An Introduction to the Libraries for Support of Optimization Problems OPT++ and TAO

http://acts.nersc.gov

acts-support@nersc.gov

Tony Drummond

Lawrence Berkley National Laboratory

SIAM CSE05 - Orlando, FL, 2/11/05

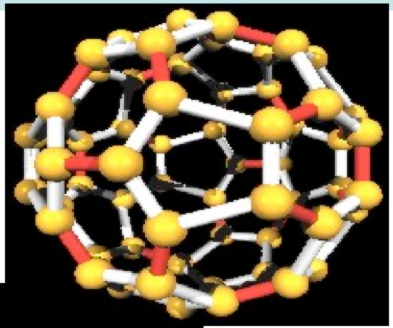
Tutorial prepared with reference materials from OPT++ (J. Meza) and TAO (Jorge More and Steve Benson)

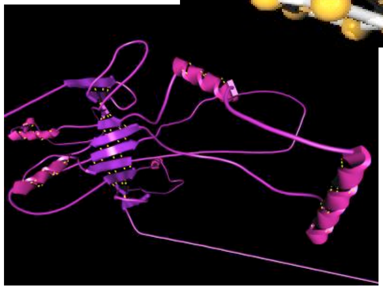
Types of Optimization Problems

- Unconstrained optimization
- Bound constrained optimization
  - Only upper and lower bounds
  - Sometimes called "box" constraints
- General nonlinearly constrained optimization
  - Equality and inequality constraints
  - Usually nonlinear
- Others
  - Linear programming (function and constraints linear)
  - Quadratic programming (quadratic function, linear constraints)

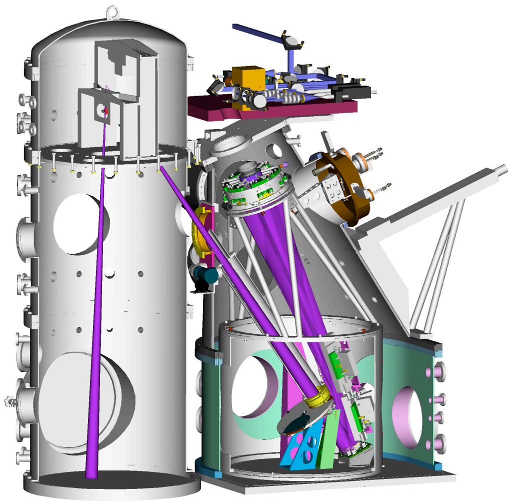
Optimization Problems

- Simulation-based optimization problems
  - Predict properties of nanostructures or design nanostructures with desired properties
  - Protein folding problems: create secondary structures and obtain predictions of a-helices and β-sheets.





- Parameter identification
  - Find model parameters, satisfying some bounds, for which the simulation matches the observed temperature profiles
  - Computing objective function requires running thermal analysis code.



General Optimization Problem

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

$$s.t. \quad h(x) = 0,$$ ← Equality constraints

$$g(x) \geq 0$$ ← Inequality constraints

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

ACTS Numerical Tools: *Functionality*

Computational Problem	Methodology	Algorithm	Library
Non-Linear Optimization	Newton Based	Newton	OPT++ TAO
		Finite-Difference Newton	OPT++ TAO
		Quasi-Newton	OPT++ TAO
		Non-linear Interior Point	OPT++ TAO
	CG	Standard Non-linear CG	OPT++ TAO
		Limited Memory BFGS	OPT++
		Gradient Projections	TAO
	Direct Search	No derivate information	OPT++

OPT++

- 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++

- 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

OPT++

	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

OPT++ Example 1 - *Constrained Optimization*

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

$$s.t.$$

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

$$-4.5 \leq x_1 \leq 4.5,$$

$$-4.5 \leq x_2 \leq 4.5,$$

$$-5.0 \leq x_3 \leq 5.0$$

OPT++ Example 1: 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);
```

OPT++ Example 1: 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);
```

OPT++ Example 1: 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();
```

OPT++ Example 2

