STK3405 – Lecture 14

A. B. Huseby

Department of Mathematics
University of Oslo, Norway



Section 7.1

Non-repairable binary monotone systems



Non-repairable binary monotone systems

Let (C, ϕ) be a binary monotone system, where $C = \{1, \ldots, n\}$, and introduce:

$$X_i(t) = \text{The state of component } i \text{ at time } t, \quad t \geq 0, \quad i \in C$$

NOTE: $X_i(t)$ is a random variable, while the development of the state of component i over time, $\{X_i(t)\} = \{X_i(t) : t \geq 0\}$, is a stochastic process.

We assume that $\{X_1(t)\}, \ldots, \{X_n(t)\}$ are independent of each other.

We also let:

$$\phi(\boldsymbol{X}(t)) = \text{The state of the system at time } t, \quad t \geq 0.$$

NOTE: $\phi(\boldsymbol{X}(t))$ is a random variable, and the development of the state of the system over time, $\{\phi(\boldsymbol{X}(t))\} = \{\phi(\boldsymbol{X}(t)) : t \geq 0\}$ is a stochastic process.



Non-repairable binary monotone systems (cont.)

Time-dependent component and system reliabilities are defined as follows for $t \geq 0$ and $i \in C$:

$$\begin{aligned} p_i(t) &= P(X_i(t) = 1) = \text{The reliability of component } i \text{ at time } t, \\ h(\boldsymbol{p}(t)) &= P(\phi(\boldsymbol{X}(t)) = 1) = \text{The reliability of the system at time } t. \end{aligned} \tag{1}$$

These quantities are closely related to the lifetimes of the system and the components. To show this we let:

$$\begin{aligned} T_i &= \text{The lifetime of component } i, \text{ for } i \in C, \\ S &= \text{The lifetime of the system.} \end{aligned} \tag{2}$$

and let:

$$\begin{aligned} P(T_i \leq t) &= F_i(t), \quad i \in C, \\ P(S \leq t) &= G(t) \end{aligned}$$



Non-repairable binary monotone systems (cont.)

We then have the following relations:

$$p_i(t) = P(X_i(t) = 1) = P(T_i > t) = 1 - F_i(t) = \bar{F}_i(t), \tag{3}$$

$$h(t) = P(\phi(\boldsymbol{X}(t)) = 1) = P(S > t) = 1 - G(t) = \bar{G}(t). \tag{4}$$

and

$$\bar{G}(t) = h(\boldsymbol{p}(t)) = h(\bar{F}_1(t), \ldots, \bar{F}_n(t)). \tag{5}$$

- $\bar{F}_i(t)$ is called the survival function of the component lifetime T_i, $i \in C$
- $\bar{G}$ is called the survival function of the system lifetime S



Failure rates

Let $\bar{F}(t) = P(T > t)$ for the lifetime T of some component, and assume that $\bar{F}(t) > 0$ for all $t \geq 0$. Then we have:

$$\bar{F}(t) = e^{-R(t)}, \quad t \geq 0,$$

where $R(t) = -\ln(\bar{F}(t))$. The function R is called the cumulative failure rate function of T.

The derivative of $R(t)$, denoted $r(t)$, is called the failure rate function of T and can be expressed as follows:

$$r(t) = \frac{\partial}{\partial t}(-\ln(\bar{F}(t))) = \frac{f(t)}{\bar{F}(t)}, \quad t \geq 0.$$

NOTE: $r(t)dt$ is the conditional probability that the component fails in the interval $(t, t + dt]$, given that the component survived the interval $[0, t]$.



Failure rates (cont.)

Note that the survival function can be expressed in terms of the failure rate function as follows:

$$\bar{F}(t) = e^{-\int_0^t r(u)du}, \quad t \geq 0, \tag{6}$$

Thus, the lifetime distribution is uniquely determined by its failure rate function.

If r is increasing, we say that the lifetime has an increasing failure rate.

If r is decreasing, we say that the lifetime has a decreasing failure rate.

- Increasing failure rates are used to describe aging. As the component gets older, it becomes more prone to failures.
- Decreasing failure rates are used to describe infant mortality. As the component gets older, early failures are eliminated or corrected.



The Weibull distribution



The Weibull distribution

A commonly used lifetime distribution in reliability is the *Weibull distribution*, named after the Swedish mathematician Waloddi Weibull, who introduced the distribution in 1951.

The probability density of a Weibull distributed variable T is given by

$$f(t) = \frac{\alpha}{\beta}\left(\frac{t}{\beta}\right)^{\alpha-1} e^{-(t/\beta)^{\alpha}}, \quad t \geq 0. \tag{7}$$

The parameter $\alpha > 0$ is called the shape parameter, while the $\beta > 0$ is called the scale parameter.



The Weibull distribution (cont.)

The cumulative distribution function:

$$F(t) = P(T \leq t) = \int_0^t f(u)du = 1 - e^{-(t/\beta)^\alpha}, \quad t \geq 0,$$

The survival function:

$$\bar{F}(t) = P(T > t) = 1 - F(t) = e^{-(t/\beta)^\alpha}, \quad t \geq 0.$$

The cumulative failure rate:

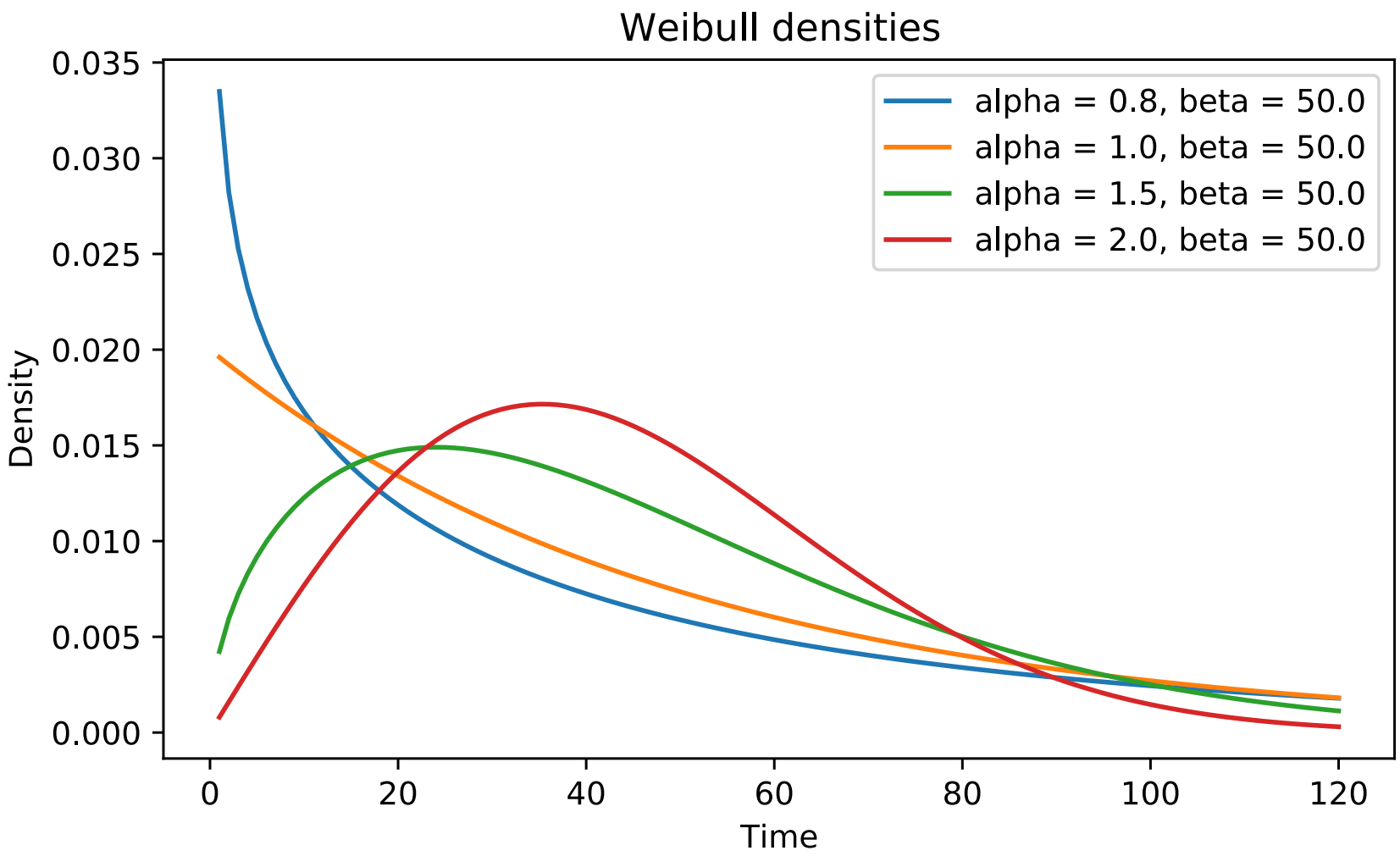
$$R(t) = -\ln(\bar{F}(t)) = \left(\frac{t}{\beta}\right)^\alpha, \quad t \geq 0,$$

The failure rate:

$$r(t) = \frac{f(t)}{\bar{F}(t)} = \frac{\alpha}{\beta}\left(\frac{t}{\beta}\right)^{\alpha-1}$$

