OPT++

A Toolkit for Nonlinear Optimization

Juan Meza

High Performance Computing Research
Lawrence Berkeley National Laboratory
http://crd.lbl.gov/~meza

Office of Science
U.S. DEPARTMENT OF ENERGY

5th International Congress on Industrial and Applied Mathematics,
Sydney, Australia, July 7-11, 2003

BERKELEY LAB

Outline

- Introduction
- Classes of Optimization Solvers
- Setting up a Problem and Algorithm
- Example 1: Unconstrained Optimization
- Example 2: Constrained Optimization
- Parallel optimization techniques
- Example application
- Summary

Introduction

- OPT++ is an open source toolkit for general nonlinear optimization problems
- Original development started in 1992 at Sandia National Labs/CA
- Major contributors
  - Juan Meza, LBNL
  - Ricardo Oliva, LBNL
  - Patty Hough, SNL/CA
  - Pam Williams, SNL/CA

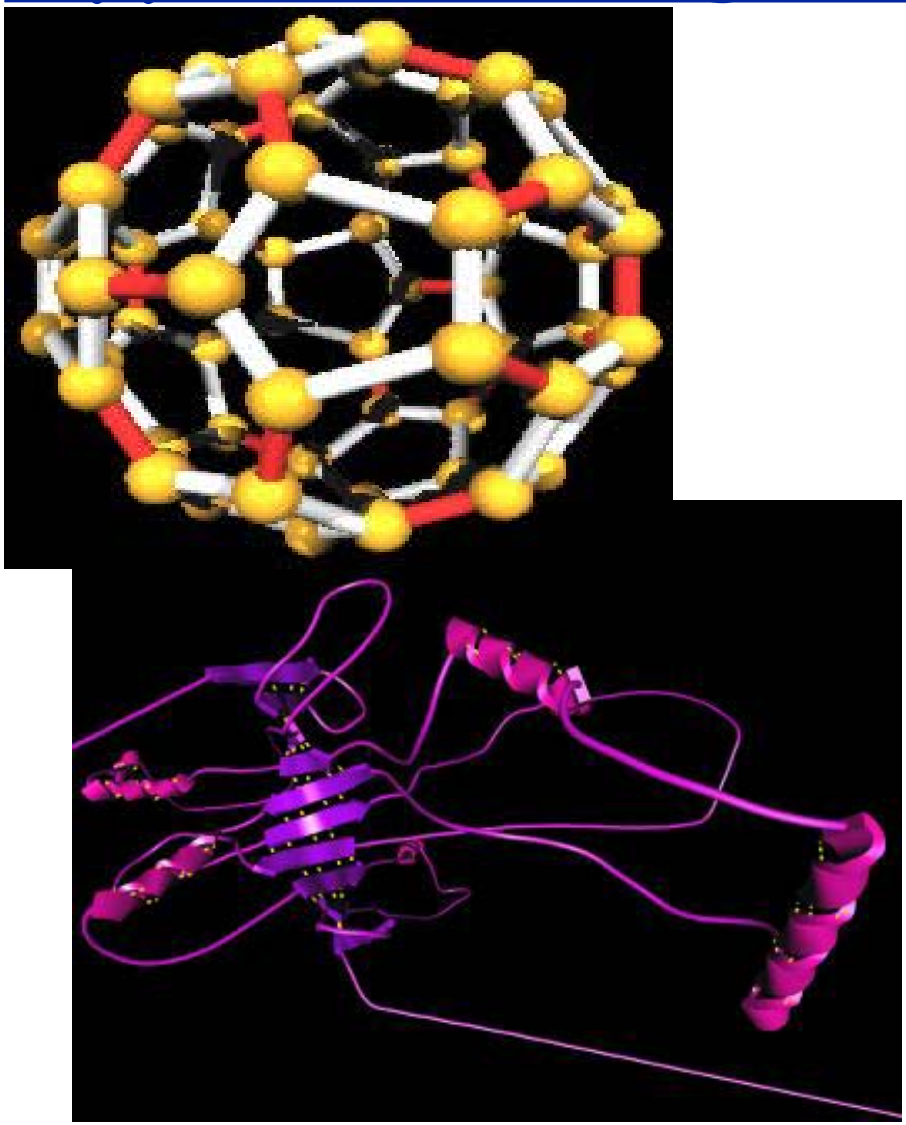
OPT++ USERS

Total = 338

Other (Country not identified) = 120

As of April 2003

Application targets



- Predict properties of nanostructures and/or design nanostructures with desired propertiesr
- Create secondary structures: obtain predictions of α-helices and β-sheets.
- Configuration of T174 generated using ProteinShop

http://graphics.cs.ucdavis.edu/~okreylos/ResDev/ProtoShop/index.html

General Optimization Problem

$$\min_{x \in \Re^n} f(x), \qquad \text{Objective function}$$

$$s.t. \quad h(x) = 0, \qquad \text{Equality constraints}$$

$$g(x) \geq 0 \qquad \text{Inequality constraints}$$

$$L = f(x) + y^T h(x) - w^T g(x)$$

Classes of Optimization Problems

- Unconstrained optimization
- Bound constrained optimization
  - Only upper and lower bounds
  - Sometimes called “box” constraints
- General nonlinearly constrained optimization
  - Equality and inequality constraints
  - Usually nonlinear
- Some special case classes (not currently handled in OPT++)
  - Linear programming (function and constraints linear)
  - Quadratic programming (quadratic function, linear constraints)

OPT++ Philosophy

- Problem should be defined in terms the user understands
  - *Do I have second derivatives available?* and not
  - *Is my objective function twice continuously differentiable?*
- Solution methods should be easily interchangeable
  - Once the problem is setup, methods should be easy to interchange so that the user can compare algorithms
- Common components of algorithms should be interchangeable
  - Algorithm developers should be able to re-use common components from other algorithms, for example line searches, step computations, etc.

