Competency	Foundational	
	Beginning skills that all students should master in the first few weeks of class.	
Computer programming	Create, compile, and interpret simple programs using basic mathematics, input and output, loops, and conditionals.	
Use of Linux/Unix	Use ssh to connect to a remote computer; use a text editor to create and edit files; manipulate files using ls, cp, mv, rm, etc.	
Algorithm analysis	Describe the difference between first-order, second-order, etc., algorithms, and recognize basic examples of these in our projects.	
Data analysis	Generate and graph data using plot	
Physical modeling	Describe the set of differential equations that describes a simple physical system with one degree of freedom	

Interpretation of simulation results	Compare the results of a numerical calculation to the analytic solution and describe, in basic terms, the validity or lack thereof of a calculation	
Technical writing and presentation	Write a basic report that describes your work on a simple project.	
Perturbative reasoning	Understand how a power series can be used to approximate the behavior of a simple function in a certain limit. Make meaning out of the different terms in power series. Create and interpret log/log plots to examine power-law scaling.	

Developing		Proficient	
The core skills that you should acquire in the first two months of class.		*Most students should achieve this level of mastery by the end of the course.*	
Use functions to modularize code; consistently write code that is clearly readable and that is written in a way that makes it flexible and modifiable.		Use arrays to deal with large amounts of data; write well-organized programs to solve substantial computational projects.	
Perform intermediate Linux tasks, such as organizing files into directories. Understand the use of <, >, and | operators to work with files on disk.		Use an advanced text editor to edit code. Demonstrate proficiency in assorted system tasks that arise in the course of the class. Demonstrate an understanding of the Unix filesystem structure.	
Determine from scaling data the order of a numerical integration algorithm. Show from the graphical picture of the Riemann sum the scaling order of basic integration routines.		Demonstrate an understanding of the computational consequences of algorithm properties, including the extraction of physical observables from simulation behavior (interpolation of endpoints, etc.) Describe the differences between the Euler, Euler-Cromer, and leapfrog algorithms.	
Collect appropriate and reasonable data to answer simple computational or physical problems, present them in appropriate ways, and draw substantive conclusions from these data. Describe clearly how these conclusions follow from the presentation of the data (graphical, numeric, or otherwise).		Generate appropriate simulation data to answer substantive physical questions that depend on multiple variables and that may depend on multiple runs of a program with varying parameters. Draw valid physical conclusions and meaningful physical insight from these data.	
For a system of a few degrees of freedom, construct and describe a set of coupled ordinary differential equations that models that system. Describe the assumptions and estimates made in a mathematical model of a physical system. Argue for its validity and discuss any limits to that validity. Appropriately use natural units in simulating systems.		Implement a physical model of a more complex physical system involving extended objects or regions. Discuss the consequences of spatial discretization on the performance, behavior, and validity of simulations. Demonstrate the ability to apply such a model so that its physical interpretation is independent of the choices of discretization parameters, except for simulation artifacts.	

Test codes to determine whether an algorithm is performing as expected. Demonstrate an understanding of the approximate nature of numerical calculations and the origins of various sources of error. Evaluate the level of physical validity of numerical calculations. Draw meaningful physics conclusions from simple simulations.		Use numerical simulations to discover novel physical phenomena that are not readily apparent to pencil-and-paper physics. Conduct a series of calculations with different parameters to carefully study whether a simulation is performing as expected, and to study ways in which a simulation does not capture the physical behavior of the system. Convincingly argue for the validity of a given simulation and any limits on that validity. Be able to distinguish between modeling artifacts, numerical artifacts, and real physical phenomena in computational results.	
Write a more detailed report describing the objectives of a computation, the model used and its motivation, the algorithms used, the data obtained, and its interpretation. Discuss challenges and approaches fluently in technically-correct language.		Write a substantial report describing an in-depth exploration of a physical system whose behavior is not known a priori. Write a coherent, cohesive narrative describing the interaction between your system, how you modeled it, the algorithm you used to solve the model (including choice of stepsizes, etc.), your data, and how you interpret those data to learn something about physics you didn't know before. This report should be comparable in quality to a paper that you might submit as a term paper in a writing class.	
Demonstrate an understanding of the use of power series to estimate the error in a numerical integration algorithm. Demonstrate basic perturbative reasoning regarding physical systems (behavior in ideal regime and departure from it). Discuss numerical calculation results in the context of perturbative behavior for simple physical systems.		Evaluate the perturbative behavior of the numerical simulation of a moderately-complex system as the perturbing factor is turned on. Relate the perturbative behavior of such a system to the assumptions made in deriving the unperturbed solution. Discuss algorithm performance and numerical and modeling artifacts with a unified vocabulary of scaling order and power counting.	