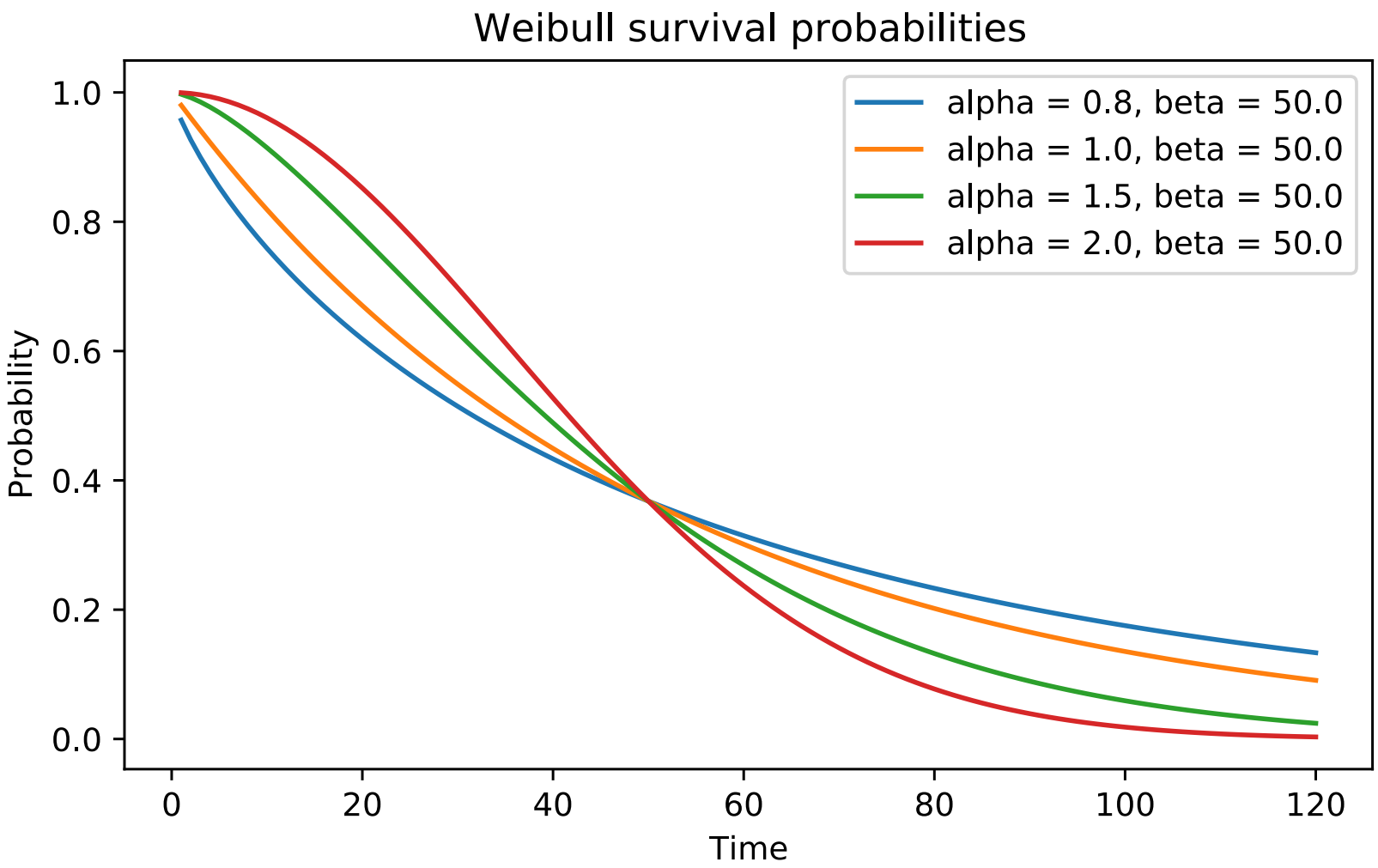


The Weibull distribution – Density





The Weibull distribution – Survival probability





The Weibull distribution – Survival probability (cont.)

NOTE: In the survival probability plot all four distributions have the same scale parameter, $\beta = 50$.

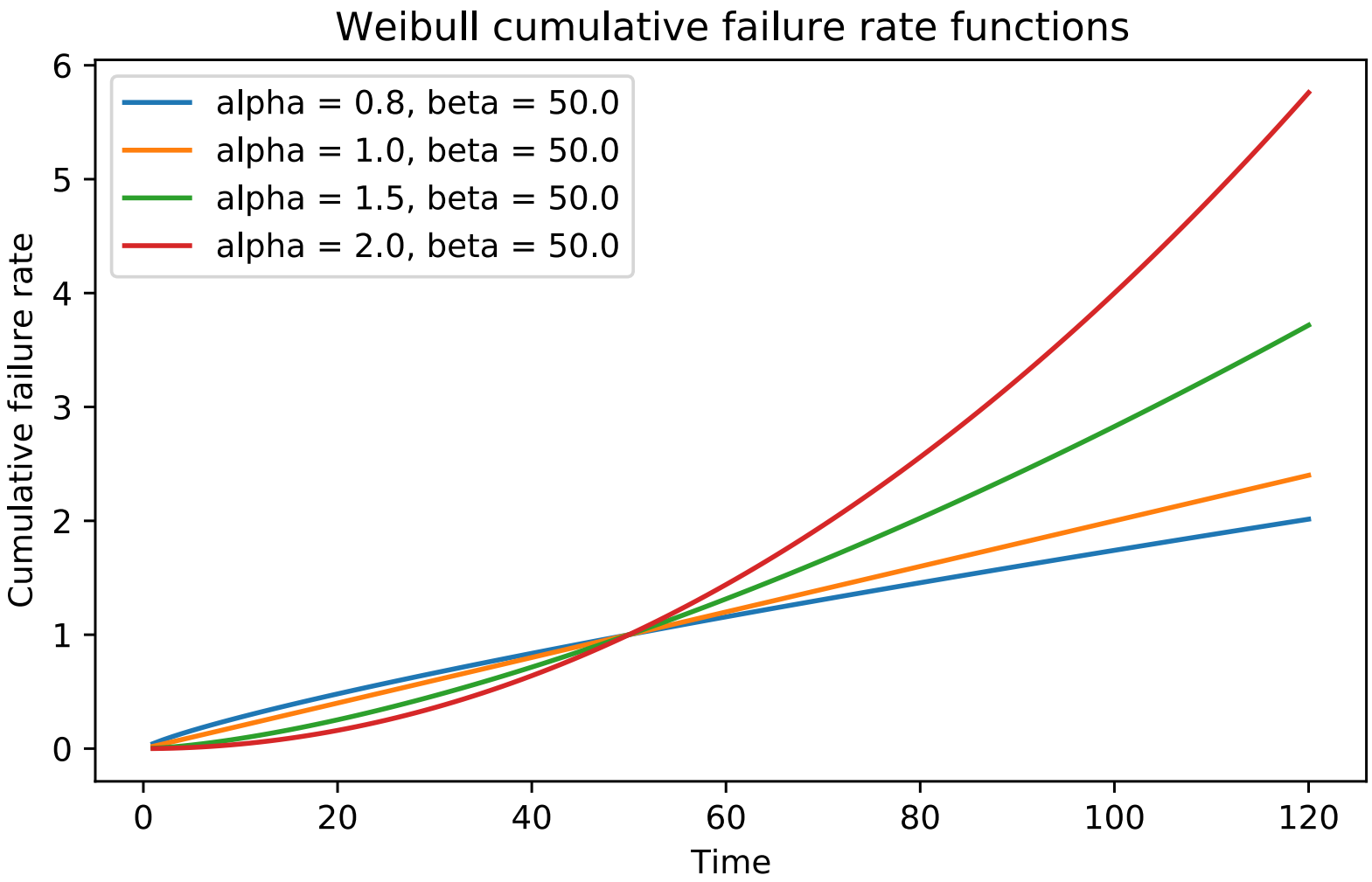
Since $\bar{F}(t) = e^{-(t/\beta)^{\alpha}}$, we have:

$$\bar{F}(\beta) = e^{-(\beta/\beta)^{\alpha}} = e^{-1}$$

Hence, all four survival functions go through the point (β, e^{-1}).



The Weibull distribution – Cumulative failure rate





The Weibull distribution – Cumulative failure rate (cont.)

NOTE: In the cumulative failure rate plot all four distributions have the same scale parameter, $\beta = 50$.

