Draft version April 18, 2021
Typeset using LaTeX default style in AASTeX631

Bootstrapping (super) gravitons using machine learning

Mohammad Saad Naeem[1] and Supervisor: Simon Caron-Huot[1]

[1] *Mcgill University*

ABSTRACT

In this paper, we study graviton interactions. Since no numerical calculate schemes have been put forward to calculate functions which describe these interactions, we use general principles of quantum mechanics to come up with constraints which bound this interaction function, a technique known as Bootstrapping. We limit our study to 2-2 super graviton scattering, and use Neural Networks to generate functions which satisfy those constraints, for which no explicit example currently exists. We set up the problem in such a way that starting from random initial conditions, the model is tasked to converge to a function such that the constraints -which were calculated using the output function- converges to 0. We show that Neural Networks can successfully model quantum scattering interactions, producing physically reasonable results. We also check overfitting by using the function to evaluate the constraints in a different space. However, we find that the final output function is still dependent on the initial conditions and converges to different results, indicating that our model has not yet found an optimal solution. Despite this, our scheme still provides a proof of concept of using machine learning for theoretically analyzing quantum graviton scattering.

Keywords: Quantum Gravity— Machine Learning — Bootstrapping

1. INTRODUCTION

Gravitons are the quantum counterparts of Gravitational Waves in low-energy theories of gravity, and understanding their scattering interactions is a key part of modelling a theory of gravity. Currently however, graviton-graviton interactions are not understood. The aim of this project is to study what happens when gravitons interact. Physicists have been working on toy models, such as string theory in 11 dimensions and Anti-de Sitter/Conformal Field Theories to build up mathematical tools to eventually describe gravity in 4 dimensions. Here, we use the method of Bootstrapping, where using physical constraints of fundamental quantum mechanics, we narrow down numerically the allowed function which describes the graviton interaction in 4 dimensions. This allowed function contains an unknown function F, which encodes how two gravitons interact with each other. No explicit example of this function currently exists. In this paper, we will show how a Neural Network can be used to find this interaction function, and model quantum scattering. Using this model, we theoretically bound how much the predictions of a putative theory of quantum gravity could in principle differ from Einstein's gravity, as far as 2 to 2 scattering processes are concerned. Note that we use super gravitons in our model, which unlike real gravitons are scalars and hence simplifies our constraints.

There are other methods of modelling a function under these constraints, such as polynomial approximations, as presented in (GPV21). However, our novel idea is to use Neural Networks. As far as we know, we are the first to use machine learning to study quantum gravity. We were motivated by the fact that machine learning has made important contributions in Physics and Mathematics in recent years, and in particular in high energy physics; see for example references (CFF19), (KMR20), (Rue20). In fact recently, machine learning has started to outperform numerical and analytic methods (AHO20), (PSW19). This paper eschews prior knowledge of general relativity.

2. SCATTERING PROCESSES INVOLVING GRAVITONS

In this section we explore how gravitons scatter, and set up physical constraints that will be used in the Neural Network (NN).

2.1. *Context in Quantum Gravity*

In the 1960's the S-matrix theory was proposed to replace the local quantum field theory as the basic principle of elementary particles, which ultimately led to the development of string theory - see (Bom16) for more details. But despite some interesting results, no nonperturbative S-matrices have been computed (CSZ20) which physically describe particle interactions. This is because all calculation scheme put forward involved unreasonable approximations. Here, using general principles of Quantum Mechanics and symmetries (hence reasonable approximations), we consider the simplest non-trivial S-matrix element which is a 2 to 2 connected scattering amplitude M(s,t). We introduce the mathematical form of M(s,t) later in section 2, but for now, M(s,t) can be visualized using a Feynman diagram:

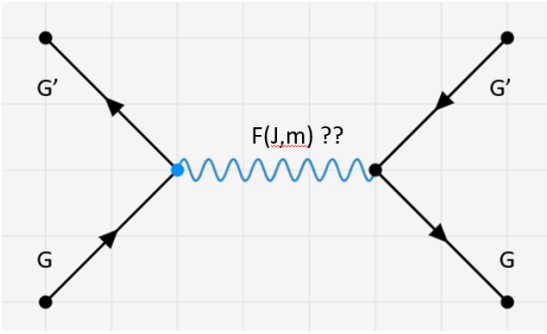


Figure 1: Feyman diagram of 2 - 2 graviton scattering

.

