



Analysis for an Agree-Disagree Model of a Political Party

by
E M C H Madushani
SC/2020/11486

Supervisor: Dr L. W. Somathilake

Project report submitted to the
Department of Mathematics
in partial fulfillment of the requirement for the
Level III Industrial Mathematics course unit (IMT3b1 β)
of the

Bachelor of Science (General) Degree

in the
Faculty of Science
University of Ruhuna
Matara
Sri Lanka.

August 2024

DECLARATION

I, E M C H Madushani, declare that the presented project report titled, “ Analysis for an Agree-Disagree Model of a Political Party” is uniquely prepared by me based on the group project carried out under the supervision of Dr L. W. Somathilake Department of Mathematics, Faculty of Science, University of Ruhuna, as a partial fulfillment of the requirements of the level III Industrial Mathematics course unit (IMT3b1 β) of the Bachelor of Science (General) Degree in Department of Mathematics, Faculty of Science, University of Ruhuna, Sri Lanka. It has not been submitted to any other institution or study program by me for any other purpose.

Signature:.....

Date:.....

SUPERVISOR’S RECOMMENDATION

I/We certify that this study was carried out by E M C H Madushani under my/our supervision.

Signature:.....

Date:.....

Dr L. W. Somathilake,
Department of Mathematics,
Faculty of Science,
University of Ruhuna.

ACKNOWLEDGMENTS

I would like convey my sincere gratitude to everyone who helped me finish this project by contributing their essential assistance. First of all, I want to sincerely thank our supervisor Dr. L. W. Somathilake for choosing this project as well as invaluable guidance, unwavering support, and patience throughout the completion of project during this time. I also want to sincerely thank everyone in my group who contributed to the success of this project and helped us all work together to achieve our shared goal. The rest of my thanks go to my parents and all of my friends for their encouragement and good comments during the effort. Lastly, I extend my gratitude to all those unnamed individuals who, directly or indirectly, contributed to this project. Without all of the mentioned contributors and joint efforts, this project would not have been feasible.

ABSTRACT

When people interact with others and form beliefs that they can either agree with or disagree with, an agree-disagree model is frequently used to explain social dynamics and collective decision-making processes. These models try to describe the way that polarisation (disagreement) or consensus (agreement) changes over time. They can be represented using statistical frameworks or network topologies. These analyses enable the prediction of outcomes under various situations, comprehension of the Agree-Disagree Model's behaviour, and identification of the parameters that have the greatest impact on system dynamics. Opinion dynamics, sociology, and political science are a few disciplines where this can be especially helpful.

This project report provides an extensive review of the main objectives, methods, discussions, and results from the Stability and Global Sensitivity Analysis for an Agree-Disagree Model using Java. Developing a technique to assess stability was the main objective of this study, with an emphasis on polling conducted before elections through a mathematical model. A thorough methodology was used to accomplish these goals, which included the fundamental reproductive number and the next-generation matrix approach. According to the study's findings, elections can be significantly impacted by changes in the percentages of those who agree, disagree, and are ignorant.

Keywords: Basic reproductive number, Next generation matrix

TABLE OF CONTENTS

TABLE OF CONTENTS	v
LIST OF TABLES	vii
LIST OF FIGURES	viii
1 Introduction	1
1.1 Background of the Study	1
1.2 Basic Reproductive Number	2
2 Literature Review	4
2.1 Stability Analysis in Opinion Models	4
2.2 Early Opinion Dynamics Models	4
2.3 The Role of Disagreement	5
2.4 Recent Developments	5
3 Problem Statement	7
3.1 Objectives of the Study	10
4 Methodology	11

4.1	Next Generation Matrix	11
4.1.1	Disagree Free Equilibrium	12
4.1.2	Agree Free Equilibrium	14
4.2	Euler Method	14
5	Results	16
6	Discussion and Conclusion	19
6.1	Discussion	19
6.2	Conclusion	20
	References	21
7	AppendixB	22
7.1	Java implementation of agree, disagree model of a political party .	22

LIST OF TABLES

5.1	Parameter values	16
-----	----------------------------	----

LIST OF FIGURES

3.1	Flowchart for model	8
5.1	Graph of parameter set 1	16
5.2	Graph of parameter set 2	16
5.3	Graph of parameter set 3	17
5.4	Graph of parameter set 4	17
5.5	Graph of parameter set 5	17
5.6	Graph of parameter set 6	17
5.7	Graph of parameter set 7	18

CHAPTER 1

Introduction

1.1 Background of the Study

According to the gazette of Democratic Socialist Republic of Sri Lanka, a 'citizen' is a person who must obey the laws of their [3] state and carry out their responsibilities. They are also entitled to all the legal rights and benefits bestowed upon the members of their constituency. Voting is a fundamental right for citizen above 18.

[2] Citizen will participate for political life if they receive benefits. A best individual economic, representation of interest, accountability, protections of rights, social change, strengthening democracy are some of them. While future outcomes allow participants to negotiate and discuss opinions based on a certain conclusion, opinion polls are surveys of intent for a sample of voters. By questioning people about their ideas, public opinion polls are frequently used to determine people's political views, voting patterns, and other behaviours. These days, polls are important for political campaigns. In here we study the conflict between voters' expectations and preferences while providing candidates with voter support evidence. In order to estimate the probability of outcomes, we first create a mathematical model that describes how opinions evolve. We then compute and examine the model's equilibrium points by determining significant stability thresholds for each equilibrium state. So we need to develop more efficient model through statistical and mathematical method. The most popular way is sensitivity analysis(SA) method. It is used to variety of reasons such as communicating, understanding, developing models etc. this poll model describes the opinions of 'Agree', 'Disagree' and 'Ignorant' regarding the idea.

