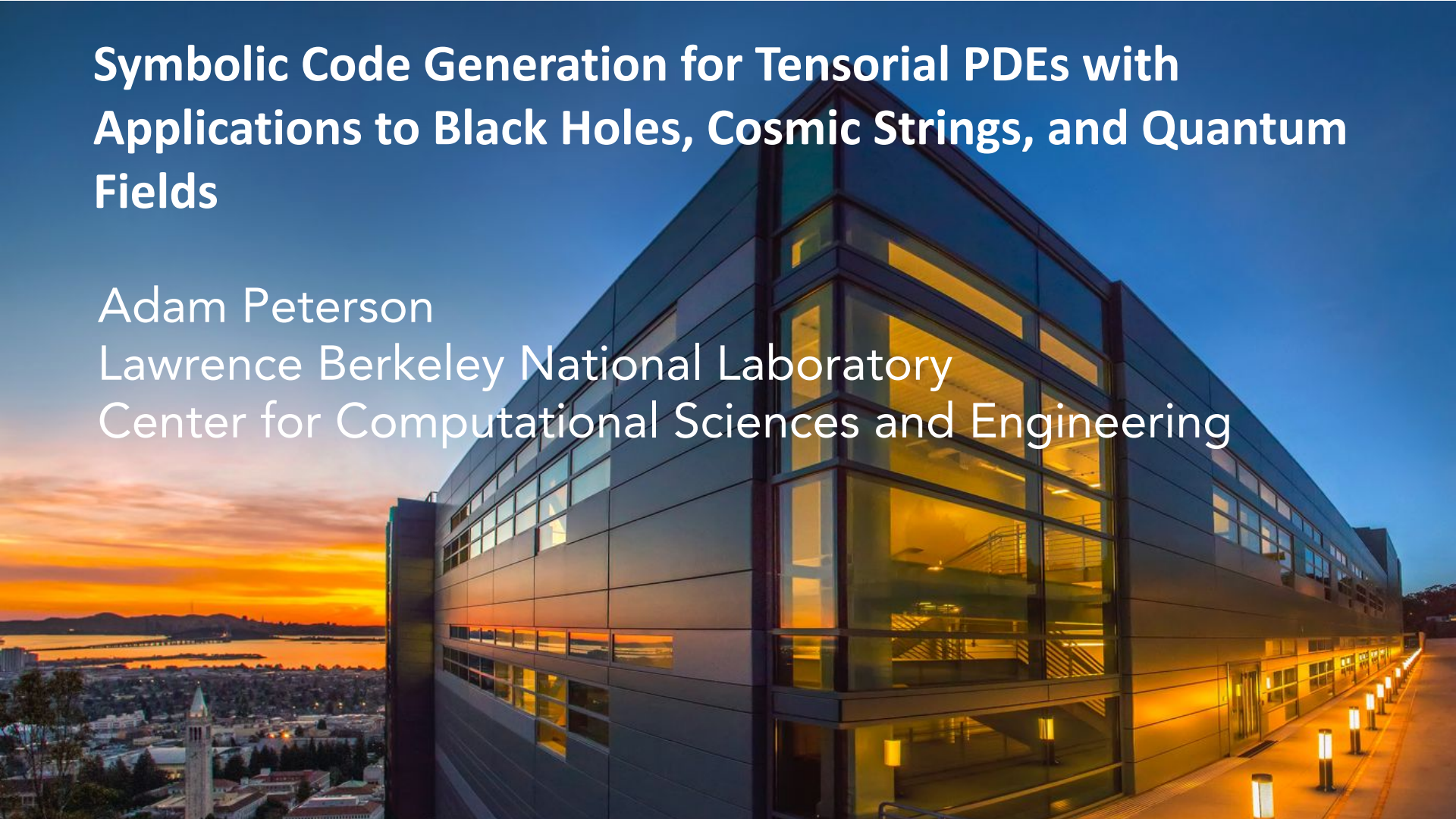
Symbolic Code Generation for Tensorial PDEs with Applications to Black Holes, Cosmic Strings, and Quantum Fields

Adam Peterson
Lawrence Berkeley National Laboratory
Center for Computational Sciences and Engineering



Motivation

High performance computing generates new avenues of research for theoretical physics

For highly non-linear coupled systems we must bridge the gap between symbolic formulations and equivalent computer compilable expressions

Examples of systems in GR and QFT come to mind

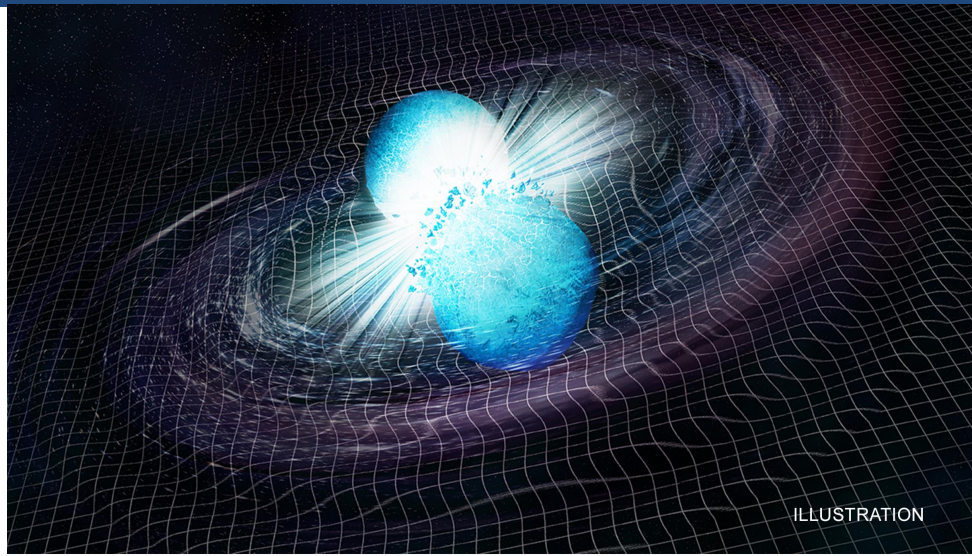


Illustration: CXC/M. Weiss

Code generation is the key!

Let us look at computational approaches to spacetime solvers

- Einstein's equation in symbolic form: $G^{\alpha\beta} = 8\pi T^{\alpha\beta}$
- All physics is contained in a very compact symbolic form.
- Highlights the overall *conceptual* simplicity of gravitation as spacetime curvature due to matter and energy distribution.

Building a computational tool ….

To solve the Einstein equations using explicit time integration, we want:

1. (Massively) parallelizable data structures and iterators for mesh data

2. Adaptive mesh refinement (AMR) with subcycling in time; to keep the BC's far away, and improve resolution in high curvature regions

3. Higher-order time integrator (4th order is usually necessary)

4. Compilable-expression of equations of motion, using higher order spatial discretizations, to convert PDEs to ODEs (Method of Lines)

Motivation (Part 2)

- Parallelizable data structures, AMR/subcycling, and higher-order time integrator all provided in frameworks like AMReX and Flash-X with immediate support for large scale problems and CPU/GPU accessibility.

- We just need to write the equations of motion in compilable expressions without making typos or undetectable errors in indexing or finite differencing.

