Fiscal Unit/Academic Org	ASC Administration - D4350
Administering College/Academic Group	Arts and Sciences
Co-adminstering College/Academic Group	
Semester Conversion Designation	New Program/Plan
Proposed Program/Plan Name	Computational Science Minor
Type of Program/Plan	Undergraduate minor
Program/Plan Code Abbreviation	
Proposed Degree Title	

Credit Hour Explanation

Program credit hour requirements		A) Number of credit hours in current program (Quarter credit hours)	B) Calculated result for 2/3rds of current (Semester credit hours)	C) Number of credit hours required for proposed program (Semester credit hours)	D) Change in credit hours
Total minimum credit hours required for completion of program				18	
Required credit hours offered by the unit	Minimum				
	Maximum				
Required credit hours offered outside of the unit	Minimum				
	Maximum				
Required prerequisite credit hours not included above	Minimum				
	Maximum				

Program Learning Goals

Note: these are required for all undergraduate degree programs and majors now, and will be required for all graduate and professional degree programs in 2012. Nonetheless, all programs are encouraged to complete these now.

Program Learning Goals

- Program learning goals will be submitted at a later date.

Assessment

Assessment plan includes student learning goals, how those goals are evaluated, and how the information collected is used to improve student learning. An assessment plan is required for undergraduate majors and degrees. Graduate and professional degree programs are encouraged to complete this now, but will not be required to do so until 2012.

Is this a degree program (undergraduate, graduate, or professional) or major proposal? No

Program Specializations/Sub-Plans

If you do not specify a program specialization/sub-plan it will be assumed you are submitting this program for all program specializations/sub-plans.

Pre-Major

Does this Program have a Pre-Major? No

Attachments

- Computational Science Minor Program Rev 6-27-13.pdf

(Program Proposal. Owner: Hanlin,Deborah Kay)

Comments

- Upload revised version *(by Vankeerbergen,Bernadette Chantal on 06/27/2013 09:46 AM)*

Workflow Information

Status	User(s)	Date/Time	Step
Submitted	Hanlin,Deborah Kay	06/12/2013 12:13 PM	Submitted for Approval
Revision Requested	Vankeerbergen,Bernadette Chantal	06/12/2013 03:41 PM	Unit Approval
Submitted	Hanlin,Deborah Kay	06/12/2013 03:55 PM	Submitted for Approval
Approved	Vankeerbergen,Bernadette Chantal	06/12/2013 04:21 PM	Unit Approval
Approved	Heysel,Garett Robert	06/13/2013 11:48 PM	College Approval
Revision Requested	Vankeerbergen,Bernadette Chantal	06/27/2013 09:46 AM	ASCCAO Approval
Submitted	Hanlin,Deborah Kay	06/27/2013 09:55 AM	Submitted for Approval
Approved	Vankeerbergen,Bernadette Chantal	06/27/2013 09:56 AM	Unit Approval
Approved	Carlson,Wayne Earl	06/28/2013 08:50 AM	College Approval
Pending Approval	Nolen,Dawn Jenkins,Mary Ellen Bigler Vankeerbergen,Bernadette Chantal Hogle,Danielle Nicole Hanlin,Deborah Kay	06/28/2013 08:51 AM	ASCCAO Approval

Proposal for an Arts and Sciences Minor Program in Computational Science

Introduction

Computational science describes the application of computing, especially high performance computing, to the solution of scientific and technical problems. Computational scientists use computers to create mathematical models that help them simulate and understand the operation of natural and mechanical processes, as well as to visualize the operation and results of these models.

Computational science (i.e., science in-silico) has become a third way of advancing knowledge along with the traditional methods of theory and experimentation. In-silico simulations and modeling afford the opportunity to "see" the unattainable – phenomena that are too small (atoms and molecules), too large (galaxies and the universe), too fast (photosynthesis), too slow (geological processes), too complex (automobile engines), or too dangerous (toxic materials). In recent years, computational studies have produced enormous advances in almost all fields of scientific and engineering inquiry, including DNA sequencing, behavioral modeling, global climatic predictions, drug design, financial systems, and medical visualization.

