Education and research in computational science and data science; from bachelor programs to research

Morten Hjorth-Jensen Email morten.hjorth-jensen@fys.uio.no[1,2]

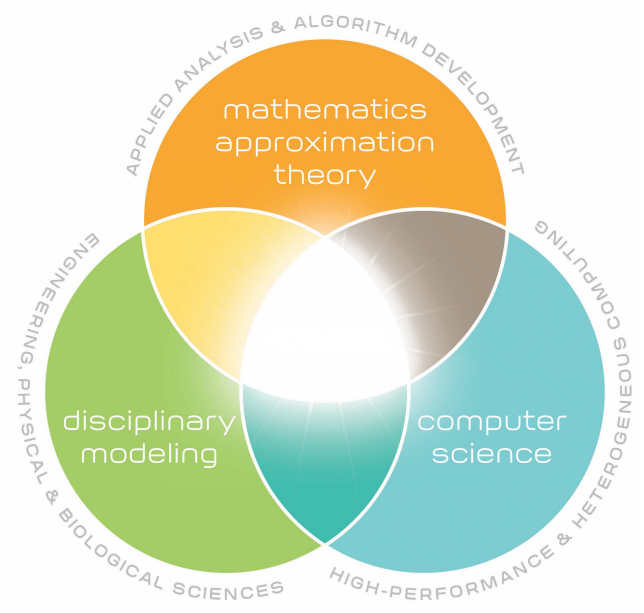
[1]National Superconducting Cyclotron Laboratory and Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, USA
[2]Department of Physics and Center for Computing in Science Education, University of Oslo, Oslo, Norway

OsloMet, October 22, 2018

Why should we focus on Computational Science and Data Science?

- By 2020, it is expected that one of every two jobs in the STEM fields will be in computing (Association for Computing Machinery, 2013)
- Computation is an essential and cross-cutting element of all STEM disciplines
- Computational science and Data science have developed into disciplines of their own right
- Computations and the understanding of large data sets play an even larger role in basically all disciplines of STEM fields, Medicine, the Social Sciences, the Humanities and education
- Students at both undergraduate and graduate level are unprepared to use computational modeling, data science, and high performance computing – skills valued by a very broad range of employers.
- The 3rd Industrial Revolution will alter significantly the demands on the workforce. To adapt a highly-qualified workforce to coming challenges requires strong fundamental bases in STEM fields. Computational Science and Data Science can provide such bases at all stages.

Computing across Disciplines at UiO



Education: Computational Science and Data Science at Norwegian universities

In Norway it is only UiO which offers Master of Science programs in Computational Science and Data Science. All other universities have only Master programs in Computer Science, with minor emphasis on computational science and/or data science. The University of Bergen has a Masters program in Applied Mathematics while UMB has a newly established program in data science. These are limited and more focused programs. Nationally, UiO is the only university which offers broad programs in Computational Science and Data Science.

Norway	University	Comp Science and Data dept	Bachelor program	Master program
	UiO	No	Planned	Yes
	NTNU	No	No	No
	UiT	No	No	No
	UiB	No	No	Applied Math
	UMB	No	No	Yes (Bioinf+stat datanalys
	OsloMet	No	No	No
	UiS	No	No	No
	UiA	No	No	No, but direction in AI
	UiN	No	No	No

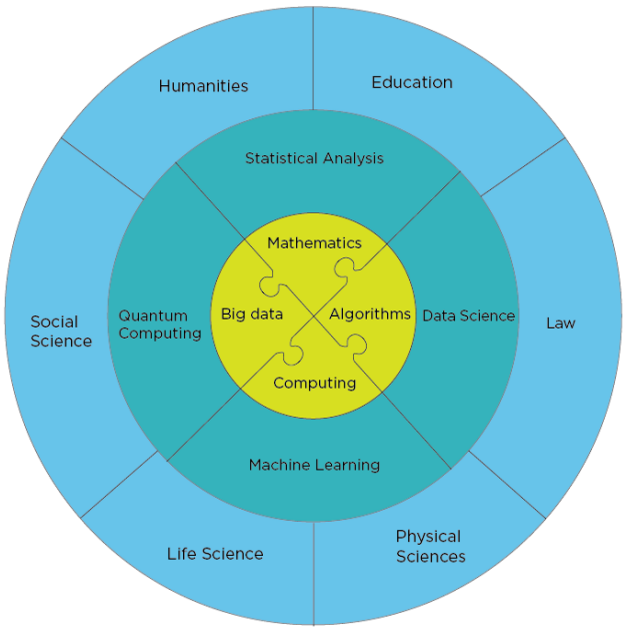
Out of 95 universities polled in the USA, there are less than 15 which have a department on Scientific Computing and more than 50 that have a center on Scientific Computing. Between 20 to 30 of these offer a bachelor, Master of Science or PhD program. On Data Science there are approximately 30 departments and 40 centers. Almost 50 of these universities offer a Masters degree in Data Science and close to 40 a PhD in Data Science.

