Montecarlo Project, Group 7

Simone Pagliuca, Cyrille de Lurion, Samuele Mezzaro

1 Structure of the system and global values

In our work we had to consider a 2-out-of-4 parallel scheme of identical components, equally sharing a common load. Every time a component fails, the common load is distributed on the remaining working components increasing their failure rate λ_i, with i being the number of working components. When only two components are working it is possible to repair two of the failed ones simultaneously with repair rate μ_s. If only one component is working, the system fails. Following the failure of the system, two components are repaired simultaneously and the component that is still working is reactivated. The mission time of the analysis is $T_m = 1000[h]$

$$\lambda_4 = 5.5 \cdot 10^{-3}[h^{-1}]$$

$$\lambda_3 = 2\lambda_4 = 11 \cdot 10^{-3}[h^{-1}]$$

$$\lambda_2 = 2\lambda_3 = 22 \cdot 10^{-3}[h^{-1}]$$

$$\mu_s = 2.2 \cdot 10^{-1}[h^{-1}]$$

In a second moment we had to consider an external event hitting the system with a rate $\lambda_c = 1.51 \cdot 10^{-3}[h^{-1}]$, with a probability $p = 0.4$ of causing the failure of any of the four components. We assumed that the external event could cause the failure of any number of the components, even if in stand-by phase (when the system is failed and there is only one working component).

2 Markov's Diagram

In Figure [1] is represented a scheme of the system through the method of the Markov's diagram. This scheme does not consider the external event. The states of the systems are described as follows:

- State 0: zero components are failed, initial state.
- State 1: one component is failed.
- State 2: two components are failed.
- State 3: three components are failed, the system is failed.

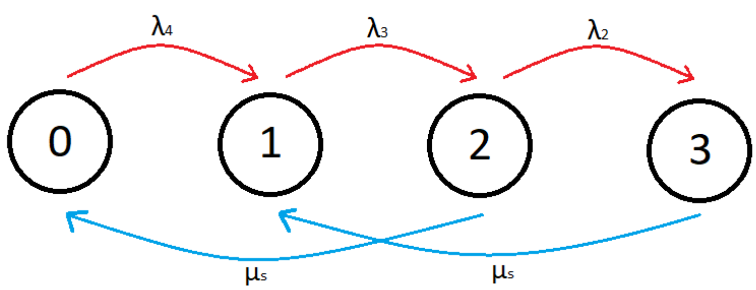

Figure 1: Markov's diagram

3 Pseudo code description

In this section, we will discuss the pseudo code utilized to solve the problem. The first step is the definition of some functions that will be used in our code. The pseudo code can be found in dedicated section.

The first function [1] computes the analytical solution for the availability and the reliability of the system, through the solution of the system of equations (A*X(0)-B), with A=transition matrix, X(0)=vector initial state of the system and B differentiation of X.

The second function [2] computes the transition matrix of the components.

The third function [3] computes the cumulative distribution of the common failure probabilities of the components due to the external event, then confronts that distribution with a random number between 0 and 1 to sample how many components have failed.

The fourth function [4] computes the cumulative distribution of the probability of transition of every component, then confronts that distribution with a random number between 0 and 1 to find which component transitioned.

The fifth function [5] computes the cumulative distribution of the probability of the different transitions of the chosen component, then confronts that distribution with a random number between 0 and 1 to find which type of transition occurred.

The sixth function [6] updates the time of the life simulation of the system.

The seventh function [7] updates the state of the system, failing or repairing the components randomly sampled.

The eighth function [8] simulates the system exploiting the functions previously described to update the counters of availability, reliability, time to failure, number of failures due to the external cause, total number of failures and number of repairs.

Using the function SimulateSystem [8] we are able in the main part of our code [9] to compute all the values requested. In fact calling the function [8] inside a For loop, we can update the values of availability, reliability, time to failure indexes, number of repairs, number of failures and common failures. Thanks to this values we are now able to print and plot the solution.

In the end the Availability and Reliability vectors are plotted over time with error bars $\pm\sigma$, obtaining: Fig.[2] and Fig.[3]

4 Results

Observing the results we have obtained for the case 1 of the simulation, without considering the common cause of failure, we can see how for both the reliability Fig.[2] and the availability Fig.[3] the Montecarlo simulation, with a total of 10000 trials reproduces the analytical solution with a good level of accuracy. On the negative side, we can observe how the simulation tends to overestimate the reliability of the system, so we do not get a conservative estimation. This could lead the system to fail before our expectations. Concerning availability, we can observe some oscillations getting wider after the analytical solution stabilizes around 0.9898 at time 140 hours, following a first rapid transient, this phenomenon is due to the Bernoulli statistics intrinsic to the problem, variance increases when the parameter's value increases. In both cases, the oscillations are quite contained.

