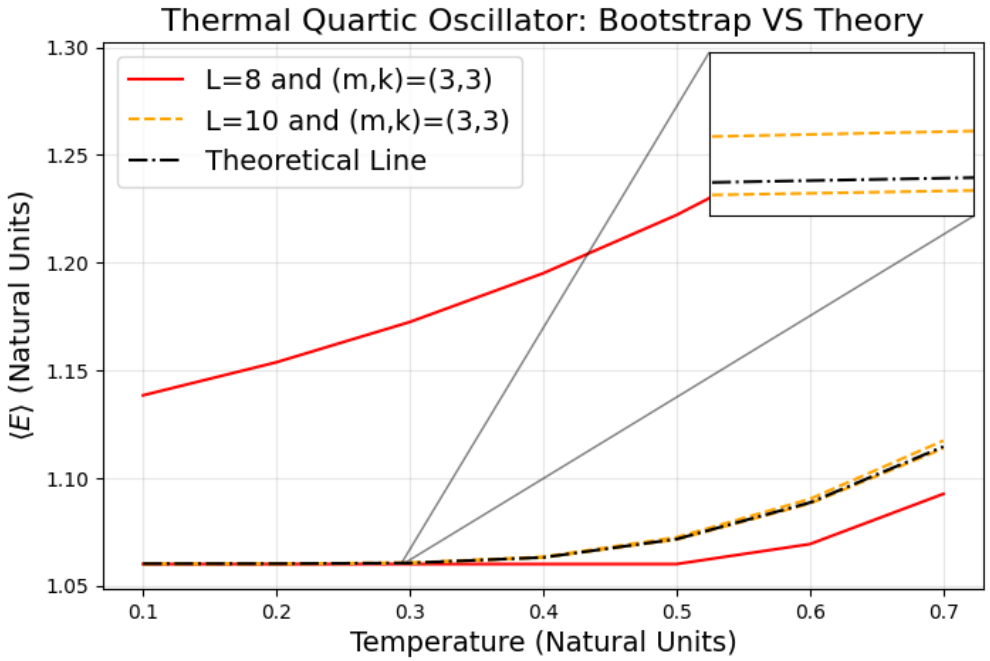

Figure 6: Plot of the expectation value of the energy vs the temperature for the bootstrap at L=8,10 and relaxation variables (m,k)=(3,3). Theoretical value computed through Hamiltonian diagonalization. For the quartic we see that it takes longer to converge to the expected value, but quickly converges for L=10. The SDP was solved with `SDPA-MULTIPRECISION` using the `sdpa-python` wrapper[20, 21, 22, 23]. For this wrapper to use `SDPA-MULTIPRECISION` one would have to build it from source. For more on this, refer to wrapper's documentaion

For the thermal cases, we used Mathematica for simplifying and generating the moment matrices and then exported them as JSON files and continued with python for the SDP, using the CVXPY[18] package with the SDPA-MULTIPRECISION solver through the `sdpa-multiprecision` wrapper [20, 21, 22, 23] for the Quartic Oscillator and with the CLARABEL[19] solver for the Harmonic.

In our final implementation we could simply input the matrices, even having complex entries, into the programmes, but we stumbled upon avenues where that would not have been possible. In such cases, one can translate any Hermitian SDP into a Real Symmetric Matrix SDP as such:

$$H = \Re(H) + i\Im(H) \succeq 0,\ H \in H^n \tag{2.3.1}$$

Where:

$$\Re(H)^T = \Re(H) \tag{2.3.2}$$
$$\Im(H)^T = -\Im(H) \tag{2.3.3}$$

Is equivalent to:

$$S_H = \begin{bmatrix} \Re(H) & \Im(H) \\ -\Im(H) & \Re(H) \end{bmatrix} \succeq 0,\ S_H \in S^{2n} \tag{2.3.4}$$

For more information on this refer to the MOSEK Modeling Cookbook[3].

This technique was used to write some ultimately unused and incomplete code that handles JSON imported matrices in a way that they can be exported into SDPA-GMP[24] friendly files such that the SDP can be executed through the terminal. We nevertheless included this code in our GitHub in case someone might find it useful.

All code and any data generated is available in our GitHub[4]. For an in-depth explanation of the code implementation refer to B.

3 The BFSS Model

Having seen how the quantum mechanical bootstrap works in some simple examples for both the 0 temperature case and the thermal case, we will now shift to why and how this technique can be of interest for High Energy Physics by focusing on the BFSS conjecture and its relation to Matrix Theory. We will outline a derivation of getting from a non-abelian extension of the DBI action to the Matrix Model by following the derivations done in [11]. We also consulted [25, 5]. For reviews on the Matrix Model refer to [11, 26, 5, 25] and for a more complete treatise of matrix theories refer to [27].

Stated, the conjecture reads:

"[...]M-theory in the infinite momentum frame is exactly equivalent to the $N \to \infty$ limit of the supersymmetric quantum mechanics described by [of 10D SYM compactified to 0+1 Dimensions]. The calculation of any physical quantity in M-theory can be reduced to a calculation in matrix quantum mechanics followed by an extrapolation to large N"

-Banks, Fischler ,Shenker, Susskind in [1] section 5.

3.1 TypeIIA, M-Theory and Kaluza-Klein compactification

10-dimensional TypeIIA string theory in the strongly coupled regime can be identified with an 11-dimensional theory named M[5] Theory[28]. The identification comes

[3] https://docs.mosek.com/modeling-cookbook/sdo.html
[4] https://github.com/pripkand/master_thesis.git
[5] M for Mother, Matrix, Magic, Membrane, Mystery etc.

from the fact that, were this to be the case, the string coupling would be related to the size of the extra dimension and so in the case where the theory was compactified (weak coupling) the coupling would be finite while as the coupling increases the size of the extra dimension would as well resulting in the decompactification of the 11^{th} dimension.

Importantly, the objects postulated to populate M Theory (M2 branes and M5 branes) produce most of the known objects of TypeIIA string theory such as D2-branes and, more importantly for what is to follow, D0-branes which come from the massive modes that correspond to the compactified dimension. Since the number of branes corresponds to massive Kaluza-Klein (KK) modes, accessing a limit where we have $N \to \infty$ such modes we access the uncompactification limit and thus M theory. We will now focus on how to describe N D0-branes in this limit.

