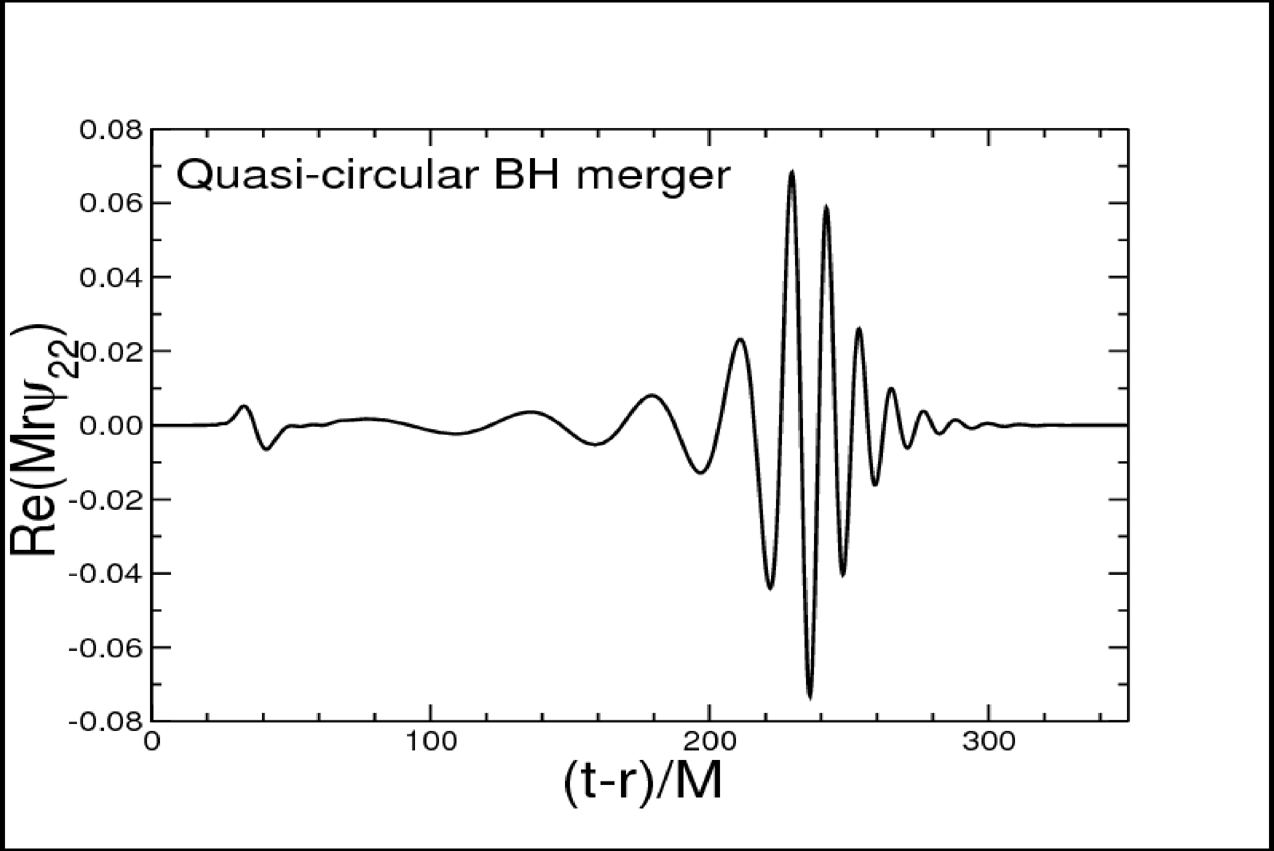
COMPUTATION OF BLACK HOLE QUASI-NORMAL RINGING BY THE "SHOOTING" METHOD

Created by Kyle MacKenzie under the advisement of Dr. Gaurav Khanna

QUASI-NORMAL RINGING (QNR)

- If disturbed, black holes “settle” with a very characteristic gravitational wave (GW) signal, called quasi-normal ringing (QNR).
- This is typically observed at the late stages of GW emission from a binary black hole merger.
- Computing these frequencies from theory is therefore, a very important problem in gravitational wave physics.

