Indian Institute of Technology, Bombay

Machine learning as a tool to obtain new black hole solutions
CS 490: RnD Project

Submitted by: Liza Dahiya, 190050063

Contents

Introduction

The aim of the research project is explore and use various neural networks libraries to solve various differential equations in black hole and general relativity to obtain solutions satisfying various constraints.

For this purpose, we employ the `neurodiffeq` library for solving the differential equations. Majorly, we focused on Schwarzchild equations and its variations and then moved on to infinitely degenerate Ricci-flat solutions in $f(R)$ gravity. The results and modeling of the various equations is what we've discussed in this report.

The project was done under the guidance of Prof S. Shankaranarayanan and co-guidance of Prof. Ganesh Ramakrishnan from IIT Bombay.

Schwarzschild Solution

We aim to solve the Einstein's field equations and try to obtain the α and β in the Schwarzschild solutions which are modelled by the equation below:

$$ds^2 = -e^{2\alpha(t,r)}\partial t^2 + e^{2\beta(t,r)}\partial r^2 + r^2\partial\Omega^2 \tag{1}$$

This equation is assumed to be in a general symmetric space-time. For calculating what α and β meant in the equation, the Riemman Tensors were calculated which contract to form Ricci Tensor which was actually used as the equations to be solved.

The Ricci Tensor equations are given below. Here, 0 represents t and 1 represents r respectively.

$$R_{00} = [\partial_0^2\beta + (\partial_0\beta)^2 - \partial_0\alpha\partial_0\beta] + e^{2(\alpha-\beta)}[\partial_1^2\alpha + (\partial_1\alpha)^2 - \partial_1\alpha\partial_1\beta + \frac{2}{r}\partial_1\alpha] \tag{2}$$

$$R_{11} = -[\partial_1^2\alpha + (\partial_1\alpha)^2 - \partial_1\alpha\partial_1\beta - \frac{2}{r}\partial\beta] + e^{2(\beta-\alpha)}[\partial_0^2\beta + (\partial_0\beta)^2 - \partial_0\alpha\partial_0\beta] \tag{3}$$

The other Ricci tensors are modelled using the following set of equations.

$$R_{01} = \frac{2}{r}\partial_0\beta$$

$$R_{22} = e^{-2\beta}[r(\partial_1\beta - \partial_1\alpha) - 1] + 1$$

$$R_{33} = R_{22}\sin^2\theta$$

Equation to be Solved

To solve the Ricci Tensor using `neurodiffeq` package, the assuming is made that the Ricci Tensor are **independent of time (t)**. Hence, in all the equations ∂t is equated 0. This is done to make the equations ordinary differential only dependent on r.

This assumption leads to R_{01} being $= 0$ and R_{00} and R_{11} being modified as follows:

$$R_{00} = e^{2(\alpha-\beta)}[\partial_1^2\alpha + (\partial_1\alpha)^2 - \partial_1\alpha\partial_1\beta + \frac{2}{r}\partial_1\alpha]$$

$$R_{11} = -[\partial_1^2\alpha + (\partial_1\alpha)^2 - \partial_1\alpha\partial_1\beta - \frac{2}{r}\partial_1\beta]$$

Now, we can see that we have 2 equations and correspondingly we can solve 2 equations from any of the 3 independent Ricci Tensor equations given (R_{33} depends on R_{22}).

Hence, we solve R_{00} and R_{11} as a system of equations fed to the neural network.

Constraints and Conditions

The first constraint added is that r is replaced by a *dimensionless quantity* ρ where $\rho = \frac{r}{\delta}$ and $\delta = \frac{2Gm}{c^2}$. This is done to avoid the explosion of the values when fed to the network. R_{00} and R_{11} are modified as follows to obtain an ODE w.r.t δ.

$$R_{00} = \frac{e^{2(\alpha-\beta)}}{\delta^2}[\partial_\rho^2\alpha + (\partial_\rho\alpha)^2 - \partial_\rho\alpha\partial_\rho\beta + \frac{2}{\rho}\partial_\rho\alpha] \tag{4}$$

$$R_{11} = -\frac{1}{\delta^2}[\partial_\rho^2\alpha + (\partial_\rho\alpha)^2 - \partial_\rho\alpha\partial_\rho\beta - \frac{2}{\rho}\partial_\rho\beta] \tag{5}$$

Initial Value for Bounds

The bound added to solve the equation is given by **Minkowski space** shown by:

$$\partial s^2 = -\partial t^2 + \partial r^2 + r^2\partial\Omega^2$$

This tells us that when r tends to infinity (i.e. rho to a certain fixed value), in our case taken to be **5**, the value of α and β tends to 0.

