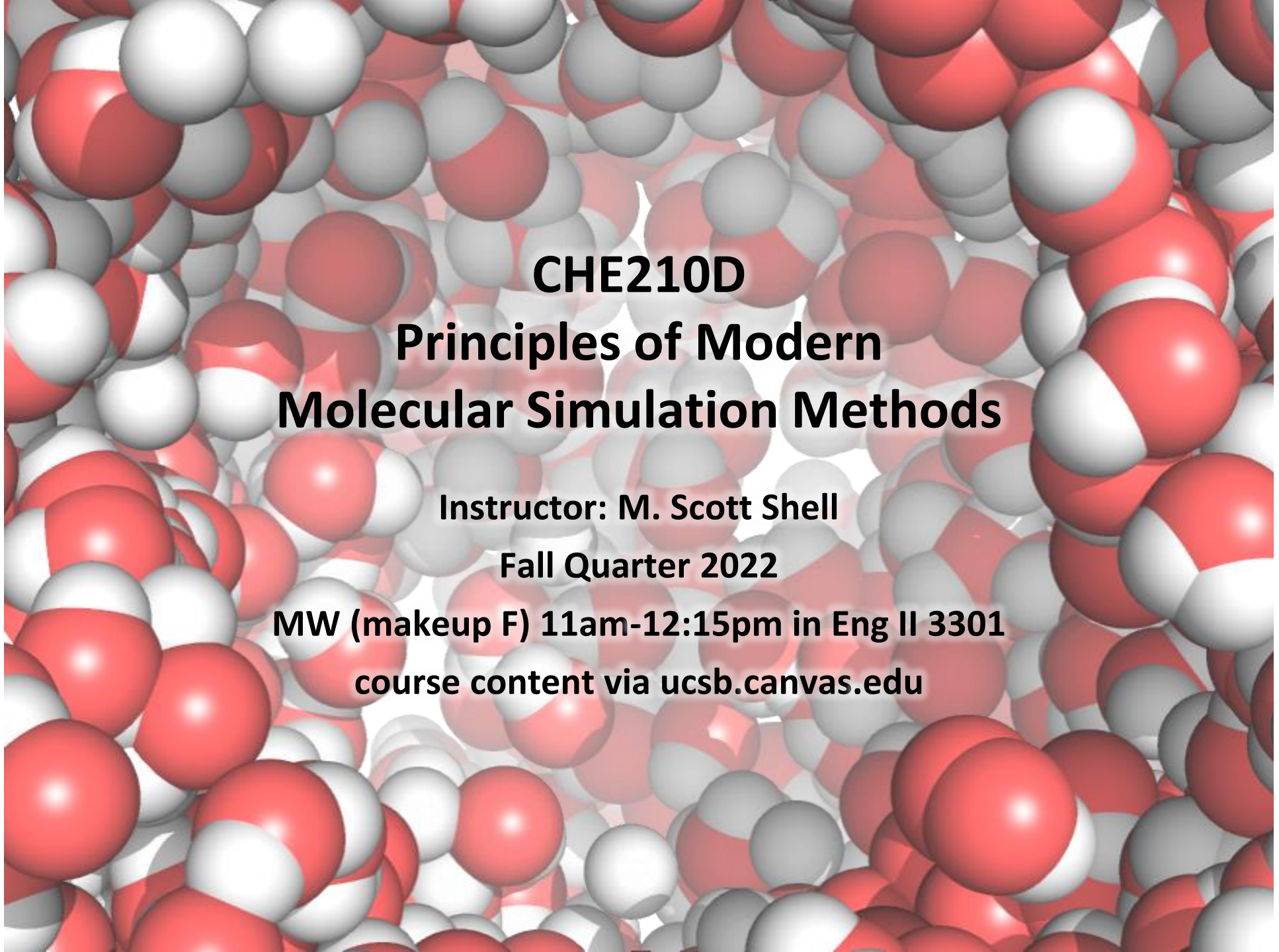


CHE210D
Principles of Modern Molecular Simulation Methods

Instructor: M. Scott Shell

Fall Quarter 2022

MW (makeup F) 11am-12:15pm in Eng II 3301

course content via ucsb.canvas.edu

The goals of this course

This course is all about *doing*.

- designing experiments
- running simulations
- analyzing results
- presenting data
- making movies
- working with existing molecular modeling software tools and online data

The goals of this course

- **formulation** of molecular models
- basic and advanced **algorithms** for computing thermodynamic and kinetic properties
- modern **analysis** techniques
- physical **intuition** for simulation “experiments”
- programming and visualization **tools**
- knowledge of **computational issues** and methods for improving efficiency

What's required

- a basic knowledge of statistical mechanics / physical chemistry
- some, but not extensive, programming experience
- access to a computer on which you can install (free, open-source) software
- **NOTE**: some examples assume Windows PC, but should be portable to other platforms

Course tracks

- **normal track**
  - undergraduate
  - 1st year graduate student in any area
  - 2nd year+ graduate student NOT involved in computational research

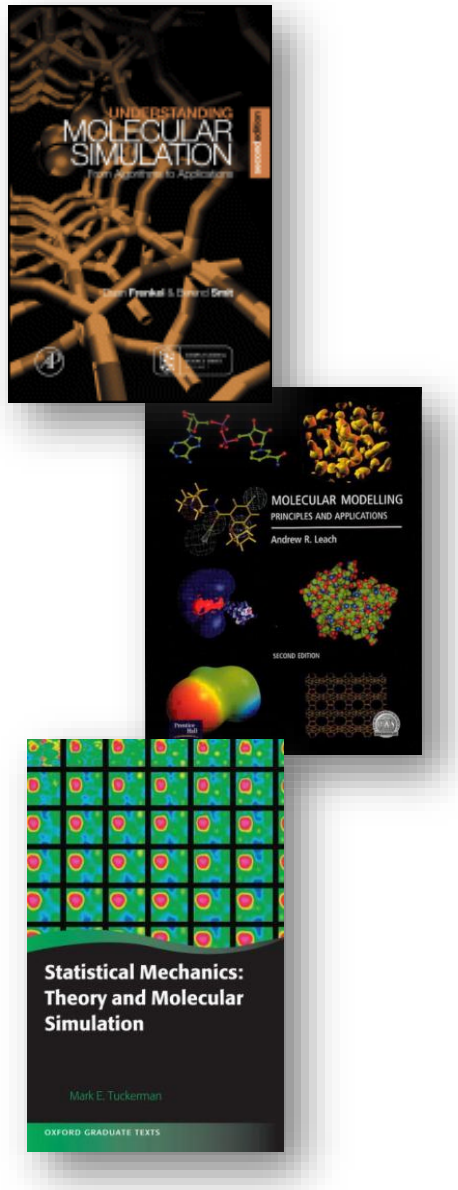
- **advanced track**
  - anything other than above

Recommended course texts

- Berend Smit and Daan Frenkel, *Understanding Molecular Simulation (2nd edition)*, Academic Press (2001).

- Andrew R. Leach, *Molecular Modelling: Principles and Applications (2nd edition)*, Prentice-Hall (2001).

- Mark Tuckerman, *Statistical Mechanics: Theory and Molecular Simulation*, Oxford University Press (2010)



Coursework and logistics

- readings (programming and software tutorials)
- 4 simulation exercises (50% of grade)
- final project (50% of grade)

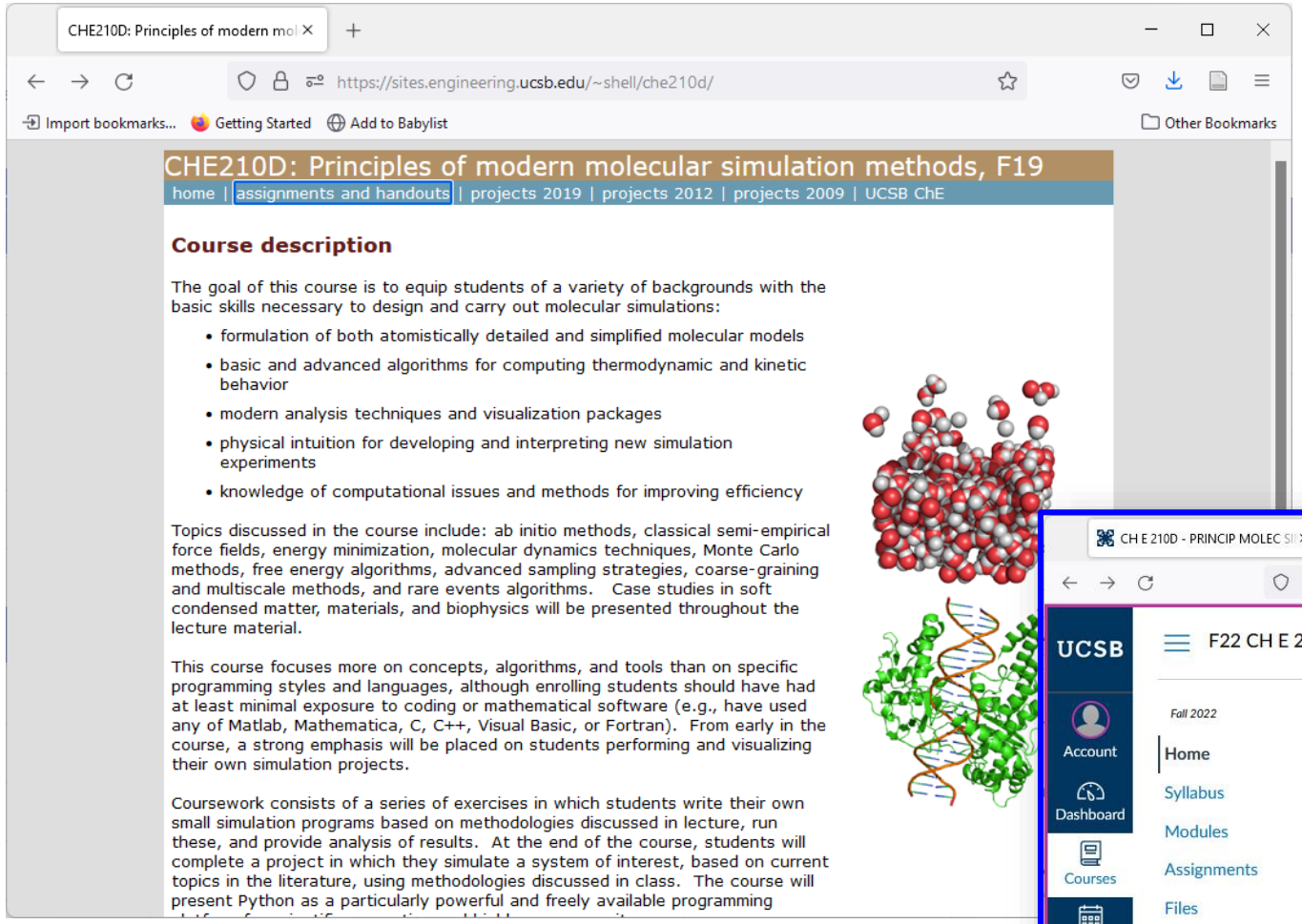
- **For Wednesday 10/5:**
  - Python and NumPy / SciPy reading
  - exercise #1

- Office hours TBA

A word of advice

- Nothing in this course is computation intense…
  …BUT simulations take time to complete!
- Debugging multiplies the simulation time!
- Don’t start last minute…. you’ll be throttled by the run times.
- Use ECI or CNSI CSC resources if your at-home computing is sluggish.

Course website

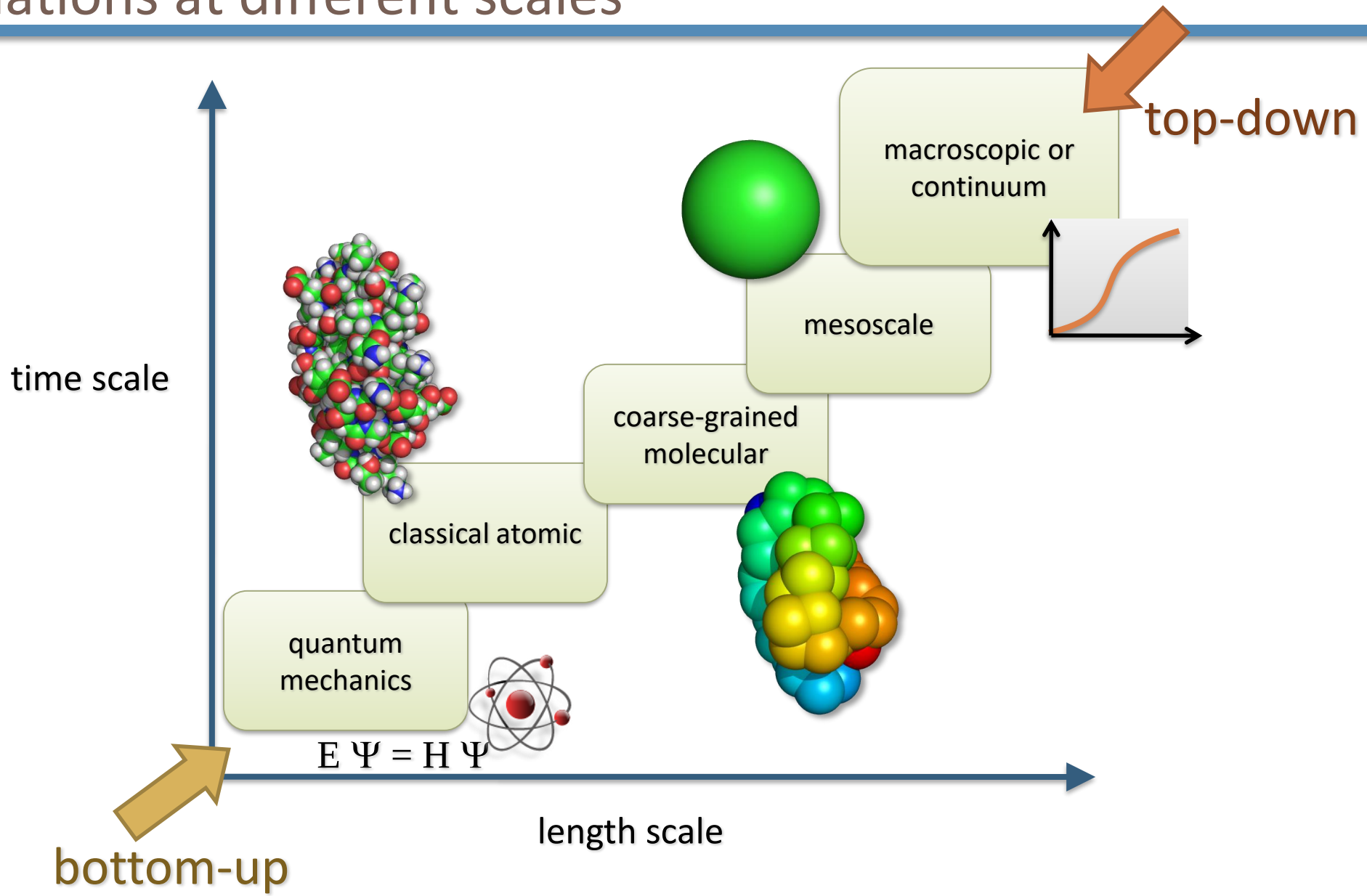


NOT THIS → OLDER SITE BUT HAS GREAT EXAMPLES/TEMPLATES

www.engr.ucsb.edu/~shell/che210d

MAIN COURSE SITE:
NEW CANVAS PLATFORM FOR COURSE
canvas.ucsb.edu

Simulations at different scales



Topics covered

- Ab initio and electronic structure calculations (brief)
- Classical semi-empirical force fields
- Basic methods for evaluating properties
  - minimization (structures)
  - molecular dynamics (thermo & kinetics)
  - Monte Carlo (thermo)
- Free energy & phase equilibria methods
- Advanced sampling approaches
- Multiscale methods and coarse-grained models

Tools we will use

- Python programming language
- NumPy and SciPy
- Fortran (basics, for numerically intense routines only)
- Visualization software (UCSF Chimera)

There are many great simulation software packages / suites that are great for regular use and “production” runs.