As we can see, two gravitons come from either direction, represented by G, some interaction occurs and they scatter, represented by G'. This scattering interaction is give by the function F(J,m), which is currently unknown, and is what we are trying to find. We can model M(s,t) under the most basic approximations of energy and momentum conservation, causality, analytic continuation from high to low energies, and probability conservation. In the following subsections we come up with constraints mapping this problem from a physical one to that which can be computationally realized.

2.2. *Constraint 1: Crossing Symmetry*

The kinematics of a two body scattering process, is given by Fig 1.1:

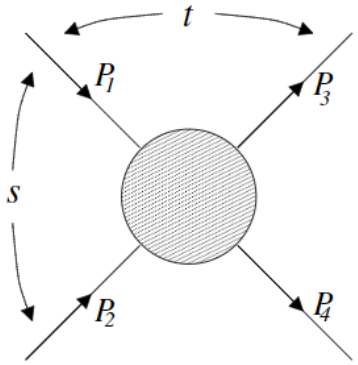


Figure 2: Scattering process of 2-2 graviton showing s and t channels (DDNL04)

.

The amplitude depends on two variables, center of mass energy and scattering angle, encoded in two Mandelstan variables s = $energy^2$ (= $mass^2$ from $E = m$, working in Planck units) and t = $-s\frac{1-cos\theta}{2}$ (momentum transfer). We also let u = -s -t. These were chosen to ensure amplitude symmetry in all permutations of s,t,u.

Crossing refers to the property that particles moving forward in time can be analytically be continued to be thought of as antiparticles moving back in time. This figure not only describes the scattering process $P_1 + P_2 \to P_3 + P_4$ in the s-channel but, by reversing the signs of some of the four-momenta, it can also represent the t-channel process $P_1 + P_{\bar{3}} \to P_{\bar{2}} + P_4$, and the u-channel process $P_1 + P_{\bar{4}} \to P_3 + P_{\bar{3}}$, where the bar denotes the antiparticle. For more details, see (DDNL04). The crossing symmetry for the amplitude M(s,t) is then given by:

$$M(s,t) = M(t,s) \tag{1}$$

Since we cannot evaluate for infinitely many points of s and t, we discretize them for s_i and t_i for i= 1,..N. Our Constraint 1 is consequently:

$$\sum_{(s_i)(t_i)} |M(s,t) - M(t,s)|^2 \tag{2}$$

2.3. *Constraint 2: Matching the high and low energy amplitudes*

The idea behind making sure the scattering amplitude at high energies converges with the amplitude at low energies is so that we have a theory that works on all energy scales, so that causality is met. This is because we know that particles can also be interpreted as waves, and to make sure that causality (interpreted as impossibility of superluminal transfer of information) is met, the front of the interacting wave-packet cannot arrive faster than the speed of light. Hence the energy or the frequency ($E = hf$) of the wave is bounded. We first look at the forms of M(s,t) at high and low energy independently, and then put a constraint to match them so that up till the first order, high energy - low energy = 0. The matching at the second order is covered by Constraint 3.

2.3.1. *High energy*

At high energies, the amplitude is given by:

$$M(s,t) = \sum_J \int_{M^2}^{\infty} ker(s,t,J,m).F_J(m^2)dm^2 \tag{3}$$

Where J is the spin of the particle and runs from J=0, 2, 4,..., ∞, and m is the $energy^2$ in plank units. The function (whose complete form is given in the Appendix) is derived by using analytical arguments of Kramers-Kronig dispersion relation (see (CSZ20) for more details), physically allowed regions of s,t,u of the same mass scatterings and conservation of probability, see (DDNL04) for more details.