Classes of Problems in OPT++

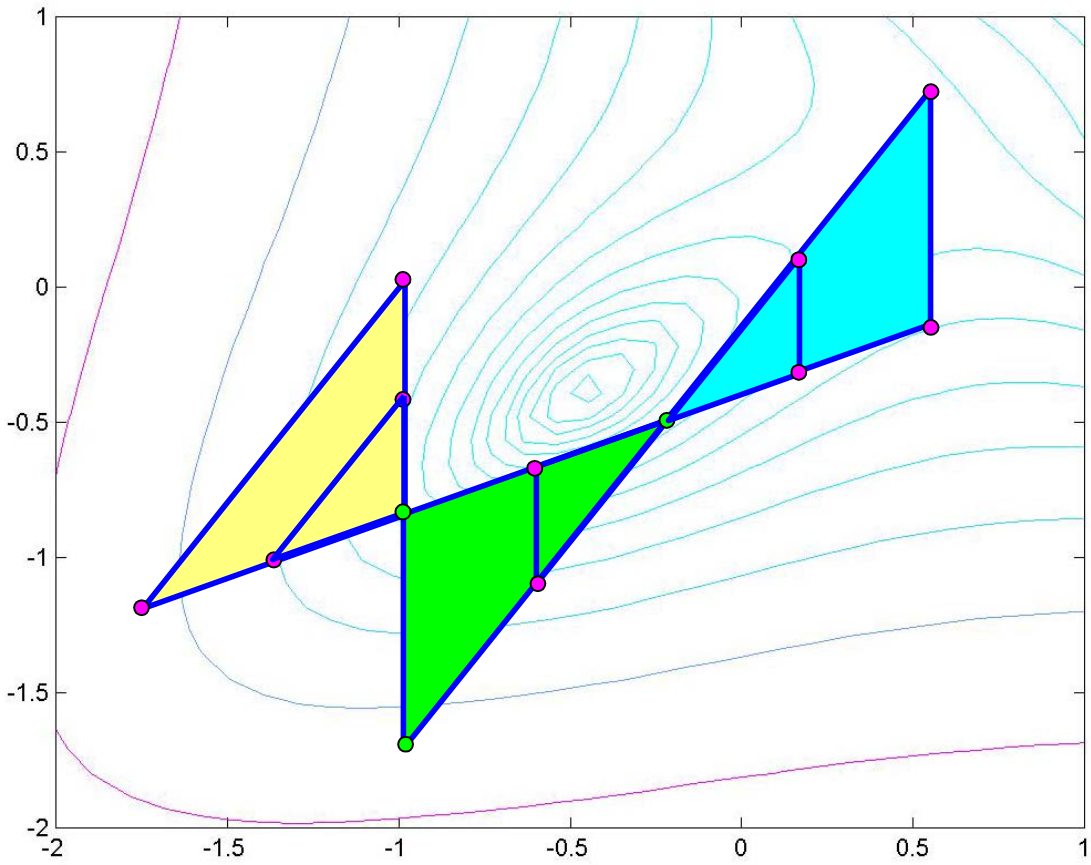
- Four major classes of problems available
  - *NLF0(ndim, fcn, init_fcn, constraint)*
    - Basic nonlinear function, no derivative information available
  - *NLF1(ndim, fcn, init_fcn, constraint)*
    - Nonlinear function, first derivative information available
  - *FDNLF1(ndim, fcn, init_fcn, constraint)*
    - Nonlinear function, first derivative information approximated
  - *NLF2(ndim, fcn, init_fcn, constraint)*
    - Nonlinear function, first and second derivative information available

Classes of Solvers in OPT++

- Direct search
  - No derivative information required
- Conjugate Gradient
  - Derivative information may be available but doesn't use quadratic information
- Newton-type methods
  - Algorithm attempts to use/approximate quadratic information
  - Newton
  - Finite-Difference Newton
  - Quasi-Newton
  - NIPS