3.2 Non-Abelian Bosonic DO-branes

The strategy in this section will be to describe the behaviour of N D0-branes as we compactify M-theory on a circle. The derivation we follow is that of [11] section 3.2. Starting from the extended DBI action[6]

$$S = -T_{D0}\int d\tau\ \mathrm{STr}(e^{-\phi}\sqrt{\det Q^i_j}\sqrt{-P[g_{00}] - P[E_{0i}](Q^{-1}-\delta)^i{}_k E^{kj} P[E_{j0}]}) \tag{3.2.1}$$
$$+T_{D0}\int \mathrm{STr}(P[e^{i\lambda^{-1}\iota_X\iota_X}(\sum_n A_{(n)}e^B)])$$

Where ϕ is the dilaton,

$$Q^i{}_j := \delta^i_j + i\lambda^{-1}[X^i, X^k]E_{kj} \tag{3.2.2}$$
$$\lambda := 2\pi l_s^2 \tag{3.2.3}$$
$$E_{\mu\nu} := g_{\mu\nu} + B_{\mu\nu} \tag{3.2.4}$$

B is the NSNS field that couples to the RR p-forms. $P[g_{00}] = g_{00}D_\tau X^0 D_\tau X^0 + 2g_{0i}D_\tau X^0 D_\tau X^i + g_{ij}D_\tau X^j D_\tau X^i$ and $P[E_{0i}] = E_{0i} + E_{ji}D_\tau X^j$ are the pullbacks of g_{00} and E_{0i} to the wordline. Lastly, i_X is the contraction operator with X such that $(\iota_X C_{(p)})\mu_1...\mu_{p-1} = X^j C_{j\mu_1...\mu_{p-1}}$ and STr is the supertrace. The Latin indices range from 1 to 9 and are the transverse directions of the target space.

Interpretively, the first term of the action is the worldvolume of the worldline of the D0 branes: the $\sqrt{detQ^i_j}$ factor is the non-abelian contribution of the NSNS field

[6] There are ways of arriving to the same results without using this action at all. For more, refer to [25]

to the geometry a D0-brane "sees" while the second square root is the worldvolume and the second term is the interactions between the RR p-forms with the NSNS field. For what follows, we are only interested in the flat, weak field limit of this action during compactification, namely:

$$S \approx -T_{D0} \int d\tau \mathrm{STr}(\sqrt{det(\delta^i_j + i\lambda^{-1}[X^i, X_j])}\sqrt{-P[g_{00}]}) + \int d\tau \mathrm{STr}(P[A_{(1)}]) \tag{3.2.5}$$

In the section that follows we will go over some arguments to find the limit of 3.2.5 during an infinite boost to go to the infinite momentum frame.

3.3 Infinite Boosts and Null Compactification

This section closely follows [11] section 3.2.

Suppose we have M-theory in a $\mathbb{R}^{1,9} \times S^1$ background compactified along the spatial dimension z that corresponds to S^1 with radius R_z, meaning:

$$z \sim z + 2\pi R_z \tag{3.3.1}$$

The line element is:

$$ds^2 = -dt^2 + dz^2 + ds^2(\mathbb{R}^9) \tag{3.3.2}$$

Where $ds^2(\mathbb{R}^9) := \sum_{i=1}^{9} dx^i dx^i$ is the line element of flat Euclidean space. Now, suppose we boost to a frame with velocity $\beta \to 1$ in the z direction. The coordinates change like:

$$\bar{z} = \gamma(z - \beta t) \tag{3.3.3}$$
$$\bar{t} = \gamma(t - \beta z) \tag{3.3.4}$$

Where $\gamma = \frac{1}{\sqrt{1-\beta^2}}$ is the Lorentz factor. Now, the periodicity condition 3.3.1 becomes:

$$\bar{z} \sim \bar{z} + 2\pi\gamma R_z \tag{3.3.5}$$
$$\bar{t} \sim \bar{t} - 2\pi\gamma R_z \tag{3.3.6}$$

Since we are interested in null compactification, it is convenient to work in lightcone coordinates, defined as:

$$\bar{x}^{\pm} := \frac{1}{\sqrt{2}}(\bar{t} \pm \bar{z}) \tag{3.3.7}$$

The new line element now is:

$$ds^2 = -2d\bar{x}^+ d\bar{x}^- + ds^2(\mathbb{R}^9) \tag{3.3.8}$$

Again, due to 3.3.7 the periodicity condition becomes:

$$\bar{x}^+ \sim \bar{x}^+ + \frac{2\pi R_z}{\sqrt{2}}\sqrt{\frac{1-\beta}{1+\beta}} \tag{3.3.9}$$

$$\bar{x}^- \sim \bar{x}^- - \frac{2\pi R_z}{\sqrt{2}}\sqrt{\frac{1+\beta}{1-\beta}} \tag{3.3.10}$$

Changing now coordinates:

$$x^- = \bar{x}^- \tag{3.3.11}$$

$$x^+ = \bar{x}^+ + \bar{x}^- \frac{1-\beta}{1+\beta} \tag{3.3.12}$$

Giving:

$$x^- \sim x^- \frac{2\pi R_z}{\sqrt{2}}\sqrt{\frac{1+\beta}{1-\beta}} \tag{3.3.13}$$

$$x^+ \sim x^+ \tag{3.3.14}$$

Importantly, this transformation changes the line element by adding a $(dx^-)^2$ term:

$$ds^2 = -2dx^+dx^- + 2\frac{1-\beta}{1+\beta}(dx^-)^2 + ds^2(\mathbb{R}^9) \tag{3.3.15}$$

We define now $R := \frac{R_z}{\sqrt{2}}\sqrt{\frac{1+\beta}{1-\beta}}$ to be fixed as $\beta \to 1$ meaning that this limit is equivalent to $R_z \to 0$. We can rescale $x^- \mapsto x^- R$ to rewrite the line element in the more suggestive form:

$$ds^2 = 2R^2\frac{1-\beta}{1+\beta}(dx^- - \frac{1}{2R}\frac{1+\beta}{1-\beta}dx^+)^2 - \frac{1}{2R^2}\frac{1+\beta}{1-\beta}(dx^+)^2 + ds^2(\mathbb{R}^9) \tag{3.3.16}$$

This line element is in the form of the Kaluza-Klein metric ansatz A.0.7 with:

$$A_\mu dx^\mu = -\frac{1}{2R}\frac{1+\beta}{1-\beta}dx^+ \tag{3.3.17}$$

$$ds^2_{10} = g_{\mu\nu}dx^\mu dx^\nu = -\frac{1}{2R^2}\frac{1+\beta}{1-\beta}(dx^+)^2 + ds^2(\mathbb{R}^9) \tag{3.3.18}$$

$$\phi = 2R^2\frac{1-\beta}{1+\beta} =: R_-^2 \equiv (l_s g_s)^2 \tag{3.3.19}$$

We can parametrize the boost as $\beta \mapsto 1 - \frac{1}{\omega^2}$, giving:

$$A_{(1)} \approx -\frac{\omega^2}{R} dx^+ \tag{3.3.20}$$

$$ds^2_{10} \approx -\frac{\omega^2}{R^2}(dx^+)^2 + ds^2(\mathbb{R}^9) \tag{3.3.21}$$

$$g_s \approx \omega^{-3/2}\hat{g}_s, \;\; \hat{g}_s := (R/l_p)^{3/2} \tag{3.3.22}$$

$$l_s \approx \omega^{1/2}\hat{l}_s, \;\; \hat{l}_s := l_p^{3/2} R^{-1/2} \tag{3.3.23}$$

$$\lambda = 2\pi l_s^2 \approx \hat{\lambda}\omega, \;\; \hat{\lambda} := 2\pi \hat{l}_s^2 \tag{3.3.24}$$