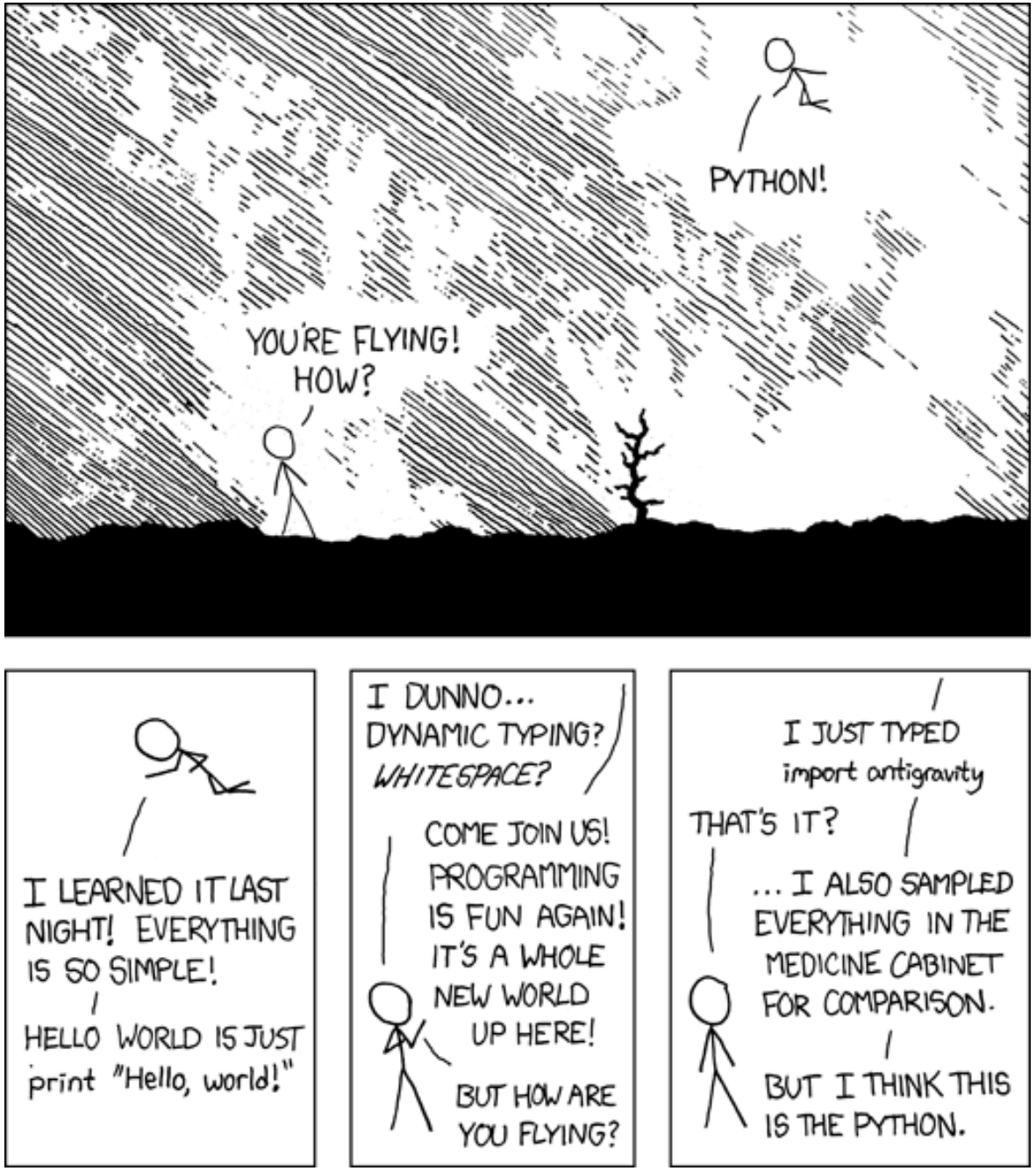
We’re deliberately not using them so that your coding exercises will provide more significant insight into how they work.

Why Python?

- free, open source, cross-platform
- **intuitive, easy to use, highly legible code**
- “batteries included” philosophy
- very popular, enormous community with add-on modules

- Python is now THE main platform for scientific computing and data science in research, even in industry.

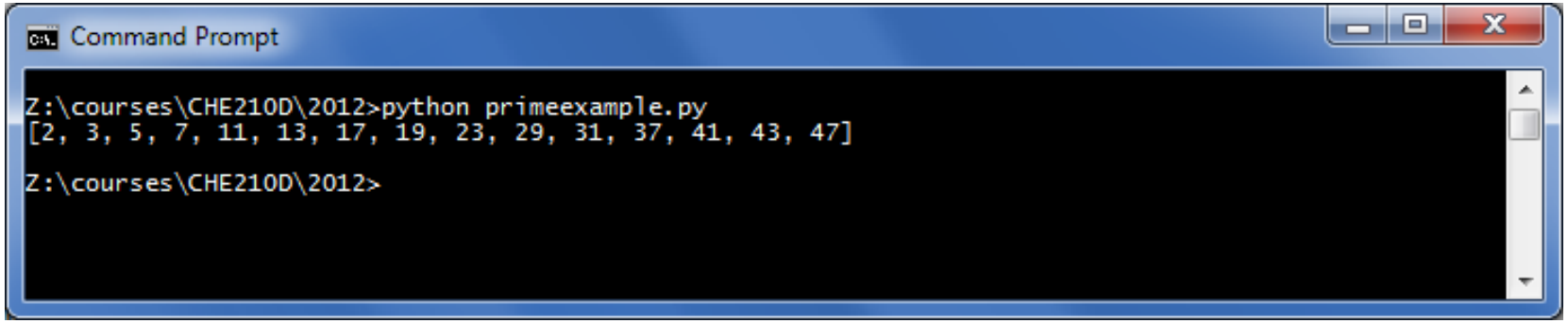
It is much more useful and career relevant for you to know Python than MatLab, despite what oldschoolers tell you.


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

www.xkcd.com

A simple Python program to compute primes

```
1  PrimeList = [2]                          #create a new list with the 1st prime
2  Current = 3                              #start with the number 3
3
4  while Current < 50:                      #examine numbers less than 50
5      IsPrime = True                       #assume a prime
6      for Prime in PrimeList:              #see if Current is divisible by any prime
7          if Current % Prime == 0:         #check the modulo
8              IsPrime = False              #it is divisible -- not a prime
9              break                        #break out of the for loop
10     if IsPrime:                          #check if we found a prime
11         PrimeList.append(Current)        #it is a prime; add to list
12     Current += 1                         #increment current
13
14 print( PrimeList )                       #print out the list
15
```



Python + NumPy + SciPy

- **NumPy** – VERY fast linear algebra and array routines, random number generation, other common numerical computations

- **SciPy** – comprehensive and very fast mathematical package with more compled algorithms for e.g.: integration, optimization, interpolation, Fourier transforms & signal processing, linear algebra, statistics

- Open source Python + NumPy + SciPy (+ community packages) is now standard and MUCH more powerful than commercial packages like MatLab

Why Python + Fortran?

- Python alone is slow for raw numerics.
- Fortran is arguably the fastest numeric programming language, and widely used in the back end of simulation packages.
- Accelerator packages (e.g., Cython, Numba) aren't nearly as fast as Fortran.
- Much existing code in the scientific community is in Fortran
- Fortran is super simple to learn and incorporate into python.

- Bottleneck routines written Fortran can be imported *transparently* into Python, almost magically

numerical.recipes/

numerical.recipes

NUMERICAL RECIPES™

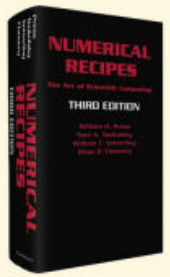
The Art of Scientific Computing

Third Edition

Important note: We have changed our web address. We are no longer "nr.com". We are now "numerical.recipes" (without any .com). Please change your bookmarks. We are the same enterprise and look forward to continuing to serve your Numerical Recipes needs into the future.

E-book readers: Our new, easy-to-remember e-book URLs are numerical.recipes/book for individual subscribers, and numerical.recipes/corporate for corporate and institutional users.

Click on any image below to display in the right column more information about the product or service.



The book.

The on-line electronic book.

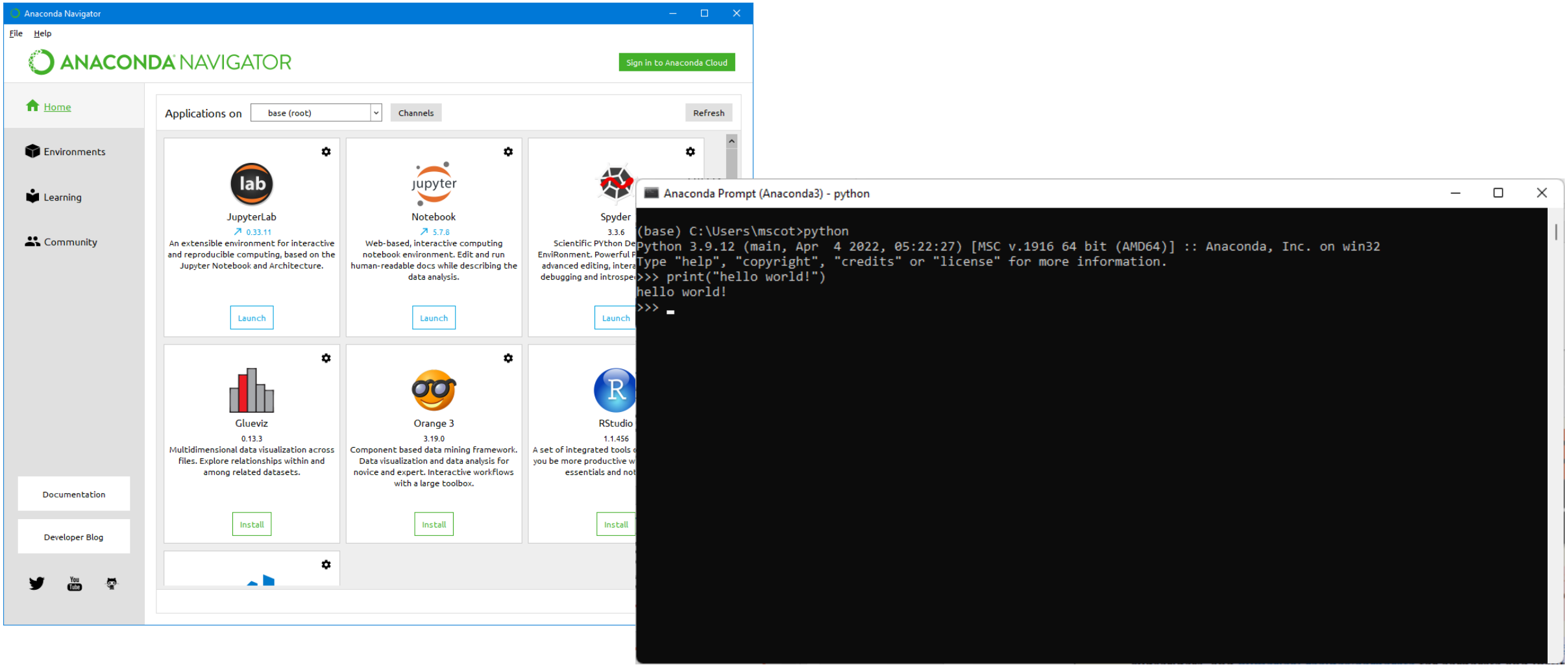
Numerical Recipes Home Page

We are **numerical.recipes**, Numerical Recipes Software. We are one of the oldest continuously operating sites on the Web, with the historic former domain **nr.com** dating back to 1993, one of the first 25,000 domains in the Internet. (Today, that number is about 200,000,000.) In partnership with Cambridge University Press, we develop the *Numerical Recipes* series of books on scientific computing and related software products.

News!

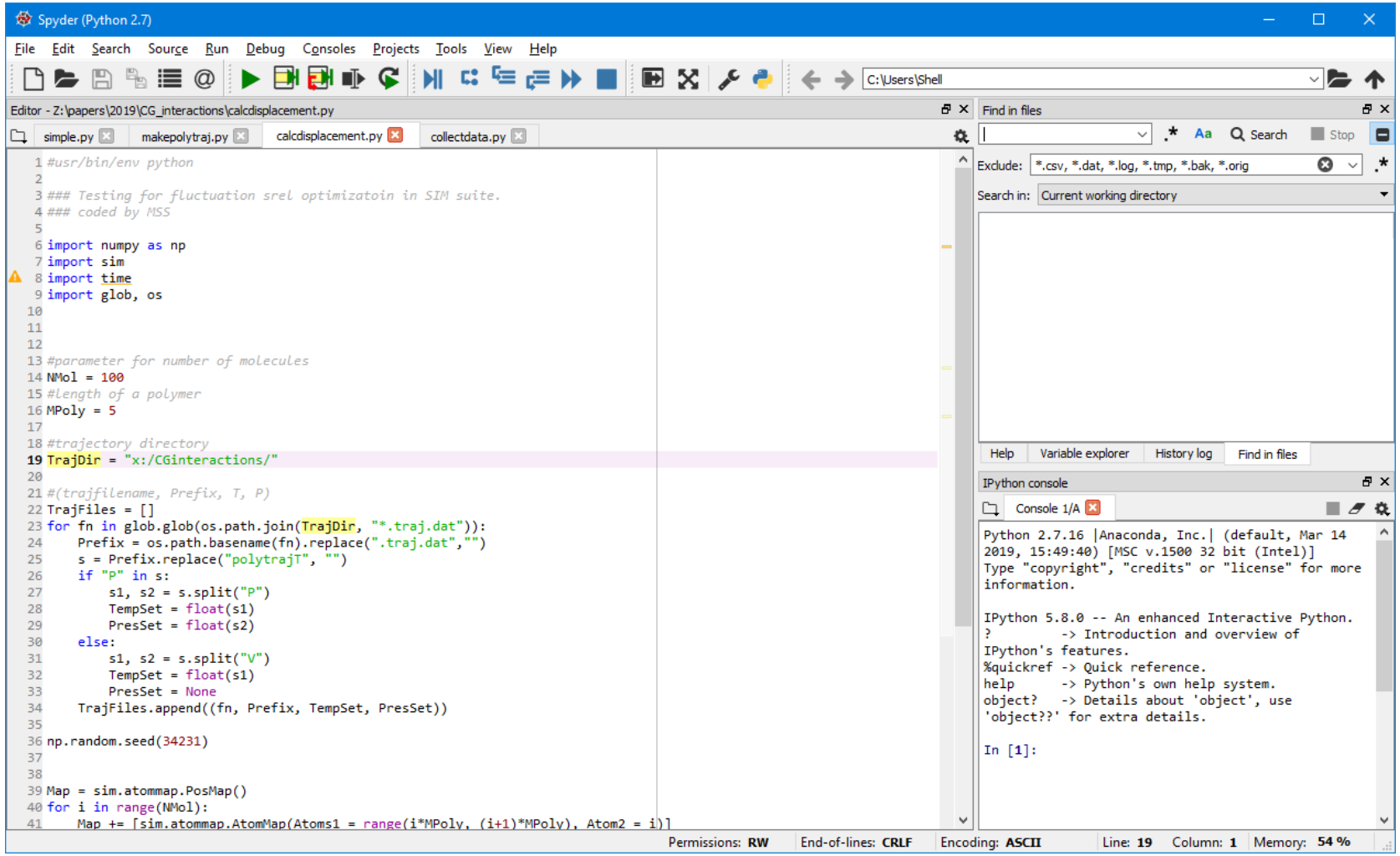
On 12/31/2020, Adobe Inc. inactivated Adobe Flash in all