A number of institutions both within and outside Ohio have initiated minor programs in computational science in recognition of its increasing importance in science and engineering research and applications. The proposed minor program focuses on providing the relevant expertise in the principles of modeling and simulation, computer science skills in programming, and related mathematics and analytics concepts. A similar minor program in engineering at OSU was approved last year and went into operation in the 2012-2013 academic year.

The basic requirements of this minor program are guided by a document associated with the Ralph Regula School of Computational Science, which was established as a “virtual school” by the Ohio Board of Regents in December 2005. RRSCS is a run from the Ohio Supercomputer Center and does not offer degrees of its own but serves to organize and coordinate statewide efforts to integrate computational science programs into the curricula at participating institutions, one of which is OSU. Computational science and engineering is an interdisciplinary field with expertise scattered among different departments and different institutions—though OSU offers expertise and coursework in all these areas. The intention of RRSCS is to sponsor shared, inter-institutional programs that take advantage of the existing expertise, make it widely available, and limit the duplication of effort and expense where possible. For completeness, and to answer any questions that might arise about this unusual aspect of this program, the full proposal for the just approved minor program is provided as Attachment B. One change in the previously approved minor is to make the “optimization” course an elective rather than a requirement. This keeps the total credit hours upon conversion to semesters within the desired range.

A faculty committee represented by several of the participating departments will oversee the minor program. They will review proposed exceptions to the course requirements for students and also periodically review the program and recommend updates and amendments. The committee will also review proposals for research and internship experiences from the participating students to ensure that those experiences meet the intended practical experience in the use of computational modeling techniques.

Overview of the Minor Curriculum

The skills underlying the minor program curriculum are based on the results of a National Science Foundation grant that created competencies in computational science for undergraduate students. An interdisciplinary team consisting of faculty from all of the related disciplines created the competencies as a part of the grant outcomes. That included a number of faculty from OSU. Subsequently, the competencies have been reviewed by an industry advisory committee and by a number of computational science experts at institutions nationwide. The

competencies are available online at http://hpcuniversity.org/educators/undergradCompetencies/ and are also attached as Appendix A.

The proposed minor curriculum consists of five required courses and at least one elective. These are shown in Table 1. All students are required to take a year of calculus, which is part of the current requirements for the target major fields. Calculus could be taken concurrently with the introductory modeling and simulation course, but is a prerequisite for the other courses. The courses are described in the table of competencies shown in Appendix A.

Coursework Requirements

The Computational Science minor requires the completion of at least 18 credits of approved coursework, including a required course in each of five designated areas and one elective, as summarized in Table 1.

Required Courses

1) Simulation and Modeling: An introductory course on the use of models (continuous and discrete) and simulation in science and engineering. An introductory course of this nature is being added by Computer Science and Engineering as CSE 2021. The course is aimed at Arts and Sciences majors with no previous computer science experience. There are also several related courses in engineering that meet this requirement for students with more advanced skills.

2) Programming and Algorithms: A course on computer programming and the use of a programming language for problem solving is the second required course. There are two Computer Science and Engineering courses that currently meet this requirement.

3) Numerical Methods: An applied introduction to use of numerical methods in solving linear and nonlinear equations, interpolation, numerical solution of differential equations. There are several mathematics and engineering courses that meet this requirement.

5) Capstone Research/ Internship Experience: Each student must complete a guided research project or internship on a computational topic. The mechanism used to satisfy this requirement may differ across departments – e.g. computationally oriented senior design project or honors thesis, or a computationally oriented independent research study with a faculty member at Ohio State, or internship experience with an external agency or company that meets the goals of this requirement. Credit will be given through available independent studies or research course designations.

The faculty oversight committee will review proposals for the capstone research experience to ensure that they meet the intended purpose of this requirement.

6) Domain-Specific Course: Any approved computationally oriented course from the student's major discipline. Currently there are related courses in biomedical informatics, chemistry, economics, geography, linguistics, mathematics, microbiology, physics, and psychology.