Where the hatted quantities are held fixed as $\omega \to \infty$. We can, now, compute the limiting behaviour of the action 3.2.5:

$$\begin{aligned}
S &\approx -T_{D0} \int d\tau \mathrm{STr}(\sqrt{det(\delta^i_j + i\lambda^{-1}[X^i, X_j])}\sqrt{-P[g_{00}]}) + \int d\tau \mathrm{STr}(P[A_{(1)}]) \\
&\approx -T_{D0} \int d\tau \mathrm{STr}(\sqrt{det(\delta^i_j + i\omega^{-1}\hat{\lambda}^{-1}[X^i, X_j])}\sqrt{\frac{\omega^2}{R^2} D_\tau X^0 D_\tau X^0 - D_\tau X^j D_\tau X^j}) \\
&\qquad + \int d\tau \mathrm{STr}(A_0 D_\tau X^0 + A_i D_\tau X^i) \\
&\approx -\hat{T}_{D0} \frac{\omega^2}{R} \int d\tau \mathrm{STr}(\sqrt{1 - 1/2\omega^{-2}\hat{\lambda}^{-2}\mathrm{Tr}([X^i, X^j][X_i, X_j]) + \mathcal{O}(\omega^{-3})}\sqrt{1 - \frac{R^2}{\omega^2} D_\tau X^j D_\tau X^j}) \\
&\qquad + \int d\tau \mathrm{STr}(-\frac{\omega^2}{R}) \\
&\approx -\hat{T}_{D0} \frac{\omega^2}{R} \int d\tau \mathrm{STr}\left[(1 + 1/4\omega^{-2}\hat{\lambda}^{-2}\mathrm{Tr}([X^i, X^j][X_i, X_j]) + \mathcal{O}(\omega^{-4}))(1 - \frac{1}{2}\frac{R^2}{\omega^2} D_\tau X^j D_\tau X^j)\right] \\
&\qquad - \int d\tau \mathrm{STr}(\frac{\omega^2}{R}) \\
&\approx \hat{T}_{D0} \int d\tau \mathrm{Tr}(\frac{R}{2} D_\tau X^i D_\tau X^i + \frac{\hat{\lambda}^{-2} R^{-1}}{4} \mathrm{Tr}([X^i, X^j][X_i, X_j]) + \mathcal{O}(\omega^{-2}))
\end{aligned}$$

Where in between the 2nd and 3rd line we used $det(I + A) \approx 1 + TrA + \frac{1}{2}((TrA)^2 - Tr(A^2))$ and we chose X^0 s.t. $D_\tau X^0 = 1$ as well as pulled a factor of ω out of the tension. The last line is, to leading order, the bosonic part of the Matrix Theory action. We can rewrite the Lagrangian in Hamiltonian form as:

$$H = \mathrm{Tr}(\frac{1}{2}(P^i)^2 - \frac{1}{4}[X^i, X^j]^2 + \frac{1}{2}\psi_\alpha \gamma^i_{\alpha\beta}[X_i, \psi_\beta]) \tag{3.3.25}$$

We also added the 16 fermionic matrices, ψ, that transform under an SO(9) transformation. The system also includes 16 Hermitian supercharges $\mathcal{Q}_a$. The Hamiltonian is U(N) invariant.

3.4 Some Known Properties of the Matrix Model

For reviews on the Matrix model, refer to [26, 25]

We will now list some known properties of the matrix model.

Coordinate Interpretation of X Let us start by focusing on the interpretation of the X^i matrices as the coordinates of the D0-branes. More precisely, if we have NxN matrices then the 9 X^i matrices' N eigenvalues describe the N 9-dimensional position vectors of the N D0-Branes, i.e. the n^{th} eigenvalue of X^i is the i^{th} component of the n^{th} brane[25, 26].

The off diagonal terms relate the interactions between different branes which increase in importance as the branes get closer together since the open strings that end on them stretch between different branes [25]. The trace of each matrix corresponds to the centre of mass coordinates. If we instead worked with traceless X matrices, then the Hamiltonian would have an SU(N) symmetry instead of U(N)[25, 5].

If the X matrices have the block diagonal form, $X = \text{diag}(x_1, ..., x_n)$ s.t. $\sum_i^n \dim(x_i) = N$ then we have a configuration of separated D0 clusters whose in-between distance is [25]:

$$r_{ij} = |\frac{\text{Tr}(x_i)}{\dim(x_i)} - \frac{\text{Tr}(x_j)}{\dim(x_j)}| \tag{3.4.1}$$

Spectrum It is known that the system without the centre of mass has bound states [29, 30, 31] in which the energy is just the centre of mass energy. These states correspond to supergraviton states. We can also describe an arbitrary number of supergravitons by considering block diagonal matrices X which correspond to block diagonal Hamiltonians of the same form with interactions between supergravitons defined by any nigh-vanishing off diagonal terms[25, 11, 5].

Supermembranes Aside from gravitons, we can also probe other objects of M theory such as supermembranes. It has been shown that 3.3.25 can be obtained from the quantisation of supermembranes[1, 11, 25, 26, 32]. We would like to illustrate a heuristic hint for this. Were we to consider a single D0-brane whose coordinates are Abelian, it would only couple to a 1-form. However, when considering N, non-abelian D0-branes, the action 3.2.1 also includes terms of the form:

$$\iota_X \iota_X A_{(1)} \wedge B \tag{3.4.2}$$

Which allow the D0-branes to act like a D2-brane by effectively coupling to a 3-form. This is only possible because the coordinates X are non-abelian[7].

[7] This is known as the Myers effect[33]

There are also several interesting interpretations of the Matrix Model if one views its relation with M Theory through the AdS/CFT correspondence. For more, refer to [5].

4 Bootstraps of the BFSS Model

In this section, we will briefly illustrate some approaches that have been taken to bootstrapping the model defined by 3.3.25. We will mostly follow [5].

The bootstrap approach works well for this case because

- In contrast with methods like Monte-Carlo, Euclidean signature sign problems are of no concern.
- We can work in limits where $N \to \infty$ directly but can also deal with finite N cases.
- The method is non-perturbative/works with strong coupling.

As we saw before, we can produce rigorous inequalities for the values of observables, provided we can deal with the steeply increasing computational cost.

4.1 Parallels and Differences

As before, the system will be constrained by the Schwinger-Dyson equation but we will be interested in its application to single trace operators i.e.:

$$\langle [H, \mathrm{Tr}\mathcal{O}] \rangle_\beta = 0, \quad \mathcal{O} \in \mathcal{B}_L \tag{4.1.1}$$

With the Hankel Matrix changing accordingly to:

$$(M^{(L)})_{ij} := \langle \mathrm{Tr}\mathcal{O}_i\mathcal{O}_j \rangle_\beta, \quad \mathcal{O}_{i,j} \in \mathcal{B}_L \tag{4.1.2}$$

While before we only had the canonical commutation relations, now we also have the cyclicity of the trace over the SU(N) indices of the matrices. For the thermal case, as before, we just need to also impose the KMS condition in much the same way as it was done above. One can also include SUSY constraints by noting that

$$\{\mathcal{Q}_a, \mathcal{Q}_b\} \propto \delta_{ab} H \tag{4.1.3}$$

meaning we can bootstrap a SUSY invariant state by imposing:

$$\langle \mathcal{Q}_a \mathcal{O} \rangle_\beta = \langle \mathcal{O}\mathcal{Q}_a \rangle_\beta = 0 \tag{4.1.4}$$

4.2 Literature Implementations

There has been a lot of recent work focusing on both 3.3.25 and other matrix models when it comes to bootstrapping. Recent work done in [2] derived non-trivial bounds on $\langle \mathrm{Tr} X^2 \rangle$ as well as improving Polchinski's Virial Theorem bound [34]. A more in depth implementation can be found in [3] which utilizes SO(9) block decomposition (for an introduction on this refer to [5] section 6.4.1)

Implementations on other matrix models can be found in [35](application to random matrix models), [36] (application to multi-matrix models), [37] (bootstrap used for finding critical exponents in matrix theories) and more.