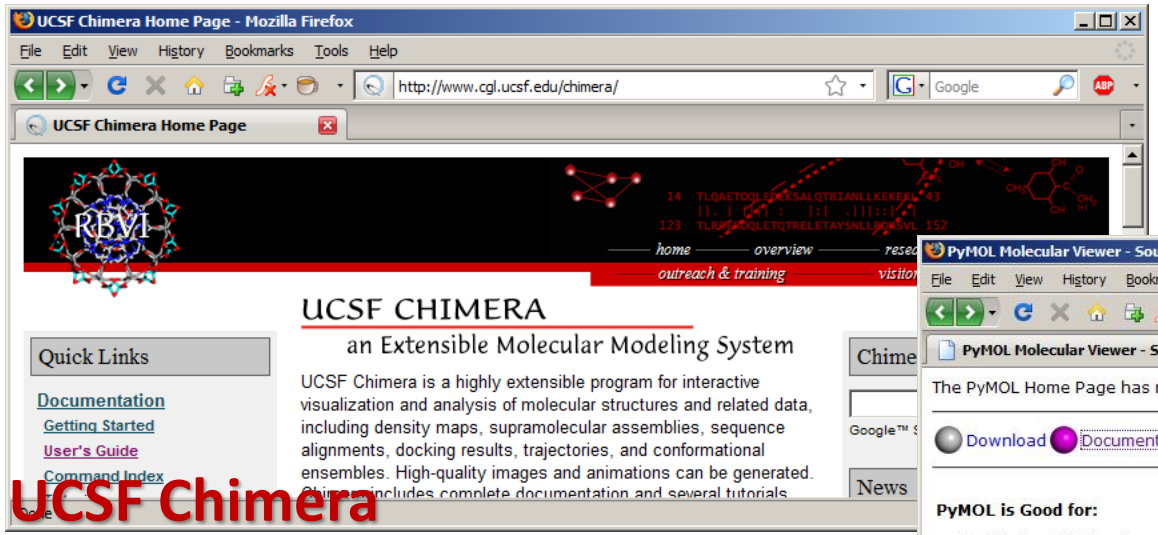
Anaconda Python distribution



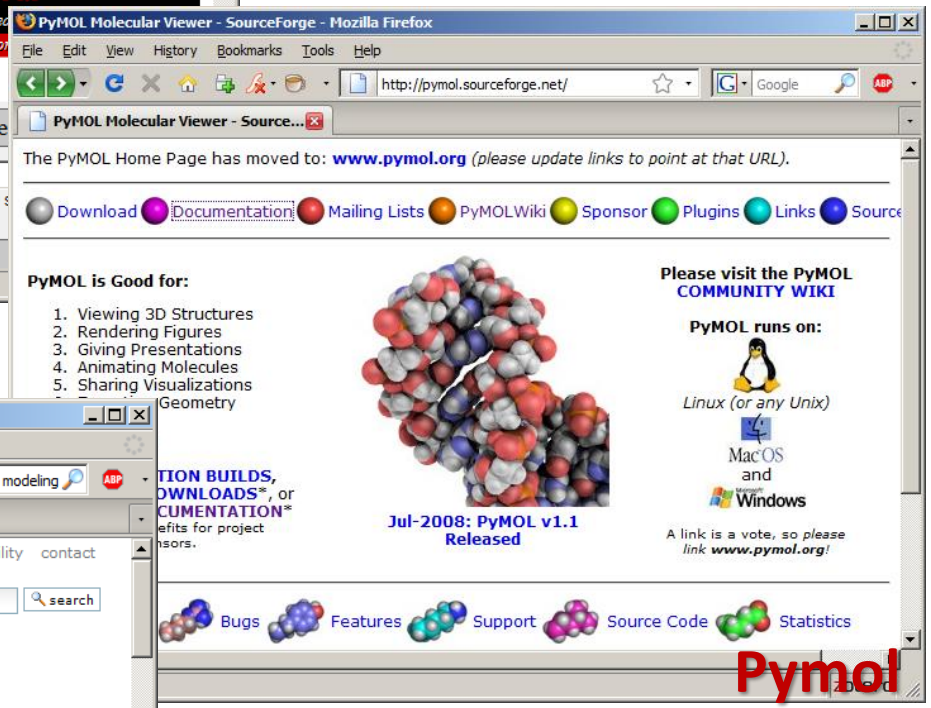
https://www.anaconda.com/distribution/

Spyder Python editor (installed with Anaconda)

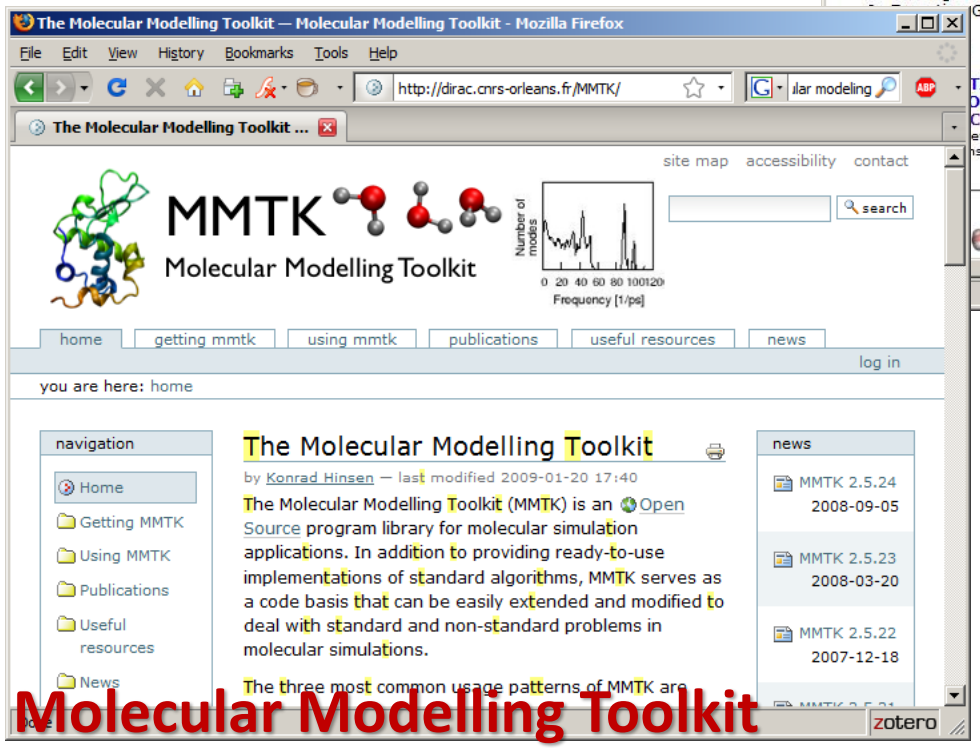




UCSF Chimera



Pymol

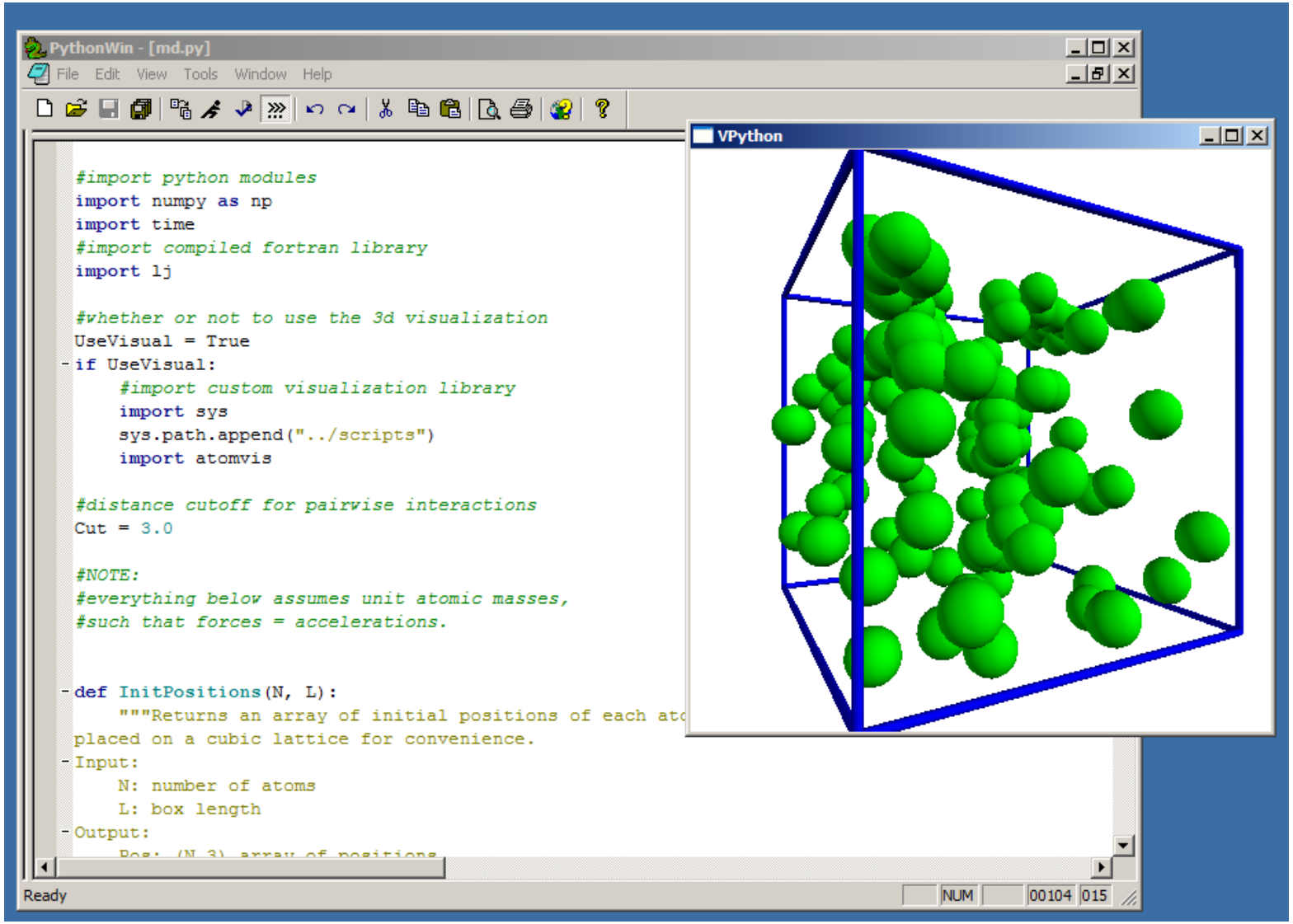


Molecular Modelling Toolkit

Can I do this?

- No difference between learning a programming language and learning equipment software
- Molecular simulation codes are generally not complex software
- Many examples / tools / templates available online
- Challenge is not so much *how* to simulate,
  but *what* to simulate and *what & how* to analyze

Example

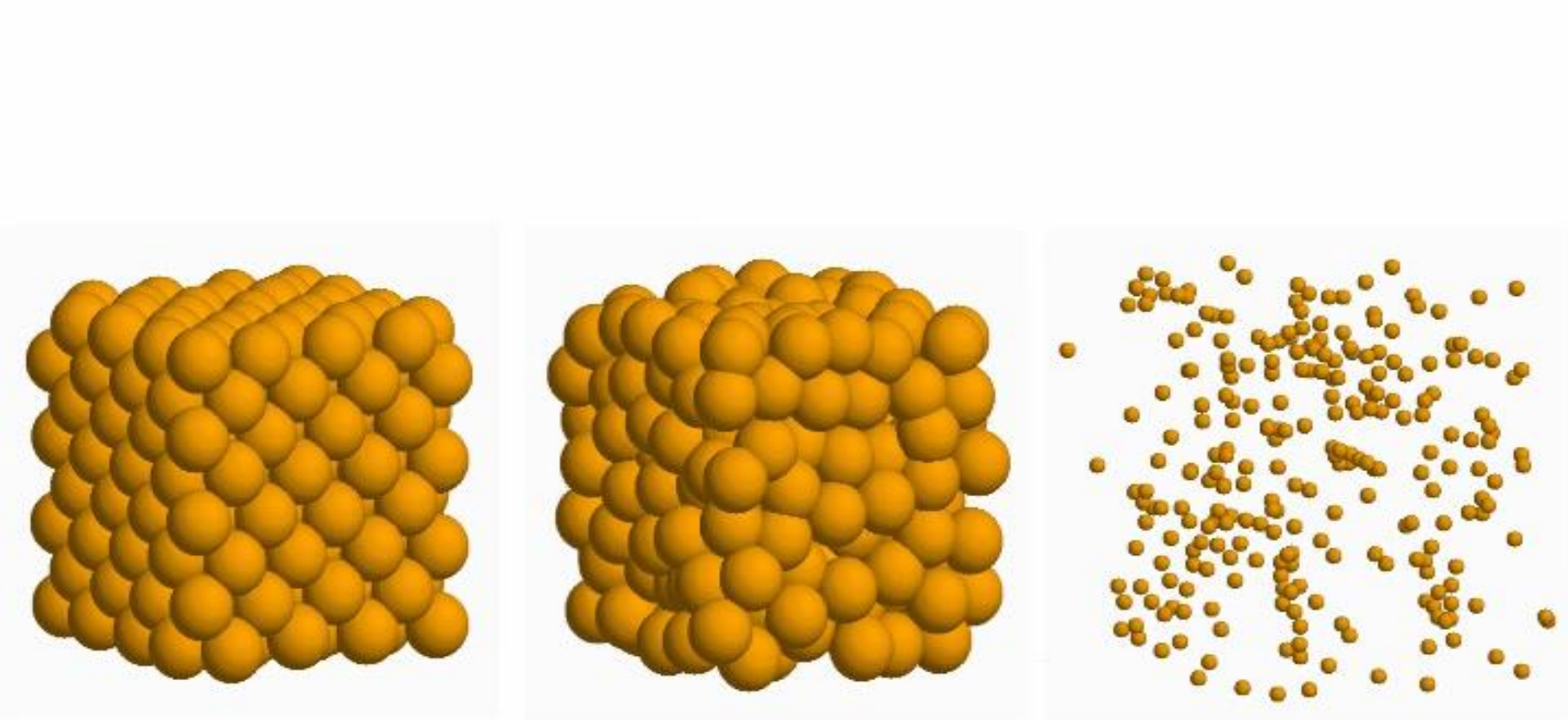

PythonWin - [md.py]
File Edit View Tools Window Help
#import python modules
import numpy as np
import time
#import compiled fortran library
import lj
#whether or not to use the 3d visualization
UseVisual = True
if UseVisual:
#import custom visualization library
import sys
sys.path.append("../scripts")
import atomvis
#distance cutoff for pairwise interactions
Cut = 3.0
#NOTE:
#everything below assumes unit atomic masses,
#such that forces = accelerations.
def InitPositions(N, L):
"""Returns an array of initial positions of each at
placed on a cubic lattice for convenience.
Input:
N: number of atoms
L: box length
Output:
Ready
NUM
00104
015
VPython

What's it all good for?

