Tolerance Design

Module: 3/EE/M Tolerance Design

Lecturer: James Grimbleby
URL: http://www.elec.rdg.ac.uk/jbg.html
email: j.b.grimbleby@reading.ac.uk

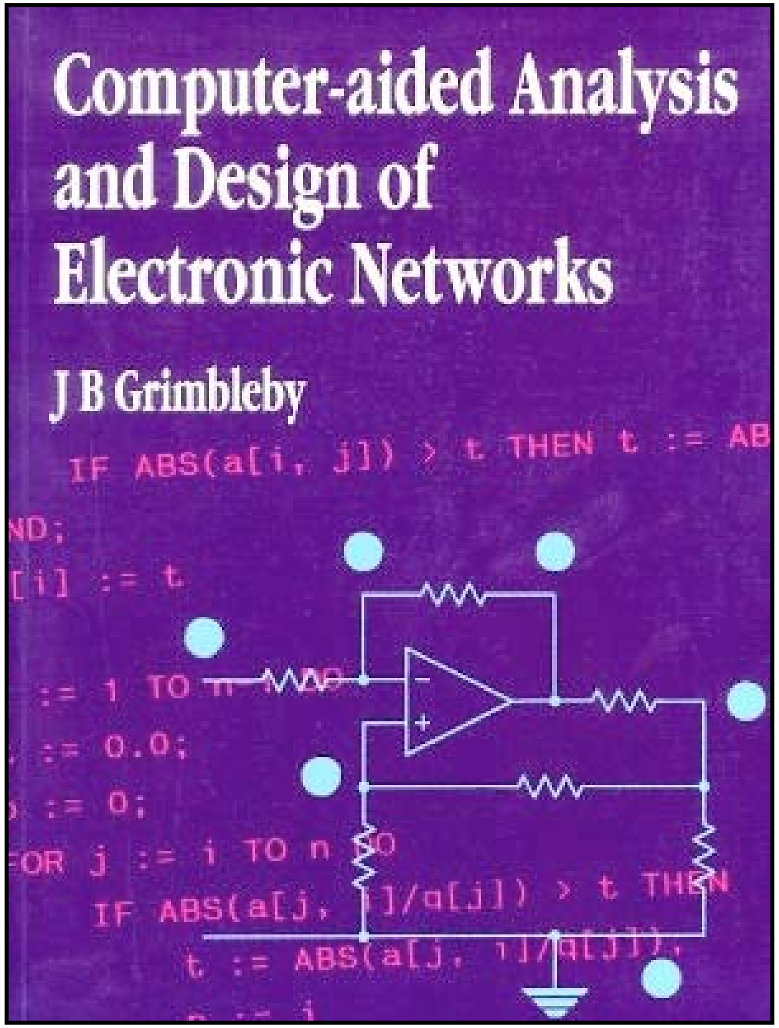
Number of Lectures: 5

Recommended text books:

Tolerance Design of Electronic Circuits, *R.Spence* and *R.SinghSoin*, Addison Wesley, ISBN 0-201-18242-4

Computer-Aided Analysis and Design of Electronic Networks, *J.B.Grimbleby*, Pitman, ISBN 0-273-03148-1

Tolerance Design



Computer-Aided Analysis and Design of Electronic Networks
J.B.Grimbleby
Pitman
ISBN 0-273-03148-1

Tolerance Design Syllabus

Introduction to component tolerancing

Worst-case tolerancing; vertex method

Statistical tolerancing methods; component value distribution functions

Method of moments

Monte-Carlo method; statistical significance of results

Pseudo-random number generators; non-uniform probability distribution functions

Design centering

Tolerance Design Syllabus

Circuits are normally designed for mass-production

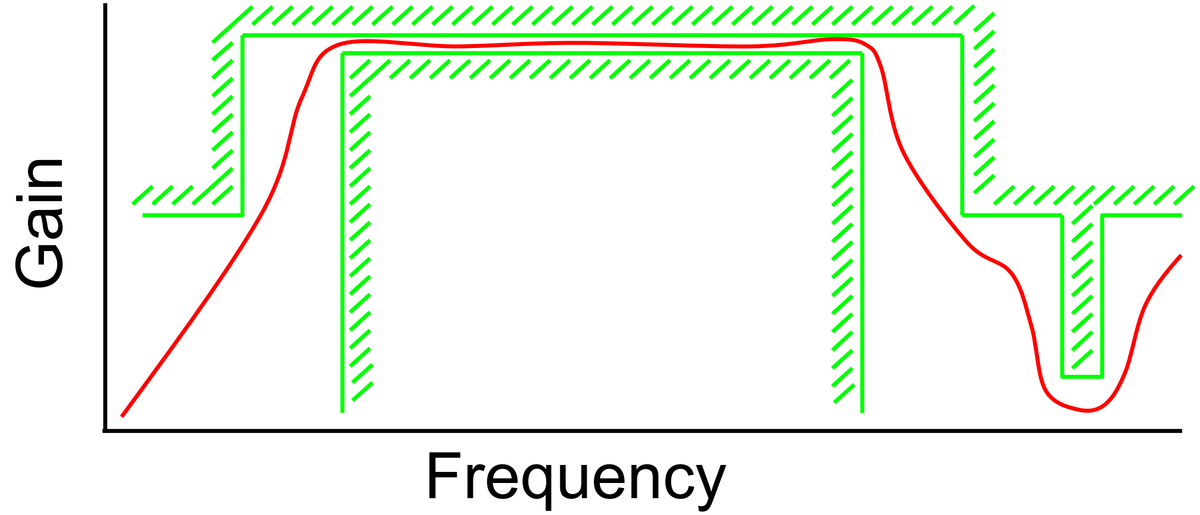
As a consequence of component tolerances the mass produced circuits will exhibit performance variations

These variations may be sufficient to violate the circuit specifications

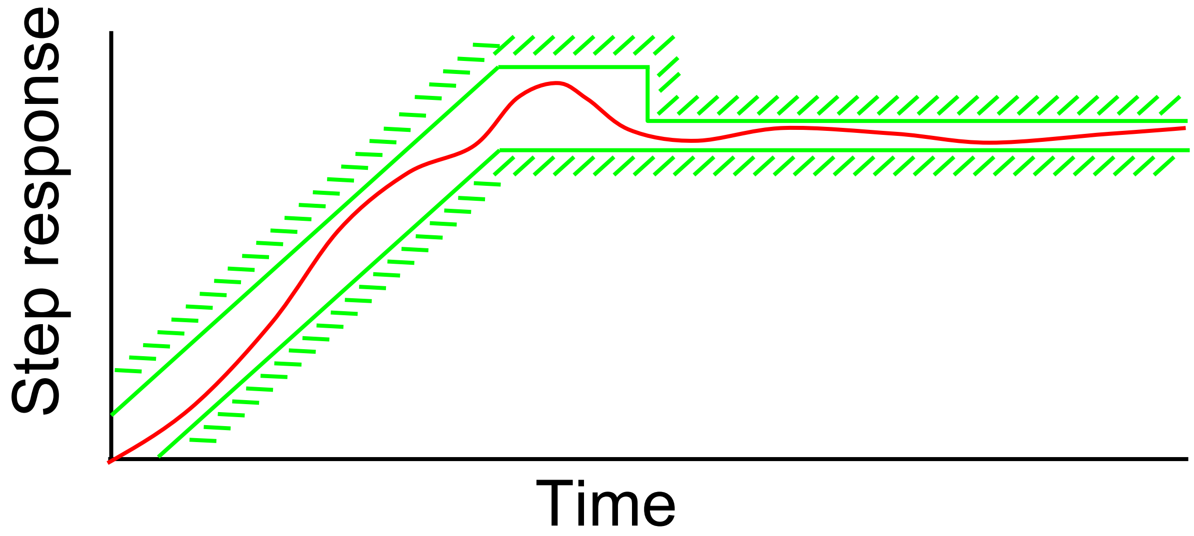
Tolerance design attempts to quantify the effects of component variations on circuit performance and if necessary to modify the design appropriately

Circuit Specifications

Power amplifier

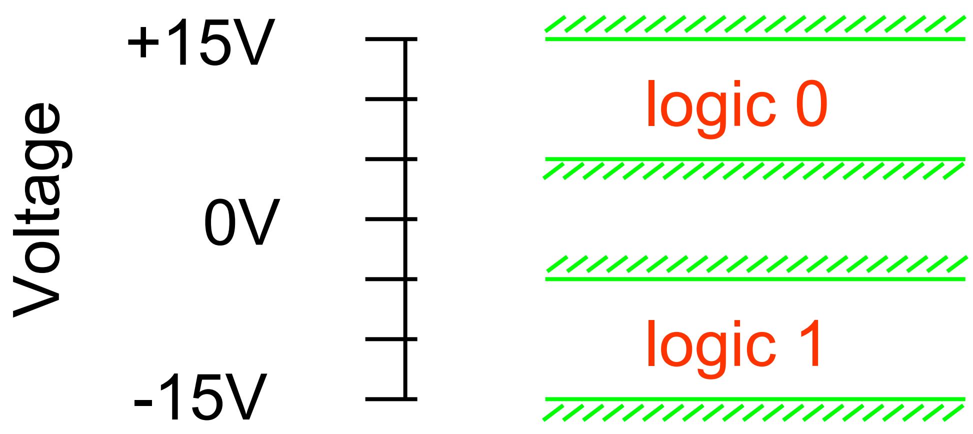


Time-domain filter

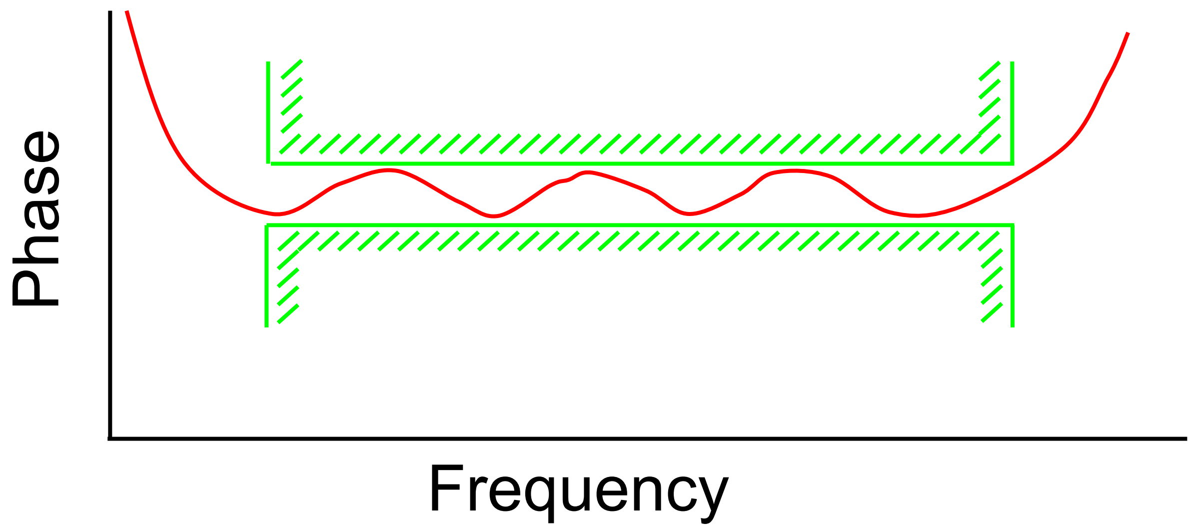


Circuit Specifications

RS232

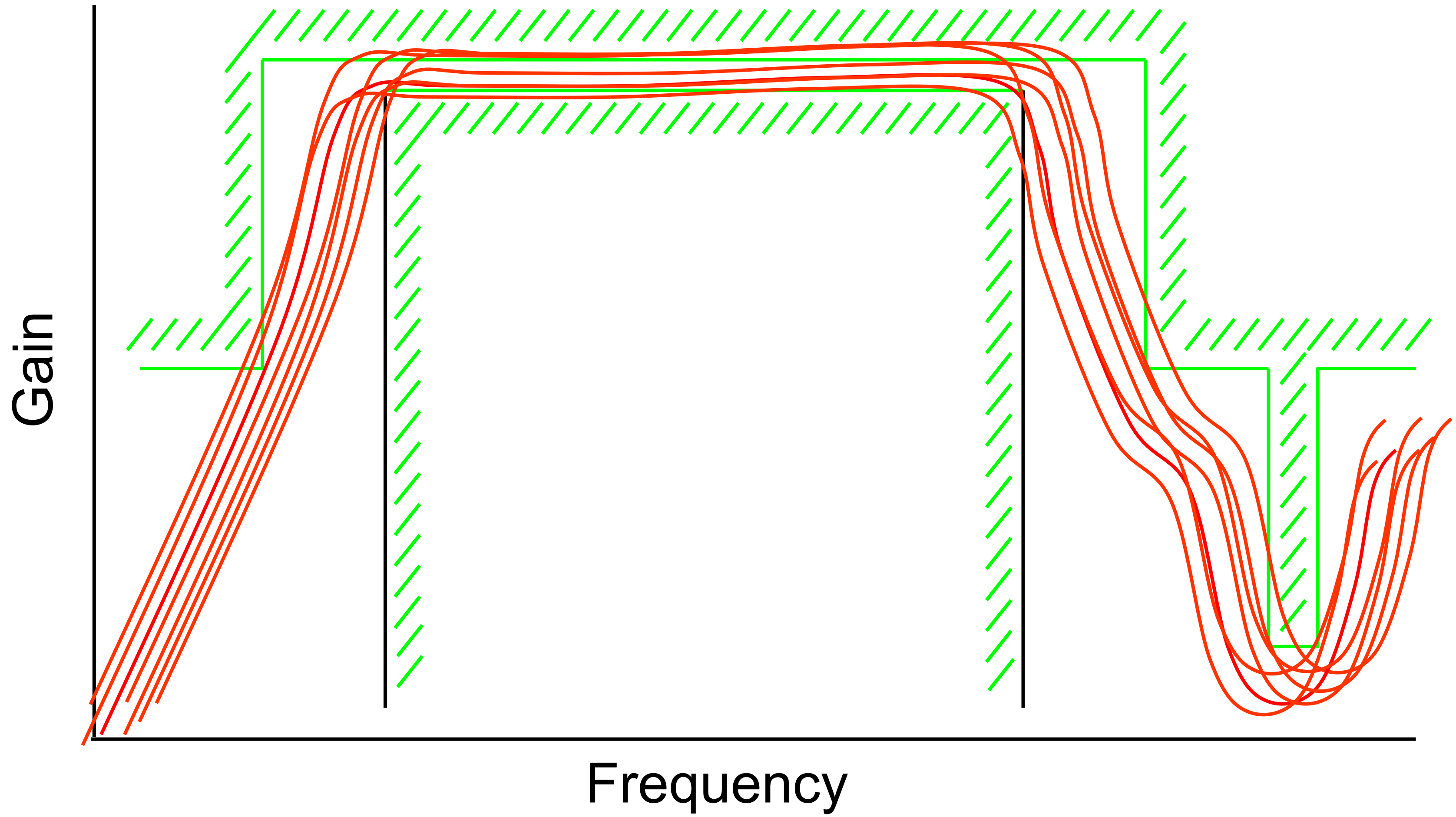


Phase
shifter



Effects of Component Tolerance

University of Reading



Tolerance Design

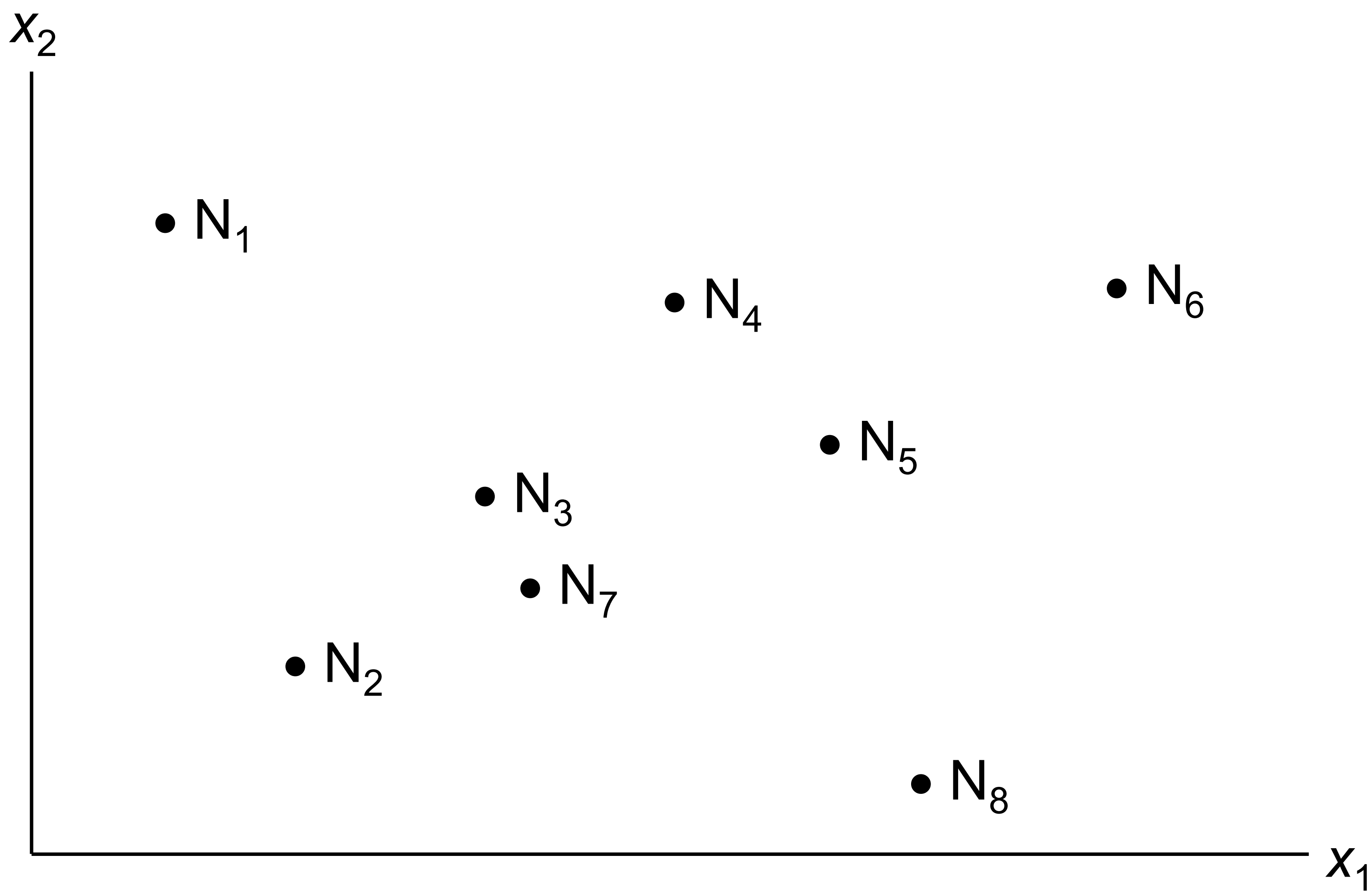
Tolerance design attempts to answer the following questions:

What will be the manufacturing yield with given component tolerances ?

