

**Ralph Regula School of Computational Science
Cyberinfrastructure Team Project
Summary of Undergraduate Minor Program Requirements
September 2006**

The minor in computational science will be offered by a number of higher education institutions in Ohio. Degrees will be offered by the students "home" institution but it will be possible for the student to take minor program materials from other, participating institutions. We have started with a minor because we believe that each student needs some domain expertise in a major field before being able to complete computationally-based projects in related areas.

There is a committee reviewing the various options for the sharing of materials, instruction, and revenues associated with cross-registered students. They will make some recommendations to the participating institutions in the coming months on the institutional procedures.

Because of the inter-institutional nature of the program, we have designed it to be competency-based. The instructional materials associated with each competency or subset of competencies can then be embedded either in existing courses on each campus, in new courses at one or more campuses, or as stand-alone modules that might be taken at distance and at the time that each student is ready for them. Competency will then be demonstrated through one or more assessments of the student's abilities on a combination of exams and projects. The competency-based approach is preferred because it gives curriculum flexibility to participating programs and gives employers some assurance of the working knowledge of each graduate.

The draft competencies created by the participating faculty has been reviewed by a business advisory committee. They confirmed the majority of the recommended competencies at a meeting on May 31, 2006 and offered some advice on topic emphasis and breadth. This document reflects their comments and is the basis on which faculty are currently proposing instructional modules that meet the competencies.

The computational science minor will enable science and engineering majors to apply computational tools to the problems in their discipline. As such, the minor breaks into three broad categories: Prerequisites, the Computational Science Core Competencies and the Discipline Specific Competencies. At this stage, we are focused on the competencies that comprise the Computational Science Core. The goal of the Core competencies is to establish a foundation that can be leveraged in the Discipline Specific Competencies that are directly applicable to the practice of the student's chosen profession. The competencies are shown in Table 1.

Our remaining decisions involve deciding what competencies are associated with each broad area, how many elective competencies are required for the minor, and the specific competencies for each discipline-oriented course. Those will be addressed in Autumn 2006 as we begin to pre-test the materials in classrooms around Ohio.

Table 1: Competencies and Requirements for Undergraduate Computational Science Minor		
Topic	Subtopics	Required/Optional
Prerequisites		
Calculus 1 and 2		Required
Computational Science Courses		
Simulation and Modeling	Should use one of the major symbolic programs Maple, MATLAB, Mathematica; need to local, hands-on support at outset of course	Required
Programming and Algorithms	Logic of programming whether a traditional computer science or programming with a symbolic language like Maple, MATLAB, Mathematica	Required
Differential Equations and Discrete Dynamical Systems	Depending upon major; linear algebra may also be needed by some	Elective
Numerical Methods	Need for a project-based course which touches the most important topics rather than a standard math course	Required
Optimization	An important topic; could be part of a modeling course or integrated with numerical methods	Required
Parallel Programming		Elective
Scientific Visualization		Elective
Discipline Specific Courses		
Capstone Research/Internship Experience		Required
Discipline Oriented Courses		One required; probably only one per field for awhile