Suppose there are some political parties as A, B, C and D. In here we consider A is one subset and B, C, D is another subset. Then the who vote for 'A' consider as 'agree' and who votes for the 'B,C or D' consider that is a 'disagree' and who doesn't like to vote both or doesn't have an idea about poll, it consider as an 'ignorant' vote.

1.2 Basic Reproductive Number

The average number of secondary cases of an infection is known as the basic reproduction number, or basic reproductive number (also known as the basic reproduction ratio or basic reproductive rate, or R_0). It is mostly used in Epidemiology.

[5]

- $R_0 > 1$ - A populace will be able to begin experiencing the infection's spread. (epidemic occurs)
- $R_0 < 1$ - Percentage of the populace that requires adequate immunisation. (No epidemic)
- $R_0 = 1$ - Remain stable in the population

Here we calculate the R_{D0} the average number of new disagreements produced by an individual disagreeing introduced in a population of ignorant. And also R_{AO} is the average number of new agreement produced by an individual agreeing introduced in a population ignorant. The rate of contact between susceptible and infected individuals, the likelihood of transmission per contact, and the length of infectiousness are some

of the major variables that affect the value of R_0 . Interventions like therapy, immunisation, or behavioural modifications that can lower the rate of transmission are not taken into account by R_0 ; instead, the effective reproductive number R_e accounts for these. Planning for public health requires an understanding of and estimation of R_0 , as it aids in determining the appropriate amount of action needed to contain an outbreak and stop its spread. We can obtain the solution for reproductive number by using 'Next Generation Matrix'.

CHAPTER 2

Literature Review

Analysis of opinion dynamics has long been an area of interest in political science, physics, sociology, and other disciplines. Opinion dynamics models, like the agree-disagree and ignorant (ADI) model, aim to elucidate the processes by which opinions propagate and settle within a population. Specifically, stability analysis sheds light on how these viewpoints converge or diverge over time with varying beginning conditions and parameter values.

2.1 Stability Analysis in Opinion Models

Understanding stability analysis is essential to comprehending these models' long-term behaviour. Specifically, bifurcation analysis and Lyapunov stability are frequently used to investigate the circumstances under which a system will oscillate or reach equilibrium. Sobkowicz (2009), for instance, investigated the stability of beliefs in a society where people alternate between being in favour of and against a certain subject. These investigations have determined crucial points, at which a system's behaviour radically alters, utilising techniques like fixed-point analysis.

2.2 Early Opinion Dynamics Models

The field of opinion dynamics has its roots in the early research of French (1956) and Harary (1959), who employed graph theory to explain how members of a network could affect one another's opinions. This approach was advanced by DeGroot (1974)

who introduced probabilistic models in which people change their beliefs based on weighted averages of the opinions of their neighbours. These seminal works mostly addressed the establishment of consensus or agreement among communities.

2.3 The Role of Disagreement

However, reaching total agreement is rarely possible in real life. Rather, groups frequently become stuck in polarised or disagreeable states. By allowing people to remain uninformed or neutral in addition to agreeing or disagreeing, the agree-disagree and ignorant (ADI) model accurately depicts this reality. Scholars such as Hegselmann and Krause (2002) have investigated models with disagreement. They created a bounded confidence model in which people engage only if their opinions are close enough.

2.4 Recent Developments

The use of network architecture and stochastic factors in ADI models has broadened the scope of stability analysis in recent years. Researchers that have examined the effects of random noise and heterogeneous networks on opinion stability include Castellano et al. (2009). These studies demonstrate how rich and varied behaviours, such as metastability and long-term fluctuations, can result from the introduction of randomness or more sophisticated social systems.

The agree-disagree and ignorant model’s stability analysis emphasises the value of mathematical and computational techniques in comprehending intricate social phenomena. A more accurate representation of opinion dynamics in a variety of

populations is now possible thanks to the addition of disagreement and ignorance to earlier models, which mostly concentrated on consensus formation. It is expected that future studies in this field will keep investigating how network structure, stochasticity, and outside factors affect the stability of opinions.

CHAPTER 3

Problem Statement

The mathematical model herewith has been formulated using three compartment such as,

- Ignorant (I) - people who don't know about the poll or abstain from voting.
- Agree (A) - people in agreement with the idea.
- Disagree (D) - people in disagreement with idea.

A set of assumptions has been used for the modeling process as follows:

1. The target population should be well mixed. So ignorant population also are homogeneously spread.
2. Mortality and recruitment are insignificant. Therefore no individual recruited or died.
3. Individual can communicate through the poll.
4. People who are unsure and abstain from voting are ignorant.

Every individual has their reason for agree, disagree or ignorant. An ignorant individual can be persuaded by someone agree, or someone disagree. Those parameters are written as follows:

- α_1 - Disagree to agree transmission rate
- α_2 - Agree to disagree transmission rate
- β_1 - Ignorant to agree transmission rate
- β_2 - Ignorant to disagree transmission rate
- γ_1 - Interest lost factor of agree individuals

