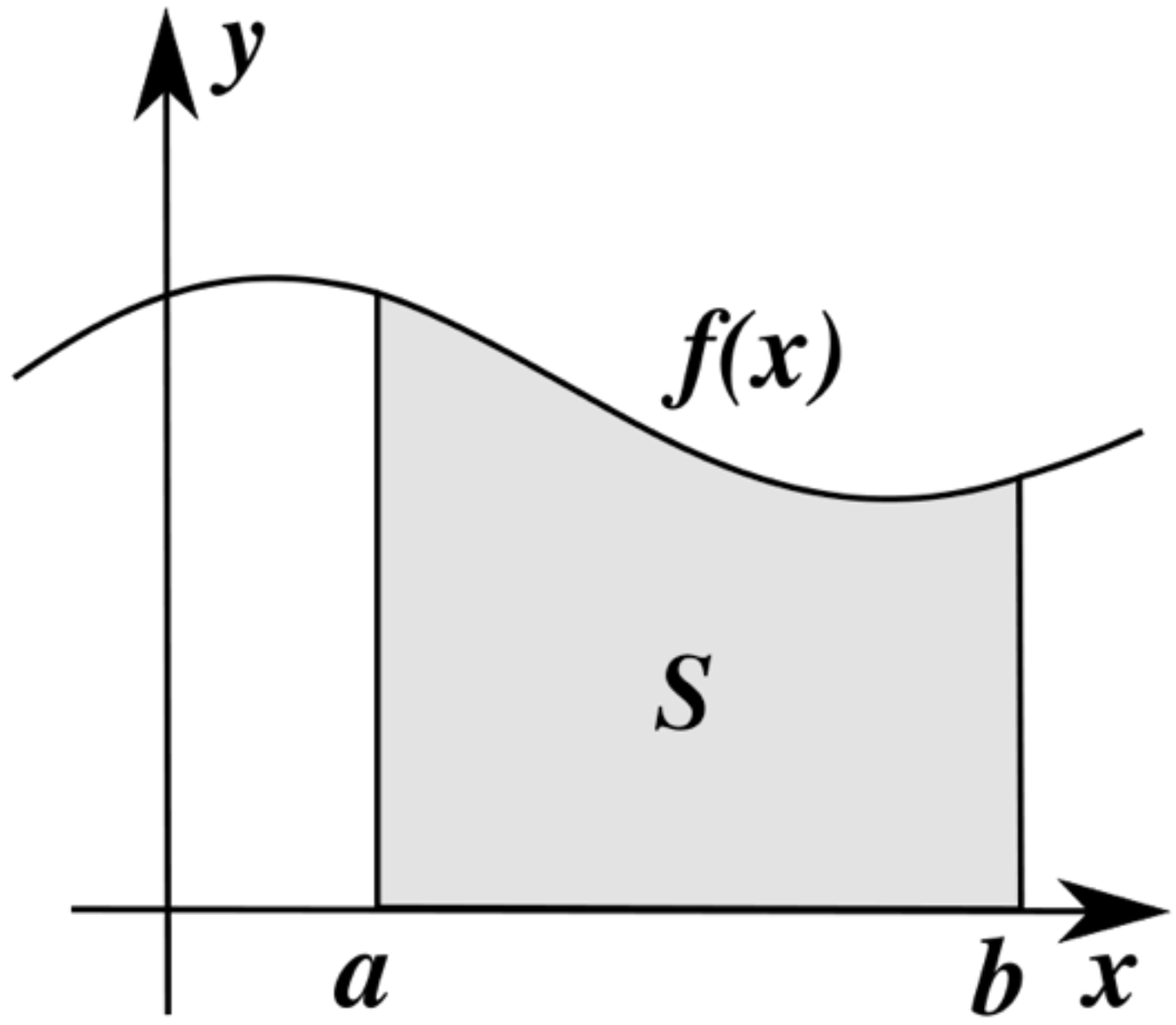
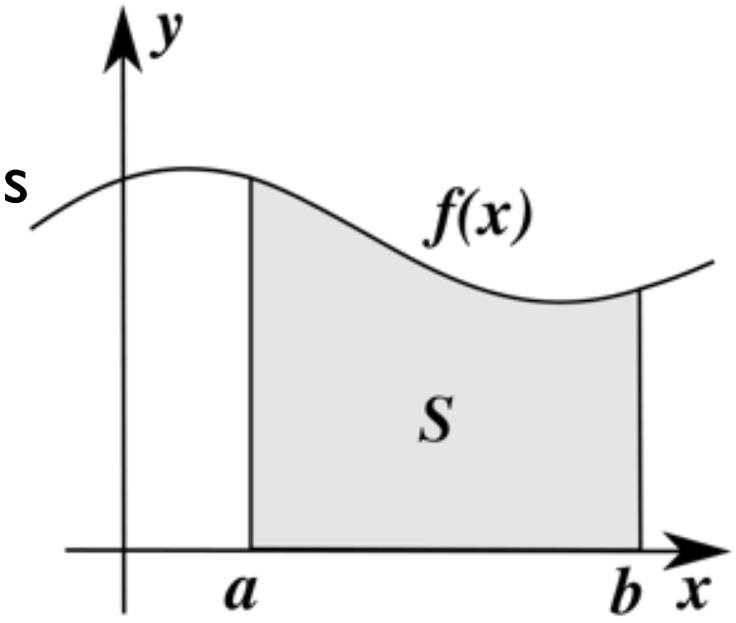
Quadrature