74) Optional course: Students then have the option of taking one of several courses as an elective. These include additional mathematics courses, courses in parallel programming, scientific visualization, or optimization. Other options may be considered in more advanced modeling and simulation, data management, or other related areas by petition.

	Topic	Course	Credit Hours	Terms offered	Required/ Elective
Prerequisites	Calculus	MATH 1151.xx	5	Au, Sp	
		MATH 1152.xx or Math 1172	5	Au, Sp	
Core Courses	Simulation and Modeling (Choose one of these courses)	MATH 1157	3	Sp	Required
		CSE 2021	3	Sp	
		ISE 5100	3	Au, Sp	
		ME 5372	3	Au	
		MATSCEN 4321	3	Au	
	Programming and Algorithms (Choose one of these courses)	CSE 1222	3	Au, Sp	Required
		CSE 2221	4	Au, Sp, Su	
	Numerical Methods (Choose one of these courses)	AERO 3581	3	Au	Required
		CSE 5361	3	Au, Sp	
		ECE 5510	3	Au	
		MATH 3607	3	Sp	
		MATH 5401	3	Sp	
		MECHENG 2850	3	Au, Sp	
Discipline Specific Courses	Capstone Research/Internship Experience (minimum 3 credits)	CIVILEN 4000.01	2	Au, Sp	Required
		MATH 4998; CHEM 4998 or other approved individualized research credits **	3-5	Au, Sp, May, Su, May + Su/ Au, Sp/Au, Sp	
	Discipline-specific Computationally oriented Course	CSE 3521	3	Au, Sp	Required
		CSE 3341	3	Au, Sp	
		MICRBIO 5161H	3	N/A	
		BMI 5730	3	Sp	
		CHEM 5440	3	Au	
		MATH 5651	3	Sp	
		PHYS 6810	4	Sp	
		LING 5801	3	Au	
		LING 5802	3	Sp	
		ECON 4050	3	Au, Sp	
		ECON 5001	3	Au, Sp	
		GEOG 5221	3	Au	
		PSYCH 5608 or 5609 or 5618	3	Sp	
Elective: Choose at least one course from the following (3 credits total required)	Differential Equation and Discrete Dynamical Course	MATH 2255	3	Su, Au, Wi, Sp	Elective
		MATH 2415	3	Su, Au, Wi, Sp	
		MATH 2568	3	Su, Au, Wi, Sp	
	Parallel Programming	CSE 5441	3	Au	Elective
	Scientific Visualization	CSE 5544	1-5	Su, Au, Wi, Sp	Elective
	Optimization	CEG (CIVILEN) 4760	3	Sp	Elective
		ECE 5759	3	Au	
		ISE 3200	3	Au, Sp	
		MATSCEN 4181	3	Au	
		Total minimum credit hours: 18			

** A proposal for the individual research project must be submitted in advance of the research experience, and then approved by the Faculty Oversight Committee for the minor program.

The Ohio State University
College of Arts and Sciences

Computational Science Minor

Arts and Sciences Advising and Academic Services
100 Denney Hall
164 West 17th Avenue
Columbus, OH 43210
http://ascadvising.osu.edu

The minor in computational science consists of a minimum of 18 credit hours of course work: Computer Science and Engineering (CSE) 2021, either CSE 1221 or 2221, a numerical methods course (CSE 5361, ECE 5510, MATH 3607, MATH 3401, or ME 2850), a capstone research or internship using an approved departmental research course designation, one discipline specific computational modeling course (CHEM 5440, MICROBIO 5161H, CSE 3521, CSE 3341, BMI 5730, MATH 5651, PHYS 6810, LING 5801, LING 5802, ECON 4050, ECON 5001, GEOG 5221, PSYCH 5608, PSYCH 5609, or PSYCH 5618), and one elective (MATH 2255, MATH 2415, MATH 2568, CSE 5441, CSE 5544, CEG 4760, ECE5759, ISE 3200, or MATSCEN 4181). Of these courses, at least 12 credit hours **must** be at the 2000 level or above.