Advanced		Exceptional
Beyond expectations: strong students will make it here in a few categories with their advanced projects.		*Work that I and most of my colleagues can learn from, and that represents a contribution beyond the scope of this class (you might give a seminar or publish something!)*
Demonstrate advanced programming techniques, such as the use of structs, recursion, external libraries, Makefiles, programs that usefully make use of multiple files, or the like.		Demonstrate high-end programming techniques, such as: * Parallelism using OpenMPI or CUDA * Complex data structures and dynamically-allocated memory * Use of assembly language, SSE, or other techniques for hyperoptimizing code * Use of the Condor cluster for computationally-expensive work
Administer a Unix machine or virtual machine of your own. (Any serious science, mathematics, or engineering student really needs one.) Demonstrate an understanding of the package manager on your system (Macports, yum, apt, etc.) Teach your professor how the filesystem on OSX works (if you have a Mac).		Find something that bothers you about a piece of open-source software that you use. Download the source code and change it for the better, then suggest the change to the authors.
Make use of an advanced algorithm (RK4, Forest-Ruth, Omelyan, etc.) to solve a physical problem. Analyze the consequences of your choice of this algorithm: what are its impacts on the validity of your simulation, or the computational costs incurred?		Devise and implement a highly-complex algorithm to solve an otherwise-intractable problem.
Analyze complex data sets and draw novel physical conclusions from them. This may involve statistical analysis, Fourier analysis, substantial data reduction, or presentation of data in complex formats (3D plots, etc.); the details will depend on your choice of advanced project.		An extension of the preceding, using even more advanced analysis techniques, such as those that are often required in professional/dissertation-level research.
Construct a physical model for a substantial physical system and implement it. Discuss the choices and assumptions in this model, and its implications for a computational simulation. The details will depend on your choice of advanced project.		An extension of the preceding, where constructing a model that captures the essential physics of the system but is numerically tractable requires a nuanced analysis of the physical factors at work and what approximations may safely be made.

Design and conduct a series of simulations of a complex system that is beyond the reach of your pen-and-paper analysis to determine its physical properties. These simulations may be limited by available computer power, forcing you to deal with noisy data or balance difficult tradeoffs between numerical artifacts, simulation artifacts, stochastic noise, and data quantity, or to use difficult data analysis techniques. Do this convincingly, distinguishing between true behavior, modeling artifacts, numerical artifacts, and other sources of error.		An extension of the preceding, but with even more sophisticated interpretation and analysis required, possibly in a situation where there is no clear understanding of what the physical behavior of the system is.
Write a detailed, nuanced report using Markdown or LaTeX describing a complex project that is of near-publishable quality; the standard here is that a PhD in physics would find it interesting to read and would learn something new from it. This report should be well-organized and present theory, methods, simulation details, results, conclusions, etc., in a coherent, physically-literate narrative.		Present results from a computational project in a seminar setting to members of the Syracuse physics department. Teach us all something you just learned that we don't know!
Present a mathematical proof that Simpson's rule is a fourth-order integration scheme, or complete the previous standard with a substantially more complex system (in your advanced project).		Perform an analytic perturbative expansion of a complex physical situation or algorithm (the nonlinear vibrating string, the Omelyan integrator, or something of that nature) and compare the results to numerical data.