Quadrature



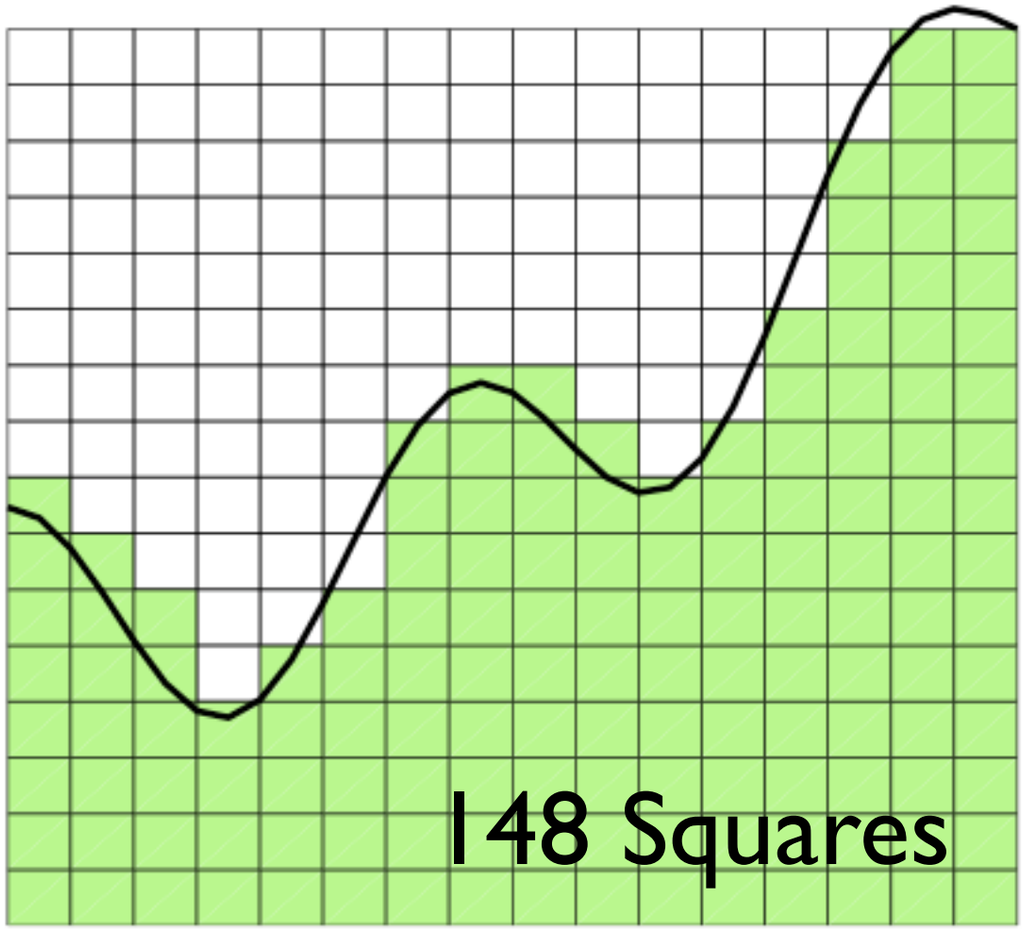
Numerical integration constitutes a broad family of algorithms for calculating the **numerical value** of a **definite integral** (sometimes the term is used to describe the numerical solution of differential equations). Quadrature is a **synonym** for **numerical integration**, especially as applied to one-dimensional integrals. Two- and higher-dimensional integration is sometimes described as cubature, although the meaning of quadrature is understood for higher dimensional integration as well.

The basic problem considered by numerical integration is to compute an **approximate solution to a definite integral**:

$$\int_a^b f(x)\,dx$$

Quadrature

Modern quadrature algorithms automatically vary an adaptive step size.



Let $f(x)$ be a real-valued function of a real variable, defined on a finite interval $a \leq x \leq b$. We seek to compute the value of the integral,

$$\int_a^b f(x)dx.$$

The word "quadrature" reminds us of an elementary technique for finding this area—plot the function on graph paper and count the number of little squares that lie underneath the curve.

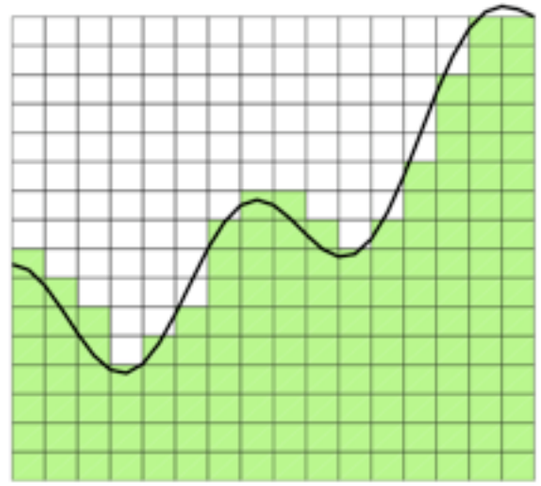
Adaptive Quadrature

Adaptive quadrature involves careful selection of the points where *f(x)* is sampled. We want to evaluate the function at **as few points as possible** while approximating the integral to **within some specified accuracy**. A fundamental additive property of a definite integral is the basis for adaptive quadrature. If c is any point between a and b, then

$$\int_a^b f(x)dx = \int_a^c f(x)dx + \int_c^b f(x)dx$$

$$\int_a^b f(x)dx = \int_a^c f(x)dx + \int_c^b f(x)dx$$

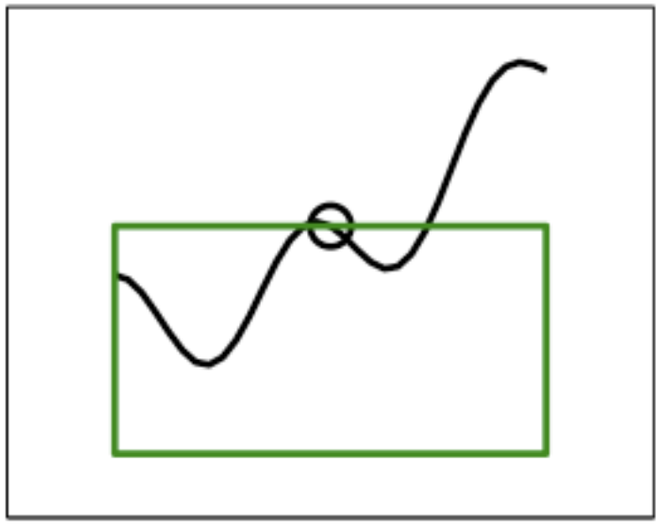
The idea is that if we can approximate each of the two integrals on the right to within a specified tolerance, then the sum gives us the desired result. If not, we can recursively apply the additive property to each of the intervals [a, c] and [c, b].



