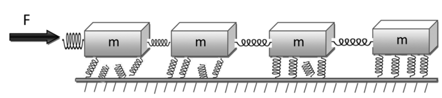
Computing in Physics Education

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

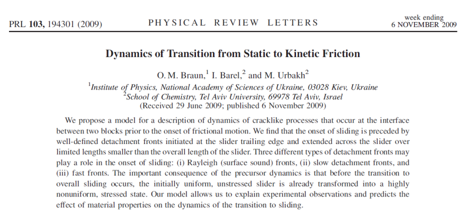
[1]National Superconducting Cyclotron Laboratory and Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, USA
[2]Department of Physics and Center for Computing in Science Education, University of Oslo, Oslo, Norway

Congresso Nazionale della SIF, Trento, 11-15 Settembre 2017

Wouldn't it be cool if your mechanics students could reproduce results in a PRL?



Grand challenge project in FYS-MEK1100 (Mechanics, University of Oslo), Second Semester: a friction model to be solved as coupled ODEs. And find problems with the article?



PRL **103,** 194301 (2009) PHYSICAL REVIEW LETTERS week ending 6 NOVEMBER 2009

Dynamics of Transition from Static to Kinetic Friction

O. M. Braun,[1] I. Barel,[2] and M. Urbakh[2]
[1]*Institute of Physics, National Academy of Sciences of Ukraine, 03028 Kiev, Ukraine*
[2]*School of Chemistry, Tel Aviv University, 69978 Tel Aviv, Israel*
(Received 29 June 2009; published 6 November 2009)

We propose a model for a description of dynamics of cracklike processes that occur at the interface between two blocks prior to the onset of frictional motion. We find that the onset of sliding is preceded by well-defined detachment fronts initiated at the slider trailing edge and extended across the slider over limited lengths smaller than the overall length of the slider. Three different types of detachment fronts may play a role in the onset of sliding: (i) Rayleigh (surface sound) fronts, (ii) slow detachment fronts, and (iii) fast fronts. The important consequence of the precursor dynamics is that before the transition to overall sliding occurs, the initially uniform, unstressed slider is already transformed into a highly nonuniform, stressed state. Our model allows us to explain experimental observations and predicts the effect of material properties on the dynamics of the transition to sliding.

DOI: 10.1103/PhysRevLett.103.194301 PACS numbers: 46.55.+d, 46.50.+a, 81.40.Pq

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 basic physics 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 physics 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.

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.

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 modelling 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 (now priority 1 of the College of Natural Science at UOslo)
- 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.
- 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, astrophysics