Using this equation, we can see how we can compute the amplitude by integrating $F_J(m^2)$ against the kernels of s and t, where $F_J(m^2)$ itself is first computed against the kernel of J and m. As discussed in section 3 in more detail, to computationally do this we need to discretize over all the kernels.

In order for the high energy amplitude to be related to the low energy amplitude, we need to simplify equation [3] to directly relate to the first order low energy expansion given by equation [5]. The trick is to series expand equation [3] at small s (since we want to match it to low energies). After simplifying, we get:

$$Y(t) = \sum_J \int_{M^2}^{\infty} ker(t,J,m).F_J(m^2)dm^2 \tag{4}$$

The full form is given in the appendix. This is actually infinitely many constraints since we must measure the same gravity at momentum transfer t. This means that there are infinitely many integrals over $F_J(m^2)$ which must give the same answer. We foreshadow that to do this we must once again discretize the space of t.

2.3.2. *Low energy*

At low energies, the amplitude M(s, small t) can be series expanded in the following way:

$$M(s,t) = \frac{8\pi G}{stu} + g_0 + g_2(s^2 + t^2 + u^2) \tag{5}$$

The 8π in the first term comes from Einstein's field equation (DDNL04). We want the the high and low energy amplitude to converge, and so we match equation [4] with the first term in equation [5]. After simplifying, our second constraint is then given by:

$$Y(t) = 8\pi G \tag{6}$$

As mentioned before, these are actually infinitely many constraints since we must measure the gravity at infinitely many points of t. Since we cant evaluate at infinite points, we descretize t as t_j for j = 1,...,M so that we now have constraint 2 as:

$$\sum_{(t_j)} |Y(t_j) - 8\pi G|^2 \tag{7}$$

2.4. *Constraint 3: Correction between high and low energies amplitude*

In constraint 2, we wanted the high energy amplitude to converge to the first term of the low energy amplitude expansion, given by equation [5]. In our Constraint 3, we want to model the first order correction, i.e the second term g_0 in equation [5]. In doing this, since we only used the most general principles of quantum mechanics, we can put a bound on how much a theory of quantum gravity could differ from Einsteins general relativity. If someone were to claim to have found a complete theory of gravity, they would have to make sure their result falls within the bounds of this correction term, as far as 2-2 super graviton scattering is concerned. Similar to what we did for Constraint 2, we expand equation (2) at small s, and relate it to the correction term g_0. What we get is:

$$g_0 = \sum_J \int_{M^2}^{\infty} ker(J, z) F_J(m) dm^2 \tag{8}$$

The full form is given in the Appendix. We want to maximise this correction, and so our Constraint 3 is simply -g_0. Having been equipped with these constraints, we are now ready to deploy our Neural Network.

3. USING NEURAL NETWORKS

Neural networks aim to mimic the neurons in the human brain, hence the term neural. We chose Neural Networks because not only are they capable of learning almost any complex functions (Nis21), but they are also easy to extend from interactions of just 2 gravitons to any number of gravitons and their interactions with other particles. As far as we know we are the first to use machine learning (ml) to understand quantum gravity, and hence the main aim of this paper is also to show proof of concept.

To establish notation for future use, let $x^{(i)}$ be the input, and $y^{(i)}$ the output, where i = 1,..., m. Our goal is to learn a hypothesis function h: X $\hookrightarrow$ Y, where $h_\theta(X) = \theta_0 + \theta_1 x_1 + ... + \theta_m x_m$. The θ_i's are the list of parameters to be learned to fit the outputs. We measure the accuracy of the hypothesis by using a loss function. There are different approaches of measuring this accuracy, the most common is simply the mean squared error. In our problem however, we encode our constraints in the loss function. Summing all the constraints, we get our loss function as:

$$J(\theta_j) = \sum_{(s_i)(t_i)} |M(s,t) - M(t,s)|^2 + \alpha \sum_{(t_j)} |Y(t_j) - 8\pi G|^2 - \beta g_0 \tag{9}$$

