sciendo

International Journal of Advanced Statistics and IT&C for Economics and Life Sciences

Volume 13
Number 1
December 2023

ISSN 2559 - 365X
eISSN 2067 - 354X

DOI: 10.2478/ijasitels-2023-0001

A Categorical Metamodel for Reactive Kripke Frames

Daniel C. Crăciunean[1]

[1]*Computer Science and Electrical and Electronics Engineering Department, Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania*
daniel.craciunean@ulbsibiu.ro

Abstract

In a reactive Kripke model, evaluation of logical operators can cause reconfiguration of the model in which the formula is evaluated. Therefore, in reactive Kripke frames, the evaluation of a logical formula in a world depends both on the world in which the evaluation is made and on the worlds it has passed through previously. The result is an extended semantics, which can specify a class of modal logics more comprehensive than the class specified by ordinary Kripke frames. This paper introduces a metamodel for reactive Kripke frames, based on the concept of categorical sketch. We believe that the categorical sketch is an appropriate metamodel for specifying a Kripke frame model.

Keywords: modal logic, Kripke frames, reactive Kripke frames, categorical sketch

1 Introduction

Modal logic extends classical logic with denoted modal operators, usually with the symbols □ and ◊, which express the way in which a logical formula is satisfied. The interpretation of modal operators depends on the concept of truth that we want to specify through logical formulas. Thus, some modalities that have been formalized in modal logic include: alethic or truth modalities, where □ has the meaning of "necessary" and ◊ has the meaning of "possible"; deontic modalities where □ has the meaning of "mandatory" and ◊ has the meaning of "permitted"; epistemic modality or modes of knowledge where □ has the meaning of "it is known that" or modes of belief where □ has the meaning of "it is believed that". Real applications define many other interpretations of these symbols. In general, the operators □ and ◊ are not independent but are linked by the relation $\Diamond\varphi=\neg\Box\neg\varphi$.

Formulas from modal logic cannot be interpreted on the basis of truth tables. For example, for □p, where p is an atomic proposition, we cannot define a truth table because when p is true in one interpretation, it does not follow that p is true in all interpretations. The reference semantic model for modal logic is the "possible worlds" model, introduced by Saul Kripke in 1959. A Kripke frame is a structure consisting of a set S of possible worlds and a binary relation R on the set S, in other words a Kripke frame is a graph whose nodes are possible worlds and whose arcs represent the accessibility of each world to other possible worlds.

In 2004, D. Gabbay introduced a concept of reactivity which assumes that reactive frames are relational structures in which the structure of a frame, at a given moment, is determined not only by the world in which it is located, but also by previous worlds [4]. In other words, each transition, from one world to another, can modify the relational structure of the frame.

The specification of such reactive structures is based on graphs endowed with several types of arrows, single arrows, double arrows and sometimes triple arrows. Double arrows connect nodes with other arrows, or arrows between them [4, 8]. In this approach, the double arrows have the role of alternately changing the state of the target arrows from active to inactive and vice versa during the transitions of the relational structure. This process is cumulative and results in the dynamic transformation of the relational structure of the system.
This idea of dynamic reconfiguration of a relational structure has applications in various modeling fields such as the extension of modal logic to reactive modal logic, the extension of context-independent grammars to reactive context-independent grammars, the definition of reactive automata and other reactive structures [5, 7, 6, 10].
In the relational semantics of modal logic, different worlds are considered in which the values of the propositions change depending on the values of the propositional variables in these worlds. Therefore, the value of a sentence depends exclusively on the accessible worlds. But in a Kripke model the accessible worlds are fixed. In a reactive Kripke model, evaluation of logical operators can cause reconfiguration of the model in which the formula is evaluated. In [4], this reconfiguration is specified by double arrows, which enable or disable other arrows.
The modeling of reactive Kripke frames is based on a type of special graphs that undergo certain transformations when its edges are traversed and which are called reactive graphs. Therefore, in reactive Kripke frames, the evaluation of a logical formula in a world depends both on the world in which the evaluation is made and on the worlds it has passed through previously. The result is an extended semantics, which can specify a class of modal logics more comprehensive than the class specified by ordinary Kripke frames [11].
This paper introduces a metamodel for reactive Kripke frames, based on the concept of categorical sketch. We believe that the categorical sketch is an appropriate metamodel for specifying a metamodel for Kripke frames.
Section 2 contains some general notions and notations used in the rest of the paper. Section 3 presents the static dimension of the Kripke frame models, section 4 presents the reactive dimension of the reactive Kripke frame models, and section 5 concludes the paper with some conclusions.