Hyperparamter Tuning

IVP value of r or (ρ)

- The loss value was **very high** (about the order of 10^{200} when the value of infinity (ρ) was set 100. And this loss value decreased as the value of infinity was decreased from 100.

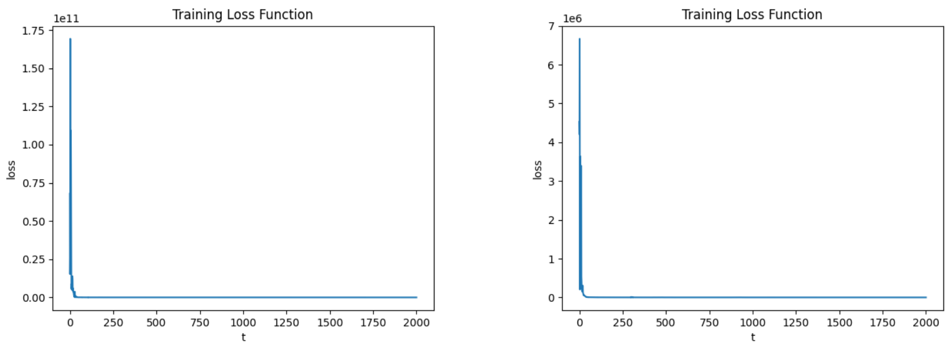

Figure 1: Training Loss when inf = 10 (left) and inf = 8 (right)

- The loss was pretty high even when the infinity was set to be 10 (order of $10^{1}0$). Hence, we decreased the value further to 5.
- Below 5, the loss during training showed abrupt variations for both training and validation losses. Hence, **5** was set as the value for infinity in our equation.

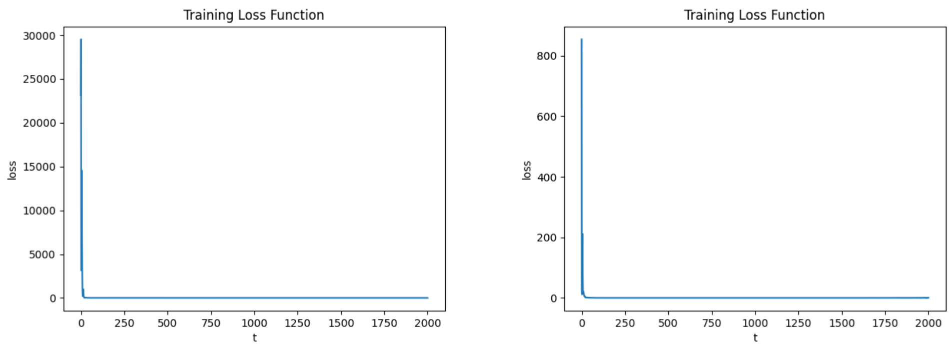

Figure 2: Training Loss when inf = 6 (left) and inf = 5 (right)

# epoches

For the **no. of epochs**, it is good to keep the value between 1500-2000 or even upto 3000. It is observed on increasing the epochs beyond that does not decrease the loss further lower than 10^{-3}. We ran the neural network for 10000 epochs and the results say that the increased epochs only cause a fluctuation of loss but not any further decrease.

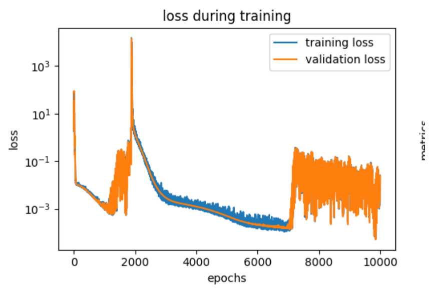

Figure 3: Loss Variation during Training

Results and Observation

The plots for α and β obtained after feeding the network equations R_{00} and R_{11} is shown in the following figure:

These suggests that on infinity both the values reach to 0 which we stated in the IVP itself and when r tends to 0, α tends to $-\infty$ and β tends to ∞.

We also analytically checked for the similarity between α and β and found the following:

$$\boxed{\alpha = -\beta} \tag{6}$$

Generic Notes

- The results are not same **even if** all the initial conditions are set to be same. There is a slight variation (sometime too much) variation when we try to run the equations again.
- Using **Solver1D** instead of **solver_system** removes all the warnings in NeuroDiffEq, however both gives similar solutions on similar conditions.

