4-17-2019

Porting Symbolic Libraries from Maple to Python

James Lewis
Utah State University

Porting Symbolic Libraries from Maple to Python

James Lewis

PHYS 4900

Abstract

Utah State University has a database of tables of the known solutions of the Einstein Field Equation. This database is primarily recorded in the symbolic computing program Maple. This database needs to be translated into Python code so the information can be more accessible to the general public. This translation was successful for one table, which means that it is possible to translate the entire database.

Experiment Date: January 7-April 8, 2019
Submission Date: April 17, 2019

Introduction

This research is part of an ongoing project that electronically stores exact solutions of the Einstein Field Equations in a database, or a library of solutions. The database uses the symbolic computational language Maple to verify and store the solutions. The library can be used to search for solutions with desired properties and check conjectures about Einstein's Field Equations. It also serves to verify and correct the existing literature.

Theory

The current work on this library involves expanding its scope and investigating the possibilities for porting the library from Maple into other computational formats. I will describe the results of the activities designed to reliably port the library of solutions into a Python format that can be utilized by software tools within Python such as Sympy and SageManifolds.

Equipment

- Laptop
- Anaconda Navigator with Spyder 3.2

Procedure

The beginning of the project consisted of finding a structure similar to a Maple table in Python. When researching, one of the first things that was tested was a table from the library Astropy. While the Astropy table was able to contain all the data from the Maple table, the Astropy table would not display all the information when called upon. While performing further research and receiving aid from the Computer Science department, I was able to find a structure within Python that was similar to a Maple table. Python can construct a dictionary that is able to hold information similar to a table in Maple.

After discovering that a Python dictionary was able to hold the same information as a Maple table, I then took a specific table (DGTable[[12, 6, 1]]) from the database and rewrote the data in Python code using the Sympy library. Sympy was able to perform symbolic calculations within Python, similar to how Maple performs symbolic calculations. The syntax for Sympy is different from the syntax of Maple. Every line of code had to be rewritten so that the syntax was correct in the Python dictionary. After this dictionary was written, the next step was writing a program that would read the tables in a Maple text file and write a Python dictionary.

A program was written that would read the same Maple table from the database. The program then read the text file containing the Maple table data, and wrote a new text file containing the same data. The generated text file formed a Python dictionary. The code produced read every line in the Maple text file and rewrote the line with the appropriate syntax. After the program ran, the syntax in the Python dictionary was the same as the one written earlier. I let the program read DGTable[[12, 6, 1]] and write a Python dictionary. When the program was finished, the new text file was the same Python dictionary I constructed earlier.

Results

It is possible to rewrite a Maple table into a Python dictionary that contains the same information. A program can be written that can read a Maple text file that contains a table with the information of a known solution to the Einstein Field Equation and write a Python dictionary with the same information. The dictionary is only able to hold information of the Einstein Field Equation. However, it is not able to calculate the Einstein Equation or the Energy Momentum Tensor. This is because the Sympy library does not contain the math to perform differential geometry or metric algebra at this time. Programmers are working on the library so these mathematical computations can be possible in the future. Also the written program can only work for DGTable[[12, 6, 1]]. The code can be optimized so that it can read multiple tables and create individual dictionaries that contain the right information.

Summary

Porting a Maple table into Python is possible with the built in dictionary function. A Python dictionary is able to hold the same information as a Maple table using the Sympy library. A program can be written that is able to read a text file containing the Maple table and write a Python dictionary with the same information and correct syntax. While the program is only able to read and write a specific table, the program can be optimized so it can read any table and create a Python dictionary. The math performed in Maple is not yet possible in the Sympy library. However, in the future, the library may be optimized so that the math contained in Maple can be calculated in Sympy.

If these optimizations are performed on the code, the information of the known solutions of the Einstein Field Equations could enter into the hands of the general public. This information would then be used to help find even more solutions for the Einstein Field Equation, which would further our knowledge.

Acknowledgments

James Lewis wishes to thank Andrew Brim and graduate students of Utah State University's Computer Science department for technical assistance.

Supplementary Materials

Table 1: Supplementary Materials in PHYS_4900.zip

Filename	Content
4900testfile.txt	Input text file that contains DGTable[[12, 6, 1]] in Maple syntax.
MapleToPython.py	Python program that reads 4900testfile.txt and outputs test.txt
Sympy_startup.py	Python dictionary that contains data for DGTable[[12, 6, 1]]
test.txt	Output text file that contains DGTable[[12, 6, 1]] in Python syntax