2 General notions and notations

The basic modal propositional logic language is built on the vocabulary formed by a set P, of atomic propositions, to which are added the usual operators from propositional logic and two unary modal operators, which we denote by □ and ◊. In the modal logic of truth, the operator □ is read "necessary", and ◊ is read "possible". In the BNF notation, well-formed formulas from modal logic are defined as follows:

<Wff>::=<Proposition>|(¬ <Wff>) |
(<Wff > ∧ < Wff >) |(<Wff > ∨ < Wff >) | (<Wff > → < Wff >) |
(<Wff > ↔ < Wff >) |□ (<Wff >) | ◊(<Wff >)

where <Proposition> represents any atomic proposition from the set P and each occurrence of <Wff> represents a well-formed formula. We denote the set of well-formed formulas, with W.

A Kripke model consists of a set S of possible worlds, a binary relation, of accessibility, R on the set S and an application $L:S\rightarrow 2^P$, which associates to each possible world $s\in S$, a set of well-formed formulas, satisfied in the world s. The pair F=(S, R), is a graph called a Kripke frame

We will denote the satisfaction relation of a well-formed formula ϕ in a world $s\in S$ with $s\Vdash\phi$. If we have a Kripke model M=(S, R, L), and a world $s\in S$, then the satisfaction for a well-formed formula ϕ is defined recursively [2] as described below.

For operators from classical logic, the satisfaction condition is that L(s) be a model for the formula φ, that is:

"$s\Vdash p$ iff $p\in L(s)$, where p is an atomic proposition; $s\Vdash\neg\varphi$ iff $s\nVdash\varphi$;

$s\Vdash\varphi\wedge\psi$ iff $s\Vdash\varphi$ and $s\Vdash\psi$; $s\Vdash\varphi\vee\psi$ iff $s\Vdash\varphi$, or $s\Vdash\psi$;

$s\Vdash\varphi\rightarrow\psi$ iff $s\Vdash\varphi$ implies $s\Vdash\psi$; $s\Vdash\varphi\leftrightarrow\psi$ iff ($s\Vdash\varphi$ iff $s\Vdash\psi$)";

and for modal operators, satisfaction is defined as follows:

"$s\Vdash\Box\psi$ iff, $\forall\ s_1\in S$ with $R(s, s_1)$, it results $s_1\Vdash\psi$;

$s\Vdash\Diamond\psi$ iff $\exists s_1\in S$ with, $R(s, s_1)$ and $s_1\Vdash\psi$."

As we can see, the satisfaction of the formula $\Box\psi$ is conditioned by the satisfaction of ψ in all accessible worlds in s, and the satisfaction of the formula $\Diamond\psi$ is conditioned by the satisfaction of ψ in at least one accessible world in s. This observation tells us that the validity of a modal logic formula, depends to a great extent on the accessibility relation R, i.e., on the graph corresponding to the Kripke model. Therefore, the validity of a modal formula can be realized or cancelled by an appropriate transformation of this graph.

The evaluation of logical formulas is done starting from a set of formulas or schemes of logical formulas that we impose as true, in the context of a model, and which are called axioms. The validity of these formulas, in a Kripke model, can be imposed by conditions on the graph that represents the relation R on the set S of possible worlds.

In this paper, we will enforce the realization of these axioms or axiom schemes, at the metamodel level, using the concept of a categorical sketch. We consider that the categorical sketch is a metamodel capable of specifying all the constraints necessary to specify a Kripke frame, customized and appropriate to the concrete requirements of practical applications.

A categorical sketch is a graph that represents a metamodel together with a series of constraints on the models represented by this metamodel.