**Minor Program in Computational Science
Competency/Topic Overview
Area 1: Simulation and Modeling**

Competency/Descriptors
<p>Explain the role of modeling in science and engineering</p> <p>Descriptors:</p> <p>Discuss the importance of modeling to science and engineering</p> <p>Discuss the history and need for modeling</p> <p>Discuss the cost effectiveness of modeling</p> <p>Discuss the time-effect of modeling (e.g. the ability to predict the weather)</p> <p>Define the terms associated with modeling to science and engineering</p> <p>List questions that would check/validate model results</p> <p>Describe future trends and issues in science and engineering</p> <p>Identify specific industry related examples of modeling in engineering (e.g., Battelle; P&G, material science, manufacturing, bioscience, etc.)</p> <p>Discuss application across various industries (e.g., economics, health, etc.)</p>
<p>Analyze modeling and simulation in computational science</p> <p>Descriptors:</p> <p>Identify different types of models and simulations</p> <p>Describe a model in terms of iterative process, linking physical and virtual worlds and the science of prediction</p> <p>Explain the use of models and simulation in hypothesis testing (e.g. scientific method)</p>
<p>Create a conceptual model</p> <p>Descriptors:</p> <p>Illustrate a conceptual modeling process through examples</p> <p>Identify the key parameters of the model</p> <p>Estimate model outcomes</p> <p>Utilize modeling software and/or spreadsheets to implement model algebraic equations (e.g. Vensim, Excel, MATLAB, Mathematica)</p> <p>Construct a simple computer visualization of the model results (e.g. infectious disease model, traffic flow, etc.)</p> <p>Validate the model with data</p> <p>Discuss model quality and the sources of errors</p>
<p>Examine various mathematical representations of functions</p> <p>Descriptors:</p> <p>Describe linear functions</p> <p>Define non-linear functions (e.g., polynomials, exponential, periodic, parameterized, etc.)</p> <p>Visualize functions utilizing software (e.g. Excel, Function flyer, etc.)</p> <p>Determine appropriate functional form to fit the data</p> <p>Demonstrate essential mathematical concepts related to modeling and simulation</p>
<p>Analyze issues in accuracy and precision</p> <p>Descriptors:</p> <p>Describe various types of numerical and experimental errors</p> <p>Explain the concept of systematic errors</p>

<p>Explain the concept of data dependent errors Illustrate calculation and measurement accuracy Identify sources of errors in modeling and approaches to checking whether model results are reasonable</p>
<p>Understand discrete and difference-based computer models Descriptors: Explain the transition of a continuous function to its discrete computer representation Represent “rate of change” using finite differences Cite examples of finite differences Explain derivatives and how they relate to model implementation on a computer Write pseudo-code for finite difference modeling</p>
<p>Demonstrate computational programming utilizing a higher level language or modeling tool (e.g. Maple, MATLABTM, Mathematica, other) Descriptors: Describe the system syntax (e.g., menus, toolbars, etc.) Define elementary representations, functions, matrices – arrays, script files, etc. Explain programming and scripting processes (e.g., relational operations, logical operations, condition statements, loops, debugging programs, etc.) Create tabular and visual outputs (e.g., 2-D and 3-D subplots) Translate the conceptual models to run with this system and assess the model results (e.g. traffic flow and/or “spread of infectious disease”) Illustrate other people’s models utilizing the modeling program</p>
<p>Assess computational models Descriptors: Assess problems with algorithms and computer accuracy Discuss techniques and standards for reviewing models Verify and validate the model Discuss the differences between the predicted outcomes of the model and the computed outcomes and relevance to the problem Discuss the suitability and limits of the model to address the problem for which the model was designed</p>
<p>Build event-based models Descriptors: Describe event-based modeling (e.g. SIMULINKTM; Extend, ARENA) Run existing models Translate conceptual models (e.g., traffic flow utilizing SIMULINKTM)</p>
<p>Complete a team-based, real-world model project Descriptors: Identify a problem, create mathematical model and translate to computational modeling Organize and present project proposal Document model development and implementation Collaborate with team members to complete the project</p>
<p>Demonstrate technical communication Descriptors: Demonstrate technical writing skills in the comprehensive report</p>

Demonstrate verbal communication skills in an oral presentation
Create and present visual representation of model and results
Address all components of a comprehensive technical report
Respond to peer review