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)
```

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.

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

A buckling beam, or a quantum mechanical particle in an infinite well

This is a two-point boundary value problem

$$R\frac{d^2u(x)}{dx^2} = -Fu(x),$$

where $u(x)$ is the vertical displacement, R is a material specific constant, F the force and $x \in [0, L]$ with $u(0) = u(L) = 0$.

Scale equations with $x = \rho L$ and $\rho \in [0, 1]$ and get (note that we change from $u(x)$ to $v(\rho)$)

$$\frac{d^2v(\rho)}{dx^2} + Kv(\rho) = 0,$$

a standard eigenvalue problem with $K = FL^2/R$.

If you replace $R = -\hbar^2/2m$ and $-F = \lambda$, we have the quantum mechanical variant for a particle moving in a well with infinite walls at the endpoints.

Discretize and get the same type of problem

Discretize the second derivative and the rhs

$$-\frac{v_{i+1} - 2v_i + v_{i-i}}{h^2} = \lambda v_i,$$

with $i = 1, 2, \ldots, n$. We need to add to this system the two boundary conditions $v(0) = v_0$ and $v(1) = v_{n+1}$. The so-called Toeplitz matrix (special case from the discretized second derivative)

$$\mathbf{A} = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & \dots & \dots & \dots & \dots & \dots \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}$$

with the corresponding vectors $\mathbf{v} = (v_1, v_2, \ldots, v_n)^T$ allows us to rewrite the differential equation including the boundary conditions as a standard eigenvalue problem

$$\mathbf{A}\mathbf{u} = \lambda \mathbf{v}.$$

The Toeplitz matrix has analytical eigenpairs!! Adding a potential along the diagonals allows us to reuse this problem for many types of physics cases.

Adding complexity, hydrogen-like atoms or other one-particle potentials

We are first interested in the solution of the radial part of Schroedinger's equation for one electron. This equation reads

$$-\frac{\hbar^2}{2m}\left(\frac{1}{r^2}\frac{d}{dr}r^2\frac{d}{dr} - \frac{l(l+1)}{r^2}\right)R(r) + V(r)R(r) = ER(r).$$

Suppose in our case $V(r)$ is the harmonic oscillator potential $(1/2)kr^2$ with $k = m\omega^2$ and E is the energy of the harmonic oscillator in three dimensions. The oscillator frequency is ω and the energies are

$$E_{nl} = \hbar\omega\left(2n + l + \frac{3}{2}\right),$$

with $n = 0, 1, 2, \ldots$ and $l = 0, 1, 2, \ldots$.

Radial Schroedinger equation

Since we have made a transformation to spherical coordinates it means that $r \in [0, \infty)$. The quantum number l is the orbital momentum of the electron. Then we substitute $R(r) = (1/r)u(r)$ and obtain

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \left(V(r) + \frac{l(l+1)}{r^2}\frac{\hbar^2}{2m}\right)u(r) = Eu(r).$$

The boundary conditions are $u(0) = 0$ and $u(\infty) = 0$.

Scaling the equations

We introduce a dimensionless variable $\rho = (1/\alpha)r$ where α is a constant with dimension length and get

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}v(\rho) + \left(V(\rho) + \frac{l(l+1)}{\rho^2}\frac{\hbar^2}{2m\alpha^2}\right)v(\rho) = Ev(\rho).$$

Let us choose $l = 0$. Inserting $V(\rho) = (1/2)k\alpha^2\rho^2$ we end up with

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}v(\rho) + \frac{k}{2}\alpha^2\rho^2 v(\rho) = Ev(\rho).$$

We multiply thereafter with $2m\alpha^2/\hbar^2$ on both sides and obtain

$$-\frac{d^2}{d\rho^2}v(\rho) + \frac{mk}{\hbar^2}\alpha^4\rho^2 v(\rho) = \frac{2m\alpha^2}{\hbar^2}Ev(\rho).$$

A natural length scale comes out automagically when scaling

We have thus

$$-\frac{d^2}{d\rho^2}v(\rho) + \frac{mk}{\hbar^2}\alpha^4\rho^2 v(\rho) = \frac{2m\alpha^2}{\hbar^2}Ev(\rho).$$

The constant α can now be fixed so that

$$\frac{mk}{\hbar^2}\alpha^4 = 1,$$

and it defines a natural length scale (like the Bohr radius does)

$$\alpha = \left(\frac{\hbar^2}{mk}\right)^{1/4}.$$

Defining

$$\lambda = \frac{2m\alpha^2}{\hbar^2}E,$$

we can rewrite Schroedinger's equation as

$$-\frac{d^2}{d\rho^2}v(\rho) + \rho^2 v(\rho) = \lambda v(\rho).$$

This is similar to the equation for a buckling beam except for the potential term. In three dimensions the eigenvalues for $l = 0$ are $\lambda_0 = 1.5, \lambda_1 = 3.5, \lambda_2 = 5.5, \ldots$.

Discretizing

Define first the diagonal matrix element

$$d_i = \frac{2}{h^2} + V_i,$$

and the non-diagonal matrix element

$$e_i = -\frac{1}{h^2}.$$

In this case the non-diagonal matrix elements are given by a mere constant. *All non-diagonal matrix elements are equal.*

With these definitions the Schroedinger equation takes the following form

$$d_i u_i + e_{i-1} v_{i-1} + e_{i+1} v_{i+1} = \lambda v_i,$$

where v_i is unknown. We can write the latter equation as a matrix eigenvalue problem

$$\begin{bmatrix} d_1 & e_1 & 0 & 0 & \dots & 0 & 0 \\ e_1 & d_2 & e_2 & 0 & \dots & 0 & 0 \\ 0 & e_2 & d_3 & e_3 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & d_{n_{\rm step}-2} & e_{n_{\rm step}-1} \\ 0 & \dots & \dots & \dots & \dots & e_{n_{\rm step}-1} & d_{n_{\rm step}-1} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ \dots \\ \dots \\ v_{n_{\rm step}-1} \end{bmatrix} = \lambda \begin{bmatrix} cv_1 \\ v_2 \\ \dots \\ \dots \\ \dots \\ v_{n_{\rm step}-1} \end{bmatrix} \tag{1}$$

or if we wish to be more detailed, we can write the tridiagonal matrix as

$$\begin{pmatrix} \frac{2}{h^2} + V_1 & -\frac{1}{h^2} & 0 & 0 & \dots & 0 & 0 \\ -\frac{1}{h^2} & \frac{2}{h^2} + V_2 & -\frac{1}{h^2} & 0 & \dots & 0 & 0 \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2} + V_3 & -\frac{1}{h^2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & \frac{2}{h^2} + V_{n_{\rm step}-2} & -\frac{1}{h^2} \\ 0 & \dots & \dots & \dots & \dots & -\frac{1}{h^2} & \frac{2}{h^2} + V_{n_{\rm step}-1} \end{pmatrix} \tag{2}$$

Recall that the solutions are known via the boundary conditions at $i = n_{\rm step}$ and at the other end point, that is for ρ_0. The solution is zero in both cases.

The Python (2.7) code

The code sets up the Hamiltonian matrix by defining the minimun and maximum values of r with a maximum value of integration points. It plots the eigenfunctions of the three lowest eigenstates.

