5heraz

by 1 1

Submission date: 08-Jun-2023 01:43PM (UTC+0500)
Submission ID: 2111639647
File name: Simulation.Assignment.docx (961.5K)
Word count: 1840
Character count: 10210

PETRI NET MODEL OF TURBINE SYSTEM

Simulation And Modeling Project

Submitted By: Sheraz Ahmed (BSE203106)

Hamza Aayan (BSE203009)

Submitted To: Sir Awais Haider

Bachelors of Software Engineering

Capital University of Science and Technology

Project By:

Sheraz Ahmed (BSE203106)

Hamza Ayan (BSE203009)

Date:

7-June-2023

Index:

1. Introduction:

1.1. Introduction to Turbine System:

A turbine system is a mechanical device that uses energy from different sources such as water, steam, or wind, and converts it into useful mechanical energy or electrical power. Turbines are essential in various industries and have been in use for centuries from pumping water to grinding grain and have only evolved in the recent century for generating electricity. They are particularly vital in power generation, where they play a crucial role in converting different energy sources into electricity. Turbine systems are also widely utilized in the aviation industry to power aircraft engines and in the oil and gas industry for extracting and refining petroleum.

This Project will solely focus on the Wind Turbine System, its components, working, petri net model, python simulation etc.

1.2. Wind Turbine System

A wind turbine system is a special type of turbine system solely designed to harness the kinetic energy from the wind and convert it into electrical energy. Wind turbines have gained significant popularity as renewable energy sources due to their environmental benefits and potential for sustainable power generation. Wind turbine system are classified into two types,

1. Horizontal-Axis Turbines
2. Vertical-Axis Turbines

They consist of several key components that work together seamlessly.

1.2.1. Components

The Wind Turbine System consists of the following components that work together to generate electricity.

1. **Rotor:** The rotor is the part of the turbine that captures the wind's energy. It typically consists of two or three blades mounted on a hub. The rotor rotates when the wind blows, converting the kinetic energy of the wind into rotational energy.
2. **Generator:** The generator is connected to the rotor and converts the rotational energy into electrical energy.
3. **Nacelle:** The nacelle houses the rotor and the generator. It is usually located at the top of the wind turbine tower and contains various control systems and equipment.

4. **Tower:** The tower provides structural support for the entire wind turbine system. It is typically made of steel or concrete and elevates the rotor and nacelle to capture the stronger winds available at higher altitudes.

5. **Control System:** The control system regulates the operation of the wind turbine, ensuring optimal performance and safety. It includes sensors, actuators, and software to monitor and control variables such as wind speed, rotor speed, and power output.

6. **Yaw System:** The yaw system allows the wind turbine to rotate horizontally to face the wind. It consists of motors, gears, and sensors that detect the wind direction and adjust the turbine's orientation accordingly. The yaw system ensures that the rotor is constantly facing into the wind for optimal energy capture.

7. **Anemometer and Wind Vane:** Anemometers measure the speed of the wind, while wind vanes determine its direction. These instruments provide crucial data to the control system, allowing it to make adjustments for optimal turbine performance.

1.2.2. Working

The turbine system works by rotating the generator to convert the kinetic wind energy to electrical energy.

- As the wind blows it rotates the blades of the turbine system
- The rotation of the blades causes the rotor to spin.
- The rotor increases the speed of rotation to the generator.
- The generator converts this rotational energy into electrical energy.
- This electrical energy is passed through a control unit that regulates the electric flow and controls the rotation of the generator.
- The electric energy is then sent to a step-up transformer.

1.2.3. Applications

- **Distributed generation:** To produce power for local use, wind turbines can be installed in smaller-scale locations like homes, businesses, or factories.
- **Off-Grid Power Systems:** Wind turbines can be used as independent power sources in isolated locations or on islands to provide electricity when grid connections are not possible.

2. Objective of Project:

The objective of this Project is to provide a basic introduction to the turbine system, with a specific focus on wind turbine systems. It aims to familiarize readers with the components, working principles, and applications of wind turbines. This documentation also provides a brief overview of the Petri Net model of a wind turbine system including its adjacent matrix, coverage analysis as well as its simulation in python. By understanding the key aspects of wind turbine systems, readers will gain insights into the technology behind harnessing wind energy and converting it into electrical power.

This assignment also has a great significance as it provides useful insights into how petri net models are designed and used. Using model tools we can design petri net models for many different systems with its nodes, transitions and input outputs.

3. Data and Result:

3.1. Petri Net Model for Wind Turbine System:

States:

P1: Rotating Blade

P2: MainShaft

P3: Rotor

P4: Generator

P5: Control Unit

P6: Electric Regulator

P7: Step-Up transformer

P8: Signal Unit

P9: Electric Grid

P10: Control house.