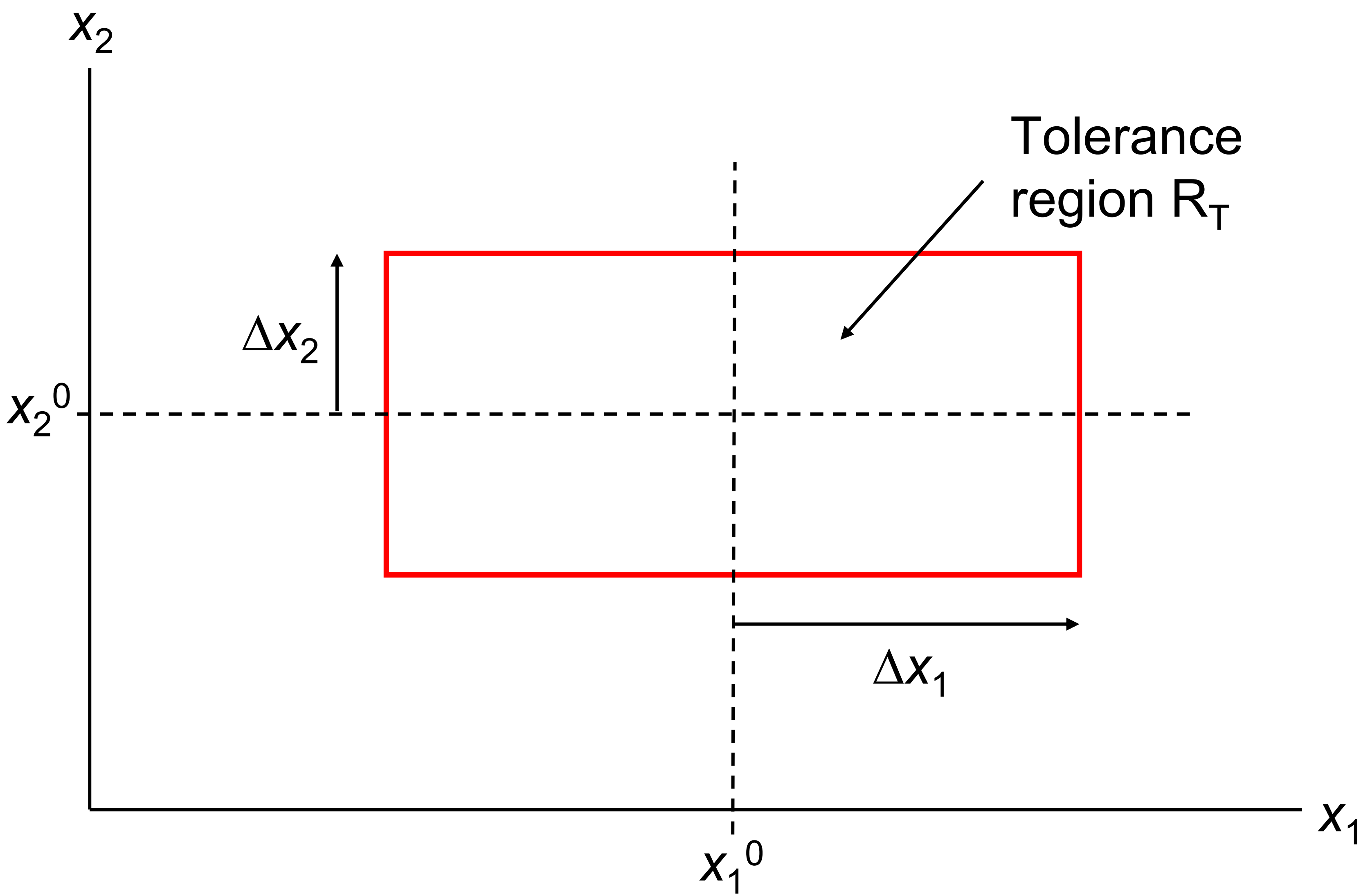
Is there some choice of component tolerances which, while giving a yield below 100%, nevertheless minimises manufacturing costs ?

Can initial design be modified to give a better yield with given component tolerances ?

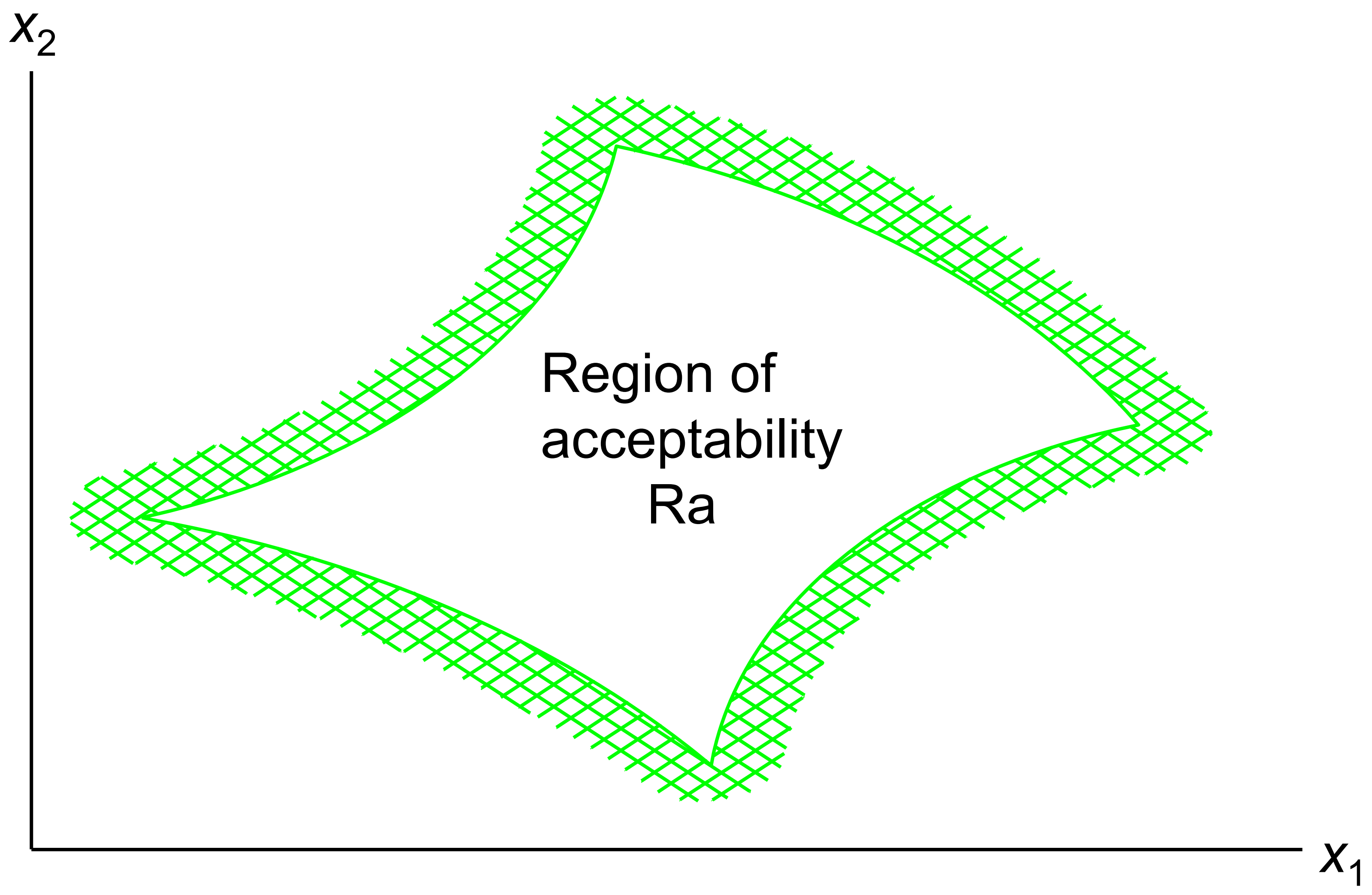
Parameter Space



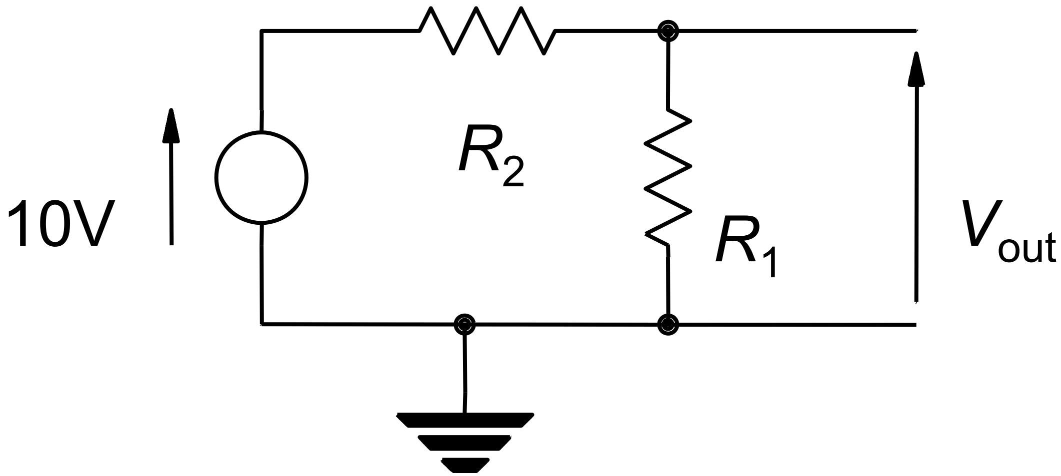
Tolerance Region



Region of Acceptability



Tolerance Design



Specification:

Output voltage: $4.5\text{V} < V_{out} < 5.5\text{V}$

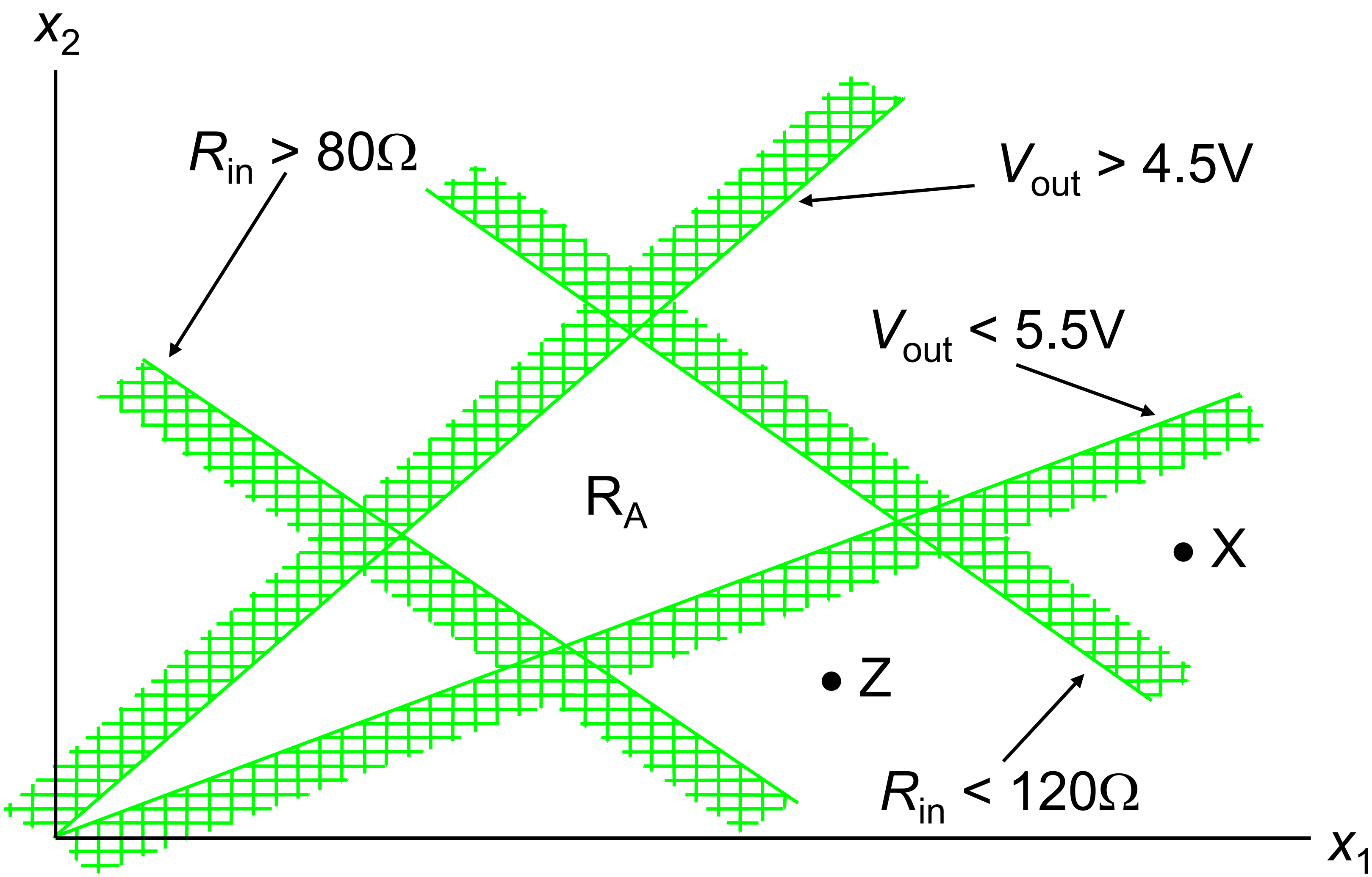
Input resistance: $80\Omega < R_{in} < 120\Omega$

Thus:

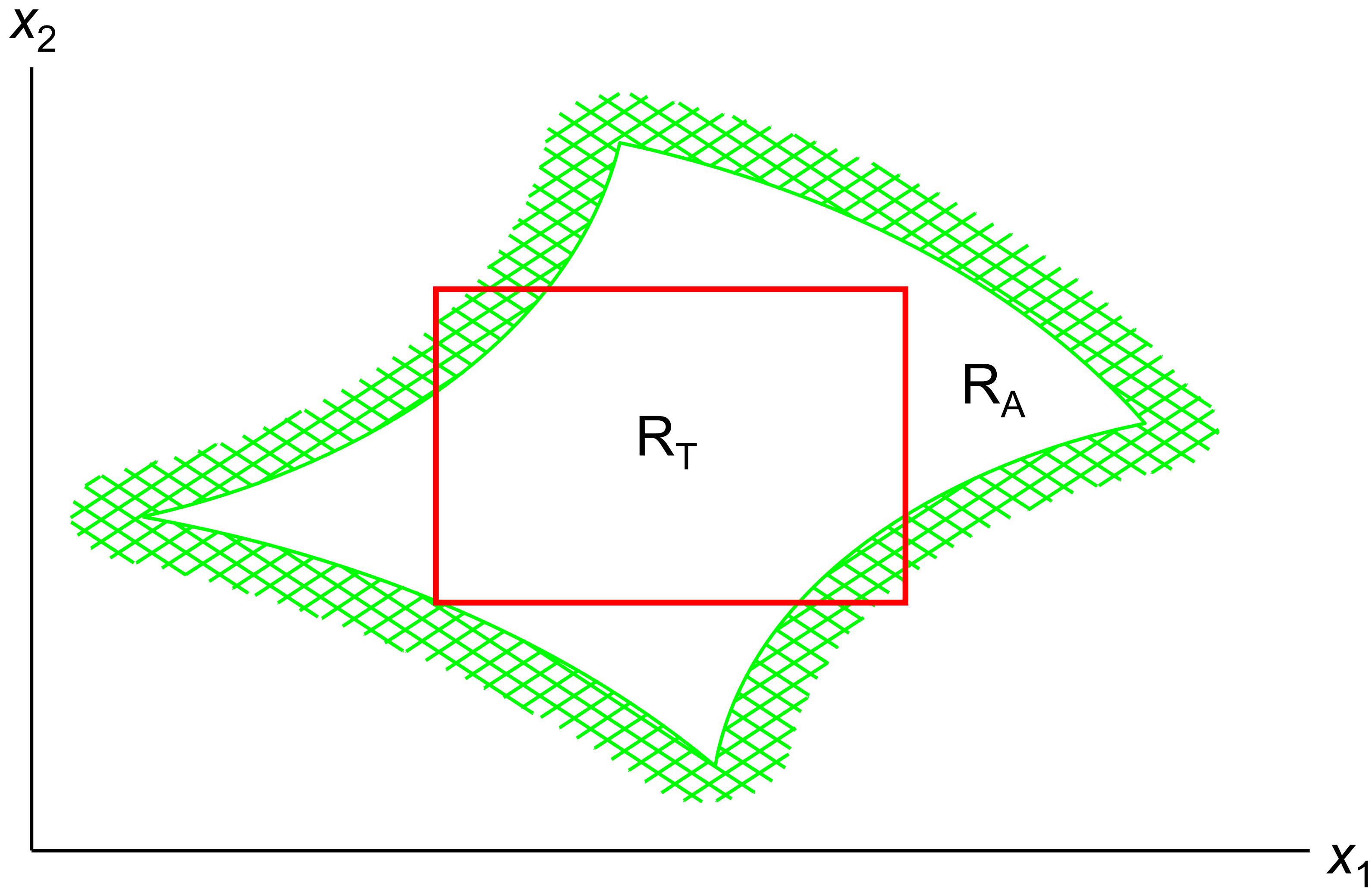
$$4.5\text{V} < 10\text{V}.\frac{R_1}{R_1 + R_2} < 5.5\text{V} \qquad 80\Omega < R_1 + R_2 < 120\Omega$$

