Programming Math in Java

Lessons from Apache Commons Math

Phil Steitz

psteitz@apache.org

Apachecon North America 2015

April 14, 2015

Agenda

- Commons Math Overview
- Examples and problems
- Lessons learned
- Getting involved

Disclaimer

Apache Commons Math is a community-based, open development project.

The discussion of problems, challenges and lessons learned represents one contributor's views.

Goals

- Self-contained, ASL-licensed Java
- Provide simple solutions to common problems
- Implement standard, well-documented algorithms
- Balance ease of use and good design
- Stay well-tested and maintained
- Maintain a strong community

Some Statistics (as of March 11, 2015)...

- 67 packages
  - 908 classes
  - 84,057 lines of code
- 54,629 lines of comments
  - 99.7% documented API
- 5862 unit tests
  - 92% of code covered
- 80 open issues
  - Average 216 issues resolved per year
- 6 active committers
- 70+ contributors
- 0 dependencies

Quick Tour

Building Blocks - util, special, complex packages

- FastMath (pure java replacement for java.lang.Math)
- Dfp (arbitrary precision arithmetic)
- Continued fractions
- Special functions (Beta, Gamma, Bessel, Erf)
- Array utilities (norms, normalization, filling, etc)
- Basic combinatorics ($\binom{n}{k}$, $n!$, Stirling numbers...)
- Basic integer arithmetic (checked ops, gcd, lcm, primality...)
- Complex numbers

Linear Algebra - linear package

- Vector and Matrix operations (dense, sparse, field)
- Decompositions (LU, QR, Cholesky, SVD, Eigenvalue)
- Solvers

Basic Numerical Analysis - analysis package

- Automatic Differentiation
- Numerical Integration
- Interpolation
- Root finders

Probability and Statistics - distributions, stat and random packages

- Probability distributions (Gaussian, Exponential, Poisson...)
- Random number generators (Well, ISAAC, Mersenne twister...)
- Random data generators (samplers, random vectors...)
- Univariate statistics (storeless and in-memory)
- Regression (matrix-based and storeless)
- Correlation
- Inference (T-, G-, ChiSquare, Kolmogorov-Smirnov tests...)

Optimization and Geometry - optim, fitting, ode, transform, geometry packages

- Fitting (non-linear least-squares, curve fitting)
- Ordinary differential equations (ODE)
- Linear/Non-linear optimization
- Computational Geometry
- Fast Fourier transform

AI / Machine learning - genetics, filter and ml packages

- Genetic algorithms
- Neural networks
- Clustering
- Kalman filter

Examples and Challenges

Example 1 - Root finding

Find all roots of the polynomial

$$p(x) = x^5 + 4x^3 + x^2 + 4 = (x + 1)(x^2 - x + 1)(x^2 + 4)$$

```
double[] coefficients = { 4.0, 0.0, 1.0, 4.0, 0.0, 1.0 };
LaguerreSolver solver = new LaguerreSolver();
Complex[] result = solver.solveAllComplex(coefficients, 0);
```

Example - Root finding (cont)

How about this one?

$$p(x) = \prod_{i=1}^{50} 15,000(i + ix + ix^2 + ix^3)$$

As of 3.4.1, this will appear to hang.
Two problems:

1. Complex iterates "escape to NaN"
2. Default max function evaluations is set to Integer.MAX_VALUE

Example 1 - Root finding (cont)

First problem: escape to NaN

- Long and painful history debating C99x Annex G
- Desire: clear, self-contained, open documentation
- Desire: simple, performant code
- Result: use standard definitions and let NaNs propagate

Second problem: hanging

- Ongoing debate about what to make defaults, whether to have defaults at all...
- Ease of use vs foot-shooting potential
- OO vs procedural / struct / primitives design

Example 2 - Kolmogorov-Smirnov Test

Problem: Given two samples S_1 and S_2, are they drawn from the same underlying probability distribution?

```java
// load sample data into double[] arrays
double[] s1 = ...
double[] s2 = ...
KolmogorovSmirnovTest test = new KolmogorovSmirnovTest();

// Compute p-value for test
double pValue = test.kolmogorovSmirnovTest(s1, s2);
```

K-S Test Implementation Challenges

Test statistic $= D_{m,n} =$ maximum difference between the empirical distribution functions of the two samples.