Measuring Robustness of the Model

We will vary different parameters in the model and rerun the model to obtain the results and check how accurate or deviated they come from the actual answer.

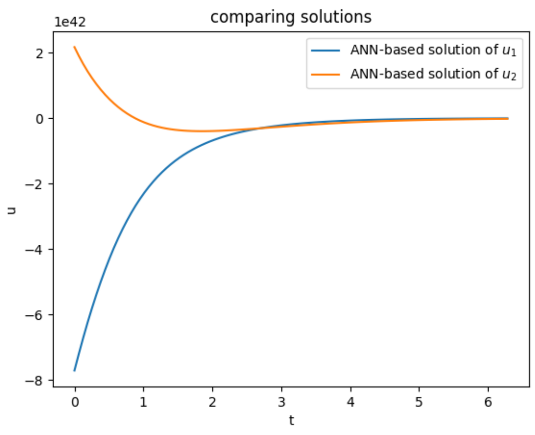

Figure 4: Plot showing variation of α and β

Variation with r

We'll rerun the model for the values of r from 10 to 2, and some larger r values as well.

r = 100

When r = 100, the validation error does not decrease with iterations and remains constantly very high (of order 10^{200}) which indicates the inaccuracy of the results.

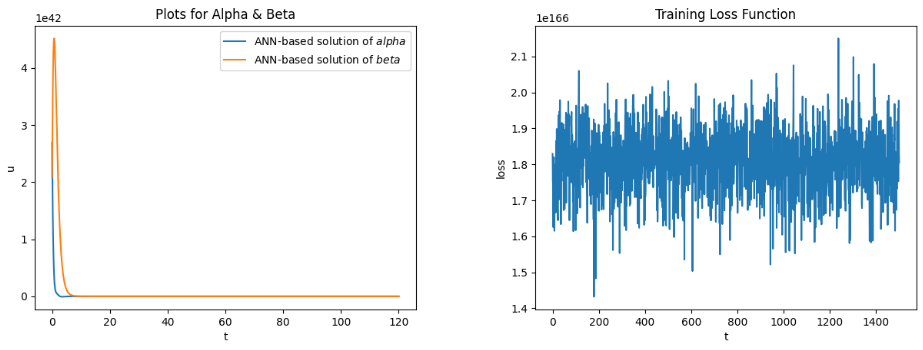

Figure 5: Results when inf = 100 (left) and training loss (right)

r = 70 to r = 10

As we decrease the value of r from 100, we observe that the validation error even though very high (10^{100} to 10^{20}) decreases with iterations.

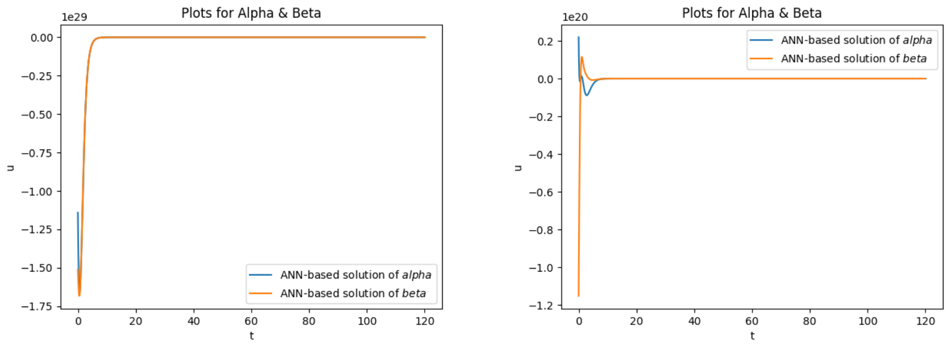

Figure 6: Results when inf = 70 (left) and inf = 50 (right)