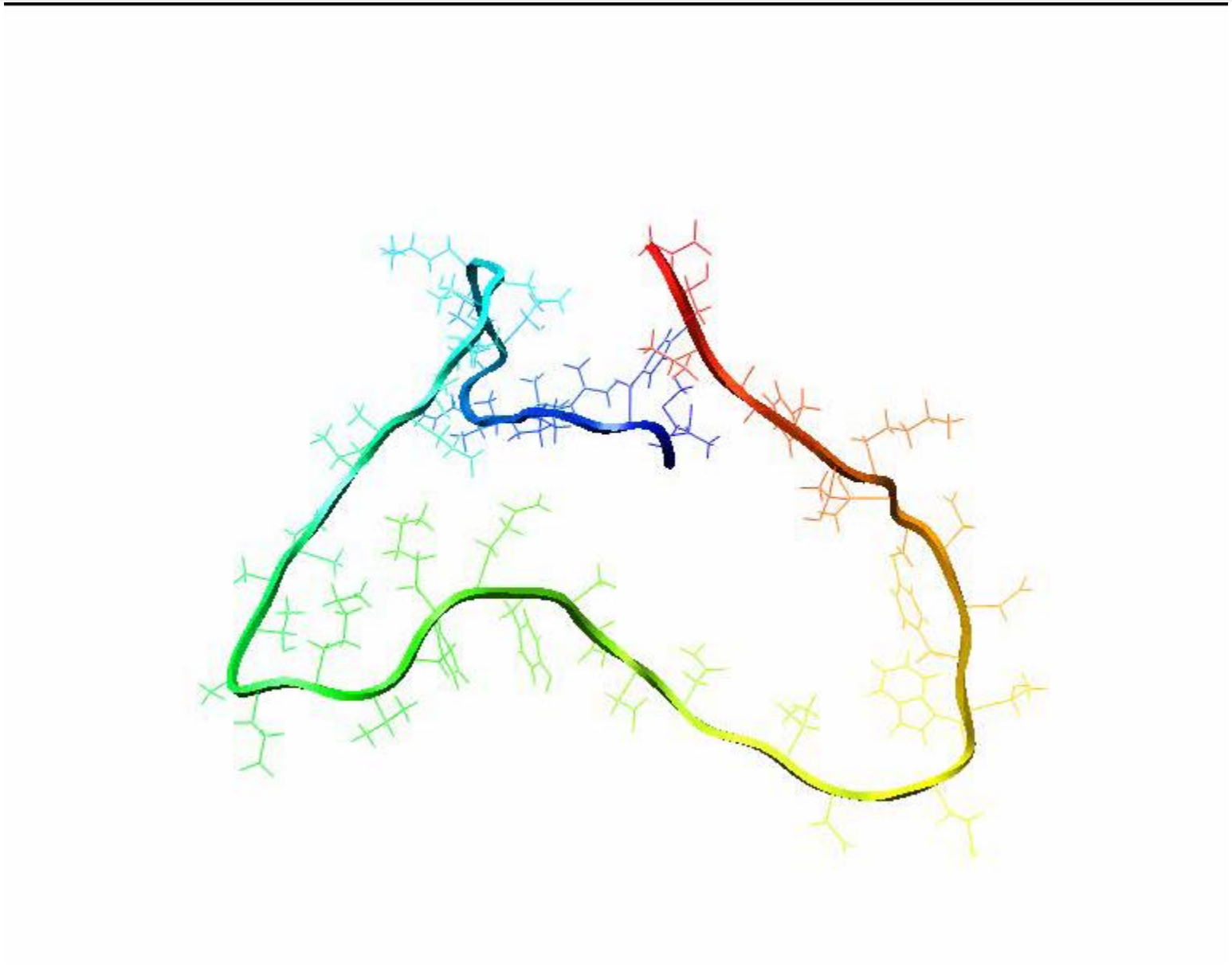
- **Qualitative** frameworks for thinking about molecular processes and mechanisms
- **Quantitative** understanding of different molecular driving forces
- **Prediction** of properties or molecular architectures for engineering design

Some examples…

Multiple phases of a simple substance: argon

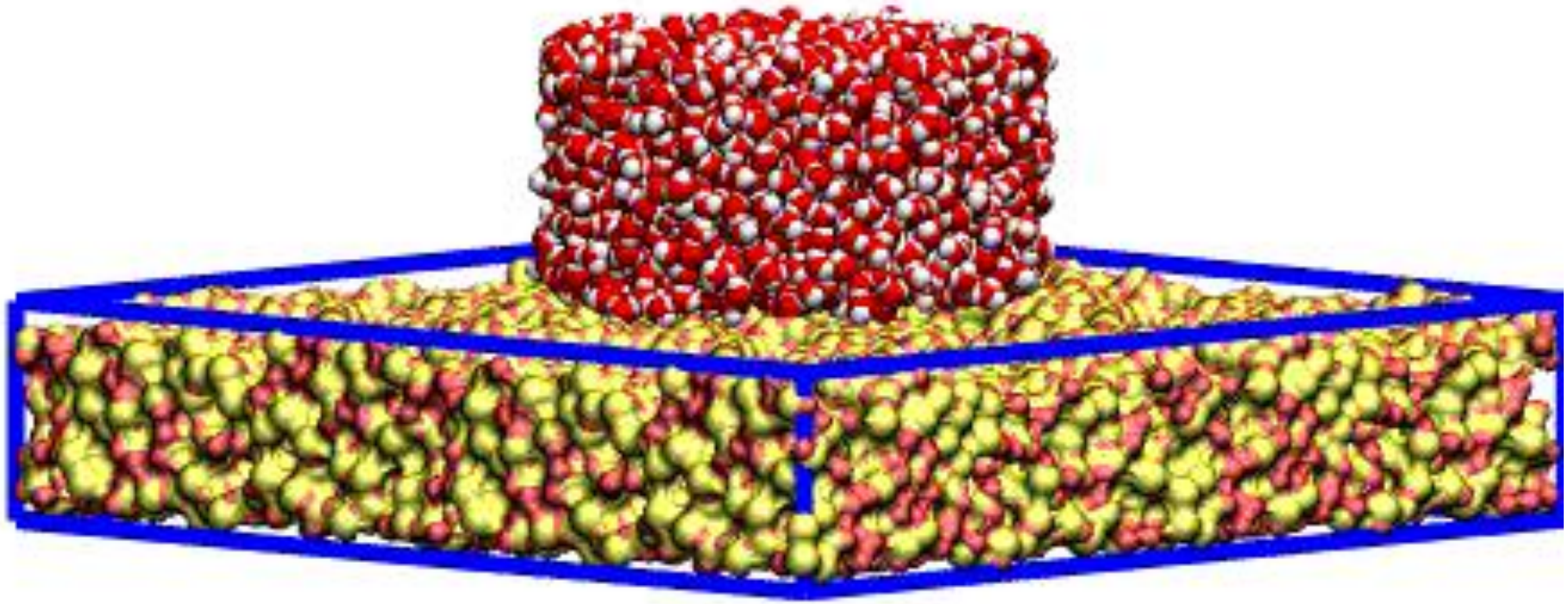


A more complex molecule: a protein



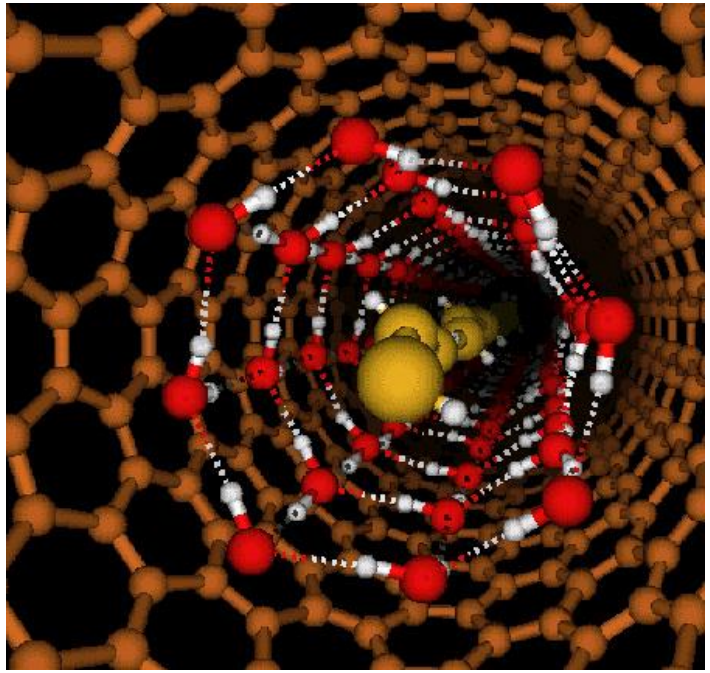
dramatization

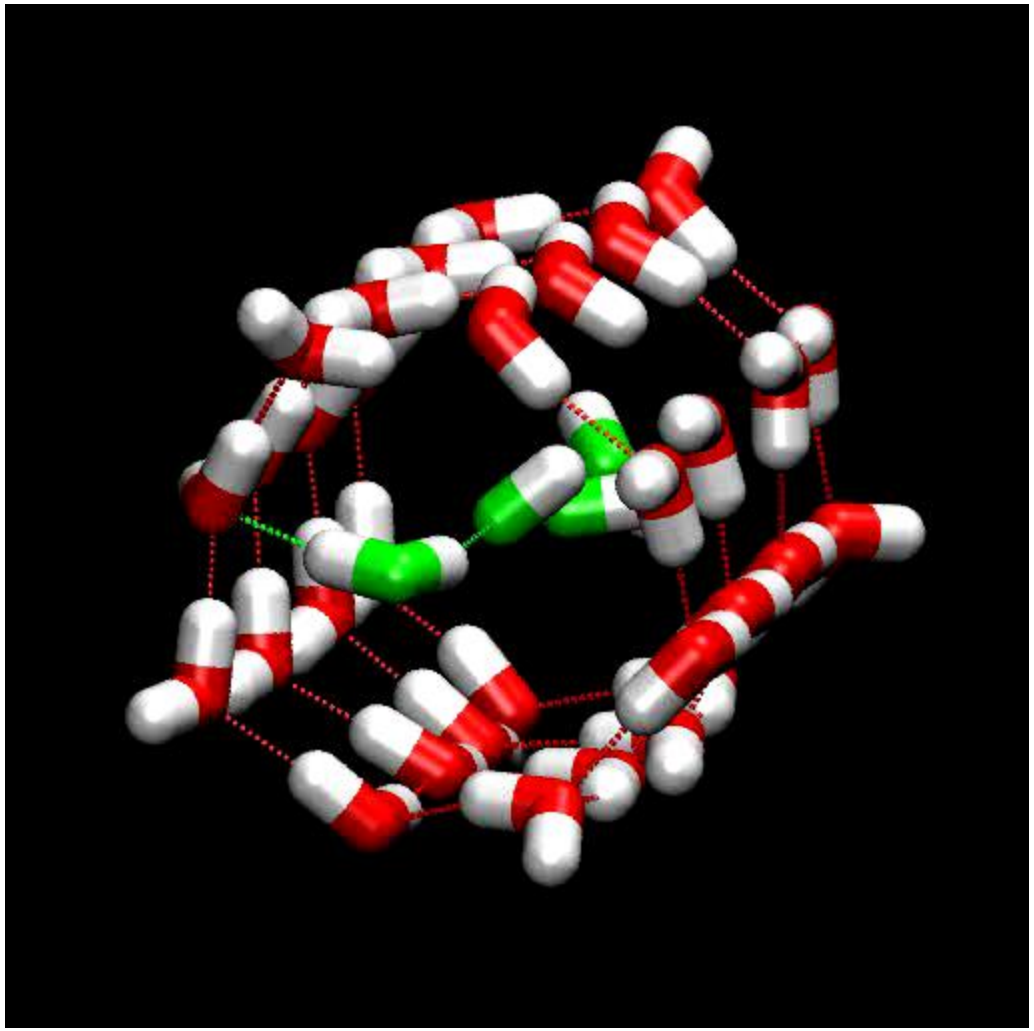
A water nanodroplet on a silica surface



simulation by E. R. Cruz-Chu, A. Aksimentiev , and K. Schulten
movie from http://www.ks.uiuc.edu/Gallery/Movies/

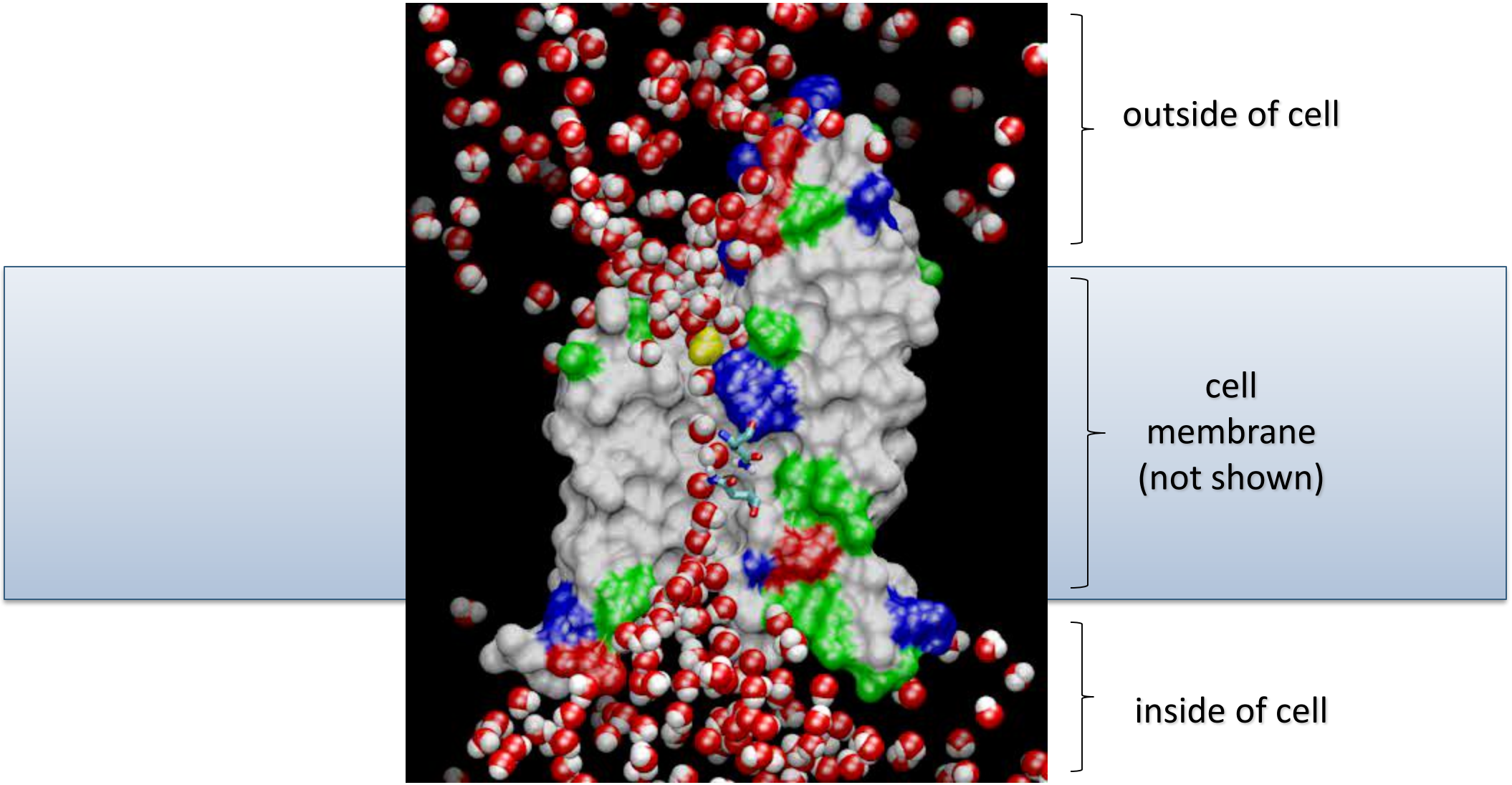
Water transport inside a carbon nanotube