Typical Einstein Equation of 2D holographic superconductor

$$
\begin{aligned}
G_{22} = {} & (-4\,x\,y^2\,(-1+y^3)\,yh^4\,(1-y\,\alpha+y^2\,\alpha)^2\,Q1[x,y]^2\,Q2[x,y]^2\,Q4[x,y]^2\,Q5[x,y]^2 \\
& ((Q2[x,y]\,(1-L^2\,x^{2\,n1}\,y^2\,\gamma\,Q6[x,y]^2)+(-1+y^3)\,(1+(-1+x)^2\,x^2\,y^4\,yh^2\,Q3[x,y]^2\,Q4[x,y]\,(-1+L^2\,x^{2\,n1}\,y^2\,\gamma\,Q6[x,y]^2)))\,Q8[x,y]^2+ \\
& L^2\,y^2\,\beta\,\gamma\,(-Q2[x,y]+(-1+x)^2\,x^2\,y^4\,(-1+y^3)\,yh^2\,Q3[x,y]^2\,Q4[x,y])\,Q8[x,y]^4+2\,y\,(-1+y^3)\,Q8[x,y]\,Q8^{(0,1)}[x,y]+y^2\,(-1+y^3)\,Q8^{(0,1)}[x,y]^2)+ \\
& x\,y\,(-1+y^3)^2\,yh^4\,(1-y\,\alpha+y^2\,\alpha)\,Q1[x,y]\,Q5[x,y] \\
& ((-1+x)^2\,x^2\,y^4\,(2+y^3)\,yh^2\,Q2[x,y]^2\,Q3[x,y]\,Q4[x,y]^2\,Q5[x,y]\,(2\,Q4[x,y]\,Q3^{(0,1)}[x,y]+Q3[x,y]\,Q4^{(0,1)}[x,y])+ \\
& Q1[x,y]\,(-4\,Q2[x,y]^3\,Q5[x,y]\,Q4^{(0,1)}[x,y]-4\,(-1+x)^2\,x^2\,y^5\,(-1+y^3)\,yh^2\,(1-y\,\alpha+y^2\,\alpha)\,Q3[x,y]\,Q4[x,y]^3\,Q5[x,y]\,Q2^{(0,1)}[x,y] \\
& (Q3^{(0,1)}[x,y]+(-1+x)^3\,x\,y^2\,Q3[x,y]\,Q3^{(1,0)}[x,y])- \\
& Q2[x,y]^2\,((1-y\,\alpha+y^2\,\alpha)\,Q5[x,y]\,(y\,Q4^{(0,1)}[x,y]^2+4\,(-1+x)^3\,x\,y^2\,Q4[x,y]^2\,Q3^{(1,0)}[x,y]\,(-2+(-1+x)^3\,x\,y^3\,Q3^{(1,0)}[x,y])+ \\
& 4\,Q4[x,y]\,Q4^{(0,1)}[x,y]\,(-1+(-1+x)^3\,x\,y^3\,Q3^{(1,0)}[x,y]))+2\,(-1+x)^2\,x\,y^4\,Q3[x,y]^2\,Q4[x,y] \\
& (Q4[x,y]\,(x\,(-1+y^3)\,yh^2\,Q4^{(0,1)}[x,y]-(-1+x)^3\,y\,(1-y\,\alpha+y^2\,\alpha)\,Q4^{(1,0)}[x,y])+ \\
& Q5[x,y]\,(-x\,(-1+y^3)\,yh^2\,Q4^{(0,1)}[x,y]+2\,(-1+x)^3\,(-2+5\,x)\,y\,(1-y\,\alpha+y^2\,\alpha)\,Q4^{(1,0)}[x,y]))+ \\
& 4\,(-1+x)^2\,y^2\,Q3[x,y]\,(x\,y^2\,Q4[x,y]^3\,(x\,(-1+y^3)\,yh^2\,Q3^{(0,1)}[x,y]-(-1+x)^3\,y\,(1-y\,\alpha+y^2\,\alpha)\,Q3^{(1,0)}[x,y])+ \\
& x\,y^2\,Q4[x,y]^2\,Q5[x,y]\,(3\,x\,(-1+y^3)\,yh^2\,Q3^{(0,1)}[x,y]+2\,(-1+x)^3\,(-2+5\,x)\,y\,(1-y\,\alpha+y^2\,\alpha)\,Q3^{(1,0)}[x,y])+ \\
& (-1+x)\,x\,y\,(1-y\,\alpha+y^2\,\alpha)\,Q5[x,y]\,Q4^{(0,1)}[x,y]\,Q4^{(1,0)}[x,y]+(1-y\,\alpha+y^2\,\alpha)\,Q4[x,y]\,Q5[x,y] \\
& ((-1+2\,x)\,y\,Q4^{(0,1)}[x,y]+2\,(-1+x)\,x\,(-1+(-1+x)^3\,x\,y^3\,Q3^{(1,0)}[x,y])\,Q4^{(1,0)}[x,y]))- \\
& (-1+x)^2\,x\,y^3\,Q2[x,y]\,Q3[x,y]\,Q4[x,y]\,Q5[x,y] \\
& (2\,(1-y\,\alpha+y^2\,\alpha)\,(Q4^{(0,1)}[x,y]+(-1+x)^3\,x\,y^2\,Q3[x,y]\,Q4^{(1,0)}[x,y])\,(x\,y^2\,(-1+y^3)\,yh^2\,Q3[x,y]\,Q4^{(0,1)}[x,y]-2\,(-1+x)\,Q2^{(1,0)}[x,y]+ \\
& (-1+x)^3\,x^2\,y^4\,(-1+y^3)\,yh^2\,Q3[x,y]^2\,Q4^{(1,0)}[x,y])+ \\
& 2\,x\,y\,yh^2\,Q4[x,y]^2\,((10-19\,y^3-9\,y\,\alpha+8\,y^2\,\alpha+18\,y^4\,\alpha-17\,y^5\,\alpha-4\,(-1+x)^2\,(-1+2\,x)\,y^3\,(-1+y^3)\,(1-y\,\alpha+y^2\,\alpha)\,Q3[x,y]-6\,(-1+x)^2\,x^2\,y^4\,(-1+y^3)^2\,yh^2\,Q3[x,y]^2) \\
& Q3^{(0,1)}[x,y]+2\,y\,(1-y\,\alpha+y^2\,\alpha)\,(-(-1+y^3)\,Q3^{(0,2)}[x,y]+(-1+x)^3\,x\,y\,Q3[x,y]\,((8-11\,y^3-2\,(-1+x)^2\,x\,y^3\,(-1+y^3)\,Q3[x,y])\,Q3^{(1,0)}[x,y]+ \\
& (-1+x)^3\,x\,y^3\,(-1+y^3)\,Q3^{(1,0)}[x,y]^2-2\,y\,(-1+y^3)\,Q3^{(1,1)}[x,y])))- \\
& x\,y\,Q4[x,y]\,(yh^2\,Q3[x,y]\,((-2+5\,y^3+y\,\alpha-4\,y^4\,\alpha+3\,y^5\,\alpha)\,Q4^{(0,1)}[x,y]+2\,y\,(-1+y^3)\,(1-y\,\alpha+y^2\,\alpha)\,Q4^{(0,2)}[x,y])+ \\
& 8\,(-1+x)^4\,y\,(1-y\,\alpha+y^2\,\alpha)\,Q2^{(1,0)}[x,y]\,Q3^{(1,0)}[x,y]+2\,(-1+x)^2\,x^2\,y^4\,(-1+y^3)\,yh^2\,Q3[x,y]^3 \\
& ((-1+y^3)\,yh^2\,Q4^{(0,1)}[x,y]+2\,(-1+x)^3\,y\,(1-y\,\alpha+y^2\,\alpha)\,Q4^{(1,0)}[x,y])+4\,(-1+x)^3\,x\,y^2\,(-1+y^3)\,yh^2\,(1-y\,\alpha+y^2\,\alpha)\,Q3[x,y]^2\,(Q4^{(1,0)}[x,y]+y\,Q4^{(1,1)}[x,y])))))+ \\
& Q4[x,y] \\
& (-x\,y\,(-1+y^3)\,yh^4\,(1-y\,\alpha+y^2\,\alpha)\,Q2[x,y]^2\,Q4[x,y]\,Q5[x,y]^2\,Q1^{(0,1)}[x,y] \\
& (-2\,(2+y^3)\,Q2[x,y]+y\,(-1+y^3)\,(2\,(-1+x)^2\,x^2\,y^3\,(2+y^3)\,yh^2\,Q3[x,y]^2\,Q4[x,y]+(1-y\,\alpha+y^2\,\alpha)\,Q1^{(0,1)}[x,y]))+ \\
& x\,y\,yh^4\,Q1[x,y]\,Q2[x,y]^2\,Q4[x,y]\,Q5[x,y]^2 \\
& ((12\,y^2-3\,y^5+4\,\alpha-8\,y\,\alpha-14\,y^3\,\alpha+16\,y^4\,\alpha+y^6\,\alpha)\,Q2[x,y]+ \\
& (-1+y^3)\,(2\,(-1+x)^2\,x^2\,y^3\,(-1+y^3)\,yh^2\,(4+5\,y^3-2\,y\,\alpha-4\,y^4\,\alpha+3\,y^5\,\alpha)\,Q3[x,y]^2\,Q4[x,y]-(2+y^3)\,(1-y\,\alpha+y^2\,\alpha)\,(2\,(1-y\,\alpha+y^2\,\alpha)\,Q1^{(0,1)}[x,y]+Q2^{(0,1)}[x,y])))- \\
& Q1[x,y]^2 \\
& (x\,y^2\,(-1+y^3)^2\,yh^4\,(1-y\,\alpha+y^2\,\alpha)\,Q4[x,y]\,Q5[x,y]^2\,Q2^{(0,1)}[x,y] \\
& (2\,(-1+x)^2\,x^2\,y^3\,yh^2\,(-2+5\,y^3+y\,\alpha-4\,y^4\,\alpha+3\,y^5\,\alpha)\,Q3[x,y]^2\,Q4[x,y]+4\,(-1+x)^4\,x^2\,(-1+2\,x)\,y^6\,(-1+y^3)\,yh^2\,(1-y\,\alpha+y^2\,\alpha)\,Q3[x,y]^3\,Q4[x,y]+ \\
& 4\,(-1+x)^4\,x^4\,y^7\,(-1+y^3)^2\,yh^4\,Q3[x,y]^4\,Q4[x,y]-3\,(1-y\,\alpha+y^2\,\alpha)\,Q2^{(0,1)}[x,y]-4\,(-1+x)^3\,x\,y^2\,(1-y\,\alpha+y^2\,\alpha)\,Q3[x,y]\,Q2^{(1,0)}[x,y])+ \\
& 2\,x\,(-1+y^3)\,Q2[x,y]^3\,(yh^4\,Q4[x,y]\,(y\,(-3\,y^2+2\,\alpha-4\,y\,\alpha+y^3\,\alpha+y^4\,\alpha)\,Q5[x,y]-2\,(1-y\,\alpha+y^2\,\alpha)^2\,Q5[x,y]^2\,(-3-2\,x^{2\,n1}\,y^2\,Q6[x,y]^2+x^{4\,n1}\,y^4\,Q6[x,y]^4)+ \\
& 2\,y\,(-1+y^3)\,(1-y\,\alpha+y^2\,\alpha)\,Q5^{(0,1)}[x,y])+y\,Q5[x,y]\,(yh^4\,(-3\,y^2+2\,\alpha-4\,y\,\alpha+y^3\,\alpha+y^4\,\alpha)\,Q5[x,y]+(-1+x)^6\,y^3\,(1-y\,\alpha+y^2\,\alpha)^2\,(2\,Q7[x,y]+x\,Q7^{(1,0)}[x,y])^2))+
\end{aligned}
$$

Typical Einstein Equation of 2D holographic superconductor

$$
\begin{aligned}
&32\,x^{1+2n1}\,y^4\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] + 32\,x^{1+2n1}\,y^5\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] + \\
&64\,x^{1+2n1}\,y^7\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] - 64\,x^{1+2n1}\,y^8\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] - \\
&32\,x^{1+2n1}\,y^{10}\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] + 32\,x^{1+2n1}\,y^{11}\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] + \\
&16\,x^{1+2n1}\,y^5\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] - 32\,x^{1+2n1}\,y^6\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] + \\
&16\,x^{1+2n1}\,y^7\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] - 32\,x^{1+2n1}\,y^8\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] + \\
&64\,x^{1+2n1}\,y^9\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] - 32\,x^{1+2n1}\,y^{10}\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] + \\
&16\,x^{1+2n1}\,y^{11}\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] - 32\,x^{1+2n1}\,y^{12}\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] + \\
&16\,x^{1+2n1}\,y^{13}\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6[x,y]\,Q6^{(0,1)}[x,y] + 8\,x^{1+2n1}\,y^4\,yh^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 - 16\,x^{1+2n1}\,y^7\,yh^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 + \\
&8\,x^{1+2n1}\,y^{10}\,yh^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 - 16\,x^{1+2n1}\,y^5\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 + 16\,x^{1+2n1}\,y^6\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 + \\
&32\,x^{1+2n1}\,y^8\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 - 32\,x^{1+2n1}\,y^9\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 - 16\,x^{1+2n1}\,y^{11}\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 + \\
&16\,x^{1+2n1}\,y^{12}\,yh^2\,\alpha\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 + 8\,x^{1+2n1}\,y^6\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 - 16\,x^{1+2n1}\,y^7\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 + \\
&8\,x^{1+2n1}\,y^8\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 - 16\,x^{1+2n1}\,y^9\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 + 32\,x^{1+2n1}\,y^{10}\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 - \\
&16\,x^{1+2n1}\,y^{11}\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 + 8\,x^{1+2n1}\,y^{12}\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 - 16\,x^{1+2n1}\,y^{13}\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 + \\
&8\,x^{1+2n1}\,y^{14}\,yh^2\,\alpha^2\,Q4[x,y]\,Q5[x,y]^2\,Q6^{(0,1)}[x,y]^2 + 2\,x^3\,y^4\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^4\,y^4\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + \\
&2\,x^5\,y^4\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^3\,y^7\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 8\,x^4\,y^7\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - \\
&4\,x^5\,y^7\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 2\,x^3\,y^{10}\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^4\,y^{10}\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + \\
&2\,x^5\,y^{10}\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^3\,y^5\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 8\,x^4\,y^5\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - \\
&4\,x^5\,y^5\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 4\,x^3\,y^6\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 8\,x^4\,y^6\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + \\
&4\,x^5\,y^6\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 8\,x^3\,y^8\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 16\,x^4\,y^8\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + \\
&8\,x^5\,y^8\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 8\,x^3\,y^9\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 16\,x^4\,y^9\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - \\
&8\,x^5\,y^9\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^3\,y^{11}\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 8\,x^4\,y^{11}\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - \\
&4\,x^5\,y^{11}\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 4\,x^3\,y^{12}\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 8\,x^4\,y^{12}\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + \\
&4\,x^5\,y^{12}\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 2\,x^3\,y^6\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^4\,y^6\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + \\
&2\,x^5\,y^6\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^3\,y^7\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 8\,x^4\,y^7\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - \\
&4\,x^5\,y^7\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 2\,x^3\,y^8\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^4\,y^8\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + \\
&2\,x^5\,y^8\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^3\,y^9\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 8\,x^4\,y^9\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - \\
&4\,x^5\,y^9\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 8\,x^3\,y^{10}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 16\,x^4\,y^{10}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + \\
&8\,x^5\,y^{10}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^3\,y^{11}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 8\,x^4\,y^{11}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - \\
&4\,x^5\,y^{11}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 2\,x^3\,y^{12}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^4\,y^{12}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + \\
&2\,x^5\,y^{12}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^3\,y^{13}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 8\,x^4\,y^{13}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - \\
&4\,x^5\,y^{13}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 2\,x^3\,y^{14}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 - 4\,x^4\,y^{14}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + \\
&2\,x^5\,y^{14}\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q7^{(0,1)}[x,y]^2 + 2\,y^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 6\,x\,y^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 6\,x^2\,y^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - \\
&2\,x^3\,y^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 2\,y^5\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 6\,x\,y^5\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 6\,x^2\,y^5\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + \\
&2\,x^3\,y^5\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 4\,y^3\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 12\,x\,y^3\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - \\
&12\,x^2\,y^3\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 4\,x^3\,y^3\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 4\,y^4\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - \\
&12\,x\,y^4\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 12\,x^2\,y^4\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 4\,x^3\,y^4\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + \\
&4\,y^6\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 12\,x\,y^6\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 12\,x^2\,y^6\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - \\
&4\,x^3\,y^6\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 4\,y^7\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 12\,x\,y^7\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - \\
&12\,x^2\,y^7\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 4\,x^3\,y^7\,\alpha\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 2\,y^4\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] -
\end{aligned}
$$