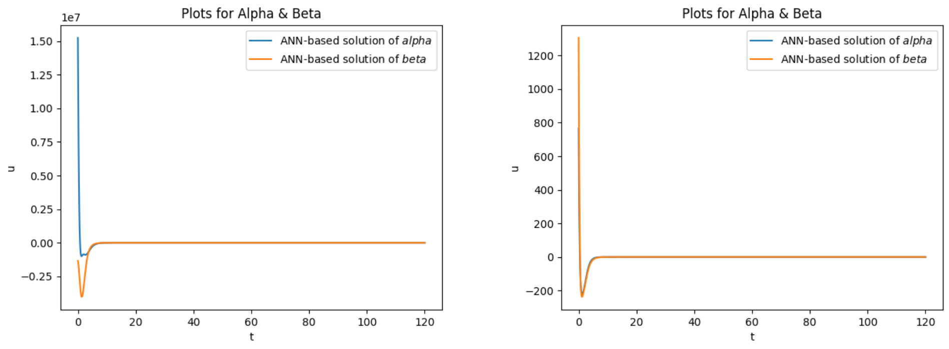

Figure 7: Results when inf = 20 (left) and inf = 10 (right)

r = 8 to r = 7

The error is still high when r = 8 and r = 7 with the order 10^7 (even though it decreases).

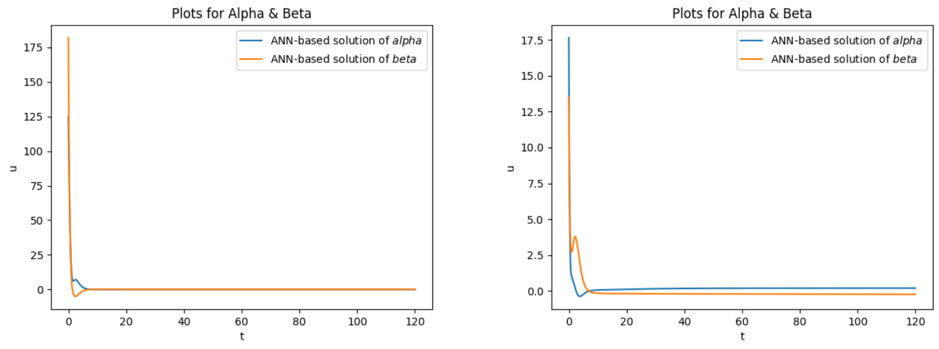

Figure 8: Results when inf = 8 (left) and inf = 7 (right)

r = 6 and r = 5

The error when r is in this range remains low enough to give accurate results by the model. It might start at considerable value but decreases low enough.

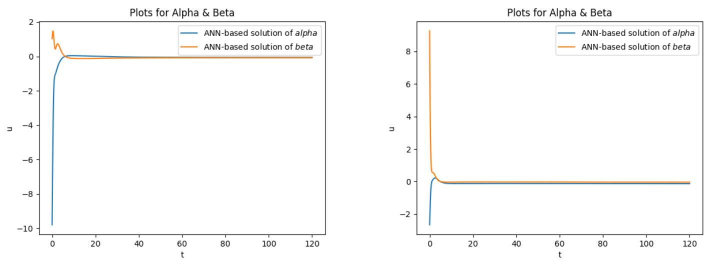

Figure 9: Results when inf = 6 (left) and inf = 5 (right)

r = 4 and r = 2

The validation and training error fluctuates with lower values.

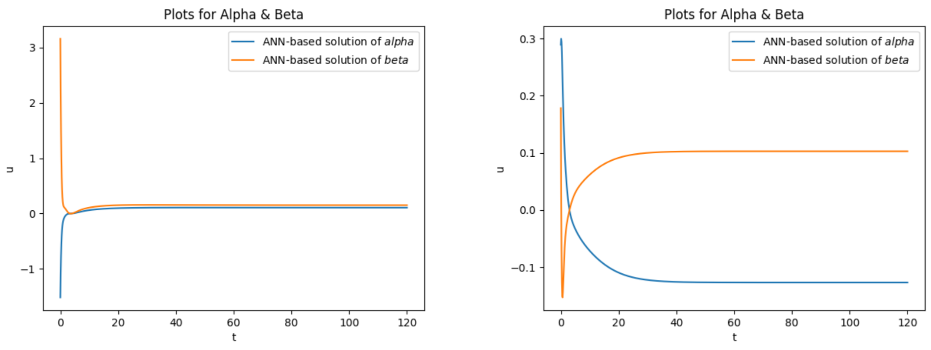

Figure 10: Results when inf = 4 (left) and inf = 2 (right)

Keeping Potential Constant

Instead of taking the potential zero, we'll keep it constant and fixed to some value.