Region of Acceptability

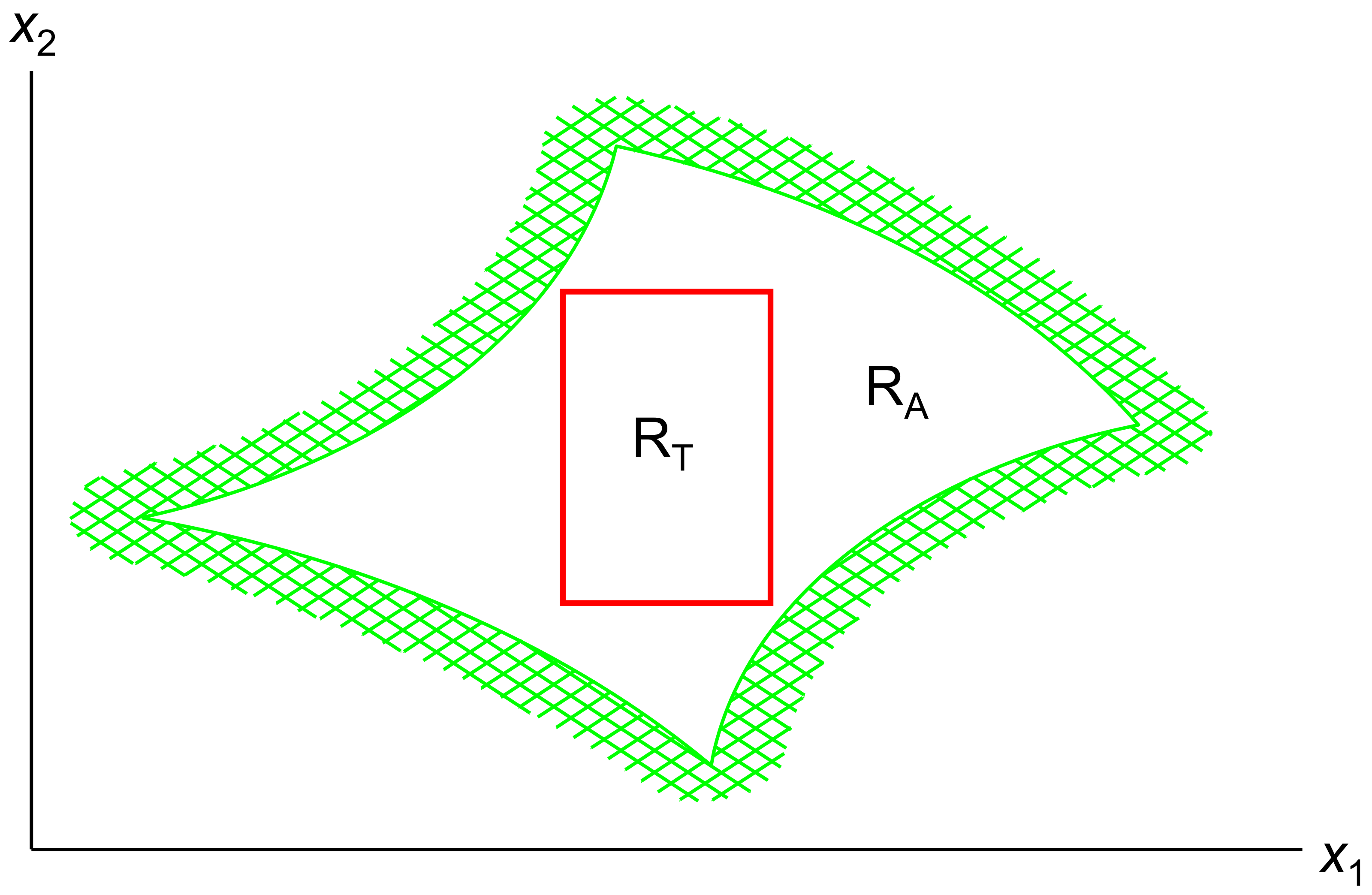
University of Reading



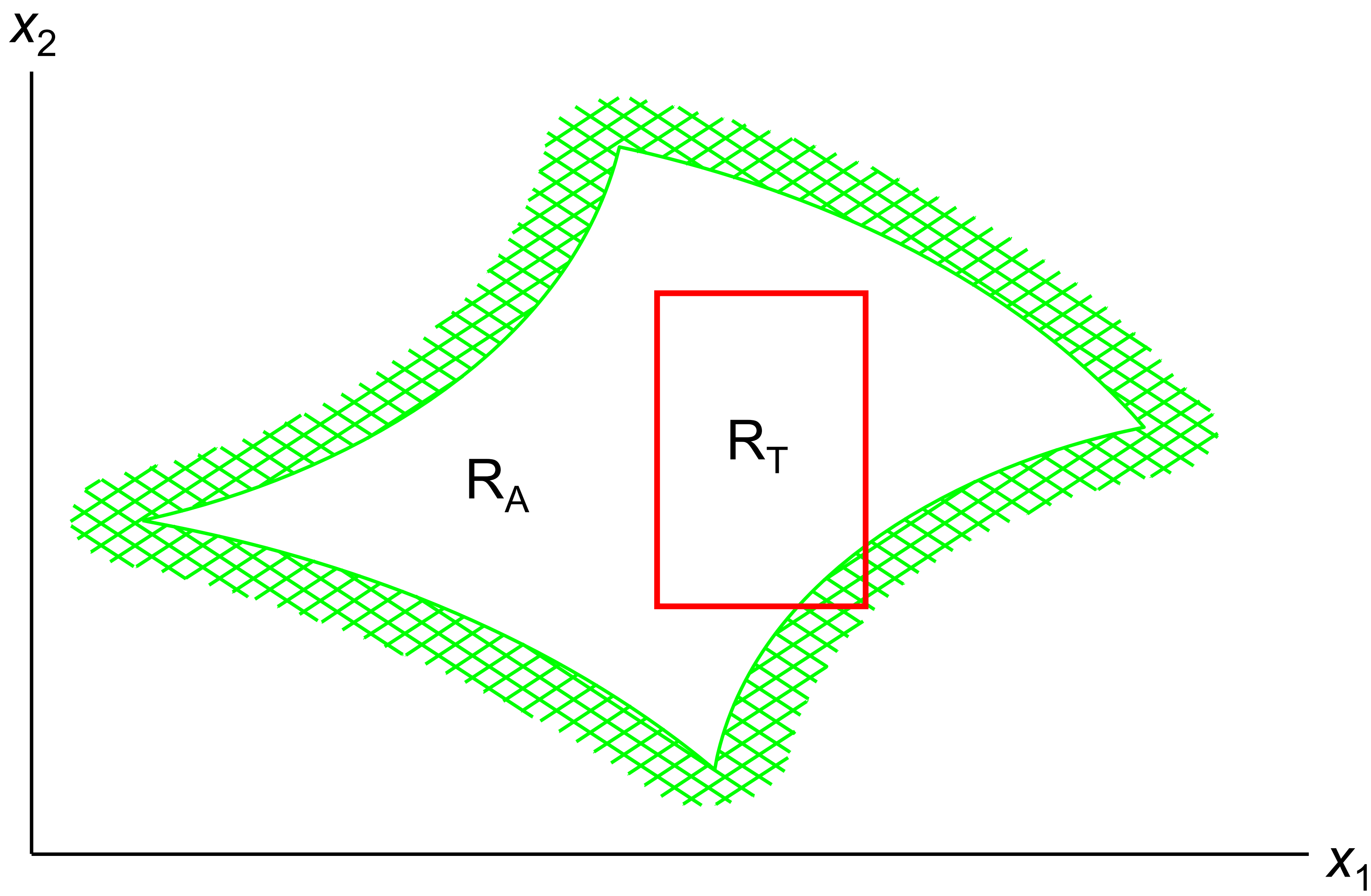
Manufacturing Yield < 100%



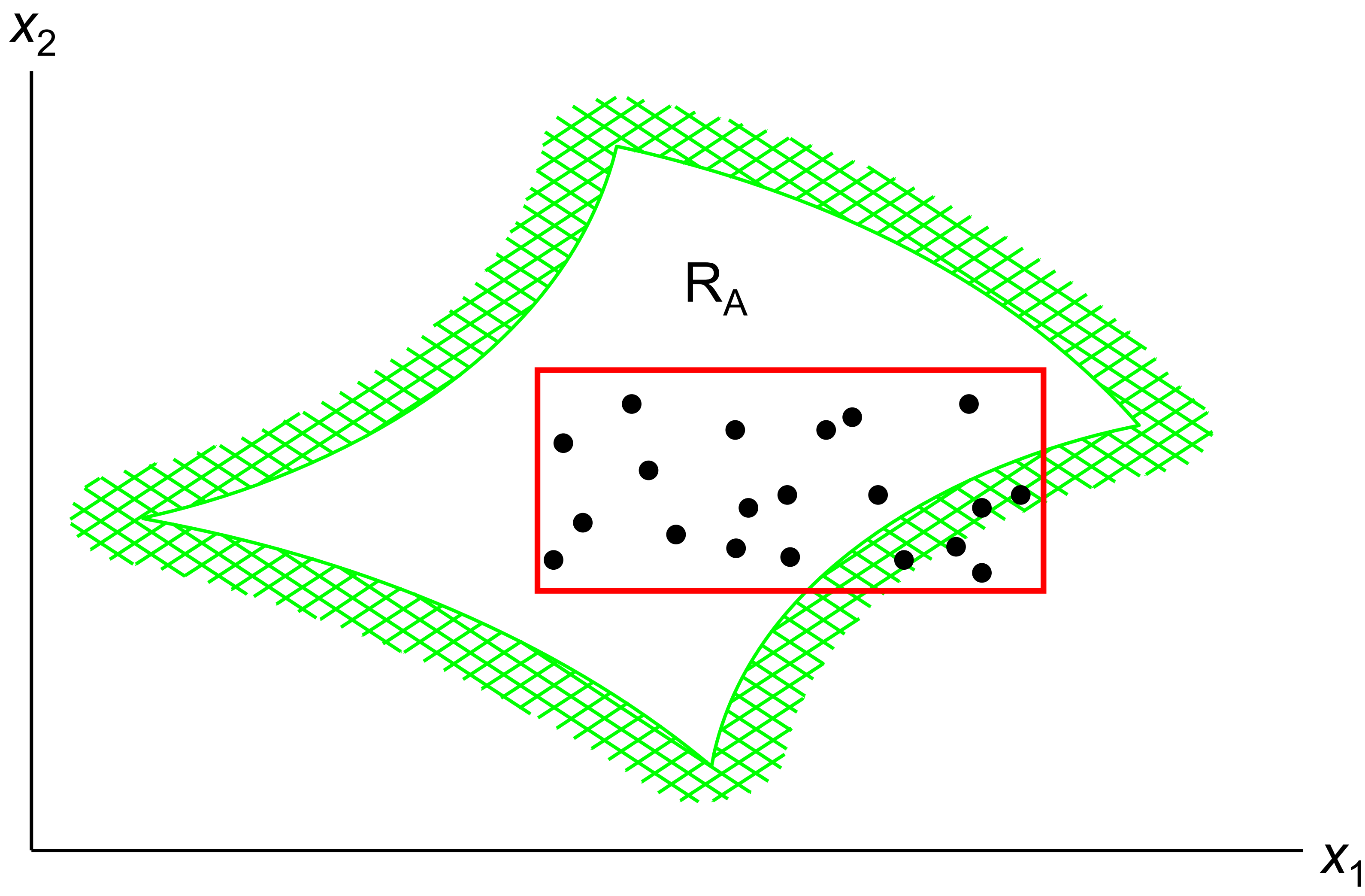
Manufacturing Yield = 100%



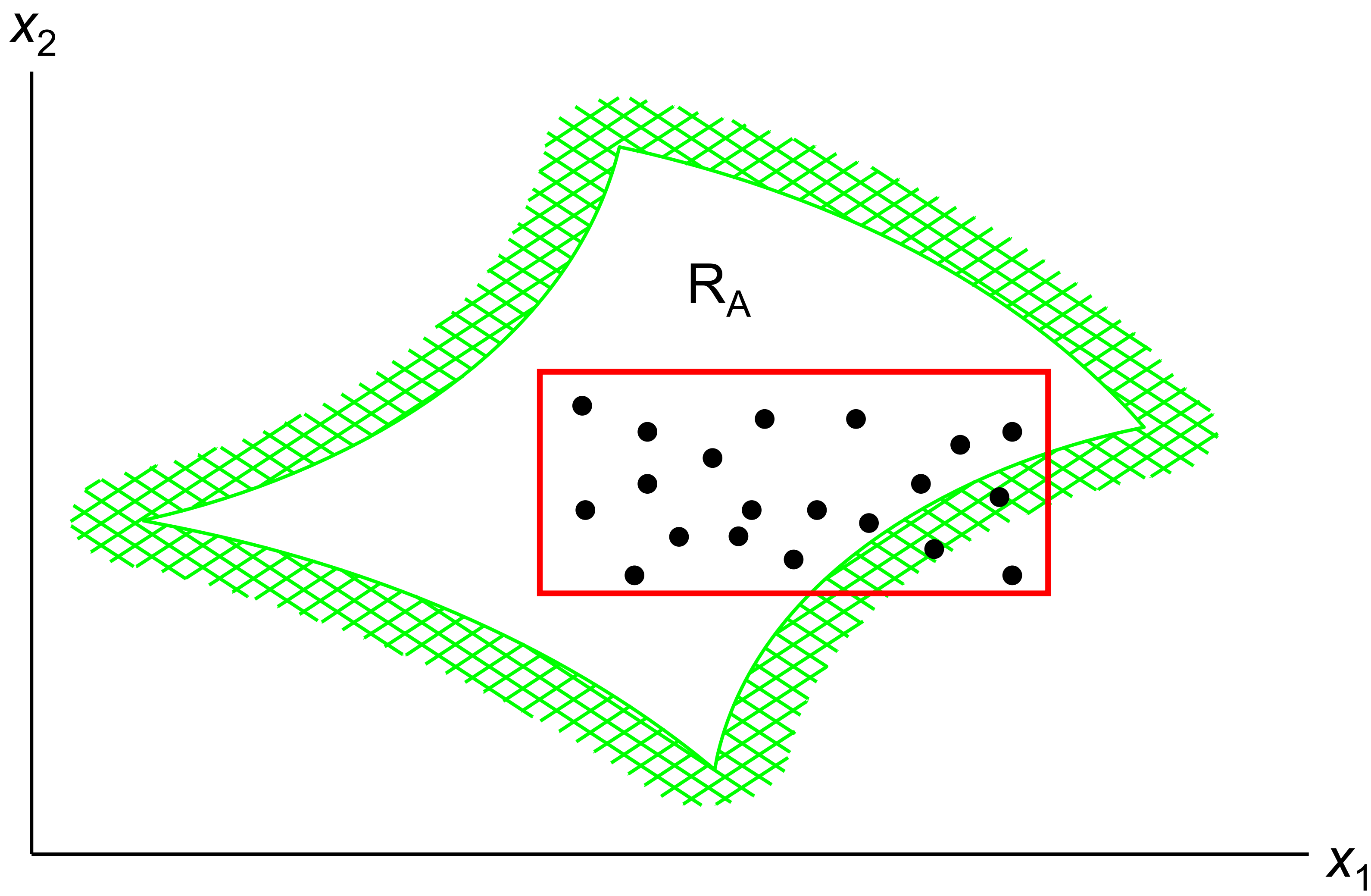
Manufacturing Yield < 100%



Estimating the Yield



Estimating the Yield



Tolerance Design

University of Reading

Tolerance Design

- Worst-Case Methods
  - Vertex Analysis
- Statistical Methods
  - Method of Moments
  - Monte-Carlo Analysis

Worst-Case Tolerance Analysis

Worst-case analysis aims to identify combinations of component values which lead to extreme (worst) performance

Only the tolerance ranges of components are of interest; the distribution of values within the range is of no account

Once the component values corresponding to extreme performance have been identified the circuit is analysed with these values to see whether it meets the performance specifications

Worst-Case Tolerance Analysis

The n component values will be represented by the vector $\mathbf{x}$ and their tolerances by $\Delta\mathbf{x}$:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \qquad \Delta\mathbf{x} = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \cdots \\ \Delta x_n \end{bmatrix}$$