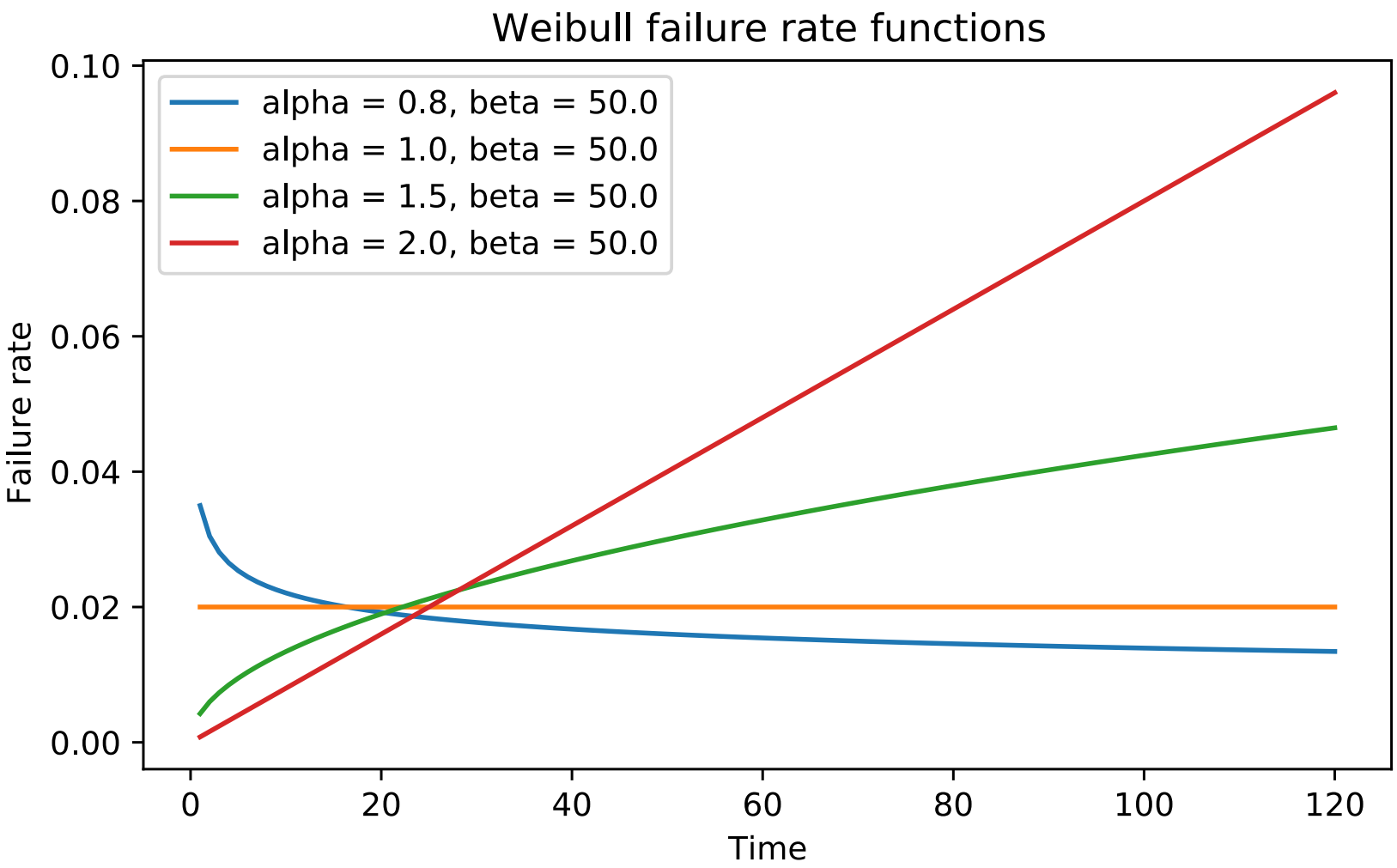
Since $R(t) = \left(\frac{t}{\beta}\right)^{\alpha}$, we have:

$$R(\beta) = \left(\frac{\beta}{\beta}\right)^{\alpha} = 1$$

Hence, all four cumulative failure rate functions go through the point $(\beta, 1)$.



The Weibull distribution – Failure rate





The Weibull distribution (cont.)

The failure rate is decreasing when $0 < \alpha < 1$, and increasing when $\alpha > 1$.

When $\alpha = 1$, the distribution is an exponential distribution, and we have:

$$\text{Density function:} \quad f(t) = \frac{1}{\beta} e^{-t/\beta}, \quad t \geq 0$$

$$\text{Cum. failure rate:} \quad R(t) = -\ln(\bar{F}(t)) = \frac{t}{\beta}, \quad t \geq 0$$

$$\text{Failure rate:} \quad r(t) = \frac{f(t)}{\bar{F}(t)} = \frac{1}{\beta}$$



The Weibull distribution – Mean and median

If T is Weibull distributed with parameters $\alpha > 0$ and $\beta > 0$, then we have:

$$E[T] = \beta \cdot \Gamma(\frac{1}{\alpha} + 1),$$

$$Med[T] = \beta \cdot [\ln(2)]^{1/\alpha}$$

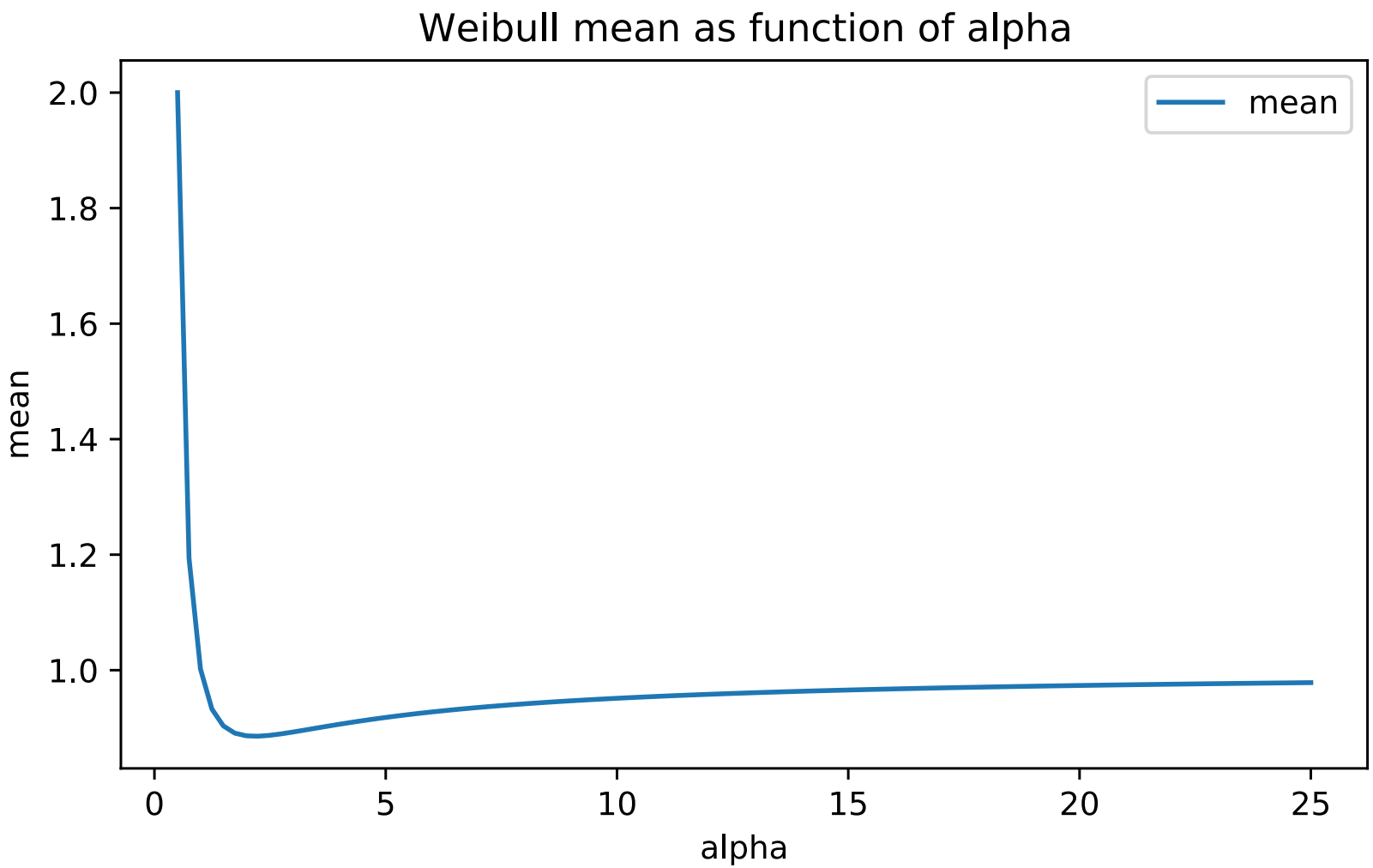
NOTE:

$$\lim_{\alpha \to 0} E[T] = \infty, \quad \lim_{\alpha \to \infty} E[T] = \beta$$

$$\lim_{\alpha \to 0} Med[T] = 0, \quad \lim_{\alpha \to \infty} Med[T] = \beta$$