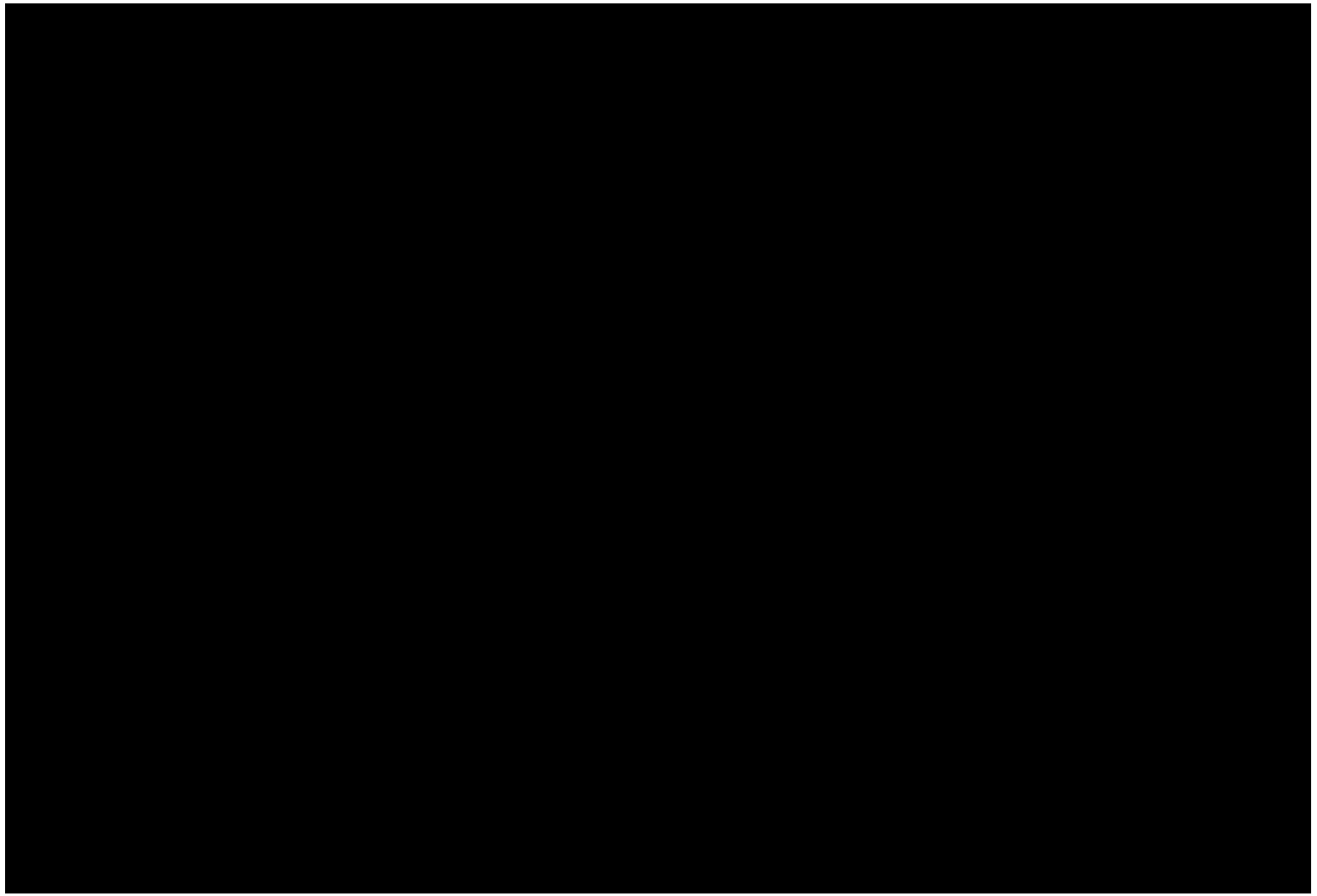
simulation by A. Kolesnikov and coworkers
movie from http://www.anl.gov/Media_Center/News/2005/IPNS050513.html

Water transport through a protein channel



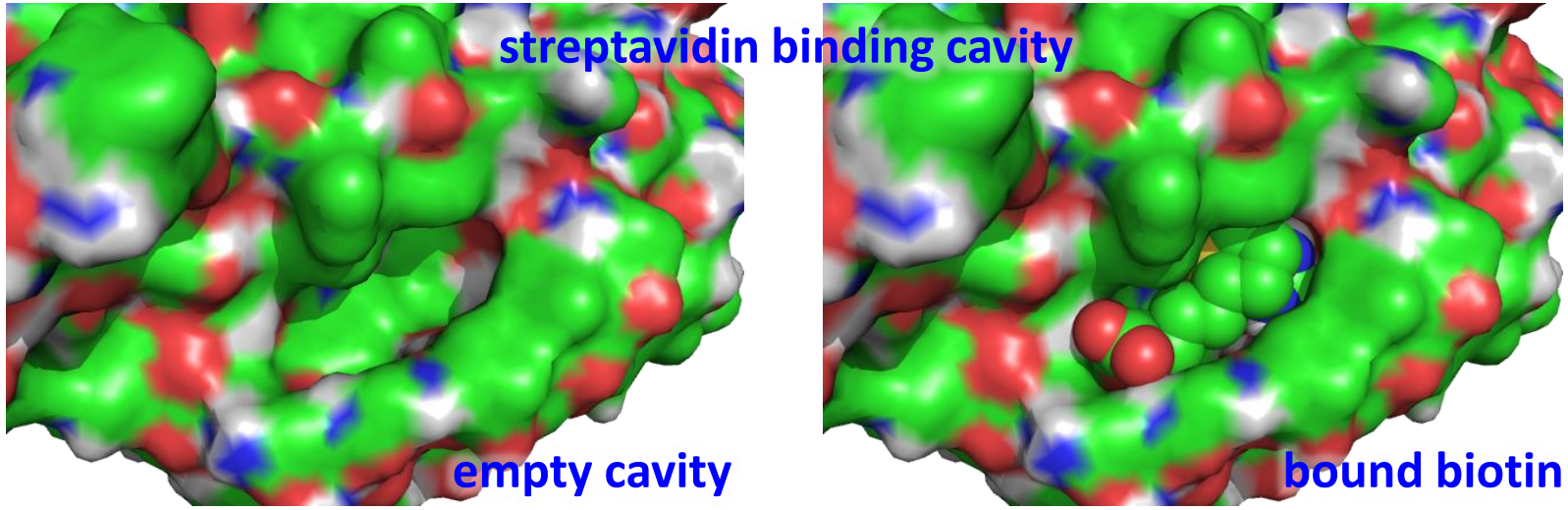
simulation by E. Tajkhorshid, K. Schulten, Y. Wang, J. Yu, F. Zhu, and M. Jensen
movie from http://www.ks.uiuc.edu/Gallery/Movies/

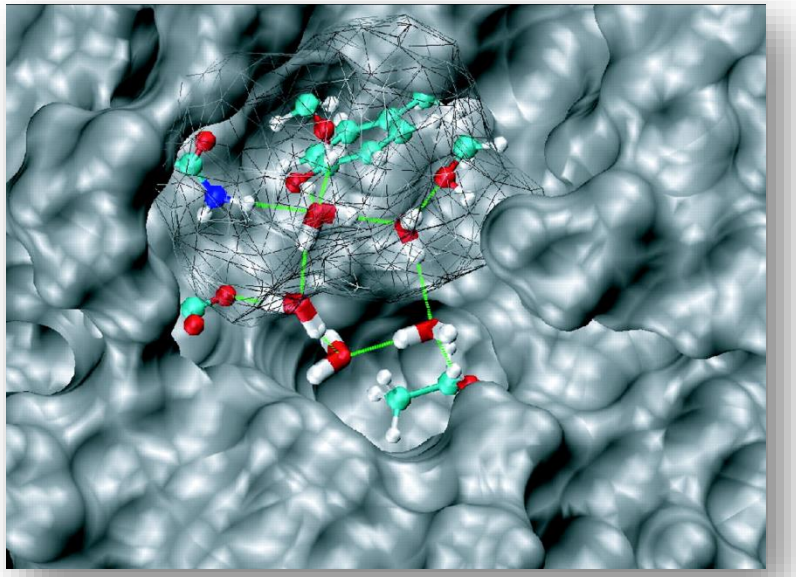
Phase separation and equilibria



simulation by A. Delapaz and L. Gelb
movie from http://www.chemistry.wustl.edu/~gelb/gchem/materials/lve/index.html

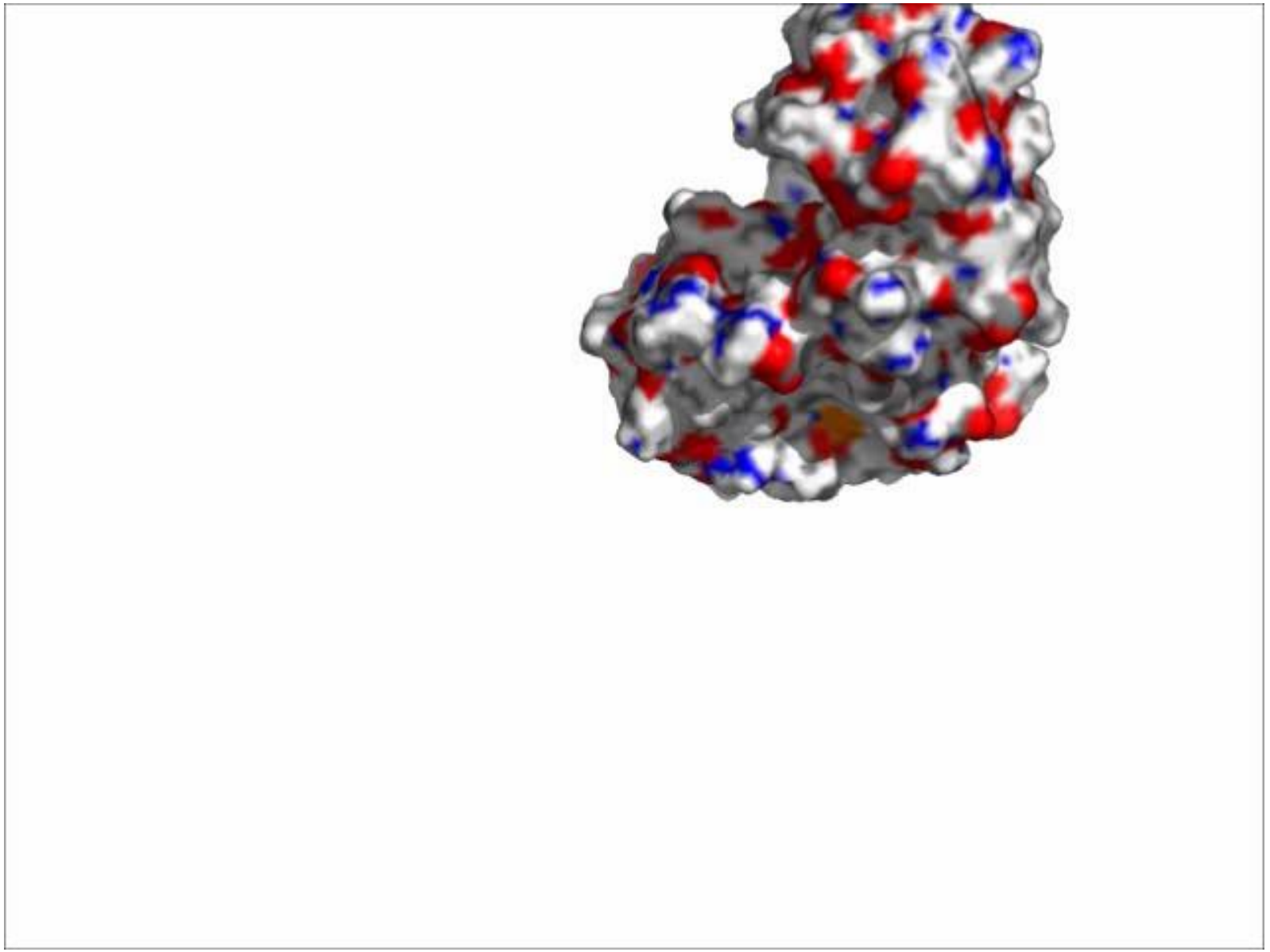
Driving forces in small-molecule binding