In general a specification will cover several aspects of performance. One such performance function will be represented by f which depends on the n component values:

$$f \equiv f(\mathbf{x})$$

Worst-Case Tolerance Analysis

The specification item will normally have a lower limit *L* and an upper limit *U*:

$$L \le f(\mathbf{x}) \le U$$

A set of component values $\mathbf{x}^L$ within the tolerance range exists such that:

$$f(\mathbf{x}^L) \le f(\mathbf{x})$$

for all **x** within the tolerance region. Similarly a set of component values $\mathbf{x}^U$ within the tolerance range exists such that:

$$f(\mathbf{x}^U) \ge f(\mathbf{x})$$

for all **x** within the tolerance region

Worst-Case Tolerance Analysis

A manufacturing yield of 100% will be obtained provided that:

$$f(\mathbf{x}^L) \geq L$$

and:

$$f(\mathbf{x}^U) \leq U$$

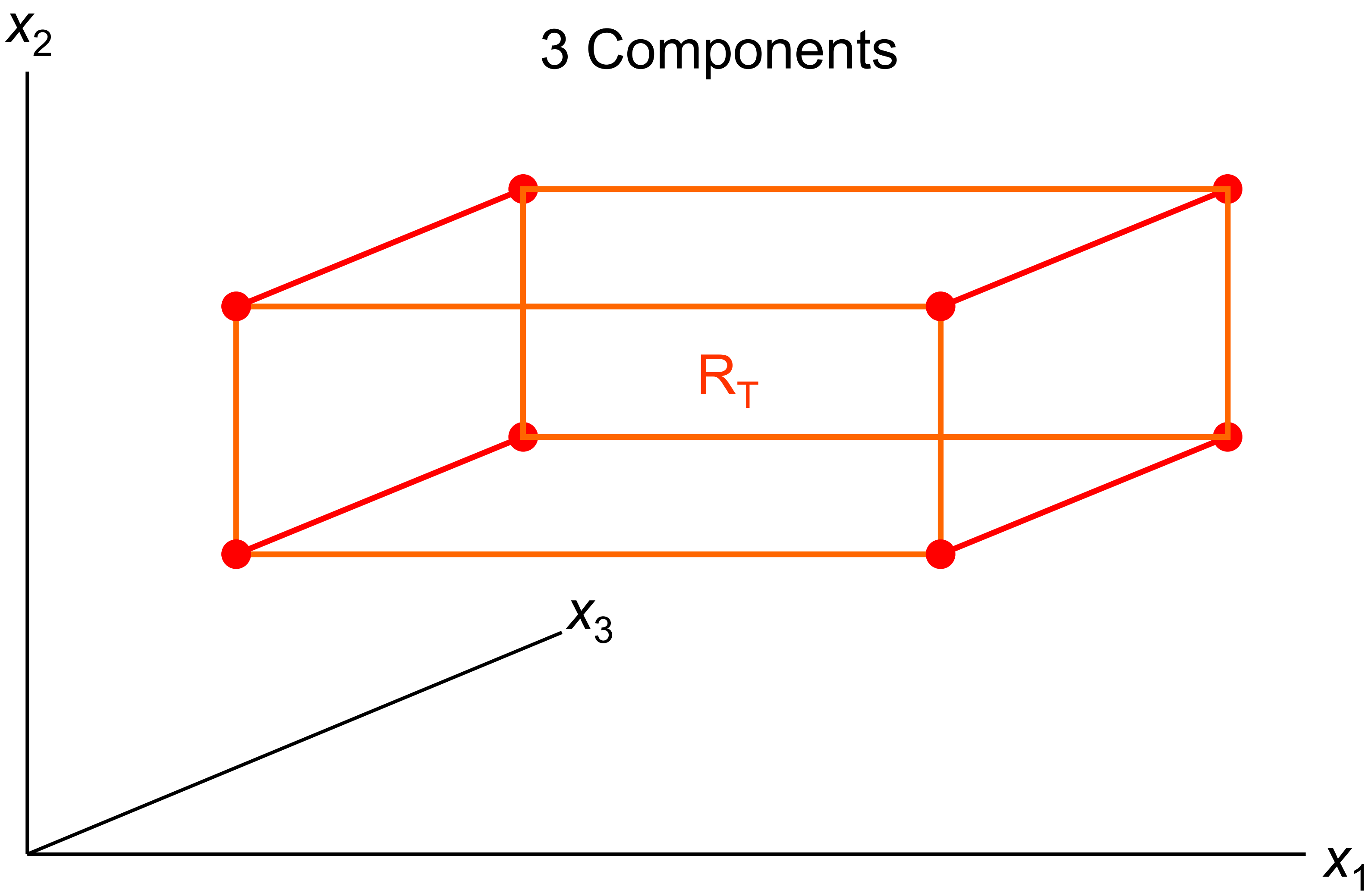
The problem lies in identifying the worst-case component value combinations $\mathbf{x}^L$ and $\mathbf{x}^U$

The only really satisfactory way of doing this is to use vertex analysis

Vertex analysis examines all combinations of the extreme component values

Vertex Analysis

3 Components



Vertex Analysis

The performance function *f* is evaluated at each of the vertices:

$$f(x_1 - \Delta x_1, x_2 - \Delta x_2, x_3 - \Delta x_3, .., x_n - \Delta x_n)$$

$$f(x_1 + \Delta x_1, x_2 - \Delta x_2, x_3 - \Delta x_3, .., x_n - \Delta x_n)$$

$$f(x_1 - \Delta x_1, x_2 + \Delta x_2, x_3 - \Delta x_3, .., x_n - \Delta x_n)$$

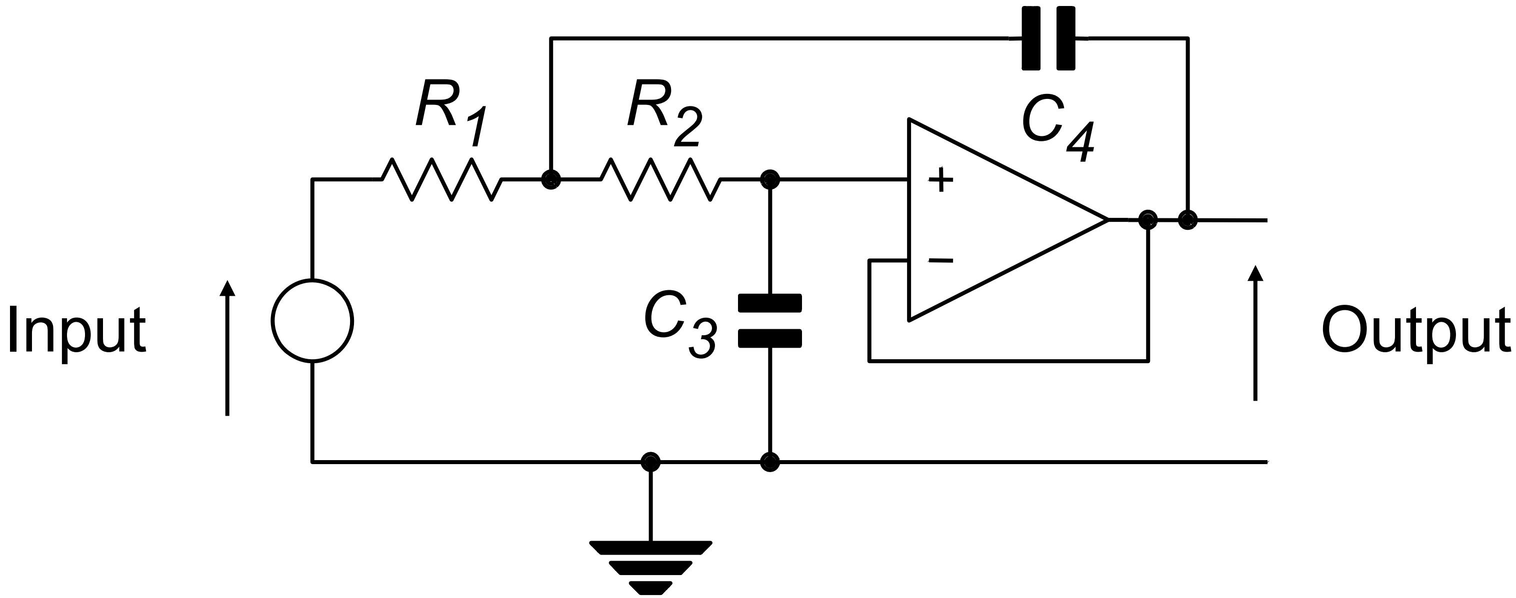
$$f(x_1 + \Delta x_1, x_2 + \Delta x_2, x_3 - \Delta x_3, .., x_n - \Delta x_n)$$

....

$$f(x_1 + \Delta x_1, x_2 + \Delta x_2, x_3 + \Delta x_3, .., x_n + \Delta x_n)$$

and a record is kept of the highest and lowest values of *f*

Tolerance Design



$$H(\mathrm{j}\omega) = \frac{1}{1 + C_3(R_1 + R_2)\mathrm{j}\omega + R_1R_2C_3C_4(\mathrm{j}\omega)^2}$$

R_1 = 100 kΩ C_3 = 1 nF

R_2 = 100 kΩ C_4 = 100 nF

Tolerance Design

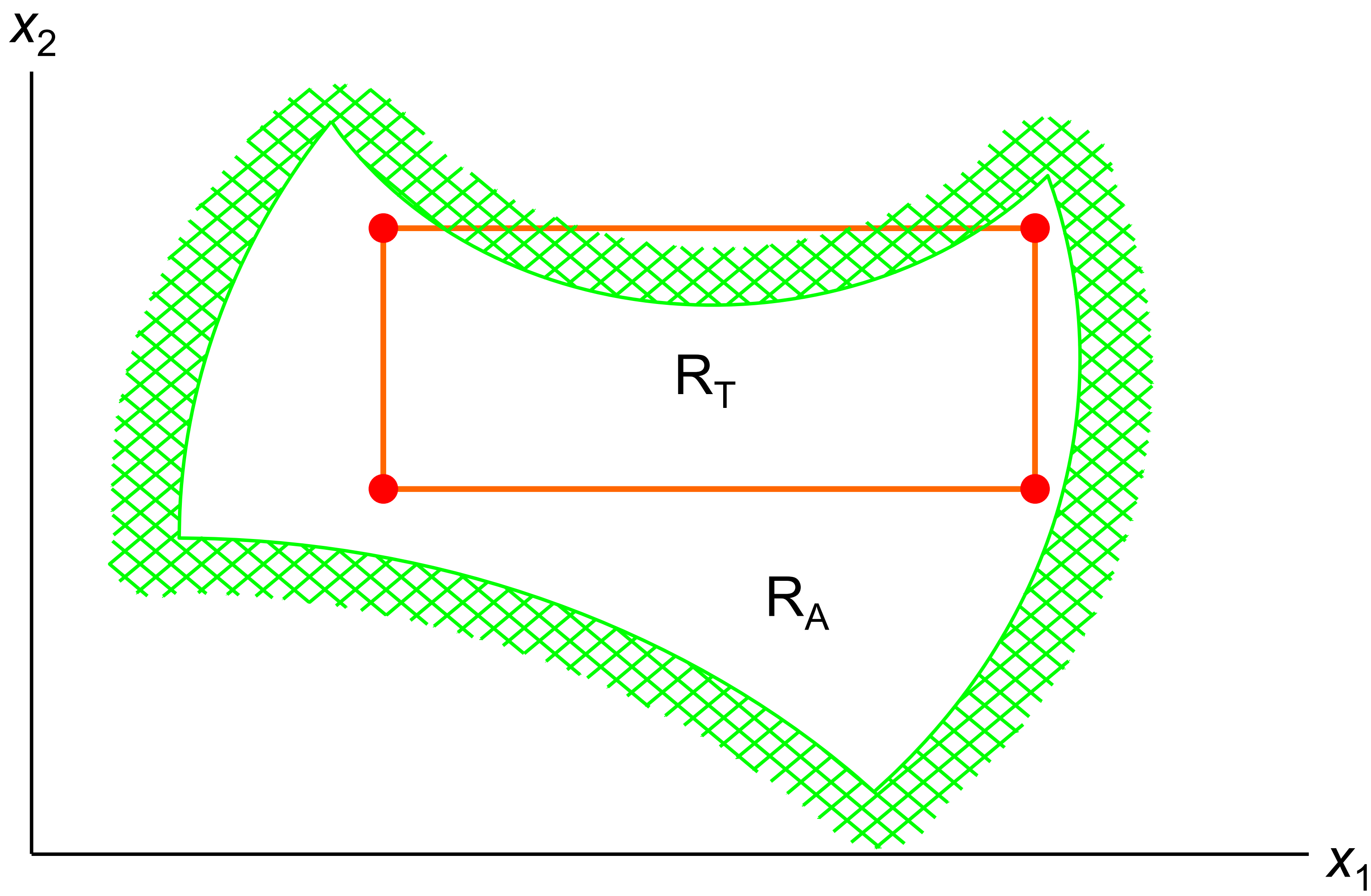
The filter specification requires the gain at ω=1000 r/s to lie between 13.0 dB and 15.0 dB

When applied to the active filter circuit, using resistors and capacitors of 5% tolerance, vertex analysis gives the following results:

Nominal value =	13.98 dB
Highest value =	14.42 dB
Lowest value =	10.22 dB

Vertex analysis therefore predicts that the manufacturing yield will be less than 100%

Limitations of Vertex Analysis



Limitations of Vertex Analysis

Vertex analysis has two other drawbacks:

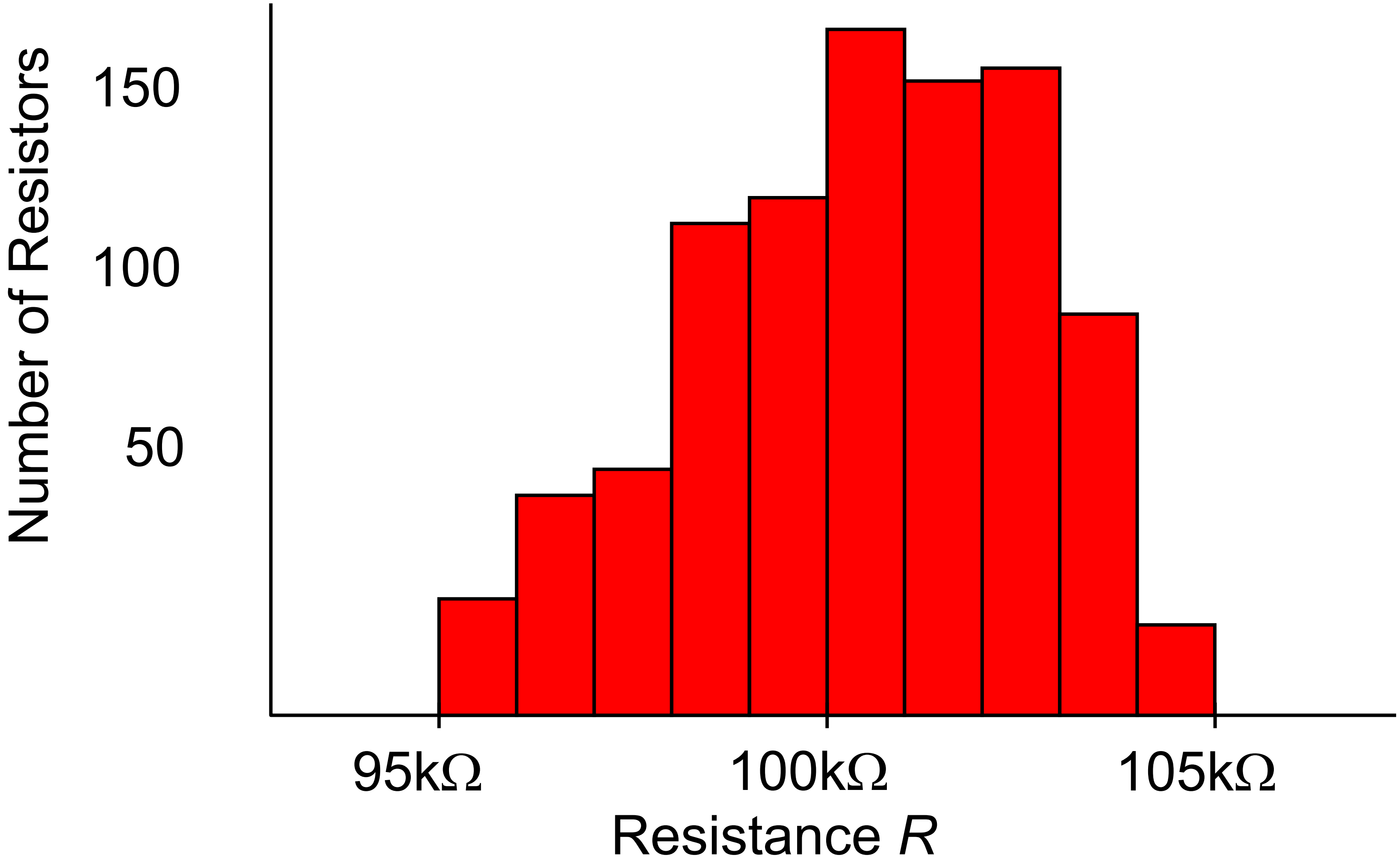
The number of evaluations of the performance function is 2^n and this may be excessive for complex circuits

If n=10 the number of evaluations is 1024 (ok) but if n=20 the number of evaluations is 1048576 (probably impractical)