After the Arts and Sciences advisor has approved your Minor Program Form, you should file the form with your college or school counselor. For further information about the minor program, contact ASC advising.

Computational Science Minor Program Guidelines

The following guidelines govern this minor.

Required for graduation No

Credit hours required A minimum of 18. Of these, at least 12 credit hours **must** be at the 2000 level or above.

Transfer credit hours allowed ~~A maximum of 6~~No more than one half of the credit hours required on the minor.

Overlap with the GE~~C~~ Permitted, unless specifically disallowed by an individual minor program.

Overlap with the major Not allowed and
- The minor is interdisciplinary and is available for any major in Arts and Sciences.
- The same courses cannot count on the minor and on the major.

Overlap between minors Each minor completed must contain 12 unique hours.

Grades required
- Minimum C– for a course to be listed on the minor.
- Minimum 2.00 cumulative point-hour ratio required for the minor.
- Course work graded Pass/Non-Pass cannot count on the minor.

Approval required The minor program should be approved by an advisor in ASC Advising or your departmental advisor in ASC. Requests to include in the minor any courses not listed herein must be submitted to the coordinator of the minor program Greg Kilcup, Department of Physics, kilcup.1@osu.edu.

Filing the minor program form The minor program form must be filed at least by the time the graduation application is submitted to a college or school counselor. You should plan to file the form as soon as you have decided to pursue the minor.

Changing the minor program Once the minor program is filed in the college office, any changes must be approved by an ASC advisor or the coordinator of the program.

Appendix A

Computational Science Minor Program Competencies

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

Competency/Descriptors
Explain the role of modeling in science and engineering **Descriptors:** Discuss the importance of modeling to science and engineering Discuss the history and need for modeling Discuss the cost effectiveness of modeling Discuss the time-effect of modeling (e.g. the ability to predict the weather) Define the terms associated with modeling to science and engineering List questions that would check/validate model results Describe future trends and issues in science and engineering Identify specific industry related examples of modeling in engineering (e.g., Battelle; P&G, material science, manufacturing, bioscience, etc.) Discuss application across various industries (e.g., economics, health, etc.)
Analyze modeling and simulation in computational science **Descriptors:** Identify different types of models and simulations Describe a model in terms of iterative process, linking physical and virtual worlds and the science of prediction Explain the use of models and simulation in hypothesis testing (e.g. scientific method)
Create a conceptual model **Descriptors:** Illustrate a conceptual modeling process through examples Identify the key parameters of the model Estimate model outcomes Utilize modeling software and/or spreadsheets to implement model algebraic equations (e.g. Vensim, Excel, MATLAB, Mathematica) Construct a simple computer visualization of the model results (e.g. infectious disease model, traffic flow, etc.) Validate the model with data Discuss model quality and the sources of errors
Examine various mathematical representations of functions **Descriptors:** Describe linear functions Define non-linear functions (e.g., polynomials, exponential, periodic, parameterized, etc.) Visualize functions utilizing software (e.g. Excel, Function flyer, etc.) Determine appropriate functional form to fit the data Demonstrate essential mathematical concepts related to modeling and simulation
Analyze issues in accuracy and precision **Descriptors:** Describe various types of numerical and experimental errors Explain the concept of systematic errors 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
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
Demonstrate computational programming utilizing a higher level language or modeling tool (e.g. Maple, MATLAB™, 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
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
Build event-based models **Descriptors:** Describe event-based modeling (e.g. SIMULINK™; Extend, ARENA) Run existing models Translate conceptual models (e.g., traffic flow utilizing SIMULINK™)
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
Demonstrate technical communication **Descriptors:** Demonstrate technical writing skills in the comprehensive report 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
Describe the fundamentals of problem solving **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.
Understand and write Pseudo code **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.
Use subprograms in program design **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.
Write code in a Programming language **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 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
Describe the solution methodology for first order linear differential and difference equations **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
Describe the solution methodology for systems of linear first order differential and difference equations **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)
Describe the solution methodology for higher order differential and difference equations **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.
Describe the solution methodology for differential equations using the Laplace Transforms **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)
Describe the solution methodology for non-linear differential equations **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-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
Understand number representation and computer errors **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
Analyze methods for solving non-linear equations **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)
Describe techniques for solving systems of linear equations **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
Analyze techniques for computing eigenvalues—eigenvectors (Optional) **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)
Describe interpolation and approximation methods **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)
Describe numerical methods for Ordinary Differential Equations **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
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
Describe the fundamental concepts of parallel programming and related architectures **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
Demonstrate parallel programming concepts using MPI **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
Demonstrate knowledge of parallel scalability **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
Demonstrate knowledge of parallel programming libraries and tools **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