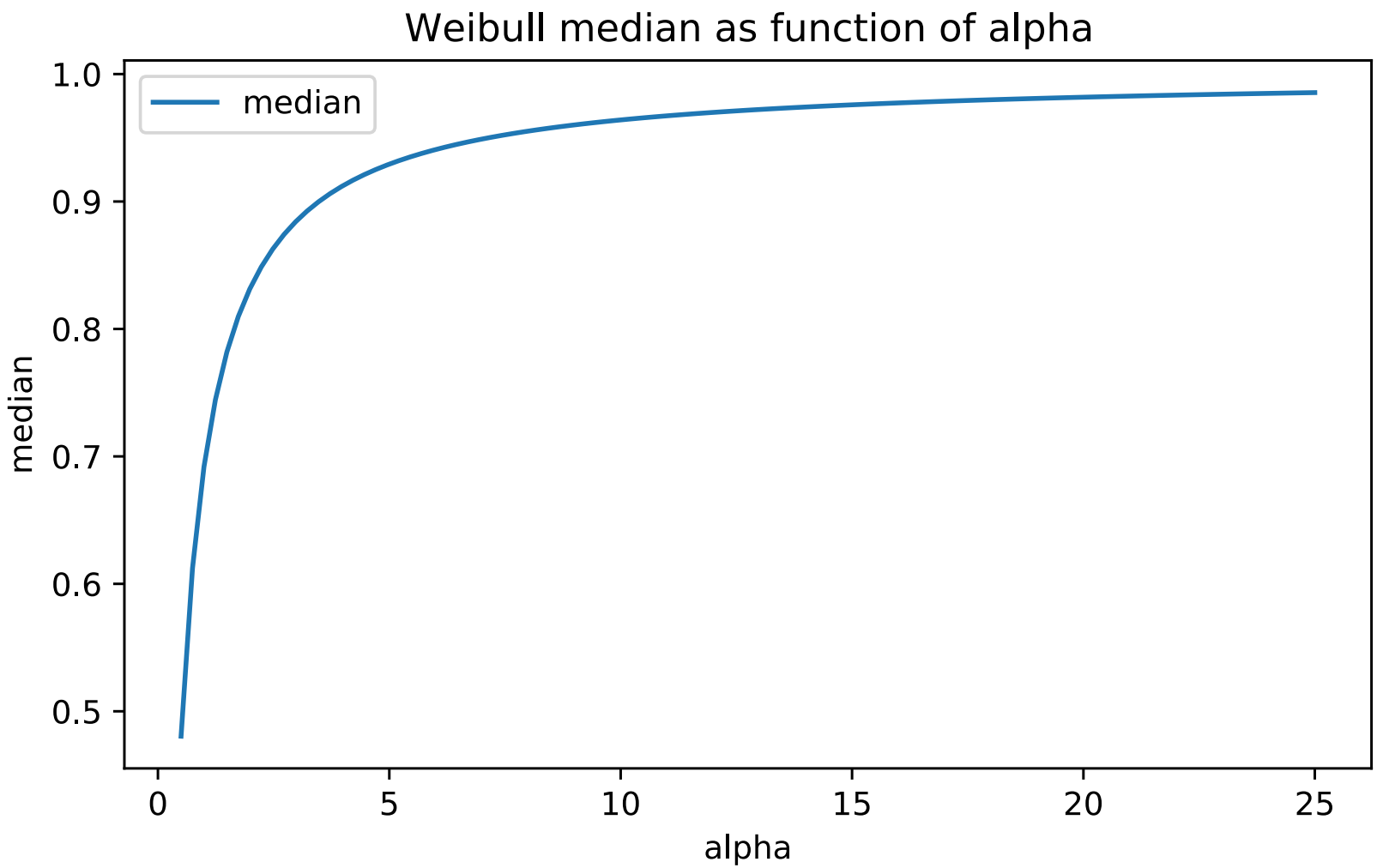


The Weibull distribution – Mean, $\beta = 1$





The Weibull distribution – Median, $\beta = 1$





The Weibull distribution – Properties

Theorem

a) If T is a Weibull distributed random variable with parameters α and β, and $K > 0$ is a constant, then $U = KT$ is Weibull distributed with parameters α and $K\beta$.

b) If $T_1, \ldots, T_n$ are independent random variables, and T_i is Weilbull distributed with parameters α and β_i, $i = 1, \ldots, n$, then $\min(T_1, \ldots, T_n)$ is Weilbull distributed with parameters α and $\gamma = (\sum_{i=1}^{n} \beta_i^{-\alpha})^{-1/\alpha}$.

c) If $T_1, \ldots, T_n$ are independent and identically Weibull distributed random variables with parameters α and β, then $n^{1/\alpha} \min(T_1, \ldots, T_n)$ is Weilbull distributed with parameters α and β.



The Weibull distribution – Properties(cont.)

PROOF (a) Assume that T is Weibull distributed with parameters α and β, and that $K > 0$ is a constant.

The survival function of $U = KT$ is given by:

$$P(U > u) = P(KT > u) = P(T > u/K) = e^{-[(u/K)/\beta]^{\alpha}} = e^{-[u/(K\beta)]^{\alpha}}$$

This is the survival function of a Weibull distributed variable with parameters α and $K\beta$.

Hence, we conclude that $U = KT$ is Weibull distributed with parameters α and $K\beta$.



The Weibull distribution (cont.)

PROOF (b) Assume that $T_1, \ldots, T_n$ are independent random variables where T_i is Weibull distributed with parameters α and β_i, $i = 1, \ldots, n$. We then let:

$$T = \min(T_1, \ldots, T_n)$$

The survival function of T is then given by:

$$P(T > t) = P(T_1 > t \cap \cdots \cap T_n > t) = \prod_{i=1}^{n} P(T_i > t)$$

$$= \prod_{i=1}^{n} e^{-(t/\beta_i)^\alpha} = \exp(-\sum_{i=1}^{n} (t/\beta_i)^\alpha)$$

$$= \exp(-t^\alpha \sum_{i=1}^{n} \beta_i^{-\alpha}) = e^{-t^\alpha \gamma^{-\alpha}} = e^{-(t/\gamma)^\alpha}$$

where $\gamma^{-\alpha} = \sum_{i=1}^{n} \beta_i^{-\alpha}$. Hence, T is Weibull with parameters α and γ.



The Weibull distribution (cont.)

PROOF (c) Assume that $T_1, \ldots, T_n$ are independent random variables where T_i is Weibull distributed with parameters α and β, $i = 1, \ldots, n$, and let:

$$T = \min(T_1, \ldots, T_n)$$

By (b) it follows that T is Weibull distributed with parameters α and γ, where:

$$\gamma = (\sum_{i=1}^{n} \beta^{-\alpha})^{-1/\alpha} = (n\beta^{-\alpha})^{-1/\alpha} = n^{-1/\alpha}\beta$$

By (a) it follows that $n^{1/\alpha}T$ is Weibull distributed with parameters α and $n^{1/\alpha}\gamma = \beta$.

NOTE: This holds for any n, and thus, also when $n \to \infty$. This is linked to the fact that the Weibull distribution is an *extreme value distribution*.



Plotting system reliability as a function of time



Plotting system reliability as a function of time

We first assume that the reliability function, $h(\boldsymbol{p}(t))$ is known.

To plot $h(\boldsymbol{p}(t))$ as a function of the time $t \geq 0$, we choose a sufficiently dense set of t-values, $0 = t_0 < t_1 < \cdots < t_K$, and calculate $h(\boldsymbol{p}(t_j))$, for these t-values.

The reliability curve for $t \in [0, t_K]$ is then obtained by drawing a curve through the points:

$$(t_0, h(\boldsymbol{p}(t_0))), (t_1, h(\boldsymbol{p}(t_1))), \ldots, (t_K, h(\boldsymbol{p}(t_K))).$$

If K is sufficiently large, a nice smooth curve is obtained.



Example – 2-out-of-3 system

Let (C, ϕ) be a 2-out-of-3 system with $C = \{1, 2, 3\}$ and structure function ϕ is given by:

$$\phi(X_1, X_2, X_3) = X_1X_2 + X_1X_3 + X_2X_3 - 2X_1X_2X_3$$

From this it follows that the reliability of the system at time $t \geq 0$ is:

$$h(\boldsymbol{p}(t)) = p_1(t)p_2(t) + p_1(t)p_3(t) + p_2(t)p_3(t) - 2p_1(t)p_2(t)p_3(t)$$

By inserting $p_i(t) = \bar{F}_i(t)$, $i = 1, 2, 3$ into the expression for $h(\boldsymbol{p}(t))$, we can calculate the system reliability for any $t \geq 0$.

We assume that the lifetime of component i, T_i is Weibull-distributed with parameters α_i and β_i, for all $i \in C$, where:

$$\alpha_1 = 2.0, \quad \alpha_2 = 2.5, \quad \alpha_3 = 3.0,$$

$$\beta_1 = 50.0, \quad \beta_2 = 60.0, \quad \beta_3 = 70.0.$$



Example – 2-out-of-3 system (cont.)

```
from math import *
import matplotlib.pyplot as plt
import numpy as np

save_plots = True

# Time parameters
max_t = 120
steps = 120

# Weibull-parameters for the components
alpha = [2.0, 2.5, 3.0]
beta = [50.0, 60.0, 70.0]

# Component reliability vector
num_comps = 3
p = np.zeros(num_comps)

# System reliabilities
h = np.zeros(steps)


def reliability(pp):
    return pp[0] * pp[1] + pp[0] * pp[2] + pp[1] * pp[2] - 2 * pp[0] * pp[1] * pp[2]

# The survival function of a Weibull-distribution
def weibull(aa, bb, t):
    if t >= 0:
        return exp(- (t / bb) ** aa)
    else:
        return 1.0
```