The system can work for around 320 hours with a Mean Time To Failure of 319.47 ± 2.63 hours, thus having the system to fail earlier than the mission time of the analysis in most of the simulations.

The last two data analyzed are on the number of component failures in a life cycle 26.98 ± 0.05 and the number of maintenance interventions 12.79 ± 0.02, with two components repaired every time

In case 2 we can now observe how the introduction of an external event, that could cause the failure of multiple components, affects the system and the parameters discussed for case 1. As expected, we can see how both the reliability Fig.[4] and the availability decrease respect to case 1. We can also observe the same behaviour in the oscillations. In this second case the the Mean Time To Failure decreases to 299.56 ± 2.46 hours and both the number of maintenance interventions and the number of components failed increase to 13.88 ± 0.03 and 29.37 ± 0.06 respectively. Both these results are to be expected, since we introduced a further cause of failure.

	MTTF [h]	Component Failures	Maintenance intervention
Case 1	319.47 ± 2.63	26.9749 ± 0.0498	12.7923 ± 0.0186
Case 2	299.56 ± 2.46	29.3684 ± 0.0548	13.8733 ± 0.0275

Table 1: Comparison Table

5 Plots

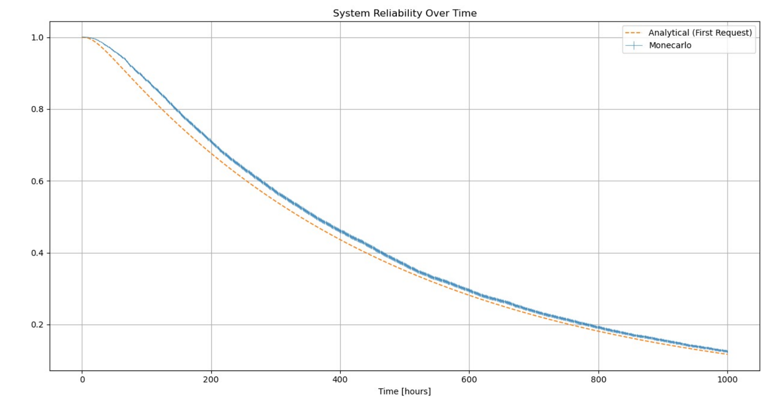

Figure 2: Reliability Case 1

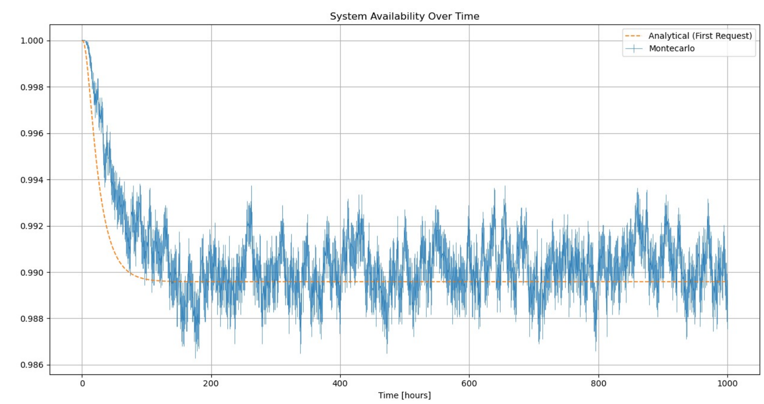

Figure 3: Availability Case 1

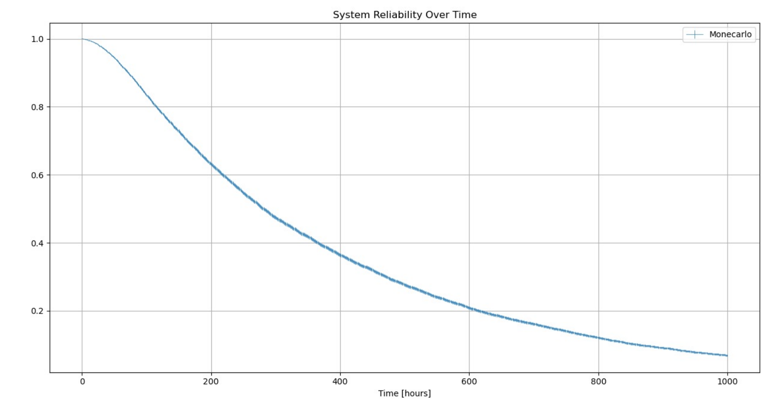

Figure 4: Reliability Case 2

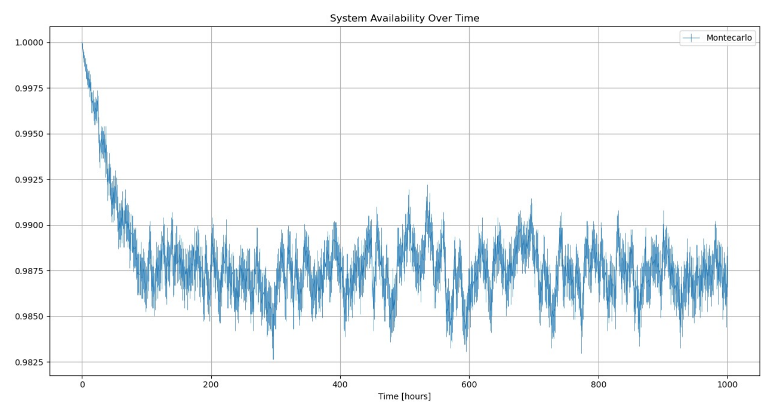

Figure 5: Availability Case 2

6 Pseudo code

Algorithm 1 Function: AvailabilityReliability

Input$\leftarrow$ FailRate, RepairRate, TimeAxis, CommonProb
Output$\leftarrow$ AvailValues, RelyValues

$x, y, z, w \leftarrow true$ ▷ boolean variables, states of the system
$X \leftarrow vector[x, y, z, w]$ ▷ vector of the states of the system
$A \leftarrow$ Transition Matrix of the system
$B \leftarrow differentiate(X)$
[q1, q2, q3, q4] $\leftarrow Solve(A * X - B)$
Availability $\leftarrow q1 + q2 + q3$
AvailabilityValues$\leftarrow$Availability(TIMEAXIS)