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

Competency/Descriptors
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
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 ▪ Transform: scale, rotate, translate) ▪ Manipulate surface ▪ Manipulate lighting and camera 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
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 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
Use visualization software to display an isosurface **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
Use visualization software to complete a volumetric rendering 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
Utilize visualization software to visualize a vector dataset **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
Explore examples of image processing **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
Use advanced techniques applied to a real problem **Descriptors:** To be chosen by instructor based on instructor/student interest. Among suggested topics

are:

- 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)

Examine SciVis problems for Biological Sciences – "OMICS" applications
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
    - Visualize (and examine) similarities and differences
    - 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

- Discuss the differences between protein folding and protein structure prediction
- Explore different methods used in protein folding and structure prediction
- Apply different methods of protein structure prediction and compare the results
- 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

- Discuss and contrast different types of expression data – e.g., microarray gene expression data, protein expression data
- Discuss different visualization (and analyses) techniques used for expression data
- Apply (and compare outcomes) hierarchical clustering and k-means clustering to the same gene microarray expression data
- 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

Appendix B
Minor in Computational Science and Engineering
Primary Contacts: Bruce W. Weide (weide.1, 292-1517) and Steven Gordon (sgordon@osc.edu, 2-6082)

1. Fiscal Unit / Academic Organization

Department of Computer Science and Engineering (1435)

2. Administering College / Academic Group

College of Engineering / Department of Computer Science and Engineering (CSE)

3. Co-administering College / Academic Group

Not applicable

4. Semester Conversion Designation

c. Converted with minimal changes to program goals and/or curricular requirements

5. Program / Plan Name

Minor in Computational Science and Engineering

6. Type of Program

Undergraduate minor

7. Program Plan Code Abbreviation

TBD

8. Degree Title

Not applicable

9. Specializations / Sub-plans

Not applicable

10. Program Learning Goals

Not required at this time for minors

11. List of Semester Courses

See Attachment #1: Computational Science and Engineering Minor Proposed Program Requirements.

12. Program Rationale

The Computational Science and Engineering Minor was approved in Sp 2010. As the program has just commenced, minor changes are proposed along with those necessitated by converting course requirements to semesters.

The basic requirements of this minor program are guided by a document associated with the Ralph Regula School of Computational Science, which was established as a "virtual school" by the Ohio Board of Regents in December 2005. RRSCS is a run from the Ohio Supercomputer Center and does not offer degrees of its own but serves to organize and coordinate statewide efforts to integrate computational science programs into the curricula at participating institutions, one of which is OSU. Computational science and engineering is an interdisciplinary field with expertise scattered among different departments and different institutions—though OSU offers expertise and coursework in all these areas. The intention of RRSCS is to sponsor shared, inter-institutional programs that take advantage of the existing expertise, make it widely available, and limit the duplication of effort and expense where possible. For completeness, and to answer any questions that might

arise about this unusual aspect of this program, the full proposal for the just-approved minor program is provided as Attachment #4. The only change in the previously approved minor is to make the optimization course an elective rather than a requirement. This keeps the total credit hours upon conversion to semesters within the desired range.

13. Quarters Curriculum Advising Sheet

See Table 1: Current Advising Sheet, which shows the minor program proposal as previously approved as in the original document (Attachment #4) and presently serves as the Computational Science and Engineering Minor Advising Sheet.