Example – 2-out-of-3 system (cont.)

```
# The survival function of a Weibull-distribution
def weibull(aa, bb, t):
    if t >= 0:
        return exp(- (t / bb) ** aa)
    else:
        return 1.0

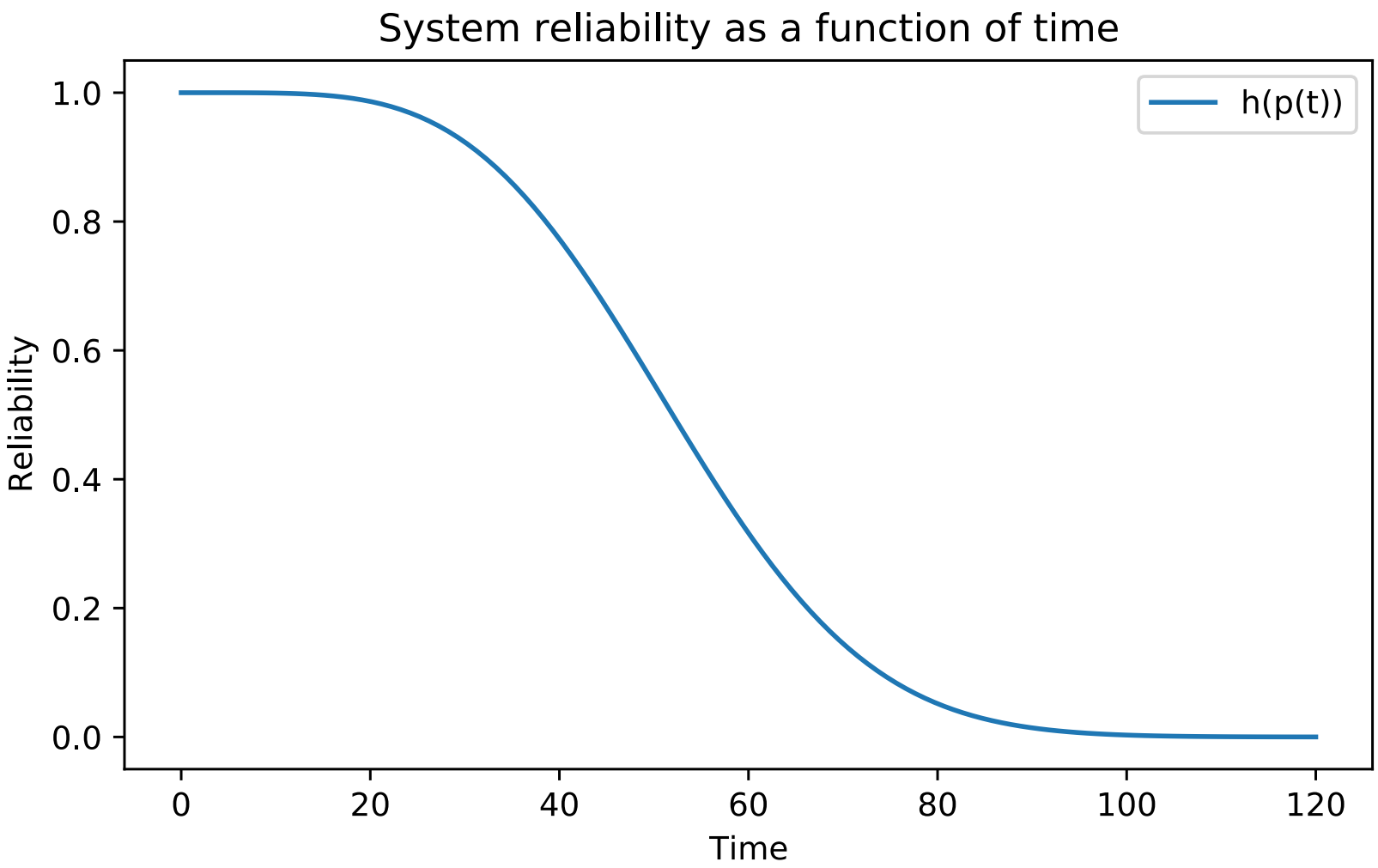

time = np.linspace(0.0, max_t, steps)

for step in range(steps):
    for i in range(num_comps):
        p[i] = weibull(alpha[i], beta[i], time[step])
    h[step] = reliability(p)

fig = plt.figure(figsize = (7, 4))
plt.plot(time, h, label='h(p(t))')
plt.xlabel('Time')
plt.ylabel('Reliability')
plt.title("System reliability as a function of time")
plt.legend()
if save_plots:
    plt.savefig("dynsys_01/reliability_01.pdf")
plt.show()
```



Example – 2-out-of-3 system (cont.)





MC simulation using minimal path sets

Given the minimal path sets $P_1, \ldots, P_p$ and the minimal cut sets $K_1, \ldots, K_k$ of the system, the relation between the component lifetimes, $T_1, \ldots, T_n$ and the system lifetime, S, can be expressed as:

$$S = \min_{1 \leq j \leq k} \max_{i \in K_j} T_i = \max_{1 \leq j \leq p} \min_{i \in P_j} T_i.$$

This formula is useful when the system reliability is estimated using Monte Carlo simulation.

Consider a binary monotone system (C, ϕ), where $C = \{1, \ldots, n\}$, and where $P_i, \ldots, P_p$ are the minimal path sets of the system.

We generate N samples of component lifetimes $\{T_{1k}, \ldots, T_{nk}\}$, where T_{ik} denotes the lifetime of the ith component in the kth simulation, $k = 1, \ldots, N$.



MC simulation using minimal path sets (cont.)

We then calculate the resulting system lifetimes $S_1, \ldots, S_N$ given by:

$$S_k = \max_{1 \leq j \leq p} \min_{i \in P_j} T_{ik}, \quad k = 1, \ldots, N.$$

Based on these lifetimes we can estimate the system reliability as a function of the time $t \geq 0$ using the following estimator:

$$\hat{h}(t) = \frac{1}{N} \sum_{k=1}^{N} I(S_k > t)$$

It follows that:

$$E[\hat{h}(t)] = \frac{1}{N} \sum_{k=1}^{N} P(S_k > t) = \bar{G}(t) = h(\boldsymbol{p}(t))$$

Thus, $\hat{h}(t)$ is an unbiased estimator for the system reliability at time $t \geq 0$.



MC simulation using minimal path sets (cont.)

In order to calculate $\hat{h}(t)$ more efficiently for all $t \geq 0$, we sort the system lifetimes in ascending order:

$$S_{(1)} < \cdots < S_{(N)},$$

assuming for simplicity that all the lifetimes are distinct. We also define $S_{(0)} = 0$, and note that:

$$\hat{h}(S_{(m)}) = \frac{1}{N} \sum_{k=1}^{N} I(S_k > S_{(m)}) = \frac{N-m}{N}, \quad m = 0, 1, \ldots, N.$$

Hence, the function $\hat{h}(t)$ passes through the following $N+1$ points:

$$(S_{(0)}, \frac{N-0}{N}), (S_{(1)}, \frac{N-1}{N}), \ldots, (S_{(N)}, \frac{N-N}{N}).$$

For sufficiently large values of N a nice smooth estimate of the function $h(\boldsymbol{p}(t))$ can then be obtained by drawing a curve through these points.



MC simulation using minimal path sets (cont.)

EXAMPLE: Let (C, ϕ) be a bridge system with $C = \{1, 2, 3, 4, 5\}$ and minimal path sets:

$$P_1 = \{1, 4\}, \quad P_2 = \{1, 3, 5\}, \quad P_3 = \{2, 3, 4\}, \quad P_4 = \{2, 5\}.$$

The lifetime of component i, T_i is Weibull-distributed with parameters α_i and β_i, for all $i \in C$ where:

$$\alpha_1 = 2.0, \quad \alpha_2 = 2.5, \quad \alpha_3 = 3.0, \quad \alpha_4 = 1.0, \quad \alpha_5 = 1.5,$$
$$\beta_1 = 50.0, \quad \beta_2 = 60.0, \quad \beta_3 = 70.0, \quad \beta_4 = 40.0, \quad \beta_5 = 50.0.$$

The reliability $h(\boldsymbol{p}(t))$ is estimated for $t \geq 0$, by running a Monte Carlo simulation with $N = 100000$ simulations.