Quick tour of some of the algorithms

Pattern search

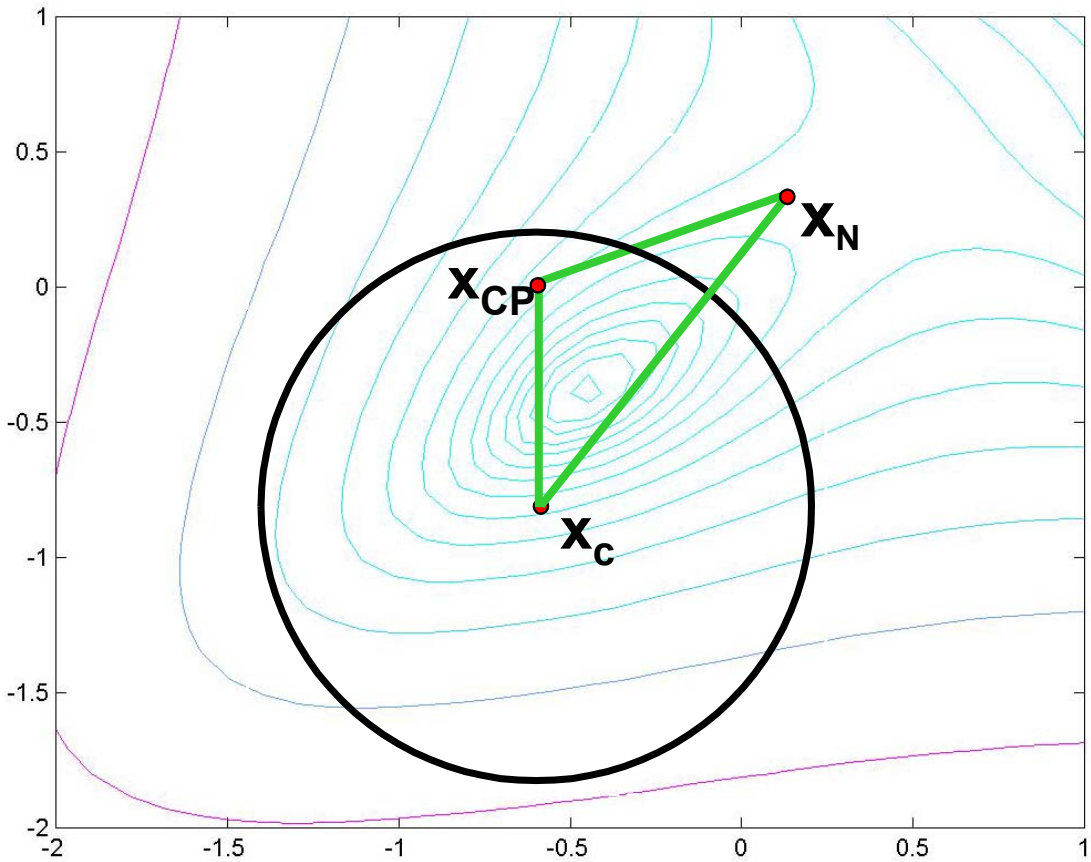


- Can handle noisy functions
- Do not require derivative information
- Inherently parallel
- Convergence can be painfully slow

Conjugate Gradient Methods

- Two major classes
- Standard nonlinear conjugate gradient
  - Two different types of line searches
- Limited Memory BFGS
  - Unconstrained version available
  - Bound constrained version under development

Newton-type Methods



- Fast convergence properties
- Good global convergence properties
- Inherently serial
- Difficulties with noisy functions

NIPS: Nonlinear Interior Point Solver

- Interior point method
- Based on Newton's method for a particular system of equations (perturbed KKT equations, slack variable form)
- Can handle general nonlinear constraints
- Can handle strict feasibility

$$F(\mu) = \begin{bmatrix} \nabla f(x) + \nabla h(x) y - \nabla g(x) w \\ w - z \\ h(x) \\ g(x) - s \\ ZSe - \mu e \end{bmatrix} = 0$$

Constraints

- Constraint types
  - BoundConstraint(numconstraints, lower, upper)
  - LinearInequality(A, rhs, stdFlag)
  - NonLinearInequality(nlprob, rhs, numconstraints, stdFlag)
  - LinearEquation(A, rhs)
  - NonLinearEquation(nlprob, rhs, numconstraints)
- The whole shebang
  - CompoundConstraint(constraints)

Algorithm Choices Depend on Problem

	NLF0	FDNLF1	NLF1	NLF2
OptPDS	x	x	x	x
OptCG		x	x	x
OptLBFGS		x	x	x
OptQNewton		x	x	x
OptBCQNewton		x	x	x
OptFDNewton		x	x	x
OptFDNIPS		x	x	x
OptNewton				x
OptBCNewton				x
OptNIPS				x

Bare bones example: unconstrained optimization

```
void init_rosen_x0(int ndim, ColumnVector& x);
void rosen(int ndim, const ColumnVector& x, double& fx, int& result);

int main() {
    int ndim = 2;
    FDNLF1 nlp(ndim, rosen, init_rosen_x0);
    nlp.initFcn();
    OptQNewton objfcn(&nlp);
    objfcn.setSearchStrategy(TrustRegion);
    objfcn.setMaxFeval(200);
    objfcn.setFcnTol(1.e-4);
    objfcn.optimize();
}
```

Example 2: Constrained optimization

$$\min \ (x_1 - x_2)^2 + (1/9)(x_1 + x_2 - 10)^2 + (x_3 - 5)^2$$

$$s.t.$$

$$x_1^2 + x_2^2 + x_3^2 \le 48,$$

$$-4.5 \le x_1 \le 4.5,$$

$$-4.5 \le x_2 \le 4.5,$$

$$-5.0 \le x_3 \le 5.0$$

Constrained optimization: Step 1

Defining the bound constraints:

$$-4.5 \le x_1 \le 4.5,$$
$$-4.5 \le x_2 \le 4.5,$$
$$-5.0 \le x_3 \le 5.0$$

```
int ndim = 3;

ColumnVector lower(ndim), upper(ndim);

lower << -4.5 << -4.5 << -5.0;

upper << 4.5 << 4.5 << 5.0 ;

Constraint bc = new BoundConstraint(ndim, lower, upper);
```

Constrained optimization: Step 2

Defining the nonlinear inequality constraint:

$$x_1^2 + x_2^2 + x_3^2 \leq 48$$

```
NLP* chs65 = new NLP(new NLF2(ndim, 1, ineq, init_hs65_x0));
Constraint nleqn = new NonLinearInequality(chs65);
```

Collecting both constraints into one constraint object :

```
CompoundConstraint* constraints =
                        new CompoundConstraint(nleqn, bc);
```

Constrained optimization: Step 3

Defining and initializing the nonlinear problem:

```
NLF2 nips(ndim, hs65, init_hs65_x0, constraints);
nips.initFcn();
```

Defining the Optimization object and optimizing it!