In general, this metamodel can be approached in any category, but in this paper, we will use the Graph category, which has graphs as objects and graph homomorphisms as arrows. The graph component of the sketch specifies the structural dimension of the models. The nodes of this graph are typed and most often represent concepts of the model.

The sketch constraints are represented by predicate symbols that together with their signatures form the concept of diagram predicate signature [15, 16]. The predicate symbols together with the logical dependencies between them form a category that we denote by $\boldsymbol{\Pi}$.

If $\boldsymbol{\Pi}$ is a category of predicates and dependencies, then a functor $\boldsymbol{\alpha}:\boldsymbol{\Pi}\rightarrow$Graph is called a graph signature. Each object $\boldsymbol{\alpha}(P)$, $P\in\boldsymbol{\Pi}_0$, is called the shape graph arity of P. Each arrow $\boldsymbol{\alpha}(r)$, $r\in\boldsymbol{\Pi}_1$, is a morphism in the Graph category: $\boldsymbol{\alpha}(r):\boldsymbol{\alpha}(P_1)\rightarrow\boldsymbol{\alpha}(P_2)$, where $P_1,P_2\in\boldsymbol{\Pi}_0$, and is called signature substitution [12].

A diagram is a functor $d:\boldsymbol{\mathcal{P}}\rightarrow\boldsymbol{\mathcal{G}}$, where $\boldsymbol{\mathcal{P}},\boldsymbol{\mathcal{G}}\in\text{Graph}_0$. The domain $\boldsymbol{\mathcal{P}}$ of a diagram d is called a shape graph. If $\boldsymbol{\mathcal{G}}\in\text{Graph}_0$, then a $\boldsymbol{\Pi}$-formula, over $\boldsymbol{\mathcal{G}}$, is a pair (P,d), where P

is a predicate $P \in \boldsymbol{\Pi}_0$, and d is a diagram $d:\boldsymbol{\alpha}P \rightarrow \boldsymbol{\mathcal{G}}$. We will denote a $\boldsymbol{\Pi}$-formula (P,d), more simply, by P(d). We denote the set of $\boldsymbol{\Pi}$-formulas over the graph $\boldsymbol{\mathcal{G}} \in \text{Graph}_0$, by $\text{Fm}(\boldsymbol{\Pi},\boldsymbol{\mathcal{G}})$. Therefore $\text{Fm}(\boldsymbol{\Pi},\boldsymbol{\mathcal{G}})=\{P(d) \mid P \in \boldsymbol{\Pi}, d \in \text{Graph}(\boldsymbol{\alpha}(P), \boldsymbol{\mathcal{G}})\}$. The set of formulas $\text{Fm}(\boldsymbol{\Pi},\boldsymbol{\mathcal{G}})$, together with the inference relation form a category [12], If there is no confusion we will call a $\boldsymbol{\Pi}$-formula, simply a formula.

A generalized sketch over a signature $\boldsymbol{\Pi}$, is made up of a graph $\boldsymbol{\mathcal{G}} \in \text{Graph}_0$, and a subcategory of $\boldsymbol{\Pi}$-formulas $\boldsymbol{\mathcal{T}} \subseteq \text{Fm}(\boldsymbol{\Pi},\boldsymbol{\mathcal{G}})$, closed to the inference, i.e., it satisfies the condition: if $P(d) \in \boldsymbol{\mathcal{T}}$ then also $Q(d(\boldsymbol{\alpha}(r))) \in \boldsymbol{\mathcal{T}}$. Therefore, a generalized sketch is a tuple $\boldsymbol{\mathcal{S}}=(\boldsymbol{\mathcal{G}},\boldsymbol{\mathcal{T}})$, where the graph $\boldsymbol{\mathcal{S}}$, specifies the structure of a model, and the objects of the category $\boldsymbol{\mathcal{T}}$ are constraints on the models imposed by formulas.