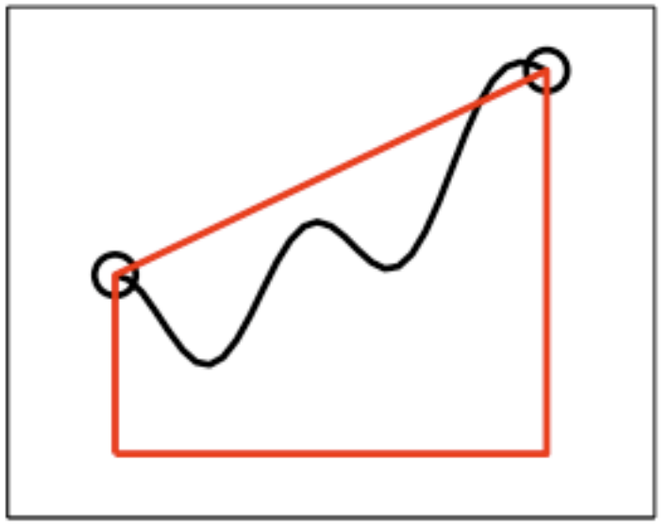
The resulting algorithm will adapt to the integrand automatically, partitioning the interval into subintervals with fine spacing where the integrand is varying rapidly and coarse spacing where the integrand is varying slowly.

Four Quadrature Rules

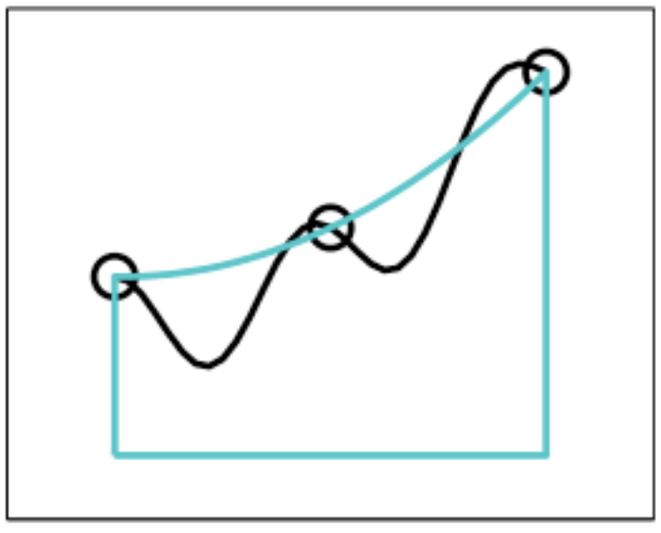
Midpoint rule



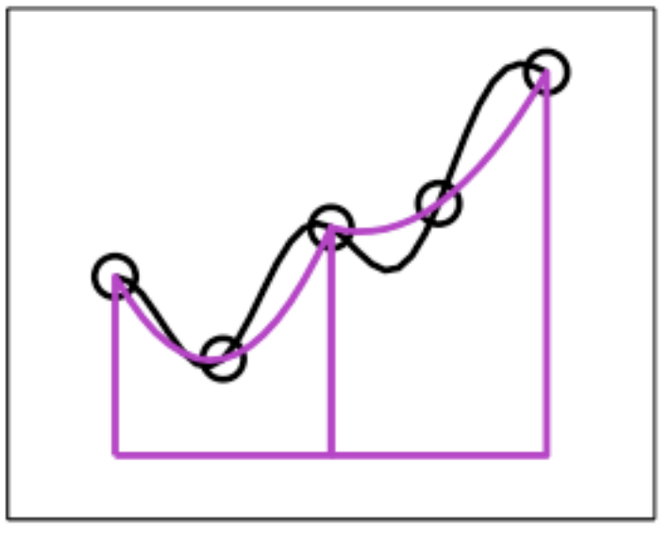
Trapezoid rule



Simpson's rule

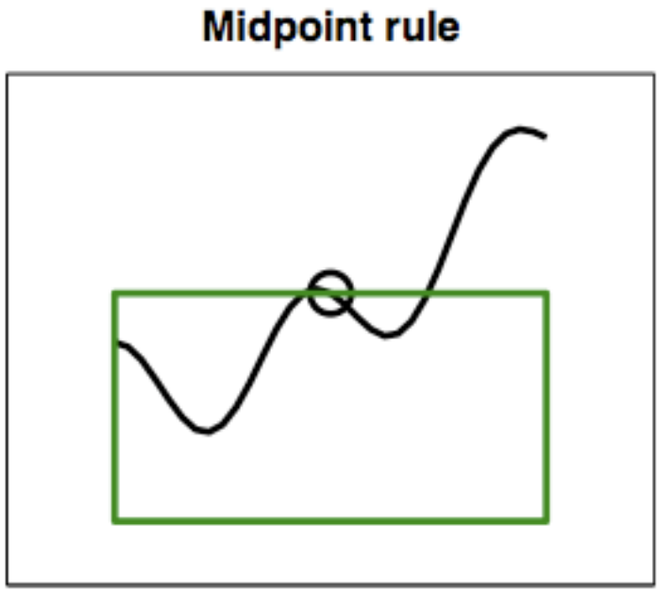


Composite Simpson's rule



The Midpoint Rule

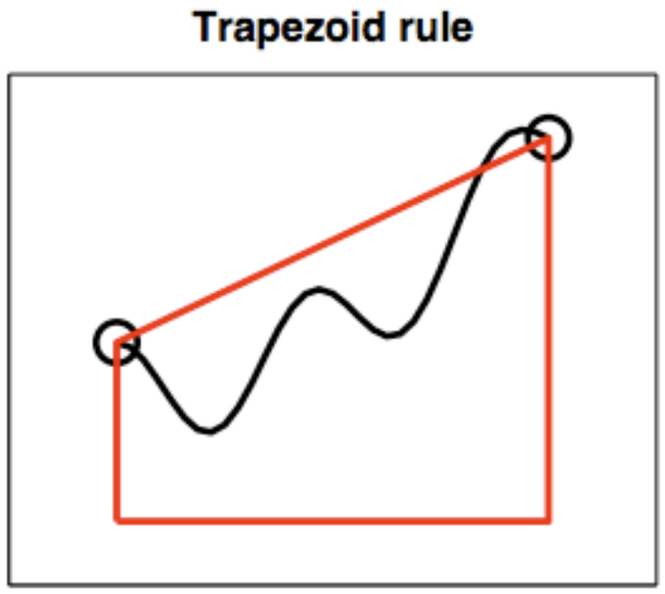
Let h=b−a be the length of the interval. The midpoint rule, M , approximates the integral by the area of a rectangle whose base has length h and whose height is the value of the integrand at the midpoint:



$$M = hf\left(\frac{a+b}{2}\right)$$

The Trapezoid Rule

The trapezoid rule, T , approximates the integral by the area of a trapezoid with base h and sides equal to the values of the integrand at the two endpoints:



$$T = h \frac{f(a) + f(b)}{2}$$

Accuracy

The accuracy of a quadrature rule can be predicted in part by examining its behavior on polynomials. The order of a quadrature rule is the degree of the lowest degree polynomial that the rule **does not** integrate exactly.