Where we have introduced parameters α and β to control how much contribution those terms have towards the loss. The goal is simply to minimize the cost function to get the best hypothesis. We have set up the problem in such a way that starting from random initial conditions, the model is tasked to converge to a function such that the constraints -which are calculated using the output function- converges to 0 (apart form the correction term g_0). We accordingly set the values of y^i - the labels of the output - as 0. Since the model is learning the function $F_J(m)$, the x^i are simply pairs (or tuples) of discretizations of the parameters J and m. Given these inputs of x and y, the model can now learn parameters to model the hypothesis function. There are different algorithms to estimate these parameters. One of the most common ones and the one we are using is Gradient Descent. "Conjugate gradient" (JZHG19), "BFGS" and "L-BFGS" (GRB20) are some of the other examples. The way we implement Gradient Descent is by taking the partial

derivative of the cost function w.r.t each θ_j, and simultaneously updating each such that we go down the steepest gradient with a fixed step size α. More formally, we can define the gradient descent algorithm to be the following until convergence:

$$\theta_{j+1} = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_j) x_j^{(i)} \tag{10}$$

The last ingredient we need to build our neural network is an activation function. Because neural networks connect each input to each neuron, suggesting a correlation, it is easy to produce examples where this performs very poorly - namely when an input is not related to a parameter. To tackle this problem, we use activation functions. Given signals, each neuron in the layer computes the activation function in parallel. If an output exceeds a threshold limit, the neuron becomes active, otherwise it remains inactive. There are of course hundreds of such activation functions, we choose ReLU - which acts like a linear function but is actually nonlinear to allow for more complex relationships in data. Please see (Aga19) for more information.

We can now use a neural network to represent a hypothesis function as follows. Our input nodes $x^{(i)}$, known as input layer, goes to other node layers - also known as hidden layers, which outputs the hypothesis function, known as the output layer. Each node in the hidden layer is represented as $a_i^{(j)}$. There is an associated matrix of weights $W^{(j)}$ for each layer which controls the mapping from layer j to j+1. The activation is a function for each neuron that decides the output of the neuron. The following is the NN architecture we used, which we based on (nnp17), a project we though was similar:

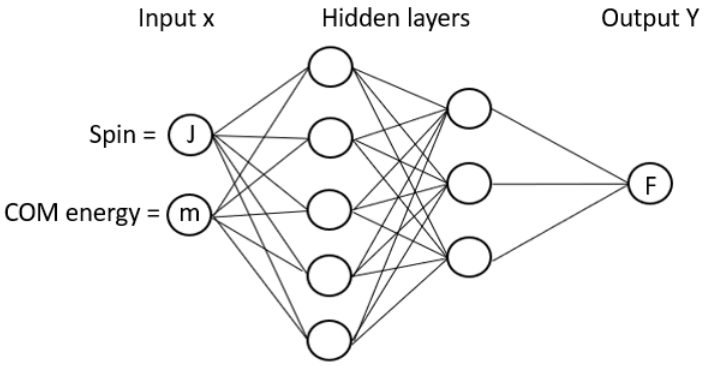


Figure 3: Our NN architecture

The F is computed by using the weights and neurons in the last layer as follows, see (K) for a mathematical introduction to Neural Networks:

$$F(J,m) = a_1^{(4)} = g(W_{10}^{(3)} a_0^{(3)} + W_{11}^{(3)} a_1^{(3)} + W_{12}^{(3)} a_2^{(3)} + W_{13}^{(3)} a_3^{(3)})$$

Where g is the activation function, for which we used ReLU. As we can see, our inputs are tuples of J and m, where m was discretized on 100 points, and J from 0 to 100 with even spins (numbers), giving a total of 4900 input tuples. The model outputs F as a (4900,1) column matrix. The trick here is that this F is being used in kernels of our 3 constraints in the loss function, given by equation [9]. For example, in the crossing symmetry term, to calculate M(s,t), we first discretize s and t as tuples, and then use F in each of the discretizations to complete the calculation. Figure 4 gives a complete picture of the calculation of the the crossing symmetry term:

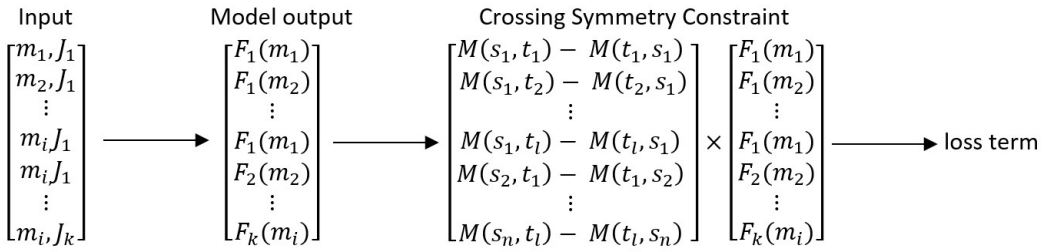


Figure 4: Diagram using the neural network to calculate terms in the loss function constraints.

As we can see, F is being used to evaluate the kernel of M(s,t) - as far as the crossing symmetry constraint is concerned - using discretization over s and t. Note that the crossing symmetry constraint is not a matrix product but a tensor contraction (or a tensor-dot). To make sure we are not over-fitting data over the pre-selected s and t (which were chosen randomly), we need to test convergence of F on a different randomly selected allowed space of s and t. For the rest of constraint terms, F is being multiplied by the kernels of the other constraints in a similar fashion to compute the loss function.

4. RESULTS

With our scheme, we get the following plot of F:

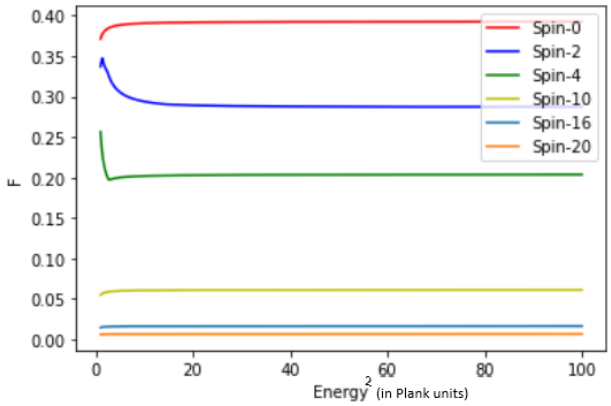


Figure 5: The function $F_J(m)$ trained by the Neural Network, as a function of Energy^2

This is plot of F which encodes the interaction between gravitons, where the y-axis represents the probability of interaction. As we can see the lower spin particles have a higher contribution in this reaction than the higher spin particles. This makes physical sense because most of the particles in the Standard Model have low spins, such as photons of spin 0, and fermions of spin 1/2, and bosons of spin 0 or 1. This begs the question why higher spin particles have a chance of being created at all? This is because the interaction particle could itself be composed of different particles. Another relation of F we see is that with energy. We did not have a prior expectation of what the relation should look like, but it is interesting to note that F is mostly constant with energy. As discussed later in the section, the loss obtained for this model is quite high, and so it casts doubt as to whether the model is depicting the correct relation of F with energy.

As mentioned earlier, we need to make sure our model did not over-fit in training for the space of s and t, and so we make a plot of a completely different set of allowed s and t:

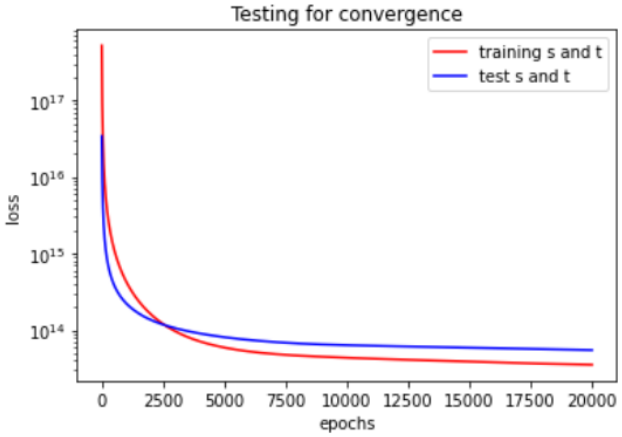


Figure 6: Testing for the convergence of $F_J(m)$ on different s and t

As we can see, the model quickly converges for both the training and test set of s and t's, and we can be confident that the model is not over-fitting.