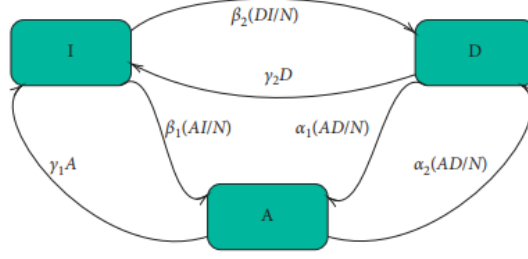


Figure 3.1: Flowchart for model

- γ_2 - Interest lost factor of agree individuals

There are few assumptions as follows:

$$\beta_1 - \gamma_1 > -\frac{\beta_1\gamma_2}{\alpha_1}$$

$$\beta_1 - \gamma_1 > -\frac{\beta_2\gamma_1}{\alpha_2}$$

$$\beta_2 - \gamma_2 > -\frac{\beta_2\gamma_1}{\alpha_2}$$

$$\beta_2 - \gamma_2 > -\frac{\beta_1\gamma_2}{\alpha_1}$$

By considering the all assumptions for this system, ordinary differential equations are written as:

$$I' = -\beta_1 \frac{AI}{N} - \beta_2 \frac{DI}{N} + \gamma_1 A + \gamma_2 D \quad (3.1)$$

$$A' = \beta_1 \frac{AI}{N} + \alpha_1 \frac{AD}{N} - \alpha_2 \frac{AD}{N} - \gamma_1 A \quad (3.2)$$

$$D' = \beta_2 \frac{DI}{N} + \alpha_2 \frac{AD}{N} - \alpha_1 \frac{AD}{N} - \gamma_2 D \quad (3.3)$$

Where,

$$I(0) \geq 0, A(0) \geq 0, D(0) \geq 0,$$

$$N = I + A + D,$$

$$N' = I' + A' + D' = 0$$

Where N is the population size

[1]

The solutions for above ordinary differential equations are non-negative. It is easy to verify that,

$$I = 0 \implies I' \geq 0,$$

$$A = 0 \implies A' \geq 0,$$

$$D = 0 \implies D' \geq 0.$$

Therefore all solutions of systems are non negative,

$N = I + A + D$ is constant

$$I \leq N, A \leq N, \text{ and } D \leq N$$

For the study we assumed the population as follows:

- I - 10
- A - 45

- $D = 45$
- $N = I + A + D = 100$.

3.1 Objectives of the Study

The main objective of this study is to examine how different mathematical model parameters affect public opinion dynamics and stability, particularly agreement and disagreement, in political contexts. To achieve this, it uses methodologies from stability analysis.

CHAPTER 4

Methodology

4.1 Next Generation Matrix

The first step is to calculate the 'Disease Free Equilibrium' (DFE). From this equilibrium point we can calculate the basic reproductive number by using 'Next Generation Matrix'. In here we have to consider the equilibrium states when $A = 0$ and $D = 0$. Let R_{A0} be the agree-free equilibrium and R_{D0} be the disagree-free equilibrium. The epidemic model can be written as, [4]

$$\frac{dx}{dt} = F(x) - V(x)$$

where $F(x)$ is the rate of new infections and $V(x)$ is the rate of transfer of individual. Here

$$F = \begin{pmatrix} \beta_1 \frac{AI}{N} + \alpha_1 \frac{AD}{N} \\ \beta_2 \frac{DI}{N} + \alpha_2 \frac{AD}{N} \end{pmatrix}$$

and

$$V = \begin{pmatrix} \alpha_2 \frac{AD}{N} + \gamma_1 A \\ \alpha_1 \frac{AD}{N} + \gamma_2 D \end{pmatrix}$$

Let's consider the matrix F . Then assume

$$f(A,D) = \beta_1 \frac{AI}{N} + \alpha_1 \frac{AD}{N}$$

and

$$g(A, D) = \beta_2 \frac{DI}{N} + \alpha_2 \frac{AD}{N}$$

Here

$$F^* = \begin{pmatrix} \frac{\partial f}{\partial A} & \frac{\partial f}{\partial D} \\ \frac{\partial g}{\partial A} & \frac{\partial g}{\partial D} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\beta_1}{N}I + \frac{\alpha_1}{N}A & \frac{\alpha_1}{N}A \\ \frac{\alpha_2}{N}D & \frac{\beta_2}{N}I + \frac{\alpha_2}{N}A \end{pmatrix}$$

For the matrix 'V' we assume that,

$$m(A, D) = \frac{\alpha_2}{N}AD + \gamma_1 A$$

$$n(A, D) = \frac{\alpha_1}{N}AD + \gamma_2 D$$

By following previous steps we can the following equations,

$$V^* = \begin{pmatrix} \frac{\alpha_2}{N}D + \gamma_1 & \frac{\alpha_1}{N}A \\ \frac{\alpha_1}{N}D & \frac{\alpha_1}{N}A + \gamma_2 \end{pmatrix}$$

4.1.1 Disagree Free Equilibrium

Suppose there $D = 0$ and no negative opinions. Then,

$$I^* = \frac{\gamma_1}{\beta_1}N, \tag{4.1}$$

$$A^* = \frac{N(\beta_1 - \gamma_1)}{\beta_1}, \quad (4.2)$$

Where I^* and A^* are numbers of ignorant and agree individuals. So disagree free equilibrium point is

$$e_1 = \left(\frac{\gamma_1}{\beta_1} N, \frac{N(\beta_1 - \gamma_1)}{\beta_1}, 0 \right) \quad (4.3)$$

According to the above equations the following equation is implied for disagree free equilibrium.

$$F = \begin{pmatrix} \gamma_1 & \frac{\alpha_1}{N}(\beta_1 + \gamma_1) \\ 0 & \frac{\beta_2}{\beta_1}\gamma_1 + \frac{\alpha_2}{\beta_1}(\beta_1 + \gamma_1) \end{pmatrix}$$

$$V = \begin{pmatrix} \gamma_1 & \frac{\alpha_2}{N}(\beta_1 + \gamma_2) \\ 0 & \frac{\alpha_1}{\beta_1}(\beta_1 + \gamma_1) + \gamma_1 \end{pmatrix}$$

The next step is to calculate the FV^{-1} . Then need to find the eigenvalues of FV^{-1} . The eigenvalues of the systems are:

$$\lambda_1 = -1 \text{ and}$$

$$\lambda_2 = \frac{\alpha_2\beta_1 - \alpha_2\gamma_1 + \beta_2\gamma_1}{\alpha_1\beta_1 - \alpha_1\gamma_1 + \beta_1\gamma_2}$$