If a quadrature rule of order p is used to integrate a smooth function over a small interval of length h, then a Taylor series analysis shows that the error is proportional to h^p . The midpoint rule and the trapezoid rule are both exact for constant and linear functions of x, but neither of them is exact for a quadratic in x, so they both have order two. (The order of a rectangle rule with height f(a) or f(b) instead of the midpoint is only one.)

Example

The accuracy of the two rules can be compared by examining their behavior on the simple integral

$$\int_0^1 x^2 dx = \frac{1}{3}.$$

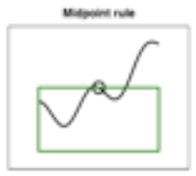
Example

The accuracy of the two rules can be compared by examining their behavior on the simple integral

$$\int_0^1 x^2 dx = \frac{1}{3}.$$

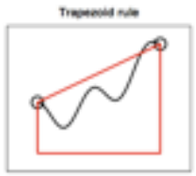
The midpoint rule gives

$$M = 1\left(\frac{1}{2}\right)^2 = \frac{1}{4}.$$



The trapezoid rule gives

$$T = 1\left(\frac{0+1^2}{2}\right) = \frac{1}{2}.$$



So the error in M is $1/12$, while the error in T is $-1/6$. The errors have opposite signs and, perhaps surprisingly, the midpoint rule is twice as accurate as the trapezoid rule.

Simpson's Rule

This turns out to be true more generally. For integrating smooth functions over short intervals, M is roughly twice as accurate as T and the errors have opposite signs. Knowing these error estimates allows us to combine the two and get a rule that is usually more accurate than either one separately. If the error in T were exactly -2 times the error in M, then solving

$$S - T = -2(S - M)$$

for S would give us the exact value of the integral. In any case, the solution

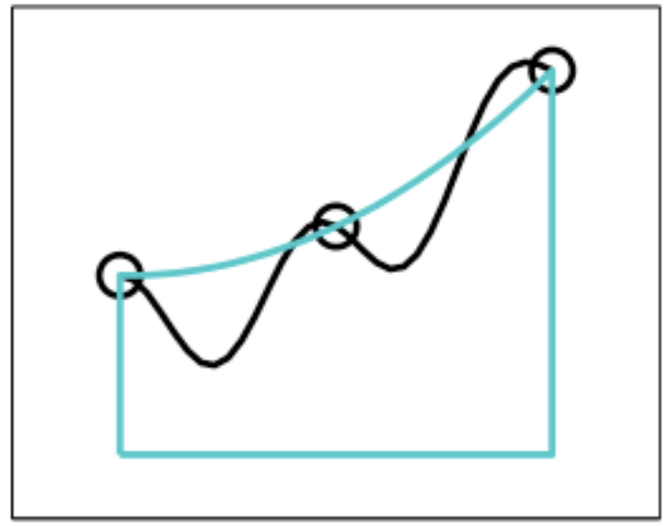
$$S = \frac{2}{3}M + \frac{1}{3}T$$

is usually a more accurate approximation than either M or T alone. This rule is known as *Simpson's rule*. It can also be derived by integrating the quadratic function that interpolates the integrand at the two endpoints, a and b, and the midpoint, $c = (a + b)/2$:

$$S = \frac{h}{6}(f(a) + 4f(c) + f(b)).$$

It turns out that S also integrates cubics exactly, but not quartics, so its order is four.

Simpson's rule



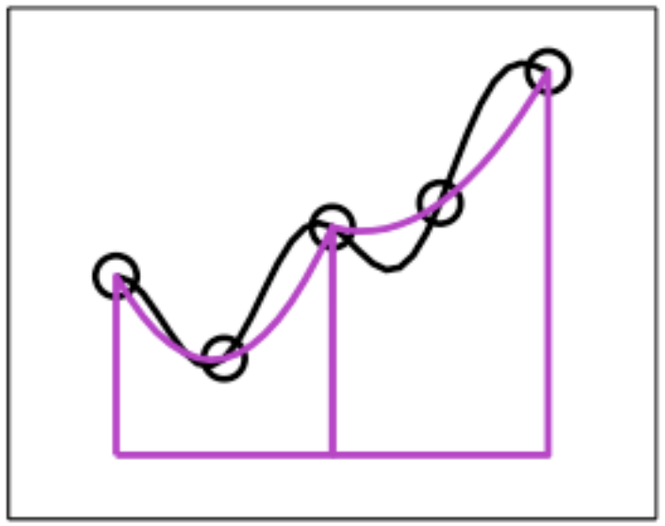
Composite Rule

We can carry this process one step further using the two halves of the interval, $[a, c]$ and $[c, b]$. Let d and e be the midpoints of these two subintervals: $d = (a+c)/2$ and $e = (c+b)/2$. Apply Simpson's rule to each subinterval to obtain a quadrature rule over $[a, b]$:

$$S_2 = \frac{h}{12}(f(a) + 4f(d) + 2f(c) + 4f(e) + f(b)).$$

This is an example of a *composite* quadrature rule.

Composite Simpson's rule



Now run the demo quadgui.m

Look at the pdf quad.pdf for more info

Reading Assignment Numerical Recipes Chapter 4

Compare Symbolic and Numeric Integration

Compare Symbolic and Numeric Integration

QuadNumericAndSymbolic.m

Monte Carlo Applications

Monte Carlo Methods

Monte Carlo methods are a widely used class of computational algorithms for simulating the behavior of various physical and mathematical systems, and for other computations.

They are distinguished from other simulation methods (such as molecular dynamics) by being **stochastic**, that is nondeterministic in some manner – usually by using random numbers (or, more often, pseudo-random numbers) – as opposed to deterministic algorithms.

Because of the repetition of algorithms and the large number of calculations involved, Monte Carlo is a method suited to calculation using a computer, utilizing many techniques of computer simulation.

Monte Carlo Algorithm

A Monte Carlo algorithm is often a numerical Monte Carlo method used to find solutions to mathematical problems (which may have many variables) that cannot easily be solved, for example, by integral calculus, or other numerical methods.

For many types of problems, its efficiency relative to other numerical methods **increases as the dimension of the problem increases**. Or it may be a method for solving other mathematical problems that relies on (pseudo-)random numbers.