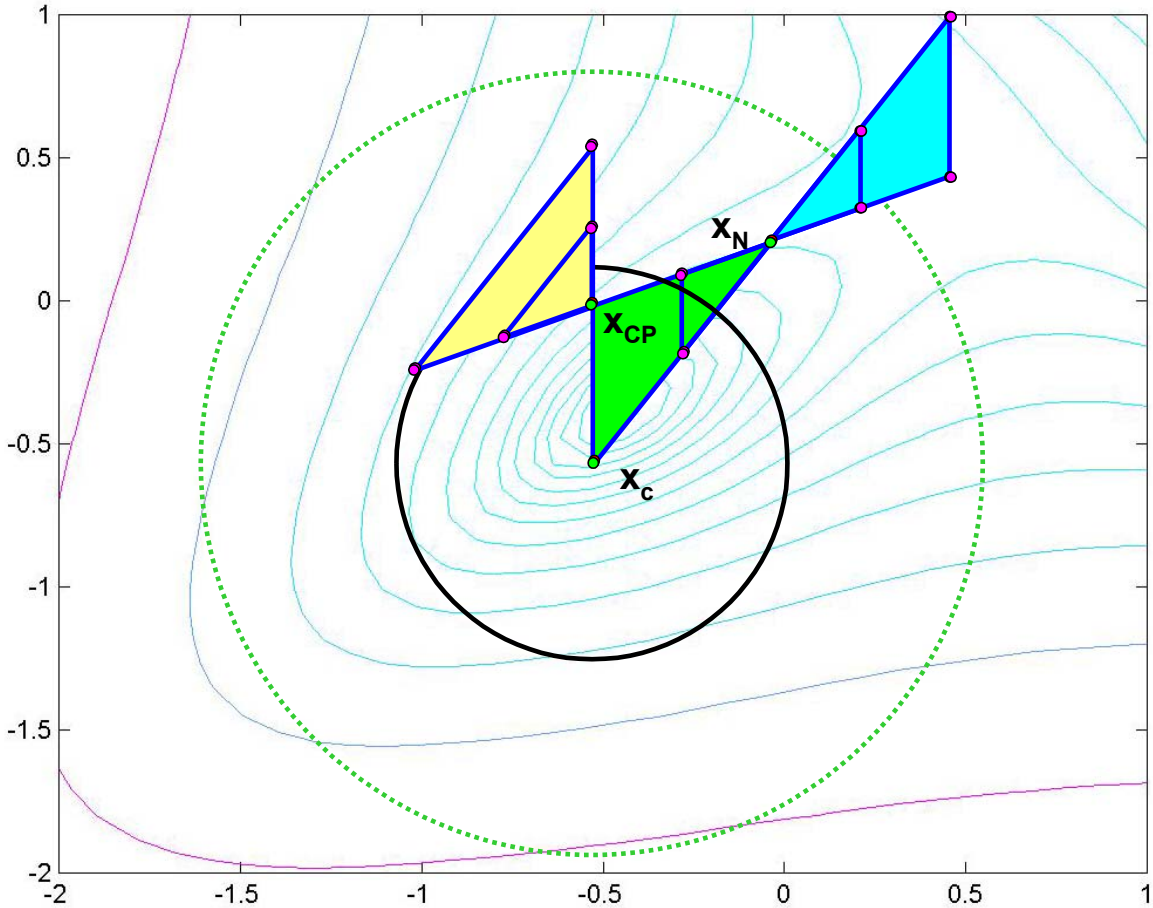
```
OptNIPS optobj(&nips);
optobj.optimize();
```

Parallel Optimization

Schnabel (1995) Identified Three Levels for Introducing Parallelism Into Optimization

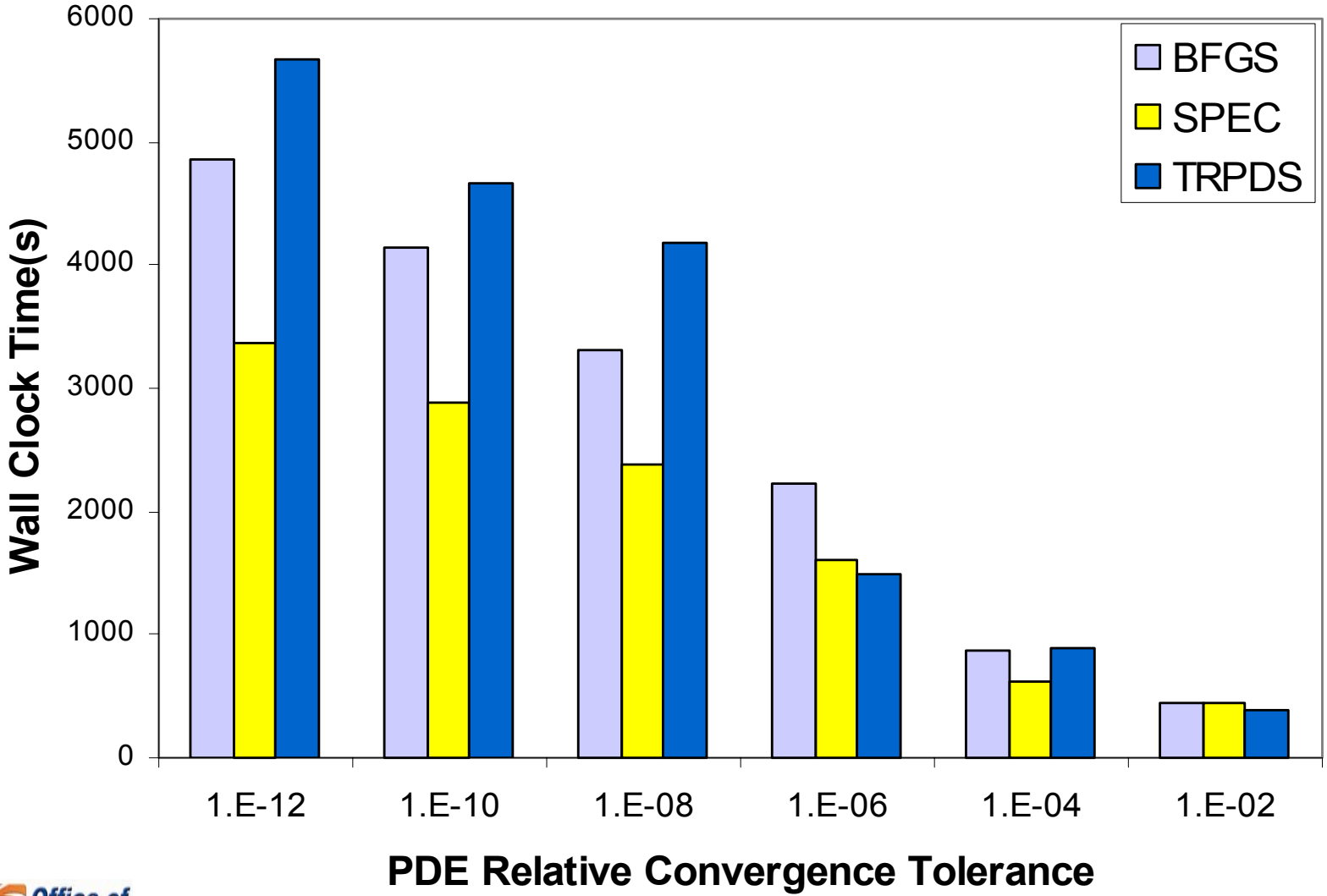
- Parallelize evaluation of function/gradient/constraints
  - May or may not be easy to implement
- Parallelize linear algebra
  - Really only useful if the optimization problem is large-scale
- Parallelize optimization algorithm at a high level
  - Multiple function evaluations in parallel