5 Discussion

Motivated by its application to the matrix model of the BFSS conjecture, we illustrated the idea of the quantum mechanical non-negative bootstrap in a few simple cases for both 0 temperature and thermal cases and saw that the bootstrap method using SDPs can yield satisfying results for modest computational power. We outlined some arguments for getting to the matrix model Hamiltonian 3.3.25 through the BFSS conjecture and mentioned how the bootstrap methodology could be applied to it.

Potential Directions

- We would have liked to do a bootstrap of the matrix model ourselves, preferably in the thermal case, trying either the SDP route as illustrated or using QICS[17].
- We would, also, like to try implementations in other cases such as CFTs or Yang-Mills theories.
- Refine the code that was used to cover more cases and serve better as an introduction to the area.

Acknowledgements

I sincerely thank my advisor, dr. Vasilis Niarchos, for his patience and guidance. I also want to thank friends and family with help in proofreading, and my partner, Afroditi Aretaki, for her support and understanding.

I would, also, like to thank Minjae Cho for relevant, helpful correspondence.

A Kaluza-Klein Reduction

Central to he BFSS conjecture is the Kaluza-Klein Reduction of M theory on a circle, yielding TypeIIA, which compactifies the extra dimensions. In this section, we will illustrate the idea of Kaluza-Klein reduction by looking at 2 examples both of which are cases of compactification on a circle (S^1).

Real Massless Scalar Field Suppose we have d-dimensional spacetime on a Manifold: $\mathbb{R}^{d-2,1} \times S^1$. Assigning a coordinate z to the S^1 direction, all objects in the theory will be periodic in z. Given the d-dimensional action of a massless Klein-Gordon Field:

$$S[\phi] = \int d^d x \frac{1}{2} \partial_M \phi(x^\mu, z) \partial^M \phi(x^\mu, z) = \int d^d x \frac{1}{2} (\partial_\mu \phi(x^\mu, z) \partial^\mu \phi(x^\mu, z) + \partial_z \phi(x^\mu, z) \partial^z \phi(x^\mu, z)) \quad \text{(A.0.1)}$$

Where with capital indices we are referring to d-dimensions and with lowercase to d-1-dimensions. Since ϕ is periodic in z we can expand it in Fourier modes as:

$$\phi(x^\mu, z) = \int dk \; \tilde{\phi}^{(k)}(x^\mu) e^{i\omega(k) z} \quad \text{(A.0.2)}$$

Where $\omega(k) := \frac{2\pi}{R_z} k$ and we have taken the principal branch of the complex logarithm such that $e^{i\omega(k)(z+R_z)} = e^{i\omega(k)z}$. By expanding inside the the action we have:

$$S[\phi] = \int d^d x \int dk \int dk' \frac{1}{2} (\partial_\mu \tilde{\phi}^{(k)}(x^\mu) \partial^\mu \tilde{\phi}^{(k')}(x^\mu) - \omega(k)\omega(k') \tilde{\phi}^{(k)}(x^\mu) \tilde{\phi}^{(k')}) e^{i(k+k')\frac{2\pi}{R_z} z} \quad \text{(A.0.3)}$$

By integrating over z we will get a delta function of the form $\delta(k + k')$ giving:

$$S[\phi] = \int d^{d-1} x \int dk \frac{1}{2} (\partial_\mu \tilde{\phi}^{(k)}(x^\mu) \partial^\mu \tilde{\phi}^{(-k)}(x^\mu) - \omega(k)\omega(-k) \tilde{\phi}^{(k)}(x^\mu) \tilde{\phi}^{(-k)}(x^\mu)) \quad \text{(A.0.4)}$$

Since $\phi : \mathbb{R}^{d-2,1} \times S^1 \to \mathbb{R}$ we also have the reality condition:

$$\int_{-\infty}^{\infty} dk \tilde{\phi}^{(k)}(x^\mu) e^{i\frac{2\pi}{R_z} kz} = \int_{-\infty}^{\infty} dk \tilde{\phi}^{(k)}(x^\mu) e^{-i\frac{2\pi}{R_z} kz} = \int_{-\infty}^{\infty} dk \tilde{\phi}^{(-k)}(x^\mu) e^{i\frac{2\pi}{R_z} kz} \implies \tilde{\phi}^{(k)} = \tilde{\phi}^{(-k)} \quad \text{(A.0.5)}$$

Giving:

$$S[\phi] = \int dk \int d^{d-1} x \frac{1}{2} (\partial_\mu \tilde{\phi} \partial^\mu \tilde{\phi} + m^2 \tilde{\phi} \tilde{\phi}) \quad \text{(A.0.6)}$$

Where $m^2 := \omega(k)^2 = (\frac{2\pi}{R_z}k)^2$. The action above integrates over the parameter k meaning we are, effectively "summing" over all actions of tachyonic Klein-Gordon Fields. As we take, however the radius of the extra dimension to zero, only the term with k=0 remains finite. This consortium of modes is known as the Kaluza-Klein tower.

Pure Gravity For gravity, we won't go through the trouble of taking the Fourier transform and will instead assume we are only taking the 0 mode, meaning that nothing depends on z explicitly. We can write the line element in the following ansatz:

$$d\hat{s}^2 = g_{\mu\nu}dx^\mu dx^\nu + \phi(dz + A_\mu dx^\mu)^2 \tag{A.0.7}$$

Hats denote D dimensional objects alongside capital indices, while unhated quantities and lower case indices denote D-1 dimensions. For this derivation, we follow closely [38]'s Chapter X Appendix: 1 and Appendix: 2. Ignoring the scalar field for now, we can define the following Veilbeins[8]:

$$\hat{e^a} = e^\alpha \tag{A.0.8}$$

$$\hat{e^z} = dz + A_\mu dx^\mu \tag{A.0.9}$$

We can calculate the spin connection ($\omega^a_{\ b}$) through Cartan's First Structural Equation:

$$\omega^a_{\ b} \wedge e^b = -de^a \tag{A.0.10}$$

The non-vanishing components of the spin connection are:

$$\hat{\omega}^z_{\ b} = \frac{1}{2}F_{ab}e^b \tag{A.0.11}$$

$$\hat{\omega}^a_{\ b} = \omega^a_{\ b} - \frac{1}{2}F^a_{\ b}e^z \tag{A.0.12}$$

Now we can calculate the Riemann Tensor from the spin connection:

$$\hat{R}^A_{\ B} = d\hat{\omega}^A_{\ B} + \hat{\omega}^A_{\ C} \wedge \hat{\omega}^C_{\ B} \tag{A.0.13}$$