Applications

Monte Carlo simulation methods are especially useful in studying systems with a large number of coupled degrees of freedom, such as liquids, disordered materials, strongly coupled solids, and cellular structures (e.g. cellular Potts model).

More broadly, Monte Carlo methods are useful for modeling phenomena with significant uncertainty in inputs, such as the calculation of risk in business.

A classic use is for the **evaluation of definite integrals**, particularly **multidimensional** integrals with **complicated boundary conditions**.

Applications

Monte Carlo methods are very important in computational physics, physical chemistry, and related applied fields, and have diverse applications from complicated quantum chromodynamics calculations (theory of the strong interaction or color force) to designing heat shields and aerodynamic forms.

Monte Carlo methods have also proven efficient in solving coupled integral differential equations of radiation fields and energy transport, and thus these methods have been used in global illumination computations which produce photorealistic images of virtual 3D models, with applications in video games, architecture, design, computer generated films, special effects in cinema, business, economics and other fields.

Monte Carlo integration

In mathematics, Monte Carlo integration is numerical quadrature using pseudo-random numbers. That is, Monte Carlo integration methods are algorithms for the approximate evaluation of definite integrals, usually multidimensional ones. The usual algorithms evaluate the integrand at a **regular grid**. Monte Carlo methods, however, **randomly choose the points** at which the integrand is evaluated.

The traditional Monte Carlo algorithm distributes the evaluation points uniformly over the integration region. Adaptive algorithms such as VEGAS and MISER use importance sampling and stratified sampling techniques to get a better result.

Monte Carlo integration

The traditional algorithm

The algorithm computes an estimate of a multidimensional definite integral of the form,

$$I = \int_{x_l}^{x_u} \int_{y_l}^{y_u} f(x, y, \ldots)\, dx\, dy \ldots = \int_V f(x, y, \ldots)\, dx\, dy \ldots$$

over the hypercube with volume V defined by $\{ (x, y, \ldots) \mid x_l \le x \le x_u, y_l \le y \le y_u, \ldots \}$.

The plain Monte Carlo algorithm samples points uniformly from the integration region to estimate the integral and its error. Suppose that the sample has size N and denote the points in the sample by $x_1, \ldots, x_N$. Then the estimate for the integral is given by

$$E(f; N) = V \cdot \langle f \rangle = V \frac{1}{N} \sum_{i=1}^{N} f(x_i),$$

where $\langle f \rangle$ denotes the sample mean of the integrand.

The variance in this result can be estimated using

$$\sigma^2(E; N) = \frac{V}{N} \sum_{i=1}^{N} (f(x_i) - \langle f \rangle)^2.$$

For large N this variance decreases asymptotically as $\mathrm{var}(f) / N$, where $\mathrm{var}(f)$ is the true variance of the function over the integration region. The error estimate itself should decrease as $\sigma(f) / \sqrt{N}$. The familiar law of errors decreasing as $1 / \sqrt{N}$ applies: to reduce the error by a factor of 10 requires a 100-fold increase in the number of sample points.

The above expression provides a statistical estimate of the error on the result. This error estimate is not a strict error bound — random sampling of the region may not uncover all the important features of the function, resulting in an underestimate of the error.

Monte Carlo integration

The traditional algorithm

The algorithm computes an estimate of a multidimensional definite integral of the form,

$$I = \int_{x_l}^{x_u} \int_{y_l}^{y_u} f(x, y, \ldots)\, dx\, dy \ldots = \int_V f(x, y, \ldots)\, dx\, dy \ldots$$

over the hypercube with volume V defined by $\{ (x, y, \ldots) \mid x_l \le x \le x_u, y_l \le y \le y_u, \ldots \}$.

The plain Monte Carlo algorithm samples points uniformly from the integration region to estimate the integral and its error. Suppose that the sample has size N and denote the points in the sample by $x_1, \ldots, x_N$. Then the estimate for the integral is given by

$$E(f; N) = V \cdot \langle f \rangle = V \frac{1}{N} \sum_{i=1}^{N} f(x_i),$$

where $\langle f \rangle$ denotes the sample mean of the integrand.

The variance in this result can be estimated using

$$\sigma^2(E; N) = \frac{V}{N} \sum_{i=1}^{N} (f(x_i) - \langle f \rangle)^2.$$

For large N this variance decreases asymptotically as $\mathrm{var}(f) / N$, where $\mathrm{var}(f)$ is the true variance of the function over the integration region. The error estimate itself should decrease as $\sigma(f) / \sqrt{N}$. The familiar law of errors decreasing as $1 / \sqrt{N}$ applies: to reduce the error by a factor of 10 requires a 100-fold increase in the number of sample points.

The above expression provides a statistical estimate of the error on the result. This error estimate is not a strict error bound — random sampling of the region may not uncover all the important features of the function, resulting in an underestimate of the error.

Monte Carlo integration

The traditional algorithm

The algorithm computes an estimate of a multidimensional definite integral of the form,

$$I = \int_{x_l}^{x_u} \int_{y_l}^{y_u} f(x, y, \ldots)\, dx\, dy \ldots = \int_V f(x, y, \ldots)\, dx\, dy \ldots$$

over the hypercube with volume V defined by $\{ (x, y, \ldots) \mid x_l \le x \le x_u, y_l \le y \le y_u, \ldots \}$.

The plain Monte Carlo algorithm samples points uniformly from the integration region to estimate the integral and its error. Suppose that the sample has size N and denote the points in the sample by $x_1, \ldots, x_N$. Then the estimate for the integral is given by

$$E(f; N) = V \cdot \langle f \rangle = V \frac{1}{N} \sum_{i=1}^{N} f(x_i),$$

where $\langle f \rangle$ denotes the sample mean of the integrand.

The variance in this result can be estimated using

$$\sigma^2(E; N) = \frac{V}{N} \sum_{i=1}^{N} (f(x_i) - \langle f \rangle)^2.$$

The error does not depend on the dimension, it is proportional to the variance

For large N this variance decreases asymptotically as $\mathrm{var}(f) / N$, where $\mathrm{var}(f)$ is the true variance of the function over the integration region. The error estimate itself should decrease as $\sigma(f) / \sqrt{N}$. The familiar law of errors decreasing as $1 / \sqrt{N}$ applies: to reduce the error by a factor of 10 requires a 100-fold increase in the number of sample points.