$x, y, z, w \leftarrow true$ ▷ boolean variables, states of the system
$A \leftarrow$ Transition Matrix of the system considering z an absorbing state
$B \leftarrow differentiate(X)$
[r1, r2, r3] $\leftarrow Solve(A * X - B)$
Reliability $\leftarrow r1 + r2 + r3$
RelibilityValues$\leftarrow$Relibility(TIMEAXIS)

Algorithm 2 Function: TransitionMatrix

Input← FailRate, RepairRate, TimeAxis, CommonRate, system
Output← TransitionMatrix

NumbWorkingComp←sum(system)
if $NumbWorkingComp \leq 2$ **then** ▷ Repair with more than two failures
    $repair = RepairRate$
else
    $repair = 0$
end if
if $NumbWorkingComp > 1$ **then**
    $fail = NumbWorkingComp \cdot (2^{4-NumbWorkingComp} \cdot FailRate)$
else ▷ If one or less components are working the system is failed
    $fail = 0$
end if
TransitionMatrix←matrix([0, repair, 0],[fail, 0, CommonRate])

Algorithm 3 Function: CommonFailureProbabilities

Input← CommonRate, system
Output← NumCommonFails

for i←1 to sum(working components) **do**
    NumFails← $1 + i$
    tot ← total number of components
    prob ← BinomialDistribution(NumFails, tot, CommonFailProb)
    TransProbArray[i]←prob
end for
CommonDistrib←CumulativeSum(TransProbArray/sum(TransProbArray))
r←RandomNumber[0; 1)
NumCommonFails←CompareDistributionRand(r≤ $CommonDistrib$)+1 ▷ Number of components that can fail due to common cause

Algorithm 4 Function: ComponentSampling

Input← Trans, system
Output← SystemRates, Component

SystemRates←SumLines(Trans)
CompoentDistrib←CumulativeSum(SystemRates/sum(SystemRates))
r←RandomNumber[0; 1)
Component←CompareDistributionRand(r≤ $ComponentDistrib$)
▷ Find the component that transitions

Algorithm 5 Function: TransitionSampling

Input← trans, system, component
Output← Transition

TransitionRatesChosenComponent←trans(system(component))
Rates←TransitionRatesChosenComponent/sum(TransitionRatesChosenComponent)
TransitionDistribution←CumulativeSum(Rates)
r←RandomNumber[0; 1)
Transition←CompareDistributionRand(r≤ $TransitionDistrib$) ▷ Find the transition that occurs