How to introduce computing in all disciplines: Future Research directions

Enhance Computational Science and Data Science across the disciplines.

- Computational Science and Data Science in Mathematics and all of the physical sciences (Astrophysics, Chemistry, Geoscience and Physics)
- Bioinformatics
- Develop research programs in Quantum Computing and Quantum Information theory. Many universities are now developing research and educational strategies in Quantum Computing
- Develop data-driven discovery research programs utilizing recent developments in machine learning
- Computational life science
- Computational Materials Science
- Computational Economy and Data Science and computing in Law and the Social Sciences
- Data Science and computing in the Humanities
- And many more

And as a figure



Strengths, Possibilities and Synergies, a UiO based perspective

- Several Centers of excellence in research where Computational Science plays a major role
- Newly established center of excellence in education research
- Newly established Master of Science programs in Computational Science and Data Science
- Several excellent groups in STEM fiels who do Computational Science
- Computational topics are included in all undergraduate STEM programs, possibility to develop a bachelor program in Computational Science and Data Science
- Several educational prizes and awards related to computational science
- Lead in the development of computations in Life Science

Goals

- Position UiO as a leader in computational science by recruiting faculty whose expertise pertains to large-scale computing and mathematical foundations of data science - both generalists (algorithm/tool developers) and specialists (focused on specific disciplines).
- Develop a comprehensive set of courses and degree programs at both undergraduate and graduate levels that will give students across the university exposure to practical computational methods, understanding how to analyse data and more generally to the idea of computers as problem-solving tools.
- Develop an all university PhD program in Computational Science and Data Science
- Develop an all university Master of Science Program in Computational Science and Data Science
- Develop courses and course modules in Computational Science and Data Science for the private and the public sectors
- Develop a PhD program in Computational Science and Data Science tailored to the needs of the private and the public sectors
- Facilitate the adoption of computational tools and techniques for both research and education across campus, through education and faculty collaboration.
- Educate the next generation of school teachers and university teachers, with a strong focus on digital competences.

Unique opportunities

Computing competence represents a central element in scientific problem solving, from basic education and research to essentially almost all advanced problems in modern societies.

Computing competence **enlarges the body of tools available to students** and scientists beyond classical tools and **allows for a more generic handling of problems**. Focusing on algorithmic aspects **results in deeper insights** about scientific problems.

- Computing in scince courses allows us to bring important elements of scientific methods at a much earlier stage in our students' education.
- It gives the students skills and abilities that are asked for by society.
- It gives us as university teachers a unique opportunity to enhance students' insights about our disciplines and how to solve scientific problems.

Computing competence

Computing means solving scientific problems using all possible tools, including symbolic computing, computers and numerical algorithms, and analytical paper and pencil solutions .

Computing is about developing an understanding of the scientific method by enhancing algorithmic thinking when solving problems.

On the part of students, this competence involves being able to:

- understand how algorithms are used to solve mathematical problems,
- derive, verify, and implement algorithms,
- understand what can go wrong with algorithms,
- use these algorithms to construct reproducible scientific outcomes and to engage in science in ethical ways, and
- think algorithmically for the purposes of gaining deeper insights about scientific problems.

Better understanding of the scientific method

All these elements are central for maturing and gaining a better understanding of the modern scientific process *per se*.

The power of the scientific method lies in identifying a given problem as a special case of an abstract class of problems, identifying general solution methods for this class of problems, and applying a general method to the specific problem (applying means, in the case of computing, calculations by pen and paper, symbolic computing, or numerical computing by ready-made and/or self-written software). This generic view on problems and methods is particularly important for understanding how to apply available, generic software to solve a particular problem.

Why should basic university education undergo a shift towards modern computing?

