

PROVEN TECHNOLOGY IN BUSINESS-CRITICAL ENVIRONMENTS


VEGA™ 3

- 5x higher performance since 1st generation
- Ultimate platform for Extreme Transaction Processing
- Provides predictable and low response times
- Pauseless garbage collection up to 670 GB memory heaps
- DirectPath for accelerating inter-VM bandwidth & latency by 100x
- RealTime Performance Monitoring in production environments


Azul Compute Appliances

OVERVIEW

New! Vega 3 7300 Series



Vega 3 3300 Series



PERFORMANCE

Vega 3 Series Compute Appliances Record-Breaking Performance

The Azul Vega™ 3 7300 and 3300 Series compute appliances offer an unprecedented amount of highly scalable compute capacity for Java-based applications, with up to 864 processor cores and 768 GB of memory in a flat SMP configuration. The unique software and hardware-assisted features of the Azul solution enable applications to transparently tap into this capacity over the network and reach unprecedented levels of scalability, throughput, and performance consistency. With this production-proven architecture the Azul Vega 3 Series has set the performance record on industry standard SPECjbb2005 and XMLMark tests.

The Vega 3 Series appliances advance the unique business benefits delivered by the Azul solution, and are the fastest path to maximizing the ROI of business-critical applications. Without changing application

VEGA 3 SPECIFICATIONS

CONTACT US

Want to learn more on how Azul Compute Appliances can help your organization? [Contact us today](#) or call 650.230.6650.

LIKE THIS PAGE?

[Add to del.icio.us »](#)

[Digg this! »](#)

[RSS](#)

NEED TO KNOW!

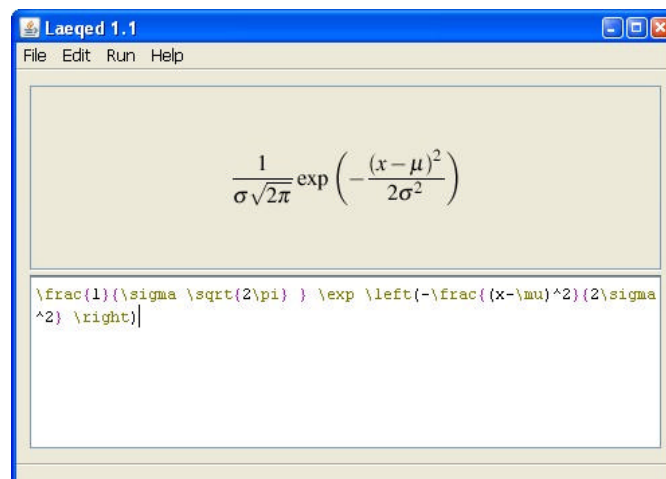
[What is Network Attached Processing? »](#)

22.01.200921.01.2009

JUG Zurich -JQuantLib

1

Laeqed



```
/**
 * Provides the probability density function (pdf) of the (unit) normal distribution
 *
 * {@latex[
 * pdf = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(x-\mu)^2}{2\sigma^2} \right)
 * ]}
 *
 * */
```

22.01.200921.01.2009

JUG Zurich -JQuantLib

2

Klasse:**C++:**

```
class GammaDistribution : public std::unary_function<Real,Real> {...}
```

Java:

```
public class GammaDistribution implements UnaryFunctionDouble {...}
```

Konstruktor:**C++:**

```
GammaDistribution(Real a) : a_(a) {
    QL_REQUIRE(a>0.0, "invalid parameter for gamma distribution");
}
```

Java:

```
public GammaDistribution(double a) {
    a_ = a;

    if (a_ < 0.0) {
        throw new ArithmeticException("invalid parameter for gamma distribution");
    }
}
```

UnaryFunctionDouble.java

```
public interface UnaryFunctionDouble {

    /**
     * Computes the value of the function; f(x)
     *
     * @param x
     * @return f(x)
     */
    public double evaluate(double x);
```