**Minor Program in Computational Science
Competency/Topic Overview
Area 2: Programming and Algorithms**

Competency/Descriptors
<p>Describe the fundamentals of problem solving</p> <p>Descriptors: Understand Top-Down thinking and program design Discuss breaking up a problem into its component tasks Understand how tasks acquire data Describe how tasks should be ordered Represent tasks in a flow-chart style format Understand the difference between high-level languages (for example Mathematica, Maple or MATLAB), medium level languages (for example FORTRAN or C) and low-level languages (assembler) and when each should be used.</p>
<p>Understand and write Pseudo code</p> <p>Descriptors: List the basic programming elements of Pseudo code Explain the logic behind an if/then/else statement Understand the iterative behavior of loops Describe the difference between several looping constructs Write Pseudo code to solve basic problems Understand how to represent data flow in and out of subprograms.</p>
<p>Use subprograms in program design</p> <p>Descriptors: Describe how logical tasks can be implemented as subprograms Understand the logical distinction between functions and subroutines Explain the control flow when a function is called Define dummy and actual arguments Discuss the different relationships dummy and actual arguments Explain how function output is used Understand how languages handle passed data into functions and subprograms, especially one and two dimensional arrays.</p>
<p>Write code in a Programming language</p> <p>Descriptors: Understand the concept of syntax in a programming language Describe the syntax of the programming language constructs List the type of subprograms available in the language Explain the concepts of argument pass-by-value and pass-by-reference Understand what a compiler and linker do Understand the difference between a compiled and interpreted language Understand the difference between a typed and an un-typed language Understand the difference between a source file and an executable file Write and run basic programs in the language of choice Understand how to de-bug code and how to "sanity check" code. Understand the importance of user-interfaces: clear input instructions including physical</p>

units if needed and clearly formatted and labeled output
Understand the numerical limits of various data types and the implications for numerical accuracy of results.

Use different approaches to data I/O in a program

Descriptors:

Explain the advantages and disadvantages of file I/O
Describe the syntax for file I/O in your programming language
Compare binary and ASCII file I/O
Write code using file I/O and keyboard/monitor I/O

Understanding and use of fundamental programming Algorithms

Descriptors:

Explain an algorithm as an ordered series of solution steps
Describe an algorithm for a simple programming problem
Learn and use “classic” programming algorithms from a field of interest to the student.
If possible, these should be algorithms used in the student’s discipline.
Describe what a software library is
Understand how library functions implement algorithms
Write code to implement your own version of “classic” algorithm
Compare with code using a library function
Understand data flow into library functions and implications of selecting any “tuning parameters” or options that may be required.

Explain various approaches to Program Design

Descriptors:

Describe Functional decomposition (Top-down Problem Solving)
Be familiar with different programming styles (e.g. function, procedural, rule based)
Understand how to modularize code
Understand the benefits of code re-use
Explain the operation of a Boss-Worker design
Compare designs based on Global Variables vs. self-contained functions
Define Object-Oriented Programming (OOP)
Contrast OOP with functional decomposition
Explain the power of Inheritance in OOP
Understand how to document code
Understand how to write and when to use stubs and drivers.

**Minor Program in Computational Science
Competency/Topic Overview
Area 3: Differential Equations and Discrete Dynamical Systems**

Competency/Descriptors
<p>Describe the solution methodology for first order linear differential and difference equations</p> <p>Descriptors: Analyze modeling problems with first order differential equations and present their solution methodology (e.g. liner, homogeneous, exact) Analyze modeling problems with first order difference equations and present their solution methodology (e.g. homogeneous, non-homogeneous). Analyze long term behavior</p>
<p>Describe the solution methodology for systems of linear first order differential and difference equations</p> <p>Descriptors: Describe modeling problems with systems of first order differential equations and present their solution methodology (e.g., homogeneous with constant coefficients, variation of parameters) Describe modeling problems with systems of first order difference equations and their solution methodology (e.g., homogeneous with constant coefficients)</p>
<p>Describe the solution methodology for higher order differential and difference equations</p> <p>Descriptors: Describe modeling problems with higher order differential equations analyze their solution methodology (e.g., homogeneous, non-homogeneous, undetermined coefficients, variation of parameters) Describe modeling problems with higher order difference equations analyze their solution methodology (e.g., homogeneous, non-homogeneous). Analyze the long-term behavior.</p>
<p>Describe the solution methodology for differential equations using the Laplace Transforms</p> <p>Descriptors: Discuss the Laplace transformation of (e.g., continuous , discontinuous, delta and convolution) functions Describe modeling problems with differential equations and present their solution methodology using Laplace transformations (use of CAS, Maple, Mathematica)</p>
<p>Describe the solution methodology for non-linear differential equations</p> <p>Descriptors: Describe the concept of an equilibrium point Model with non-linear differential equations and present the phase –portrait analysis Understand and demonstrate how chaos is generated in the solution process of non-</p>

linear differential equations.