Typical Einstein Equation of 2D holographic superconductor

$$
\begin{aligned}
&6\,x\,y^4\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 6\,x^2\,y^4\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 2\,x^3\,y^4\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - \\
&4\,y^5\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 12\,x\,y^5\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 12\,x^2\,y^5\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + \\
&4\,x^3\,y^5\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 2\,y^6\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 6\,x\,y^6\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + \\
&6\,x^2\,y^6\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 2\,x^3\,y^6\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 2\,y^7\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + \\
&6\,x\,y^7\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 6\,x^2\,y^7\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 2\,x^3\,y^7\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + \\
&4\,y^8\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 12\,x\,y^8\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 12\,x^2\,y^8\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - \\
&4\,x^3\,y^8\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 2\,y^9\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 6\,x\,y^9\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - \\
&6\,x^2\,y^9\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] + 2\,x^3\,y^9\,\alpha^2\,Q4[x,y]\,Q5[x,y]\,Q2^{(1,0)}[x,y] - 4\,x\,y^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 12\,x^2\,y^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - \\
&12\,x^3\,y^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 4\,x^4\,y^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 4\,x\,y^5\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 12\,x^2\,y^5\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + \\
&12\,x^3\,y^5\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 4\,x^4\,y^5\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 8\,x\,y^3\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 24\,x^2\,y^3\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + \\
&24\,x^3\,y^3\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 8\,x^4\,y^3\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 8\,x\,y^4\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 24\,x^2\,y^4\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - \\
&24\,x^3\,y^4\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 8\,x^4\,y^4\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 8\,x\,y^6\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 24\,x^2\,y^6\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - \\
&24\,x^3\,y^6\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 8\,x^4\,y^6\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 8\,x\,y^7\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 24\,x^2\,y^7\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + \\
&24\,x^3\,y^7\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 8\,x^4\,y^7\,\alpha\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 4\,x^4\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 12\,x^2\,y^4\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - \\
&12\,x^3\,y^4\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 4\,x^4\,y^4\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 8\,x\,y^5\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 24\,x^2\,y^5\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + \\
&24\,x^3\,y^5\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 8\,x^4\,y^5\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 4\,x\,y^6\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 12\,x^2\,y^6\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - \\
&12\,x^3\,y^6\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 4\,x^4\,y^6\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 4\,x\,y^7\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 12\,x^2\,y^7\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + \\
&12\,x^3\,y^7\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 4\,x^4\,y^7\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 8\,x\,y^8\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 24\,x^2\,y^8\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - \\
&24\,x^3\,y^8\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 4\,x^4\,y^8\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 4\,x\,y^9\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 12\,x^2\,y^9\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + \\
&12\,x^3\,y^9\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] - 4\,x^4\,y^9\,\alpha^2\,Q5[x,y]^2\,Q2^{(1,0)}[x,y] + 2\,x\,y^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 8\,x^2\,y^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&12\,x^3\,y^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 8\,x^4\,y^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 2\,x^5\,y^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 2\,x\,y^5\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&8\,x^2\,y^5\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 12\,x^3\,y^5\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 8\,x^4\,y^5\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 2\,x^5\,y^5\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - \\
&4\,x\,y^3\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 16\,x^2\,y^3\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 24\,x^3\,y^3\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 16\,x^4\,y^3\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - \\
&4\,x^5\,y^3\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 4\,x\,y^4\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 16\,x^2\,y^4\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 24\,x^3\,y^4\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - \\
&16\,x^4\,y^4\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 4\,x^5\,y^4\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 4\,x\,y^6\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 16\,x^2\,y^6\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&24\,x^3\,y^6\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 16\,x^4\,y^6\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 4\,x^5\,y^6\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 4\,x\,y^7\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&16\,x^2\,y^7\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 24\,x^3\,y^7\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 16\,x^4\,y^7\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 4\,x^5\,y^7\,\alpha\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&2\,x\,y^4\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 8\,x^2\,y^4\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 12\,x^3\,y^4\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 8\,x^4\,y^4\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&2\,x^5\,y^4\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 4\,x\,y^5\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 16\,x^2\,y^5\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 24\,x^3\,y^5\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&16\,x^4\,y^5\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 4\,x^5\,y^5\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 2\,x\,y^6\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 8\,x^2\,y^6\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&12\,x^3\,y^6\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 8\,x^4\,y^6\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 2\,x^5\,y^6\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 2\,x\,y^7\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&8\,x^2\,y^7\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 12\,x^3\,y^7\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 8\,x^4\,y^7\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 2\,x^5\,y^7\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&4\,x\,y^8\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 16\,x^2\,y^8\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 24\,x^3\,y^8\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 16\,x^4\,y^8\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&4\,x^5\,y^8\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 2\,x\,y^9\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 8\,x^2\,y^9\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 12\,x^3\,y^9\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + \\
&8\,x^4\,y^9\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] - 2\,x^5\,y^9\,\alpha^2\,Q5[x,y]^2\,Q2^{(2,0)}[x,y] + 4\,(-1+x)^2\,x\,y^3\,(-1+y^3)^2\,(1-y\,\alpha+y^2\,\alpha)^2\,Q3[x,y]\,Q4[x,y] \\
&\Big(yh^2\,Q4[x,y]\,\big(2\,Q5[x,y] - y\,Q5^{(0,1)}[x,y]\big) + Q5[x,y]\,\big(-(-1+x)^3\,x^2\,y^3\,Q7^{(0,1)}[x,y]\big)\,\big(2\,Q7[x,y] + x\,Q7^{(1,0)}[x,y]\big) + \\
&\qquad yh^2\,Q5[x,y]\,\big(2-4\,x+(-1+x)^4\,x^2\,y^3\,Q3^{(2,0)}[x,y]\big)\Big) + 2\,(-1+x)^2\,x\,y^4\,(-1+y^3)^2\,Q3[x,y]^2 \\
&\Big(2\,yh^2\,Q4[x,y]^2\,\big(\big(-(-1+x)^2\,(-1+2\,x)\,y^2 - 2\,x^2\,yh^2 + x^2\,y\,yh^2\,\alpha - (-1+x)^2\,(-1+2\,x)\,y^6\,\alpha^2 + y^5\,\alpha\,\big(x^2\,\big(3\,yh^2 - 10\,\alpha\big) - 2\,\alpha + 8\,x\,\alpha + 4\,x^3\,\alpha\big) + \\
&\qquad y^3\,\big(5\,x^2\,\big(yh^2 - 2\,\alpha\big) - 2\,\alpha + 8\,x\,\alpha + 4\,x^3\,\alpha\big) + y^4\,\alpha\,\big(2+\alpha-4\,x\,(2+\alpha) - 2\,x^3\,(2+\alpha) + x^2\,\big(10 - 4\,yh^2 + 5\,\alpha\big)\big)\big)\,Q5[x,y] + \\
&\qquad x^2\,yh^2\,\big(1-y\,\alpha+y^2\,\alpha\big)^2\,Q5[x,y]^2\,\big(-3-2\,x^{2\,n1}\,y^2\,Q6[x,y]^2 + x^{4\,n1}\,y^4\,Q6[x,y]^4\big) - x^2\,y\,\big(-1+y^3\big)\,yh^2\,\big(1-y\,\alpha+y^2\,\alpha\big)\,Q5^{(0,1)}[x,y]\big) -
\end{aligned}
$$

Typical Einstein Equation of 2D holographic superconductor

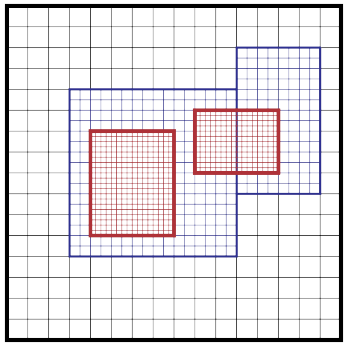
```
Q4[x, y] Q5[x, y] (-yh^2 (-12 x^2 yh^2 + 8 x^2 y yh^2 α + 2 (-1 + x)^2 (1 - 8 x + 10 x^2) y^6 α^2 + y^5 α (x^2 (19 yh^2 - 108 α) - 4 α + 40 x α + 112 x^3 α - 40 x^4 α) +
      y^3 (27 x^2 (yh^2 - 4 α) - 4 α + 40 x α + 112 x^3 α - 40 x^4 α) + y^2 (2 - 20 x - 56 x^3 + 20 x^4 + x^2 (54 - 4 yh^2 α)) +
      y^4 α (2 (2 + α) - 20 x (2 + α) - 56 x^3 (2 + α) + 20 x^4 (2 + α) + x^2 (-23 yh^2 + 54 (2 + α)))) Q5[x, y] + 2 (-1 + x)^6 x^2 y^4 (1 - y α + y^2 α)^2 (2 Q7[x, y] + x Q7^(1,0)[x, y])^2) +
(-1 + x)^4 x^2 y^2 yh^2 (1 - y α + y^2 α)^2 Q5[x, y]^2 Q4^(2,0)[x, y])))));
```

- Indexing for tensors and finite differencing for simulations in C/Fortran becomes impractical
- Packages exist that do this for many spacetime solvers for cases involving black holes, neutron stars, etc. But what about more exotic cases (cosmic strings) or completely unrelated cases (Lattice field theory problems)
- We need a better method for automating translation of symbolic expressions to executable C/Fortran code

Anatomy of PDE Solver

HPC

Block structured AMR
Parallel Data Structures
Inter-node Communication
CPU/GPU Architectures



PDE Solver

Declaration of Data
- 3D Arrays, etc

Initialize Data
Level Advance
- RHS for time integrator
- Subcycling + RK4

Logic for Refining

C/Fortran

Evolution equations

Typed in symbolic/tensor form
copied from “textbook” expressions

python/sympy

Code Gen

Discretize symbolic equations

Convert symbolic expressions to C or Fortran

Wishlist for a code generator for symbolic tensor manipulations

- Specification of finite difference ordering, stenciling, dimensionality etc. as inputs which code gen can automatically expand into correct compilable expressions.
- Objects such as metric induced connections/covariant derivatives, Riemann tensors, and generalized formulas for energy momentum tensors should be standard in the code generator.
- Syntaxing sufficiently versatile to easily write code in many different languages with minimal tinkering from the user.

We use Code Gen to translate python to C++

- Fields contained in 4-dimensional arrays indexed by 3-dimensional grid points i, j, k, and a fourth index n for multiple components like vector fields etc.

- E.g. a field variable $\psi(x,y,z)$ is contained in the object *array(i, j, k, Idx::Psi)*

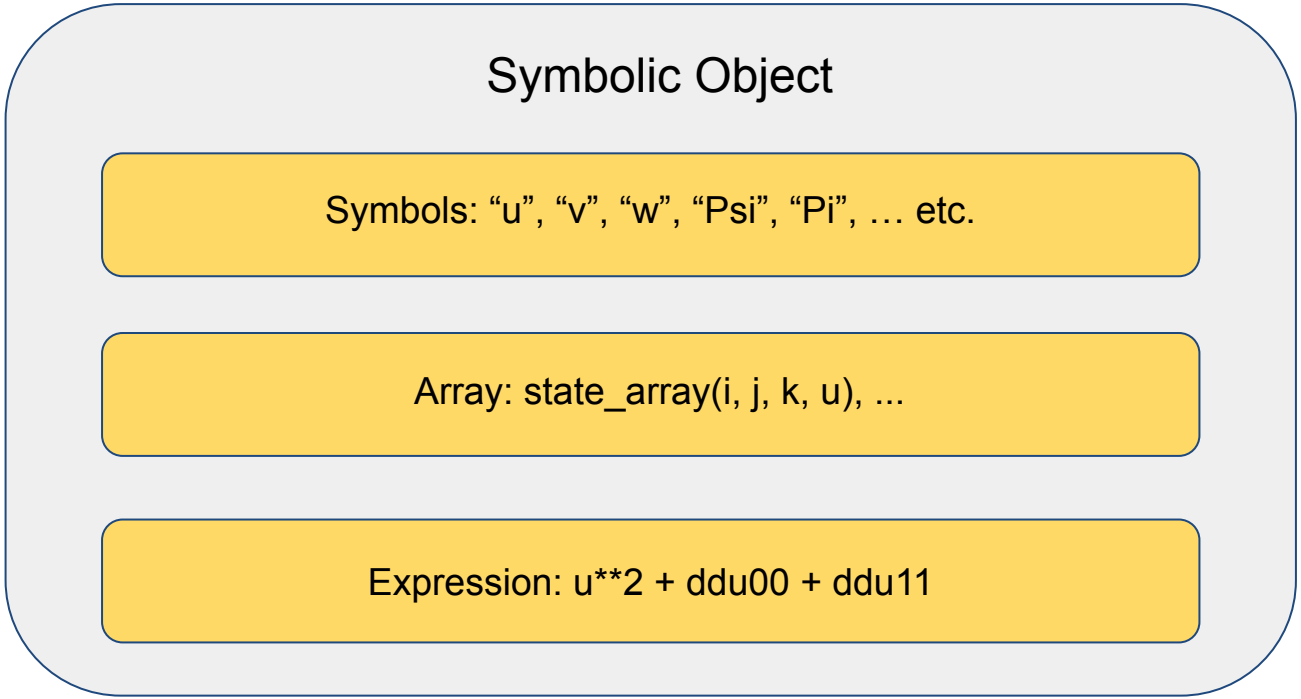
- Expanded finite differencing statements:

  $\partial_x u \rightarrow$ *(array(i+1,j,k,u) - array(i-1,j,k,u))/(2*dx[0])*

- Expanded indexed expressions:

  $\sum_k u_k v_k \rightarrow$ *array(i,j,k,u1)*array(i,j,k,v1)+array(i,j,k,u2)*array(i,j,k,v2)*

Objects contain symbolic name, array info, and expression



Code gen can print C/Fortran statements in various forms

In [1]:
```
from SpacetimeVarNew import *
```

In [2]:
```
stVar.declState = []
u = stVar('u', state = True)
v = stVar('v', state = True)
v.expr = u.symb**2 + Dsymb(u.symb,'00') + Dsymb(u.symb, '11')
```