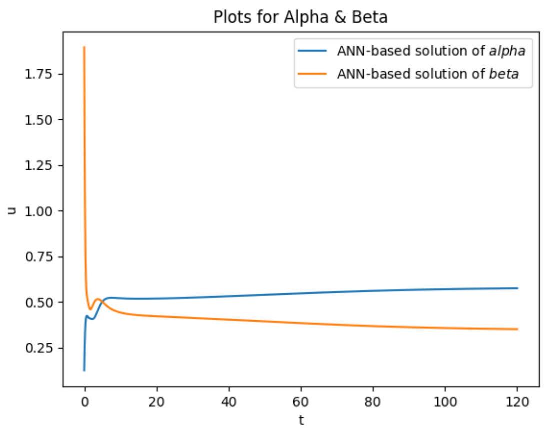

Figure 11: Potential at $r \rightarrow \infty = 0.5$

Solving single variable Ricci Tensor equation

When solving the two variable equation, we obtain the relation between in α and β as given in equation 6, which is as follows:

$$\boxed{\beta = -\alpha}$$

We put this relation in the R_{00} equation to obtain differential in one equation.

$$\begin{aligned} R_{00} &= \frac{e^{2(\alpha-\beta)}}{\delta^2}[\partial_\rho^2\alpha + (\partial_\rho\alpha)^2 - \partial_\rho\alpha\partial_\rho\beta + \frac{2}{\rho}\partial_\rho\alpha] \\ &= \frac{e^{4\alpha}}{\delta^2}[\partial_\rho^2\alpha + 2(\partial_\rho\alpha)^2 + \frac{2}{\rho}\partial_\rho\alpha] \end{aligned}$$

We put this in the `neurodiffeq` solver and obtain the following results.

Constraints and Conditions

The IVP used is when $r \rightarrow \infty(3.0)$ and $\alpha \rightarrow 10e^{-5}$. However, the solution we obtain with these settings were giving different results every time, we were trying to run the model. This was happening because the equation we specified above wasn't complete in constraints. This is a 2nd order ODE. Typically, we need two constraints to uniquely determine the solution. Hence, we added the following constraint as well:

$$\frac{\partial\alpha}{\partial\rho}|_{r\rightarrow\infty} = 0.0$$

This would ensure that the equation gives us a unique solution.

Using Bernaulli Equations

We perform the following substitutions to simplify the original differential equation which we rewrite below:

$$\frac{e^{4\alpha}}{\delta^2}[\partial_\rho^2\alpha + 2(\partial_\rho\alpha)^2 + \frac{2}{\rho}\partial_\rho\alpha] = 0 \tag{7}$$

We assume the following variables and follow these series of substitutions:

$$v = \frac{\partial\alpha}{\partial\rho} \tag{8}$$

$$u = \frac{1}{v} \tag{9}$$

$$\frac{\partial v}{\partial \rho} = \frac{\partial^2 \alpha}{\partial^2 \rho} \tag{10}$$

Substitute (8) and (10) equations in equation (7).

$$\frac{\partial v}{\partial \rho} + 2(v)^2 + \frac{2}{\rho} v = 0 \tag{11}$$

Also, $v = \frac{1}{u}$ and $\frac{\partial v}{\partial \rho} = -\frac{1}{u^2}\frac{\partial u}{\partial \rho}$ which we again substitute in the above equation (11).

$$-\frac{1}{u^2}\frac{\partial u}{\partial \rho} + 2\frac{1}{u^2} + \frac{2}{u\rho} = 0$$
$$\frac{\partial u}{\partial \rho} - 2 - \frac{2u}{\rho} = 0$$

$$\frac{\partial u}{\partial \rho} - \frac{2u}{\rho} = 2 \tag{12}$$

The above equation corresponds to the famous **Bernoulli equation** which has a formulated solution and hence we solve this first degree ODE equation.

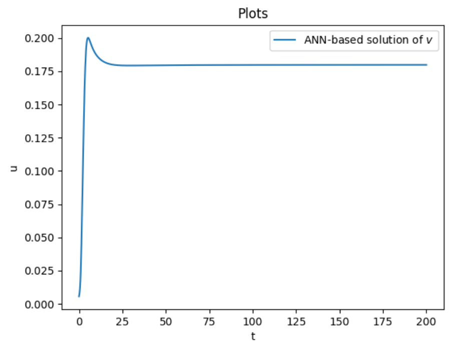

Figure 12: $v = \frac{1}{u}$ where u is solution of the Bernoulli equation

Infinitely degenerate Ricci-flat solutions in $f(R)$ gravity

Aim: Obtain the following solution of $A(r)$ which is obtained when $\delta(r)$.

$$A(r) = 1 + C_2 r^2 - \frac{C_3}{r^2}$$

where $C_2 = \frac{\alpha_0}{12\alpha_1} > 0$ and C_3 is the constant of integration.