- The distribution of $D_{m,n}$ is asymptotically an easy-to-compute distribution
- For very small m, n, the distribution can be computed exactly, but expensively
- What to do for mid-size samples $(100 < m \times n < 10,000)$?
- Should this distribution be in the distributions package?
- Reference data / correctness very hard to verify

Example 3 - Multivariate optimization

Find the minimum value of $100(y - x^2)^2 + (1 - x)^2$, starting at the point $(-1, 1)$

```
MultivariateFunction f = new MultivariateFunction() {
    public double value(double[] x) {
        double xs = x[0] * x[0];
        return (100 * FastMath.pow((x[1] -  xs), 2)) +
           FastMath.pow(1 - x[0], 2);
    }
};
PowellOptimizer optim = new PowellOptimizer(1E-13, 1e-40);
PointValuePair result = optim.optimize(new MaxEval(1000),
                        new ObjectiveFunction(f),
                        GoalType.MINIMIZE,
                        new InitialGuess(new double[] {-1, 1}));
```

Works OK in this case, but...

Example 3 - Optimization (cont)

User question: How do I choose among the available optimizers?
Developer question: How do we define a consistent API?

Another solution for Example 3:

```
BOBYQAOptimizer optim = new BOBYQAOptimizer(4);
PointValuePair result = optim.optimize(new MaxEval(1000),
                        new ObjectiveFunction(f),
                        GoalType.MINIMIZE,
                        new InitialGuess(new double[] {-1, 1}),
                        SimpleBounds.unbounded(2));
```

Note additional parameter to optimize.

Example 3 - Optimization (cont)

To allow arbitrary parameter lists, we settled on varargs ...

```
public abstract class BaseOptimizer<PAIR>
...
/**
 * Stores data and performs the optimization.
 *
 * The list of parameters is open-ended so that sub-classes can extend it
 * with arguments specific to their concrete implementations.
 *
 * When the method is called multiple times, instance data is overwritten
 * only when actually present in the list of arguments: when not specified,
 * data set in a previous call is retained (and thus is optional in
 * subsequent calls).
...
 * @param optData Optimization data.
...
 */
public PAIR optimize(OptimizationData... optData)
```

Optimization (cont) - version 4 direction

Refactor all classes in "o.a.c.m.optim"

- Phase out "OptimizationData"
- Use "fluent" API
- Separate optimization problem from optimization procedure

Experimentation has started with least squares optimizers:

- LevenbergMarquardtOptimizer
- GaussNewtonOptimizer

Optimization (cont) - version 4 direction

New classes/interfaces in "o.a.c.m.fitting.leastsquares":

- LeastSquaresProblem and LeastSquaresProblem.Evaluation
- LeastSquaresOptimizer and LeastSquaresOptimizer.Optimum
- LeastSquaresFactory and LeastSquaresBuilder

Usage:

```
LeastSquaresProblem problem = LeastSquaresFactory.create( /* ... */ );
LeastSquaresOptimizer optim = new LevenbergMarquardtOptimizer( /* ... */ );
Optimum result = optim.optimize(problem);
```

Yet to be done: Refactor all the other optimizers.

Example 3 - Optimization (cont)

Some additional challenges:

- Stochastic algorithms are tricky to test
- FORTRAN ports create awful Java code, but "fixing" can create discrepancies
- BOBYQAOptimizer port has never really been fully supported
- Answering user questions requires expertise and time

Message from our sponsors...

Come join us!

- Code is interesting!
- Math is interesting!
- Bugs are interesting!
- People are interesting!

Come join us!

http://commons.apache.org/math

Example 4 - Aggregated statistics

Problem: Aggregate summary statistics across multiple processes

```
// Create a collection of stats collectors
Collection<SummaryStatistics> aggregate =
     new ArrayList<SummaryStatistics>();

// Add a collector, hand it out to some process
// and gather some data...
SummaryStatistics procStats = new SummaryStatistics();
aggregate.add(procStats);
procStats.addValue(wildValue);
...
// Aggregate results
StatisticalSummary aggregatedStats =
     AggregateSummaryStatistics.aggregate(aggregate);
```

- SummaryStatistics instances handle large data streams
- StatisticalSummary is reporting interface

Question: how to distribute instances / workload?

Threading and workload management

Questions:

1. Should we implement multi-threaded algorithms?
2. How can we be more friendly for Hadoop / other distributed compute environments?

- So far (as of 3.4.1), we have not done 1.
- So far, answer to 2 has been "leave it to users."

Could be both answers are wrong...