Note that we have suppressed the non-Veilbein indices. The Ricci components then are:

$$\hat{R}_{ab} = R_{ab} - \frac{1}{2}F^c_{\ a}F_{cb} \tag{A.0.14}$$

$$\hat{R}_{zz} = \frac{1}{4}F_{ca}F^{ca} \tag{A.0.15}$$

[8]For a brief introduction refer to [39] Appendix J

With the scalar being:

$$\hat{R} = R - \frac{1}{4}F_{ab}F^{ab} \tag{A.0.16}$$

By coordinate invariance arguments illustrated in [38] Chapter X Appendix: 2 which we will not repeat here, the scalar field appears as so:

$$\hat{R} = R - \frac{\phi^2}{4}F_{\mu\nu}F^{\mu\nu} - 2\frac{\Box\phi}{\phi} \tag{A.0.17}$$

The determinant of the D dimensional metric is as follows:

$$\det \hat{g} = \det \phi^2 \det \left(g_{\mu\nu} + A_\mu A_\nu - \phi^2 \frac{A_\mu A_\nu}{\phi^2}\right) = \phi^2 \det g \tag{A.0.18}$$

Inserting the above into the Einstein-Hilbert Action gives the Jordan Action:

$$S_{\text{Jordan}} \approx \int d^{D-1}x\sqrt{-g}(\phi R - \frac{1}{4}\phi^3 F_{\mu\nu}F^{\mu\nu} - 2\Box\phi) \tag{A.0.19}$$

Where we have ignored numerical factors.

B The Code Implementation in Depth and GitHub

Here we will go over the code implementation in detail.

B.1 Harmonic Oscillator at T=0

The relevant files here are "zero-temperature_harmonic_oscillator.nb" and "plotting_harmonic_oscillator.ipynb".

zero-temperature_harmonic_oscillator.nb We implement the recursion relation:

$$n\langle x^n\rangle = 2E(n-1)\langle x^{n-2}\rangle + \frac{1}{4}(n-1)(n-2)(n-3)\langle x^{n-4}\rangle \tag{B.1.1}$$

in the function `reqRel[]` which we, then, use for simplifying the Hankel Matrix on initialisation by repeatedly substituting the solution of B.1.1 with respect to the highest appearing moment in the function `buildMatrix[]`. We, then, solve the SDP using the function `doSDAtK[]` returning the optimisation variables (in this case just the slack variable t), the energy, the minimum eigenvalue computed numerically for the optimal variables and the value of the indicator function $\mathbb{1}_S(E)$ where $S = \{E|\ t_\star(E) \geq 0\}$ is the set of allowed energies for a given depth which is added to the rest of the values through the call of `allowedEnergies[]` from inside `doSDAtK[]` and just checks if $t_\star$ is non-negative. Several results for different Hankel Matrix Sizes are exported to a json file.

plotting_harmonic_oscillator.ipynb We just read off of the json file and plot.

B.2 Anharmonic Oscillator at T=0

The relevant files here are "zero_temperature_anharmonic_oscillator.py" and "plotting_anharmonic_oscillator.ipynb".

zero_temperature_anharmonic_oscillator.py This file is basically an implementation of the same code as zero-temperature_harmonic_oscillator.nb but in python using sympy [40] and cvxpy [18] using the SDPA solver through the sdpa-python wrapper [20, 21, 22, 23].

plotting_anharmonic_oscillator.ipynb Again, this file is only used for making the plots

B.3 Thermal Cases

Both the harmonic and anharmonic oscillator are treated using the same code with the only difference being the input files, which we export from mathematica as we will explain shortly, and the objective function. The relevant files are "matrix_maker.nb", "thermal_bootstrap_header_file.py", "thermal_harmonic_bootstrap.py", "thermal_anharmonic_bootstrap.py" and "plotting_anharmonic.py" and "plotting_harmonic.py" for plotting.

matrix_maker.nb This file basically takes care of constructing the matrices needed for 2.2.11 except for the Z and T matrices which are computed later just before executing the SDP. It takes care of the non commuting algebraic computations to arrive to operator expressions before converting them to expectation value expressions and simplifying the moment matrices. These are then exported into json files. The indepth explanation of the file and the functions is as follows:

1. `generateOperatorSet[]`

   The function `generateOperatorSet[operatorLength, operators]` constructs all noncommutative words up to length L built from a given operator list ($\{x, p\}$).

   This generates the basis $\mathcal{B}_L$ used to define moment matrices later.

2. **Commutation Rules and Hermiticity**

   The algebra is encoded as replacement rules:

$$p**x \to -i + x**p, \tag{B.3.1}$$
$$aj[x] \to x, \tag{B.3.2}$$
$$aj[p] \to p. \tag{B.3.3}$$

This, along with NCAlgebra's `NCExpandReplaceRepeated[]`, assures that all operators are changed into a sum of operators of the form: $x^n p^m$

3. `NCProductList[]`

   `NCProductList[expr]` processes expressions by:

   - Expanding sums,
   - Removing scalar prefactors,
   - Discarding pure constants,
   - Returning only noncommutative monomials.

   Then returns a list only containing the operator monomials present in the expression given.

4. `getSymbols[]`

   `getSymbols[expr]` takes the expression given (assuming it's a monomial of operators) and returns a list containing the operators present in order of appearance.

5. `compress[]`

   Takes a list of words of and returns a shorthand version of the word. In practice, it lowers the power of the operators.

   $$PPP \to P3, \quad PXXPP \to PX2P2 \tag{B.3.4}$$

6. `generateWords[]`

   Called as `generateWords[char_list,L]` returns all possible combination of words with the letters chars_list up to and including length L after passing it through `compress[]`. It is the commuting variables equivalent of `generateOperatorSet[]`

7. `expectationValue[]`

   `expectationValue[operator]` returns the expectation value of the operator given meaning operators such as

   $$\langle xpx \rangle \to \texttt{XPX}$$

   The function:

- Extract operator monomials using `NCProductList[]`,
- Convert them to strings,
- Replace $x \to X$, $p \to P$,
- Interpret the result as a symbolic variable.

This converts operator equations into algebraic equations.

8. `produceSwingerDyson[]`

   The function `produceSwingerDyson[operatorWordList, H]` implements

   $$\langle [H, \mathcal{O}] \rangle = 0$$

   for every operator $\mathcal{O}$ in the basis.

   For each $\mathcal{O}$ and for the given Hamiltonian H:

   (a) Computes $\mathcal{O}H - H\mathcal{O}$,

   (b) Expands using commutation rules,

   (c) Converts to expectation variables using `expectationValue[]`,

   (d) Sets the expression equal to zero.

   This yields linear constraints among moments which we will later solve to produce replacement rules.

9. `solveRelations[]`

   The function `solveRelations[relationList]`:

   - Iteratively solves linear equations,
   - Eliminates higher-order (longer) words first,
   - Identifies independent expectation variables.

   The output consists of substitution rules and the remaining free variables.