14. Semesters Curriculum Advising Sheet

See Table 2 and Attachment #3: Proposed Advising Sheet.

15. Curricular Map

Not applicable

16. Associated Pre-Major or Area of Interest

Not applicable

17. Credit-Hour Changes

	Number of qtr-cr-hrs in current program[1]	Calculated result for 2/3 of current qtr-cr-hrs	Number of sem-cr-hrs required for proposed program	Change in cr-hrs
Total minimum cr-hrs required for completion of program	22	14.7	18	+3.3
Required cr-hrs offered by the unit	4 - 19	2.7 - 12.7	3 - 13	–0.7 - +0.3
Required cr-hrs offered outside of the unit	3 - 18	3.0 - 15.0	3 - 16	+3.0 - +4.0
Required prerequisite cr-hrs not included above	15	10.0	10	0.0

18. Rationale for Significant Change in Credit Hours

While none of the cr-hr changes exceeds 4.0, a brief explanation is in order, particularly for the first line: minimum total cr-hrs. Simply because of the "breakage" in cr-hrs as OSU courses are converted to semesters in different units *in the best interest of majors in those units* (i.e., without regard to the impact on this interdisciplinary minor program), meeting the overall coverage requirements of the RRSCS curriculum entails an increase in the minimum from an equivalent of 14.7 sem-cr-hrs to 18 sem-cr-hrs. We view this as still a very reasonable minor that will remain accessible to interested students in the sciences and engineering.

19. Transition Policy

No student who begins the Computational Science and Engineering Minor under quarters will have progress toward completion impeded by the transition to semesters. Computational Science and Engineering Minor requirements beginning Summer 2012 will be those in force for students under

[1] Numbers in this column are computed by considering the OSU courses listed in Attachment #2 that would be needed to satisfy the current requirements in all competency areas of the minor coursework. The official overall minimum for the minor is listed as 20 cr-hrs rather than 22 cr-hrs, as some courses at other institutions might qualify as substitutes for these OSU courses.

semesters; but *every* quarter-credit-hour that would have counted toward a Computational Science and Engineering Minor under the quarter-based program will count (as 2/3 of a semester-credit-hour) toward the requirements for the semester version. If necessary, a revision of specific requirements will be worked out for any Computational Science and Engineering Minor student who is caught in the transition, in consultation with the CSE Associate Chair.

— Xiaodong Zhang, CSE Department Chair

20. Assessment Practices

Not applicable

Table 1: Requirements for Undergraduate Computational Science Minor Current Advising Sheet (Quarter Version)		
Topic	**Courses**	**Required/Elective**
Prerequisites		
Calculus	Math 151, 152, 153 (or equivalent, e.g. Math 161, 162 or Math H190, H191)	
Core Courses		
Simulation and Modeling	One of: CEG 640, CSE 778, ISE 521, 704, ME 785, MSE 533	Required
Programming and Algorithms	One of: CSE 202, CSE 294P, CSE 221	Required
Numerical Methods	One of: AAE 581, CEG 406, CSE 541, ECE 715, Math 606, Math 607, ME 250	Required
Optimization	CEG 776, ECE 759, ISE 522, ME 761, MSE 600	Required
Discipline Specific Courses		
Capstone Research/Internship Experience (3 credits minimum)	CE 660, CSE 699 or H783,ME 564 or 565, MSE 695 or other approved individualized research credits	Required
Discipline-specific Computationally oriented Course	CSE 630, 655, 660, 670, 675, 680, Chem 644, MSE 756, Phys 780	Required
Elective: Choose at least one course from the following		
Differential Equations and Discrete Dynamical Systems	Math 255, Math 415, Math 568, Math 571	Elective
Parallel Programming	CSE 621	Elective
Scientific Visualization	CSE 694L	Elective

Table 2: Requirements for Undergraduate Computational Science Minor (Semester Version)