- Algorithms involving pen and paper are traditionally aimed at what we often refer to as continuous models.
- Application of computers calls for approximate discrete models.
- Much of the development of **methods for continuous models are now being replaced by methods for discrete models** in science and industry, simply because **much larger classes of problems can be addressed** with discrete models, often also by simpler and more generic methodologies.

However, verification of algorithms and understanding their limitations requires much of the classical knowledge about continuous models.

So, why should basic university education undergo a shift towards modern computing?

Why is computing competence important?

The impact of the computer on mathematics and science is tremendous: **science and industry now rely on solving mathematical problems through computing**.

- Computing can increase the relevance in education by solving more realistic problems earlier.
- Computing through programming can be excellent training of creativity.
- Computing can enhance the understanding of abstractions and generalization.
- Computing can decrease the need for special tricks and tedious algebra, and shifts the focus to problem definition, visualization, and "what if" discussions.

Disasters attributable to poor management of code and more

Have you been paying attention in your numerical analysis or scientific computation courses? If not, it could be a costly mistake. Here are some real life examples of what can happen when numerical algorithms are not correctly applied.

1. The Patriot Missile failure, in Dharan, Saudi Arabia, on February 25, 1991 which resulted in 28 deaths, is ultimately attributable to poor handling of rounding errors.
2. The explosion of the Ariane 5 rocket just after lift-off on its maiden voyage off French Guiana, on June 4, 1996, was ultimately the consequence of a simple overflow.
3. The sinking of the Sleipner A offshore platform in Gandsfjorden near Stavanger, Norway, on August 23, 1991, resulted in a loss of nearly one billion dollars. It was found to be the result of inaccurate finite element analysis.
4. Scaling error, metric and imperial system of units, see the Mars expedition.

Modeling and computations as a way to enhance algorithminc thinking

Algorithmic thinking as a way to

- Enhance instruction based teaching
- Introduce Research based teaching from day one
- Trigger further insights in math and other disciplines
- Validation and verification of scientific results (the PRL example), with the possibility to emphasize ethical aspects as well. Version control is central.
- Good working practices from day one.

The foundation, Computing in Science Education at UiO: Can we catch many birds with one stone?

- How can we include and integrate an algorithmic (computational) perspective in our basic education?
- Can this enhance the students' understanding of mathematics and science?
- Can it strengthen research based teaching?

What is needed?

Programming. A compulsory programming course with a strong mathematical flavour. *Should give a solid foundation in programming as a problem solving technique in mathematics.* Programming is understanding! The line of thought when solving mathematical problems numerically enhances algorithmic thinking, and thereby the students' understanding of the scientific process.

Mathematics and numerics. Mathematics is at least as important as before, but should be supplemented with development, analysis, implementation, verification and validation of numerical methods. Science ethics and better understanding of the pedagogical process, almost for free!

Sciences. Training in modeling and problem solving with numerical methods and visualisation, as well as traditional methods in Science courses, Physics, Chemistry, Biology, Geology, Engineering...

Implementation

Crucial ingredients.

- Support from governing bodies (high priorities at the College of Natural Science at UOslo and the Physics and Astronomy department of Michigan State University)
- Cooperation across departmental boundaries
- Willingness by individuals to give priority to teaching reform

Consensus driven approach.

Implementation in Oslo: The C(omputing in)S(cience)E(Education) project

What we do.

- Coordinated use of computational exercises and numerical tools in most undergraduate courses.
- Help update the scientific staff's competence on computational aspects and give support (scientific, pedagogical and financial) to those who wish to revise their courses in a computational direction.
- Teachers get good summer students to aid in introducing computational exercises
- Develop courses and exercise modules with a computational perspective, both for students and teachers. New textbooks!!
- Basic idea: mixture of mathematics, computation, informatics and topics from the physical sciences.

Interesting outcome: higher focus on teaching and pedagogical issues!!

Example of bachelor program, Physics and Astronomy (astro path), University of Oslo

6th semester	**AST3210** Radiation I	**Choice**	**Choice**
5th semester	**FYS2160** Thermodynamics and statistical physics	**AST2120** The stars	**AST2210** Observational astronomy
4th semester	**FYS2140** Quantum physics	**Choice**	**EXPHIL03** Examen philosophicum
3rd semester	**FYS1120** Electromagnetism	**AST1100** Introduction to astrophysics / **GEF1100** The climate system	**MAT1120** Linear algebra
2nd semester	**FYS-MEK1110** Mechanics	**MEK1100** Vector calculus	**MAT1110** Calculus and linear algebra
1st semester	**MAT1100** Calculus	**MAT-INF1100** Modelling and computations	**INF1100** Introduction to programming with scientific applications
	10 credits	**10 credits**	**10 credits**

Table 2. Programme option for Astronomy in the bachelor programme Physics, Astronomy and Meteorology at UiO.