QUASI-NORMAL RINGING (QNR)



- This is an example of a binary black hole merger's signal, borrowed from CERN's database
- As one can see, the true wave function is rather complicated
- This project will serve as a precursor for such a complicated wave with some calculated simplifications

DERIVATION

- To use the “shooting” method, one must first have an ODE and two boundary conditions
- Computation of QNR for the simplest black hole may be reduced down to solving a simple wave equation with a potential with radiative boundary conditions

$$\frac{\partial^2 \varphi}{\partial t^2} - \frac{\partial^2 \varphi}{\partial x^2} + V(x)\varphi = 0$$

THE POTENTIAL

- For this project, a scalar field is used as a proxy for a true black hole potential

$$V(x) = \frac{1}{cosh^2(x)}$$

- The potential used is called the Poschl-Teller potential
  - This potential is very similar to that of a black hole
  - There are exact solutions to check against

DERIVATION

- Using the anzatz that $\varphi = \vartheta(x)e^{-iwt}$, one can remove the time dependence in the wave equation

$$\frac{\partial^2\vartheta}{\partial x^2} - V(x)\vartheta = -\omega^2\vartheta$$

- This is now an eigenvalue problem where ω^2 is a complex eigenvalue

DERIVATION

- Using the anzatz that $\vartheta = e^{ia(x)}$ and solving the equation for $a'(x)$, we can arrive at a simplified version of the equation we wish to solve numerically

$$a'(x) = -i(a(x)^2 + V(x) - w^2)$$

- This form solving for $a'(x)$ is easier to work with because it has very simple boundary conditions

$$a'(-\infty) \to -\omega$$
$$a(+\infty) \to +\omega$$

THE “SHOOTING” METHOD

- An iterative method for boundary-value problems for ordinary differential equations (ODEs)
- The method begins with having a know left and right boundary value
- Then, a variant of Newton’s method is used to iterate from the left boundary to the right boundary
- At this point, the numerical value is subtracted from the true value, at this right boundary, and this is taken as the error

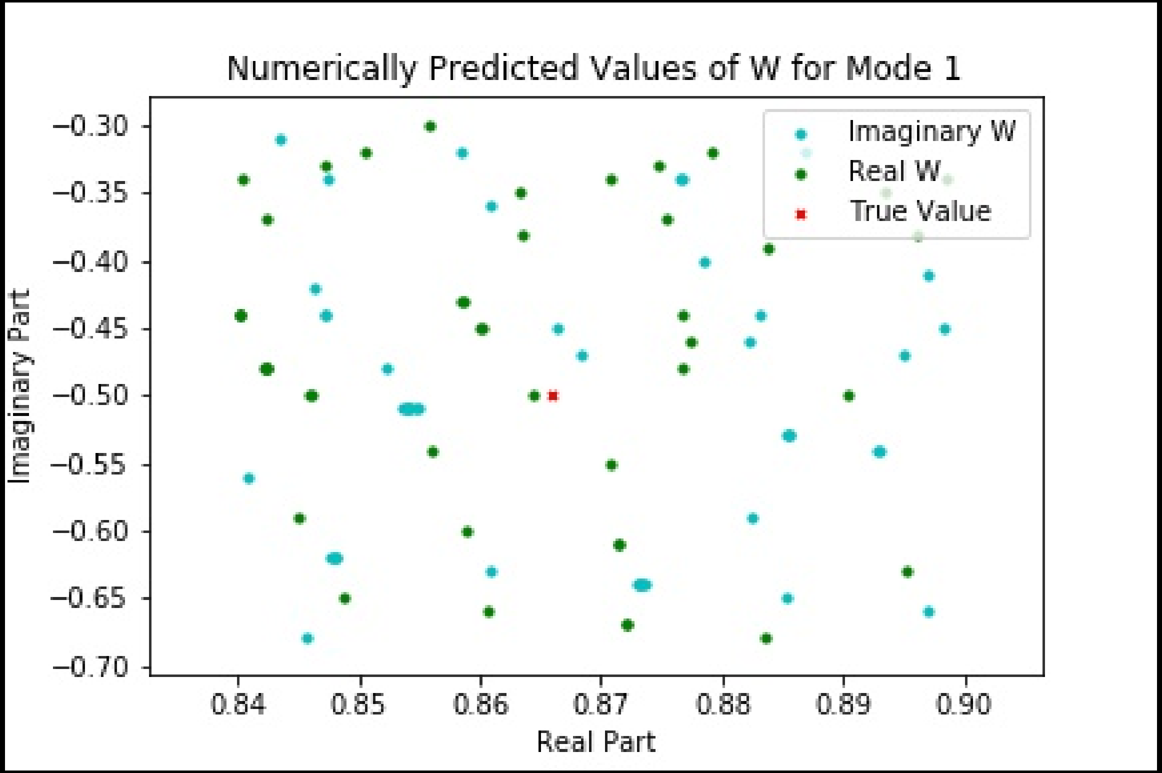
PROGRAM

- For the numerical method that is used to solve for $a'(x)$ the Runge-Kutta-4 method was chosen for its reliability and lower error over other methods
- Since the ω values can be complex, the program will choose test values out of a complex disk
- The Poschl-Teller potential is multi-moded, which each higher mode being increasingly hard to compute
  - Because of computational constrictions, only the first mode will be considered here