The above expression provides a statistical estimate of the error on the result. This error estimate is not a strict error bound — random sampling of the region may not uncover all the important features of the function, resulting in an underestimate of the error.

Monte Carlo integration

The traditional algorithm

The algorithm computes an estimate of a multidimensional definite integral of the form,

$$I = \int_{x_l}^{x_u} \int_{y_l}^{y_u} f(x, y, \ldots)\, dx\, dy \ldots = \int_V f(x, y, \ldots)\, dx\, dy \ldots$$

over the hypercube with volume V defined by $\{ (x, y, \ldots) \mid x_l \le x \le x_u, y_l \le y \le y_u, \ldots \}$.

The plain Monte Carlo algorithm samples points uniformly from the integration region to estimate the integral and its error. Suppose that the sample has size N and denote the points in the sample by $x_1, \ldots, x_N$. Then the estimate for the integral is given by

$$E(f; N) = V \cdot \langle f \rangle = V \frac{1}{N} \sum_{i=1}^{N} f(x_i),$$

where $\langle f \rangle$ denotes the sample mean of the integrand.

The variance in this result can be estimated using

$$\sigma^2(E; N) = \frac{V}{N} \sum_{i=1}^{N} (f(x_i) - \langle f \rangle)^2.$$

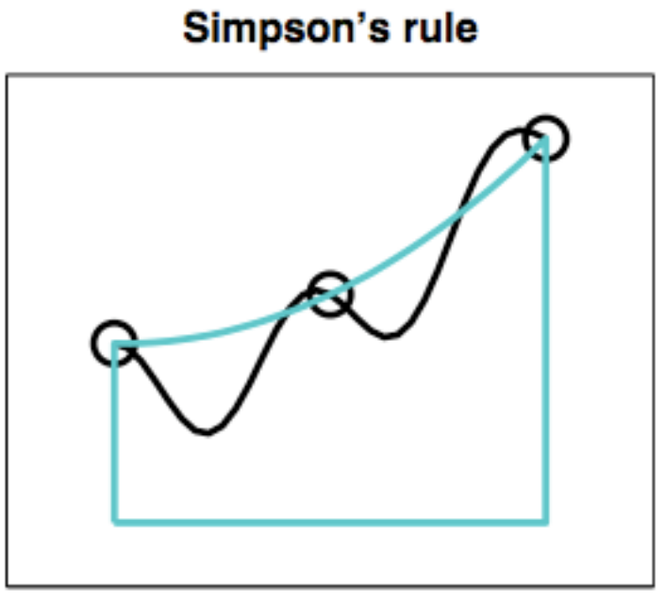
For large N this variance decreases asymptotically as var(f) / N, where var(f) is the true variance of the function over the integration region. The error estimate itself should decrease as $\sigma(f) / \sqrt{N}$. The familiar law of errors decreasing as $1 / \sqrt{N}$ applies: to reduce the error by a factor of 10 requires a 100-fold increase in the number of sample points.

The above expression provides a statistical estimate of the error on the result. This error estimate is not a strict error bound — random sampling of the region may not uncover all the important features of the function, resulting in an underestimate of the error.

The error does not depend on the dimension, it is proportional to the variance

A MC integration is more efficient than an an algorithm with order k error when dimension, d>2k

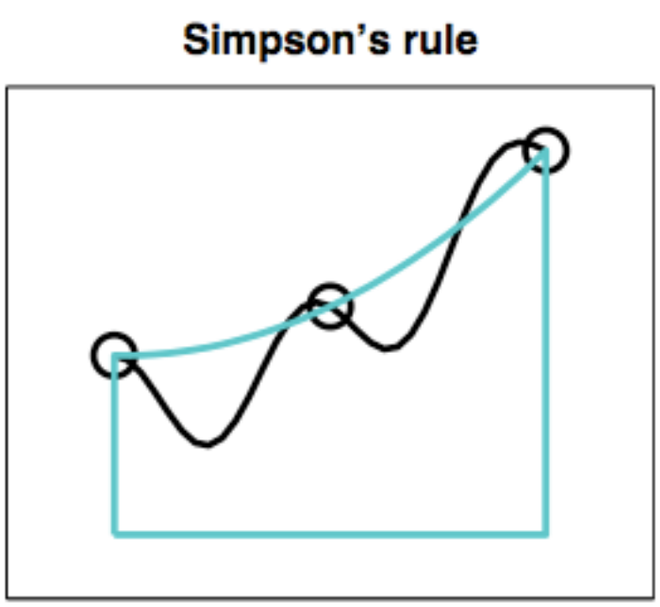
Monte Carlo integration



$$S = \frac{h}{6}(f(a) + 4f(c) + f(b)).$$

Monte Carlo integration

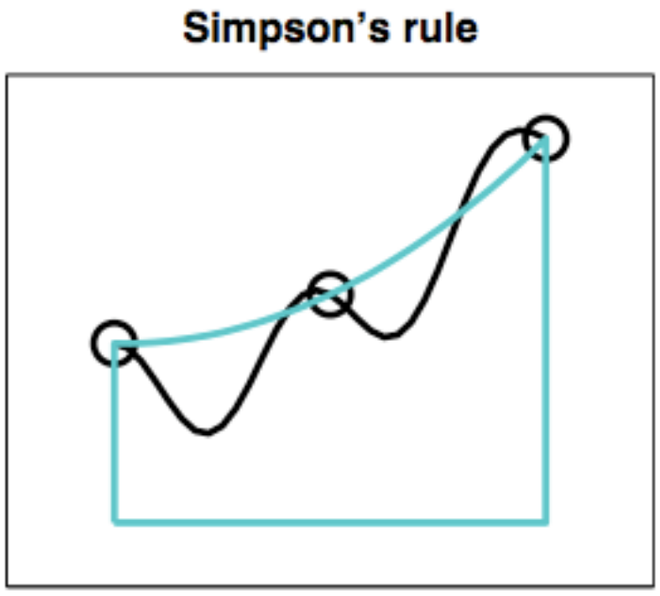
A MC integration is more efficient than an algorithm with order k error when dimension, **d>2k**



$$S = \frac{h}{6}(f(a) + 4f(c) + f(b)).$$

Monte Carlo integration

A MC integration is more efficient than an algorithm with order k error when dimension, **d>2k**



$$S = \frac{h}{6}(f(a) + 4f(c) + f(b)).$$

For Simpson's rule k=5. For a description of the errors associated with regular quadrature see chapter 4 of Numerical Recipes by Press et al.