Example: Computations from day one

Differentiation. Three courses the first semester: MAT1100, MAT-INF1100 og INF1100.

- Definition of the derivative in MAT1100 (Calculus and analysis)

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}.$$

- Algorithms to compute the derivative in MAT-INF1100 (Mathematical modelling with computing)

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2).$$

- Implementation in Python in INF1100

```
def differentiate(f, x, h=1E-5):
    return (f(x+h) - f(x-h))/(2*h)
```

Example: Computations from day one

Differentiation and comparison with symbolic expressions. Combined with the possibility of symbolic calculations with *Sympy*, Python offers an environment where students and teachers alike can test many different aspects of mathematics and numerical mathematics, in addition to being able to verify and validate their codes. The following simple example shows how to extend the

simple function for computing the numerical derivative with the possibility of obtaining the closed form or analytical expression

```
def differentiate(f, x, h=1E-5, symbolic=False):
    if symbolic:
        import sympy
        return sympy.lambdify([x], sympy.diff(f, x))
    else:
        return (f(x+h) - f(x-h))/(2*h)
```

Other Examples

Integration by Trapezoidal Rule.

- Definition of integration in MAT1100 (Calculus and analysis).
- The algorithm for computing the integral vha the Trapezoidal rule for an interval $x \in [a, b]$

$$\int_a^b (f(x)dx \approx \frac{1}{2}\left[f(a) + 2f(a+h) + \cdots + 2f(b-h) + f(b)\right]$$

- Taught in MAT-INF1100 (Mathematical modelling)
- The algorithm is then implemented in INF1100 (programming course).

Typical implementation first semester of study

Integration by Trapezoidal Rule.

```
from math import exp, log, sin
def Trapez(a,b,f,n):
   h = (b-a)/float(n)
   s = 0
   x = a
   for i in range(1,n,1):
       x = x+h
       s = s+ f(x)
   s = 0.5*(f(a)+f(b)) +s
   return h*s

def f1(x):
    return exp(-x*x)*log(1+x*sin(x))

a = 1;  b = 3; n = 1000
result = Trapez(a,b,f1,n)
print(result)
```

Symbolic calculations and numerical calculations in one code

Python offers an extremely versatile programming environment, allowing for the inclusion of analytical studies in a numerical program. Here we show an

example code with the **trapezoidal rule** using **SymPy** to evaluate an integral and compute the absolute error with respect to the numerically evaluated one of the integral $\int_0^1 dx x^2 = 1/3$:

```
from math import *
from sympy import *
def Trapez(a,b,f,n):
   h = (b-a)/float(n)
   s = 0
   x = a
   for i in range(1,n,1):
       x = x+h
       s = s+ f(x)
   s = 0.5*(f(a)+f(b)) +s
   return h*s

#  function to compute pi
def function(x):
    return x*x

a = 0.0;  b = 1.0; n = 100
result = Trapez(a,b,function,n)
print("Trapezoidal rule=", result)
# define x as a symbol to be used by sympy
x = Symbol('x')
exact = integrate(function(x), (x, 0.0, 1.0))
print("Sympy integration=", exact)
# Find relative error
print("Relative error", abs((exact-result)/exact))
```

Error analysis

The following extended version of the trapezoidal rule allows you to plot the relative error by comparing with the exact result. By increasing to 10^8 points one arrives at a region where numerical errors start to accumulate.

```
from math import log10
import numpy as np
from sympy import Symbol, integrate
import matplotlib.pyplot as plt
# function for the trapezoidal rule
def Trapez(a,b,f,n):
   h = (b-a)/float(n)
   s = 0
   x = a
   for i in range(1,n,1):
       x = x+h
       s = s+ f(x)
   s = 0.5*(f(a)+f(b)) +s
   return h*s
#  function to compute pi
def function(x):
    return x*x
# define integration limits
a = 0.0;  b = 1.0;
# find result from sympy
# define x as a symbol to be used by sympy
x = Symbol('x')
exact = integrate(function(x), (x, a, b))
```

```
# set up the arrays for plotting the relative error
n = np.zeros(9); y = np.zeros(9);
# find the relative error as function of integration points
for i in range(1, 8, 1):
    npts = 10**i
    result = Trapez(a,b,function,npts)
    RelativeError = abs((exact-result)/exact)
    n[i] = log10(npts); y[i] = log10(RelativeError);
plt.plot(n,y, 'ro')
plt.xlabel('n')
plt.ylabel('Relative error')
plt.show()
```

