Mathematical methods compendium

Classes

`com.google.common.math.DoubleMath`
- Knuth's mean – likely eclipsed by `DoubleStat.mean` and the likes
- factorial – likely eclipsed by `CombinatoricsUtil.factorial`

`info.yeppp.Core`
- very fast elementwise arithmetics
- very fast summing (incl. absolute and square summing), albeit without precision enhancement

`info.yeppp.Math`
- very fast elementwise mathematics

`org.apache.commons.math3.special.Gamma`
- gamma, digamma and trigamma functions

`org.apache.commons.math3.util.CombinatoricsUtil`
- binomial coefficients and factorial

`org.apache.commons.math3.util.MathArrays`
- sort, shuffle, copy
- calculate L2 norm and L1, L2, Linf distance
- check order and sign (for Comparables and doubles)
- check equality
- normalization (without precision compensation, but with safeguards for NaNs and Infs); cf. `NormalizationUtil`

`org.jetbrains.bio.jni.DoubleStat`
- reasonably fast and precise statistics
- non-weighted versions partially eclipsed by `info.yeppp.Core`: slower but more precise

`org.jetbrains.bio.jni.SIMDMath`
- reasonably fast elementwise mathematics, incl. logAddExp and logSumExp
- other methods in all likeness completely eclipsed by `info.yeppp.Math`

`org.jetbrains.bio.statistics.stat.IntStat`
- statistics for int arrays

`org.jetbrains.bio.statistics.stat.ShortStat`
- statistics for short arrays

`org.jetbrains.bio.util.MathUtil`
- logAddExp, log1pexp, logAbsDiffExp, logit, sigmoid
- logFactorial produces an array of log-factorials; for single value use `CombinatoricsUtil.factorialLog`

`org.jetbrains.bio.util.collections.ArrayUtil`
- create a double array with a given initial value
- apply arbitrary unary [binary] operation to a double array[s]
- specialization for exp and log eclipsed by `yeppp.Math` and `SIMDMath`
- prefix sum for ints

`org.jetbrains.bio.util.collections.NormalizationUtil`
- [log-]rescaling with precision compensation; similar to `MathArrays.normalizeArray`

`org.jetbrains.bio.util.collections.PrimitiveArrays`
- filter, reorder, reverse, merge, sorted, isSorted (for ints)
- elementwise division for short arrays
- max index, median

Methods

Array to array (reorder, filter)

`MathArrays.sortInPlace`
`MathArrays.copyOf`
`MathArrays.shuffle`
`PrimitiveArrays.filter`
`PrimitiveArrays.reorder`
`PrimitiveArrays.reverse`
`PrimitiveArrays.sorted`

Array to array (operations)

Elementwise

`ArrayUtil.transform[InPlace]` – applies arbitrary elementwise unary operation to a double array
`ArrayUtil.exp[InPlace]` – semantically equivalent to `yeppp.Math.Exp*` and `SIMDMath.exp`
`ArrayUtil.log[InPlace]` – semantically equivalent to `yeppp.Math.Log*` and `SIMDMath.log`
`SIMDMath.exp` – semantically equivalent to `ArrayUtil.exp` and `yeppp.Math.Exp*`
`SIMDMath.log` – semantically equivalent to `ArrayUtil.log` and `yeppp.Math.Log*`
`yeppp.Core.Negate*`
`yeppp.Math.Exp*` – semantically equivalent to `ArrayUtil.exp` and `SIMDMath.exp`
`yeppp.Math.Log*` – semantically equivalent to `ArrayUtil.log` and `SIMDMath.log`
`yeppp.Math.Sin*`
`yeppp.Math.Cos*`
`yeppp.Math.Tan*`

Prefix sum

`ArrayUtil.cumulativeSum` – for integers
`DoubleStat.prefixSum` – for doubles, with Kahan-like precision enhancement