Generating finite differencing for derivatives

```
In [8]: fileRHS.write(DstVar(Psi, 1, orderD = DiffOrder, DIM = 2).AMReXSymb2Expr())
        fileRHS.write(DstVar(Psi, 2, orderD = DiffOrder, DIM = 2).AMReXSymb2Expr())
```

```
        amrex::Real dPsi0 = ((2.0/3.0)*state_fab(i + 1, j, k, Idx::Psi) - 1.0/12.0*state_fab(i + 2, j, k, Idx::Psi) -
2.0/3.0*state_fab(i - 1, j, k, Idx::Psi) + (1.0/12.0)*state_fab(i - 2, j, k, Idx::Psi))/dx[0];
        amrex::Real dPsi1 = ((2.0/3.0)*state_fab(i, j + 1, k, Idx::Psi) - 1.0/12.0*state_fab(i, j + 2, k, Idx::Psi) -
2.0/3.0*state_fab(i, j - 1, k, Idx::Psi) + (1.0/12.0)*state_fab(i, j - 2, k, Idx::Psi))/dx[1];
        amrex::Real ddPsi00 = ((4.0/3.0)*state_fab(i + 1, j, k, Idx::Psi) - 1.0/12.0*state_fab(i + 2, j, k, Idx::Psi) +
(4.0/3.0)*state_fab(i - 1, j, k, Idx::Psi) - 1.0/12.0*state_fab(i - 2, j, k, Idx::Psi) - 5.0/2.0*state_fab(i, j, k,
Idx::Psi))/std::pow(dx[0], 2);
        amrex::Real ddPsi11 = ((4.0/3.0)*state_fab(i, j + 1, k, Idx::Psi) - 1.0/12.0*state_fab(i, j + 2, k, Idx::Psi) +
(4.0/3.0)*state_fab(i, j - 1, k, Idx::Psi) - 1.0/12.0*state_fab(i, j - 2, k, Idx::Psi) - 5.0/2.0*state_fab(i, j, k,
Idx::Psi))/std::pow(dx[1], 2);
```

Generate equations of motion

Declare, generate, and write right hand side for KG equations $\dot{\psi} = \pi, \quad \dot{\pi} = \nabla^2\psi - m^2\psi$

In [11]:
```
RHS_Psi = stVar('Psi')
RHS_Pi = stVar('Pi')
```

In [12]:
```
RHS_Psi.expr = Pi.symb
RHS_Pi.expr += Dsymb(Psi.symb, '00') + Dsymb(Psi.symb, '11')-m.symb**2*Psi.symb
```

In [13]:
```
fileRHS.write(RHS_Psi.AMReXSetRHS())
fileRHS.write(RHS_Pi.AMReXSetRHS())
```

Generate equations of motion

Declare, generate, and write right hand side for KG equations $\dot{\psi} = \pi, \quad \dot{\pi} = \nabla^2\psi - m^2\psi$

In [11]:
```
RHS_Psi = stVar('Psi')
RHS_Pi = stVar('Pi')
```

In [12]:
```
RHS_Psi.expr = Pi.symb
RHS_Pi.expr += Dsymb(Psi.symb, '00') + Dsymb(Psi.symb, '11')-m.symb**2*Psi.symb
```

In [13]:
```
fileRHS.write(RHS_Psi.AMReXSetRHS())
fileRHS.write(RHS_Pi.AMReXSetRHS())
```

Generate equations of motion

Declare, generate, and write right hand side for KG equations $\dot{\psi} = \pi, \quad \dot{\pi} = \nabla^2\psi - m^2\psi$

In [11]:
```
RHS_Psi = stVar('Psi')
RHS_Pi = stVar('Pi')
```

In [12]:
```
RHS_Psi.expr = Pi.symb
RHS_Pi.expr += Dsymb(Psi.symb, '00') + Dsymb(Psi.symb, '11')-m.symb**2*Psi.symb
```

In [13]:
```
fileRHS.write(RHS_Psi.AMReXSetRHS())
fileRHS.write(RHS_Pi.AMReXSetRHS())
```

```
rhs_fab(i, j, k, Idx::Psi) = Pi;
rhs_fab(i, j, k, Idx::Pi) = -Psi*std::pow(m, 2) + ddPsi00 + ddPsi11;
```

Reminder of Einstein equation in Z4c formulation

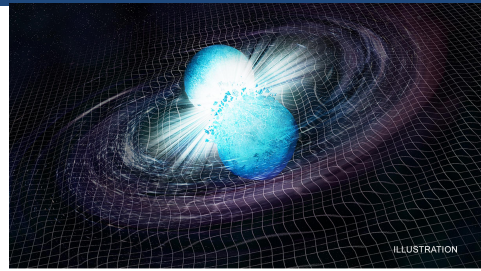


Illustration: CXC/M. Weiss

- Sample of evolution equations in 3+1 Z4c formulation:

$$\partial_t \tilde{\gamma}_{ij} = -2\alpha \tilde{A}_{ij} + 2\tilde{\gamma}_{k(i}\partial_{j)}\beta^k - \tfrac{2}{3}\tilde{\gamma}_{ij}\partial_k\beta^k + \beta^k\partial_k\tilde{\gamma}_{ij}$$

$$\begin{aligned}\partial_t \tilde{A}_{ij} = {} & \chi[-D_iD_j\alpha + \alpha(R_{ij} - 8\pi S_{ij})]^{\mathrm{tf}} \\ & + \alpha[(\hat{K} + 2\Theta)\tilde{A}_{ij} - 2\tilde{A}_{ik}\tilde{A}^k_j] \\ & + 2\tilde{A}_{k(i}\partial_{j)}\beta^k - \tfrac{2}{3}\tilde{A}_{ij}\partial_k\beta^k + \beta^k\partial_k\tilde{A}_{ij}\end{aligned}$$

$$\partial_t \chi = 2/3\chi[\alpha(\hat{K} + 2\Theta) - D_i\beta^i]$$

- More equations and auxiliary expressions not shown

We can do the same thing for the more complex Z4c

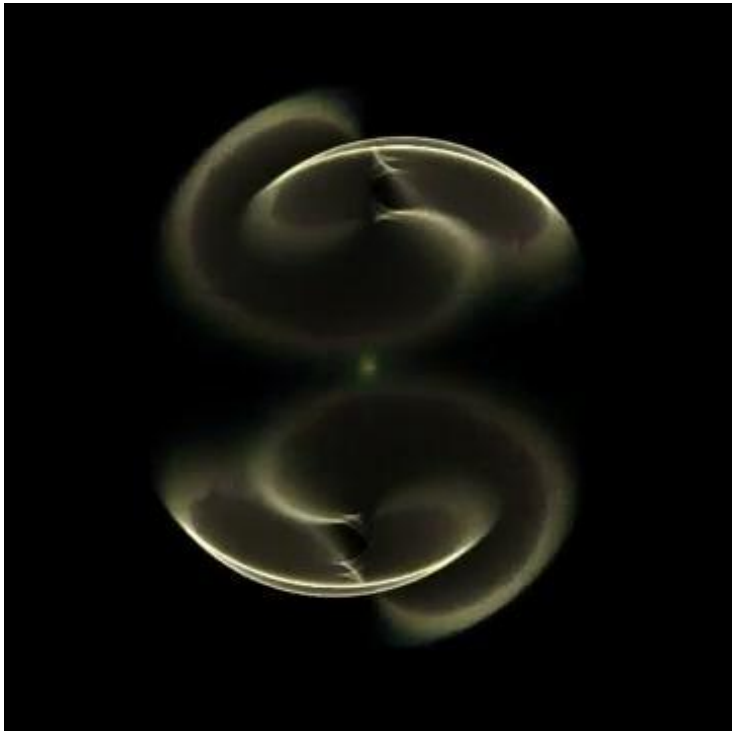
In [59]:
```
RHS_Atilde_LL = stvarrank2('Atilde_LL', sym = 'sym01', gridvar = True, addtolist = False, vartype = "state_fab", varprefix = "rhs_fab")

for i in range(3):
    for j in range(3):
        RHS_Atilde_LL.expr[i][j] += chi.symb*(-CovDDalphaTF_LL.symb[i][j]+alpha.symb*RTF_LL.symb[i][j])+alpha.symb*(Khat.symb+2*theta.symb)*Atilde_LL.s
        RHS_Atilde_LL.expr[i][j] += AdvDAtilde_LL.symb[i][j]
        for k in range(3):
            RHS_Atilde_LL.expr[i][j] += -2*alpha.symb*Atilde_LL.symb[i][k]*Atilde_UL.symb[k][j]
            RHS_Atilde_LL.expr[i][j] += Atilde_LL.symb[k][i]*dDbeta_UL.symb[k][j]+Atilde_LL.symb[k][j]*dDbeta_UL.symb[k][i]-2/3*Atilde_LL.symb[i][j]*dD

for i in range(3):
    for j in range(3):
        RHS_Atilde_LL.expr[i][j] += KOsigma.symb*dKODFullAtilde_LL.symb[i][j]
```

In [60]:
```
RHSString += RHS_Atilde_LL.setisymb()
```

We can do the same thing for the more complex Z4c

```
rhs_fab(i, j, k, Idx::Atilde_LL_00) = AdvDbetaAtilde_LL_00 - 2*Atilde_LL_00*Atilde_UL_00*alpha + Atilde_LL_00*alpha*(Khat + 2*theta) +
(4.0/3.0)*Atilde_LL_00*dDbeta_UL_00 - 2.0/3.0*Atilde_LL_00*dDbeta_UL_11 - 2.0/3.0*Atilde_LL_00*dDbeta_UL_22 -
2*Atilde_LL_01*Atilde_UL_10*alpha + 2*Atilde_LL_01*dDbeta_UL_10 - 2*Atilde_LL_02*Atilde_UL_20*alpha + 2*Atilde_LL_02*dDbeta_UL_20 +
KOSigma*dKODAtilde_LL_00 + (-CovDDalphaTF_LL_00 + RTF_LL_00*alpha)*std::exp(-4*phi);

rhs_fab(i, j, k, Idx::Atilde_LL_01) = AdvDbetaAtilde_LL_01 - 2*Atilde_LL_00*Atilde_UL_01*alpha + Atilde_LL_00*dDbeta_UL_01 -
2*Atilde_LL_01*Atilde_UL_11*alpha + Atilde_LL_01*alpha*(Khat + 2*theta) + (1.0/3.0)*Atilde_LL_01*dDbeta_UL_00 +
(1.0/3.0)*Atilde_LL_01*dDbeta_UL_11 - 2.0/3.0*Atilde_LL_01*dDbeta_UL_22 - 2*Atilde_LL_02*Atilde_UL_21*alpha + Atilde_LL_02*dDbeta_UL_21 +
Atilde_LL_11*dDbeta_UL_10 + Atilde_LL_12*dDbeta_UL_20 + KOSigma*dKODAtilde_LL_01 + (-CovDDalphaTF_LL_01 + RTF_LL_01*alpha)*std::exp(-4*phi);

rhs_fab(i, j, k, Idx::Atilde_LL_02) = AdvDbetaAtilde_LL_02 - 2*Atilde_LL_00*Atilde_UL_02*alpha + Atilde_LL_00*dDbeta_UL_02 -
2*Atilde_LL_01*Atilde_UL_12*alpha + Atilde_LL_01*dDbeta_UL_12 - 2*Atilde_LL_02*Atilde_UL_22*alpha + Atilde_LL_02*alpha*(Khat + 2*theta) +
(1.0/3.0)*Atilde_LL_02*dDbeta_UL_00 - 2.0/3.0*Atilde_LL_02*dDbeta_UL_11 + (1.0/3.0)*Atilde_LL_02*dDbeta_UL_22 + Atilde_LL_12*dDbeta_UL_10 +
Atilde_LL_22*dDbeta_UL_20 + KOSigma*dKODAtilde_LL_02 + (-CovDDalphaTF_LL_02 + RTF_LL_02*alpha)*std::exp(-4*phi);

rhs_fab(i, j, k, Idx::Atilde_LL_11) = AdvDbetaAtilde_LL_11 - 2*Atilde_LL_01*Atilde_UL_01*alpha + 2*Atilde_LL_01*dDbeta_UL_01 -
2*Atilde_LL_11*Atilde_UL_11*alpha + Atilde_LL_11*alpha*(Khat + 2*theta) - 2.0/3.0*Atilde_LL_11*dDbeta_UL_00 +
(4.0/3.0)*Atilde_LL_11*dDbeta_UL_11 - 2.0/3.0*Atilde_LL_11*dDbeta_UL_22 - 2*Atilde_LL_12*Atilde_UL_21*alpha + 2*Atilde_LL_12*dDbeta_UL_21 +
KOSigma*dKODAtilde_LL_11 + (-CovDDalphaTF_LL_11 + RTF_LL_11*alpha)*std::exp(-4*phi);

rhs_fab(i, j, k, Idx::Atilde_LL_12) = AdvDbetaAtilde_LL_12 - 2*Atilde_LL_01*Atilde_UL_02*alpha + Atilde_LL_01*dDbeta_UL_02 +
Atilde_LL_02*dDbeta_UL_01 - 2*Atilde_LL_11*Atilde_UL_12*alpha + Atilde_LL_11*dDbeta_UL_12 - 2*Atilde_LL_12*Atilde_UL_22*alpha +
Atilde_LL_12*alpha*(Khat + 2*theta) - 2.0/3.0*Atilde_LL_12*dDbeta_UL_00 + (1.0/3.0)*Atilde_LL_12*dDbeta_UL_11 +
(1.0/3.0)*Atilde_LL_12*dDbeta_UL_22 + Atilde_LL_22*dDbeta_UL_21 + KOSigma*dKODAtilde_LL_12 + (-CovDDalphaTF_LL_12 +
RTF_LL_12*alpha)*std::exp(-4*phi);

rhs_fab(i, j, k, Idx::Atilde_LL_22) = AdvDbetaAtilde_LL_22 - 2*Atilde_LL_02*Atilde_UL_02*alpha + 2*Atilde_LL_02*dDbeta_UL_02 -
2*Atilde_LL_12*Atilde_UL_12*alpha + 2*Atilde_LL_12*dDbeta_UL_12 - 2*Atilde_LL_22*Atilde_UL_22*alpha + Atilde_LL_22*alpha*(Khat + 2*theta) -
2.0/3.0*Atilde_LL_22*dDbeta_UL_00 - 2.0/3.0*Atilde_LL_22*dDbeta_UL_11 + (4.0/3.0)*Atilde_LL_22*dDbeta_UL_22 + KOSigma*dKODAtilde_LL_22 + (-
CovDDalphaTF_LL_22 + RTF_LL_22*alpha)*std::exp(-4*phi);
```

Black hole merger produced from Z4c code generator



How do the waveforms compare to observations?