Describe the solution methodology for non-linear difference equations

Descriptors:

Describe the method of linearization

Describe the concepts of Logistic and Henon Maps

Model with non-linear difference equations and demonstrate understanding of fundamental concepts from Bifurcation theory (e.g., fixed, periodic points, chaos)

Describe techniques for controlling chaos

Understand concepts of numerical accuracy applied to each solution approach

**Minor Program in Computational Science
Competency/Topic Overview
Area 4: Numerical Methods**

Competency/Descriptors
<p>Understand number representation and computer errors</p> <p>Descriptors: Understand the pros and cons of floating point and integer arithmetic Describe various kinds of computing errors (e.g., round-off, chopping) Describe absolute, relative error and percent error Discuss error propagation Describe loss of significance – methods to avoid loss of significance</p>
<p>Analyze methods for solving non-linear equations</p> <p>Descriptors: Discuss and contrast fixed point methods (e.g., bisection, secant, Newton’s) for a single equation Describe a fixed point method for a system of equations (e.g., Newton’s)</p>
<p>Describe techniques for solving systems of linear equations</p> <p>Descriptors: Describe the naïve Gauss elimination and the partial pivoting method Understand the concepts of condition number and ill-conditioning problems Discuss and contrast factorization methods (e.g., LU, QR, Cholesky, SVD) Discuss and contrast iterative methods (e.g., Jacobi, Gauss Siedel) Describe convergence and stopping criteria of iterative methods</p>
<p>Analyze techniques for computing eigenvalues—eigenvectors (Optional)</p> <p>Descriptors: Describe and give examples of eigenvalue –eigenvector problems using specific, applied examples and their significance Describe canonical forms of matrices Describe and contrast direct methods for computing eigenvalues (e.g., power method, inverse power method) Describe and contrast transformation methods (e.g., QR algorithm)</p>
<p>Describe interpolation and approximation methods</p> <p>Descriptors: Describe and contrast interpolation methods (e.g., Lagrange, Chebyshev, FFT) Describe interpolation with spline functions (e.g., piecewise linear, quadratic, natural cubic) Discuss approximation using the method of least squares (linear .vs. non-linear)</p>
<p>Describe numerical methods for Ordinary Differential Equations</p> <p>Descriptors: Describe and compare basic methods for IVPs (e.g., Euler, Taylor, Runge-Kutta) Describe and compare predictor-corrector methods Describe and compare multistep methods Discuss and contrast numerical methods for BVPs (e.g., shooting method, finite difference method) Compare the accuracy, memory requirements, and precision of each of the approaches</p>

Describe numerical methods for Partial Differential Equations

Descriptors:

Describe and compare numerical methods for parabolic PDEs (e.g., finite difference, Crank-Nicolson)

Describe numerical methods for hyperbolic PDEs

Describe numerical methods for elliptic PDEs (e.g. finite difference, Gauss-Seidel)

Discuss the finite element method for solving PDEs

Describe Monte Carlo Methods

Describe applications of Monte Carlo models with examples

Discuss algorithms for Monte Carlo methods

**Minor Program in Computational Science
Competency/Topic Overview
Area 5: Optimization**

Describe and use Optimization techniques

Descriptors:

Describe and contrast unconstrained optimization methods (e.g., Golden section search, Steepest descent, Newton's method, conjugate gradient, simulated annealing, genetic algorithms)

Describe and contrast constrained optimization methods (e.g., Lagrange multiplier, quasi-Newton, penalty function method)

Implement linear and non-linear programs

Analyze linear programming methods (e.g., simplex method)

Describe non-linear programming methods (e.g., interior, exterior, mixed methods)

Demonstrate ability to correctly use software systems (e.g., Matlab, IMSL, NAG) to solve practical optimization problems

Minor Program in Computational Science
Competency/Topic Overview
Area 6: Parallel Programming