So second generation approach for the R_{D0} is,

$$R_{D0} = \frac{\alpha_2\beta_1 - \alpha_2\gamma_1 + \beta_2\gamma_1}{\alpha_1\beta_1 - \alpha_1\gamma_1 + \beta_1\gamma_2}$$

4.1.2 Agree Free Equilibrium

By following the above steps for this system, the threshold can be obtained as,

$$R_{A0} = \frac{\alpha_1\beta_2 - \alpha_1\gamma_2 + \beta_1\gamma_2}{\alpha_2\beta_2 - \alpha_2\gamma_2 + \beta_2\gamma_1}$$

4.2 Euler Method

The Euler method is a numerical technique used to approximate the solutions to ordinary differential equations (ODEs). When differential equations describe the behaviour of the system, it is very helpful in initial value problems. Applying the Euler approach to your stability analysis might be useful when working with dynamical systems such as the agree-disagree and ignorant model. The general Euler differential equation is,

$$\frac{dy}{dt} = f(t, y)$$

and update rule is:

$$y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

Where:

- y_n - current value of the function
- h - step size
- $f(t_n, y_n)$ - derivative function

- y_{n+1} - new value of the function after one step

Suppose there is a system of differential equations that describe the dynamics of a model that is agree-disagree-ignorant. Assume the following differential equations reflect the system:

$$\begin{aligned}\frac{dy}{dt} &= f_I(A, I, D, t) \\ \frac{dy}{dt} &= f_A(A, I, D, t) \\ \frac{dy}{dt} &= f_D(D, I, A, t)\end{aligned}$$

Where $I(t)$ is the number of individuals who ignorant, $A(t)$ is the number who agree, and t is unit time. $D(t)$ is the number of individuals who disagree.

We can write equations for the Euler method, as follows.

$$A_{n+1} = A_n + h \cdot f(t_n, y_n)$$

$$I_{n+1} = I_n + h \cdot f(t_n, y_n)$$

$$D_{n+1} = D_n + h \cdot f(t_n, y_n)$$

f_I, f_A, f_D could be derived from the equations describing the rate of change of opinions in the system.

CHAPTER 5

Results

The following values are used to analyze the stability during political vote.

set no	β_1	β_2	α_1	α_2	γ_1	γ_2	R_{D0}	R_{A0}
1	0.0010	0.1010	0.1010	0.3010	0.5010	0.3010	1.9901	2.0730
2	0.5000	0.1010	0.1010	0.3010	0.5010	0.3010	0.3344	-13.5743
3	0.0010	0.5000	0.1010	0.3010	0.5010	0.3010	-1.9921	0.0657
4	0.0010	0.1010	0.5000	0.3010	0.5010	0.3010	0.4000	10.3864
5	0.0010	0.1010	0.1010	0.5000	0.5010	0.3010	3.9722	0.4028
6	0.0010	0.1010	0.1010	0.3010	1.0000	0.3010	1.9851	-0.4877
7	0.0010	0.1010	0.1010	0.3010	0.5010	0.5000	1.9979	0.5727

Table 5.1: Parameter values

This is the values of we get for the α_1 , α_2 , β_1 , β_2 , γ_1 , γ_2 . By substituting above values to equations we can get following graphs to analyse what will happen in future by analysing the values of R_{D0} and R_{A0} .

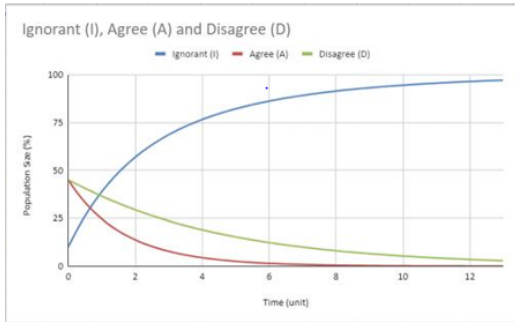


Figure 5.1: Graph of parameter set 1

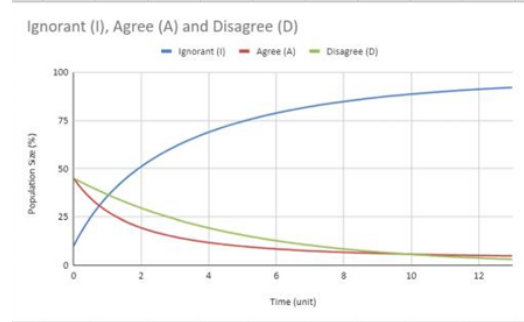


Figure 5.2: Graph of parameter set 2

Let's consider Figure 5.1 graph as initial graph. The rate of ignorant individual increase in here and both agree and disagree individuals decrease with the time.

According to the Figure 5.2 graph , the value of β_1 is increased. Ignorant percentage decrease and Agree percentage increase with the time.

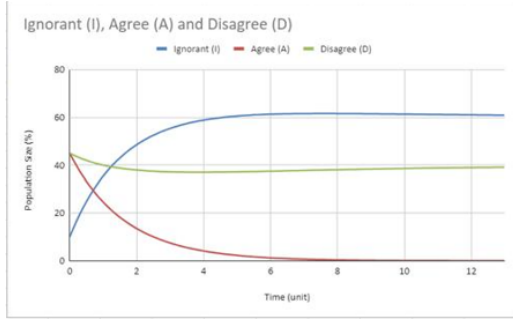


Figure 5.3: Graph of parameter set 3

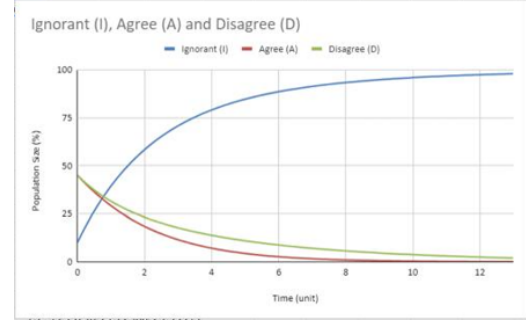


Figure 5.4: Graph of parameter set 4

According to the values of set 3 in above table Figure 5.3 shows increase of the Disagree percentage and decrease of Ignorant percentage. Here we increased the value of β_2 . Also When increase the value of α_1 the Figure 5.4 show the increase of Agree percentage and decrease of the Disagree percentage.