P11: Generator off signal.

P12: Generator On signal.

P13: Generator is not ready.

P14: Generator is ready.

Transitions:

T1: The rotating blade (p1) rotates the main-shaft (p2).

T2: The rotating main-shaft (p2) rotates the rotor (p3).

T3: (p14) indicates that the generator is ready and the rotor (p3) increases the rotation speed of generator (p4).

T4: The generator (p4) sends electric output to control unit (p5).

T5: The control unit (p5) sends a signal to the electric regulator (p6).

T6: The electric regulator(p6) sends the adjusted generator output to the step-up transformer(p7).

T7: The step-up transformer (p7) sends the electric output to the grid (p9).

T8: The control unit (p5) sends a signal to the Signal Unit (p8).

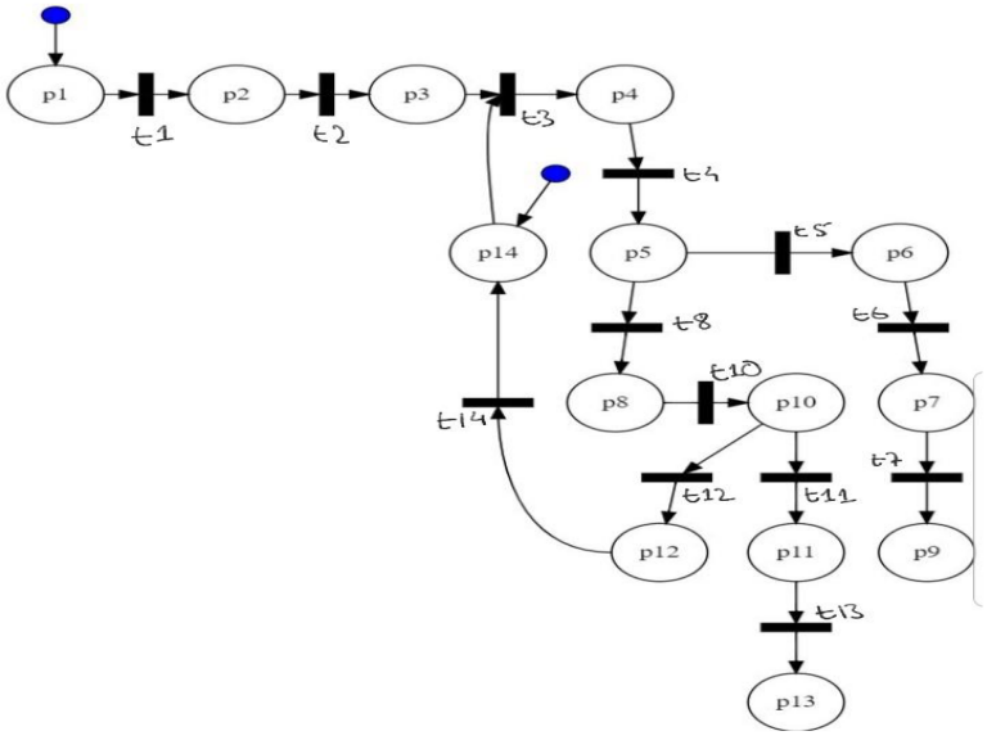
T10: The signal unit (p8) sends signals to Control House (p10).

T11: The control house (p10) sends generator closed signal (p11).

T12: The control house (p10) sends generator on signal (p12).

T13: The generator closed signal (p11) sends a signal that generator is not ready (p13).

T14: The generator on signal (p12) sends a signal that generator is ready (p14).