Trust Region + PDS



- Fast convergence properties of Newton method
- Good global convergence properties of trust region approach
- Inherent parallelism of PDS
- Ability to handle noisy functions

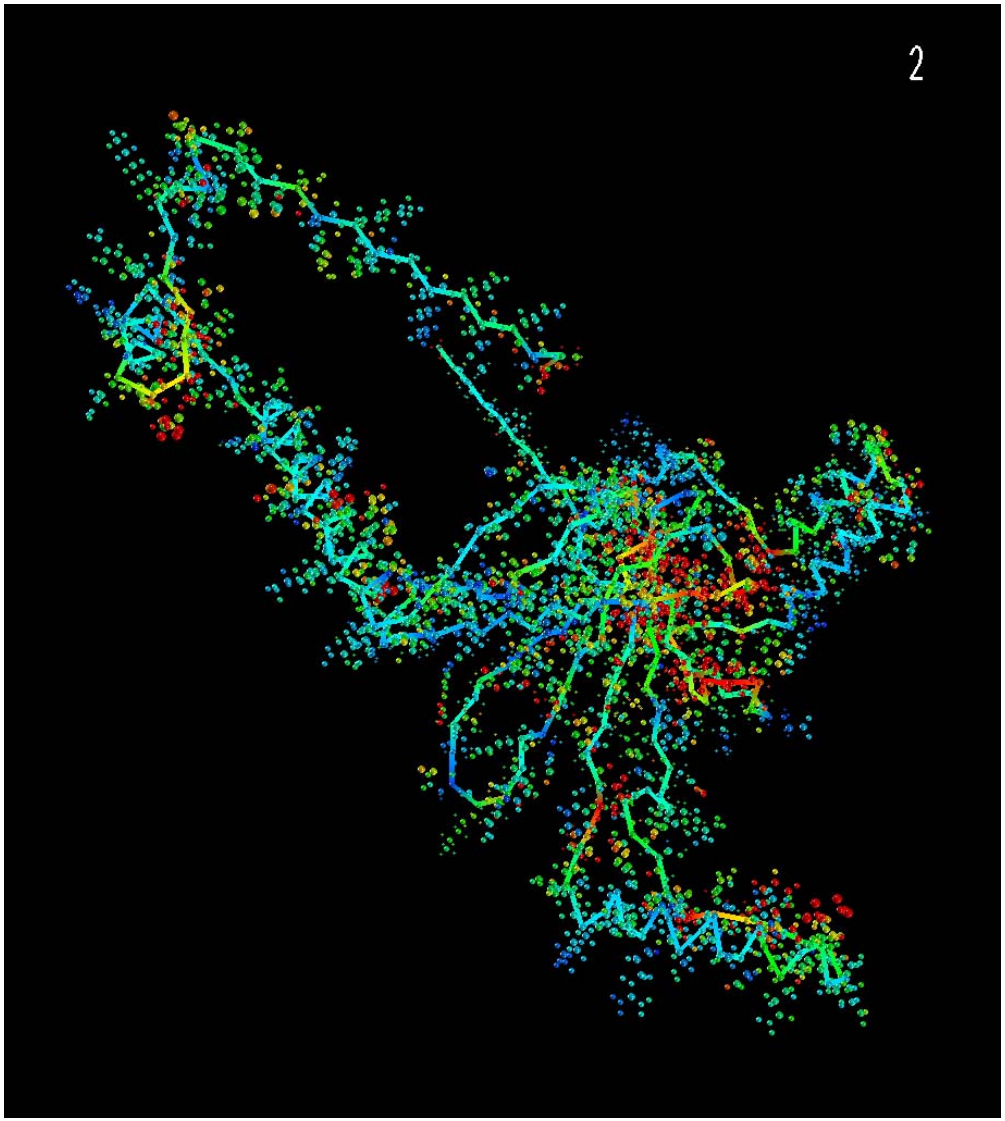
Comparison of TRPDS with other approaches



Application: Protein Folding

```
void init_X0(int ndim, ColumnVector& x);
void eval_energy(int ndim, const ColumnVector& x, double& fx, int&
    result);
int main() {
    PDB pdb("t162.pdb");  // loads pdb file
    int ndim = 3 * pdb.NumAtoms();
    FDNLF1 nlp(ndim, eval_energy, init_X0);
    nlp.initFcn();
    OptLBFGS optobj(&nlp);
    optobj.setMaxFeval(10000);
    optobj.setFcnTol(1.e-6);
    optobj.optimize();
}
```