Appendix A

Below is the code to from MapleToPython.py

```
#Packages that will be used
from sympy import *

#Dragging file so it can open
userinput=input('Drag file here: ')

userinput=userinput.replace("´",""")

newfile=""

file = open('test.txt', 'w')

#Writing the first few lines so the symbols can work.
file.write("from sympy import *\ n")
file.write("u, v, y, z=symbols('u v y z')\ n")
file.write("_Lambda=symbols('_Lambda')\ n")
file.write("_kappa0=symbols('_kappa0')\ n")

#Replacing certain text in table to make it compatible in Python
with open(userinput,'r') as f:
 for line in f:
 if line[0]=="#":
```

```
 file.write(line)
 if not ’DGTable’ in line or ’Check’ in line:
 continue
 elif ’table()’ in line:
 line = line.replace(”DGTable[[12, 6, 1]]”, ”t”)
 line = line.replace(”:”, ””)
 line = line.replace(’table()’, ’’)
 else:
 line = line.replace(’table()’, ’’)
 line = line.replace(”DGTable[[12, 6, 1]]”, ”t”)
 line = line.replace(”:= ”, ”= [”)
 line = line.replace(’\ n’, ’]\ n’)
 line = line.replace(’abs’,’Abs’)
 line = line.replace(’ˆ’, ’**’)
 line = line.replace(’:’, ”)
 line = line.replace(’;’, ”)
 line = line.replace(’u=0’,’Eq(u,0)’)#This line is specific to DGTable[[12, 6, 1]]
 line = line.replace(’v=0’, ’Eq(v,0)’)#This line is specific to DGTable[[12, 6, 1]]
 line = line.replace(’y=1’,’Eq(y,0)’)#This line is specific to DGTable[[12, 6, 1]]
 line = line.replace(’z=0’, ’Eq(y,0)’)#This line is specific to DGTable[[12, 6, 1]]
file.write(line)

print(’wrote file’)

#Run this file one more time in order for it to work.
```

Appendix B

Below is the code for Sympy_startup.py

```
from sympy import *

u, v, y, z=symbols(’u v y z’)
_Lambda=symbols(’_Lambda’)
_kappa0=symbols(’_kappa0’)

###########################################################
# Chapter 12.2
###########################################################
# Metric 12.6
# Feb 20, 2009
###########################################################
t={}
```

```
t[”Reference”]=[”Stephani”]
t[”PrimaryDescription”]=[”PureRadiation”]
t[”SecondayDescription”]=[['Homogenous']]
t[”Authors”]=[[”Defrise (1969)”]]
t[”Comments”]=[[”_Lambda < 0 required for a pure radiation solution”]]
t[”Coordinates”]=[[u, v, y, z]]
t[”Domains”]=[[[y>0]]]
t[”BasePoints”]=[[[Eq(u,0), Eq(v,0), Eq(y,1), Eq(z,0)]]]
t[”Parameters”]=[[[_Lambda, _kappa0]]]
t[”SideConditionsAssuming”]=[[[_Lambda<0, 0<_kappa0]]]
t[”SideConditionsSimplify”]=[[[]]]
t[”CoefficientInfo”]=[[[]]]
t[”Metric”]=[[[0,-(3/2)/(y**(2)*Abs(_Lambda)), 0, 0],[-(3/2)/(y**(2)*Abs(_Lambda)), 3*_Lambda/(y**(4)*Al
0, 0], [0, 0, 3/(y**(2)*Abs(_Lambda)), 0], [0, 0, 0, 3/(y**(2)*Abs(_Lambda))]]]
t[”CosmologicalConstant”]=[_Lambda]
t[”NewtonConstant”]=[_kappa0]
t[”EnergyMomentumTensor”]=[[[(20/9)*_Lambda**(2)/_kappa0, 0, 0, 0], [0, 0, 0, 0], [0, 0,
0, 0], [0, 0, 0, 0]]]
t[”OrthonormalTetrad”]=[[[(1/2)*2**(1/2)-(1/3)*2**(1/2)*_Lambda-(1/3)*2**(1/2)*y**(2)*_Lambda,
-(1/3)*2**(1/2)*y**(2)*_Lambda, 0, -(1/3)*2**(1/2)*y**(2)*_Lambda], [(2/3)*3**(1/2)*y*(Abs(_Lambda))**
0, 0, (1/3)*3**(1/2)*y*(Abs(_Lambda))**(1/2)], [0, 0, (1/3)*3**(1/2)*y*(Abs(_Lambda))**(1/2),
0], [(1/3)*2**(1/2)*_Lambda+(1/3)*2**(1/2)*y**(2)*_Lambda+(1/2)*2**(1/2), (1/3)*2**(1/2)*y**(2)*_Lam
0, (1/3)*2**(1/2)*y**(2)*_Lambda]]]
t[”NullTetrad”]=[[[1, 0, 0, 0], [-(2/3)*_Lambda-(2/3)*y**(2)*_Lambda, -(2/3)*y**(2)*_Lambda,
0, -(2/3)*y**(2)*_Lambda], [(1/3)*2**(1/2)*3**(1/2)*y*(Abs(_Lambda))**(1/2), 0, (1/6)*I*2**(1/2)*3**(1/2
(1/6)*2**(1/2)*3**(1/2)*y*(Abs(_Lambda))**(1/2)], [(1/3)*2**(1/2)*3**(1/2)*y*(Abs(_Lambda))**(1/2),
0, -(1/6)*I*2**(1/2)*y*(Abs(_Lambda))**(1/2), (1/6)*2**(1/2)*3**(1/2)*y*(Abs(_Lambda))**(1/2)]]]
t[”PetrovType”]=[”N”]
t[”PlebanskiPetrovType”]=[”O”]
t[”SegreType”]=[”[(2,11)]”]
t[”IsometryDimension”]=[6]
t[”KillingVectors”]=[[[0, 2*v, y, z], [0, 0, 0, 1], [y**(2)+z**(2), v**(2), y*v, z*v], [2*z, 0, 0,
v], [1, 0, 0, 0], [0, 1, 0, 0]]]
t[”OrbitDimension”]=[4]
t[”OrbitType”]=[”PseudoRiemannian”]
t[”IsotropyType”]=[”F10”]

#t.keys() to print off the table library.
```