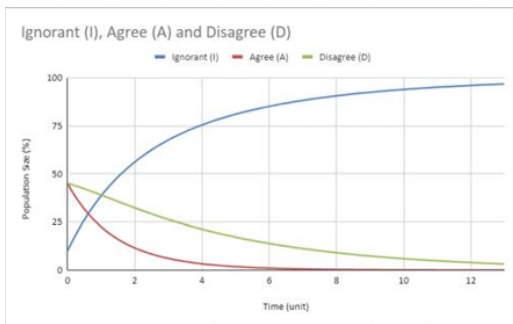


Figure 5.5: Graph of parameter set 5

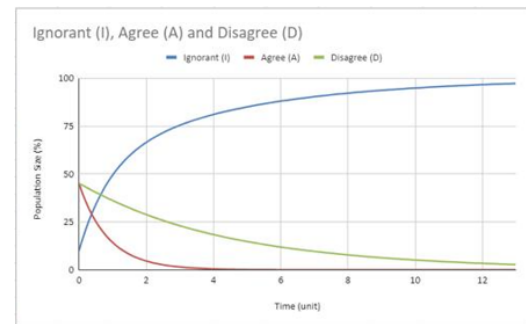


Figure 5.6: Graph of parameter set 6

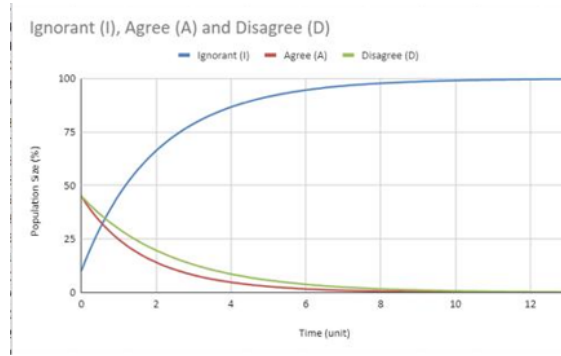


Figure 5.7: Graph of parameter set 7

While the increase of α_2 value the Figure 5.5 show the increase of Disagree percentage and decrease of the Agree percentage. Also when increase the γ_1 Figure 5.6 displays decrease of the agree individual percentage and increase of the ignorant individual percentage.

According to the final values of parameter set in table, the Figure 5.7 show the decrease of disagree percentage and increase of ignorant percentage.

By analysing the values of R_{A0} and R_{D0} or analysing the graphs we can guess what will happen in future for each political parties in unit time.

If $R_0 > 1$ - an epidemic occurs

If $R_0 < 1$ - there will probably be no epidemic.

CHAPTER 6

Discussion and Conclusion

6.1 Discussion

A political party's agree-disagree model seeks to represent the processes of opinion development within a population. According to this methodology, people can choose to support a certain political ideology, oppose it, or stay neutral (ignorant). It is likely that the research looks into what influences these states' transitions and how they come together to form a stable opinion distribution. The stability analysis's findings show the circumstances in which a certain level of agreement or disagreement takes hold inside the political party. For instance, one opinion may become the consensus while other opinions gradually disappear when specific parameters (such the rate of opinion adoption or abandonment) are adjusted.

Because the media is such a potent force in forming and influencing public opinion, its function in a "Agree-Disagree Model of a Political Party" is vital. Within the framework of this concept, media can have a substantial impact on the rates at which people adopt a political ideology, either by agreeing with it or not, or by staying ignorant (indifferent). So the role of media effect for the all parameters $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2$.

6.2 Conclusion

A useful framework for understanding the dynamic processes of opinion formation inside a political party is the agree-disagree model. By classifying people into three states 'agree', 'disagree', and 'ignorant' the model accurately depicts the complexities of actual political processes as well as how polarisation and consensus change over time.

The stability analysis showed that the parameters influencing opinion change, like peer influence, media exposure, and people's stubbornness to change their views, have a significant impact on the likelihood of reaching consensus, polarisation, or maintaining a significant portion of indifferent members. The media, in particular, is very important since it may act as a catalyst to promote some viewpoints while stifling others, which can affect how political perspectives are distributed throughout the party.

To wrap it up, the agree-disagree model emphasises how crucial it is to comprehend the mechanisms underlying opinion dynamics in political contexts. Political leaders have the ability to steer their party towards more unity or take advantage of newly formed divisions for their own strategic gain by adjusting important criteria. In order to further improve the model's predictive ability, future research might concentrate on incorporating more intricate social variables, like personal identification or economic circumstances.

References

- [1] Sara Bidah, Omar Zakary, and Mostafa Rachik. Stability and global sensitivity analysis for an agree-disagree model: Partial rank correlation coefficient and latin hypercube sampling methods. *International Journal of Differential Equations*, 2020(1):5051248, 2020.
- [2] C.Mathew Co. Fundamental rights in sri lanka. <https://www.cmathew.com/sri-lanka-law/263-fundamental-rights-in-sri-lanka>.
- [3] The Parliament of Sri Lanka. Purposes of voting. <https://www.parliament.lk/en/how-parliament-works/business-of-parliament/purposes-of-voting>, December 2023.
- [4] unknown. Next-generation matrix. https://en.wikipedia.org/wiki/Next-generation_matrix, May 2023.
- [5] Mark EJ Woolhouse, Daniel T Haydon, and Rustom Antia. Emerging pathogens: the epidemiology and evolution of species jumps. *Trends in ecology & evolution*, 20(5):238–244, 2005.