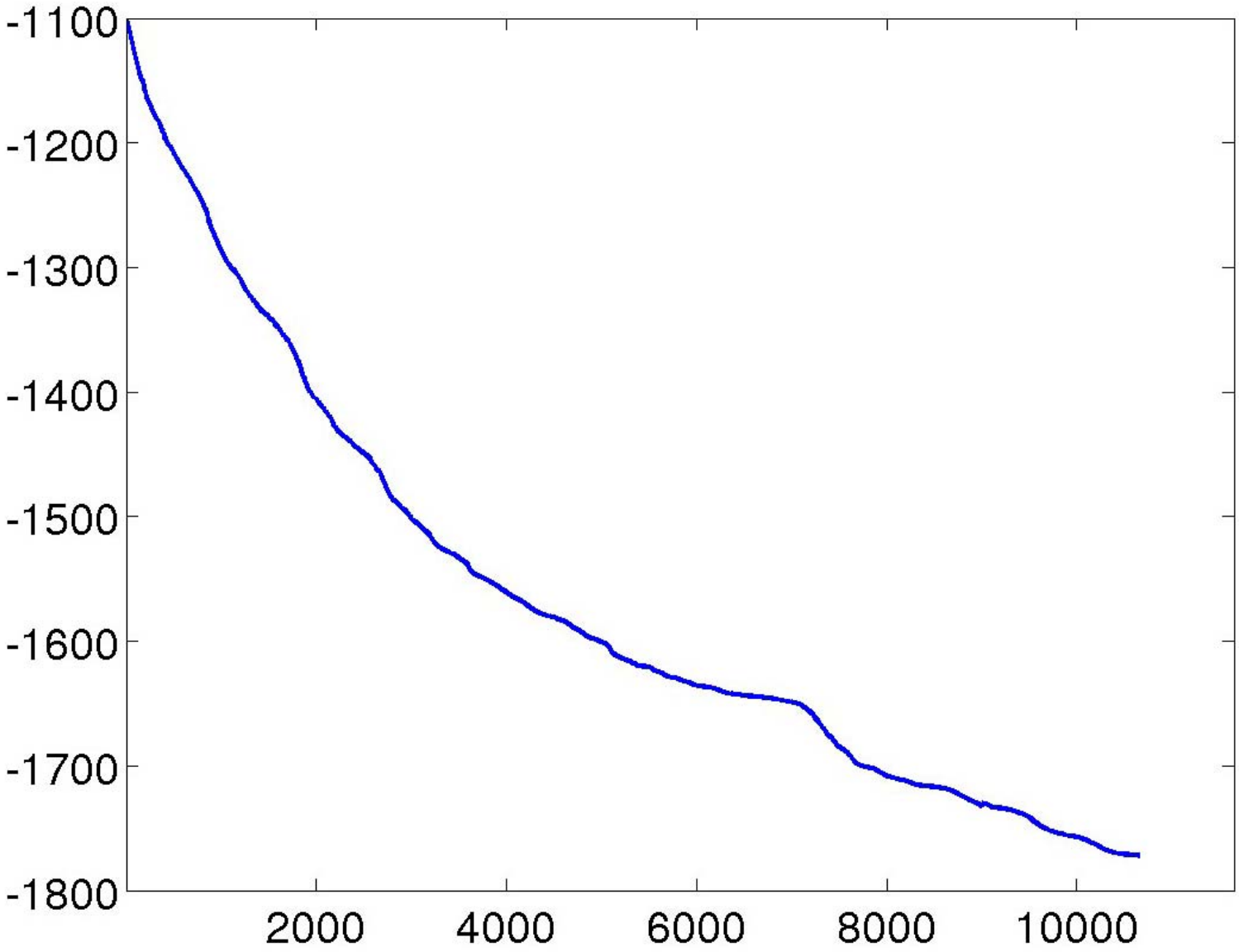
Energy Minimization Using LBFGS



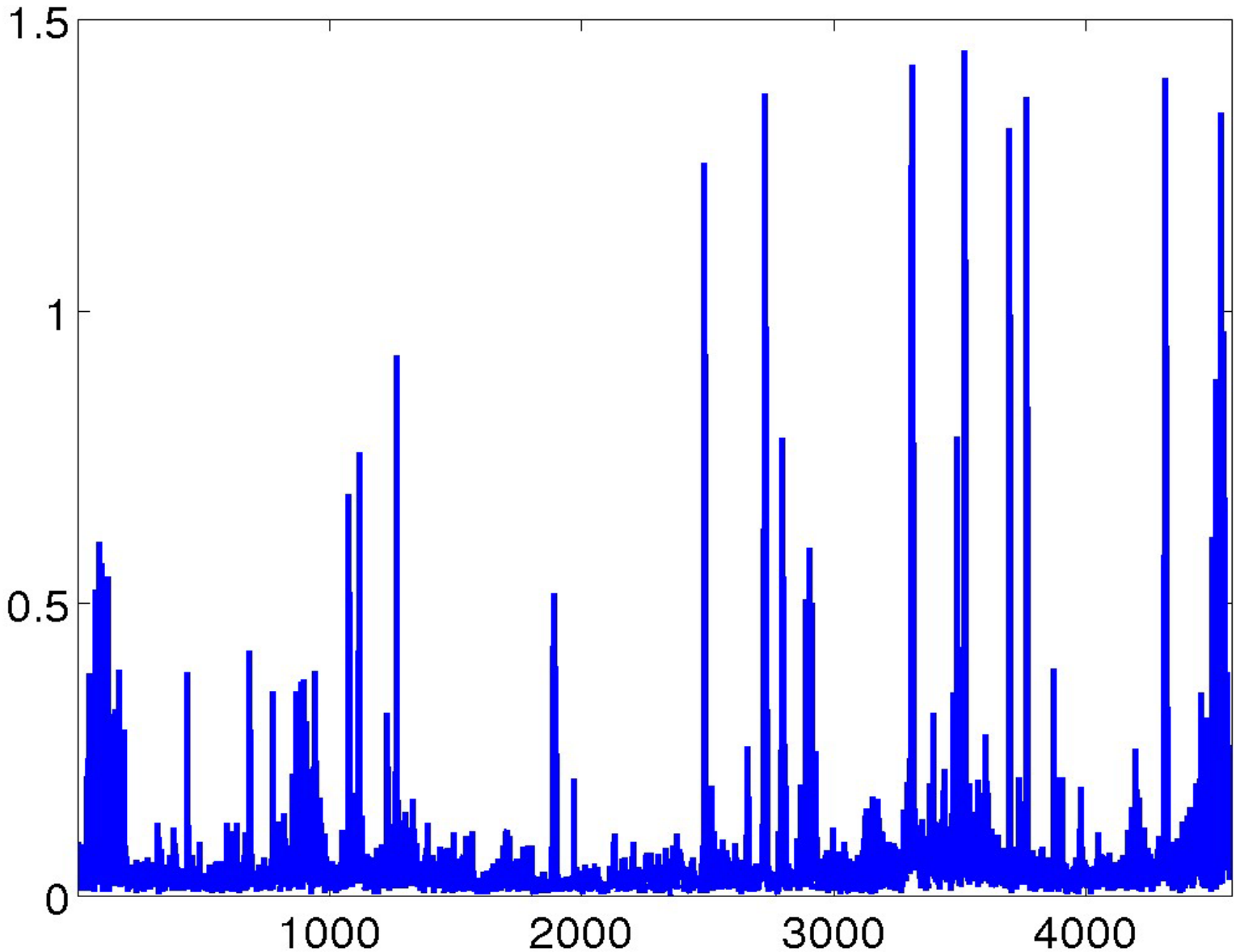
- N=13728 (4576 Atoms)
- Energy Function: AMBER
- LBFGS with M=15
- 11656 iterations, 11887 function evaluations
- Stop on *f* tolerance with *ftol*=1e-6
- Each function eval: ~5sec

Protein T162 (from CASP5)