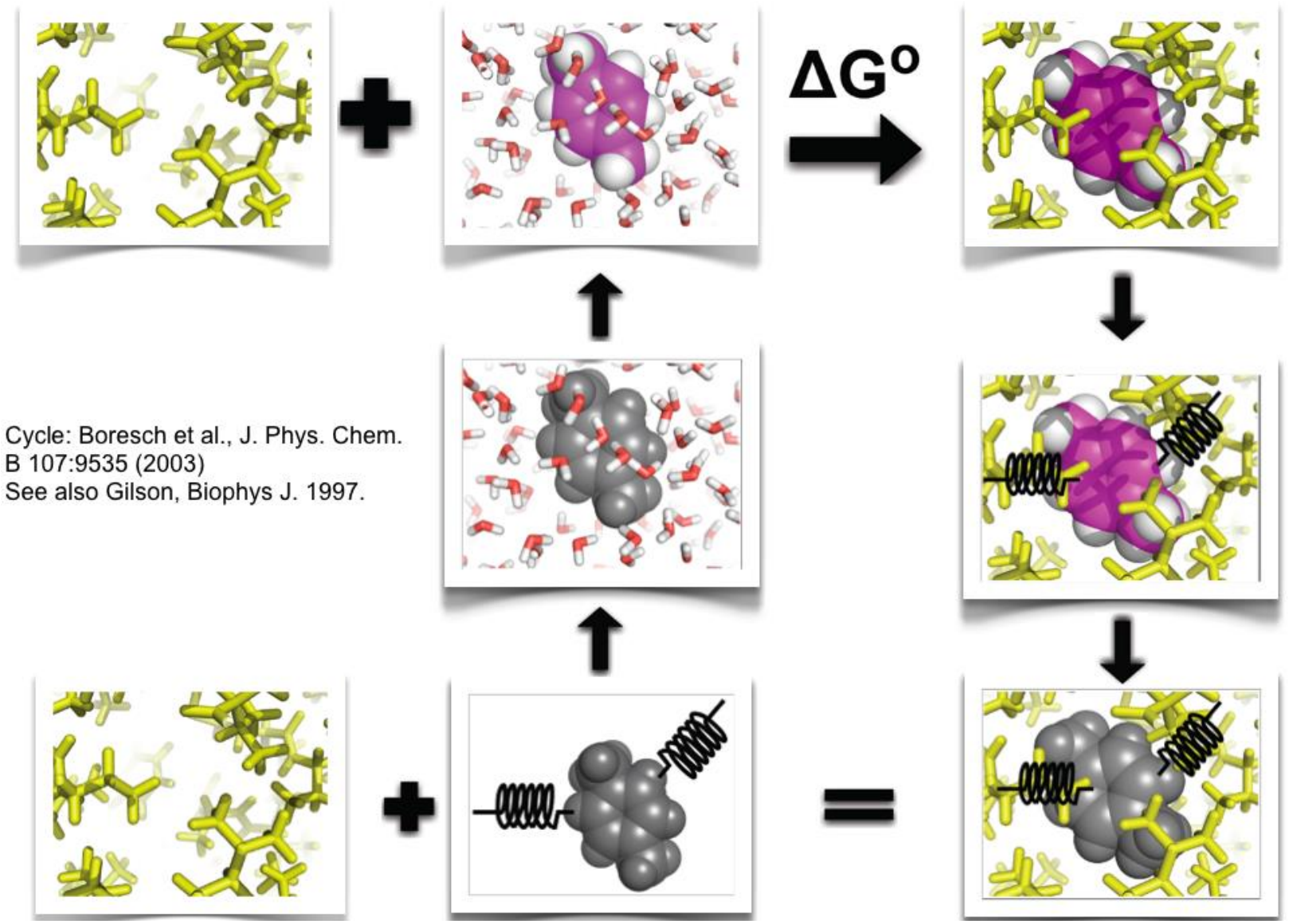
Young, et al., PNAS, 2007

Solvation and binding free energies



simulation by D. Mobley

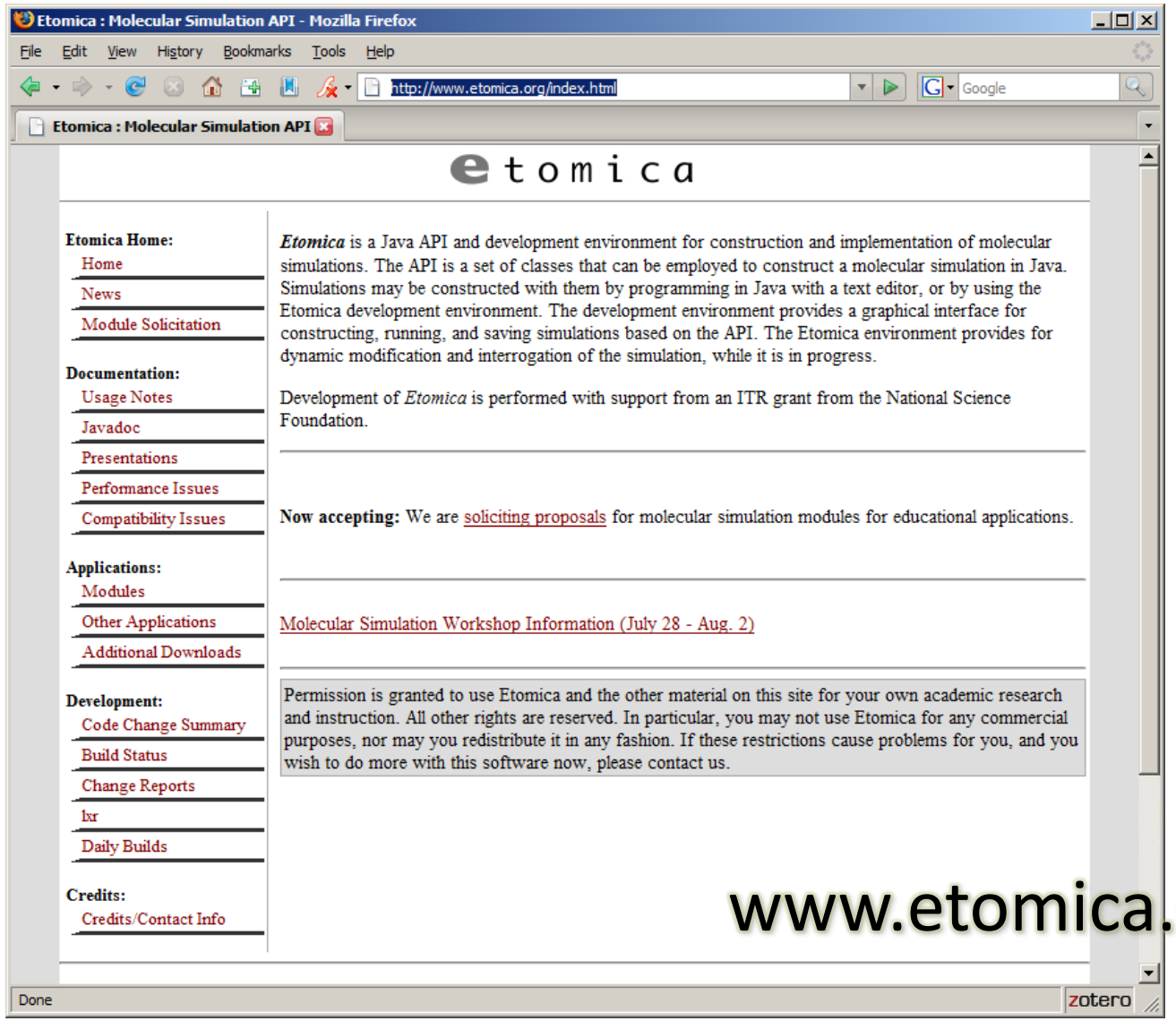
Artificial thermodynamic cycles for binding



Cycle: Boresch et al., J. Phys. Chem. B 107:9535 (2003)
See also Gilson, Biophys J. 1997.

figure from D. Mobley

Try out some interactive simulations yourself



www.etomica.org

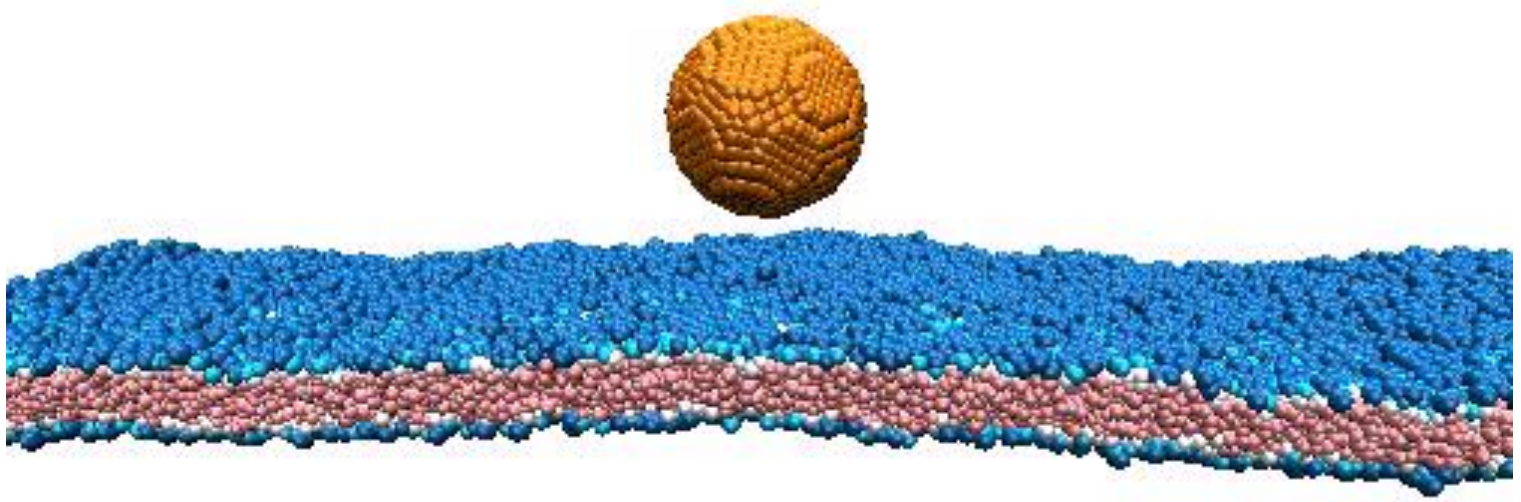
Some examples from our group...

Nanoparticle interactions
with model cellular membranes
for drug delivery and toxicology.


Gary Leal

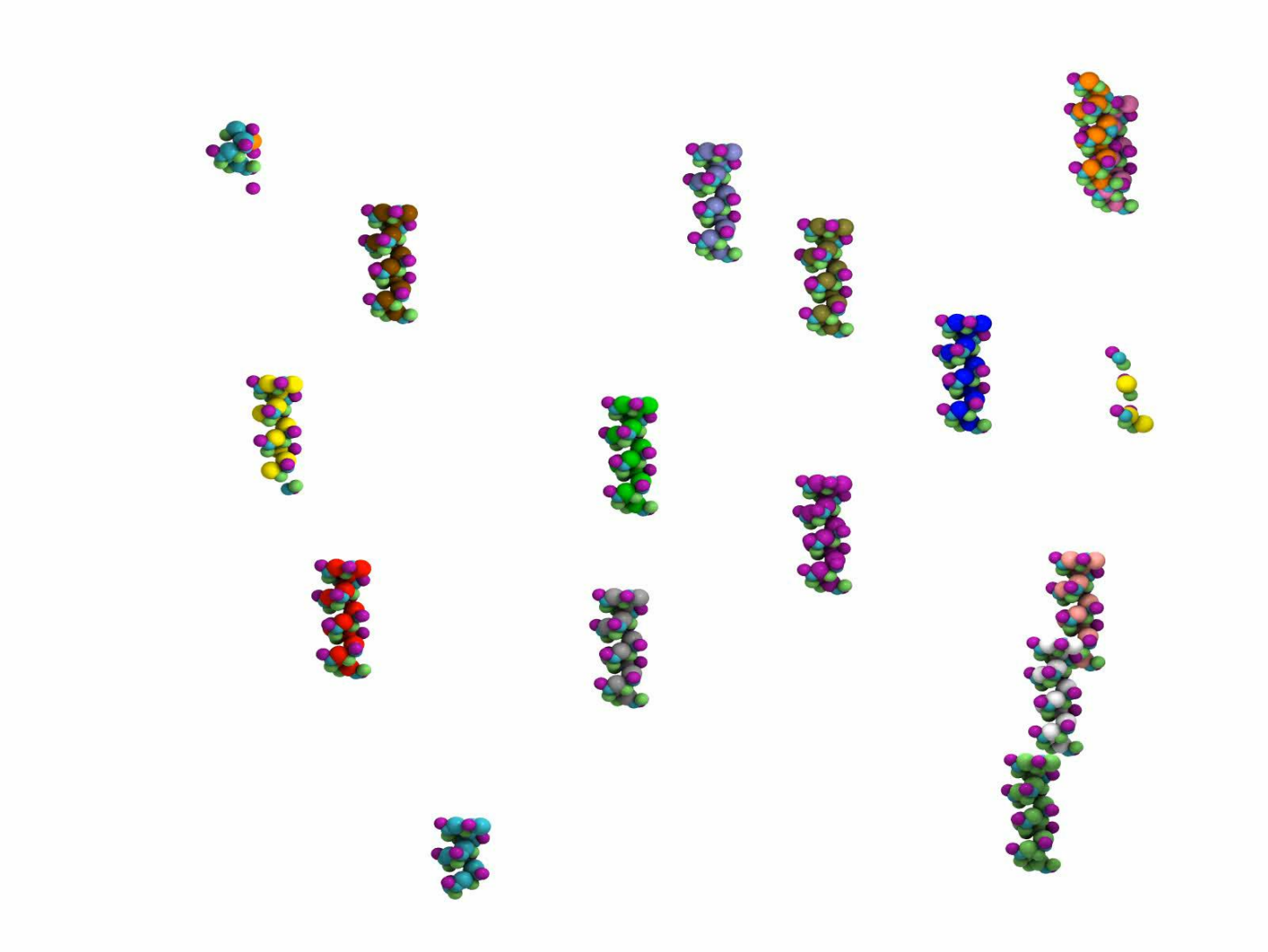

Samir Mitragotri


Dave Smith



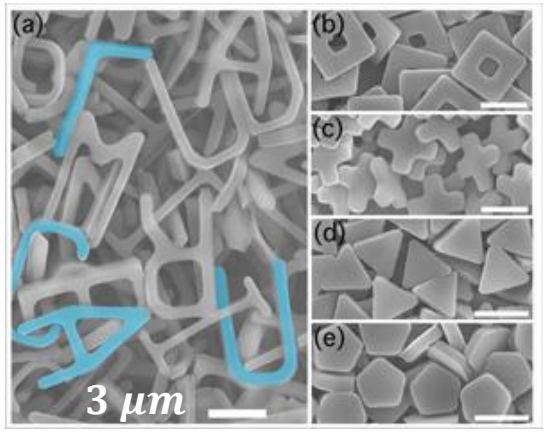
Peptide self-assembly and aggregation
in biomaterials and neurodegenerative disease.

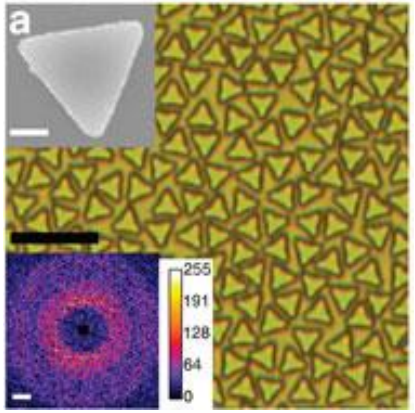
Scott Carmichael



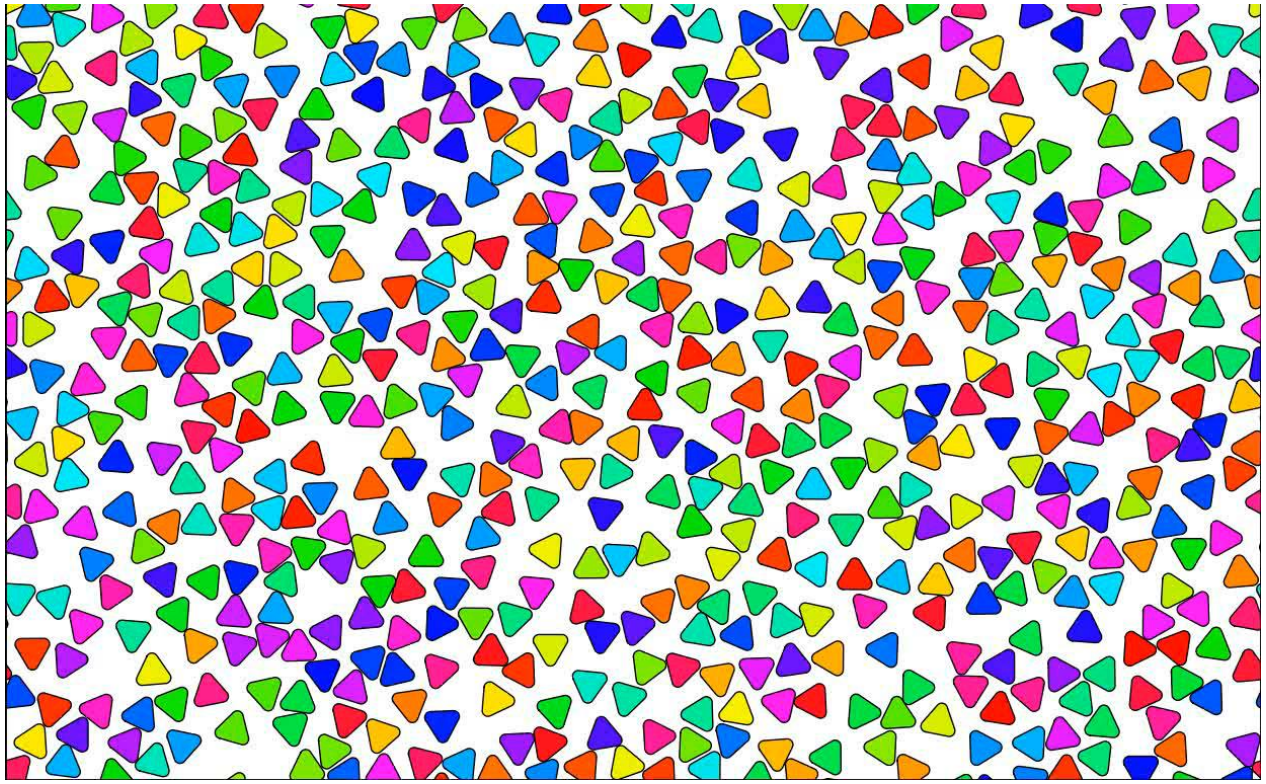
Structured material surfaces through quasi-2D colloidal particle assembly.