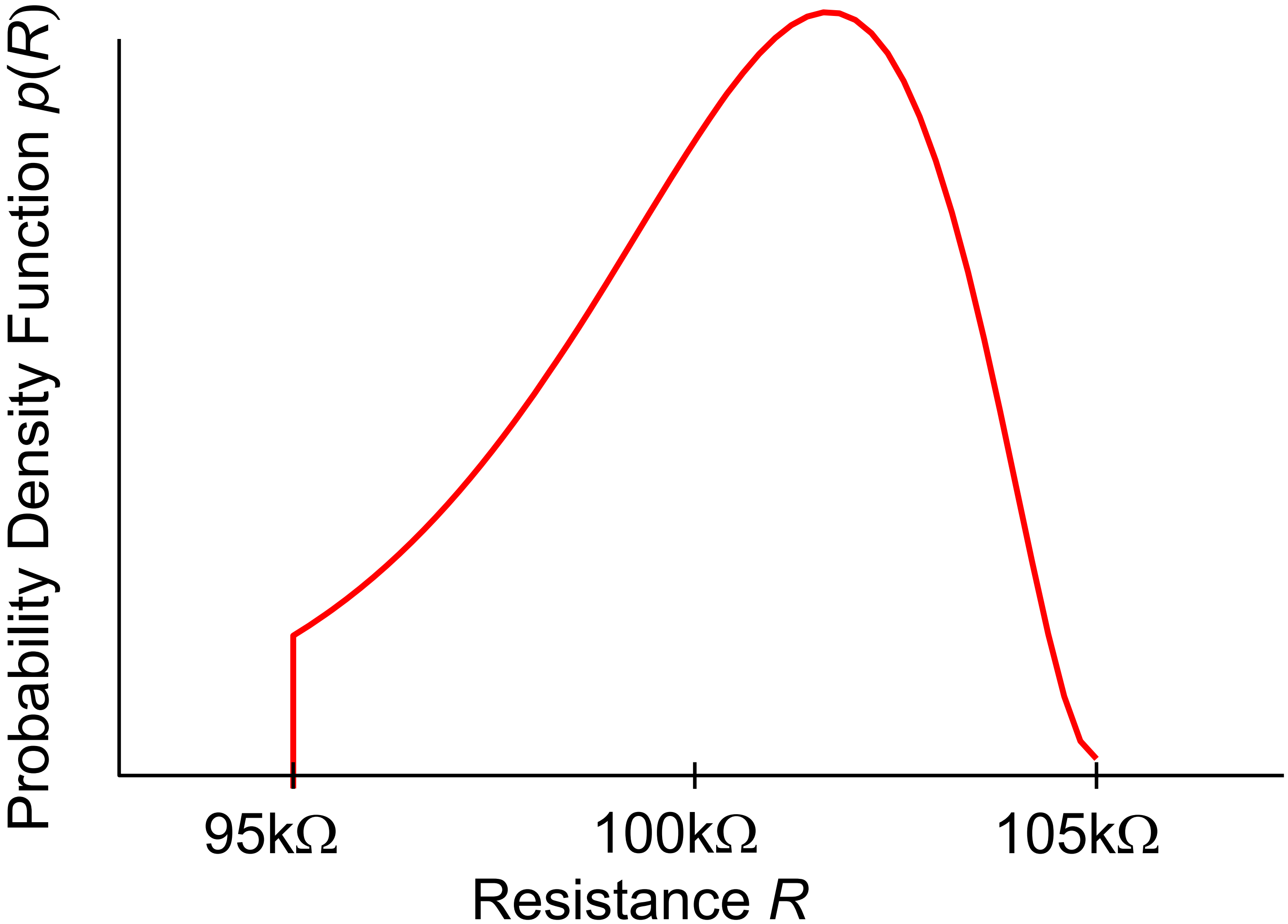
Vertex analysis is pessimistic.

It would indicate that the specification is violated, even if this only occurs with 1 component value combination in 10000

Component Value Distributions



Component Value Distributions



Gaussian Distribution

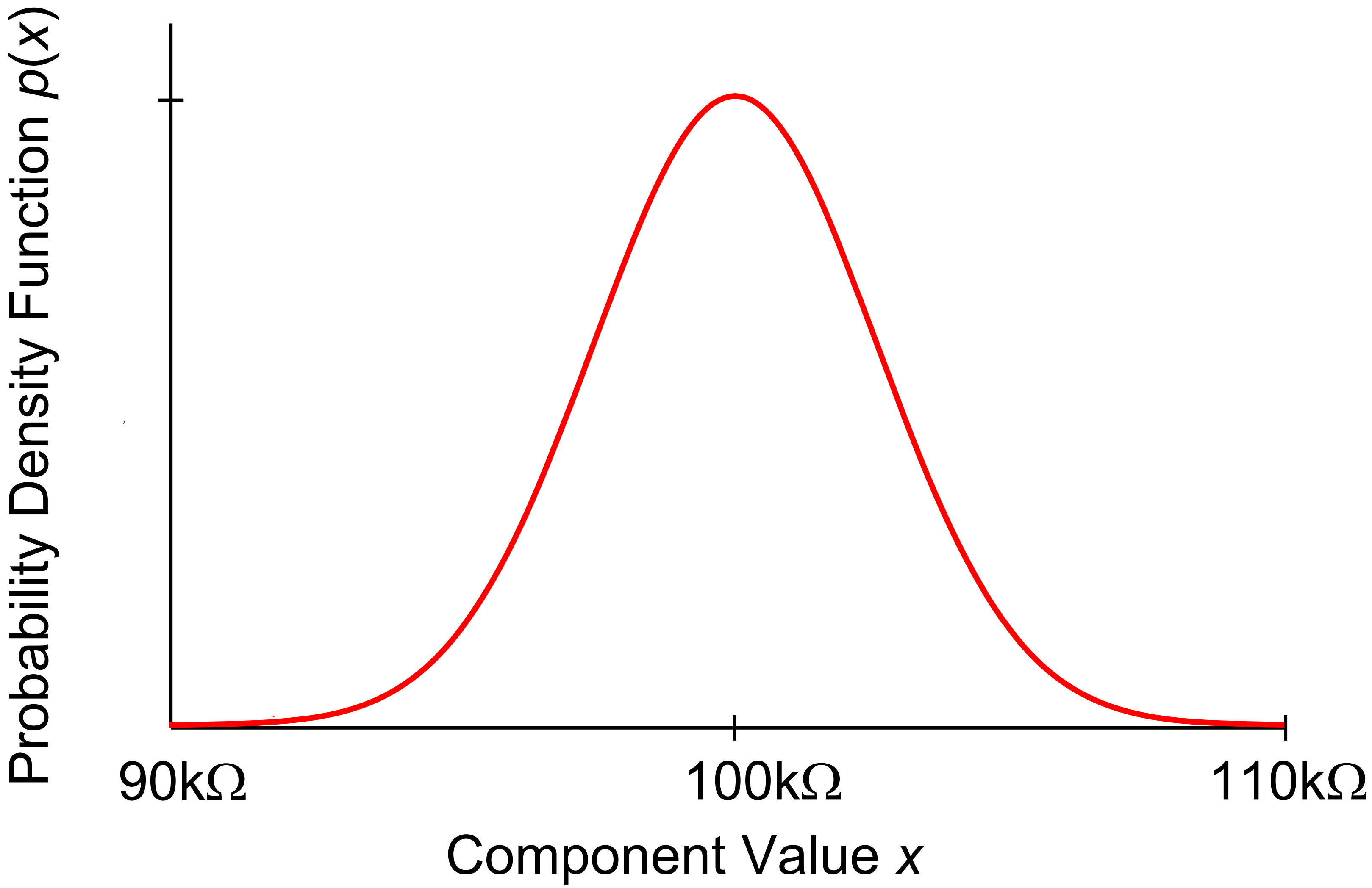
If component value selection is not employed, then then probability density function $p(x)$ for manufactured components is likely to be Gaussian:

$$p(x) \quad = \quad \frac{1}{\sigma\sqrt{2\pi}} \exp - \frac{(x-\mu)^2}{2\sigma^2}$$

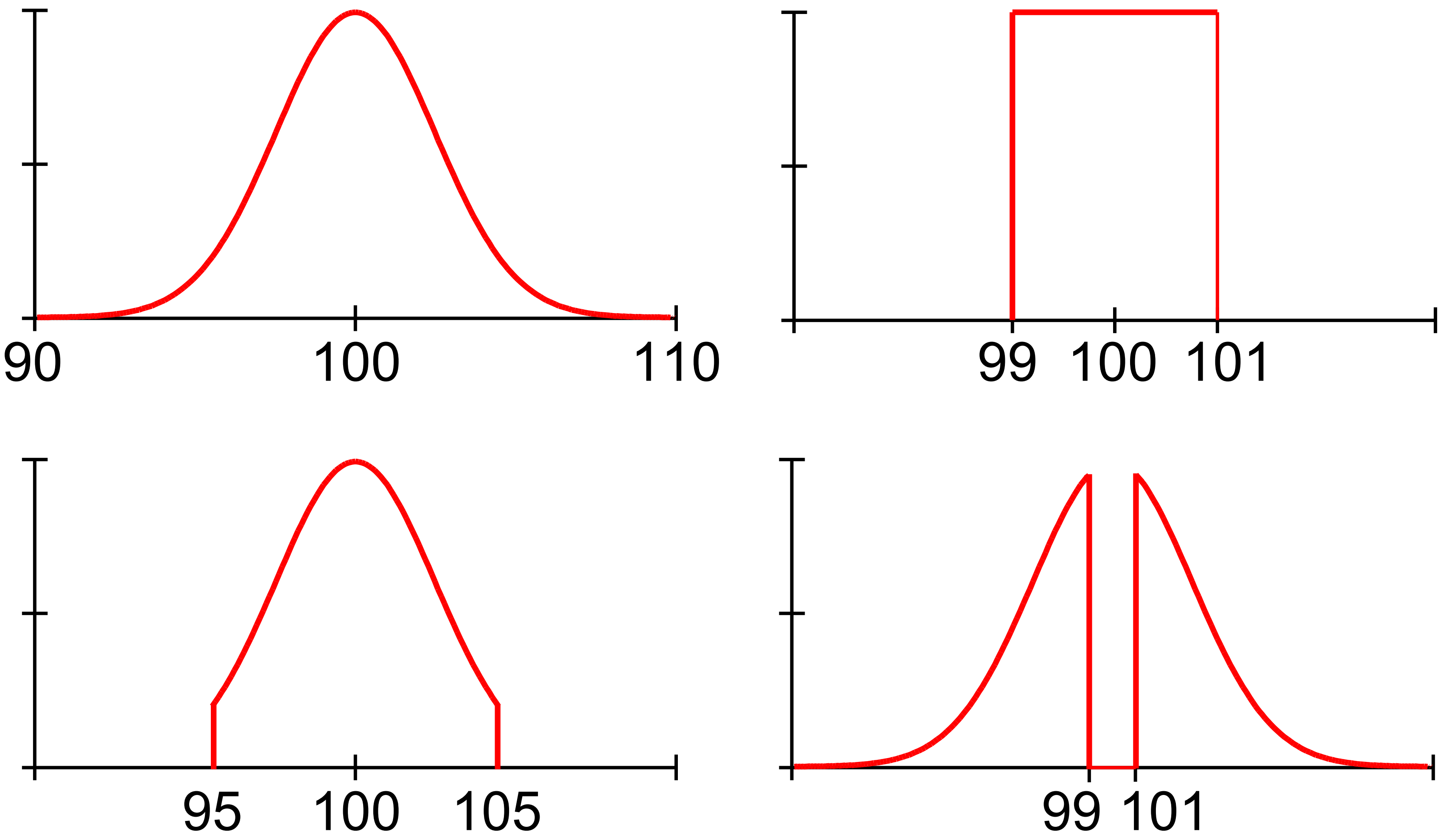
where μ is the mean and σ the standard deviation of the distribution.

This is a consequence of the Central Limit Theorem which states that if a number of random processes contribute to some quantity, then its distribution will be Gaussian irrespective of the distributions of the contributing processes

Gaussian Distribution



Component Value Distributions



Moments of a Distribution

University of Reading

The mean μ is the 1st moment of a distribution:

$$\mu = \int_{-\infty}^{\infty} x p(x)\,\mathrm{d}x$$

The rth moment μ_r of a distribution about the mean is:

$$\mu_r = \int_{-\infty}^{\infty} (x-\mu)^r\, p(x)\,\mathrm{d}x$$

The second moment μ_2 of a distribution is a measure of its width and is called the variance:

$$\mu_2 = \sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2\, p(x)\,\mathrm{d}x$$

Method of Moments

Assume that the dependence of the performance function *f* on the component values is known

Then it should be possible to calculate the moments of the probability density function *p*(*f*) of *f* from the moments of the component value distributions

If *p*(*f*) is known then the manufacturing yield can be determined

In practice a number of simplifying assumptions need to be made

Method of Moments

Each of the component values influences the performance function *f*

The performance function *f* will therefore have an approximately Gaussian probability density function (Central Limit Theorem)

A Gaussian distribution is completely characterised by its mean and variance.

It is therefore only necessary to determine the first and second moments of the distribution function for *f*

Method of Moments

Let $\mathbf{x}^0$ represent the mean values of components. Expand f around $\mathbf{x}^0$:

$$f(\mathbf{x}^0 + \delta\mathbf{x}) \approx f(\mathbf{x}^0) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \delta x_i$$

Suppose that f is evaluated for a large number m of sets of random component values. The mean μ of f is given by:

$$\mu = \frac{1}{m} \sum_{j=1}^{m} f(\mathbf{x}^j) = \frac{1}{m} \sum_{j=1}^{m} \left\{ f(\mathbf{x}^0) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \delta x_i^j \right\}$$

where δx_i^j represents the jth deviation of the ith component from its mean value x_i^0

Method of Moments

Rearranging:

$$\begin{aligned}\mu &= f(\mathbf{x}^0) + \sum_{i=1}^{n}\left\{\frac{\partial f}{\partial x_i}\frac{1}{m}\sum_{j=1}^{m}\delta x_i^j\right\} \\ &\cong f(\mathbf{x}^0) \quad \text{for large } m\end{aligned}$$

In other words the mean of the distribution of *f* is simply the value of *f* evaluated at the mean component values

A similar method can be used to derive the variance σ^2 of the distribution of *f*

Method of Moments

$$\sigma^2 = \frac{1}{m}\sum_{j=1}^{m}\left\{f(\mathbf{x}^j)-\mu\right\}^2 = \frac{1}{m}\sum_{j=1}^{m}\left\{\sum_{i=1}^{n}\frac{\partial f}{\partial x_i}\delta x_i^j\right\}^2$$

The component values are uncorrelated:

$$\text{for } i \neq k: \quad \frac{1}{m}\sum_{j=1}^{m}\delta x_i^j \delta x_k^j \quad \to 0 \quad \text{as} \quad m \to 0$$

If m is large then:

$$\sigma^2 = \sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)^2 \frac{1}{m}\sum_{j=1}^{m}\left(\delta x_i^j\right)^2 = \sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)^2 \sigma_i^2$$

where σ_i^2 is the variance of the *i*th component

Method of Moments

The partial derivatives of f must be evaluated numerically:

$$\frac{\partial f}{\partial x_i} \cong \frac{f(x_1^0, x_2^0, .. x_i^0 + \varepsilon_i, .. x_n^0) - f(x_1^0, x_2^0, .. x_i^0, .. x_n^0)}{\varepsilon_i}$$

If the increments ε_i are chosen to be equal to the standard deviations σ_i of the component values:

$$\sigma^2 \cong \sum_{i=1}^{n} \left\{ f(x_1^0, x_2^0, .. x_i^0 + \sigma_i, .. x_n^0) - f(\mathbf{x}^{\mathbf{0}}) \right\}^2$$

Method of Moments

The mean μ and the variance σ^2 of the probability density function $p(f)$ can now be determined