MC simulation using minimal path sets (cont.)

```
from math import *
from random import *
import matplotlib.pyplot as plt
import numpy as np

save_plots = True

# Number of simulations
num_sims = 100000

# Weibull-parameters for the components
alpha = [2.0, 2.5, 3.0, 1.0, 1.5]
beta = [50.0, 60.0, 70.0, 40.0, 50.0]

# Number of components and minimal paths
num_comps = 5
num_paths = 4


# Minimal path sets
paths = [{1,4}, {1,3,5}, {2,3,4}, {2,5}]


# Calculate the system lifetime given the component lifetimes
def sys_lifetime(tt):
    sys_life = 0.0
    for j in range(num_paths):
        path_life = np.inf
        for i in paths[j]:
            if tt[i-1] < path_life:
                path_life = tt[i-1]
        if path_life > sys_life:
            sys_life = path_life
    return sys_life
```



MC simulation using minimal path sets (cont.)

```
# The upper percentile levels
q = np.zeros(num_sims)

# The simulated system lifetimes
s = np.zeros(num_sims)

# The simulated lifetimes of the components
t = np.zeros(num_comps)

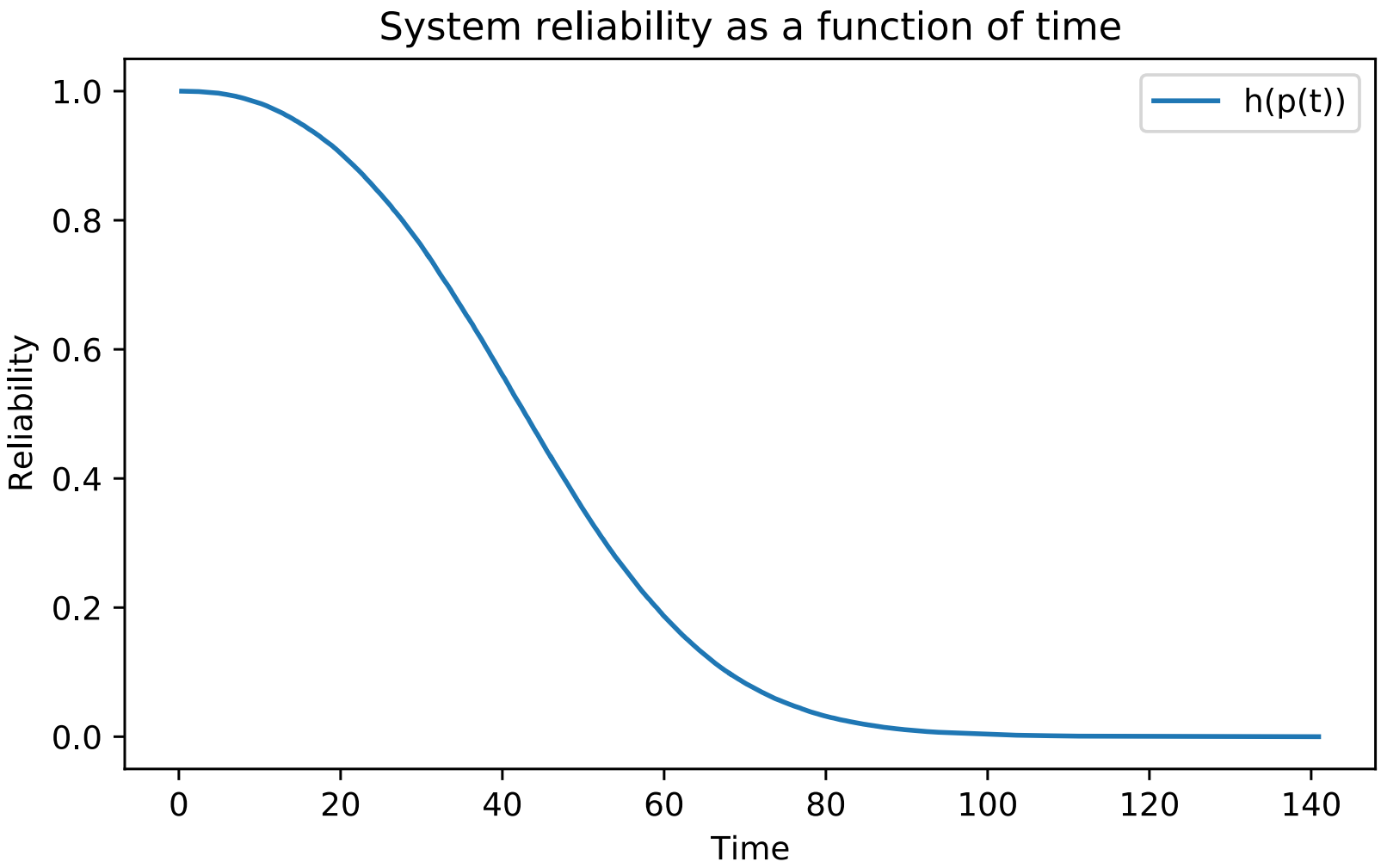
for k in range(num_sims):
    q[k] = 1.0 - k / num_sims
    for i in range(num_comps):
        t[i] = weibullvariate(beta[i], alpha[i])
    s[k] = sys_lifetime(t)


s.sort()

fig = plt.figure(figsize = (7, 4))
plt.plot(s, q, label='h(p(t))')
plt.xlabel('Time')
plt.ylabel('Reliability')
plt.title("System reliability as a function of time")
plt.legend()
if save_plots:
    plt.savefig("dynsys_02/reliability_02.pdf")
plt.show()
```



MC simulation using minimal path sets (cont.)





MC simulation using sorted component lifetimes

It is possible to estimate the reliability function without knowing the minimal path sets provided that we can calculate the structure function efficiently.

IDEA: Use the fact that the lifetime of the system always coincides with the lifetime of one of the components.

Consider a binary monotone system (C, ϕ), where $C = \{1, \ldots, n\}$ where the lifetime of the ith component is T_i, $i \in C$.

By sorting the lifetimes the order at which the components fail is identified, assuming that all the component lifetimes are distinct.

If $T_{i_1} < \cdots < T_{i_n}$ are the sorted lifetimes, the components fail in the order $i_1, \ldots, i_n$.

For convenience we also define $T_{i_0} = 0$.



MC simulation using sorted component lifetimes (c.)

Let $\boldsymbol{X}_j$ denote the component state vector after the jth failure, $j = 1, \ldots, n$, and let $\boldsymbol{X}_0 = \mathbf{1}$. Finally, let:

$$j^* = \min\{j : \phi(\boldsymbol{X}_j) = 0\}$$

Then the lifetime of the system, S, is given by:

$$S = T_{i_{j^*}}$$

In the trivial case where $\phi(\boldsymbol{X}_0) = \phi(\mathbf{1}) = 0$, we get that $j^* = 0$. Thus, $S = T_{i_0} = 0$ in such cases.

In the trivial case where $\phi(\boldsymbol{X}_n) = \phi(\mathbf{0}) = 1$, j^* becomes undefined. To avoid this issue we simply define S to be ∞ in such cases.

Having found a method for calculating the system lifetime, we proceed as before and obtain an estimate of the system reliability as a function of time.



MC simulation using sorted component lifetimes (c.)

EXAMPLE: Let (C, ϕ) be a threshold system with component set $C = \{1, 2, 3, 4, 5\}$ with structure function:

$$\phi(\boldsymbol{X}) = \mathrm{I}(\sum_{i=1}^{5} \omega_i X_i \geq 5.0)$$

where the component weights $\omega_1, \ldots, \omega_5$ are:

$$\omega_1 = 2.0, \quad \omega_2 = 4.0, \quad \omega_3 = 2.3, \quad \omega_4 = 3.5, \quad \omega_5 = 1.9$$

The lifetime of component i, T_i is Weibull-distributed with parameters α_i and β_i, for all $i \in C$ where:

$$\alpha_1 = 2.0, \quad \alpha_2 = 2.5, \quad \alpha_3 = 3.0, \quad \alpha_4 = 1.0, \quad \alpha_5 = 1.5,$$
$$\beta_1 = 50.0, \quad \beta_2 = 60.0, \quad \beta_3 = 70.0, \quad \beta_4 = 40.0, \quad \beta_5 = 50.0.$$

The reliability $h(\boldsymbol{p}(t))$ is estimated for $t \geq 0$, by running a Monte Carlo simulation with $N = 100000$ simulations.



MC simulation using sorted component lifetimes (c.)

```
from math import *
from random import *
import matplotlib.pyplot as plt
import numpy as np

save_plots = True

# Number of simulations
num_sims = 100000

# Weibull-parameters for the components
alpha = [2.0, 2.5, 3.0, 1.0, 1.5]
beta = [50.0, 60.0, 70.0, 40.0, 50.0]

# Threshold system weights and threshold
w = [2.0, 4.0, 2.3, 3.5, 1.9]
threshold = 5.0

# Number of components
num_comps = 5

# Component state vector
x = np.zeros(num_comps)

# The structure function of the threshold system
def phi(xx):
    wsum = 0.0
    for i in range(num_comps):
        wsum += w[i] * xx[i]
    if wsum >= threshold:
        return 1
    else:
        return 0
```



MC simulation using sorted component lifetimes (c.)

```
# Calculate the system lifetime given the component lifetimes
def sys_lifetime(tt):
    # Initialize component states
    for i in range(num_comps):
        x[i] = 1
    # Initialize system state and lifetime
    sys_state = phi(x)
    sys_life = 0.0
    # Determine the ordering of the failure times
    order = tt.argsort()
    # Initialize failure counter
    c = 0
    # Switch off one component at a time in the order of the failure times
    while (sys_state == 1) and (c < num_comps):
        x[order[c]] = 0
        sys_life = tt[order[c]]
        sys_state = phi(x)
        c += 1
    if sys_state == 0:
        return sys_life
    else: # This happens only for trivial systems where phi(0,...,0) = 1.
        return np.inf
```



MC simulation using sorted component lifetimes (c.)

```
# The upper percentile levels
q = np.zeros(num_sims)

# The simulated system lifetimes
s = np.zeros(num_sims)

# The simulated lifetimes of the components
t = np.zeros(num_comps)

for k in range(num_sims):
    q[k] = 1.0 - k / num_sims
    for i in range(num_comps):
        t[i] = weibullvariate(beta[i], alpha[i])
    s[k] = sys_lifetime(t)

s.sort()

fig = plt.figure(figsize = (7, 4))
plt.plot(s, q, label='h(p(t))')
plt.xlabel('Time')
plt.ylabel('Reliability')
plt.title("System reliability as a function of time")
plt.legend()
if save_plots:
    plt.savefig("dynsys_03/reliability_03.pdf")
plt.show()
```



Exact reliability function using ROBDD

Instead of using MC simulations, we could use an ROBDD-representation of the system, and then calculate the reliability as a function of time, $h(\boldsymbol{p}(t))$ analytically.