PROGRAM METHOD

- Since the real and imaginary parts of the ω seemed to be approaching the true value at different rates, the decision was made to record all of the values of ω that either satisfied low real error or low imaginary error
- After gathering these “good” values, they could be graphed against their error
- The “best” ω value, with the lowest error, from the real prediction and imaginary prediction could then be averaged into the best guess of the true eigenvalue
  - In theory, they would both produce the identical ω

GRAPH FOR MODE 1



- These are all values that passed an “error filter” of sorts
- The lower the relative error, the closer the points lie to the center of the graph
- As one can see, two points (one real and one imaginary) predicted values very close to the true solution

DATA FOR MODE 1

Real Prediction	0.8643 – 0.4999i
Imaginary Prediction	0.8683 – 0.4699i
Averaged Prediction	0.8663 – 0.4849i
True Eigenvalue	0.8660 – 0.5000i
Error	0.0003 + 0.0151i

- Thus, this method seems to nearly find the real part of the true value, but has higher error in the imaginary part
- To increase accuracy, a smaller step size could be used in order to sample a larger range of more accurate ω values
- However, more samples takes exponentially more time

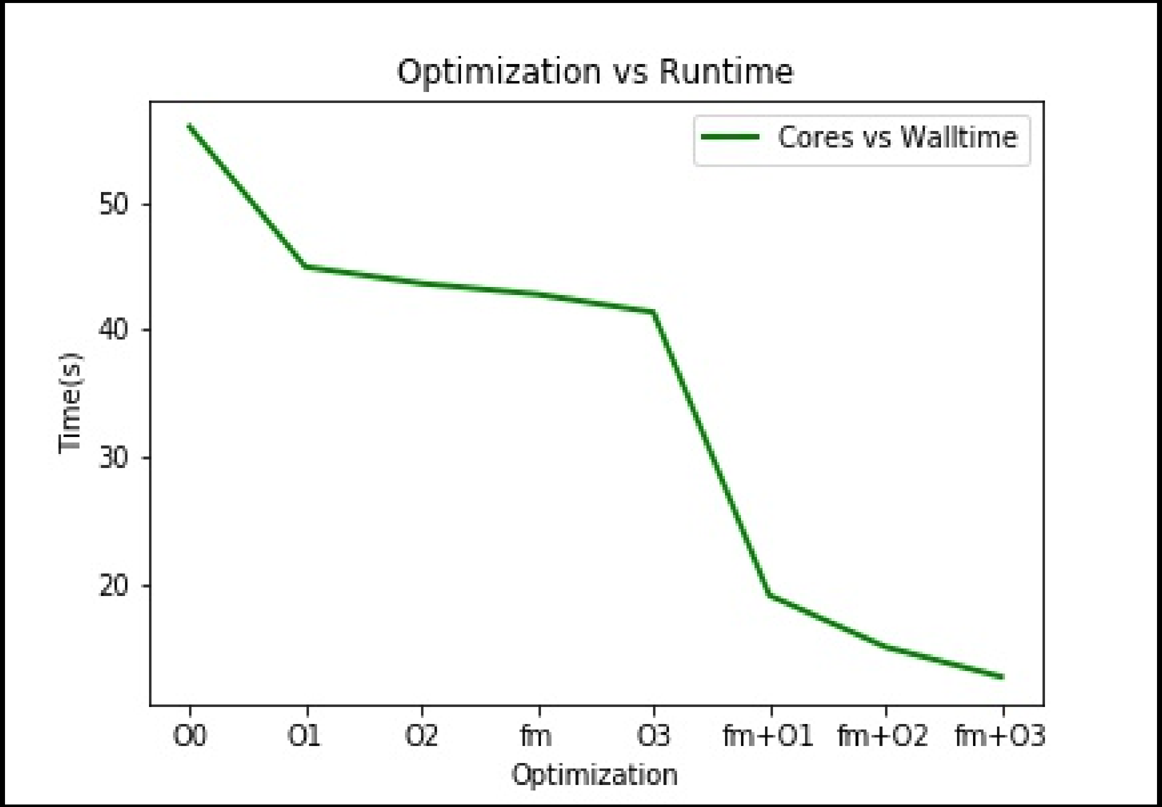
INCREASED EFFICIENCY

- In order to allow for this higher degree of accuracy, I first began by testing various command optimizations
- Since the numbers being dealt with are very small, gcc's –ffast-math can be used to simplify a lot of the mathematical calculations with little effect on the output
  - For numbers this small, the error created by this optimization would be on the order of 10^{-10}

INCREASED EFFICIENCY

- There is also the gcc –O optimization
- These -O0, -O1, -O2 and -O3 flags slightly increase memory usage and compiling time, but increasingly reduce execution time
    - For a medium sized program, the compiling time will not cause a noticeable difference

INCREASED EFFICIENCY

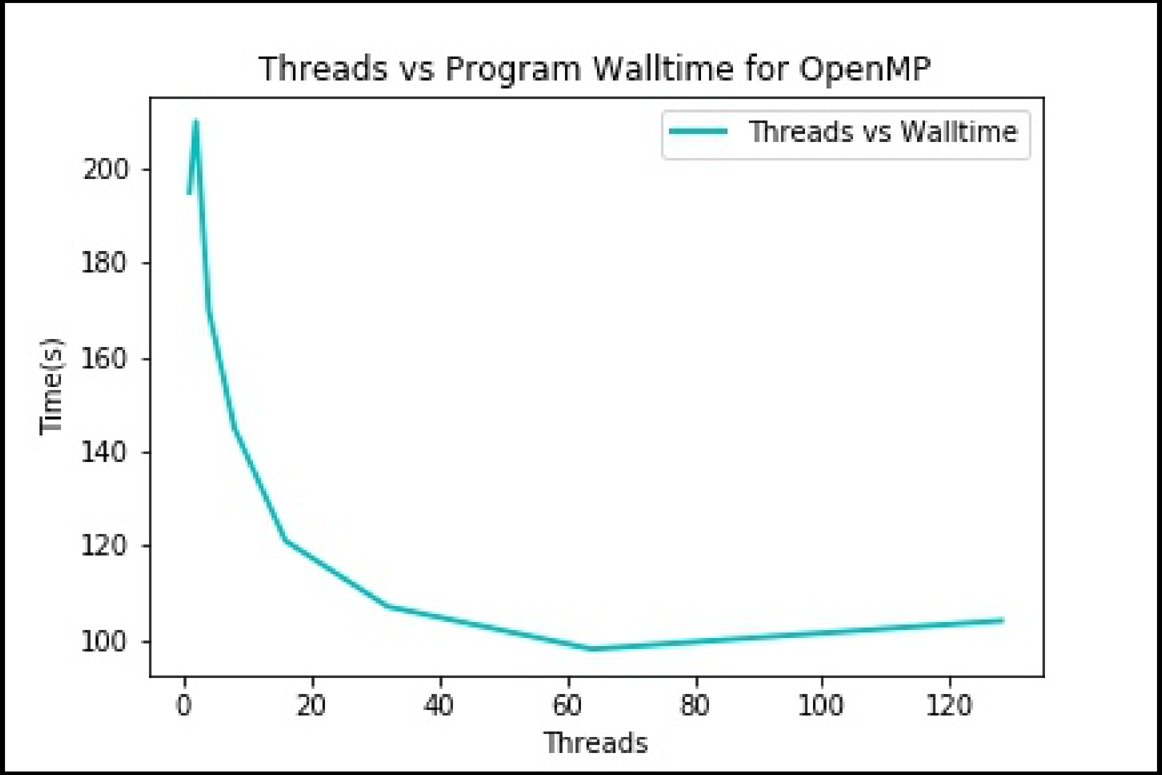


- This graph shows the change in execution time for each combination of the --ffast-math flag and -OX flags
- One can see that --fast-math and -O3 create the most efficient code
- Using these flags reduced the execution time by 440%