```
#Program which solves the one-particle Schrodinger equation
#for a potential specified in function
#potential().

from  matplotlib import pyplot as plt
```

```
import numpy as np
#Function for initialization of parameters
def initialize():
    RMin = 0.0
    RMax = 10.0
    lOrbital = 0
    Dim = 400
    return RMin, RMax, lOrbital, Dim
# Different types of potentials
def potential(r):
    return 0.5*r*r
    #  return 0.0
    #  return -1.0/r
    #if r >= 0.0 and r <= 10.0:
    #    V = -0.05
    #else:
    #    V =0.0
    #return V

#Get the boundary, orbital momentum and number of integration points
RMin, RMax, lOrbital, Dim = initialize()

#Initialize constants
Step    = RMax/(Dim+1)
DiagConst = 1.0/ (Step*Step)
NondiagConst =  -0.5 / (Step*Step)
OrbitalFactor = 0.5*lOrbital * (lOrbital + 1.0)

#Calculate array of potential values
v = np.zeros(Dim)
r = np.linspace(RMin,RMax,Dim)
for i in xrange(Dim):
    r[i] = RMin + (i+1) * Step;
    v[i] = potential(r[i]) + OrbitalFactor/(r[i]*r[i]);

#Setting up a tridiagonal matrix and finding eigenvectors and eigenvalues
Matrix = np.zeros((Dim,Dim))
Matrix[0,0] = DiagConst + v[0];
Matrix[0,1] = NondiagConst;
for i in xrange(1,Dim-1):
    Matrix[i,i-1]  = NondiagConst;
    Matrix[i,i]    = DiagConst + v[i];
    Matrix[i,i+1]  = NondiagConst;
Matrix[Dim-1,Dim-2] = NondiagConst;
Matrix[Dim-1,Dim-1] = DiagConst + v[Dim-1];
# diagonalize and obtain eigenvalues, not necessarily sorted
EigValues, EigVectors = np.linalg.eig(Matrix)
# sort eigenvectors and eigenvalues
permute = EigValues.argsort()
EigValues = EigValues[permute]
EigVectors = EigVectors[:,permute]
# now plot the results for the three lowest lying eigenstates
for i in xrange(3):
    print EigValues[i]
FirstEigvector = EigVectors[:,0]
SecondEigvector = EigVectors[:,1]
ThirdEigvector = EigVectors[:,2]
plt.plot(r, FirstEigvector**2 ,'b-',r, SecondEigvector**2 ,'g-',r, ThirdEigvector**2 ,'r-')
plt.axis([0,4.6,0.0, 0.025])
plt.xlabel(r'$r$')
plt.ylabel(r'Radial probability $r^2|R(r)|^2$')
```

```
plt.title(r'Radial probability distributions for three lowest-lying states')
plt.savefig('eigenvector.pdf')
plt.show()
```

The power of numerical methods

The last example 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 example allows the student to also 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.
- With an ipython notebook the students can keep exploring similar examples and turn them in as their own notebooks.

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.

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

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.
- Resources and expenses.
- Tailor to specific disciplines.
- Organizational matters.

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.
- 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?

More links

- Python and our first programming course, first semester course. Excellent new textbook by Hans Petter Langtangen, click here for the textbook or the online version
- Mathematical modelling course, first semester course. Textbook by Knut Morken to be published by Springer.

- Mechanics, second semester course. New textbook by Anders Malthe-Sorenssen, published by Springer, Undergraduate Lecture Notes in Physics
- Computational Physics I, fifth semester course. Textbook to be published by IOP in 2016, with online version