Finally, we find the value of the correction term g_0 of a theory of quantum gravity on general relativity. To make it dimensionless, we make it $g_0 \times M^6$. To do this, we first evaluate the loss at the end of training and decompose it in to each of the three terms:

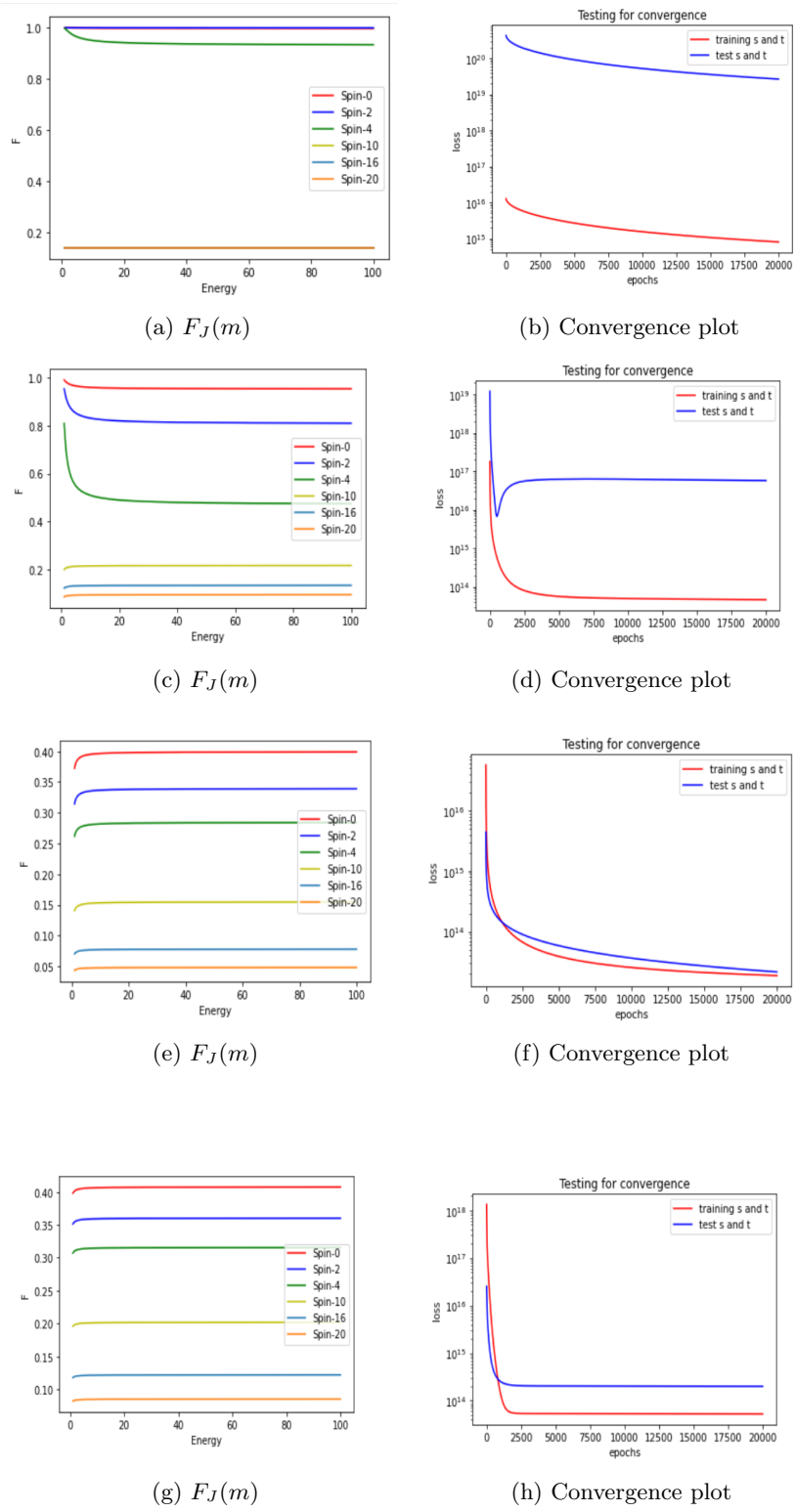
[533867201445.0467, 2946041.406026207, 182.3597036852658]