Competency/Descriptors
<p>Describe the fundamental concepts of parallel programming and related architectures</p> <p>Descriptors: Describe the differences between distributed and shared memory architectures Describe the difference between domain and functional decomposition in parallel Describe a parallel programming approach to an introductory problem Compare parallel, distributed, and grid computing concepts</p>
<p>Demonstrate parallel programming concepts using MPI</p> <p>Descriptors: Describe the MPI programming model Create, compile, and run an MPI parallel program Create MPI programs that utilize point-to-point communications Create an MPI program that uses point-to-point blocking communications Create an MPI program that uses point-to-point non-blocking communications Create an MPI program that uses collective communications Create an MPI programs that use parallel I/O Create MPI programs that use derived data types Create MPI programs that use vector derived data type Create MPI programs that use structure derived data type</p>
<p>Demonstrate knowledge of parallel scalability</p> <p>Descriptors: Use mathematical formulas to determine speed-up and efficiency metrics for a parallel algorithm. Demonstrate the use of graphical systems such as MATLAB to display speed-up and efficiency graphs</p>
<p>Demonstrate knowledge of parallel programming libraries and tools</p> <p>Descriptors: Demonstrate the use of performance tools for profiling programs (e.g., GNU GPROF or MATLAB profiler) Create parallel programs with calls to parallel libraries (e.g. BLAS, BLACS, ScaLAPACK or FFTW) Demonstrate the use of MPI tracing tools (e.g., VAMPIR) to determine parallel performance bottlenecks</p>

**Minor Program in Computational Science
Competency/Topic Overview
Area 7: Scientific Visualization**

Competency/Descriptors
<p>Define SciVis needs; relationships to human visualization; basic techniques Define Scientific Visualization (Sci Vis) Discuss needs of SciVis (in the framework of a large variety of possible application areas) Survey different platforms for Visualization (e.g. AVS, VTK, OpenGL, VRLM) Discuss the different techniques and visualization methods used in SciVis Explain the human visualization system – capabilities and perceptions Explain the different steps in the visualization pipeline Discuss different sources of data for SciVis and explain the terms applied to data types (i.e. scalar, vector, normal, tensor) Discuss different types of grids (e.g., regular vs. irregular grids) Discuss the different methods used to gather data Describe and explore the use of different file formats for sharing data (netCDF, XML, TIFF, GIF, JPEG, Wavefront OBJ) Discuss limitations of different methods Discuss future applications in emerging fields Metadata needs for graphics libraries</p>
<p>Overview of computer graphic concepts Descriptors: Overview of SciVis concepts (pixels, rgb colors, 3D coordinate system, mapping 3D data to a 2Dscreen, continuous vs. discrete) Discuss polygonal representation Discuss lighting/shading Overview of classification/segmentation and transfer functions Discuss concept of rendering pipeline (no details about matrices) Discuss hardware rendering (mainly for polygonal models, few specialized volumetric hardware cards) Identify terms used in virtual space and in graphics elements Navigate in virtual space and manipulate primitive objects</p> <ul style="list-style-type: none"> ▪ Transform: scale, rotate, translate) ▪ Manipulate surface ▪ Manipulate lighting and camera <p>Explore colormaps and examine conceptual definitions for different color maps (pertaining to color spaces HSV, RGB, etc.) as related to representing data and relationships to perception</p>
<p>Describe approaches to visualization for different scientific problems Descriptors: Examine different computational solutions to scientific problems Explain the different techniques used in visualization (i.e. glyphs, iso-contours, streamlines, image processing, volume-data) Examine the application of problems to visualization techniques</p>

Utilize software tools to implement visual image of a solution
Discuss the use of time in animation

Utilize software to implement grid representations of data

Descriptors:

Identify the various cell representations (i.e. points, polygons, 3d geometries)

Discuss the application to different grid types (i.e. structured, unstructured, random)

Discuss raycasting methods and texture mapping

Examine the details of raycasting sampling (FAT(low resolution sampling), interpolation techniques).

Examine algorithms: Direct Composite, SFP, use of transparency.