Normalize, rescale

`MathArrays.normalizeArray` – scales an array to sum to an arbitrary number; skips NaNs, throws on infinities, no precision compensation
`MathArrays.scale[InPlace]` – multiplies array elements by a given number
`NormalizationUtil.[log]Rescale` – scales an array so that its sum [resp. logSumExp]

becomes 1 [resp. 0]; non-log variant includes precision compensation
`NormalizationUtil.rescaleSigmoid`

Arrays to array (elementwise)

`ArrayUtil.combine` – applies arbitrary elementwise binary operation to two double arrays

Arithmetic

`MathArrays.ebeAdd` – for two double arrays
`MathArrays.ebeSubtract` – for two double arrays
`MathArrays.ebeMultiply` – for two double arrays
`MathArrays.ebeDivide` – for two double arrays
`PrimitiveArrays.ebeDivide` – for two short arrays
`yeppp.Core.Add*` – for two arrays of any primitive type
`yeppp.Core.Subtract*` – for two arrays of any primitive type
`yeppp.Core.Multiply*` – for two arrays of any primitive type
`yeppp.Core.Min*` – for two arrays of any primitive type
`yeppp.Core.Max*` – for two arrays of any primitive type

Arrays to array (concatenate, merge)

`PrimitiveArrays.merge` – concatenates two arrays of same primitive type discarding the duplicates from the second array; throws if duplicates are present in the first array

Array to number (reduction)

`MathArrays.safeNorm` – L2 double array norm with over- and underflow protection

Statistics

`DoubleStat.sum` – semantically equivalent to `yeppp.Core.Sum*`; enhanced precision due to balanced summing
`DoubleStat.mean`
`DoubleStat.standardDeviation`
`DoubleMath.mean` – implemented for double, int and long arrays and number iterators; employs Knuth's algorithm to avoid calculating sum separately
`yeppp.Core.Min*`
`yeppp.Core.Max*`
`yeppp.Core.Sum*` – semantically equivalent to `DoubleStat.sum`; no precision enhancement
`yeppp.Core.SumAbs*`
`yeppp.Core.SumSquares*`
`PrimitiveArrays.maxIndex`
`PrimitiveArrays.median`
`IntStat.*`
`ShortStat.*`

Array to boolean (property)

Order

`MathArrays.isMonotonic` – for Comparables and doubles

`MathArrays.checkOrder` – for doubles

`PrimitiveArrays.isSorted` – for ints

Sign

`MathArrays.checkPositive`

`MathArrays.checkNonNegative`

Arrays to boolean

`MathArrays.equals`
`MathArrays.equalsIncludingNaN`

Arrays to number

Dot product

`MathArrays.linearCombination` – essentially dot product, so semantically equivalent to `DoubleStat.weightedSum` and `yeppp.Core.DotProduct*`; has enhanced accuracy

`yeppp.Core.DotProduct*` – semantically equivalent to `DoubleStat.weightedSum` and `MathArrays.linearCombination`; straightforward

Distance

`MathArrays.distance[1|Inf]` – L1, L2 and L-inf array distances for doubles and ints; straightforward

Statistics

`DoubleStat.weightedSum` – essentially dot product, so semantically equivalent to `MathArrays.linearCombination` and `yeppp.Core.DotProduct*`; employs balanced summing for accuracy enhancement

`DoubleStat.weightedMean`
`DoubleStat.weightedStandardDeviation`

Number(s) to number

`MathUtil.logAddExp`

`MathUtil.log1pexp`

`MathUtil.logAbsDiffExp`

`MathUtil.logit`

`MathUtil.sigmoid`

`NormalizationUtil.normalize` – translates a point on an interval linearly on another interval

`Gamma.logGamma`

Gamma.gamma
Gamma.digamma
Gamma.trigamma
DoubleMath.log2
DoubleMath.factorial
CombinatoricsUtil.binomialCoefficient[Double|log]
CombinatoricsUtil.factorial[Double|Log]
CombinatoricsUtil.stirlingS2

Number to array

MathUtil.logFactorial
ArrayUtil.create

Matrices

MatrixUtil.create
MatrixUtil.copy
MatrixUtil.fill
MatrixUtil.exp[InPlace]
MatrixUtil.log[InPlace]
MatrixUtil.transpose