BinaryFunctionDouble.java

```
public interface BinaryFunctionDouble {

    /**
     * Computes the value of the function; f(x, y)
     *
     * @param x
     * @param y
     * @return f(x, y)
     */
    public double evaluate(double x, double y);
```

```

Real GammaDistribution::operator()(Real x) const {
    if (x <= 0.0) return 0.0;

    Real gln = GammaFunction().logValue(a_);

    if (x < (a_+1.0)) {
        Real ap = a_;
        Real del = 1.0/a_;
        Real sum = del;
        for (Size n=1; n<=100; n++) {
            ap += 1.0;
            del *= x/ap;
            sum += del;
            if (std::fabs(del) < std::fabs(sum)*3.0e-7)
                return sum*std::exp(-x + a_*std::log(x) - gln);
        }
    } else {
        Real b = x + 1.0 - a_;
        Real c = QL_MAX_REAL;
        Real d = 1.0/b;
        Real h = d;
        for (Size n=1; n<=100; n++) {
            Real an = -1.0*n*(n-a_);
            b += 2.0;
            d = an*d + b;
            if (std::fabs(d) < QL_EPSILON) d = QL_EPSILON;
            c = b + an/c;
            if (std::fabs(c) < QL_EPSILON) c = QL_EPSILON;
            d = 1.0/d;
            Real del = d*c;
            h *= del;
            if (std::fabs(del - 1.0) < QL_EPSILON)
                return h*std::exp(-x + a_*std::log(x) - gln);
        }
    }
    QL_FAIL("too few iterations");
}

```

```

public double evaluate(double x) /* Read-only */ {
    if (x <= 0.0) {
        return 0.0;
    }

    GammaFunction gf = new GammaFunction();
    double gln = gf.logValue(a_);

    if (x < (a_ + 1.0)) {
        double ap = a_;
        double del = 1.0 / a_;
        double sum = del;
        for (int n = 1; n <= 100; n++) {
            ap += 1.0;
            del *= x / ap;
            sum += del;
            if (Math.abs(del) < Math.abs(sum) * 3.0e-7) {
                return sum * Math.exp(-x + a_ *
                    Math.log(x) - gln);
            }
        }
    } else {
        double b = x + 1.0 - a_;
        double c = Constants.QL_MAX_REAL;
        double d = 1.0 / b;
        double h = d;
        for (int n = 1; n <= 100; n++) {
            double an = -1.0 * n * (n - a_);
            b += 2.0;
            d = an * d + b;

            if (Math.abs(d) < Constants.QL_EPSILON) {
                d = Constants.QL_EPSILON;
            }
            c = b + an / c;
            if (Math.abs(c) < Constants.QL_EPSILON) {
                c = Constants.QL_EPSILON;
            }
            d = 1.0 / d;
            double del = d * c;
            h *= del;
            if (Math.abs(del - 1.0) < Constants.QL_EPSILON) {
                return h * Math.exp(-x + a_ * Math.log(x) - gln);
            }
        }
    }
    throw new ArithmeticException("too few iterations");
}



```

Numerical Recipes in C - Microsoft Internet Explorer provided by 3M/IE 6.0

File Edit View Favorites Tools Help

Back Forward Stop Reload Home Search Favorites




Address <http://www.nrbook.com/a/bookcpdf.php>





Important: Effective 1/1/07 (6/15/07 for Linux), you will need the free FileOpen plugin for Adobe [Acrobat] Reader® to use this free resource. If you are getting blank pages, or error boxes, instead of Numerical Recipes chapter sections, then you probably have not installed the plugin. The plugin is available for Windows, Mac OSX, and Linux. Please see: [How to download and install the plugin](#)

Numerical Recipes in C, Second Edition (1992)






Obsolete edition, no longer supported. Please consider using the much-expanded and improved Third Edition (2007) in C++.