Example: Radioactive Decay

To simulate how a **small** number of radioactive particles decay. In particular, to determine when radioactive decay looks exponential and when it looks stochastic (determined by chance).

Because the exponential decay law is only a large number approximation to the natural process, our simulation is closer to nature than the exponential decay law.

Example: Radioactive Decay

Theory: Spontaneous Decay

Spontaneous decay is a natural process in which a particle, with no external stimulation, and at one instant in time, decays into other particles.

Because the exact moment when any one particle decays is random, it does not matter how long the particle has been around or what is happening to the other particles.

Example: Radioactive Decay

In other words, the probability P of any one particle decaying per unit time is constant, and when one particle decays it is gone forever. So as the number of particles decrease with time, so will the number of decays.

The number of decay events -dN over an interval dT is proportional to the number of atoms N

$$-\frac{dN}{dt} \propto N$$

The probability of decay -dN/N is proportional to dT

$$-\frac{dN}{N} = \lambda \cdot dt$$

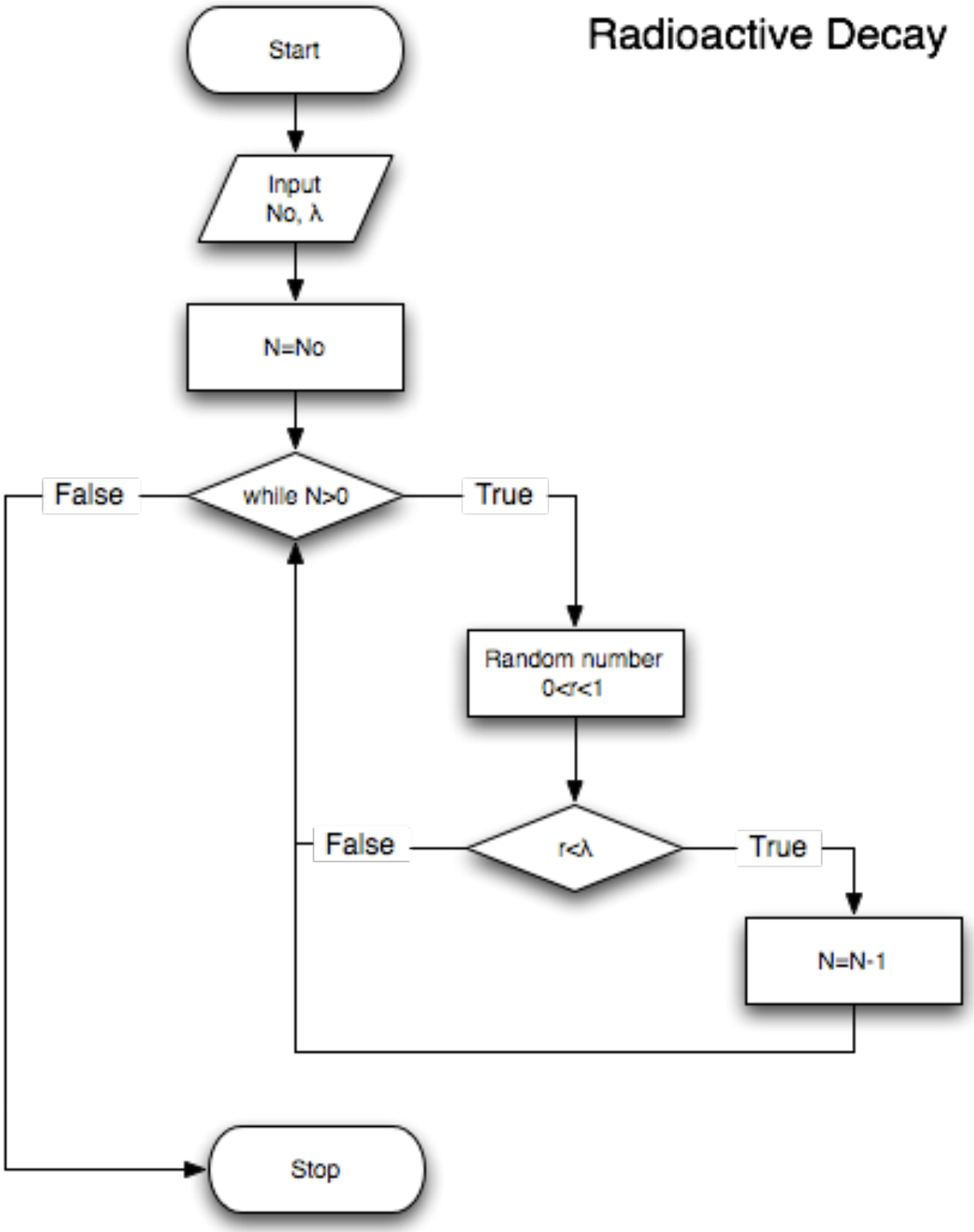
Each radionuclide has its own decay constant, λ, describing its decay. λ has units of 1/time.