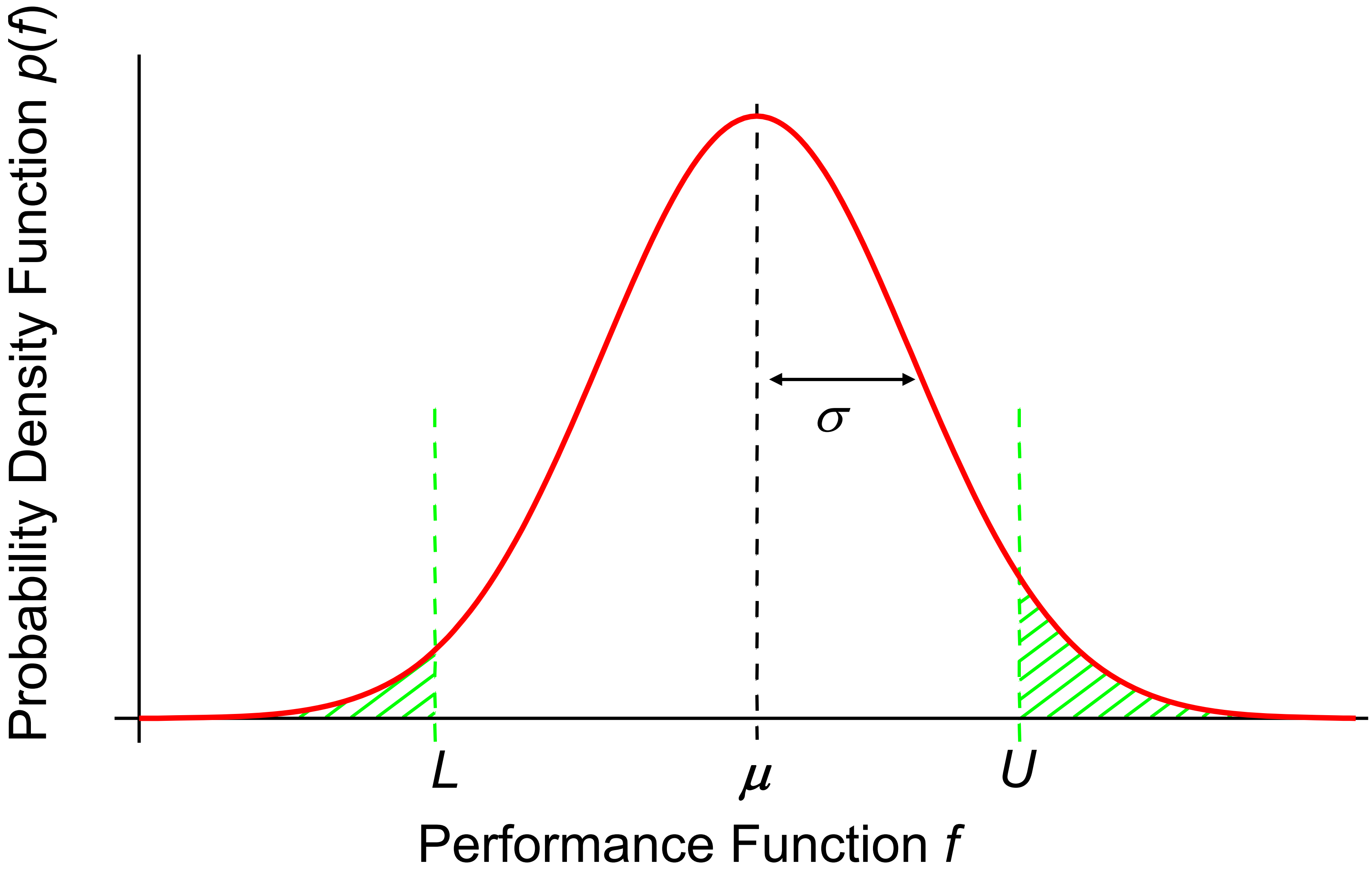
Assuming that $p(f)$ is Gaussian:

$$p(f) \quad = \quad \frac{1}{\sigma\sqrt{2\pi}} \exp - \frac{(f-\mu)^2}{2\sigma^2}$$

Suppose that L and U represent the lower and upper specification limits for f

The areas underneath $p(f)$ below L and above U represent the probability of failure

Probability of Failure



Method of Moments

The probability of failure through f being below the lower limit L is:

$$p_L = \int_{-\infty}^{L} p(f)\,\mathrm{d}f$$

The probability of failure through f being above the upper limit U is:

$$p_U = \int_{U}^{\infty} p(f)\,\mathrm{d}f$$

Unfortunately a Gaussian function cannot be integrated algebraically and p_L and p_U must be evaluated numerically

Method of Moments

The complementary error function erfc(x) is defined:

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} \exp(-y^2)\, \mathrm{d}y$$

The failure probabilities can be written in terms of erfc:

$$p_L = \frac{1}{2}\,\mathrm{erfc}\left(\frac{\mu - L}{\sigma\sqrt{2}}\right)$$

$$p_U = \frac{1}{2}\,\mathrm{erfc}\left(\frac{U - \mu}{\sigma\sqrt{2}}\right)$$

The values of erfc can be obtained from statistical tables

Method of Moments

The filter specification requires the gain at ω=1000 rad/s to lie between 13.0 dB and 15.0 dB

When applied to the active filter circuit, using resistors and capacitors of $5/\sqrt{3}$ % standard deviation, moments analysis gives the following results:

Mean of $p(f)$ = 13.98
SD of $p(f)$ = 0.456
Failed rate low = 1.58%
Failed rate high = 1.25%
Total failure rate = 2.83%

Limitations of Method of Moments

The method of moments is a fast but approximate technique for calculating the effects of component tolerances.

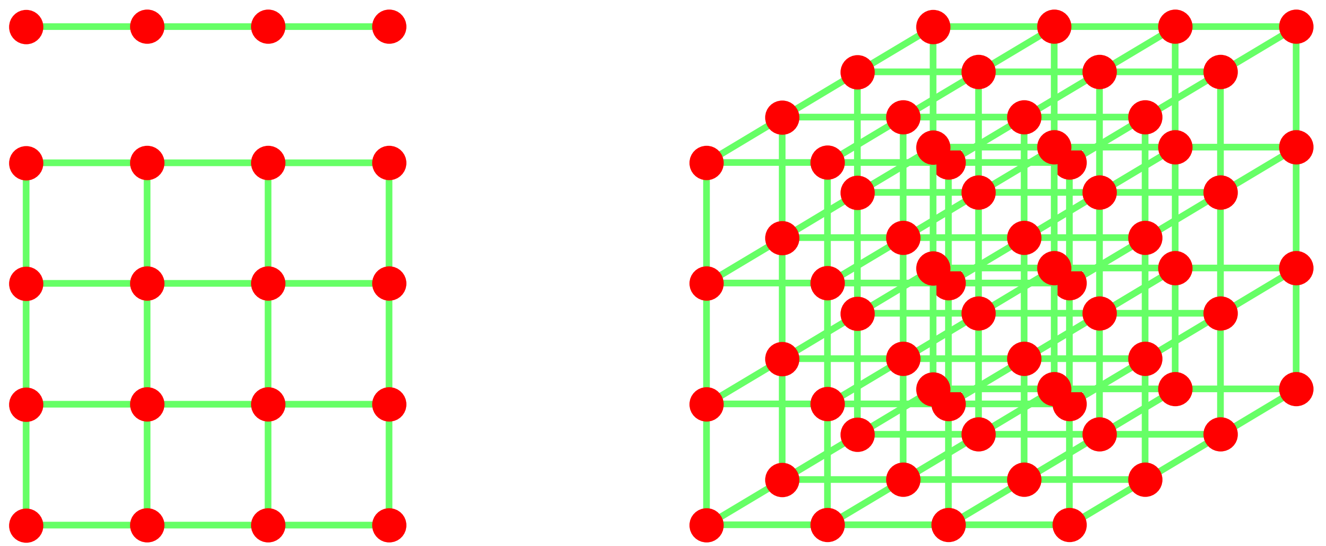
Because the Taylor series is truncated after the first-order term the method will only give good results if the component tolerances are small

The assumption that the distribution of f is Gaussian is only valid if the individual component distributions are themselves Gaussian, or alternatively if there are a large number of components.

Monte-Carlo Analysis

Monte-Carlo analysis is method for exploring the component space in order to establish the region of acceptability Ra

It exhibits *dimensional independence*:



Monte-Carlo Analysis

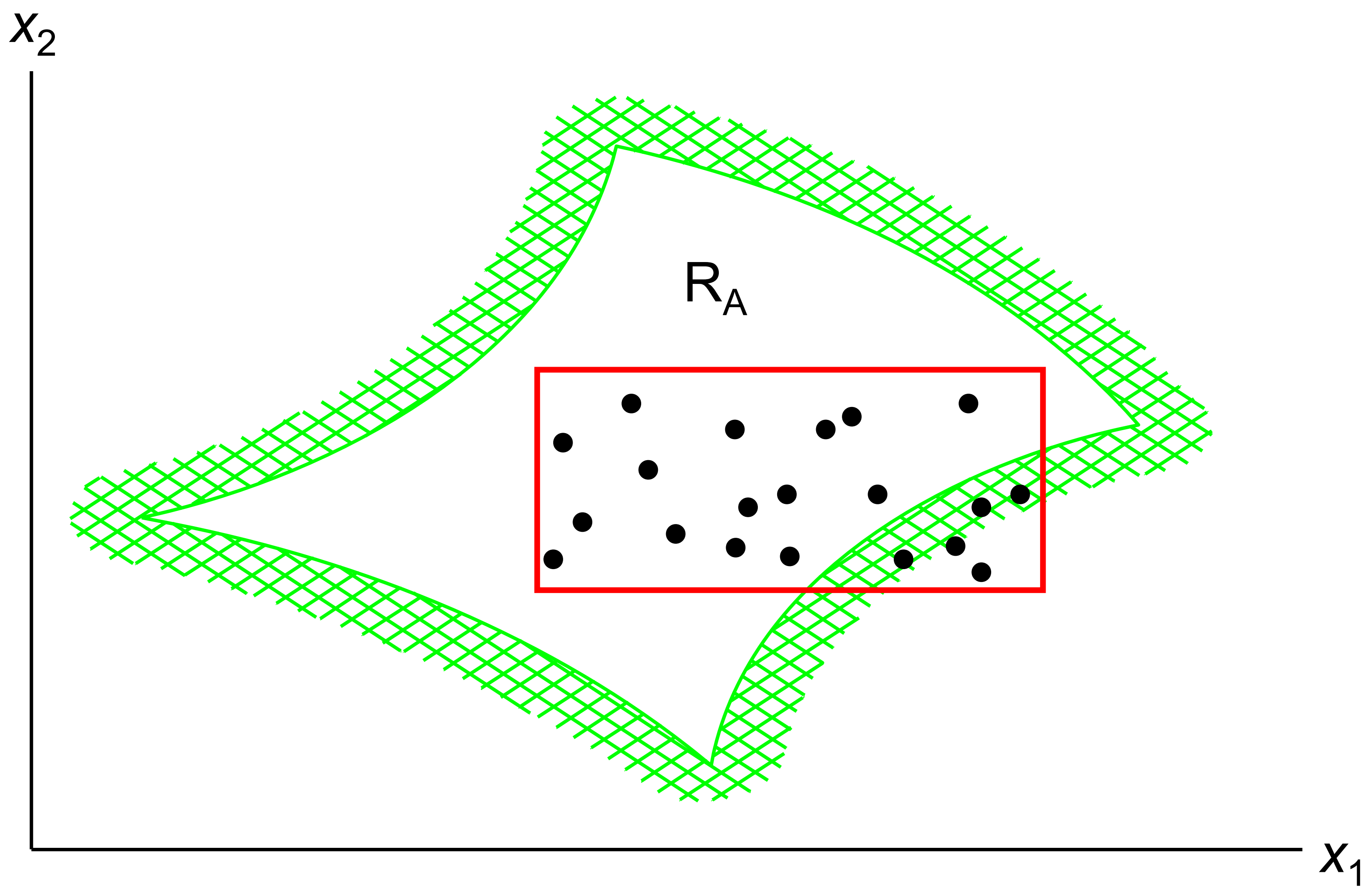
Monte-Carlo tolerance analysis simulates the manufacturing process

Each component is given a random value in accordance with its known distribution. The performance of the circuit is calculated and compared with the specification

This trial is repeated a large number of times and the failure rate calculated

Since it is a statistical method it can only give an estimate of the manufacturing yield.

Estimating the Yield



Monte-Carlo Analysis

The filter specification requires the gain at ω=1000 rad/s to lie between 13.0 dB and 15.0 dB

When applied to the active filter circuit, using resistors and capacitors with a uniform distribution of values and 5 % tolerance, Monte-Carlo analysis gives the following results:

Number of trials = 1000
Failed rate low = 103
Failed rate high = 0
Total failure rate = 10.3%

Failure Rates

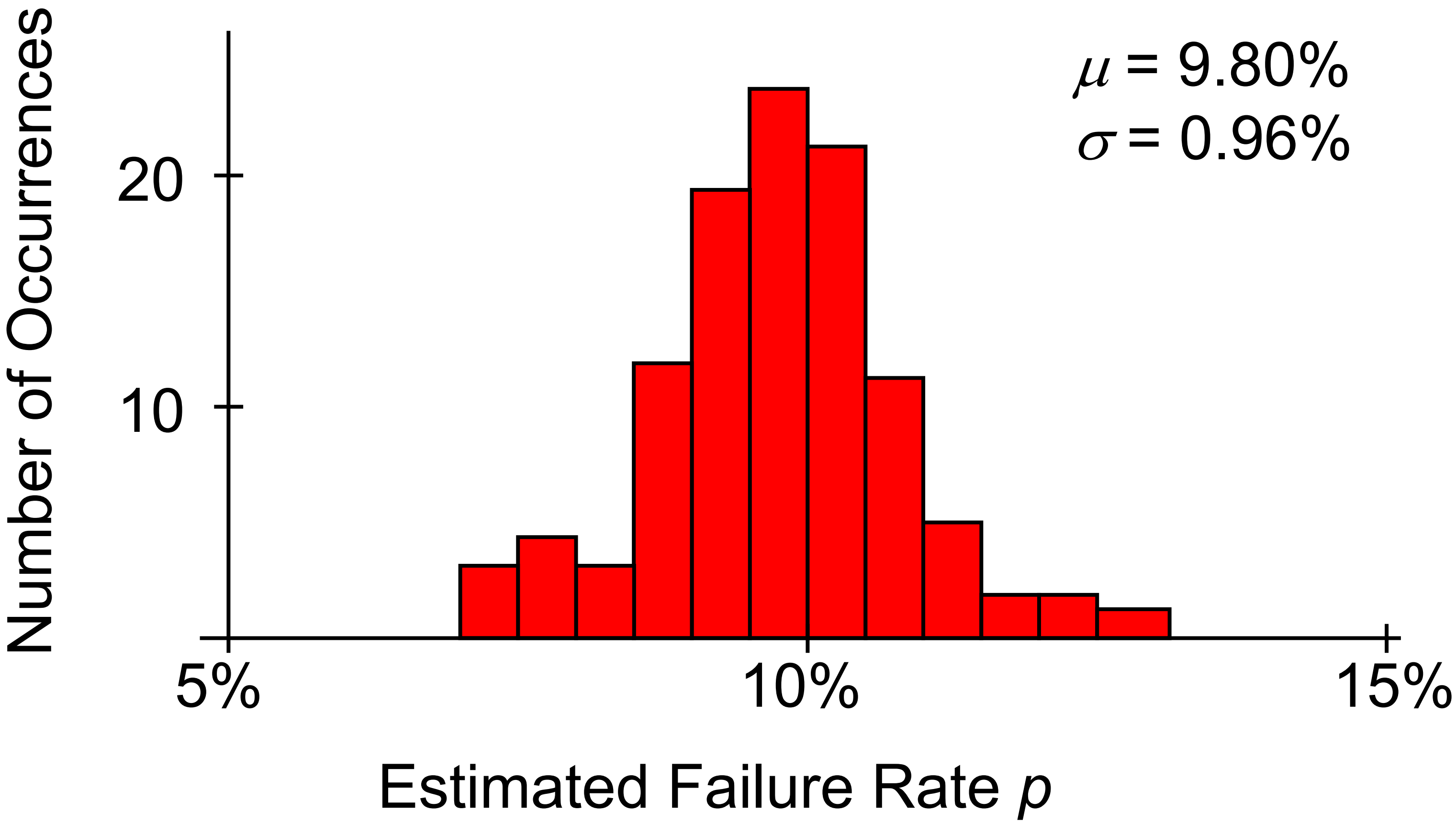
The true failure probability p_∞ is defined to be the proportion of component value combinations that would fail to meet the specification in an infinite number of trials

How does the measured failure probability p=10.3% obtained for 1000 trials relate to p_∞ ?

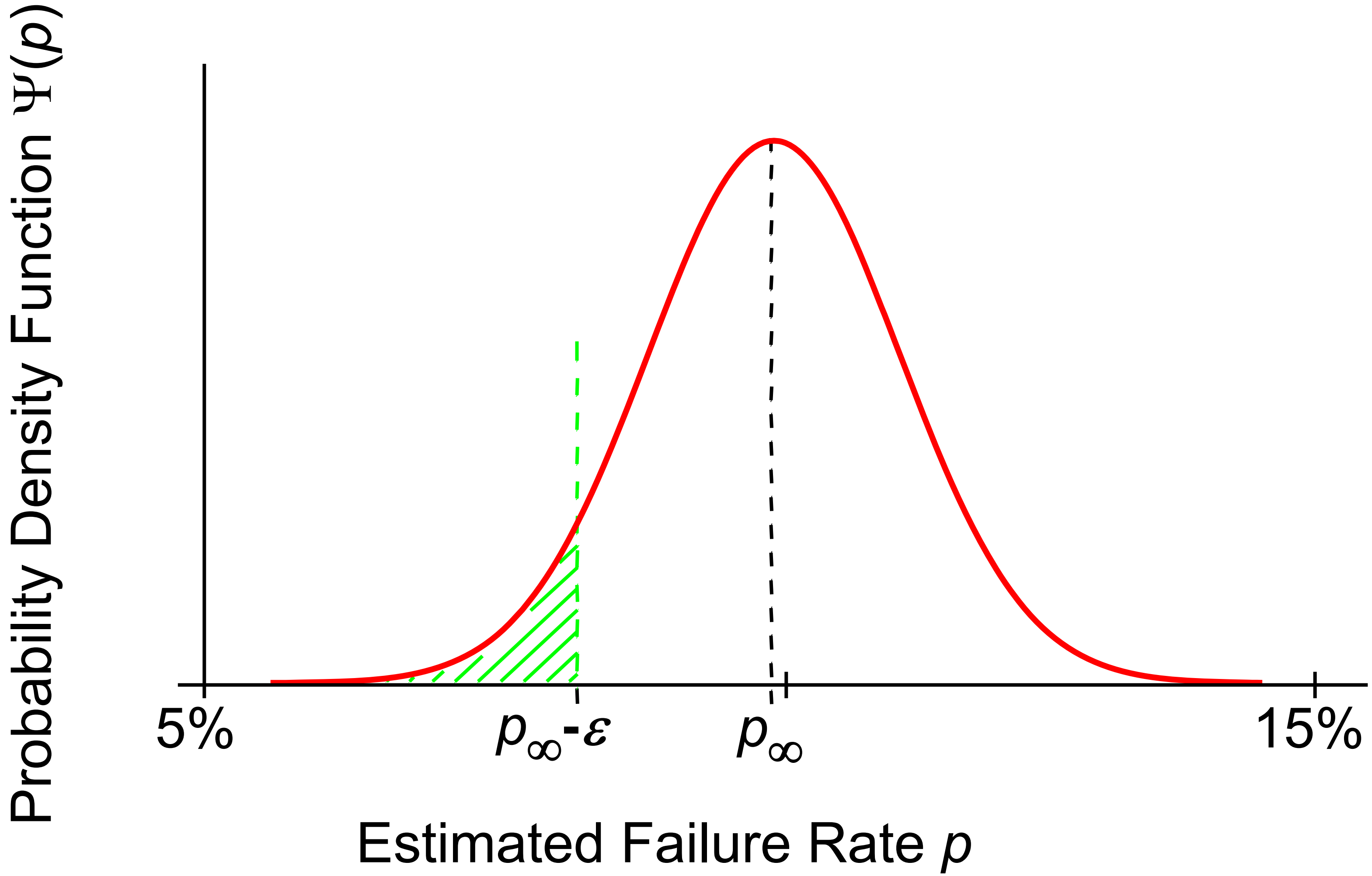
Further Monte-Carlo analyses of 1000 trials gave failure probabilities of: 9.4%, 10.6%, 8.8%, 9.9%.