Identify the grid representation and data(color reps.) (regular grids, 1, 8, 24 and 32 bit color information)

Discuss algorithms for manipulating images, distortion, fft's, enhancement, restoration, frequency domains

Utilize software to implement different grid types

Discuss limitations of grids

<p>Use visualization software to display an isosurface</p> <p>Descriptors: Discuss different data types used: scalar vs. vector data Discuss the different grid types Discuss the different algorithms (Marching Cubes etc) Introduce details of the system being used in a course (e.g., VTK, AVS, etc.) Apply the system to extract and display an isosurface of some data set (could be tailored towards the teacher's and student's interests/application areas) Discuss limitations of these methods</p>
<p>Use visualization software to complete a volumetric rendering</p> <p>Descriptors: Discuss direct volumetric rendering (raycasting and texture mapping) and its advantages/disadvantages vs. surface rendering Discuss segmentation/classification and transfer functions Discuss and illustrate how to use a system (VTK, AVS, etc.) to do volumetric rendering Using the system, visualize a data set using raycasting Using the system, visualize a data set using texture mapping Discuss limitations of this method</p>
<p>Utilize visualization software to visualize a vector dataset</p> <p>Descriptors: Discuss vector data Discuss different methods for vector visualizing (particles, stream ribbons, vector glyphs, etc.) Discuss the use of structured grid types: (ir)regular, cylindrical, spherical Discuss application areas for vector visualization (air flow, etc.) Using a system (VTK, AVS, etc.), visualize a vector data set Discuss limitations of this method</p>
<p>Explore examples of image processing</p> <p>Descriptors: Discuss basic steps and goals in image processing Discuss variety of data sources of images and how they can be represented Discuss algorithms used for image processing Explore examples of image processing (e.g., noise reduction, image enhancement, feature extraction etc) Discuss challenges and limitations in image processing</p>
<p>Use advanced techniques applied to a real problem</p> <p>Descriptors: To be chosen by instructor based on instructor/student interest. Among suggested topics are:</p> <ul style="list-style-type: none"> - visualizing an irregular grid; - visualizing a data set specific to the area of interest (see 3.2 and 3.3 for specific examples) - writing a segmentation tool - implementing a visualization algorithm from scratch (such as marching cubes or raycasting)
<p>Examine SciVis problems for Biological Sciences – "OMICS" applications</p>

Descriptors:

Examine different problems existing in 'OMICS sciences that require visualization solution (overview)

Discuss challenges of representing biomedical/biological data (i.e., representing protein structure or genomic sequence with all their attributes as a visual metaphor)

Discuss challenges associated with visualization of scattered data such as text information and bioinformatics data (e.g., phylogenetic information)

Gene finding in genomic sequences

- Examine different components of a gene structure
- Visualize genomic structure of an individual gene
- Build a comparison between genomic features from several genomes
 - o Visualize (and examine) similarities and differences
 - o Discuss goal-dependent options of parsing the results to be explored elsewhere (e.g., as plain text, XML-marked)

Protein folding and protein structure prediction

- o Discuss the differences between protein folding and protein structure prediction
- o Explore different methods used in protein folding and structure prediction
- o Apply different methods of protein structure prediction and compare the results
- o Construct, visualize and examine structure-based protein alignment

Biological networks (e.g., protein-protein or protein-DNA interaction networks)

Visualization of various types of expression data

- o Discuss and contrast different types of expression data – e.g., microarray gene expression data, protein expression data
- o Discuss different visualization (and analyses) techniques used for expression data
- o Apply (and compare outcomes) hierarchical clustering and k-means clustering to the same gene microarray expression data
- o Discuss pros and cons of different clustering methods, their shortcomings, and ways to access the quality of clusters

Discuss potential applications of SciVis techniques in biomedical and drug design fields

Utilize MATLAB to implement/solve the above problems

Explore SciVis techniques in BioMedical applications**Descriptors:**

Explore a variety of biomedical applications of SciVis to explore large datasets such as MRI and confocal microscopy data

Overview of volume visualization techniques in biomedical problems

Examine and different ways MRI (Magnetic Resonance Imaging) data can be visualized (e.g., 2-D image versus isocontour slices).

Discuss potential applications (interpretation) of each of the techniques.

Utilize software tools (MATLAB, VTK) to apply the techniques above