Modeling the equation

Non-trivial solutions for the above equation are possible if T_1 or T_2 vanish. So, we'll take $T_2[A(r), \delta(r)]$ and put $\delta(r) = 0$ in this equation.

$$\begin{aligned} T_2[A(r),\delta(r)] = {} & r^2 A(r)\left[(\frac{\Phi(r)}{r})' + \left[(\frac{4+r}{2r}) + \frac{3}{2}[\ln(A(r))]' \right] (\frac{\Phi(r)}{r}) \right] \\ & - \frac{r^2 A(r)}{2}\left([\ln A(r)]'^2 + (\frac{4+r}{r})[\ln(A(r))]' - \frac{4}{r}\left[\frac{1}{r}+2\right]\right) - 2(1+\frac{\alpha_0 r^2}{2\alpha_1}) \end{aligned}$$

We have the following relation:

$$\begin{aligned} \Phi(r) &= r(\delta'(r) + [\ln A(r)]') \quad \text{as} \quad \delta(r) = 0 = \delta'(r) \\ &= r[\ln A(r)]' \end{aligned}$$

Let us now perform the following substitution:

$$\begin{aligned} B(r) &= \ln(A(r)) \\ B'(r) &= [\ln(A(r))]' \end{aligned}$$

Using these, we obtain new version of T_2.

$$\begin{aligned} T_2[e^{B(r)}, 0] = {} & r^2 e^{B(r)}\left[B''(r) + \left[(\frac{4+r}{2r}) + \frac{3}{2}B'(r)\right] B'(r)\right] \\ & - \frac{r^2 e^{B(r)}}{2}\left([B'(r)]^2 + (\frac{4+r}{r})B'(r) - \frac{4}{r}\left[\frac{1}{r}+2\right]\right) - 2(1+\frac{\alpha_0 r^2}{2\alpha_1}) \end{aligned}$$

To form the differential equation to be solved we, put $T_2 = 0$ in the above equation:

$$\begin{aligned} 0 = {} & r^2 e^{B(r)}\left[B''(r) + (\frac{4+r}{2r})B'(r) + \frac{3}{2}(B'(r))^2 \right] \\ & - \frac{r^2 e^{B(r)}}{2}\left([B'(r)]^2 + (\frac{4+r}{r})B'(r) - \frac{4}{r}\left[\frac{1}{r}+2\right]\right) - 2(1+\frac{\alpha_0 r^2}{2\alpha_1}) \end{aligned}$$

Simplifying the equation further:

$$r^2 e^{B(r)}\left[B''(r) + (B'(r))^2 + \frac{2}{r}\left[\frac{1}{r}+2\right]\right] - 2(1+\frac{\alpha_0 r^2}{2\alpha_1}) = 0 \tag{13}$$

Here, α_0 is dimensionless constant while α_1 is constant having dimension of $[L]^2$.

Equation to be solved by `neurodiffeq`

Rewriting above equation:

$$\left[B''(r) + (B'(r))^2 + \frac{2}{r}\left(\frac{1}{r} + 2\right)\right] - 2e^{-B(r)}(\frac{1}{r^2} + \frac{\alpha_0}{2\alpha_1}) = 0 \tag{14}$$

The equations here are dimensionless in r and for simplicity we take both $\alpha_0 = 1$ and $\alpha_1 = 1$. And hence, the above equation turns out as follows. We'll code up the above equation now in `neurodiffeq`.

$$\left[B''(r) + (B'(r))^2 + \frac{2}{r}\left(\frac{1}{r} + 2\right)\right] - 2e^{-B(r)}(\frac{1}{r^2} + 0.5) = 0 \tag{15}$$

IVP for the equation would be if $r \to 10$, then $A(r) \to 100$ and $A'(r) \to 10$. And from this, we 'll calculate $B(r)$ and $B'(r)$.

Observations

Currently, training using these constraints is giving very high error and hence we need to understand what is making up the model blow up so much, The equation has an exponential term which might be the reasoning we got `nan` values for all possible values of r.

Work Action Plan

The research project still needs a completion and these are some of the tasks currently in that direction which we need to do.

- Obtain solutions for Ricci Tensors by hyperparameter tuning
- Understand cases in which the solutions blow up (ex: using log and exp).
- Making results more robust than they're currently
- Address the equations that gave different results on reruning the model everytime.