To define an instance of a generalized sketch we will use the slice category [1]. If $\boldsymbol{\mathcal{G}}$ is an object $\boldsymbol{\mathcal{G}} \in \text{Graph}_0$, the slice category, which we denote by $\text{Graph}\downarrow\boldsymbol{\mathcal{G}}$, has as objects, pairs $(\boldsymbol{\mathcal{H}}, \tau)$, where $\boldsymbol{\mathcal{H}} \in \text{Graph}_0$ and $\tau \in \text{Graph}_1$ is a morphism $\tau:\boldsymbol{\mathcal{H}} \rightarrow \boldsymbol{\mathcal{G}}$, and as arrows between each two objects $(\boldsymbol{\mathcal{H}}_1, \tau_1)$, $(\boldsymbol{\mathcal{H}}_2, \tau_2)$, a morphism from Graph $\varphi:\boldsymbol{\mathcal{H}}_1 \rightarrow \boldsymbol{\mathcal{H}}_2$.

If $\boldsymbol{\mathcal{S}}=(\boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{T}})$, is a sketch, and $\boldsymbol{\mathcal{T}}$ is the empty category, i.e. it does not contain any constraints, then for any object $\tau:\boldsymbol{\mathcal{H}} \rightarrow \boldsymbol{\mathcal{G}}$ from the slice category $\text{Graph}\downarrow\boldsymbol{\mathcal{G}}$, the domain $\boldsymbol{\mathcal{H}}$ of τ is an instance of the sketch $\boldsymbol{\mathcal{S}}$. We denote by $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{G}})$, the set of all these instances.

To define the instances subject to certain constraints, we will use the concept of pullback. For a cospan(φ,ψ), we denote the corresponding pullback by $\text{PLB}(\varphi,\psi)$. The pullback $\text{PLB}(\varphi,\psi)$, is a span that we denote by span(φ^*,ψ^*), with the property that $\varphi \circ \varphi^* = \psi \circ \psi^*$. We also denote $\varphi^* = \text{PLB}_\psi(\varphi)$ and $\psi^* = \text{PLB}_\varphi(\psi)$.

We notice that if we have two instances $\boldsymbol{\mathcal{H}}_1, \boldsymbol{\mathcal{H}}_2 \in \boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{G}})$, corresponding to the objects $(\boldsymbol{\mathcal{H}}_1,\tau_1),(\boldsymbol{\mathcal{H}}_2,\tau_2) \in \text{ob}(\text{Graph}\downarrow\boldsymbol{\mathcal{G}})$, then $\text{dom}(\text{PLB}(\tau_1,\tau_2))=\text{dom}(\tau_1^*, \tau_2^*)$, is also an instance, $\text{dom}(\text{PLB}(\tau_1,\tau_2)) \in \boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{G}})$, because there is a morphism $\tau^* = \tau_1 \circ \tau_1^* = \tau_2 \circ \tau_2^*$.

Now we can define an instance $\boldsymbol{\mathcal{H}} \in \boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{G}})$, which satisfies a formula P(d). If $\boldsymbol{\mathcal{S}}=(\boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{T}})$, is a generalized sketch, then an instance $\boldsymbol{\mathcal{H}} \in \boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{G}})$, corresponding to the object $(\boldsymbol{\mathcal{H}},\tau)$, from the $\text{Graph}\downarrow\boldsymbol{\mathcal{G}}$ category, satisfies the formula $P(d) \in \boldsymbol{\mathcal{T}}_0$, if there is a another instance $\boldsymbol{\mathcal{H}}_1 \in \boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{G}})$, corresponding to the object $(\boldsymbol{\mathcal{H}}_1,\tau_1)$, from the $\text{Graph}\downarrow\boldsymbol{\mathcal{G}}$ category, so that $\boldsymbol{\mathcal{H}}=\text{dom}(\text{PLB}(\tau_1,d))$. We denote the set of instances that satisfy the formula P(d) by $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{G}},P(d))$.\

We can now define an instance of a sketch. If $\boldsymbol{\mathcal{S}}=(\boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{T}})$, is a generalized sketch, then an instance $\boldsymbol{\mathcal{H}} \in \boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{G}})$, is the instance of the sketch $\boldsymbol{\mathcal{S}}$, if $\boldsymbol{\mathcal{H}}$ satisfies all the formulas $P(d) \in \boldsymbol{\mathcal{T}}_0$. We denote by $\boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{S}})$ the set of all instances of the sketch $\boldsymbol{\mathcal{S}}$.