CHAPTER 7

AppendixB

7.1 Java implementation of agree, disagree model of a political party

```

1 import java.io.FileOutputStream;
2 import java.io.IOException;
3 import org.apache.poi.ss.usermodel.*;
4 import org.apache.poi.xssf.usermodel.XSSFWorkbook;
5 import java.util.Scanner;
6
7
8 public class EulerBasic3 {
9
10     public static void main(String[] args) {
11
12         //Taking values for the parameters of the differential
13         equations from the keyboard
14
15         Scanner scanner = new Scanner(System.in);
16
17
18         System.out.println("Enter alpha_1:");
19         double alpha1 = scanner.nextDouble();
20
21
22         System.out.println("Enter alpha_2:");
23         double alpha2 = scanner.nextDouble();
24
25
26         System.out.println("Enter beta_1:");
27         double beta1 = scanner.nextDouble();
28
29
30         System.out.println("Enter beta_2:");
31         double beta2 = scanner.nextDouble();
32
33         System.out.println("Enter gamma_1:");
34         double gamma1 = scanner.nextDouble();
35
36         System.out.println("Enter gamma_2:");
37         double gamma2 = scanner.nextDouble();
38
39         System.out.println("Enter delta_1:");
40         double delta1 = scanner.nextDouble();
41
42         System.out.println("Enter delta_2:");
43         double delta2 = scanner.nextDouble();
44
45         System.out.println("Enter epsilon_1:");
46         double epsilon1 = scanner.nextDouble();
47
48         System.out.println("Enter epsilon_2:");
49         double epsilon2 = scanner.nextDouble();
50
51         System.out.println("Enter zeta_1:");
52         double zeta1 = scanner.nextDouble();
53
54         System.out.println("Enter zeta_2:");
55         double zeta2 = scanner.nextDouble();
56
57         System.out.println("Enter eta_1:");
58         double eta1 = scanner.nextDouble();
59
60         System.out.println("Enter eta_2:");
61         double eta2 = scanner.nextDouble();
62
63         System.out.println("Enter theta_1:");
64         double theta1 = scanner.nextDouble();
65
66         System.out.println("Enter theta_2:");
67         double theta2 = scanner.nextDouble();
68
69         System.out.println("Enter phi_1:");
70         double phi1 = scanner.nextDouble();
71
72         System.out.println("Enter phi_2:");
73         double phi2 = scanner.nextDouble();
74
75         System.out.println("Enter chi_1:");
76         double chi1 = scanner.nextDouble();
77
78         System.out.println("Enter chi_2:");
79         double chi2 = scanner.nextDouble();
80
81         System.out.println("Enter psi_1:");
82         double psi1 = scanner.nextDouble();
83
84         System.out.println("Enter psi_2:");
85         double psi2 = scanner.nextDouble();
86
87         System.out.println("Enter omega_1:");
88         double omega1 = scanner.nextDouble();
89
90         System.out.println("Enter omega_2:");
91         double omega2 = scanner.nextDouble();
92
93         System.out.println("Enter sigma_1:");
94         double sigma1 = scanner.nextDouble();
95
96         System.out.println("Enter sigma_2:");
97         double sigma2 = scanner.nextDouble();
98
99         System.out.println("Enter tau_1:");
100        double tau1 = scanner.nextDouble();
101
102        System.out.println("Enter tau_2:");
103        double tau2 = scanner.nextDouble();
104
105        System.out.println("Enter nu_1:");
106        double nu1 = scanner.nextDouble();
107
108        System.out.println("Enter nu_2:");
109        double nu2 = scanner.nextDouble();
110
111        System.out.println("Enter xi_1:");
112        double xi1 = scanner.nextDouble();
113
114        System.out.println("Enter xi_2:");
115        double xi2 = scanner.nextDouble();
116
117        System.out.println("Enter eta_1:");
118        double eta1 = scanner.nextDouble();
119
120        System.out.println("Enter eta_2:");
121        double eta2 = scanner.nextDouble();
122
123        System.out.println("Enter theta_1:");
124        double theta1 = scanner.nextDouble();
125
126        System.out.println("Enter theta_2:");
127        double theta2 = scanner.nextDouble();
128
129        System.out.println("Enter phi_1:");
130        double phi1 = scanner.nextDouble();
131
132        System.out.println("Enter phi_2:");
133        double phi2 = scanner.nextDouble();
134
135        System.out.println("Enter chi_1:");
136        double chi1 = scanner.nextDouble();
137
138        System.out.println("Enter chi_2:");
139        double chi2 = scanner.nextDouble();
140
141        System.out.println("Enter psi_1:");
142        double psi1 = scanner.nextDouble();
143
144        System.out.println("Enter psi_2:");
145        double psi2 = scanner.nextDouble();
146
147        System.out.println("Enter omega_1:");
148        double omega1 = scanner.nextDouble();
149
150        System.out.println("Enter omega_2:");
151        double omega2 = scanner.nextDouble();
152
153        System.out.println("Enter sigma_1:");
154        double sigma1 = scanner.nextDouble();
155
156        System.out.println("Enter sigma_2:");
157        double sigma2 = scanner.nextDouble();
158
159        System.out.println("Enter tau_1:");
160        double tau1 = scanner.nextDouble();
161
162        System.out.println("Enter tau_2:");
163        double tau2 = scanner.nextDouble();
164
165        System.out.println("Enter nu_1:");
166        double nu1 = scanner.nextDouble();
167
168        System.out.println("Enter nu_2:");
169        double nu2 = scanner.nextDouble();
170
171        System.out.println("Enter xi_1:");
172        double xi1 = scanner.nextDouble();
173
174        System.out.println("Enter xi_2:");
175        double xi2 = scanner.nextDouble();
176
177        System.out.println("Enter eta_1:");
178        double eta1 = scanner.nextDouble();
179
180        System.out.println("Enter eta_2:");
181        double eta2 = scanner.nextDouble();
182
183        System.out.println("Enter theta_1:");
184        double theta1 = scanner.nextDouble();
185
186        System.out.println("Enter theta_2:");
187        double theta2 = scanner.nextDouble();
188
189        System.out.println("Enter phi_1:");
190        double phi1 = scanner.nextDouble();
191
192        System.out.println("Enter phi_2:");
193        double phi2 = scanner.nextDouble();
194
195        System.out.println("Enter chi_1:");
196        double chi1 = scanner.nextDouble();
197
198        System.out.println("Enter chi_2:");
199        double chi2 = scanner.nextDouble();
200
201        System.out.println("Enter psi_1:");
202        double psi1 = scanner.nextDouble();
203
204        System.out.println("Enter psi_2:");
205        double psi2 = scanner.nextDouble();
206
207        System.out.println("Enter omega_1:");
208        double omega1 = scanner.nextDouble();
209
210        System.out.println("Enter omega_2:");
211        double omega2 = scanner.nextDouble();
212
213        System.out.println("Enter sigma_1:");
214        double sigma1 = scanner.nextDouble();
215
216        System.out.println("Enter sigma_2:");
217        double sigma2 = scanner.nextDouble();
218
219        System.out.println("Enter tau_1:");
220        double tau1 = scanner.nextDouble();