Combining gives p=9.8% from 5000 trials

Estimated Failure Rates



Estimated Failure Rates



Estimated Failure Rates

The probability of underestimation P_ε is equal to the area of the Gaussian function below p_∞-ε:

$$P_\varepsilon = \int_{-\infty}^{p_\infty - \varepsilon} \Psi(p)\, \mathrm{d}p = \frac{1}{2} \operatorname{erfc} \frac{\varepsilon}{\sigma_p \sqrt{2}}$$

Suppose that a monte-carlo analysis produces an estimated failure rate p; then:

$$p_\infty < p + \varepsilon$$

with confidence level 1-P_ε

Also:

$$p_\infty > p - \varepsilon$$

with the same confidence level 1-P_ε

Estimated Failure Rates

If $\varepsilon = 2\sigma$ then the probability of underestimation is 2.3%. Thus:

$$p_\infty \quad < \quad p + 2\sigma$$

at the 97.7% confidence limit

If $\varepsilon = 3\sigma$ then the probability of underestimation is 0.13%. Thus:

$$p_\infty \quad < \quad p + 3\sigma$$

at the 99.87% confidence limit

The remaining problem is to determine the standard deviation σ of the distribution $\Psi(p)$

Estimated Failure Rates

In an N-trial Monte-Carlo analysis, the failure rate p is the sum of the results of the individual trials divided by N

The variance σ^2 of the probability density function $\psi(p)$ is given by:

$$\sigma^2 = \sigma_q^2 / N$$

where $\sigma_q{}^2$ is the variance of individual trials.

Each trial has value 1 with probability p_∞ and value 0 with probability 1-p_∞ :

$$\sigma_q^2 = p_\infty(1 - p_\infty)^2 + (1 - p_\infty)p_\infty^2 = p_\infty(1 - p_\infty)$$

Estimated Failure Rates

The variance σ^2 of the probability density function $\Psi(p)$ is given by:

$$\sigma^2 = p_\infty(1 - p_\infty)/N$$

Unfortunately p_∞ is not know, but the estimate p from the Monte-Carlo analysis should be a reasonable approximation to p_∞ :

$$\sigma^2 \cong p(1-p)/N \cong p/N$$

The 1000-trial Monte-Carlo analysis gave a failure rate p = 10.3% which gives σ = 0.96%

Therefore the true failure rate p_∞ is less than 12.2% at the 97% confidence level

Random Numbers

Random numbers are generated initially with a uniform distribution in the range 0.0 to 1.0

Any value in the range is equally likely

A random number must be entirely independent of previously generated numbers

Non-uniform distributions are generated by applying transformations to uniform random numbers

True random numbers cannot be generated on a normal computer

Pseudo-Random Numbers

The numbers used in Monte-Carlo analysis do not need to be true random numbers.

Monte-Carlo analysis can equally-well be performed using pseudo-random numbers

Pseudo-random numbers are indistinguishable from true random numbers according to any statistical tests, and yet are completely predictable.

Never rely on the pseudo-random number generator supplied with a compiler

Pseudo-Random Numbers

All pseudo-random number generators initially generate a sequence of large pseudo-random integers within a specified range.

These are then converted to real values in the range 0.0 to 1.0

All pseudo-random number generators are cyclic - that is after a certain number of values the sequence repeats.

The length of the sub-sequence, which by repetition forms the complete sequence, is known as the period

Pseudo-Random Numbers

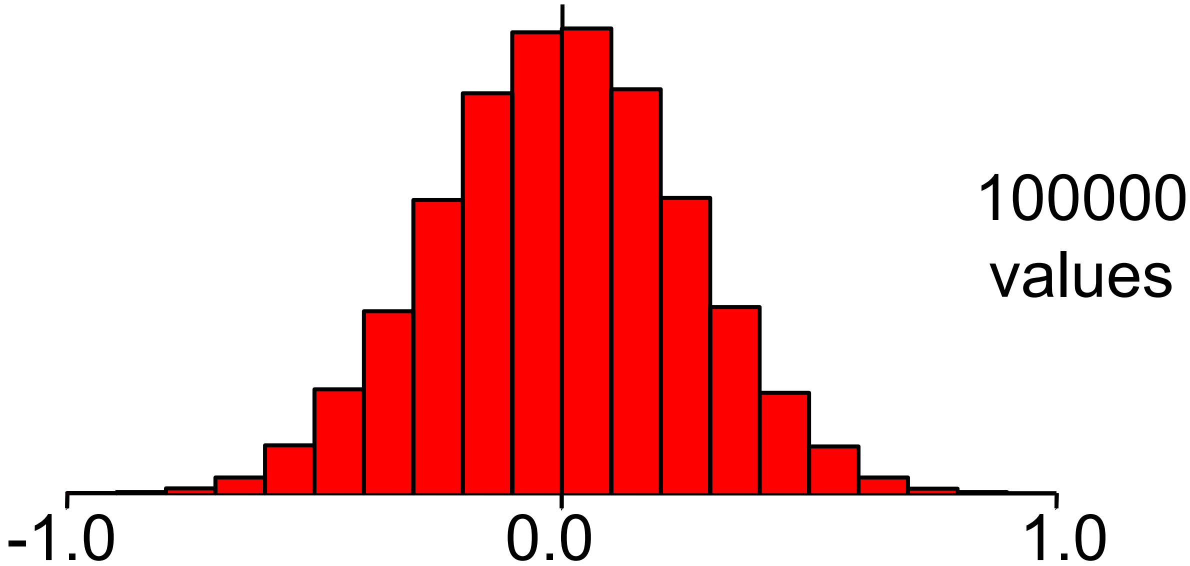
```
const double m = 4294967296.0;
const unsigned long int d  = 0x09E3779B9L;
const unsigned long int k0 = 0x0C7D7A8B4L;
const unsigned long int k1 = 0x09ABFB3B6L;
const unsigned long int k2 = 0x073DC1683L;
const unsigned long int k3 = 0x017B7BE43L;

unsigned long int y = 123456789L;
unsigned long int z = 987654321L;

double rnd()
{
    unsigned long int s = 0;
    unsigned int n = 32;
    while (n-- > 0) {
        s += d;
        y += (z<<4) + k0^z + s^(z>>5) + k1;
        z += (y<<4) + k2^y + s^(y>>5) + k3;
    }
    return (z + y / m) / m;
}
```

Gaussian Distribution

```
double gaussian()
{
    double q = -6.0;
    int i;
    for (i = 0; i < 12; ++i)
        q += rnd();
    return q / 4.0;
}
```



Gaussian Distribution

An exact method for generating a Gaussian distribution is to use the Box-Müller method

If x_1 and x_2 are random numbers with a uniform distribution, then:

$$y_1 = \sqrt{-2\log_e x1}\cos 2\pi x_2$$

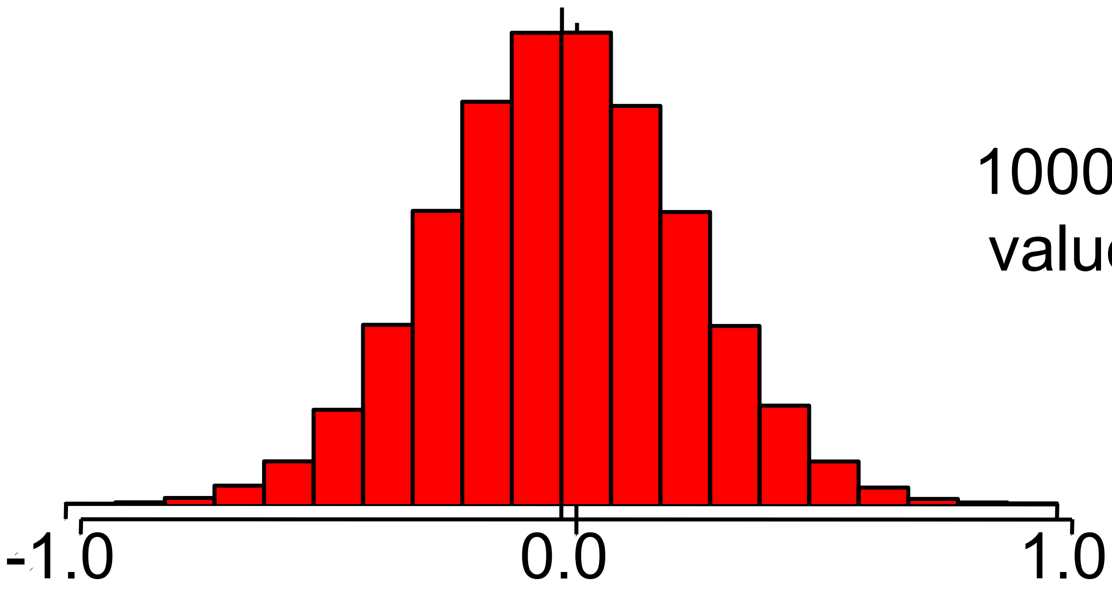
$$y_2 = \sqrt{-2\log_e x1}\sin 2\pi x_2$$

y_1 and y_2 are independent random numbers with Gaussian distributions

For the Box-Müller method to work it is essential that x_1 and x_2 are uncorrelated.

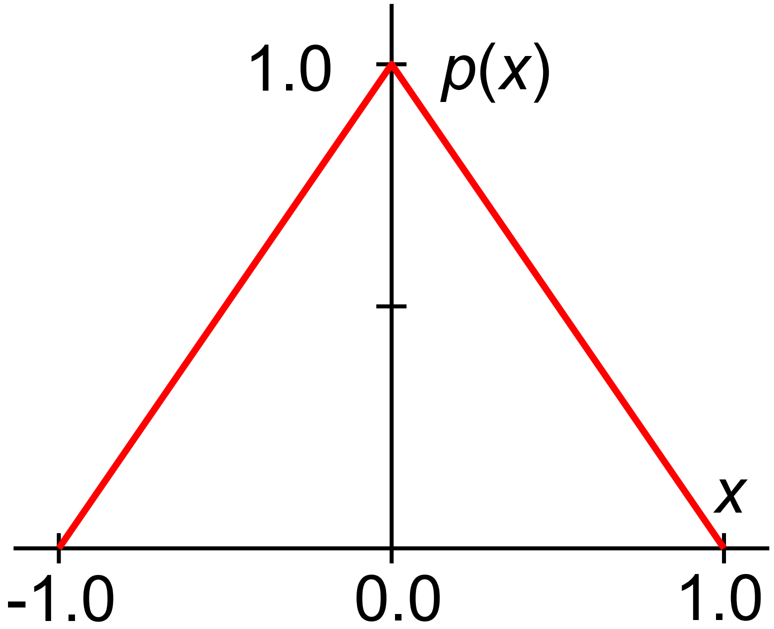
Gaussian Distribution

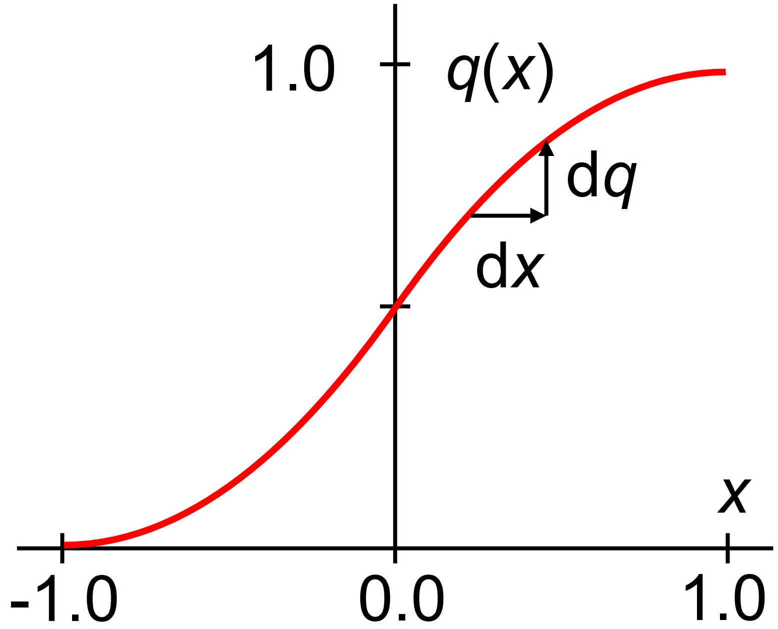
```
double gaussian()
{
    return sqrt(-2.0 * log(rnd()))
           * cos(2.0 * pi * rnd()) / 4.0;
}
```



100000 values

Non-Uniform Distributions





$$
\begin{aligned}
p(x) &= 0 && \text{for} \quad x < -1 \\
&= 1 + x && \text{for} \quad -1 \le x < 0 \\
&= 1 - x && \text{for} \quad 0 \le x < 1 \\
&= 0 && \text{for} \quad 1 \le x
\end{aligned}
$$

Non-Uniform Distributions

$$q(x) = \int_{-\infty}^{x} p(x)\,\mathrm{d}x \quad \rightarrow \quad \frac{\mathrm{d}q}{\mathrm{d}x} = p(x)$$

Uniform random numbers q are generated in the range 0.0 - 1.0. Probability that a number will lie in range q to q+dq is dq

Convert random number q to $x \equiv x(q)$

Probability that the number $x \equiv x(q)$ will lie in range x to x+dx is also dq = $p(x)$ dx

This random number x therefore has the required distribution function: $p(x)$

Non-Uniform Distributions

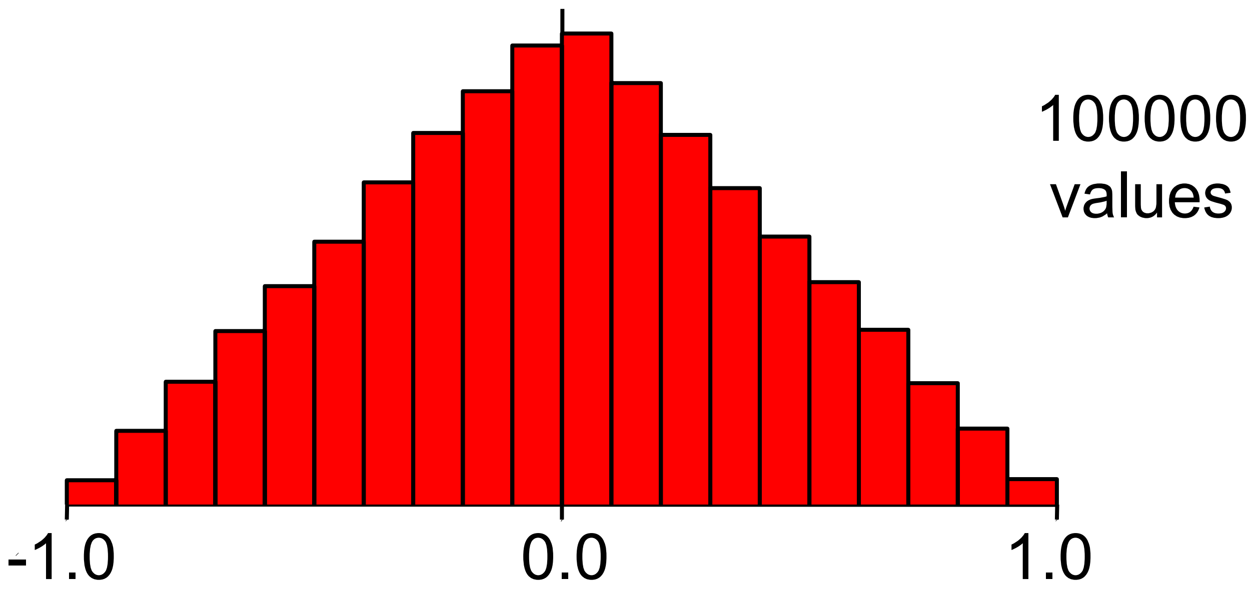
$$
\begin{aligned}
p(x) &= 0 && \text{for} \quad x < -1 \\
&= 1 + x && \text{for} \quad -1 \le x < 0 \\
&= 1 - x && \text{for} \quad 0 \le x < 1 \\
&= 0 && \text{for} \quad 1 \le x
\end{aligned}
$$

$$
\begin{aligned}
q(x) &= 0 && \text{for} \quad x < -1 \\
&= 0.5 + x + 0.5x^2 && \text{for} \quad -1 \le x < 0 \\
&= 0.5 + x - 0.5x^2 && \text{for} \quad 0 \le x < 1 \\
&= 1 && \text{for} \quad 1 \le x
\end{aligned}
$$