Figure 7: The loss decomposed in to three constraints: Crossing Symmetry, Convergence of high to low energy, Correction term

We find that the final loss is 5.3×10^{11} (the sum of the three terms), while $g_0 \times M^6$ is 182.36. The loss is quite high, since ideally we would want it to converge it to 0. After decomposing the loss in to its 3 separate constraint terms in Figure 7, we see that the Crossing Symmetry condition was not being learned well at all by the Neural Network, and had the sole contribution to the loss being on the order of 10^{11}. On the other hand, the other two terms were being satisfied fairly well by the function generated by the Neural Network. However, we cannot analytically verify whether or not the loss being high translates into the value of $g_0 \times M^6$ being invalid, although we must approach it with caution.

To try to reduce the loss, we also investigated the effect of other parameters. Referring back to the loss function in equation (9), we introduced two dummy parameters α and β to control the contribution of the three terms to the loss. We found that the loss was lowest for α and $\beta \approx 10^1 - 10^3$, and kept both as 10^3 in all the models. This technique however, had a limited effect on how the model failed to learn a function which obeys crossing symmetry, and hence the order of magnitude of the loss. We also tried using different training algorithms, such as "Adam", "Sgd" and "Adagrad", see (Rud16) for more details. Overall, we found it quite difficult to make the model learn a non-trivial function, since a $F_J(m)$ composed of 0 elements trivially yields the loss as 0.

It was also found that our model is sensitive to the initial conditions of the weights, and converges to different results given a different set of starting points. The following figures shows four examples of the F on different initial conditions, followed by their convergence test on s and t:



(a) $F_J(m)$ (b) Convergence plot

(c) $F_J(m)$ (d) Convergence plot

(e) $F_J(m)$ (f) Convergence plot

(g) $F_J(m)$ (h) Convergence plot

Figure 4.5: Different output of the model each with different initial conditions

It is clear that the plots do not converge even roughly to the same value of the function $F_J(m)$. It is also interesting to note that some do not pass the convergence test, such as in (b) and (d). We subsequently ignore those plots. It was found that on average, the loss was on the order of 10^{11}, and $g_0 \times M^6$ was 556.67. In calculating the average, the

plots where the convergence test failed were ignored.

5. CONCLUSION

Using constraints of general principles of quantum mechanics, we bound the scattering amplitude M(s,t) of 2-2 gravtions, and find $F_J(m)$, the function which encodes the graviton interactions. These constraints included the crossing symmetry, first and second order corrections of matching the high to low energy amplitude. In reference (GPV21), it was demonstrated that the interaction function can be modelled using polynomial approximation. In this paper, we use machine learning. We find that that the model is producing physically reasonable plots of $F_J(m)$, but the final loss is very high $\approx 10^{11}$, due to the model not being able to learn a function that satisfies crossing symmetry. We hence cannot conclude the validity of the shape of the plot of $F_J(m)$.

The dimensionless parameter $g_0 \times M^6$ was found to be at an average of 182.36, which represents the correction to general relativity a theory of quantum gravity can have (by studying second order correction of the convergence of high to low energy amplitude). We also found the model was converging to different solutions, given different initial conditions (given by the initial weights of the model). This could mean either one of two things. Either our problem does not have a unique solution, or our Network is not finding solutions that are optimal. We are confident that our formulation is correct and does have a unique solution, since (GPV21) found a bound to the correction term using polynomial approximation to $F_J(m)$, and hence attribute convergence to different solutions to the latter.