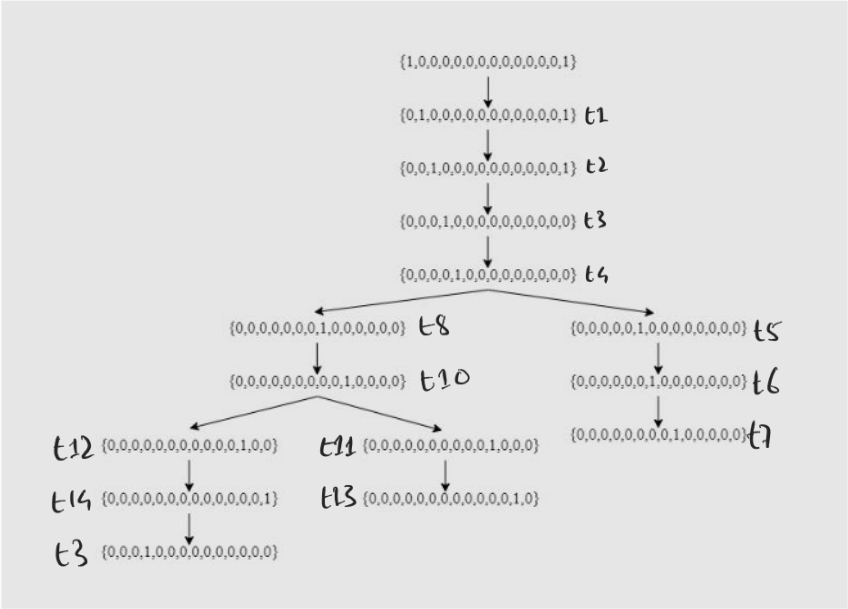


3.2. Petri Net Model code

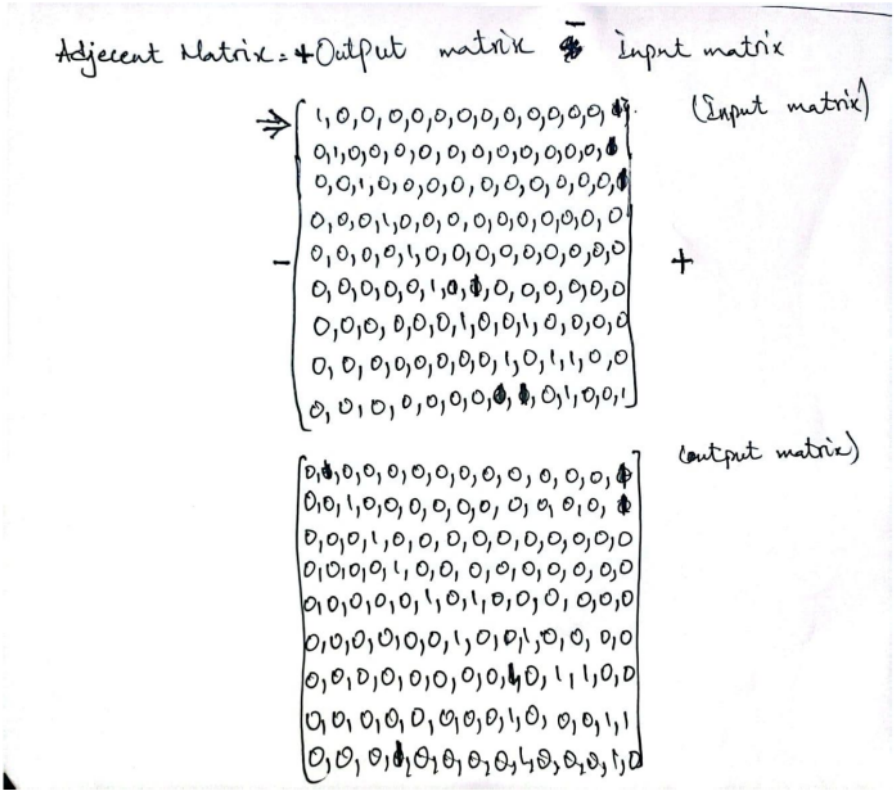
The Above given petri net model has been designed using “graphviz”, the code is provided below.