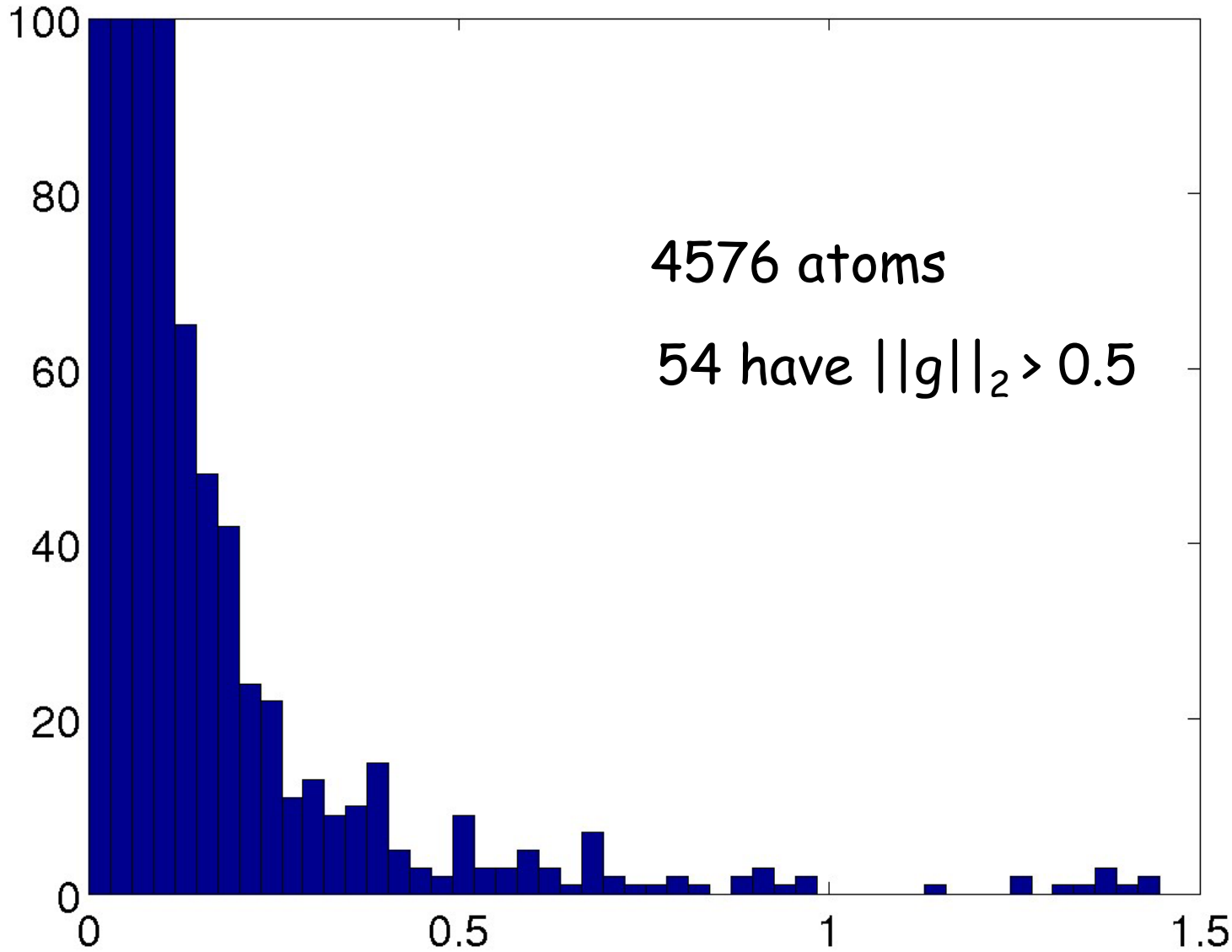
Energy vs. LBFGS iterations



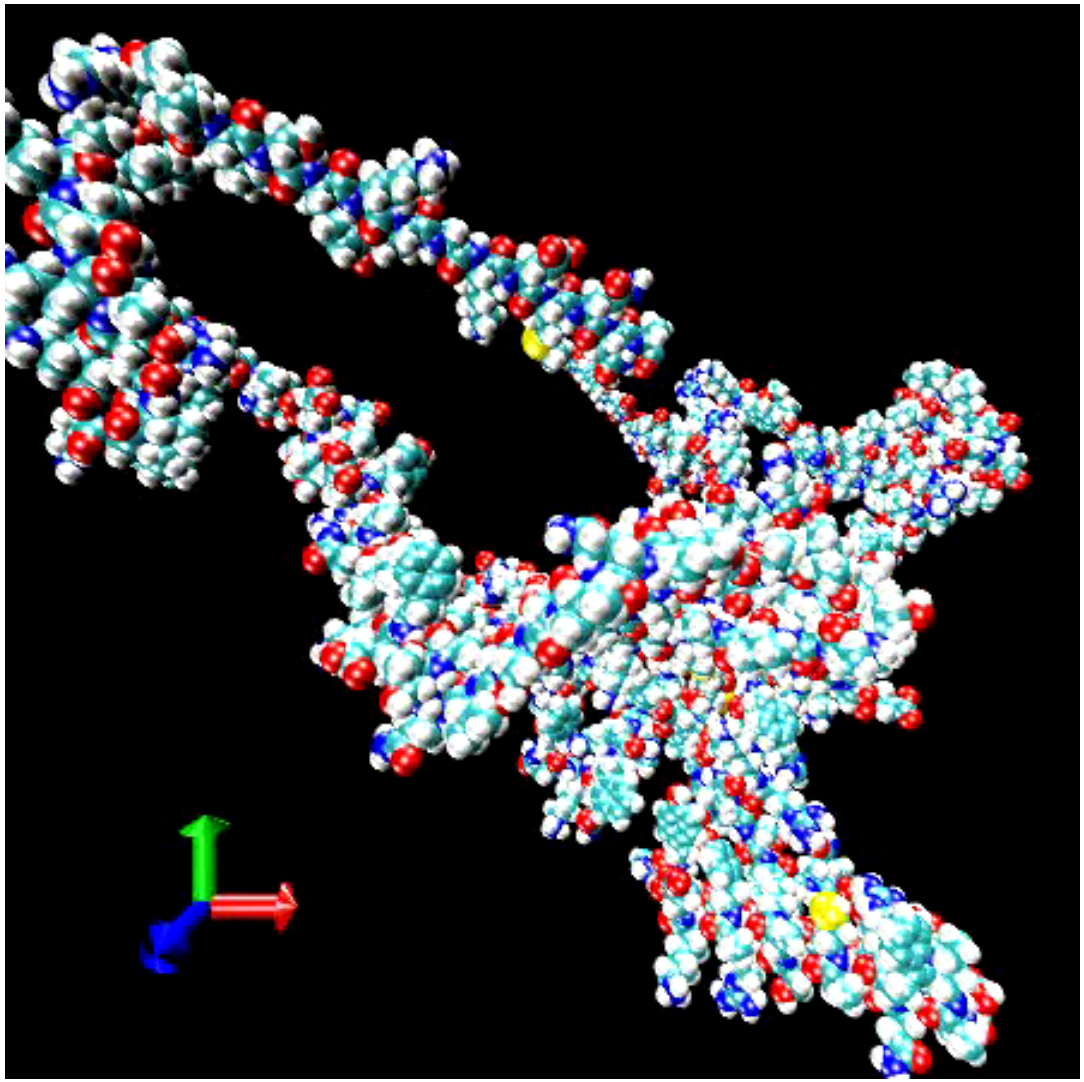
Norm of *gradient* for each atom



Distribution of ||gradient|| by atom



Protein T162 (from CASP5)



- Initial configuration created using ProteinShop (S. Crivelli)
- Energy minimization computed using OPT++/LBFGS
- Final RMSD change: 3.9 (avg)
- Total simulation took approximately 32hours on a 1.7GHz machine with 512 RAM

Summary

- OPT++ can handle many types of nonlinear optimization problems
- The toolkit can be used to compare the effectiveness of several algorithms on the same problem easily
- The user needs to provide only functions for the objective function and the constraints
  - If additional information is available it can be easily incorporated
- The code is open source and available at either
  - http://www.nersc.gov/~meza/projects/opt++
  - http://csmr.ca.sandia.gov/opt++

References

- Other links
  - http://sal.kachinatech.com/B/3/index.shtml
  - http://www-neos.mcs.anl.gov/neos
  - http://www.mcs.anl.gov/tao
  - http://endo.sandia.gov/DAKOTA/index.html
- Books/Papers
  - Dennis and Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, 1983
  - Gill, Murray, Wright, *Practical Optimization*, Academic Press, 1981
  - El-Bakry, Tapia, Tsuchiya, Zhang, *On the Formulation and Theory of the Newton Interior-Point Method for Nonlinear Programming*, JOTA, Vol. 89, No.3, pp.507-541, 1996
  - More´ and Wright, Optimization Software Guide, SIAM, 1993

OPT++ Users around the world

Total = 235

Other (Country not identified) = 34