-  Front Matter, Contents, and Prefaces xi
-  Legal Matters xvi
-  Computer Programs by Chapter and Section xix

1 Preliminaries

-  1.0 Introduction 1
-  1.1 Program Organization and Control Structures 5
-  1.2 Some C Conventions for Scientific Computing 15
-  1.3 Error, Accuracy, and Stability 15

2 Solution of Linear Algebraic Equations

-  2.0 Introduction 32
-  2.1 Gauss-Jordan Elimination 36
-  2.2 Gaussian Elimination with Backsubstitution 41
-  2.3 LU Decomposition and Its Applications 43
-  2.4 Tridiagonal and Band Diagonal Systems of Equations 50

GammaDistribution (JQuantLib v0.1.2-SNAPSHOT API Javadocs) - Microsoft Internet Explorer provided by 3M/IE 6.0

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Print View Source

Address http://www.jquantlib.org/maven2/sites/jquantlib/apidocs/index.html Go Links Context Org

JQuantLib v0.1.2-SNAPSHOT API Javadocs

All Classes

Packages

- org.jquantlib
- org.jquantlib.assets
- org.jquantlib.cashflow
- org.jquantlib.daycounters
- org.jquantlib.examples
- org.jquantlib.exercise
- org.jquantlib.experimental.collection

FDShoutCondition

FDStepConditionEngine

FDVanillaEngine

FiniteDifferenceModel

Finland

FixedRateBondHelper

FlatExtrapolator2D

FlatForward

ForwardFlat

ForwardFlatInterpolation

ForwardIterator

ForwardRateStructure

FrankfurtStockExchangeCalendar

FraRateHelper

Frequency

FunctionDate

FunctionDouble

FuturesRateHelper

GammaDistribution

GammaFunction

GapPayoff

Garch11

GarmanKlassAbstract

GarmanKlassOpenClose

GarmanKlassSigma1

GarmanKlassSigma3

GarmanKlassSigma4

Overview Package Class Use Tree Deprecated Index Help

PREV CLASS NEXT CLASS

SUMMARY: NESTED | FIELD | CONSTR | METHOD

FRAMES NO FRAMES

DETAIL: FIELD | CONSTR | METHOD

org.jquantlib.math.distributions

Class GammaDistribution

```

classDiagram
    class GammaDistribution
    class UnaryFunctionDouble {
        <<interface>>
    }
    GammaDistribution --|> UnaryFunctionDouble
  
```

java.lang.Object

└ org.jquantlib.math.distributions.GammaDistribution

All Implemented Interfaces:

UnaryFunctionDouble

public class **GammaDistribution**

extends java.lang.Object

implements [UnaryFunctionDouble](#)

The gamma distribution is a two-parameter family of continuous probability distributions.

A gamma distribution is a general type of statistical distribution that is related to the beta distribution and arises naturally in processes for which the waiting times between Poisson distributed events are relevant.

Author:

Richard Gomes, Dominik Holenstein

See Also:

[Gamma Distribution](#), [Gamma Distribution on Wolfram MathWorld](#)

Type Tokens (1)

C++

```
template <class T>
    class Klass : public T {...}
```

Java

```
public class Klass<T> extends T {...}
}
```

Type Tokens (2)

Lösung

```
abstract class VanillaOption {...}

class EuropeanOption extends VanillaOption {...}

class AmericanOption extends VanillaOption {...}

class Klass<T extends VanillaOption> {
    private T delegate;
    public Klass() {
        this.delegate = null;
    }
    try {
        delegate = (T) TypeToken.getClazz(this.getClass()).newInstance();
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}

Klass<EuropeanOption> klass = new Klass();
Klass<AmericanOption> klass = new Klass();
```

Type Tokens (3)

```
public class TypeToken {

    static public Type getType(final Class<?> klass) {
        return getType(klass, 0);
    }

    static public Type getType(final Class<?> klass, final int pos) {
        Type superclass = klass.getGenericSuperclass();
        if (superclass instanceof Class) {
            throw new IllegalArgumentException("Class should be generic");
        }
        Type[] types = ((ParameterizedType) superclass).getActualTypeArguments();
        if (pos >= types.length) {
            throw new IllegalArgumentException("Missing parameter");
        }
        return types[pos];
    }

    static public Class<?> getClazz(final Class<?> klass) {
        return getClazz(klass, 0);
    }

    static public Class<?> getClazz(final Class<?> klass, final int pos) {
        Type type = getType(klass, pos);
        return (type instanceof Class<?>) ? (Class<?>) type : (Class<?>) ((ParameterizedType)
        type).getRawType();
    }
}
```