PARALLEL COMPUTING

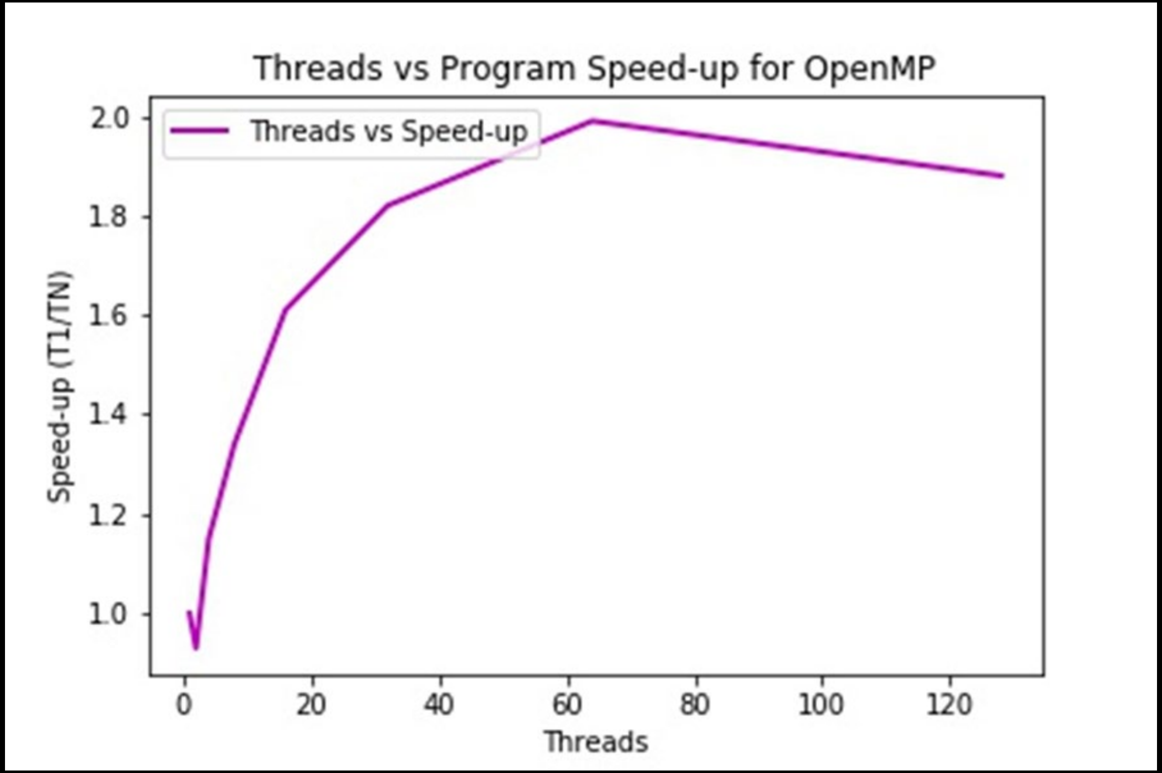
- Since the program exists in the form of a loop, making this loop a parallel region will allow for many value of ω to be sampled at the same time
- For this, I have used Stampede2, a supercomputer based in Texas and OpenMP (a parallel computing package)
- By allowing multiple computing nodes to access my program, Stampede can immediately outpace my laptop that has only one open node for computation

PARALLEL COMPUTING



- As one can see, the availably of more nodes greatly reduced the wall time for the program
- At around 64 nodes, the program's wall time begins to level off, indicting that there is little reason to allow more nodes to access the program
  - This would prove more costly and be of inconsequent gain

PARALLEL COMPUTING



- Speed-up time shows the relative difference in wall time when compared to the serial version (1 node)
- As mentioned in the last slide, it seems that at 64 nodes, the program has its largest speed-up value
- This would be the optimum amount of nodes to use for this program

FUTURE IMPROVEMENT

- One way to make this program more accurate is to use a parallel-shooting method instead of the standard method
- This method decreases propagation error by matching points inside and at the boundaries of the function
    - Perhaps another value(s) can be found in order to employ such a method, especially when moving to a more complicated potential

CONCLUSION

- The “shooting” method, with an RK4 scheme, was shown to be able to find the eigenvalues of a GW-like wave function
- With command-line optimizations, the program was able to execute 440% faster than the original version with a negligible change to the output
- By making the code work in parallel, the program was able to execute even faster, making it possible to use a smaller step size and more accurate numbers in the numerical approximations

NEXT STEPS

- One can now explore the higher modes of the Poschl-Teller potential because of the more efficient program

- Then, a true black hole potential can be introduced and one can see how well the program can handle a simple, real-world case

- Finally, one should be able to use any arbitrary black hole potential and obtain the eigenvalue frequency of its QNR