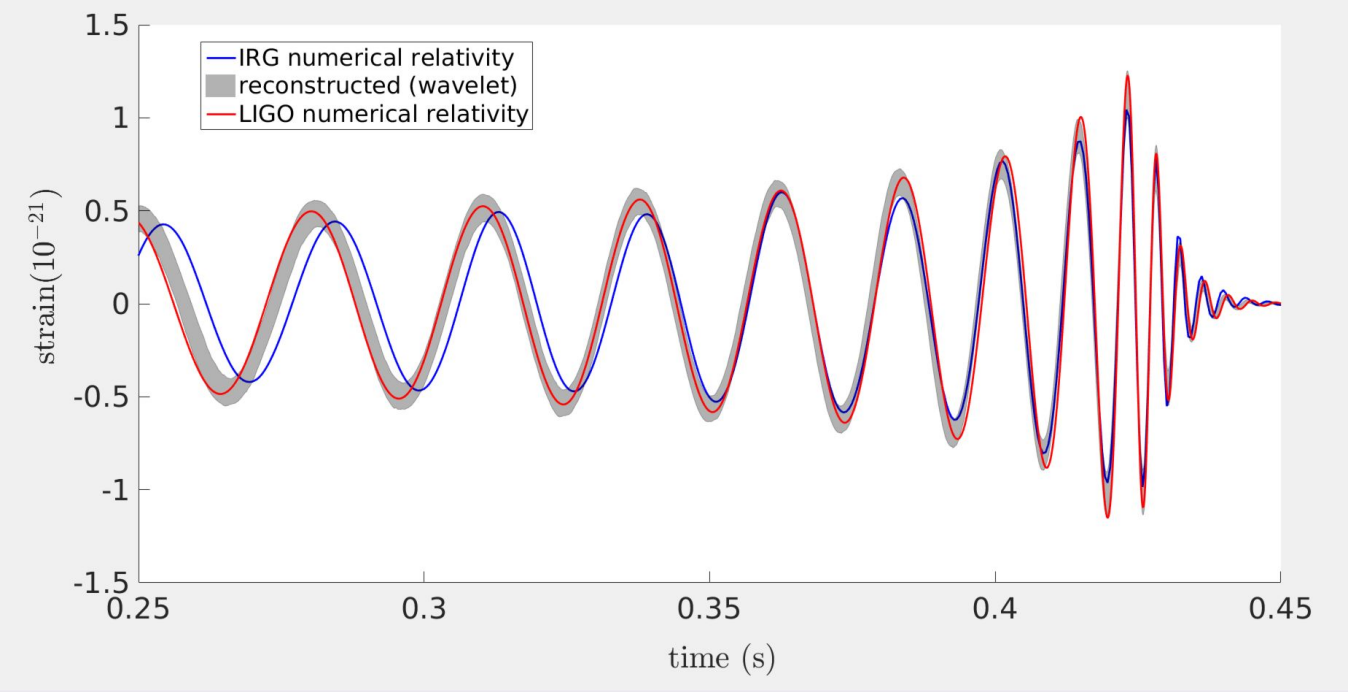
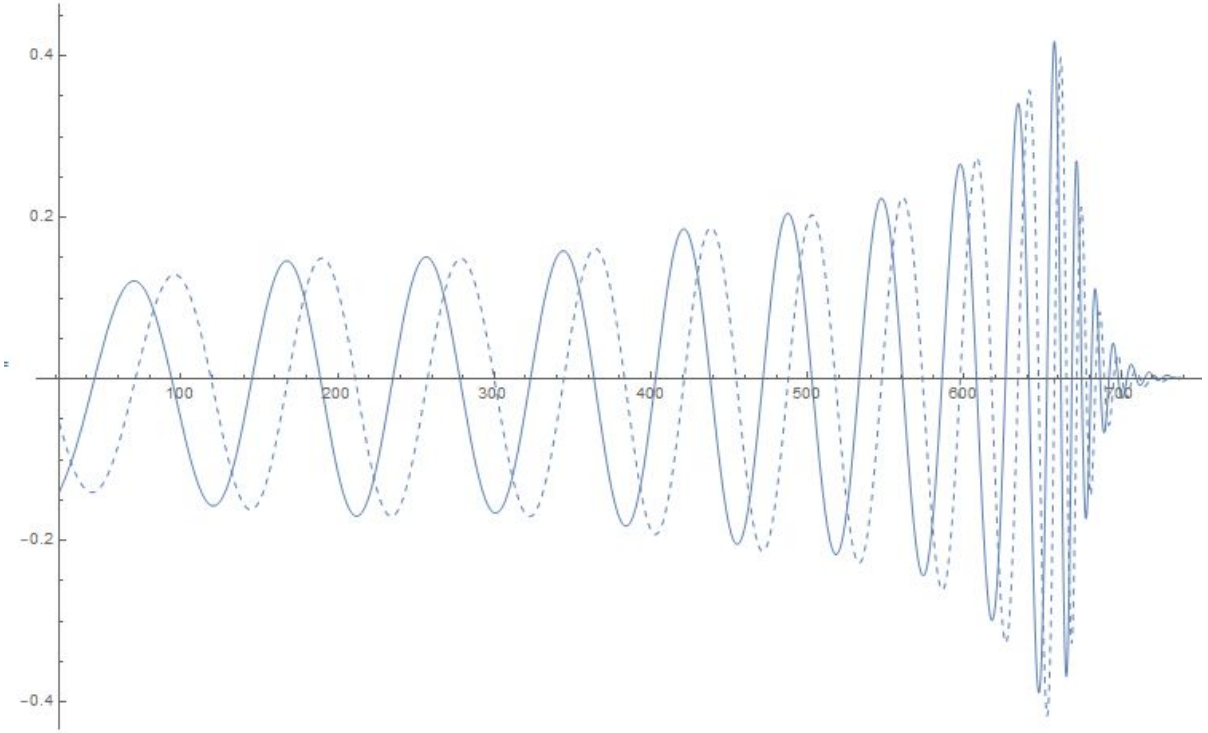
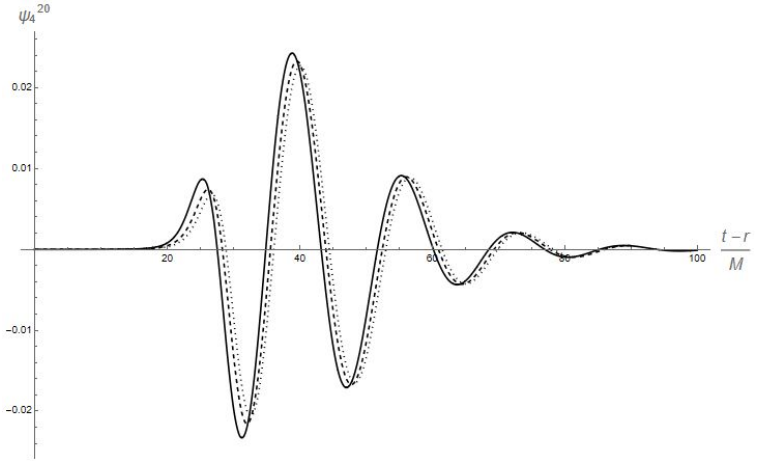


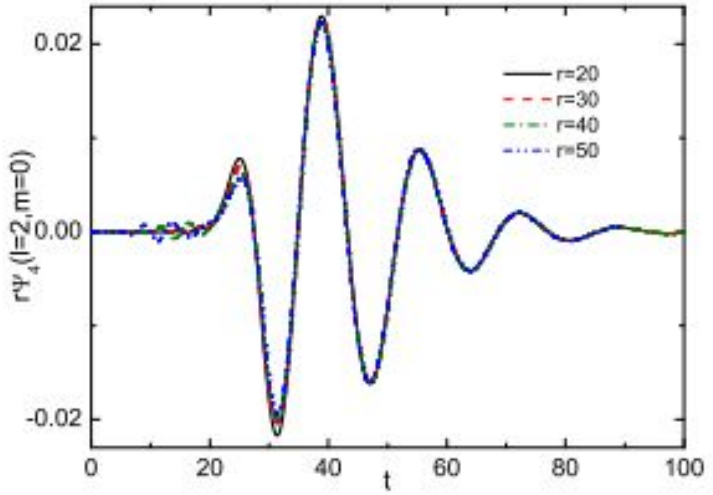
Image Reference: https://losc.ligo.org/events/GW150914

How do the waveforms compare to observations?



How does the waveform compare to other groups?

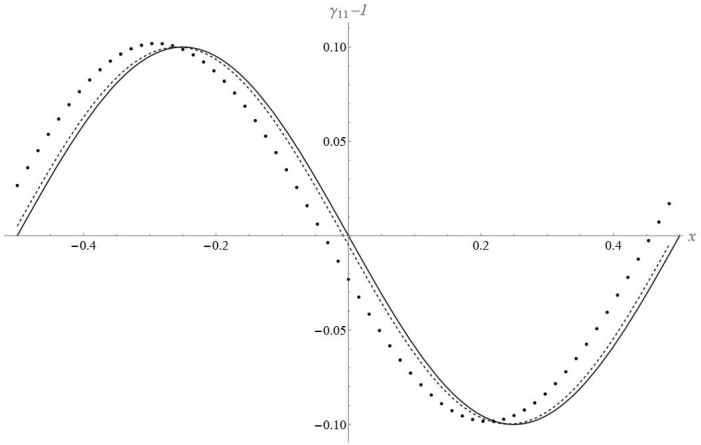


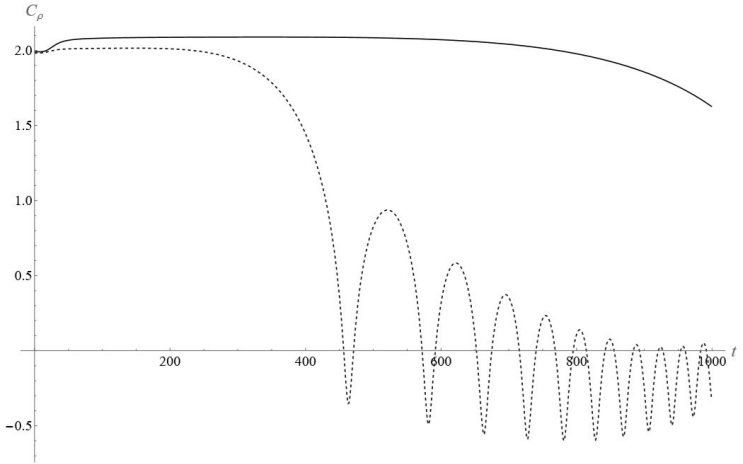


Source: Z. Cao, H-J. Yo, J-P. Yu, arXiv:0812.0641 (2008).