	Topic	Course	Credit Hours	Quarters offered	Required/ Elective
Prerequisites	Calculus	MATH 1151.xx	5	Au, Wi, Sp	
		MATH 1152.xx or Math 1172	5	Au, Wi, Sp	
Core Courses	Simulation and Modeling (One of Courses)	CEG 640	4	Au	Required
		ISE 7200	3	Au	
		ISE 5100	3	Au, Sp	
		ME 5372	3	Au	
		MATSCEN 4321	3	Au	
	Programming and Algorithms (One of Courses)	CSE 1222	3	Au, Sp	Required
		CSE 1221	2	Au, Sp	
		CSE 2221	4	Au, Sp, Su	
	Numerical Methods (One of Courses)	AERO 2581	N/A	N/A	Required
		CEG (CIVILEN) 2090	1	Au, Sp	
		CSE 5361	3	Au, Sp	
		ECE 5510	3	Au	
		MATH 3607	3	Sp	
		MECHENG 2850	3	Au, Sp	
Discipline Specific Courses	Capstone Research/Internship Experience (minimum 3 credits)	CIVILEN 4000.01	2	Au, Sp	Required
		CSE 4998, or MECHENG 4901.01, or 4902.01, or other approved individualized research credits	3-5	Au, Sp, May, Su, May + Su/ Au, Sp/Au, Sp	
	Discipline-specific Computationally oriented Course	CSE 3521	3	Au, Sp	Required
		CSE 3341	3	Au, Sp	
		CSE 3431	N/A	TBD	
		CSE 3241	3	Au, Sp	
		CSE 2331	3	Au, Sp, Su	
		CSE 5331	3	Au, Wi, Sp	
		CHEM 644	3	Au	
		NATSEN 6766	N/A	N/A	
		PHYS 780	N/A	N/A	
Elective: Choose at least one course from the following (3 credits total required)	Differential Equation and Discrete Dynamical Course	MATH 2255	3	Su, Au, Wi, Sp	Elective
		MATH 2415	3	Su, Au, Wi, Sp	
		MATH 2568	3	Su, Au, Wi, Sp	
	Parallel Programming	CSE 5441	3	Au	Elective
	Scientific Visualization	CSE 5544	1-5	Su, Au, Wi, Sp	Elective
	Optimization	CEG (CIVILEN) 4760	3	Sp	Elective
		ECE 5759	3	Au	
		ISE 3200	3	Au, Sp	
		MATSCEN 4181	3	Au	
		Total minimum credit hours: 18			

Minor Program Form
College of Arts and Sciences

Name _____

Student ID Number _____ Name .# _____

Minor **Computational Science**

This form should be submitted to your college or school office.

College/School of enrollment _____ Major _____

Expected date of graduation _____

Have you filed a degree application in your college office? Yes ☐ No ☐

Computational Science minor program guidelines

- The minor in computational science consists of a minimum of **18 credit hours** of course work: Computer Science and Engineering (CSE) 2021, either CSE 1221 or 2221, a numerical methods course (CSE 5361, ECE 5510, MATH 3607, MATH 3401, or ME 2850), a capstone research or internship using an approved departmental research course designation, one discipline specific computational modeling course (CHEM 5440, MICROBIO 5161H, CSE 3521, CSE 3341, BMI 5730, MATH 5651, PHYS 6810, LING 5801, LING 5802, ECON 4050, ECON 5001, GEOG 5221, PSYCH 5608, PSYCH 5609, or PSYCH 5618) and one elective (MATH 2255, MATH 2415, MATH 2568, CSE 5441, CSE 5544, CEG 4760, ECE5759, ISE 3200, or MATSCEN 4181).
- Minimum C– for a course to be listed on the minor.
- Of these courses, at least 12 credit hours **must** be at the 2000 level or above.
- Minimum 2.00 cumulative point-hour ratio required for the minor.

Course	Hours	Final Grade

Total Hours _____ **Original** ☐ **Revision** ☐

_____

Signature of Advisor Date

_____

Please Print Name of Advisor

_____

Academic Unit Campus Telephone and/or E-Mail