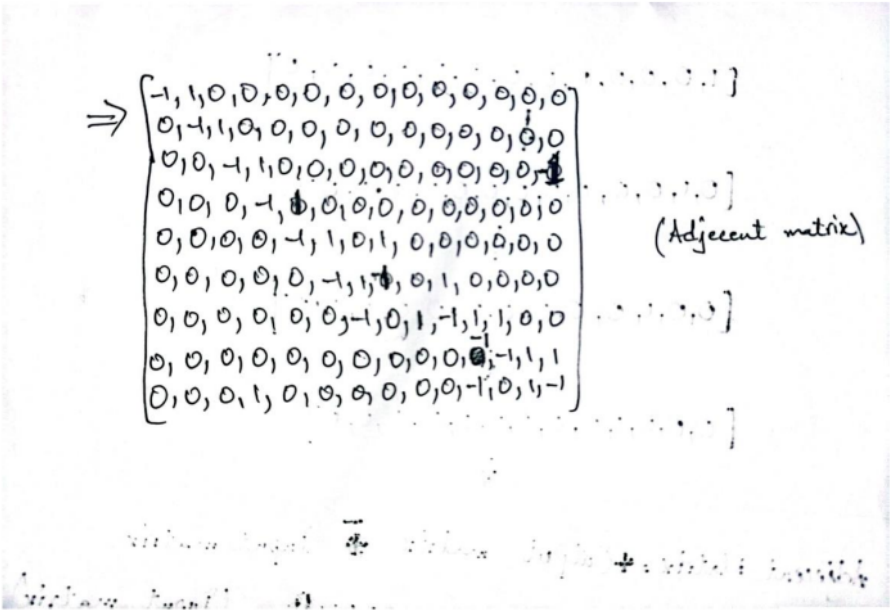
```
digraph petri_net {
    // Places
    node [shape=circle, width=0.7];
    p1, p2, p3, p4, p6, p5, p7, p8, p9, p10, p11, p12, p13, p14;
    // Transitions
    node [shape=rectangle, style=filled, fillcolor=black, label="", width=0.5,
height=0.1];
    t4, t6, t7, t8, t11, t12, t14, t13;

    node [shape=rectangle, style=filled, fillcolor=black, label="", width=0.1,
height= 0.5];
    t1, t2, t3, t5, t10;
    // Tokens
    node [shape=point, width=0.2];
    p1Token [fillcolor=blue, label=""];
    p14Token [fillcolor=blue, label=""];
    // Arc 1st Turbine
    p1 -> t1;
    t1 -> p2 -> t2 -> p3 -> t3 -> p4 -> t4 -> p5 -> t8-> p8;
    p8 -> t10[tailport=e][headport=w];
    t10 -> p10 -> t11 -> p11 -> t13 -> p13;
    p10 -> t12 -> p12;
    p12 -> t14[tailport=w];
    t14 -> p14;
    p14 -> t3[headport=w];
    p5 -> t5[tailport=e][headport=w];
    t5 -> p6 -> t6 -> p7 -> t7 -> p9;
    // Token placements
    p1Token -> p1;
    p14Token -> p14;
    // Graph attributes
    rankdir = TB;  // Left-to-right layout
    edge [arrowhead="vee", arrowtail="none"];  // Token flow direction
    {rank=same; p8;p10;t14}
    {rank=same; t10;p10}
    {rank=same; p1;t1;p2;t2;p3;t3;p4}
    {rank=same; p5;t5;p6;p14}
}
```

3.3. Coverage Analysis:



3.4. Adjacent matrix:





3.5. Python Simulation

```
import random

# Define the initial marking of places
marking = {
    'p1': 1,
    'p14': 1
}

# Define the enabled transitions and their arcs
transitions = {
    't1': {'input': {'p1': 1}, 'output': {'p2': 1}},
    't2': {'input': {'p2': 1}, 'output': {'p3': 1}},
    't3': {'input': {'p3': 1, 'p14': 1}, 'output': {'p4': 1}},
    't4': {'input': {'p4': 1}, 'output': {'p5': 1}},
    't5': {'input': {'p5': 1}, 'output': {'p6': 1}},
    't6': {'input': {'p6': 1}, 'output': {'p7': 1}},
    't7': {'input': {'p7': 1}, 'output': {'p9': 1}},
    't8': {'input': {'p5': 1}, 'output': {'p8': 1}},
    't10': {'input': {'p8': 1}, 'output': {'p10': 1}},
    't11': {'input': {'p10': 1}, 'output': {'p11': 1}},
    't12': {'input': {'p10': 1}, 'output': {'p12': 1}},
    't13': {'input': {'p11': 1}, 'output': {'p13': 1}},
    't14': {'input': {'p12': 1}, 'output': {'p14': 1}}
```

```
}

# Define the simulation function
def simulate():
    print("Initial marking:", marking)

    while True:
        # Find enabled transitions
        enabled_transitions = []
        for transition, arcs in transitions.items():
            if all(place in marking and marking[place] >=
arcs['input'][place] for place in arcs['input']):
                enabled_transitions.append(transition)

        if not enabled_transitions:
            break

        # Choose a random enabled transition
        chosen_transition = random.choice(enabled_transitions)

        # Fire the chosen transition
        for place in transitions[chosen_transition]['input']:
            marking[place] -=
transitions[chosen_transition]['input'][place]

        for place in transitions[chosen_transition]['output']:
            if place in marking:
                marking[place] +=
transitions[chosen_transition]['output'][place]
            else:
                marking[place] =
transitions[chosen_transition]['output'][place]

        print("Fired transition:", chosen_transition)
        print("Updated marking:", marking)
        print()

simulate()