3 The static dimension of the model

In the modelling of real systems, the set S of possible worlds overlaps, most of the time, with the set of states of the model, characterized by logical formulas [13, 3, 14]. Also, in such models, the transition from one state to another is done by the actions of an agent who acts in order to achieve certain objectives specified by logical formulas that must be satisfied.

In this context, a transition system evolves through the actions of an agent who seeks to achieve some objectives expressed through logical formulas. Usually, the agent acts on the basis of a plan that involves the execution of several successive actions. When

there are several plans, which lead to the achievement of the objectives, the agent must choose the optimal plan based on a logic that characterizes it [3, 14].

Modal logics are characterized by schemes of axioms which are schemes of logical formulas, assumed to be true without demonstration, and which are the basis of reasoning. Thus an important scheme of formulas, assumed to be true in almost all modal logics, is "$\Box(\varphi\rightarrow\psi) \wedge \Box\varphi\rightarrow\Box\psi$", called axiom K, after Saul Kripke, who introduced the Kripke models [2, 9].

A series of other axioms, which were imposed in modal logic, received names such as: axiom T "$\Box\varphi\rightarrow\varphi$"; axiom B "$\varphi\rightarrow\Box\Diamond\varphi$"; axiom D "$\Box\varphi\rightarrow\Diamond\varphi$"; axiom 4 "$\Box\varphi\rightarrow\Box\Box\varphi$" and axiom 5 "$\Diamond\varphi\rightarrow\Box\Diamond\varphi$". Based on these names of the axioms, modal logics were also named over time. Similarly, many modal logics have the prefix of the name K, which expresses the satisfaction of the axiom K, followed by the names of the other axioms. For example, KT45 logic, characterized by axioms T, 4 and 5, which is also sometimes called S5 modal logic and is used to reason about knowledge.

As we saw in section 2, the evaluation of a formula, in modal logic, largely depends on the graph structure of the Kripke model, i.e. the Kripke frame. It is demonstrated that in a Kripke model M=(S,R,L), there is an implicit correspondence between the relation R and the satisfaction of the modal logic formulas [2, 14]. Thus, axiom K is satisfied in any Kripke frame, axiom T is satisfied if and only if R is reflexive, axiom B is satisfied if and only if R is symmetric, axiom D is satisfied if and only if R is serial, axiom 4 is satisfied if and only if R is transitive, axiom 5 is satisfied if and only if R is Euclidean, and so on. Therefore, if we want, for example, an agent to rationalize in KT45 logic, we will have to set the condition that the Kripke model according to which it evaluates the logical formulas is equipped with a relation R, reflexive, transitive and Euclidean.

All these restrictions on the relation R can be expressed by logical predicates. Therefore, the axioms of modal logic can be expressed by logical predicates. Since a Kripke frame involves the concepts of world and relationship, the predicates will have parameters of these types, that is, the world type, which we denote by w, and the relation type, which we denote by ρ. Also, if $u,v\in w$, and $u\rho v$, we will denote this fact with $(u,v)\in\rho$.

With these notations, the axiom T can be expressed by the predicate: $PT(u,v)=(u\in w)\rightarrow(u,u)\in\rho$.

Axiom B is satisfied if and only if the predicate: $PB(u,v)=(u\in w)\wedge(v\in w)\wedge(u,v)\in\rho\rightarrow(v,u)\in\rho$, is satisfied.

Axiom D, is satisfied if and only if the predicate: $PD(u,v)=(u\in w)\rightarrow(\exists v\in w)\wedge(u,v)\in\rho$, is satisfied.

Axiom 4 is satisfied if and only if the predicate: $P4(u,v)=(u\in w)\wedge(v\in w)\wedge(t\in w)\wedge((u,v)\in\rho)\wedge((v,t)\in\rho)\rightarrow(u,t)\in\rho$, is satisfied.

Axiom 5 is satisfied if and only if the predicate: $P5(u,v)=(u\in w)\wedge(v\in w)\wedge(t\in w)\wedge((u,v)\in\rho)\wedge((u,t)\in\rho)\rightarrow(v,t)\in\rho$, is satisfied.