RECALL: By using an ROBDD-representation we can calculate the reliability function $h(\boldsymbol{p})$ analytically for any $\boldsymbol{p}$.

Thus, we choose a sufficiently dense set of t-values, $0 = t_0 < t_1 < \cdots < t_K$, and calculate $h(\boldsymbol{p}(t_j))$, for these t-values.

The reliability curve for $t \in [0, t_K]$ is then obtained by drawing a curve through the points:

$$(t_0, h(\boldsymbol{p}(t_0))), (t_1, h(\boldsymbol{p}(t_1))), \ldots, (t_K, h(\boldsymbol{p}(t_K))).$$

If K is sufficiently large, a nice smooth curve is obtained.



Exact reliability function using ROBDD (cont.)

```
from math import exp
import matplotlib.pyplot as plt
import numpy as np
from c_robdd import ROBDDSystem
from c_threshold import ROBDDThreshold

save_plots = True

# Time parameters
max_t = 175
steps = 175

# Weibull-parameters for the components
alpha = [2.0, 2.5, 3.0, 1.0, 1.5]
beta = [50.0, 60.0, 70.0, 40.0, 50.0]

# Number of components in the system
num_comps = len(alpha)


# Threshold system weights and threshold
a = [2.0, 4.0, 2.3, 3.5, 1.9]
b = 5.0

# Calculate the weight sum
asum = 0
for i in range(num_comps):
    asum += a[i]

# The ROBDD representation of the threshold system
sys = ROBDDSystem(ROBDDThreshold(a, asum, b))

# The reliability function of the system
def reliability(pp):
    result = sys.calculateReliability(pp)
    return result[1]
```



Exact reliability function using ROBDD (cont.)

```
# The survival function of a Weibull-distribution
def weibull(aa, bb, t):
    if t >= 0:
        return exp(- (t / bb) ** aa)
    else:
        return 1.0

# Component reliability vector
p = np.zeros(num_comps)

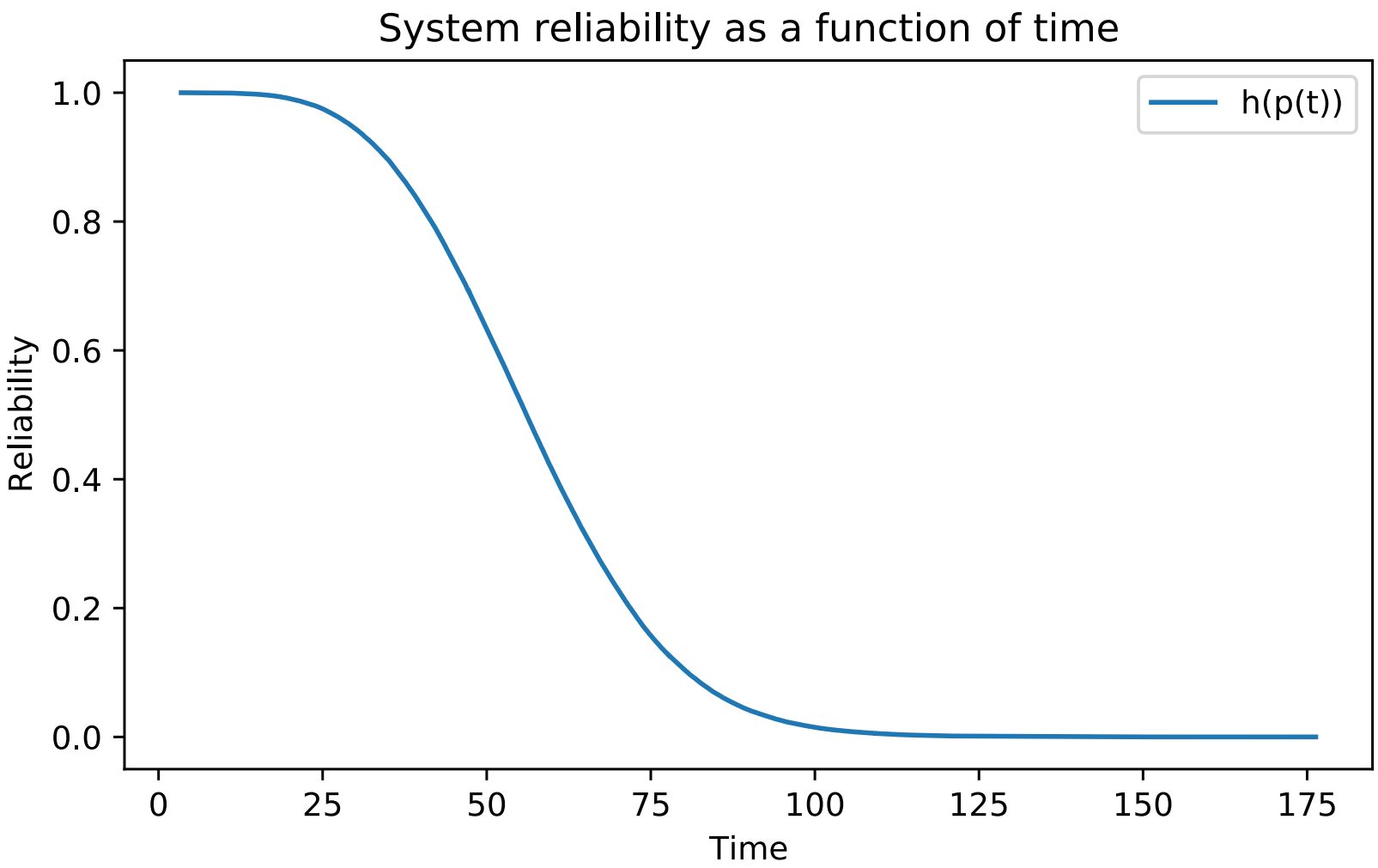
# Time values and Reliability function
time = np.linspace(0.0, max_t, steps)
h = np.zeros(steps)

# Calculate the Reliability function
for step in range(steps):
    for i in range(num_comps):
        p[i] = weibull(alpha[i], beta[i], time[step])
    h[step] = reliability(p)

# Plot the results
fig = plt.figure(figsize = (7, 4))
plt.plot(time, h, label='h(p(t))')
plt.xlabel('Time')
plt.ylabel('Reliability')
plt.title("System reliability as a function of time")
plt.legend()
if save_plots:
    plt.savefig("dynsys_03b/reliability_03b.pdf")
plt.show()
```