To be more precise…

- Convergence testing on gauge waves shows consistency to 4th order finite differencing

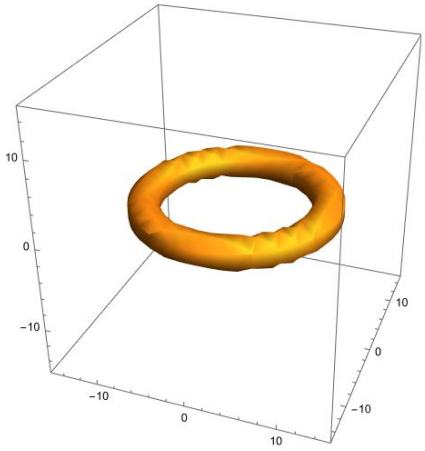




$$C_\rho = \log_4\left(\frac{||u_\rho - u_{\rho/2}||}{||u_{\rho/2} - u_{\rho/4}||}\right)$$

Venturing into the world of theoretical cosmology (Cosmic Strings)

- All has been done before, so what about new problems?
- Take the problem of collapsing cosmic string loops with internal currents. Can the code generator be of use here?
- Initial data becomes more complicated
- Matter sources with energy/momentum become involved



Cosmic string with non-Abelian internal current

$$G^{\alpha\beta} = 8\pi T^{\alpha\beta}$$

$$-D_\mu D^\mu \phi + \frac{\partial V(\phi)}{\partial \bar{\phi}} = 0$$

$$\nabla_\mu F^{\mu\nu} = -eJ^\nu \quad \nabla_\mu A^\mu = 0$$

$$-\nabla_\mu \nabla^\mu \sigma^i + \frac{\partial V(\phi,\sigma)}{\partial \sigma^i} = 0$$

$$V(\phi,\sigma) = \frac{\lambda}{2}(|\phi|^2 + |\sigma|^2 - \eta^2)^2 + \delta|\sigma|^2|\phi|^2$$

- Initial ansatz involves a time dependent winding: $\sigma(t,r,\theta,\varphi) = \sigma(r,\theta)\exp\{i(\omega t - m\varphi)\}$
- Initial data now involve energy density and angular momentum terms.

In addition to Z4c we have field equations for matter sources

- Including field equations for matter fields and electromagnetic fields…

$$\partial_t \phi_a = \alpha \Pi_{M,a} + \beta^i \partial_i \phi_a,$$

$$\begin{aligned}\partial_t \Pi_{M,a} = {} & \beta^i \partial_i \Pi_{M,a} + \alpha \partial_i \partial^i \phi_a + \partial_i \phi_a \partial^i \alpha \\ & + \alpha \left(K \Pi_{M,a} - \gamma^{ij} \Gamma^k_{ij} \partial_k \phi_a + \frac{dV}{d\phi_a} \right) \\ & + \alpha(-e^2 A_\mu A^\mu \phi_a \pm e \phi_{a+1} \nabla_\mu A^\mu \\ & \pm 2 e A^\mu \partial_\mu \phi_{a+1}),\end{aligned}$$

$$\begin{aligned}\partial_t E^i = {} & \alpha K E^i + e \alpha \chi \tilde{\gamma}^{ij} \mathcal{J}_j + \alpha \chi \tilde{\gamma}^{ij} \partial_j Z \\ & + \chi^2 \tilde{\gamma}^{ij} \tilde{\gamma}^{kl} \partial_l \alpha (\partial_j \mathcal{A}_k - \partial_k \mathcal{A}_j) \\ & + \chi^2 \tilde{\gamma}^{ij} \tilde{\gamma}^{kl} (\tilde{D}_k \partial_j \mathcal{A}_l - \tilde{D}_k \partial_l \mathcal{A}_j) \\ & + \frac{\alpha}{2} \chi \tilde{\gamma}^{ij} \tilde{\gamma}^{kl} (\partial_j \mathcal{A}_l \partial_k \chi - \partial_k \mathcal{A}_j \partial_l \chi) \\ & + \beta^j \partial_j E^i - E^j \partial_j \beta^i\end{aligned}$$

$$\begin{aligned}\partial_t \mathcal{A} = {} & \alpha K \mathcal{A} - \alpha \chi \tilde{\gamma}^{ij} \partial_j \mathcal{A}_i + \alpha \chi \mathcal{A}_i \tilde{\Gamma}^i - \alpha Z \\ & + \frac{\alpha}{2} \mathcal{A}_i \tilde{\gamma}^{ij} \partial_j \chi - \chi \tilde{\gamma}^{ij} \mathcal{A}_i \partial_j \alpha + \beta^j \partial_j \mathcal{A},\end{aligned}$$

$$\begin{aligned}\partial_t \mathcal{A}_i = {} & -\alpha \chi^{-1} \tilde{\gamma}_{ij} E^j - \alpha \partial_i \mathcal{A} - \mathcal{A} \partial_i \alpha \\ & + \beta^j \partial_j \mathcal{A}_i + \partial_i \beta^j \mathcal{A}_j,\end{aligned}$$

$$\partial_t Z = \alpha \tilde{\nabla}_i E^i - \frac{3}{2} \frac{\alpha}{\chi} E^i \partial_i \chi - \alpha e \mathcal{J} - \alpha \kappa Z + \beta^j \partial_j Z,$$

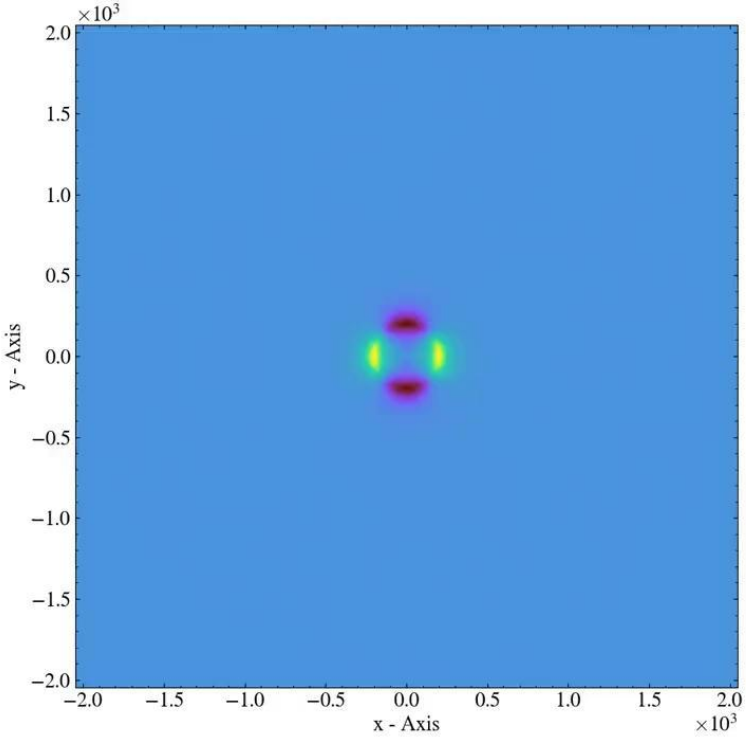
Taking the electric EOM as an example

In [118]:
```
RHS_E_U = stvarrank1('E_U', gridvar = True, addtolist = False, vartype = "state_fab", varprefix = "rhs_fab")

for i in range(3):
    RHS_E_U.expr[i] += alpha.symb*Ksclr.symb*E_U.symb[i] + AdvDE_U.symb[i] + dKODFullE_U.symb[i]
    RHS_E_U.expr[i] += -alpha.symb*echarge.symb*scrJ_U.symb[i]

    for j in range(3):
        RHS_E_U.expr[i] += alpha.symb*chi.symb*gamtilde_UU.symb[i][j]*dDZ_L.symb[j]
        RHS_E_U.expr[i] += -E_U.symb[j]*dDbeta_UL.symb[i][j]

        for k in range(3):
            for l in range(3):
                RHS_E_U.expr[i] += gam_UU.symb[i][j]*gam_UU.symb[k][l]*dDalpha_L.symb[l]*(dDA_LL.symb[k][j]-dDA_LL.symb[j][k])
                RHS_E_U.expr[i] += alpha.symb*gam_UU.symb[i][j]*gam_UU.symb[k][l]*(CovDtildedA_LLL.symb[l][j][k]-CovDtildedA_LLL.
                RHS_E_U.expr[i] += (alpha.symb/2)*chi.symb*gamtilde_UU.symb[i][j]*gamtilde_UU.symb[k][l]*(dDA_LL.symb[l][j]*dDchi

RHSString += RHS_E_U.setisymb()
```

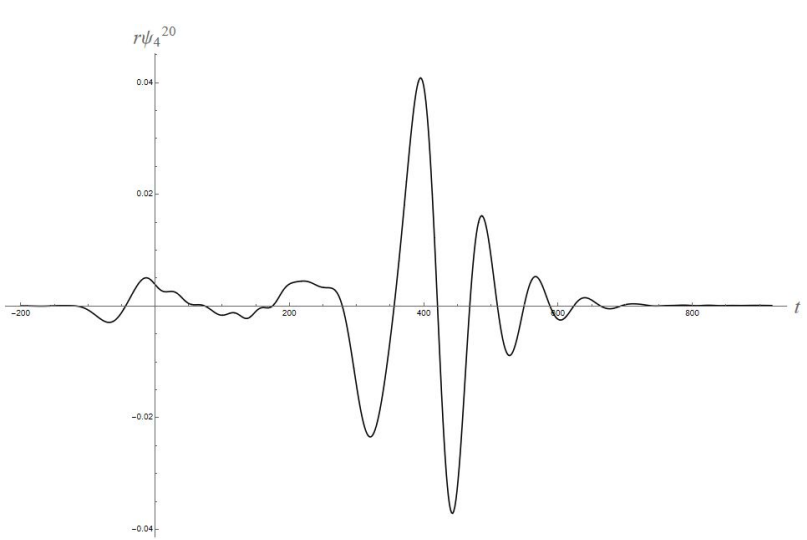
Printing the EOM for electric field in executable form