To improve the model in the future, we can use a bigger set of discretization's of the kernels of the constraints, especially that of the crossing symmetry kernel s and t. To narrow down on the optimal solution, we can use more convergence tests, and use the weights of the model that passes the test (ignoring those which do not) as input to another model with different parameters. We can also use different model architectures. Although the model was run on a few different architectures, no direct influence of the architecture on the loss was studied. Finally, we can try different training algorithms for the model. One particularity set of algorithms are Genetic Algorithms (nnp17). With these techniques, we are confident that we can improve our model.

In this paper we have successfully demonstrated how we can use machine learning to describe graviton interactions, and hope that this will provide a completely different tool-set to physicists modelling quantum gravity. We hope that in the future, physicists will use our model to extend our study of 2-2 super gravitons to realistic gravitons, any number of gravitons, and interactions of gravitons with other particles.

APPENDIX

A. FULL FORM OF THE KERNELS

Equation [3]

$$M(s,t) = \frac{2}{\pi}\sum_J \int_{M^2}^{\infty} \frac{(2m^2+t)(m^2+t)\tilde{P}_J(1+\frac{2t}{m^2})}{(m^{d+2})(m^2+s+t)s(s+t)}.F_J(m^2)dm^2 \tag{A1}$$

Where d is the number of dimensions=4, and $\tilde{P}_J$ is related to the Legendre polynomials P_J (for d=4) as:

$$\tilde{P}_J = 16\pi(2J+1)P_J \tag{A2}$$

Equation [4]

$$Y(t) = \sum_J \int_{M^2}^{\infty} -\frac{2t}{\pi}\frac{(2m^2+t)}{m^{d+4}}.\tilde{P}_J(1+2t/m^2).F_J(m^2)dm^2 \tag{A3}$$

Equation [8]

$$g_0 = \frac{4}{\pi}\sum_J \int_{M^2}^{\infty} (\frac{1}{m^2})^{\frac{d}{2}+1}.\tilde{P}_J(1)dm^2 \tag{A4}$$

REFERENCES

[Aga19]Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019.

[AHO20]Anthony Ashmore, Yang-Hui He, and Burt A. Ovrut. Machine learning calabi–yau metrics. *Fortschritte der Physik*, 68(9):2000068, Aug 2020.

[Bom16]Diego Bombardelli. S-matrices and integrability. *Journal of Physics A: Mathematical and Theoretical*, 49(32):323003, Jul 2016.

[CFF19]Iulia M. Comsa, Moritz Firsching, and Thomas Fischbacher. So(8) supergravity and the magic of machine learning. *Journal of High Energy Physics*, 2019(8), Aug 2019.

[CSZ20]Miguel Correia, Amit Sever, and Alexander Zhiboedov. An analytical toolkit for the s-matrix bootstrap, 2020.

[DDNL04]S. Donnachie, Hans Gunter Dosch, O. Nachtmann, and P. Landshoff. *Pomeron physics and QCD*, volume 19. Cambridge University Press, 12 2004.

[GPV21]Andrea Guerrieri, Joao Penedones, and Pedro Vieira. Where is String Theory? 2 2021.

[GRB20]Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-newton methods for training deep neural networks, 2020.

[JZHG19]X. Jin, X. Zhang, K. Huang, and G. Geng. Stochastic conjugate gradient algorithm with variance reduction. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1360–1369, 2019.

[K]Dasaradh S. K. A gentle introduction to math behind neural.

[KMR20]Chethan Krishnan, Vyshnav Mohan, and Soham Ray. Machine learning gauged supergravity. *Fortschritte der Physik*, 68(5):2000027, Mar 2020.

[Nis21]Takato Nishijima. Universal approximation theorem for neural networks, 2021.

[nnp17]Neural networks, 2017.

[PSW19]Maria Laura Piscopo, Michael Spannowsky, and Philip Waite. Solving differential equations with neural networks: Applications to the calculation of cosmological phase transitions. *Physical Review D*, 100(1), Jul 2019.

[Rud16]Sebastian Ruder. An overview of gradient descent optimization algorithms, Jan 2016.

[Rue20]Fabian Ruehle. Data science applications to string theory. *Physics Reports*, 839:1 – 117, 2020. Data science applications to string theory.