```

61.1. Simulation Output

```
PS C:\Users\hithi\documents> py code.py
Initial marking: {'p1': 1, 'p14': 1}
Fired transition: t1
Updated marking: {'p1': 0, 'p14': 1, 'p2': 1}

Fired transition: t2
Updated marking: {'p1': 0, 'p14': 1, 'p2': 0, 'p3': 1}

Fired transition: t3
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 1}

Fired transition: t4
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 1}

Fired transition: t8
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 0, 'p8': 1}

Fired transition: t10
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 0, 'p8': 0, 'p10': 1}

Fired transition: t11
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 0, 'p8': 0, 'p10': 0, 'p11': 1}

Fired transition: t13
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 0, 'p8': 0, 'p10': 0, 'p11': 0, 'p13': 1}
```

```
PS C:\Users\hithi\documents> py code.py
Initial marking: {'p1': 1, 'p14': 1}
Fired transition: t1
Updated marking: {'p1': 0, 'p14': 1, 'p2': 1}

Fired transition: t2
Updated marking: {'p1': 0, 'p14': 1, 'p2': 0, 'p3': 1}

Fired transition: t3
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 1}

Fired transition: t4
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 1}

Fired transition: t8
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 0, 'p8': 1}

Fired transition: t10
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 0, 'p8': 0, 'p10': 1}

Fired transition: t12
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 0, 'p8': 0, 'p10': 0, 'p12': 1}

Fired transition: t14
Updated marking: {'p1': 0, 'p14': 1, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 0, 'p8': 0, 'p10': 0, 'p12': 0}
```

```
PS C:\Users\hithi\documents> py code.py
Initial marking: {'p1': 1, 'p14': 1}
Fired transition: t1
Updated marking: {'p1': 0, 'p14': 1, 'p2': 1}

Fired transition: t2
Updated marking: {'p1': 0, 'p14': 1, 'p2': 0, 'p3': 1}

Fired transition: t3
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 1}

Fired transition: t4
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 1}

Fired transition: t5
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 0, 'p6': 1}

Fired transition: t6
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 0, 'p6': 0, 'p7': 1}

Fired transition: t7
Updated marking: {'p1': 0, 'p14': 0, 'p2': 0, 'p3': 0, 'p4': 0, 'p5': 0, 'p6': 0, 'p7': 0, 'p9': 1}
```

5. Discussion:

In the given project we have performed the following;

- **Graphviz:**
  By using graphviz, a graph visualization tool, we have made a petri net model with places denoted with “p” and transitions denoted with “t”. The p1 and p14 nodes have tokens denoted by a blue filled circle. All the transitions between the places have been defined in the code. Note that there is no t9 transition.
- **Python:**
  We use python to simulate the petri net system which generates random transition outputs. The Python simulation allowed us to observe how the system evolved from one state to another based on the firing of transitions triggered by the presence of tokens and the defined firing rules.
- Performed coverage analysis to assess the completeness and effectiveness of the model
- Adjacent matrix was has been created to provide a precise representation of the connections between places and transitions in the Petri net model.

6. Conclusion:

In this project, we have explored the Petri net model of a wind turbine system and have simulated the model in Python, while keeping in view the components, workings, and applications of wind turbines. While completing comprehensive analysis of the net model, we have;

- Learned how to design a petri net model in graphviz.

- Learned how to simulate in python.
- The inner operations of a Wind Turbine System.
- Gained valuable insights into the complex behavior and performance of wind turbine systems.

In conclusion, this project has contributed to our understanding of wind turbine systems, emphasizing the significance of the Petri net model and its Python simulation for studying their behavior. By diving deep into the components, workings, and applications of wind turbines, we have highlighted the importance of this renewable energy technology in the transition towards a sustainable future. The knowledge gained from this project can inform further advancements in wind turbine design, operation, and maintenance, ultimately driving the adoption of clean and efficient energy solutions.

7. References:

"Use of Petri Nets to model the maintenance of wind turbines."
J.M.Leigh and S.J.
Loughborough University

5heraz

ORIGINALITY REPORT

5%	3%	1%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Etoju Jacob, Hooman Farzaneh. "Dynamic modeling and experimental validation of a standalone hybrid microgrid system in Fukuoka, Japan", Energy Conversion and Management, 2022 Publication	1%
2	Submitted to Universiti Malaysia Pahang Student Paper	1%
3	Submitted to De Montfort University Student Paper	1%
4	Submitted to Turner Fenton Secondary School Student Paper	1%
5	Submitted to Higher Education Commission Pakistan Student Paper	1%
6	www1.appstate.edu Internet Source	1%
7	rstudio-pubs-static.s3.amazonaws.com Internet Source	1%

8	www.engineeringchoice.com Internet Source	<1%

Exclude quotes	On	Exclude matches	< 8 words
Exclude bibliography	On		