10. `finalSearchSpace[]`

    Called as `finalSearchSpace[matrix, independentVariables]`, this function is used to identify variables that didn't appear in the Schwinger-Dyson relations and stores them as independent. It is used exclusively with the Hankel Moment Matrix $M_{ij}^{(L)} := \langle \mathcal{O}_i \mathcal{O}_j \rangle, \quad \mathcal{O}_{i,j} \in \mathcal{B}_L$.

11. **Moment Matrix Constructors**

    Here we will list three related functions that all construct matrices of different forms. Given a basis $\mathcal{B}_L$, the three matrices constructed are:

    - $A_{ij} = \langle O_i^\dagger O_j \rangle$,
    - $B_{ij} = \langle O_i O_j^\dagger \rangle$,
    - $C_{ij} = \langle O_i^\dagger [H, O_j] \rangle$.

    And correspond to the functions:

    - `generateODaggerOMatrix[operatorBasis]`
    - `generateOODaggerMatrix[operatorBasis]`
    - `generateODaggerCommutatorHOMatrix[operatorBasis]`

    accordingly. These are all the moment matrices that appear in the final formulation of the bootstrap.

12. `asignDomains[]`

    Called as `asignDomains[list]`, returns an association (disctionary) with keys being the variables as strings and the values being Booleans on whether the variables are complex or not. This is because, when we export to use this information with cvxpy we will need to define a variable as complex or not using a keyword argument that accepts Booleans.

13. `returnCoefficients[]`

    Central to how the matrices are exported, the function `returnCoefficients[matrix]` returns the matrix-coefficients of the expansion:

    $$M = A_0 + \sum_i A_i \cdot x_i \tag{B.3.5}$$

    In the form of an association as:

    $$\langle |\text{"constnat"} \to \{\Re(A_0), \Im(A_0)\}, \texttt{ToString}[x_i] \to \{\Re(A_i), \Im(A_i)\} | \rangle \tag{B.3.6}$$

    We split between imaginary and real because mathematica and python handle imaginary numbers differently.

14. **Gauss–Radau Approximation of** $\log(x)$

    Since in the KMS condition 2.2.11 only the abscissas and the weights appear, we need only calculate those.

- Abscissas are roots of
$$\frac{P_{m-1}(x) + P_m(x)}{1+x},$$
where P_m are Legendre polynomials.
- Weights are
$$w_j = \frac{1 - x_j}{m^2 P_{m-1}(x_j)^2}.$$
- Nodes are shifted to $[0, 1]$ via $x \mapsto \frac{x+1}{2}$.
- Weights via $w_j \mapsto \frac{w_j}{2}$ with the end point being $\frac{2}{m^2}$

This procedure is taken care of by `getAbscissas[m]`,`getWeight[m,x]`, which implement the corresponding relations from above, and `getQuadrature[m]` which returns a list with the abscissa-weight pairs.

15. **Example Use**

```
(* Anharmonic Oscillator *)

L = 6;
m = 3;
k = 3;

operatorLetters = {x, p};

H := p ** p + x ** x ** x ** x;

BcalLby2        = generateOperatorSet[L/2, operatorLetters];
BcalLminus2     = generateOperatorSet[L - 2, operatorLetters];
BcalLby2minus2  = generateOperatorSet[L/2 - 2, operatorLetters];

(* Produce and Solve the Schwinger-Dyson Relations *)

schwingerDysonRelations = produceSwingerDyson[BcalLminus2, H];

{schwingerDysonReplacementRule, independentVariables} =
  solveRelations[schwingerDysonRelations];

(* Set up the matrices M^(L), A^(L), B^(L), C^(L) *)

MofL = generateODaggerOMatrix[BcalLby2] /.
   schwingerDysonReplacementRule;
```

```
CofL = generateODaggerCommutatorHOMatrix[BcalLby2minus2, H] /.
   schwingerDysonReplacementRule;

AofL = generateODaggerOMatrix[BcalLby2minus2] /.
   schwingerDysonReplacementRule;

BofL = generateOODaggerMatrix[BcalLby2minus2] /.
   schwingerDysonReplacementRule;

(* Get the coefficients of the matrices *)

dictM = returnCoefficients[MofL];
dictA = returnCoefficients[AofL];
dictB = returnCoefficients[BofL];
dictC = returnCoefficients[CofL];

(* Define the search space parameters and constraints *)

EcalL        = finalSearchSpace[MofL, independentVariables];
domainedEcalL = asignDomains[EcalL];

(* Export parameters and matrices as a JSON file *)

Export[
 "/home/pripoll/Documents/Uni_Classes/Masters_thesis/anharmonic_thermal/" <>
 "thermal_anharmonic/wolfram_output/anharmonic_output_for_l=" <>
 ToString[L] <> "_m=" <> ToString[m] <> "_k=" <> ToString[k] <> ".json",
 <|
   "parameters" -> <|
     "type" -> "Anharmonic",
     "L" -> L,
     "quadrature" -> getQuadrature[m],
     "n" -> Length[BcalLby2minus2],
     "k" -> k
   |>,
   "domains" -> domainedEcalL,
   "M" -> dictM,
   "A" -> dictA,
   "C" -> dictC,
   "B" -> dictB
 |>
]
```

thermal_bootstrap_header_file.py A header file for converting the exported mathematica matrices into CVXPY variables and executing the bootstrap. In detail:

1. `build_matrix()`

   This function is used for translating the matrices we got from mathematica into CVXPY variables Arguments:

   - `coefficients`: A dictionary of the form

     $$\{\texttt{"constant"} : [\Re(A_0), \Im(A_o)],\ \texttt{variable\_name} : [\Re(A_i), \Im(A_i)], \dots\}$$

     where each entry contains real and imaginary parts of a matrix.
   - `variables`: A dictionary mapping variable names to CVXPY variables.

   The real and imaginary parts are combined into complex matrices:

   $$M_{\text{key}} = \Re + \mathtt{j}\Im.$$

   The function returns the affine matrix expression

   $$M = M_{\text{constant}} + \sum_{\text{key}} M_{\text{key}} \cdot x_{\text{key}},$$

   where the sum runs over keys common to both the coefficient dictionary and the variable dictionary. The result is a CVXPY-compatible matrix expression.

2. `run_sdp()`

   This is the main function for the bootstrap. It constructs and solves the SDP using `build_matrix()` and stores results in HDF5 files. It automatically detects between harmonic or anharmonic systems through the mathematica outputted JSON which carry a field that specifies which system the file refers to. The chosen solver has to be hardcoded in.

   Arguments:

   - `temp_range`: Array of temperatures over which the problem is solved.
   - `input_file`: JSON file containing parameters, domains, and matrix coefficient data.
   - `output_folder`: Directory for the generated HDF5 file.
   - `verbose`: If `True`, prints solver information.
   - `maximize`: If `True`, maximizes the objective instead of minimizing. This is used to get upper or lower bounds

- `solver_parameters`: Optional dictionary of solver settings. These depend on which solver is used. For the final plots, this wasn't used.

The function builds the matrices M, A, B, and C using `build_matrix`. These matrices are affine in the optimization variables. The inverse temperature β is used as a CVXPY parameter for faster execution through CVXPY's warm start (for more, refer to CVXPY solver documentation).