Scott Carmichael





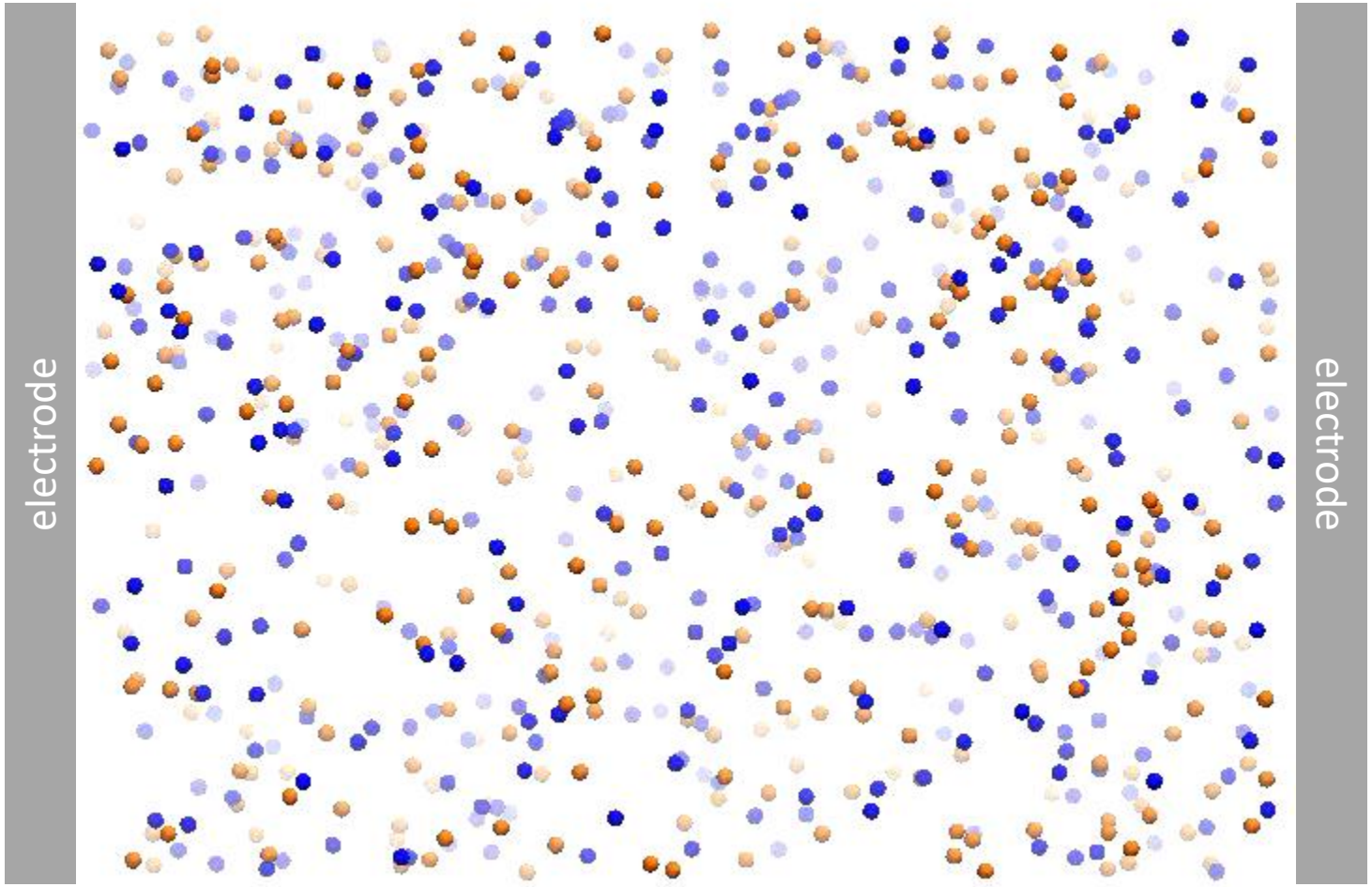
Hernandez and Mason, J. Phys. Chem. C (2007); Zhao, Bruinsma, and Mason, Nature Comm. (2012)



Energy storage through electric double layer supercapacitors.


Todd Squires


Brian Giera



Some early milestones in molecular simulation

- 1953: Monte Carlo method applied to hard spheres (Metropolis, Rosenbluth, Rosenbluth, Teller & Teller)
- 1954: perturbation approach to free energies (Zwanwig)
- 1956: molecular dynamics of hard spheres (Alder and Wainwright)
- 1963: computation of the chemical potential (Widom)
- 1964: molecular dynamics of liquid argon (Rahman)
- 1971: molecular dynamics of liquid water (Rahman & Stillinger)

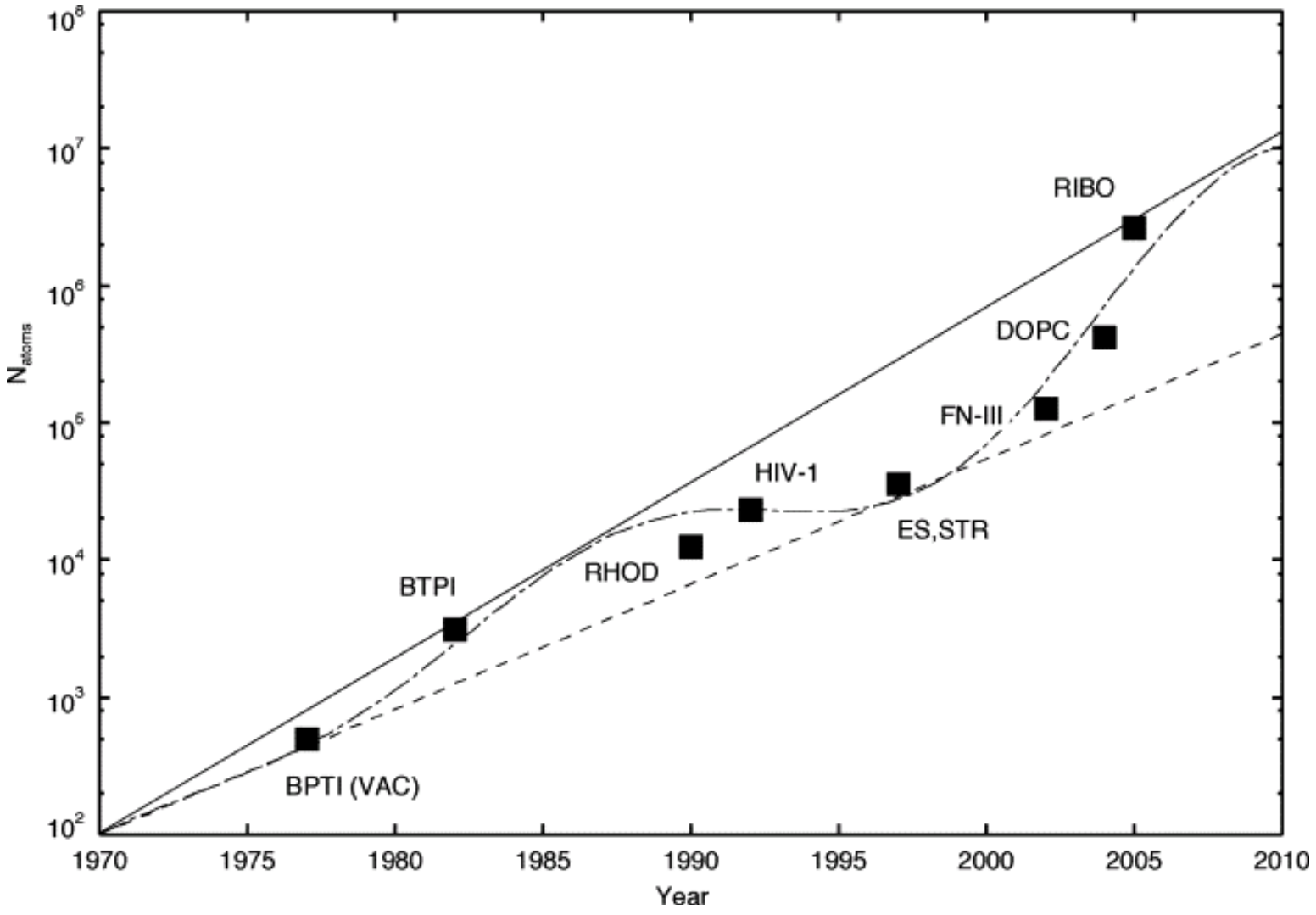
Advances in models and algorithms

- 1976: optimal estimates of free energy differences (Bennett)
- 1976: first simulation of protein dynamics (McCammon et al.)
- 1977: non-Boltzmann sampling and artificial ensembles (Torrie & Valleau)
- early 1980s: community-developed transferable classical potential models and software suites (CHARMM, AMBER)
- 1987-1995: robust & rigorous techniques for predicting phase equilibria (Panagiotopoulos, Wilding, Kofke)
- 1989, 1992: generalized, optimal techniques for extracting free energy estimates (Ferrenberg, Swendsen, et al)

Recent accomplishments

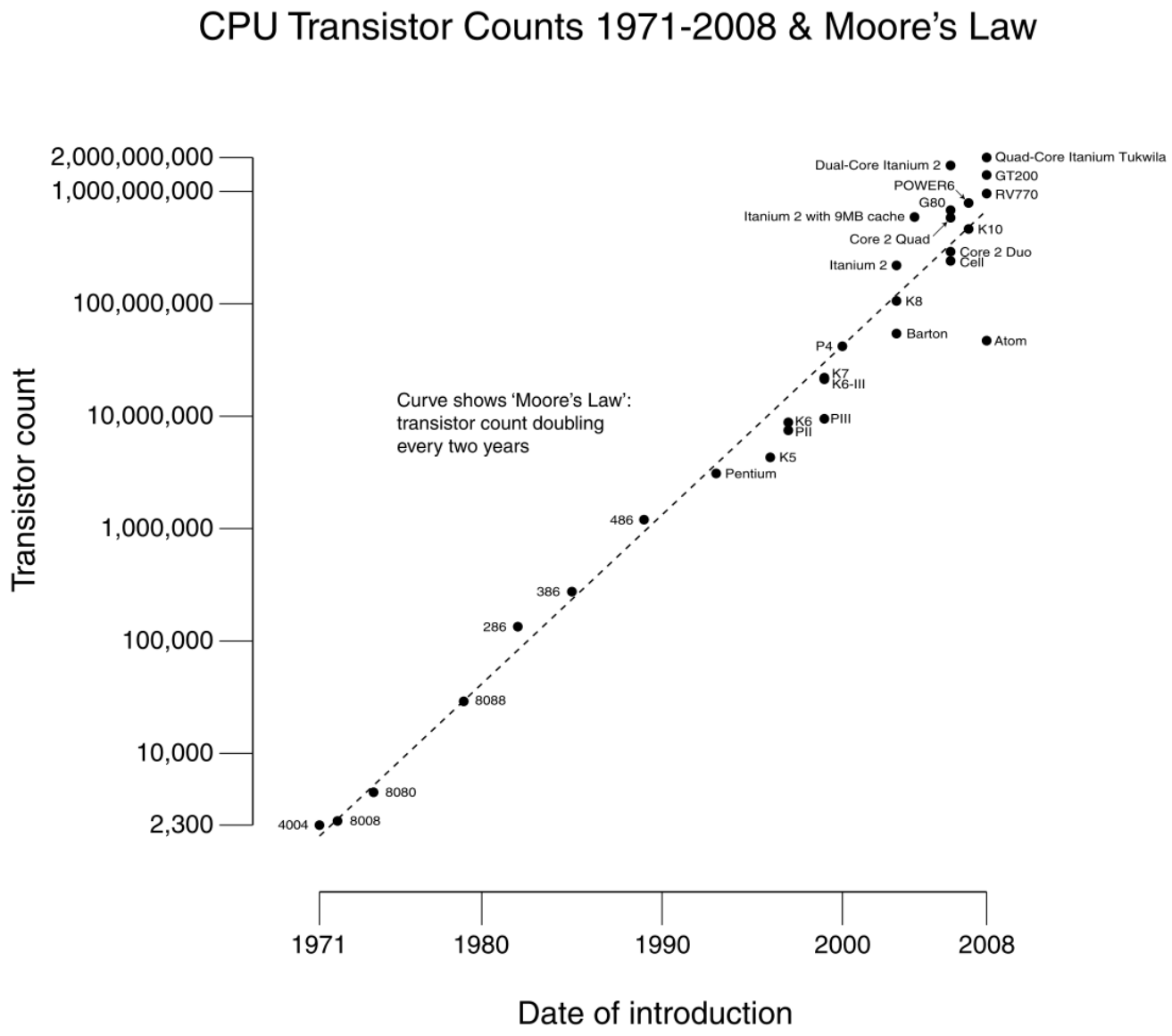
- 1997-1999: theory for equilibrium properties from nonequilibrium measurements (Jarzynski, Crooks)
- 1998: 1 μs simulation of miniprotein folding (Duan and Kollman)
- 1999-2002: generalized and extended ensemble methods (Sugita & Okamoto, Wang & Landau)
- 2002: water freezing from 6 μs simulation (Matsumoto et al.)
- 2002-2003: massive distributed computing for small protein folding (Folding@Home, Pande et al.)
- 2004: design of an entirely new protein fold (Baker et al.)
- 2010 - present: emergence of GPU-based molecular dynamics

System size versus time

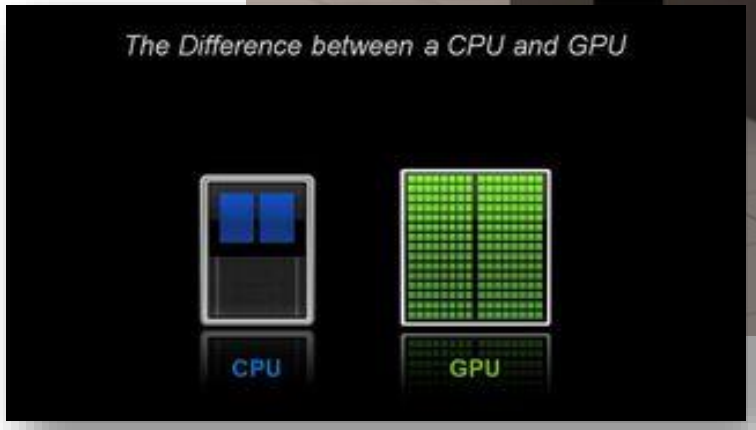


K.Y. Sanbonmatsu and C.S. Tung, "Performance computing in biology: Multimillion atom simulations of nanoscale systems," J. Structural Biology, 157, 470 (2007)

Moore’s law

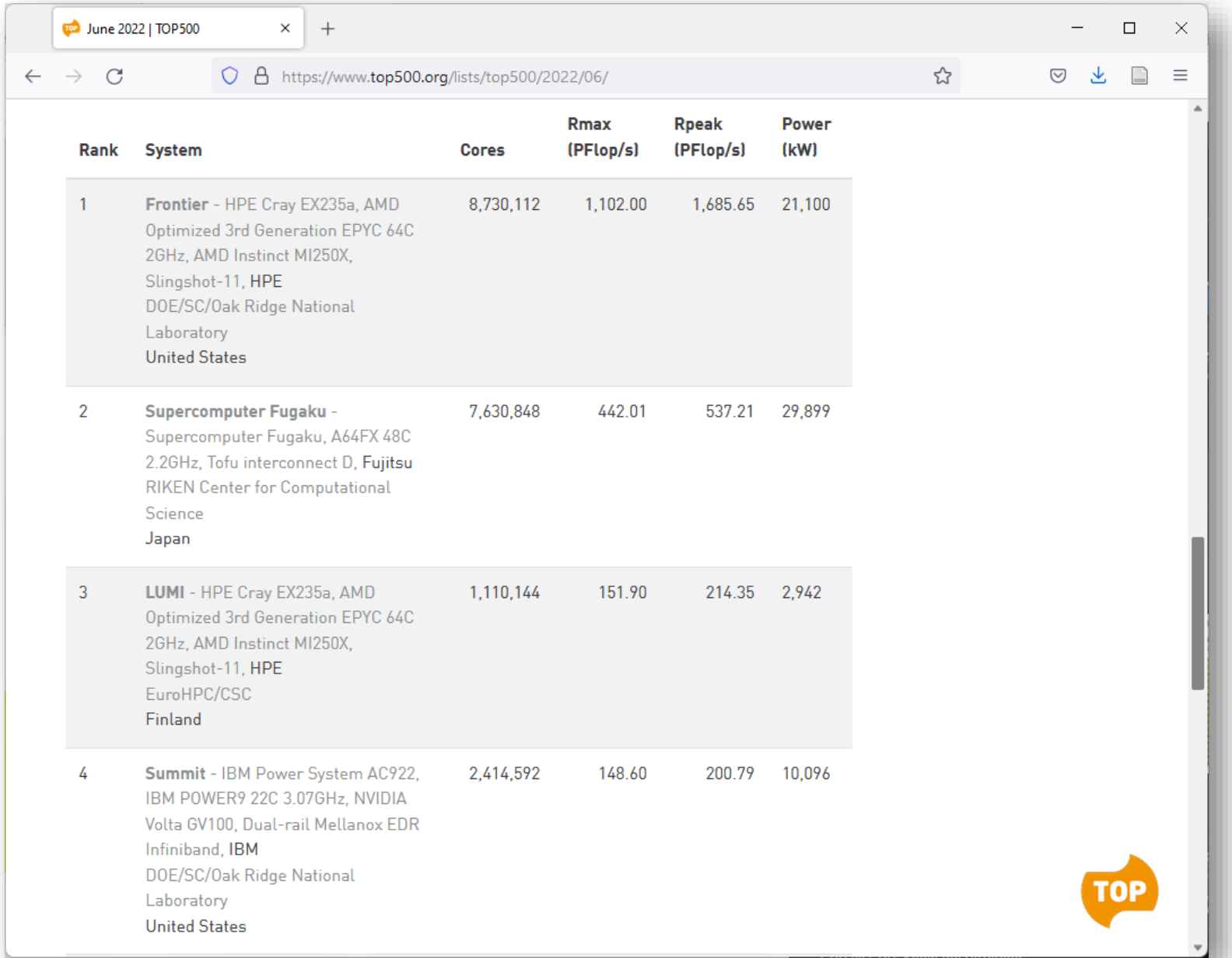

CPU Transistor Counts 1971-2008 & Moore’s Law
Transistor count
2,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
Curve shows ‘Moore’s Law’:
transistor count doubling
every two years
Dual-Core Itanium 2
Quad-Core Itanium Tukwila
GT200
POWER6
RV770
G80
Itanium 2 with 9MB cache
K10
Core 2 Quad
Core 2 Duo
Cell
Itanium 2
K8
Barton
Atom
P4
K7
K6-III
K6
PII
PIII
K5
Pentium
486
386
286
8088
8080
4004
8008
1971
1980
1990
2000
2008
Date of introduction