221
222        System.out.println("Enter tau_2:");
223        double tau2 = scanner.nextDouble();
224
225        System.out.println("Enter nu_1:");
226        double nu1 = scanner.nextDouble();
227
228        System.out.println("Enter nu_2:");
229        double nu2 = scanner.nextDouble();
230
231        System.out.println("Enter xi_1:");
232        double xi1 = scanner.nextDouble();
233
234        System.out.println("Enter xi_2:");
235        double xi2 = scanner.nextDouble();
236
237        System.out.println("Enter eta_1:");
238        double eta1 = scanner.nextDouble();
239
240        System.out.println("Enter eta_2:");
241        double eta2 = scanner.nextDouble();
242
243        System.out.println("Enter theta_1:");
244        double theta1 = scanner.nextDouble();
245
246        System.out.println("Enter theta_2:");
247        double theta2 = scanner.nextDouble();
248
249        System.out.println("Enter phi_1:");
250        double phi1 = scanner.nextDouble();
251
252        System.out.println("Enter phi_2:");
253        double phi2 = scanner.nextDouble();
254
255        System.out.println("Enter chi_1:");
256        double chi1 = scanner.nextDouble();
257
258        System.out.println("Enter chi_2:");
259        double chi2 = scanner.nextDouble();
260
261        System.out.println("Enter psi_1:");
262        double psi1 = scanner.nextDouble();
263
264        System.out.println("Enter psi_2:");
265        double psi2 = scanner.nextDouble();
266
267        System.out.println("Enter omega_1:");
268        double omega1 = scanner.nextDouble();
269
270        System.out.println("Enter omega_2:");
271        double omega2 = scanner.nextDouble();
272
273        System.out.println("Enter sigma_1:");
274        double sigma1 = scanner.nextDouble();
275
276        System.out.println("Enter sigma_2:");
277        double sigma2 = scanner.nextDouble();
278
279        System.out.println("Enter tau_1:");
280        double tau1 = scanner.nextDouble();
281
282        System.out.println("Enter tau_2:");
283        double tau2 = scanner.nextDouble();
284
285        System.out.println("Enter nu_1:");
286        double nu1 = scanner.nextDouble();
287
288        System.out.println("Enter nu_2:");
289        double nu2 = scanner.nextDouble();
290
291        System.out.println("Enter xi_1:");
292        double xi1 = scanner.nextDouble();
293
294        System.out.println("Enter xi_2:");
295        double xi2 = scanner.nextDouble();
296
297        System.out.println("Enter eta_1:");
298        double eta1 = scanner.nextDouble();
299
300        System.out.println("Enter eta_2:");
301        double eta2 = scanner.nextDouble();
302
303        System.out.println("Enter theta_1:");
304        double theta1 = scanner.nextDouble();
305
306        System.out.println("Enter theta_2:");
307        double theta2 = scanner.nextDouble();
308
309        System.out.println("Enter phi_1:");
310        double phi1 = scanner.nextDouble();
311
312        System.out.println("Enter phi_2:");
313        double phi2 = scanner.nextDouble();
314
315        System.out.println("Enter chi_1:");
316        double chi1 = scanner.nextDouble();
317
318        System.out.println("Enter chi_2:");
319        double chi2 = scanner.nextDouble();
320
321        System.out.println("Enter psi_1:");
322        double psi1 = scanner.nextDouble();
323
324        System.out.println("Enter psi_2:");
325        double psi2 = scanner.nextDouble();
326
327        System.out.println("Enter omega_1:");
328        double omega1 = scanner.nextDouble();
329
330        System.out.println("Enter omega_2:");
331        double omega2 = scanner.nextDouble();
332
333        System.out.println("Enter sigma_1:");
334        double sigma1 = scanner.nextDouble();
335
336        System.out.println("Enter sigma_2:");
337        double sigma2 = scanner.nextDouble();
338
339        System.out.println("Enter tau_1:");
340        double tau1 = scanner.nextDouble();
341
342        System.out.println("Enter tau_2:");
343        double tau2 = scanner.nextDouble();
344
345        System.out.println("Enter nu_1:");
346        double nu1 = scanner.nextDouble();
347
348        System.out.println("Enter nu_2:");
349        double nu2 = scanner.nextDouble();
350
351        System.out.println("Enter xi_1:");
352        double xi1 = scanner.nextDouble();
353
354        System.out.println("Enter xi_2:");
355        double xi2 = scanner.nextDouble();
356
357        System.out.println("Enter eta_1:");
358        double eta1 = scanner.nextDouble();
359
360        System.out.println("Enter eta_2:");
361        double eta2 = scanner.nextDouble();
362
363        System.out.println("Enter theta_1:");
364        double theta1 = scanner.nextDouble();
365
366        System.out.println("Enter theta_2:");
367        double theta2 = scanner.nextDouble();
368
369        System.out.println("Enter phi_1:");
370        double phi1 = scanner.nextDouble();
371
372        System.out.println("Enter phi_2:");
373        double phi2 = scanner.nextDouble();
374
375        System.out.println("Enter chi_1:");
376        double chi1 = scanner.nextDouble();
377
378        System.out.println("Enter chi_2:");
379        double chi2 = scanner.nextDouble();
380
381        System.out.println("Enter psi_1:");
382        double psi1 = scanner.nextDouble();
383
384        System.out.println("Enter psi_2:");
385        double psi2 = scanner.nextDouble();
386
387        System.out.println("Enter omega_1:");
388        double omega1 = scanner.nextDouble();
389
390        System.out.println("Enter omega_2:");
391        double omega2 = scanner.nextDouble();
392
393        System.out.println("Enter sigma_1:");
394        double sigma1 = scanner.nextDouble();
395
396        System.out.println("Enter sigma_2:");
397        double sigma2 = scanner.nextDouble();
398
399        System.out.println("Enter tau_1:");
400        double tau1 = scanner.nextDouble();
401
402        System.out.println("Enter tau_2:");
403        double tau2 = scanner.nextDouble();
404
405        System.out.println("Enter nu_1:");
406        double nu1 = scanner.nextDouble();
407
408        System.out.println("Enter nu_2:");
409        double nu2 = scanner.nextDouble();
410
411        System.out.println("Enter xi_1:");
```

```

23
24     System.out.println("Enter beta_2:");
25     double beta2 = scanner.nextDouble();
26
27     System.out.println("Enter gamma_1:");
28     double gamma1 = scanner.nextDouble();
29
30     System.out.println("Enter gamma_2:");
31     double gamma2 = scanner.nextDouble();
32
33     // Close the scanner after reading all inputs
34     scanner.close();
35
36     // Calculating R_D0
37     double numeratorRD0 = alpha2 * beta1 - alpha2 * gamma1 +
beta2 * gamma1;
38     double denominatorRD0 = alpha1 * beta1 - alpha1 * gamma1 +
beta1 * gamma2;
39     double RD0 = numeratorRD0 / denominatorRD0;
40
41     // Calculating R_A0
42     double numeratorRA0 = alpha1 * beta2 - alpha1 * gamma2 +
beta1 * gamma2;
43     double denominatorRA0 = alpha2 * beta2 - alpha2 * gamma2 +
beta2 * gamma1;
44     double RA0 = numeratorRA0 / denominatorRA0;
45
46     System.out.println("R_D0 = " + RD0);