The auxiliary matrix variables (the Z and T matrices used in 2.2.11 are also set up inside this function

The constraints imposed finally are:

- Positivity constraint:
$$M^{(L)} \succeq 0.$$
- Initial condition:
$$Z_0 = B.$$
- Block semidefinite constraints for Z:
$$\begin{pmatrix} Z_i & Z_{i+1} \\ Z_{i+1} & A \end{pmatrix} \succeq 0.$$
- Block constraints for abscissa point t_j:
$$\begin{pmatrix} Z_k - A - T_j & -\sqrt{t_j}T_j \\ -\sqrt{t_j}T_j & A - t_j T_j \end{pmatrix} \succeq 0.$$
- Quadrature sum constraint:
$$\sum_j w_j T_j + 2^{-k}\beta C = 0.$$
Where w_j are the quadrature weights as mentioned above

The objective function is automatically determined from the JSON file and already implements the simplifications due to the constraints

- For the Harmonic Oscillator:
$$\text{objective} = P^2.$$
- Otherwise:
$$\text{objective} = \frac{3}{2}P^2.$$

The problem is either minimized or maximized depending on the `maximize` flag.

The function then iterates through the temperature range such that:

- For each temperature T:
  - Update $\beta = 1/T$.
  - Solve the SDP using the hardcoded solver (SDPA-Multiprecision for the anharmonic and CLARABEL for the harmonic).
  - Record:
    * Problem status.
    * Objective value (energy).
    * Variable values.
    * Solver statistics.

The data is then exported into HDF5 files. File attributes include system type, size L, quadrature count m, relaxation order k, and bound type.The datasets include:

- Energy values.
- Solver status.
- Temperatures.
- Variable values.

The naming convention for the outfiles is:

`<system>_L=<L>_m=<m>_k=<k>_<upper/lower>.hdf5`.

The suffix depends on whether the objective was maximized or minimized. The function returns `None`

3. `anharmonic_thermal_energy()` Numerically diagonalizes the Hamiltonian of the quartic oscillator $\hat{H} = \hat{p}^2 + \hat{x}^4$ and returns the thermal expectation value of $\hat{H}$ using a truncated harmonic oscillator basis.

   Arguments:

   - `beta`: The inverse temperature
   - `N`: The size of the truncated basis. Default: 140
   - `omega`: The frequency of the truncated basis. Default: 2.0

thermal_harmonic_bootstrap.py/thermal_anharmonic_bootstrap.py Both this files simply call `run_sdp()` appropriately for the corresponding Mathematica-exported JSON files for upper and lower bounds

plotting_harmonic.py/plotting_anharmonic.py These files simply plot from the HDF5 files gotten through `run_sdp()`. In "plotting_anharmonic.py" also the `anharmonic_thermal_energy()` function is used for the theoretical plot.

References

[1] T. Banks et al. "M theory as a matrix model: A conjecture". In: *Physical Review D* 55.8 (Apr. 1997), pp. 5112–5128. ISSN: 1089-4918. DOI: 10.1103/physrevd.55.5112. URL: http://dx.doi.org/10.1103/PhysRevD.55.5112.

[2] Henry W. Lin. "Bootstrap bounds on D0-brane quantum mechanics". In: *Journal of High Energy Physics* 2023.6 (June 2023). ISSN: 1029-8479. DOI: 10.1007/jhep06(2023)038. URL: http://dx.doi.org/10.1007/JHEP06(2023)038.

[3] Henry W. Lin and Zechuan Zheng. *Bootstrapping Ground State Correlators in Matrix Theory, Part I*. 2025. arXiv: 2410.14647 [hep-th]. URL: https://arxiv.org/abs/2410.14647.

[4] Henry W. Lin. "Bootstrap bounds on D0-brane quantum mechanics". In: *Journal of High Energy Physics* 2023.6 (June 2023). ISSN: 1029-8479. DOI: 10.1007/jhep06(2023)038. URL: http://dx.doi.org/10.1007/JHEP06(2023)038.

[5] Henry W. Lin. *TASI lectures on Matrix Theory from a modern viewpoint*. 2025. arXiv: 2508.20970 [hep-th]. URL: https://arxiv.org/abs/2508.20970.

[6] D. Poland, S. Rychkov, and A. Vichi. "The Conformal Bootstrap: Theory, Numerical Techniques, and Applications". In: *Reviews of Modern Physics* 91 (2019), p. 015002. arXiv: 1805.04405 [hep-th]. URL: https://arxiv.org/abs/1805.04405.

[7] D. Simmons-Duffin. "The Conformal Bootstrap". In: *TASI Lectures* (2017). arXiv: 1602.07982 [hep-th]. URL: https://arxiv.org/abs/1602.07982.

[8] Xizhi Han, Sean A. Hartnoll, and Jorrit Kruthoff. "Bootstrapping Matrix Quantum Mechanics". In: *Physical Review Letters* 125.4 (July 2020). ISSN: 1079-7114. DOI: 10.1103/physrevlett.125.041601. URL: http://dx.doi.org/10.1103/PhysRevLett.125.041601.

[9] Peter D. Anderson and Martin Kruczenski. "Loop equations and bootstrap methods in the lattice". In: *Nuclear Physics B* 921 (Aug. 2017), pp. 702–726. ISSN: 0550-3213. DOI: 10.1016/j.nuclphysb.2017.06.009. URL: http://dx.doi.org/10.1016/j.nuclphysb.2017.06.009.

[10] Minjae Cho et al. *Thermal Bootstrap of Matrix Quantum Mechanics*. 2025. arXiv: 2410.04262 [hep-th]. URL: https://arxiv.org/abs/2410.04262.

[11] Chris Blair. "Matrix Theory: a journal club introduction". In: 2024. URL: https://www.maths.tcd.ie/~cblair/MxIntro.pdf.

[12] Grant Hulsey. "Semidefinite and Bootstrap Methods in One-Dimensional Quantum Systems". ProQuest ID: Hulsey_ucsb_0035D_16608; Merritt ID: ark:/13030/m57f4692. Ph.D. Dissertation. University of California, Santa Barbara, 2024. URL: https://escholarship.org/uc/item/9c17m919.

[13] Huzihiro Araki and Geoffrey L. Sewell. "KMS conditions and local thermodynamical stability of quantum lattice systems". In: *Communications in Mathematical Physics* 52 (1977), pp. 103–109. DOI: 10.1007/BF01625778. URL: https://doi.org/10.1007/BF01625778.

[14] Hamza Fawzi, Omar Fawzi, and Samuel O. Scalet. "Certified algorithms for equilibrium states of local quantum Hamiltonians". In: *Nature Communications* 15.1 (Aug. 2024). ISSN: 2041-1723. DOI: 10.1038/s41467-024-51592-3. URL: http://dx.doi.org/10.1038/s41467-024-51592-3.

[15] G. L. Sewell. "KMS conditions and local thermodynamical stability of quantum lattice systems. II". In: *Communications in Mathematical Physics* 55 (Feb. 1977), pp. 53–61.

[16] Eric W. Weisstein. *Radau Quadrature. From MathWorld–A Wolfram Resource*. https://mathworld.wolfram.com/RadauQuadrature.html. 2025.