Algorithm 6 Function: UpdateTime

Input← time, systemRates, component, timeAxis
Output← NewTime, IndexPre, IndexPost

systemRate←systemRates(component)
tSample← $-\log(1 - randomNumber)/systemRate$
NewTime←time+tSample
for i←0 to length(TimeAxis) **do**
  if $time \geq TimeAxis[i]$ **then**
    $IndexPre \leftarrow i$
    $i \leftarrow length(TimeAxis) + 1$
  end if
end for
for i←0 to length(TimeAxis) **do**
  if $Newtime \geq TimeAxis[i]$ **then**
    $IndexPost \leftarrow i$
    $i \leftarrow length(TimeAxis) + 1$
  end if
end for

Algorithm 7 Function: UpdateSystem

```
Input← system, component, transition, CommonProb
Output← system, NumCommonFails

NumCommonFails←0
if transition == 1 then
    system[component]←1                          ▷ repair sampled component
    system[random(Compare(system==0))=1]         ▷ repair another random component
else
    if transition == 0 then
        system[component]←0                      ▷ fail sampled component
    else
        if transition == 2 then
            NumCommonFails←commonFail(system)
            system[component]←0                  ▷ fail sampled component
            OtherFailures←NumCommonFails-1
        end if
        if OtherFailures > 0 then
            RandomIndexes←system[random(Compare(system==0))=1]
            system[RandomIndexes]←0 Comment fix random components
        end if
    end if
end if
```

Algorithm 8 Function: SimulateSystem

```
Input← FailRate, RepairRate, TimeAxis, CommonRate, CommonProb,
MissionTime, time
Output← Availability, Reliability, TTFIndex,NumCommonFails, NumFails,
NumRepairs

Firstfailure←True
Availability←zeros(length(TimeAxis))
Reliability←zeros(length(TimeAxis))
TTFIndex←0
NumCommonFails←0
NumFails←0
NumRepairs←0
while time ≤ MissionTime do
                                    ▷ Calling the function previously presented
    matrix←TransitionMatrix(system)
    SystemRates, component←ComponentSampling(matrix, system)
    Transition←TransitionSampling(matrix, system, component)
    time, IndexPre, IndexPost←UpdateTime(time, SystemRates, component)
    system, ComFailedComp←UpdateSystem(system, component, transition)
                                                        ▷ Update counters
    if transition == 1 and sum(system) < 4 then
        Availability[IndexPre to IndexPost]←0
        if FirstFailure == True then
            Reliability[IndexPre to end]←0
            FirstFailure←False
            TTfIndex←IndexPre
        end if
    else
        Availability[IndexPre to IndexPost]←1
    end if
                                        ▷ Update repair and failure counters
    if transition == 1 then
        NumRepairs←+1
    else
        if transition == 0 then
            NumFails←+1
        else
            NumCommonFails←NumCommonFails+ComFailedComp
        end if
    end if

end while
```

Algorithm 9 Function: Main

AvailabilityVec←zeros(length(TIMEAXIS))
RelybilityVec←zeros(length(TIMEAXIS))
TtfIndexesVec←[]
IIRepairs←zeros(STORIES)
IIFails←zeros(STORIES)
IICommon←zeros(STORIES)

for i←0 to STORIES **do**
    time←0
    system←[1,1,1,1]
    call function SimulateSystem
    AvailabilityVec←AvailabilityVec+availability
    RelybilityVec←RelybilityVec+reliability
    TtfIndexesVec[i]←TtfIndex
    IIRepairs[i]←NumRepairs
    IIFails[i]←NumFails
    IICommon[i]←NumCommonFails
end for

IIRepairs←mean(IIRepairs)
IIRepairsError← $std(IIRepairs)/\sqrt{STORIES}$

IIFails←mean(IIFails)
IIFailsError← $std(IIFails)/\sqrt{STORIES})$

MTTF←mean(TIMEAXIS[TtfIndexesVec])
MTTFError← $std(TIMEAXIS[TtfIndexesVec])/\sqrt{STORIES}$

ReliabilityPlot ← ReliabilityVec / STORIES
ReliabilityErr ← $\sqrt{\frac{ReliabilityPlot-ReliabilityPlot^2}{STORIES}}$

AvailabilityPlot ← AvailabilityVec / STORIES
AvailabilityErr ← $\sqrt{\frac{AvailabilityPlot-AvailabilityPlot^2}{STORIES}}$