Integrating numerical mathematics with calculus

The last example shows the potential of combining numerical algorithms with symbolic calculations, allowing thereby students and teachers to

- Validate and verify their algorithms.
- Including concepts like unit testing, one has the possibility to test and validate several or all parts of the code.
- Validation and verification are then included *naturally* and one can develop a better attitude to what is meant with an ethically sound scientific approach.
- The above example allows the student to also test the mathematical error of the algorithm for the trapezoidal rule by changing the number of integration points. The students get trained from day one to think error analysis.
- With an Jupyter/Ipython notebook the students can keep exploring similar examples and turn them in as their own notebooks.

Additional benefits: A structured approach to solving problems

In this process we easily bake in

1. How to structure a code in terms of functions
2. How to make a module
3. How to read input data flexibly from the command line
4. How to create graphical/web user interfaces
5. How to write unit tests (test functions or doctests)
6. How to refactor code in terms of classes (instead of functions only)
7. How to conduct and automate large-scale numerical experiments
8. How to write scientific reports in various formats (LaTeX, HTML)

Additional benefits: A structure approach to solving problems

The conventions and techniques outlined here will save you a lot of time when you incrementally extend software over time from simpler to more complicated problems. In particular, you will benefit from many good habits:

1. New code is added in a modular fashion to a library (modules)
2. Programs are run through convenient user interfaces
3. It takes one quick command to let all your code undergo heavy testing
4. Tedious manual work with running programs is automated,
5. Your scientific investigations are reproducible, scientific reports with top quality typesetting are produced both for paper and electronic devices.

Learning outcomes three first semesters

Knowledge of basic algorithms.

- Differential equations: Euler, modified Euler and Runge-Kutta methods (first semester)
- Numerical integration: Trapezoidal and Simpson's rule, multidimensional integrals (first semester)
- Random numbers, random walks, probability distributions and Monte Carlo integration (first semester)
- Linear Algebra and eigenvalue problems: Gaussian elimination, LU-decomposition, SVD, QR, Givens rotations and eigenvalues, Gauss-Seidel. (second and third semester)
- Root finding and interpolation etc. (all three first semesters)
- Processing of sound and images (first semester).

The students have to code several of these algorithms during the first three semesters.

Later courses

Later courses should build on this foundation as much as possible.

1. In particular, the course should not be too basic! There should be progression in the use of mathematics, numerical methods and programming, as well as science.
2. Computational platform: Python, fully object-oriented and allows for seamless integration of c++ and Fortran codes, as well as Matlab-like programming environment. Makes it easy to parallelize codes as well.

Coordination

- Teachers in other courses need therefore not use much time on numerical tools. Naturally included in other courses.

Challenges...

.. and objections. *Standard objection: computations take away the attention from other central topics in 'my course'.*

CSE incorporates computations from day one, and courses higher up do not need to spend time on computational topics (technicalities), but can focus on the interesting science applications. Coordination and synchronization across departments and courses. Increases collaboration on teaching and awareness of pedagical research.

- To help teachers: Developed pedagogical modules which can aid university teachers. Own course for teachers.

Physical Sciences implementation: Examples of simple algorithms, initial value problems and proper scaling of equations

1. Ordinary differential equations (ODE): RLC circuit
2. ODE: Classical pendulum
3. ODE: Solar system
4. and many more cases

Can use essentially the **same algorithms to solve these problems**, either some simple modified Euler algorithms or some Runge-Kutta class of algorithms or perhaps the so-called Verlet class of algorithms. **Algorithms students use in one course can be reused in other courses**.