Example 5 - Genetic algorithm

Another example where concurrent execution would be natural...

```java
// initialize a new genetic algorithm
GeneticAlgorithm ga = new GeneticAlgorithm(
            new OnePointCrossover<Integer>(),
            CROSSOVER_RATE,
            new BinaryMutation(),
            MUTATION_RATE,
            new TournamentSelection(TOURNAMENT_ARITY)
);

// initial population
Population initial = getPopulation();

// stopping conditions
StoppingCondition stopCond = new FixedGenerationCount(NUM_GENERATIONS);

// run the algorithm
Population finalPopulation = ga.evolve(initial, stopCond);

// best chromosome from the final population
Chromosome bestFinal = finalPopulation.getFittestChromosome();
```

Question: how to distribute "evolve" workload?

Example 6 - OLS regression

Three ways to do it

OLSMultipleRegression (stat.regression) in-memory

MillerUpdatingRegression (stat.regression) streaming

LevenbergMarquardtOptimizer (fitting.leastsquares) in-memory

1. How to make sure users discover the right solution for them?
2. How much dependency / reuse / common structure to force?

Example 6 - OLS regression (cont)

Simplest, most common:

```
OLSMultipleLinearRegression regression = new OLSMultipleLinearRegression();

// Load data
double[] y = new double[]{11.0, 12.0, 13.0, 14.0, 15.0, 16.0};
double[] x = new double[6][];
x[0] = new double[]{0, 0, 0, 0, 0};
x[1] = new double[]{2.0, 0, 0, 0, 0};
...
x[5] = new double[]{0, 0, 0, 0, 6.0};
regression.newSample(y, x);

// Estimate model
double[] beta = regression.estimateRegressionParameters();
double[] residuals = regression.estimateResiduals();
double rSquared = regression.calculateRSquared();
double sigma = regression.estimateRegressionStandardError();
```

Question: What happens if design matrix is singular?

Example 6 - OLS regression (cont)

Answer: if it is "exactly" singular, `SingularMatrixException`

If it is near-singular, garbage out
So make singularity threshold configurable:

```
OLSMultipleLinearRegression regression =
     new OLSMultipleLinearRegression(threshold);
```

- What exactly is this threshold?
- What should exception error message say?
- How to doc it?
- What if we change underlying impl?

Message from our sponsors...

Come join us!

- Code is interesting!
- Math is interesting!
- Bugs are interesting!
- People are interesting!

Come join us!

http://commons.apache.org/math

Example 7 - Rank correlation

Given two double arrays x[] and y[], we want a statistical measure of how well their rank orderings match.

Use Spearman's rank correlation coefficient.

```
double[] x = ...
double[] y = ...
SpearmansCorrelation corr = new SpearmansCorrelation();
corr.correlation(x, y);
```

How are ties in the data handled?
What if one of the arrays contains NaNs?

Example 7 - Rank correlation (cont)

- As of 3.4.1, by default ties are averaged and NaNs cause NotANumberException
- Both are configurable via RankingAlgorithm constructor argument
- RankingAlgorithm is a ranking implementation plus
  - TiesStrategy - what ranks to assign to tied values
  - NaNStrategy - what to do with NaN values
- Default in Spearman's is natural ranking on doubles with ties averaged and NaNs disallowed

Example 7 - Rank correlation (cont)

NaNStrategies:

- MINIMAL - NaNs are treated as minimal in the ordering
- MAXIMAL - NaNs are treated as maximal in the ordering
- REMOVED - NaNs are removed
- FIXED - NaNs are left "in place"
- FAILED - NaNs cause NotANumberException

Problems:

1. What if user supplies a NaturalRanking with the NaN strategy that says remove NaNs?
2. How to doc clearly and completely?

Example 8 - Random Numbers

Commons Math provides alternatives to the JDK PRNG and a pluggable framework

- RandomGenerator interface is like j.u.Random
- All random data generation within Commons Math is pluggable

Low-level example:

```
// Allocate an array to hold random bytes
byte[] bytes = new byte[20];

// Instantiate a Well generator with seed = 100;
Well19937c gen = new Well19937c(100);

// Fill the array with random bytes from the generator
gen.nextBytes(bytes);
```

Example 8 - Random Numbers (cont)

High-level example:

```java
// Create a RandomDataGenerator using a Mersenne Twister
RandomDataGenerator gen =
     new RandomDataGenerator(new MersenneTwister());

// Sample a random value from the Poisson(2) distribution
long dev = gen.nextPoisson(2);
```

Problems:

1. Well generators (used as defaults a lot) have some initialization overhead. How / when to take this hit?
2. Distribution classes also have sample() methods, but that makes them dependent on generators