```
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();
```

TAO Development

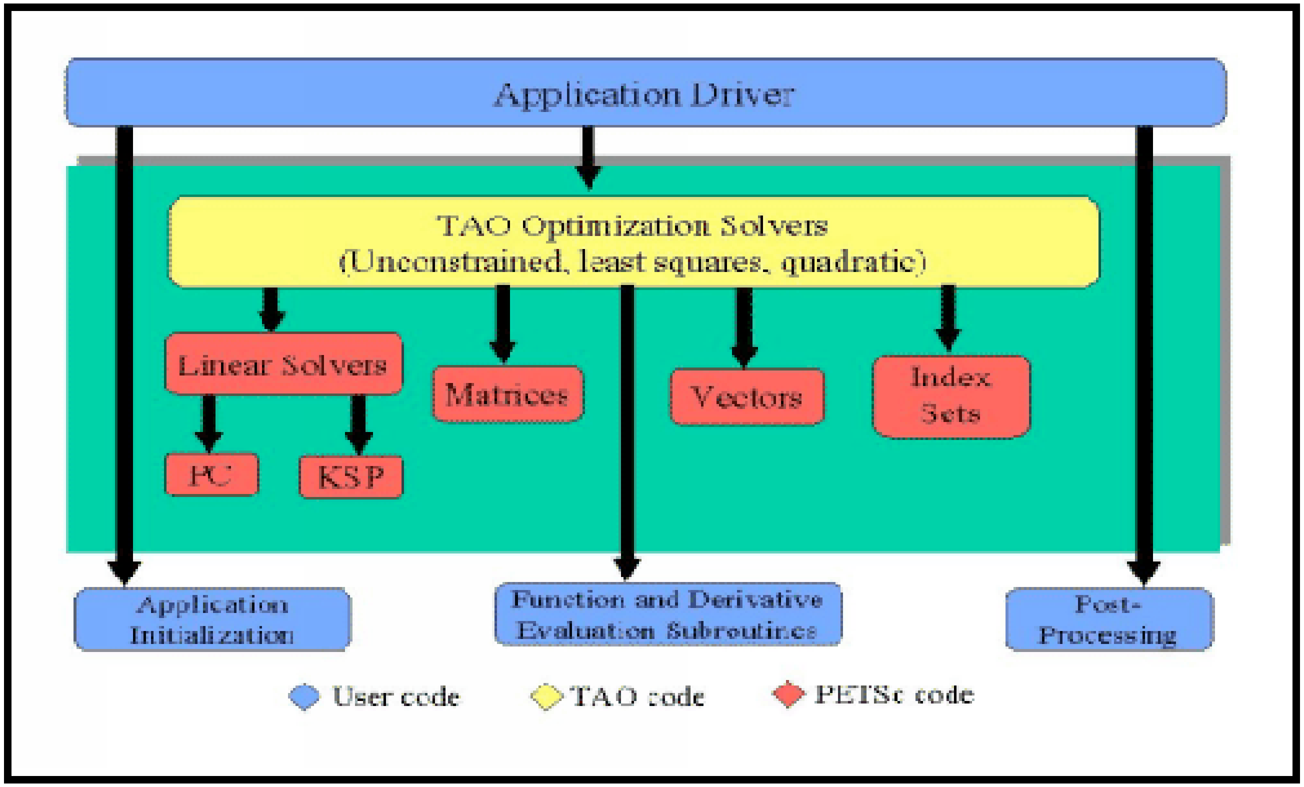
***T**oolkit for **A**dvanced **O**ptimization*

Argonne National Laboratory
Mathematics and Computer Science Division

- Steven Benson
- Louis Curfman-McInnes
- Jorge Moré
- Jaison Sarich

** The material used in the preparation of this tutorial comes from the TAO User's Guide, tutorials and examples prepared by the TAO development team*

TAO - Interface with PETSc



TAO Example (Using PETSc)

```
TAO_SOLVER      tao;            /* TAO Optimization solver          */
TAO_APPLICATION app;            /* TAO Application using PETSc      */
AppCtx          user;           /* user-defined application context */
Vec             x;              /* solution vector                  */
Mat             H;              /* Hessian Matrix                   */

VecCreateSeq(PETSC_COMM_SELF,n,&x);
MatCreateSeqAIJ(PETSC_COMM_SELF,n,n,nz,PETSC_NULL,&H);
TaoCreate(PETSC_COMM_SELF,''tao_lmvm'',&tao);
TaoApplicationCreate(PETSC_COMM_SELF,&app);
TaoAppSetInitialSolutionVec(app,x);
TaoAppSetObjectiveRoutine(app, FormFunction,(void *)&user);
TaoAppSetGradientRoutine(app,FormGradient,(void *)&user);
TaoAppSetHessianMat(app,H,H);
TaoAppSetHessianRoutine(app,FormHessian,(void *)&user);
TaoSolveApplication(app,tao);
VecView(x,PETSC_VIEWER_STDOUT_SELF);
```