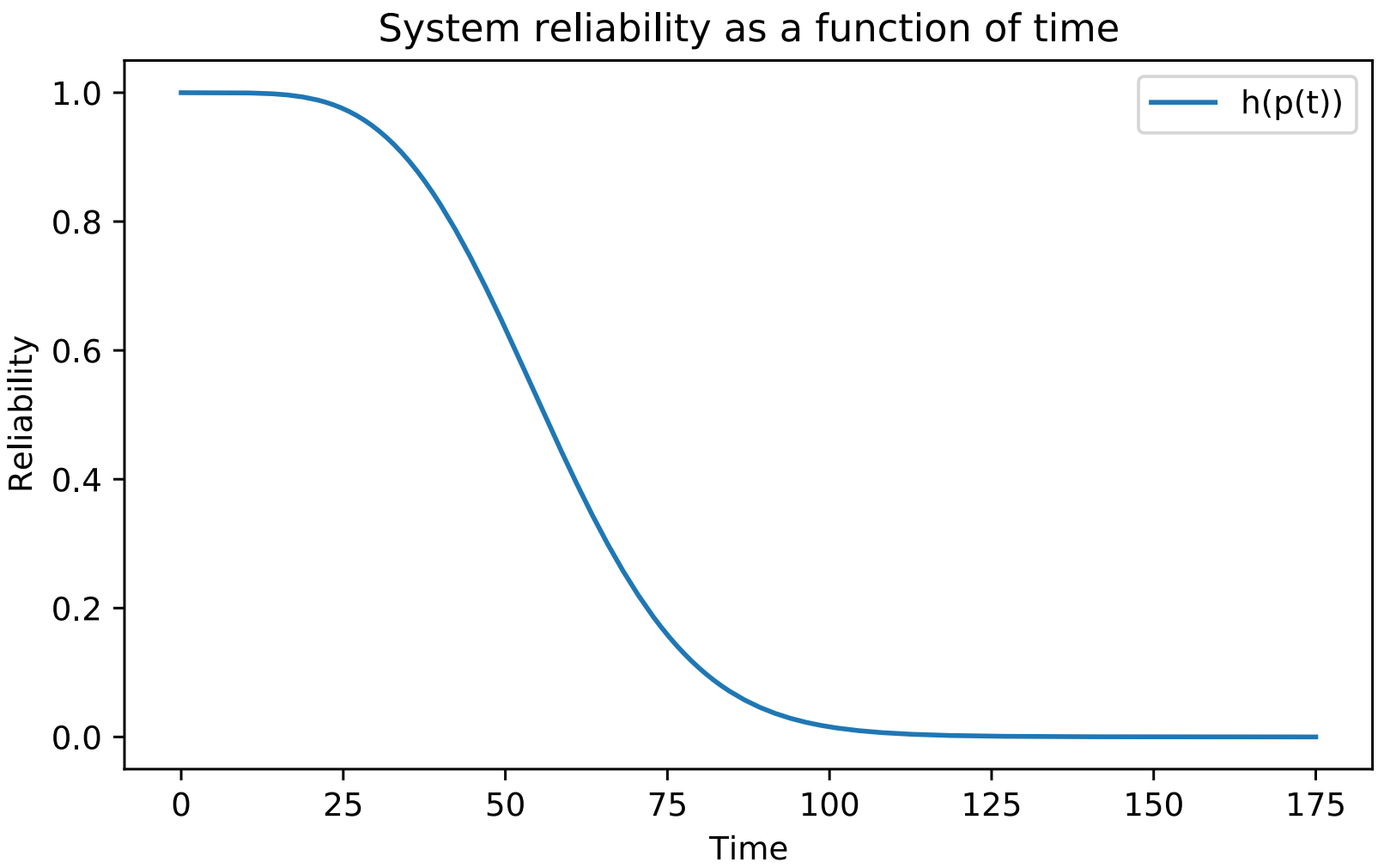


The estimated reliability function



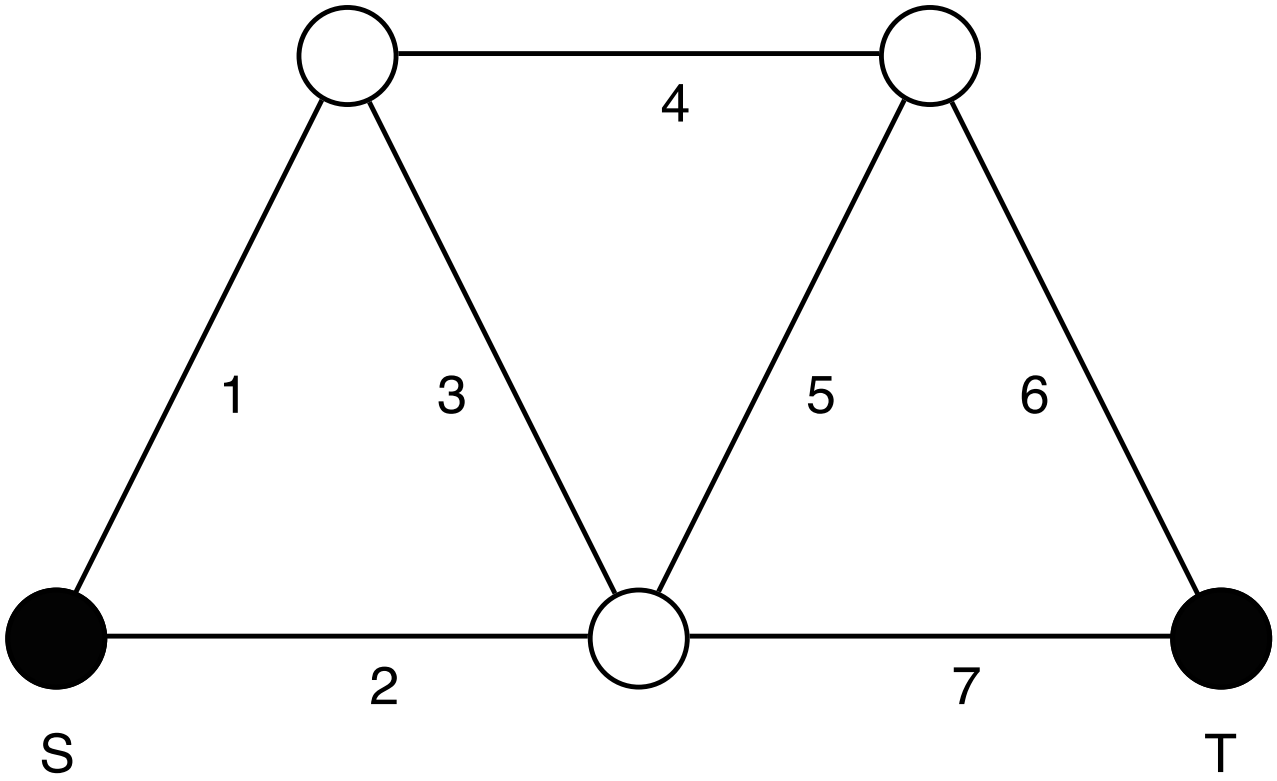


The exact reliability function





MC simulation using sorted component lifetimes (c.)





MC simulation using sorted component lifetimes (c.)

EXAMPLE: Let (C, ϕ) be an undirected network system with component set $C = \{1, \ldots, 7\}$. The structure function of this system is given by:

$$\phi(\boldsymbol{X}) = X_4 \cdot \phi(1_4, \boldsymbol{X}) + (1 - X_4) \cdot \phi(0_4, \boldsymbol{X}),$$

where:

$$\begin{aligned}\phi(1_4, \boldsymbol{X}) &= (X_3 \amalg X_5) \cdot (X_1 \amalg X_2) \cdot (X_6 \amalg X_7) \\ &\quad + (1 - X_3 \amalg X_5) \cdot ((X_1 X_6) \amalg (X_2 X_7))\end{aligned}$$

$$\phi(0_4, \boldsymbol{X}) = ((X_1 X_3) \amalg X_2) \cdot ((X_5 X_6) \amalg X_7)$$



MC simulation using sorted component lifetimes (c.)

The lifetime of component i, T_i is Weibull-distributed with parameters α_i and β_i, for all $i \in C$ where:

$$\alpha_1 = 2.0, \quad \alpha_2 = 2.5, \quad \alpha_3 = 3.0, \quad \alpha_4 = 1.0,$$
$$\alpha_5 = 1.5, \quad \alpha_6 = 2.0, \quad \alpha_7 = 2.5,$$

$$\beta_1 = 50.0, \quad \beta_2 = 60.0, \quad \beta_3 = 70.0, \quad \beta_4 = 40.0,$$
$$\beta_5 = 50.0, \quad \beta_6 = 55.0, \quad \beta_7 = 65.0.$$

The reliability $h(\boldsymbol{p}(t))$ for $t \geq 0$ is estimated by running a Monte Carlo simulation with $N = 100000$ simulations.



MC simulation using sorted component lifetimes (c.)

```
from math import *
from random import *
import matplotlib.pyplot as plt
import numpy as np

save_plots = True

# Number of simulations
num_sims = 100000

# Weibull-parameters for the components
alpha = [2.0, 2.5, 3.0, 1.0, 1.5, 2.0, 2.5]
beta = [50.0, 60.0, 70.0, 40.0, 50.0, 55.0, 65.0]

# Number of components
num_comps = 7

# Component state vector
x = np.zeros(num_comps)

# The coproduct function
def coprod(x1, x2):
    return x1 + x2 - x1 * x2

# The structure function given that component 3 is functioning
def phi_3_1(xx):
    return coprod(xx[2], xx[4]) * coprod(xx[0], xx[1]) * coprod(xx[5], xx[6]) \
        + (1 - coprod(xx[2], xx[4])) * coprod(xx[0] * xx[5], xx[1] * xx[6])

# The structure function given that component 3 is failed
def phi_3_0(xx):
    return coprod(xx[0] * xx[2], xx[1]) * coprod(xx[4] * xx[5], xx[6])

# The structure function of the network system (using pivotal decomposition)
def phi(xx):
    return xx[3] * phi_3_1(xx) + (1 - xx[3]) * phi_3_0(xx)
```



MC simulation using sorted component lifetimes (c.)

```
# Calculate the system lifetime given the component lifetimes
def sys_lifetime(tt):
    # Initialize component states
    for i in range(num_comps):
        x[i] = 1
    # Initialize system state and lifetime
    sys_state = phi(x)
    sys_life = 0.0
    # Determine the ordering of the failure times
    order = tt.argsort()
    # Initialize failure counter
    c = 0
    # Switch off one component at a time in the order of the failure times
    while (sys_state == 1) and (c < num_comps):
        x[order[c]] = 0
        sys_life = tt[order[c]]
        sys_state = phi(x)
        c += 1
    if sys_state == 0:
        return sys_life
    else: # This happens only for trivial systems where phi(0,...,0) = 1.
        return np.inf
```



MC simulation using sorted component lifetimes (c.)

```
# The upper percentile levels
q = np.zeros(num_sims)

# The simulated system lifetimes
s = np.zeros(num_sims)

# The simulated lifetimes of the components
t = np.zeros(num_comps)

for k in range(num_sims):
    q[k] = 1.0 - k / num_sims
    for i in range(num_comps):
        t[i] = weibullvariate(beta[i], alpha[i])
    s[k] = sys_lifetime(t)

s.sort()

fig = plt.figure(figsize = (7, 4))
plt.plot(s, q, label='h(p(t))')
plt.xlabel('Time')
plt.ylabel('Reliability')
plt.title("System reliability as a function of time")
plt.legend()
if save_plots:
    plt.savefig("dynsys_04/reliability_04.pdf")
plt.show()
```



MC simulation using sorted component lifetimes (c.)