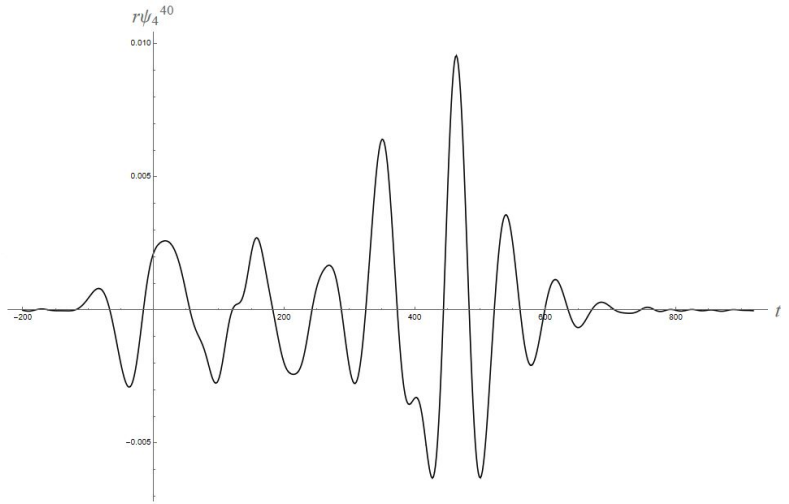
```
rhs_fab(i, j, k, state_fabIdx::E_U_0) = AdvDbetaE_U_0 + E_U_0*Ksclr*alpha - E_U_0*dDbeta_UL_00 - E_U_1*dDbeta_UL_01 - E_U_2*dDbeta_UL_02 +
alpha*chi*dDZ_L_0*invgamtilde_UU_00 + alpha*chi*dDZ_L_1*invgamtilde_UU_01 + alpha*chi*dDZ_L_2*invgamtilde_UU_02 +
(1.0/2.0)*alpha*chi*invgamtilde_UU_00*invgamtilde_UU_01*(-dDA_LL_00*dDchi_L_1 + dDA_LL_10*dDchi_L_0) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_00*invgamtilde_UU_01*(dDA_LL_00*dDchi_L_1 - dDA_LL_01*dDchi_L_0) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_00*invgamtilde_UU_01*(dDA_LL_01*dDchi_L_0 - dDA_LL_10*dDchi_L_0) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_00*invgamtilde_UU_02*(-dDA_LL_00*dDchi_L_2 + dDA_LL_20*dDchi_L_0) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_00*invgamtilde_UU_02*(dDA_LL_00*dDchi_L_2 - dDA_LL_02*dDchi_L_0) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_00*invgamtilde_UU_02*(dDA_LL_02*dDchi_L_0 - dDA_LL_20*dDchi_L_0) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_00*invgamtilde_UU_11*(-dDA_LL_01*dDchi_L_1 + dDA_LL_10*dDchi_L_1) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_00*invgamtilde_UU_12*(-dDA_LL_01*dDchi_L_2 + dDA_LL_20*dDchi_L_1) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_00*invgamtilde_UU_12*(-dDA_LL_02*dDchi_L_1 + dDA_LL_10*dDchi_L_2) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_00*invgamtilde_UU_22*(-dDA_LL_02*dDchi_L_2 + dDA_LL_20*dDchi_L_2) +
(1.0/2.0)*alpha*chi*std::pow(invgamtilde_UU_01, 2)*(dDA_LL_01*dDchi_L_1 - dDA_LL_11*dDchi_L_0) +
(1.0/2.0)*alpha*chi*std::pow(invgamtilde_UU_01, 2)*(-dDA_LL_10*dDchi_L_1 + dDA_LL_11*dDchi_L_0) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_01*invgamtilde_UU_02*(dDA_LL_01*dDchi_L_2 - dDA_LL_12*dDchi_L_0) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_01*invgamtilde_UU_02*(dDA_LL_02*dDchi_L_1 - dDA_LL_21*dDchi_L_0) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_01*invgamtilde_UU_02*(-dDA_LL_10*dDchi_L_2 + dDA_LL_21*dDchi_L_0) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_01*invgamtilde_UU_02*(dDA_LL_12*dDchi_L_0 - dDA_LL_20*dDchi_L_1) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_01*invgamtilde_UU_12*(-dDA_LL_11*dDchi_L_2 + dDA_LL_21*dDchi_L_1) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_01*invgamtilde_UU_12*(dDA_LL_11*dDchi_L_2 - dDA_LL_12*dDchi_L_1) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_01*invgamtilde_UU_22*(-dDA_LL_12*dDchi_L_2 + dDA_LL_21*dDchi_L_2) +
(1.0/2.0)*alpha*chi*std::pow(invgamtilde_UU_02, 2)*(dDA_LL_02*dDchi_L_2 - dDA_LL_22*dDchi_L_0) +
(1.0/2.0)*alpha*chi*std::pow(invgamtilde_UU_02, 2)*(-dDA_LL_20*dDchi_L_2 + dDA_LL_22*dDchi_L_0) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_02*invgamtilde_UU_11*(dDA_LL_12*dDchi_L_1 - dDA_LL_21*dDchi_L_1) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_02*invgamtilde_UU_12*(dDA_LL_12*dDchi_L_2 - dDA_LL_22*dDchi_L_1) +
(1.0/2.0)*alpha*chi*invgamtilde_UU_02*invgamtilde_UU_12*(-dDA_LL_21*dDchi_L_2 + dDA_LL_22*dDchi_L_1) - alpha*echarge*scrJ_U_0 +
alpha*invgam_UU_00*invgam_UU_01*(-CovDtildedA_LLL_010 + CovDtildedA_LLL_100) + alpha*invgam_UU_00*invgam_UU_01*(CovDtildedA_LLL_010 -
CovDtildedA_LLL_100) + alpha*invgam_UU_00*invgam_UU_02*(-CovDtildedA_LLL_020 + CovDtildedA_LLL_200) + alpha*invgam_UU_00*invgam_UU_02*
(CovDtildedA_LLL_020 - CovDtildedA_LLL_200) + alpha*invgam_UU_00*invgam_UU_11*(-CovDtildedA_LLL_011 + CovDtildedA_LLL_101) +
alpha*invgam_UU_00*invgam_UU_12*(-CovDtildedA_LLL_012 + CovDtildedA_LLL_102) + alpha*invgam_UU_00*invgam_UU_12*(-CovDtildedA_LLL_021 +
CovDtildedA_LLL_201) + alpha*invgam_UU_00*invgam_UU_22*(-CovDtildedA_LLL_022 + CovDtildedA_LLL_202) + alpha*std::pow(invgam_UU_01, 2)*
(CovDtildedA_LLL_011 - CovDtildedA_LLL_101) + alpha*invgam_UU_01*invgam_UU_02*(CovDtildedA_LLL_012 - CovDtildedA_LLL_102) +
alpha*invgam_UU_01*invgam_UU_02*(CovDtildedA_LLL_021 - CovDtildedA_LLL_201) + alpha*invgam_UU_01*invgam_UU_02*(-CovDtildedA_LLL_120 +
CovDtildedA_LLL_210) + alpha*invgam_UU_01*invgam_UU_02*(CovDtildedA_LLL_120 - CovDtildedA_LLL_210) + alpha*invgam_UU_01*invgam_UU_12*(-
CovDtildedA_LLL_121 + CovDtildedA_LLL_211) + alpha*invgam_UU_01*invgam_UU_22*(-CovDtildedA_LLL_122 + CovDtildedA_LLL_212) +
alpha*std::pow(invgam_UU_02, 2)*(CovDtildedA_LLL_022 - CovDtildedA_LLL_202) + alpha*invgam_UU_02*invgam_UU_11*(CovDtildedA_LLL_121 -
CovDtildedA_LLL_211) + alpha*invgam_UU_02*invgam_UU_12*(CovDtildedA_LLL_122 - CovDtildedA_LLL_212) + dDalpha_L_0*invgam_UU_00*invgam_UU_01*
(-dDA_LL_01 + dDA_LL_10) + dDalpha_L_0*invgam_UU_00*invgam_UU_01*(dDA_LL_01 - dDA_LL_10) + dDalpha_L_0*invgam_UU_00*invgam_UU_02*(-dDA_LL_02
+ dDA_LL_20) + dDalpha_L_0*invgam_UU_00*invgam_UU_02*(dDA_LL_02 - dDA_LL_20) + dDalpha_L_0*invgam_UU_01*invgam_UU_02*(-dDA_LL_12 +
dDA_LL_21) + dDalpha_L_0*invgam_UU_01*invgam_UU_02*(dDA_LL_12 - dDA_LL_21) + dDalpha_L_1*invgam_UU_00*invgam_UU_11*(-dDA_LL_01 + dDA_LL_10)
+ dDalpha_L_1*invgam_UU_00*invgam_UU_12*(-dDA_LL_02 + dDA_LL_20) + dDalpha_L_1*std::pow(invgam_UU_01, 2)*(dDA_LL_01 - dDA_LL_10) +
dDalpha_L_1*invgam_UU_01*invgam_UU_02*(dDA_LL_02 - dDA_LL_20) + dDalpha_L_1*invgam_UU_01*invgam_UU_12*(-dDA_LL_12 + dDA_LL_21) +
dDalpha_L_1*invgam_UU_02*invgam_UU_11*(dDA_LL_12 - dDA_LL_21) + dDalpha_L_2*invgam_UU_00*invgam_UU_12*(-dDA_LL_01 + dDA_LL_10) +
dDalpha_L_2*invgam_UU_00*invgam_UU_22*(-dDA_LL_02 + dDA_LL_20) + dDalpha_L_2*invgam_UU_01*invgam_UU_02*(dDA_LL_01 - dDA_LL_10) +
dDalpha_L_2*invgam_UU_01*invgam_UU_22*(-dDA_LL_12 + dDA_LL_21) + dDalpha_L_2*std::pow(invgam_UU_02, 2)*(dDA_LL_02 - dDA_LL_20) +
dDalpha_L_2*invgam_UU_02*invgam_UU_12*(dDA_LL_12 - dDA_LL_21) + dKODE_U_0;
```

Well behaved cosmic vortons (m = 2)



Extracting the cosmic string waveforms





Summary

- Code generation allows us to build a fully functional spacetime solver in C++ using symbolic manipulation in python/sympy or Mathematica
- Effectively bridges the gap between symbolic textbook expressions and equivalent numerical code
- Sufficiently versatile to be made compatible with other architectures based on C or Fortran, e.g. Flash-X applications based on Fortran.
- Code generation is more than a single application. It is a method.

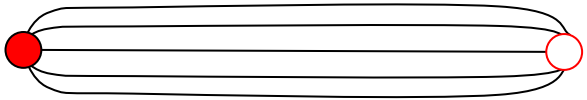
Great! We have a code generator, now what?

- Translating large sets of EOMs in GR is not the only use for code gen
- Can we use it for generating code for simulating the behavior of subatomic physics?
- Research suggests that multigrid methods applied to Lattice gauge theories can resolve current problems such as topological freezing etc.
- Can code generators provide an avenue to translate LQFT expressions into useable code for AMR applications?

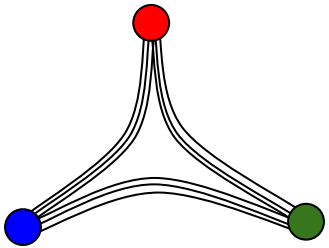
Quantum Chromodynamics (QCD) Review

- Hadronic matter including protons, neutrons, and pions are made up of quarks and gluons

Meson (Pions, Kaons,...)



Baryon (Protons, Neutrons,...)

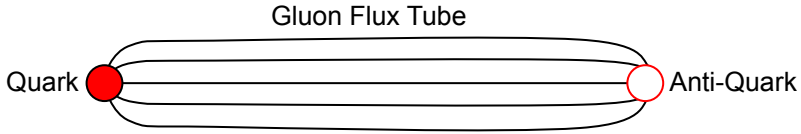


- Quantum path integral integrates over all fields weighted by a phase term:

- QCD: $\int d[\psi]d[\bar{\psi}]d[A]e^{i\int d^4x\mathcal{L}}$ $\mathcal{L} = -\frac{1}{4}\mathrm{Tr}G_{\mu\nu}G^{\mu\nu} + \bar{\psi}\gamma^{\mu}(\partial_{\mu} + igA_{\mu})\psi$

Confinement prevents direct observation of quarks and gluons

- Charged particles in EM obey inverse square law potential
- Charges for strong coupling tend to form tightly bound flux tube ‘strings’, and obey linear potentials. This is known as confinement



- Confinement suggested for SU(n) gauge theory
- Strong coupling makes the theory inaccessible to standard perturbation techniques

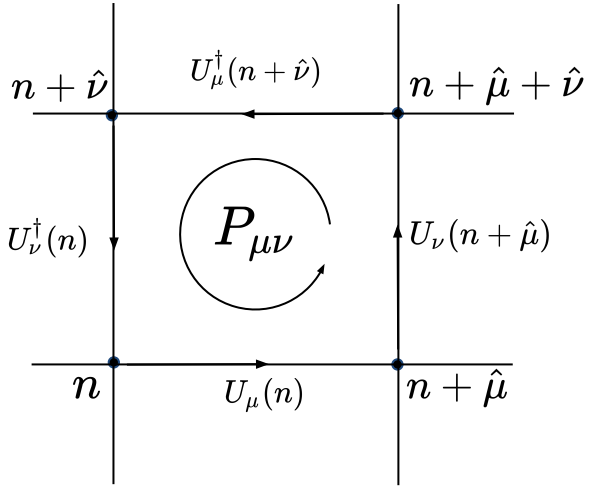
Lattice Field Theory allows us to approach strongly coupled models

- Fields discretized on a lattice with finite spacing and number of sites.

$$S_G[U] = \beta \sum_{n\in\Lambda} \sum_{\mu<\nu} \mathrm{Re}\,[1 - P_{\mu\nu}]$$

$$P_{\mu\nu} = U_\mu(n) U_\nu(n+\hat{\mu}) U_\mu(n+\hat{\nu})^\dagger U_\nu(n)^\dagger$$

$$S_F = a^4 \sum_{n\in\Lambda} \left(\bar{\psi}\gamma_\mu \frac{U_\mu(n)\psi(n+\hat{\mu})}{2a} + \mathrm{h.c.} \right) + m\bar{\psi}\psi$$



- Lattice actions agree with continuum versions in the limit of small lattice spacing

2-D Schwinger Models let us study confinement in simplified setting

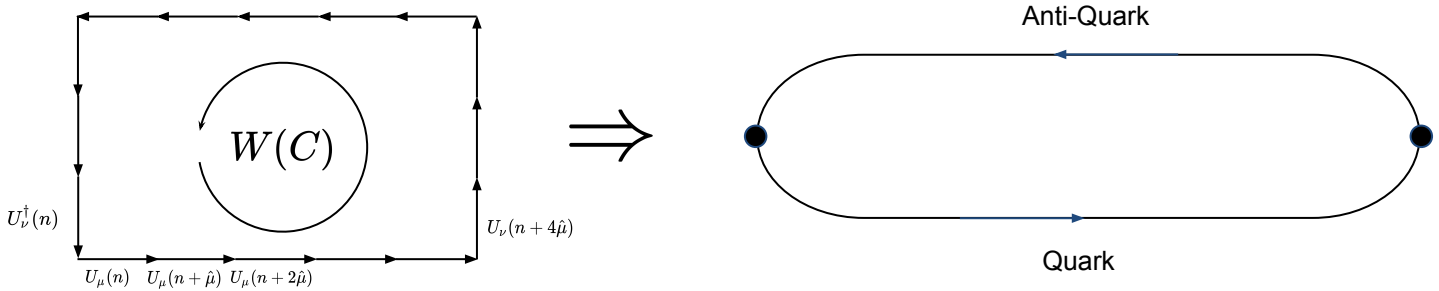
- 2 dimensional simplification of gauge models like QCD
- Schwinger model can be exactly solved analytically and with lattice models.
- U(1) gauge group is Abelian (just like Electromagnetism)
- One spatial dimension implies confinement since flux lines are restricted to only one available dimension

Wilson Loops

- Useful object for studying gauge theories in non-perturbative limit is the Wilson loop

$$W(C) = P \exp\left[ig \oint_C dx^\mu A_\mu\right] \rightarrow \prod_{n,\mu \in C} U_\mu(n)$$

- Can be interpreted as adding a current interacting with the gauge field. In otherwards, it measures the action associated with separated charges.