[17] Kerry He, James Saunderson, and Hamza Fawzi. *QICS: Quantum Information Conic Solver*. 2024. arXiv: 2410.17803 [math.OC]. URL: https://arxiv.org/abs/2410.17803.

[18] Steven Diamond and Stephen Boyd. "CVXPY: A Python-Embedded Modeling Language for Convex Optimization". In: *Journal of Machine Learning Research* (2016). To appear. URL: https://stanford.edu/~boyd/papers/pdf/cvxpy_paper.pdf.

[19] Paul J. Goulart and Yuwen Chen. *Clarabel: An interior-point solver for conic programs with quadratic objectives*. 2024. arXiv: 2405.12762 [math.OC].

[20] Makoto Yamashita, Katsuki Fujisawa, and Masakazu Kojima. "Implementation and evaluation of SDPA 6.0 (Semidefinite Programming Algorithm 6.0)". In: *Optimization Methods and Software* 18.4 (2003), pp. 491–505. DOI: 10.1080/1055678031000118482. eprint: https://doi.org/10.1080/1055678031000118482. URL: https://doi.org/10.1080/1055678031000118482.

[21] Maho Nakata. "A numerical evaluation of highly accurate multiple-precision arithmetic version of semidefinite programming solver: SDPA-GMP, -QD and -DD." In: *2010 IEEE International Symposium on Computer-Aided Control System Design*. 2010, pp. 29–34. DOI: 10.1109/CACSD.2010.5612693.

[22] Makoto Yamashita et al. "Latest Developments in the SDPA Family for Solving Large-Scale SDPs". In: *Handbook on Semidefinite, Conic and Polynomial Optimization*. Ed. by Miguel F. Anjos and Jean B. Lasserre. Boston, MA: Springer US, 2012, pp. 687–713. ISBN: 978-1-4614-0769-0. DOI: 10.1007/978-1-4614-0769-0_24. URL: https://doi.org/10.1007/978-1-4614-0769-0_24.

[23] Sunyoung Kim et al. "Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion". In: *Mathematical Programming* 129.1 (Sept. 2011), pp. 33–68. ISSN: 1436-4646. DOI: 10.1007/s10107-010-0402-6. URL: https://doi.org/10.1007/s10107-010-0402-6.

[24] Maho Nakata. "A numerical evaluation of highly accurate multiple-precision arithmetic version of semidefinite programming solver: SDPA-GMP, -QD and -DD." In: *2010 IEEE International Symposium on Computer-Aided Control System Design*. 2010, pp. 29–34. DOI: 10.1109/CACSD.2010.5612693.

[25] Adel Bilal. "M(atrix) theory : a pedagogical introduction". In: *Fortschritte der Physik* 47.1–3 (Jan. 1999), pp. 5–28. ISSN: 1521-3978. DOI: 10.1002/(sici)1521-3978(199901)47:1/3<5::aid-prop5>3.0.co;2-b. URL: http://dx.doi.org/10.1002/(SICI)1521-3978(199901)47:1/3%3C5::AID-PROP5%3E3.0.CO;2-B.

[26] Washington Taylor. "M(atrix) theory: matrix quantum mechanics as a fundamental theory". In: *Reviews of Modern Physics* 73.2 (June 2001), pp. 419–461. ISSN: 1539-0756. DOI: 10.1103/revmodphys.73.419. URL: http://dx.doi.org/10.1103/RevModPhys.73.419.

[27] Harold C. Steinacker. *Quantum Geometry, Matrix Theory, and Gravity*. Cambridge University Press, 2024.

[28] Edward Witten. "String theory dynamics in various dimensions". In: *Nuclear Physics B* 443.1–2 (June 1995), pp. 85–126. ISSN: 0550-3213. DOI: 10.1016/0550-3213(95)00158-O. URL: http://dx.doi.org/10.1016/0550-3213(95)00158-O.

[29] Savdeep Sethi and Mark Stern. "D-Brane Bound States Redux". In: *Communications in Mathematical Physics* 194.3 (June 1998), pp. 675–705. ISSN: 1432-0916. DOI: 10.1007/s002200050374. URL: http://dx.doi.org/10.1007/s002200050374.

[30] Massimo Porrati and Alexander Rozenberg. "Bound states at threshold in supersymmetric quantum mechanics". In: *Nuclear Physics B* 515.1–2 (Mar. 1998), pp. 184–202. ISSN: 0550-3213. DOI: 10.1016/s0550-3213(97)00804-3. URL: http://dx.doi.org/10.1016/S0550-3213(97)00804-3.

[31] Edward Witten. "Bound states of strings and p-branes". In: *Nuclear Physics B* 460.2 (Feb. 1996), pp. 335–350. ISSN: 0550-3213. DOI: 10.1016/0550-3213(95)00610-9. URL: http://dx.doi.org/10.1016/0550-3213(95)00610-9.

[32] Lyonell Boulton, María Pilar García del Moral, and Álvaro Restuccia. "Measure of the potential valleys of the supermembrane theory". In: *Physics Letters B* 797 (2019), p. 134873. ISSN: 0370-2693. DOI: https://doi.org/10.1016/j.physletb.2019.134873. URL: https://www.sciencedirect.com/science/article/pii/S0370269319305878.

[33] Robert C Myers. "Dielectric-branes". In: *Journal of High Energy Physics* 1999.12 (Dec. 1999), pp. 022–022. ISSN: 1029-8479. DOI: 10.1088/1126-6708/1999/12/022. URL: http://dx.doi.org/10.1088/1126-6708/1999/12/022.

[34] Joseph Polchinski. "M-Theory and the Light Cone". In: *Progress of Theoretical Physics Supplement* 134 (1999), pp. 158–170. ISSN: 0375-9687. DOI: 10.1143/ptps.134.158. URL: http://dx.doi.org/10.1143/PTPS.134.158.

[35] Henry W. Lin. "Bootstraps to strings: solving random matrix models with positivity". In: *Journal of High Energy Physics* 2020.6 (June 2020). ISSN: 1029-8479. DOI: 10.1007/jhep06(2020)090. URL: http://dx.doi.org/10.1007/JHEP06(2020)090.

[36] Vladimir Kazakov and Zechuan Zheng. "Analytic and numerical bootstrap for one-matrix model and "unsolvable" two-matrix model". In: *Journal of High Energy Physics* 2022.6 (June 2022). ISSN: 1029-8479. DOI: 10.1007/jhep06(2022)030. URL: http://dx.doi.org/10.1007/JHEP06(2022)030.

[37] Masoud Khalkhali et al. *Bootstrapping the critical behavior of multi-matrix models*. 2025. arXiv: 2409.07565 [math-ph]. URL: https://arxiv.org/abs/2409.07565.

[38] A. Zee. *Einstein gravity in a nutshell*. Princeton, NJ: Princeton University Press, 2013. ISBN: 978-0-691-14558-7.

[39] Sean M. Carroll. *Spacetime and Geometry*. Illustrated, reprint, reissue. Cambridge University Press, 2019, p. 516. ISBN: 1108488390.

[40] Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103. URL: https://doi.org/10.7717/peerj-cs.103.