```



```

47      System.out.println("R_A0 = " + RAO);
48
49      double N = 100.0;  // Total population
50
51      // Initial values of I, A and D
52      double I = 10.0;
53      double A = 45.0;
54      double D = 45.0;
55
56      // Time parameters for euler method
57      double dt = 0.1;  // Time step
58      double T = 13;  // Total time
59
60      // Number of iterations
61      int steps = (int)(T / dt);
62
63      // Creating a new Workbook
64      Workbook workbook = new XSSFWorkbook();
65      // Creating a new sheet
66      Sheet sheet = workbook.createSheet("Euler Method Results");
67
68      // Creating header row
69      Row header = sheet.createRow(0);
70      header.createCell(0).setCellValue("Time (t)");
71      header.createCell(1).setCellValue("Ignorant (I)");
72      header.createCell(2).setCellValue("Agree (A)");
73      header.createCell(3).setCellValue("Disagree (D)");
74

```

```

75     // Euler method loop
76     for (int i = 0; i <= steps; i++) {
77         double time = i * dt;
78
79         // Creating a new row in the sheet
80         Row row = sheet.createRow(i + 1);
81         row.createCell(0).setCellValue(time);
82         row.createCell(1).setCellValue(I);
83         row.createCell(2).setCellValue(A);
84         row.createCell(3).setCellValue(D);
85
86         double I_prime = -beta1 * (A * I) / N - beta2 * (D * I)
/ N + gamma1 * A + gamma2 * D;
87         double A_prime = beta1 * (A * I) / N + alpha1 * (A * D)
/ N - alpha2 * (A * D) / N - gamma1 * A;
88         double D_prime = beta2 * (D * I) / N + alpha2 * (A * D)
/ N - alpha1 * (A * D) / N - gamma2 * D;
89
90         // Update values
91         I += I_prime * dt;
92         A += A_prime * dt;
93         D += D_prime * dt;
94     }
95
96     // Writing the output to an external excel file
97     try (FileOutputStream fileOut = new FileOutputStream("
EulerResults.xlsx")) {
98         workbook.write(fileOut);

```

```
99         } catch (IOException e) {
100             e.printStackTrace();
101         } finally {
102             try {
103                 workbook.close();
104             } catch (IOException e) {
105                 e.printStackTrace();
106             }
107         }
108
109         System.out.println("Results can be seen on EulerResults.xlsx
110 ");
111     }
```