Confinement is studied with Wilson loops

- Taking the quantum average of the Wilson loops determines the vacuum energy contribution to the gauge field excited by the quark pair

$$\langle W(C)\rangle = \int d[\Phi] W(C) e^{-S[\Phi]}$$

$$\langle W(C)\rangle = \sum_n A_n e^{-TE_n} \rightarrow e^{-TE_0} = e^{-TV(R)}$$

- We can study potential between charges by calculating the quantum average of Wilson loops of various spatial widths

$$\log\langle W(R,T)\rangle \propto TV(R)$$

What about the fermion 'quark' contribution?

- Fermions are represented by anti-commuting Grassmann numbers.
- Gaussian integrals of Grassmann numbers are counterintuitive

$$\int d[U]d[\psi]d[\bar{\psi}]\exp\left(-S_U+\bar{\psi}D\psi\right)=\int d[U]\det D^\dagger D\exp(-S_U)$$

- Determinant is extremely difficult to calculate
- Solution is to use pseudo-fermions

$$\det D^\dagger D=\int d[\phi]d[\phi^\star]\exp\left(-\phi^\star(D^\dagger D)^{-1}\phi\right)$$

- Drawback is we have to calculate differential operator, multiply by conjugate and invert

$$D=(m+2r)\delta_{nm}-r\sum_{\mu=\pm1}^{\pm2}(1-\gamma_\mu)U_\mu(n)\delta_{n+\hat{\mu},m}$$

Dirac Operators are a particular challenge to code in AMReX

$$D = (m + 2r)\delta_{nm} - r\sum_{\mu=\pm1}^{\pm2}(1 - \gamma_\mu)U_\mu(n)\delta_{n+\hat{\mu},m}$$

- This differential is a complex off diagonal matrix, mixing components, and mixing real and imaginary parts.
- Operator must be multiplied with conjugate and inverted.
- Extracting real and imaginary parts for all components is a nightmare!

Code Generation translates symbolic operators to AMReX code

- Code generator used earlier along with Sympy features is perfect for this task.

In [8]:
```
def MuDPsi(psi_U,u_U, r):
    MuDpsi_U = stvarrank1('uDpsi_U', dim = 2)

    conjugateu_U = stvarrank1('conju_U', dim = 2)
    conjugateu_U.isymb[0] = sp.conjugate(u_U.isymb[0])
    conjugateu_U.isymb[1] = sp.conjugate(u_U.isymb[1])

    psi_U_tmp = sp.Matrix([psi_U.isymb[0], psi_U.isymb[1]])

    M1 = sp.eye(2)*r - gamma_1
    M2 = sp.eye(2)*r - gamma_2

    Mm1 = sp.eye(2)*r + gamma_1
    Mm2 = sp.eye(2)*r + gamma_2

    M1psi_U = M1*psi_U_tmp
    M2psi_U = M2*psi_U_tmp

    Mm1psi_U = Mm1*psi_U_tmp
    Mm2psi_U = Mm2*psi_U_tmp

    MuDpsi_U.isymb[0] = u_U.isymb[0]*shift(M1psi_U[0],[1,0,0,0])+u_U.isymb[1]*shift(M2psi_U[0],[0,1,0,0])
    MuDpsi_U.isymb[1] = u_U.isymb[0]*shift(M1psi_U[1],[1,0,0,0])+u_U.isymb[1]*shift(M2psi_U[1],[0,1,0,0])

    MuDpsi_U.isymb[0] += shift(conjugateu_U.isymb[0],[-1,0,0,0])*shift(Mm1psi_U[0],[-1,0,0,0])+shift(conjugateu_U.isymb[1],[0,-1
    MuDpsi_U.isymb[1] += shift(conjugateu_U.isymb[0],[-1,0,0,0])*shift(Mm1psi_U[1],[-1,0,0,0])+shift(conjugateu_U.isymb[1],[0,-1

    return MuDpsi_U
```

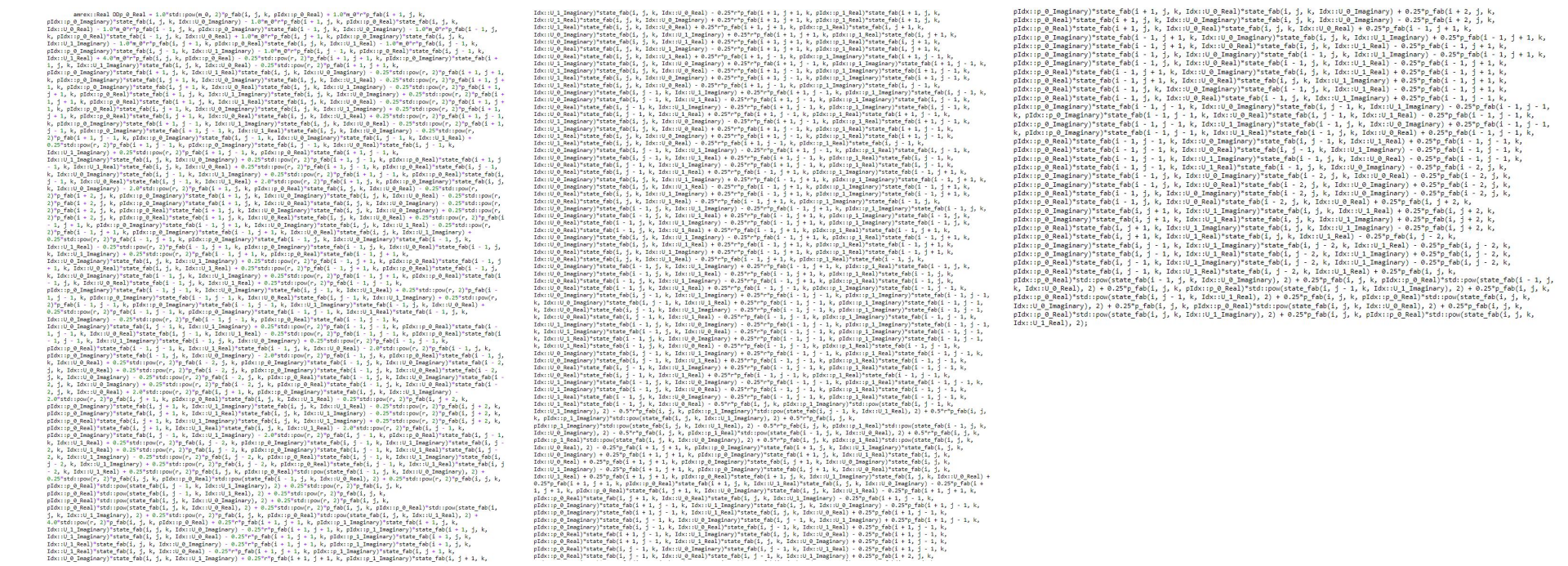
Code Generation translates symbolic operators to AMReX code

- Code generator used earlier along with Sympy features is perfect for this task.

In [11]:
```
def OpPsi(Psi_U, u_U, m, r):
    OpPsi_U = stvarrank1('OpPsi_U', dim = 2)

    OpPsi_U.isymb[0] = (m+2)*Psi_U.isymb[0]
    OpPsi_U.isymb[1] = (m+2)*Psi_U.isymb[1]

    MuDPsi_U = MuDPsi(Psi_U, u_U, r)

    OpPsi_U.isymb[0] += MuDPsi_U.isymb[0]
    OpPsi_U.isymb[1] += MuDPsi_U.isymb[1]

    OpPsi_U.isymb[0] = sp.simplify(OpPsi_U.isymb[0])
    OpPsi_U.isymb[1] = sp.simplify(OpPsi_U.isymb[1])


    return OpPsi_U
```

In [17]:
```
OpOpPsi_U = OpPsi(OpPsi(p_U,u_U,m,r), u_U,m,r)
```

Code Generation translates symbolic operators to AMReX code

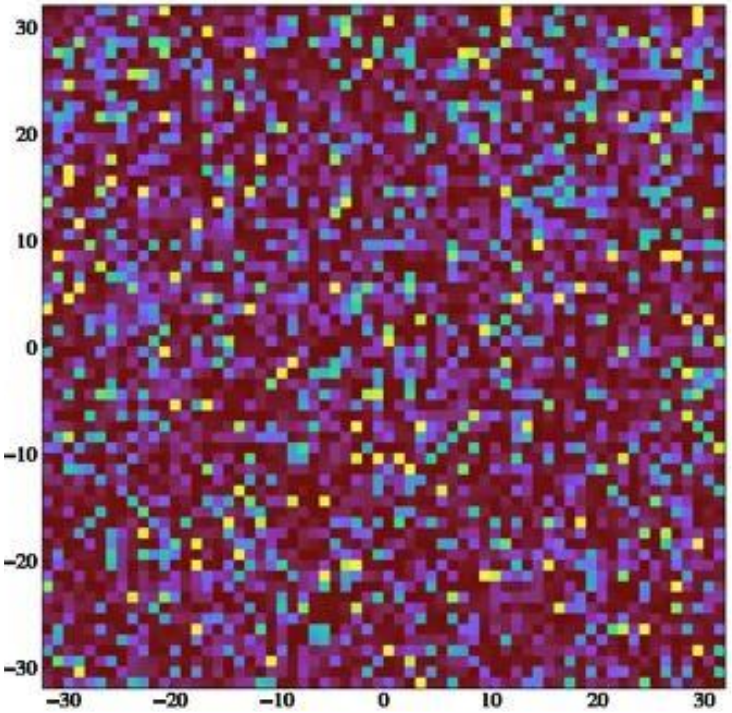


- Clearly code gen was the right move!

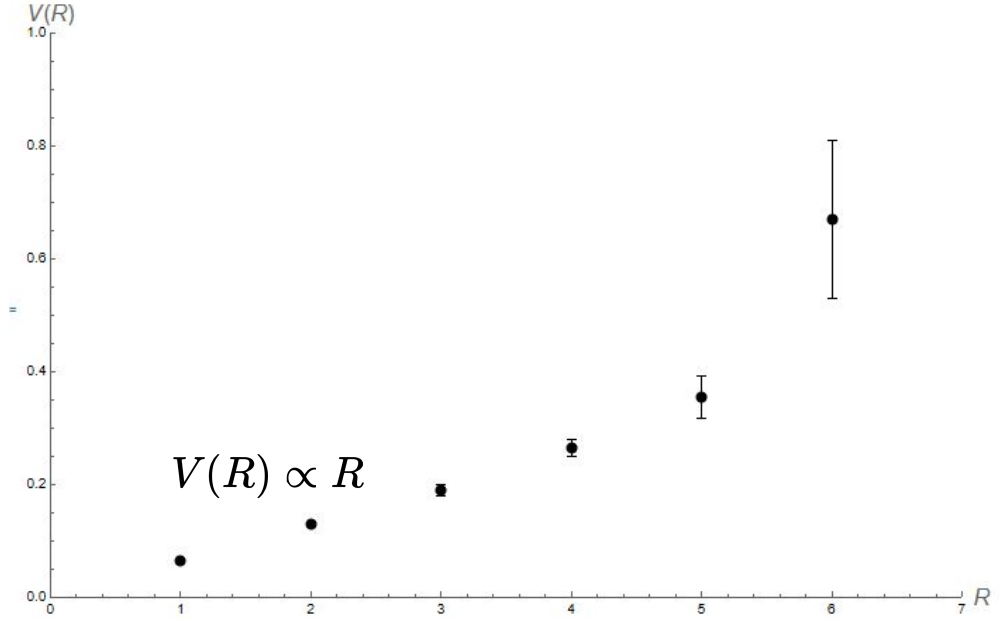
AMReX implementation of quantum lattice model

- The AMReX code implements Hybrid Monte-Carlo algorithm to calculate quantum averages:
    1. Perturb gauge fields and generate random pseudo-fermion field
    2. Evolve gauge fields using molecular dynamics leapfrog evolution
    3. Perform Monte-Carlo accept reject step by comparing initial action to final action
    4. Repeat
    5. Calculate average of observables (Wilson loops, potentials…)

2D Simulation of Schwinger model using HMC



Does it exhibit confining behavior?



Yes!

Summary

- Code generation streamlines the process of translating symbolic expressions to executable code for simulations
- We have demonstrated this process for the case of simulations using the AMReX architecture, specifically black hole evolution and lattice QFT
- Separation of variable information in the symbol objects makes generalization to other platforms easy

What's to be done?

- Code generation for matter filled spacetimes implementing hydro/MHD
- Future developments for AMReX based lattice QFT simulations will no doubt benefit from code generation techniques, especially as we consider more involved models (real QCD!... 4-D AMReX?)
- Implementation of multi-gridding in lattice simulations to improve statistics for observations

Acknowledgements

- Don Willcox (LBNL)
- Dean Howarth (LBNL)
- Erik Schnetter (Perimeter Institute)
- Vasili Mewes (Oak Ridge)
- Philipp Moesta (University of Amsterdam)
- Weiqun Zhang (LBNL)
- Ann Almgren (LBNL)