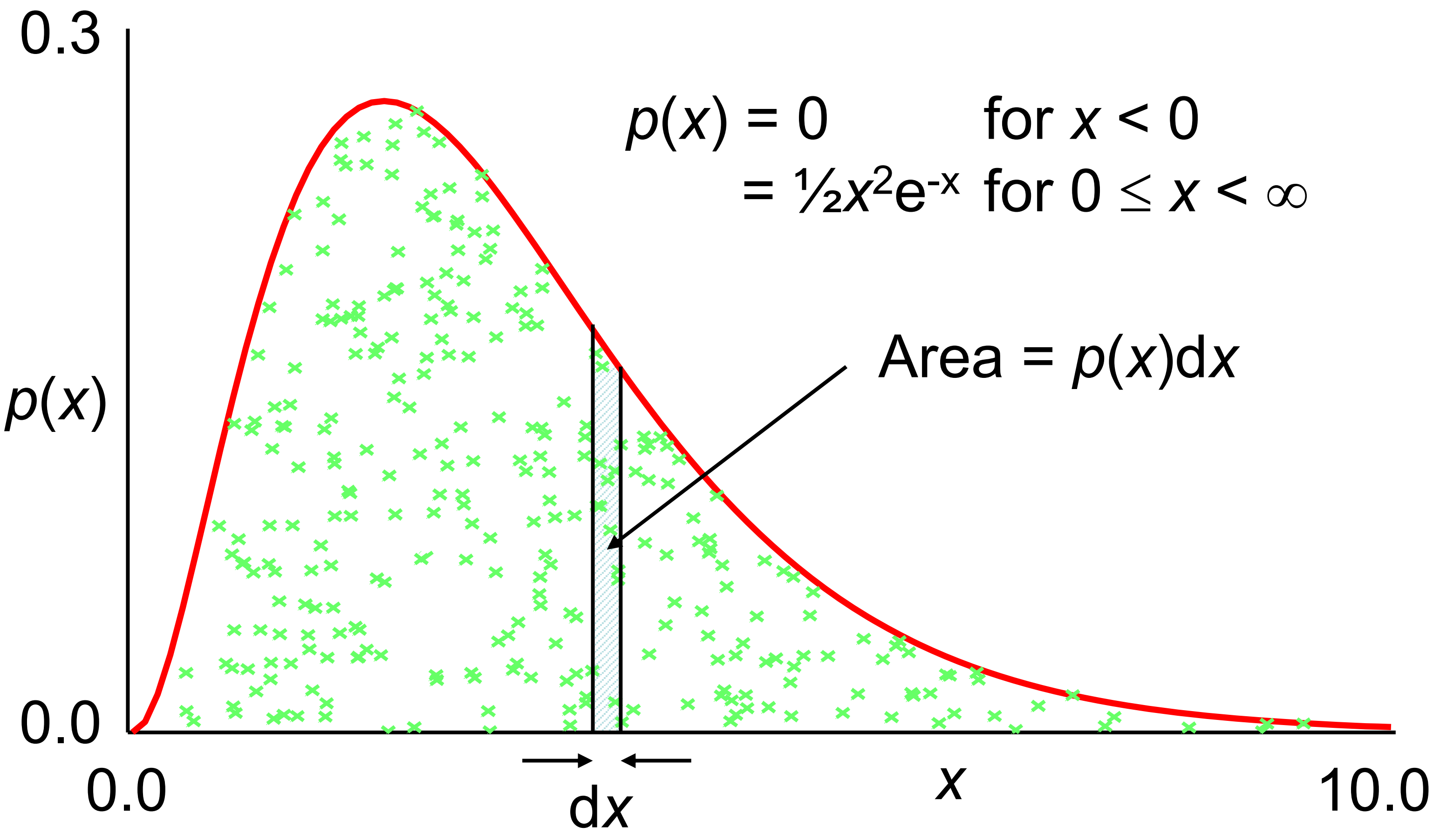
$$
\begin{aligned}
x &= -1 + \sqrt{2q} && \text{for} \quad q < 0.5 \\
&= 1 - \sqrt{2 - 2q} && \text{for} \quad q \ge 0.5
\end{aligned}
$$

Non-Uniform Distributions

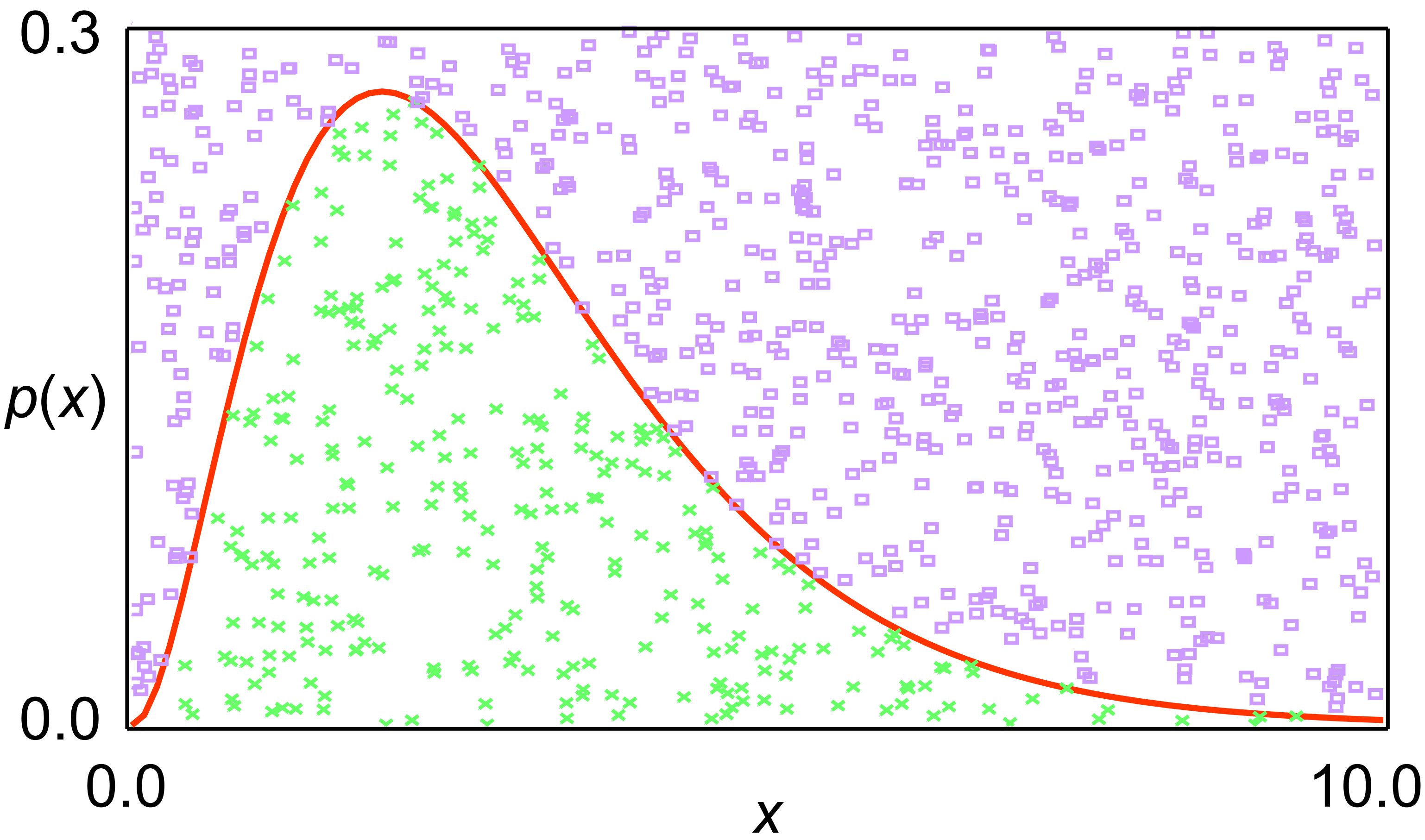
```
double triangular_distribution()
{
    double q = rnd();
    if (q < 0.5)
        return -1.0 + sqrt(2.0 * q);
    else
        return 1.0 - sqrt(2.0 - 2.0 * q);
}
```



Accept-Reject Method

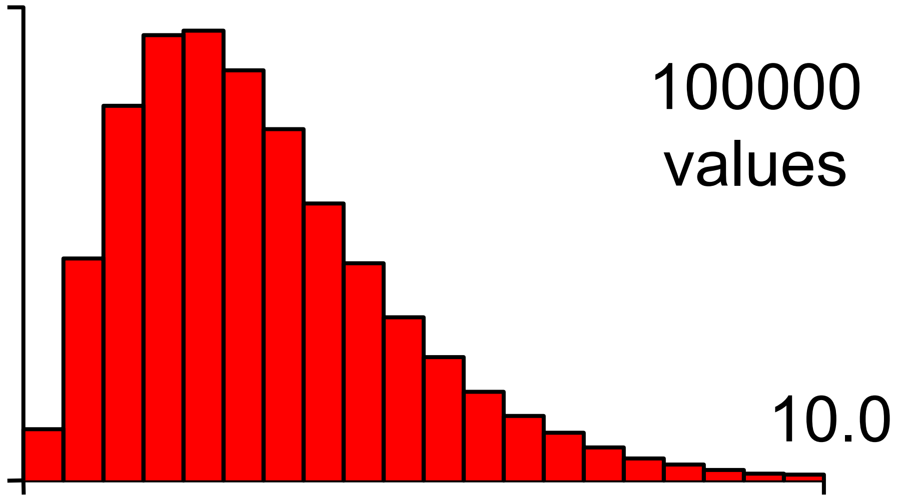


Accept-Reject Method

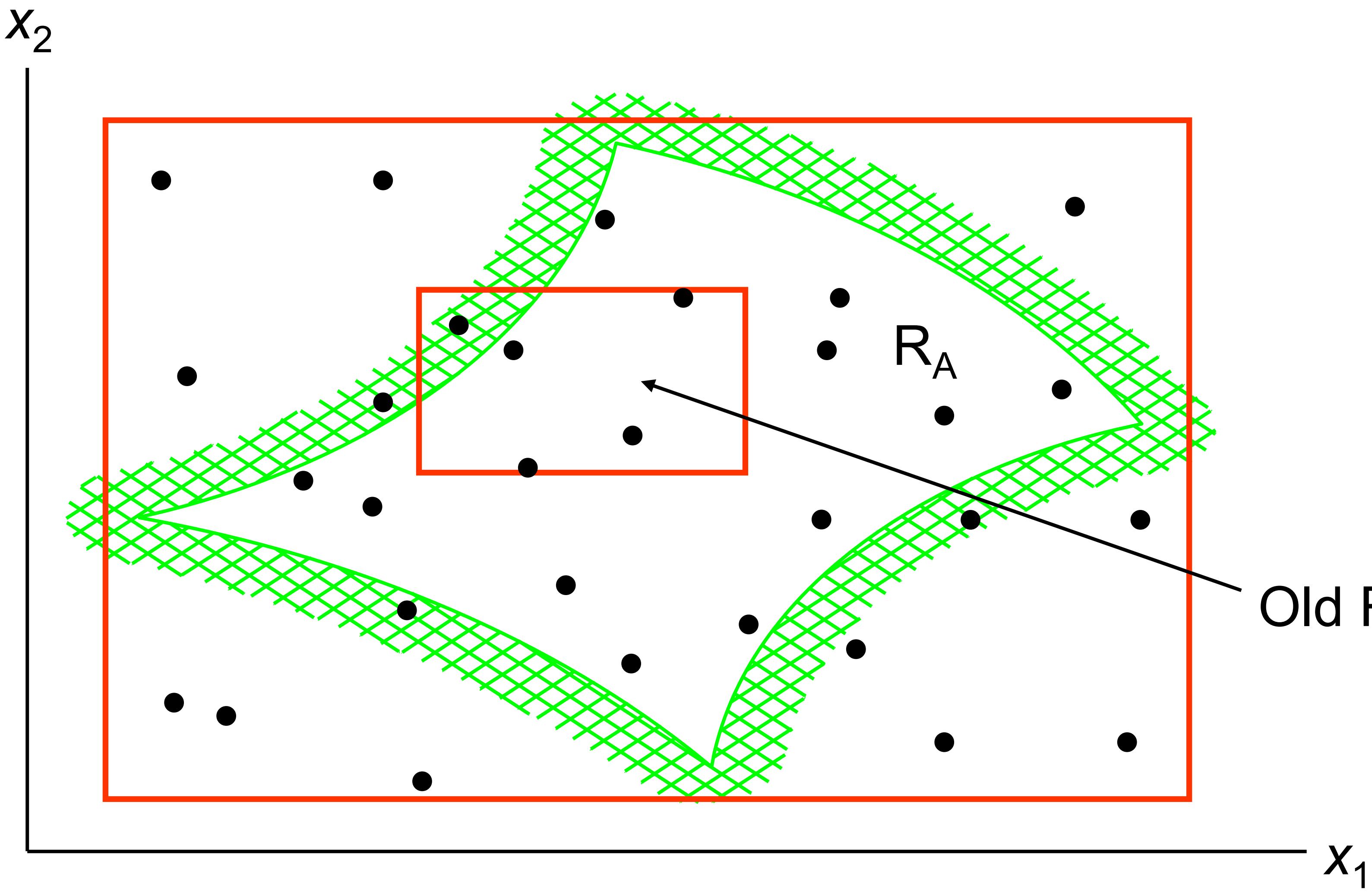


Accept-Reject Method

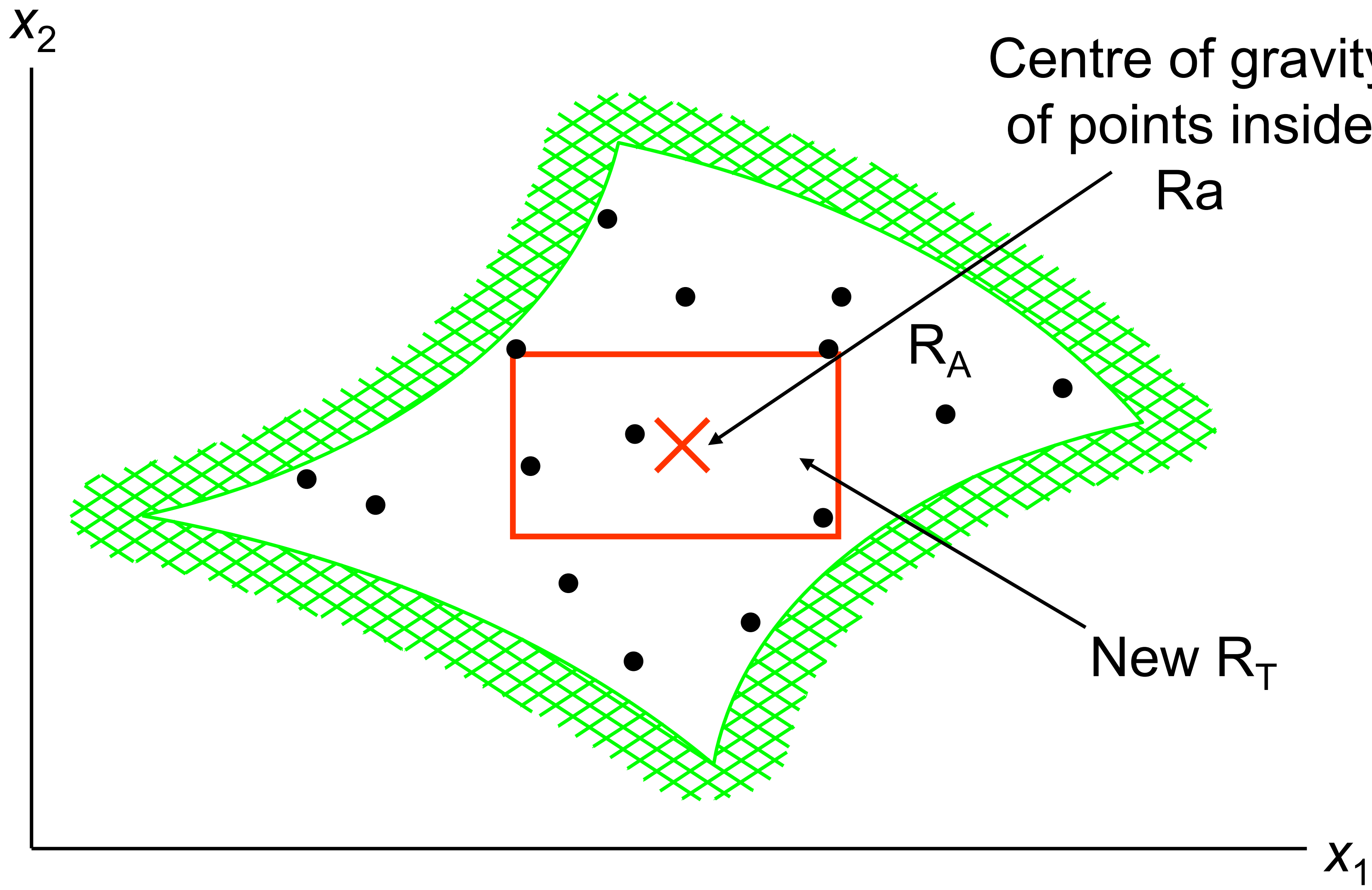
```
double accept_reject()
{
    const double xm = 10.0, ym = 0.3;
    double x, y, pdf;
    do {
        x = xm * rnd();
        y = ym * rnd();
        pdf = p(x);
    } while (pdf < y);
    return x;
}
```



Design Centering



Design Centering



Tolerance Design