Example 9 - Linear Programming

Maximize $7x_1 + 3x_2$ subject to
$3x_1 - 5x_3 \leq 0$
$2x_1 - 5x_4 \leq 0$
... (more constraints)

```
LinearObjectiveFunction f =
  new LinearObjectiveFunction(new double[] { 7, 3, 0, 0 }, 0 );

List<LinearConstraint> constraints = new ArrayList<LinearConstraint>();
constraints.add( new LinearConstraint(new double[] { 3, 0, -5, 0 },
                                      Relationship.LEQ, 0.0));
constraints.add(new LinearConstraint(new double[] { 2, 0, 0, -5 },
                                      Relationship.LEQ, 0.0));
...
SimplexSolver solver = new SimplexSolver();
PointValuePair solution = solver.optimize(new MaxIter(100), f,
                          new LinearConstraintSet(constraints),
                          GoalType.MAXIMIZE,
                          new NonNegativeConstraint(true));
```

Example 9 - Linear Programming (cont)

Question: What happens if the algorithm does not converge?
Answer: `TooManyIterationsException`

What if I want to know the last `PointValuePair`?

```
SolutionCallback callback = new SolutionCallback();
try {
    PointValuePair solution = solver.optimize(new MaxIter(100), f,
                          new LinearConstraintSet(constraints),
                          GoalType.MAXIMIZE,
                          new NonNegativeConstraint(true), callback);
} catch (TooManyIterationsException ex) {
PointValuePair lastIterate = callback.getSolution();
}
```

(Rejected) Alternatives:

1. Don't throw, but embed some kind of status object in return
2. Shove context data in exception messages

Message from our sponsors...

Come join us!

- Code is interesting!
- Math is interesting!
- Bugs are interesting!
- People are interesting!

Come join us!

http://commons.apache.org/math

Example 10 - Geometry

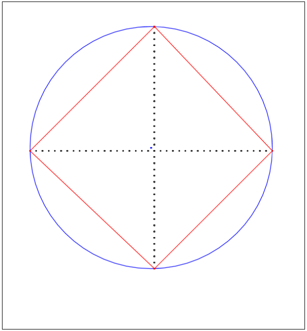
Calculate the convex hull and enclosing ball of a set of 2D points:

```
List<Vector2D> points = ...
ConvexHullGenerator2D generator = new MonotoneChain(true, 1e-6);
ConvexHull2D hull = generator.generate(points);

Encloser<Euclidean2D, Vector2D> encloser =
  new WelzlEncloser<Euclidean2D, Vector2D>(1e-6, new DiskGenerator());
EnclosingBall<Euclidean2D, Vector2D> ball = encloser.enclose(points);
```

Problems:

1. algorithms are prone to numerical problems
2. important to specify a meaningful tolerance, but depends on input data
3. does it make sense to support a default tolerance?



Lessons Learned

Key Challenges Recap...

- Balancing performance, mathematical correctness and algorithm fidelity
- Fully documenting behavior and API contracts
- Steering users toward good solutions as simply as possible
- Balancing OO design and internal reuse with approachability
- Obscure and abandoned contributions
- Reference implementations and / or data for validation
- Backward compatibility vs API improvement

Some lessons learned

- Let practical use cases drive performance / accuracy / fidelity decisions
- Be careful with ports and large contributions
- Limit public API to what users need
- Prefer standard definitions and algorithms
- Take time to fully research bugs and patches
- Constantly improve javadoc and User Guide
- Try to avoid compatibility breaks, but bundle into major version releases when you have to

Get Involved!

Using Apache Commons Math

Maven:

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-math3</artifactId>
    <version>3.4.1</version>
</dependency>
```

Download:

http://commons.apache.org/math/download_math.cgi

Watch for new versions!

Links

- Project homepage: http://commons.apache.org/math/
- Issue tracker:
  https://issues.apache.org/jira/browse/MATH
- Mailinglists: dev@commons.apache.org &
  user@commons.apache.org
  e-mail subject: [math]
- Wiki: http://wiki.apache.org/commons/MathWishList

How to contribute?

- Check out the user guide
  http://commons.apache.org/math/userguide
- Ask - and answer - questions on the user mailing list
- Participate in discussions on the dev list
- Create bug reports / feature requests / improvements in JIRA
- Create and submit patches (code, documentation, examples)
- Look at
  http://commons.apache.org/math/developers.html
- Run `mvn site` and review CheckStyle, Findbugs reports when creating patches
- Don't be shy asking for help with maven, git, coding style, math ... anything

- Questions?