$$N(t)=N_0 e^{-\lambda t}=N_0 e^{-\frac{t}{\tau}}$$

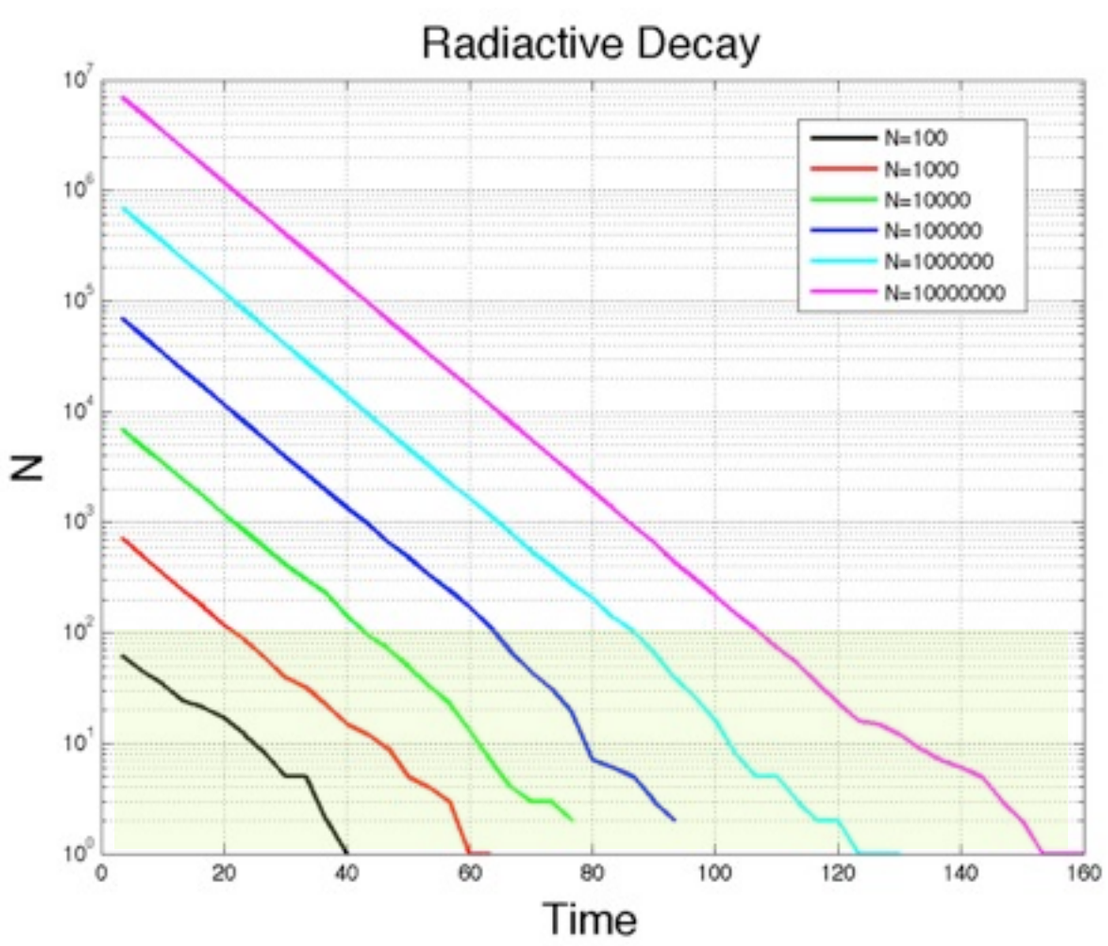
$$\tau = \frac{1}{\lambda}, \text{ and } \lambda = \frac{1}{\tau}$$

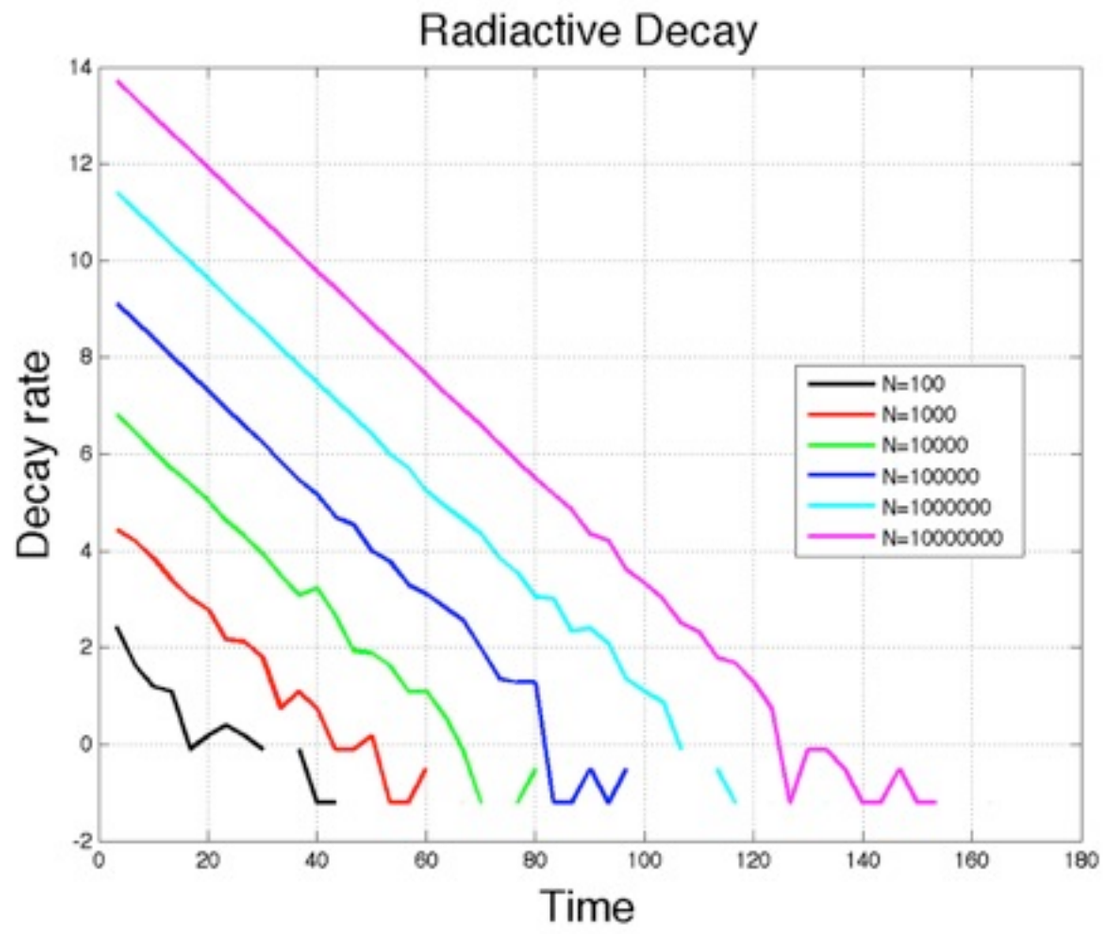
$$t_{\frac{1}{2}} = \frac{\ln 2}{\lambda} = \tau \ln 2$$

Homework 2



- Implement and visualize a Monte Carlo Simulation of radioactive decay. Start with 10^n radioactive atoms where n is 2,3,4,5,6, and 7.
- First of all start with λ=0.3. Time for one generation is λ^{-1}.
- Determine when the decay starts to be stochastic.
- Plot the decay versus time.
- Plot the decay rate ($\lambda\Delta N$) versus time.
- Try a range of λ, and verify that the decay is still exponential and that λ determines the decay rate.


Radiactive Decay
N
Time
N=100
N=1000
N=10000
N=100000
N=1000000
N=10000000


Radiactive Decay
Decay rate
Time
N=100
N=1000
N=10000
N=100000
N=1000000
N=10000000

Reading Assignment Numerical Recipes Chapter 7