Mechanics and electromagnetism, initial value problems

When properly scaled, these equations are essentially the same. Scaling is important. Classical pendulum with damping and external force as it could appear in a mechanics course (PHY 321)

$$ml\frac{d^2\theta}{dt^2} + \nu\frac{d\theta}{dt} + mgsin(\theta) = Acos(\omega t).$$

Easy to solve numerically and then visualize the solution. Almost the same equation for an RLC circuit in the electromagnetism course (PHY 482)

$$L\frac{d^2Q}{dt^2} + \frac{Q}{C} + R\frac{dQ}{dt} = Acos(\omega t).$$

Mechanics and electromagnetism, initial value problems and now proper scaling

Classical pendulum equations with damping and external force

$$\frac{d\theta}{d\hat{t}} = \hat{v},$$

and

$$\frac{d\hat{v}}{d\hat{t}} = Acos(\hat{\omega}\hat{t}) - \hat{v}\xi - \sin(\theta),$$

with $\omega_0 = \sqrt{g/l}$, $\hat{t} = \omega_0 t$ and $\xi = mg/\omega_0\nu$.

The RLC circuit

$$\frac{dQ}{d\hat{t}} = \hat{I},$$

and

$$\frac{d\hat{I}}{d\hat{t}} = Acos(\hat{\omega}\hat{t}) - \hat{I}\xi - Q,$$

with $\omega_0 = 1/\sqrt{LC}$, $\hat{t} = \omega_0 t$ and $\xi = CR\omega_0$.

The equations are essentially the same. **Great potential for abstraction**.

Other examples of simple algorithms that can be reused in many courses, two-point boundary value problems and scaling

These physics examples can all be studied using almost the same types of algorithms, simple eigenvalue solvers and Gaussian elimination with the same starting matrix!

1. A buckling beam and Toeplitz matrices (mechanics and mathematical methods), eigenvalue problems
2. A particle in an infinite potential well, quantum eigenvalue problems
3. A particle (or two) in a general quantum well, quantum eigenvalue problems
4. Poisson's equation in one dim, linear algebra (electromagnetism)
5. The diffusion equation in one dimension (Statistical Physics), linear algebra
6. and many other cases

The power of numerical methods

The last examples shows the potential of combining numerical algorithms with analytical results (or eventually symbolic calculations), allowing thereby students and teachers to

- make abstraction and explore other physics cases easily where no analytical solutions are known
- Validate and verify their algorithms.
- Including concepts like unit testing, one has the possibility to test and validate several or all parts of the code.
- Validation and verification are then included *naturally* and one can develop a better attitude to what is meant with an ethically sound scientific approach.
- The above examples allow the student to test the mathematical error of the algorithm for the eigenvalue solver by changing the number of integration points. The students get trained from day one to think error analysis.
- The algorithm can be tailored to any kind of one-particle problem used in quantum mechanics or eigenvalue problems
- A simple rewrite allows for reuse in linear algebra problems for solution of say Poisson's equation in electromagnetism, or the diffusion equation in one dimension.

Which aspects are important for a successful introduction of CSE?

- Early introduction, programming course at beginning of studies linked with math courses and science and engineering courses.
- Crucial to learn proper programming at the beginning.
- Good TAs
- Choice of software.
- Textbooks and modularization of topics, ask for details
- Resources and expenses.
- Tailor to specific disciplines.
- Organizational matters.
- Strong education research groups.

CSE Summary

- Make our research visible in early undergraduate courses, enhance research based teaching
- Possibility to focus more on understanding and increased insight.
- Impetus for broad cooperation in teaching. Broad focus on university pedagogical topics.
- Strengthening of instruction based teaching (expensive and time-consuming).
- Give our candidates a broader and more up-to-date education with a problem-based orientation, often requested by potential employers.
- And perhaps the most important issue: does this enhance the student's insight in the Sciences?

We invite you to visit (online and/or in real life) our new center on Computing in Science Education

Summary: New Master of Science programs

The two new Master of Science programs

- Computational Science
- Data Science

are a direct consequence of the Computing in Science Education project, which has aimed at a full integration and coordination of computational methods in basic undergrad natural science programs.

We are planning new undergraduate programs in Computational Science and Data Science, as well as cross-disciplinary programs at all levels.

Summary: Research

We give students an education relevant for many exciting future research directions.

- Computational Science and Data Science in Mathematics and all of the physical sciences (Astrophysics, Chemistry, Geoscience and Physics)
- Bioinformatics
- Develop research programs in Quantum Computing and Quantum Information theory. Many universities are now developing research and educational strategies in Quantum Computing

- Develop data-driven discovery research programs utilizing recent developments in machine learning
- Computational life science
- Computational Materials Science
- Computational Economy and Data Science and computing in Law and the Social Sciences
- Data Science and computing in the Humanities
- And many more