Today's supercomputers are massive clusters



Frontier Supercomputer at Oak Ridge

Current top supercomputers



Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	1,110,144	151.90	214.35	2,942
4	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096

as of 6/22
www.top500.org

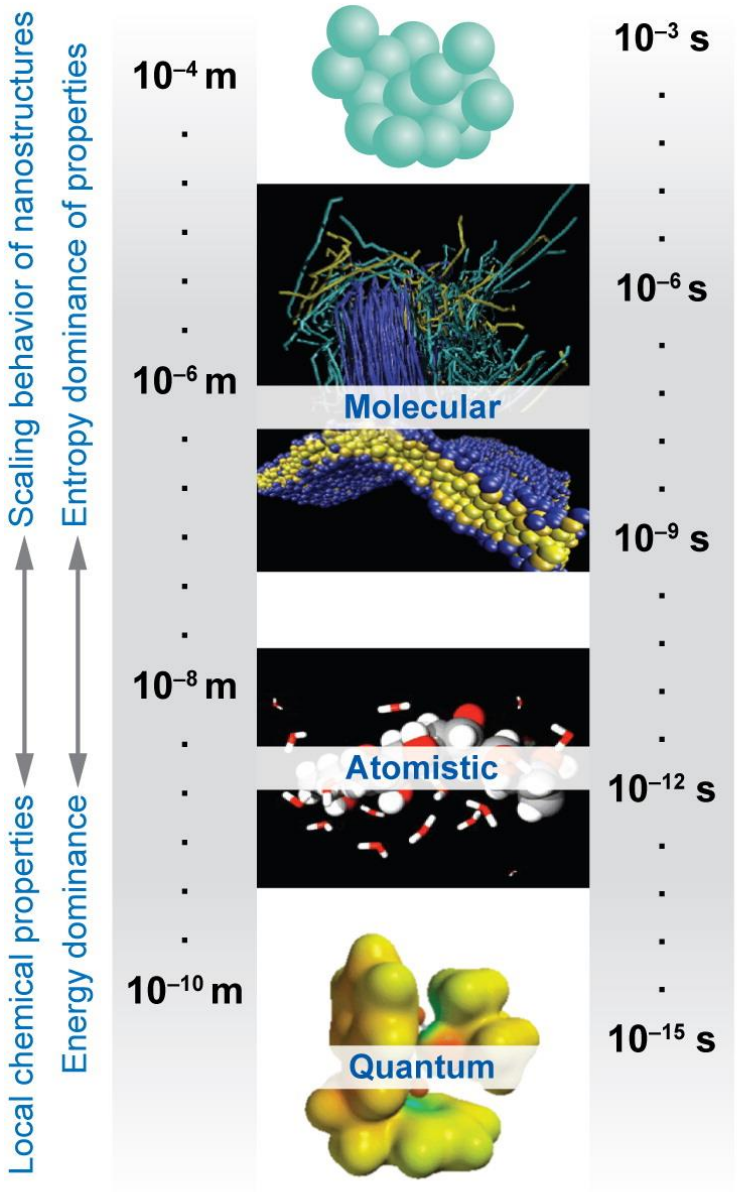
Growth of simulation power

- 10^7 increase in single processor speed since 1977
- 20-500 further increase due to parallelization
- 10^4 – 10^6 further increase due to algorithms

- NET: 13-15 orders of magnitude improvement

- BUT: still orders of magnitude behind reality (longest molecular dynamics simulations are ~100s μs)

adapted from K. Gubbins at http://chumba.che.ncsu.edu/che596m/



MACRO

Finite-element calculations
Continuum hydrodynamics
Dissipative particle dynamics
MD-MC-Lattice Boltzmann

MESO

MC statics/dynamics
MD, NEMD

MICRO

Classical force field MD, MC
Embedded atomistic-quantum methods
Car-Parrinello MD
Quantum MC
Quantum chemical methods

Praprotink M, et al. 2008.
Annu. Rev. Phys. Chem. 59:545–71.

Secrets to modeling (AKA, the hard parts)

- **Develop a molecular model capable of capturing the behavior of interest**
  - scaling laws? basic driving forces? molecular structures? quantitative predictions?
- **Use a simulation approach that addresses the physics of interest and any bottlenecks / challenges**
  - long time scales? pathways? specific interactions?
- **Connect results to statistical-mechanics**
  - free energies? phase behavior?

This week and next

- Review of probability and statistical mechanics (brief)
- Introduction to Python, NumPy, and SciPy
  (mostly through reading)
- Ab initio methods
- Classical semi-empirical models
- Exploring the potential energy landscape

Pay attention to *Pythonic* coding styles

TASK: determine if a string is in an array

```
arr = ["apples", "oranges", "bananas", "grapes"]
s = "cherries"
found = False
size = len(arr)
for i in range(0, size):
    if arr[i] == s:
        found = True
```

VERSUS

```
arr = ["apples", "oranges", "bananas", "grapes"]
found = "cherries" in arr
```

Pay attention to *Pythonic* coding styles

TASK: find the centroid of a set of coordinates

```
Centroid = numpy.zeros(3)
n = len(Pos)
for i in range(0, n):
     Centroid = Centroid + Pos(i,:)
Centroid = Centroid / n
```

VERSUS

```
Centroid = numpy.mean(Pos, axis=0)
```

Produce professional graphs sized appropriately

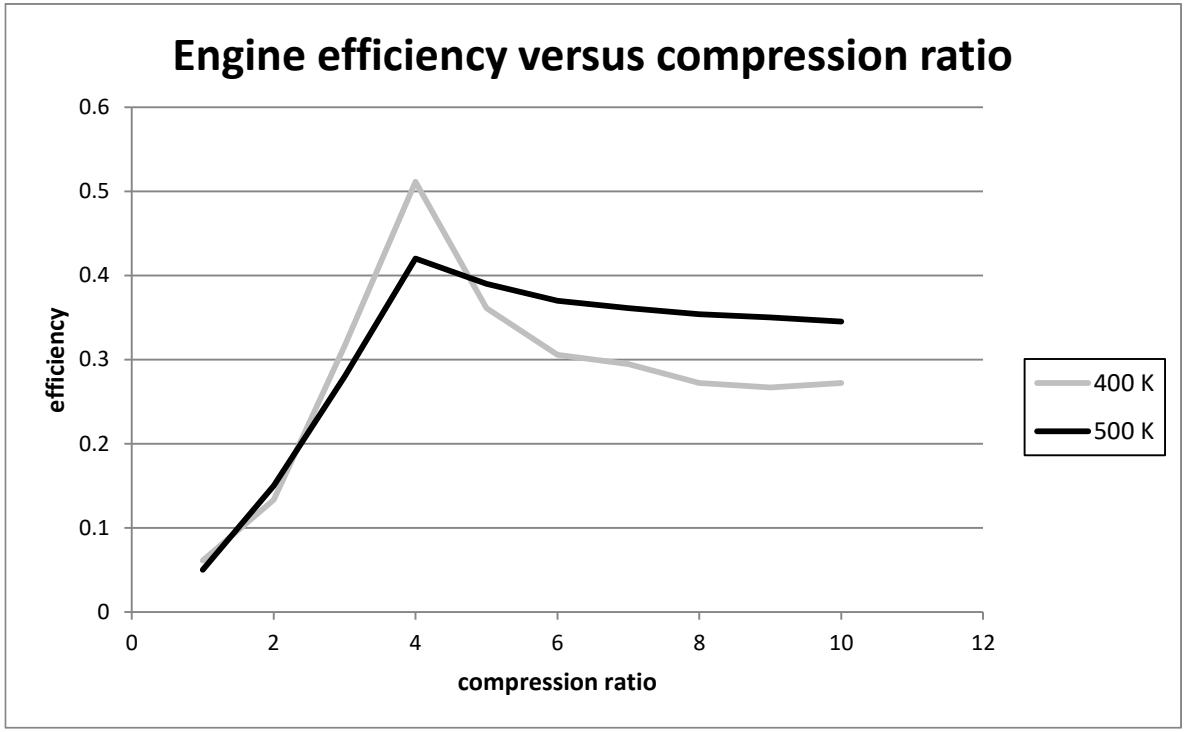


Fig. 5. **Engine efficiency vs. compression ratio.** The graph plots two temperature cases, showing that a higher temperature provides a slight increase in efficiency at elevated compression ratios.

Produce professional graphs sized appropriately

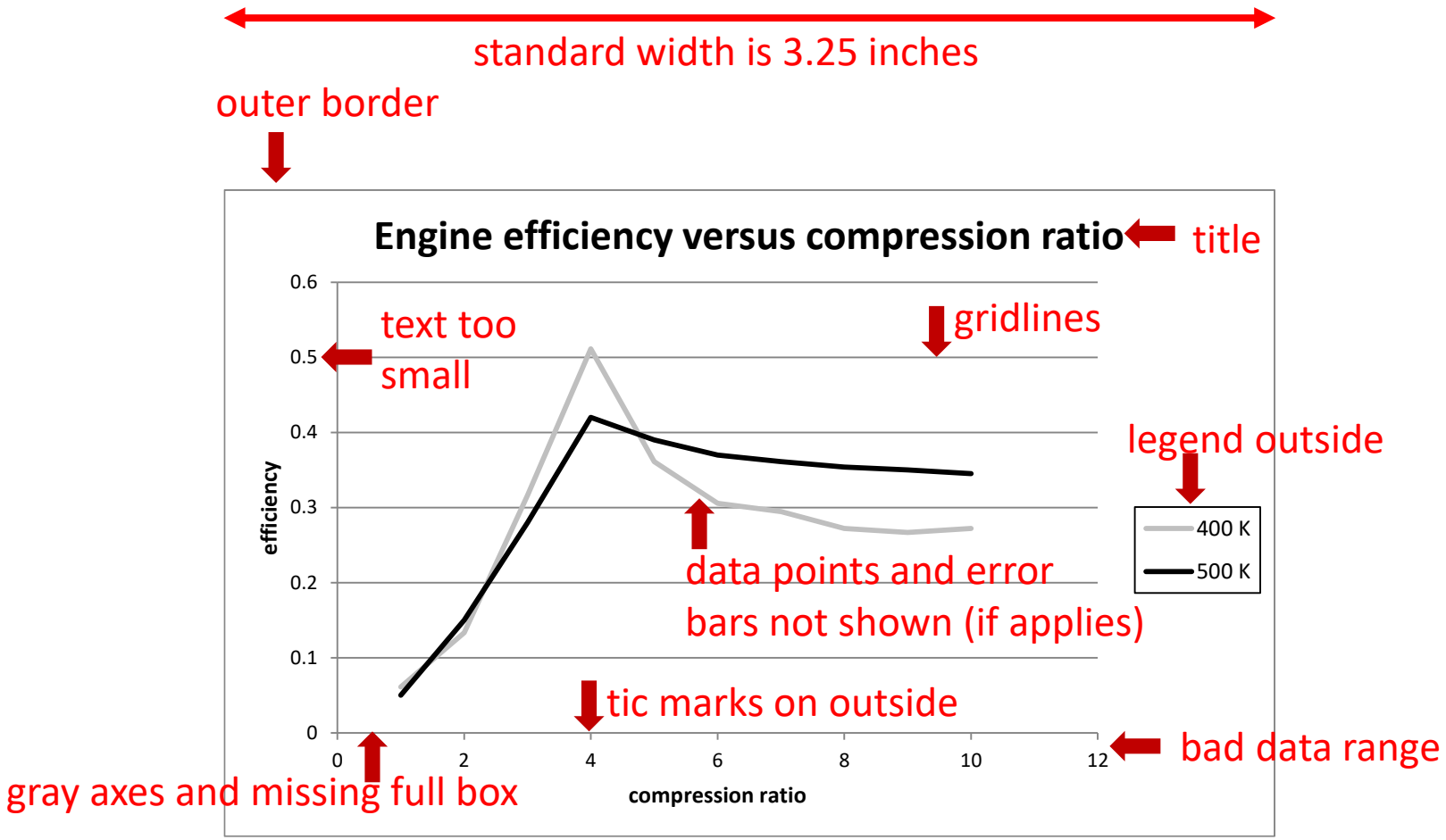


Fig. 5. **Engine efficiency vs. compression ratio.** The graph plots two temperature cases, showing that a higher temperature provides a slight increase in efficiency at elevated compression ratios.

Produce professional graphs sized appropriately

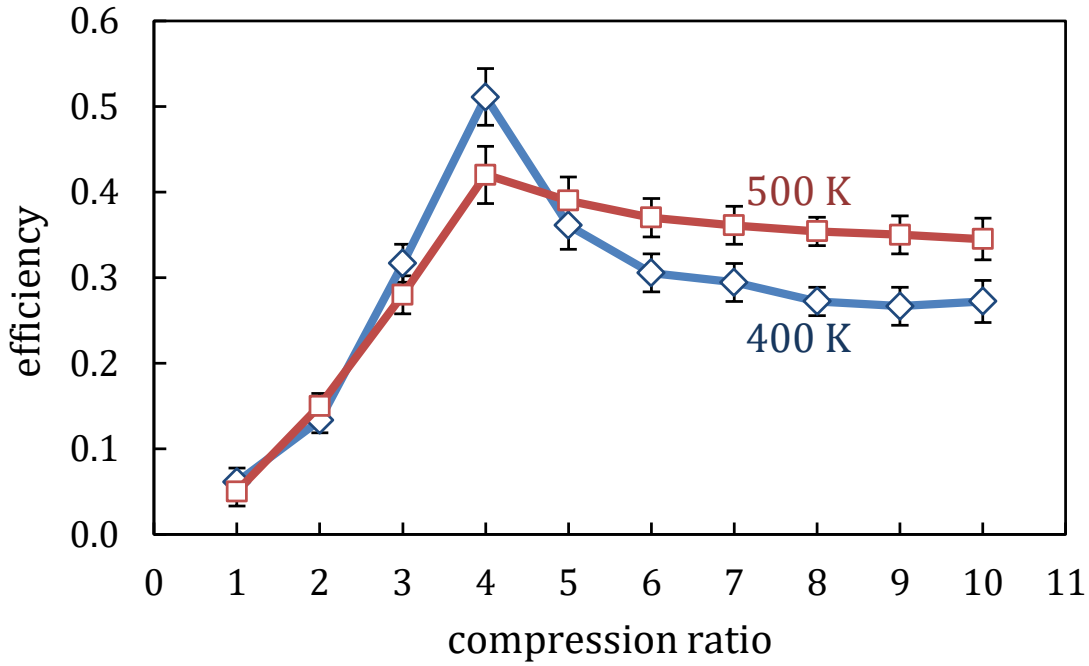


Fig. 5. **Engine efficiency vs. compression ratio.** A higher temperature provides a slight increase in efficiency at elevated compression ratios.

Do me a favor

- If you find major typos or errors in the tutorials and lecture notes, please send me a quick email!