In this way, we can impose other logical formulas on the Kripke frame, specific to the logic we want to impose on an agent. For example, the formula "$\Box\varphi\leftrightarrow\Diamond\varphi$" is satisfied if and only if the Kripke frame is functional, i.e. if and only if it satisfies the predicate $PF(u,v)=(u\in w\rightarrow(\exists!v\in w)\wedge(u,v)\in\rho$, and the formula "$\Box(\varphi\wedge\Box\varphi\rightarrow\psi)\vee\Box(\psi\wedge\Box\psi\rightarrow\varphi)$" is satisfied if and only if the Kripke frame is linear, i.e. if and only if it satisfies the predicate :

$PL(u,v)=(u\in w)\wedge(v\in w)\wedge(t\in w)\wedge((u,v)\in\rho)\wedge((u,t)\in\rho)\rightarrow((v,t)\in\rho)\vee(v=t)\vee((t,v)\in\rho)$.

When we want to specify a Kripke model, we will include in the set of objects of the $\boldsymbol{\Pi}$ category, all the predicates that represent formulas that we want to impose on the model. For example, if we want an agent to reason in a KTB4 logic, we will include in the set of objects of the $\boldsymbol{\Pi}$ category, the predicates PT, PB and P4. We will denote this set of objects by Π_0^i, and we will call them atomic formulas.
We then define the category $\boldsymbol{\Pi}$, inductive, as follows:

i) If $P \in \Pi_0^i$, then $P \in \boldsymbol{\Pi}_0$.
ii) If $P_1, P_2 \in \boldsymbol{\Pi}_0$ then $P=\neg(P_1)$, $P=(P_1)\wedge(P_2)$, $P=(P_1)\vee(P_2)$, $P=(P_1)\rightarrow(P_2)$, $P=(P_1)\leftrightarrow(P_2) \in \boldsymbol{\Pi}_0$, and the dependencies $P_1 \dashv P$, $P_2 \dashv P \in \boldsymbol{\Pi}_1$.
iii) If $P_1 \in \boldsymbol{\Pi}_0$, then $P= (\forall x P_1)$, $P=(\exists x P_1) \in \boldsymbol{\Pi}_0$, and the dependencies $P_1 \dashv P \in \boldsymbol{\Pi}_1$.
iv) Any object in $\boldsymbol{\Pi}_0$, and any arrow in $\boldsymbol{\Pi}_1$, are obtained by successively applying, a finite number of times, rules i), ii) and iii).

For example, in the case of KTB4 logic, the predicates: $PTB_1(u,v)=PT(u,v)\wedge PB(u,v)$, $PTB_2(u,v)=PT(u,v)\vee PB(u,v)$, $PTB4_1(u,v)=PT(u,v)\wedge PB(u,v)\wedge P4(u,v)$, will be contained in the set of objects $\boldsymbol{\Pi}_0$, and the dependencies $PT(u,v)\dashv PTB_1(u,v)$, $PB(u,v)\dashv PTB_1(u,v)$, $PT(u,v)\dashv PTB_2(u,v)$, $PB(u,v)\dashv PTB_2(u,v)$, $PT(u,v)\dashv PTB4_1(u,v)$, $PB(u,v)\dashv PTB4_1(u,v)$, $P4(u,v)\dashv PTB4_1(u,v) \in \boldsymbol{\Pi}_1$.
Since a Kripke frame has only one type of node, namely type w, and only one type of arrow, namely ρ, all the predicates defined above have as shape graph, a graph $\boldsymbol{\mathcal{P}}$, with a single node and a single arrow on which we denote, as in the case of parameters, with w and ρ respectively. Therefore, the shape graph arity application $\boldsymbol{\alpha}:\boldsymbol{\Pi}\rightarrow$Graph, is defined as $\boldsymbol{\alpha}(w)=w$ and $\boldsymbol{\alpha}(\rho)=\rho$, for all predicates in $\boldsymbol{\Pi}_0$.
Let's now build the categorical sketch corresponding to a Kripke frame. As we defined it, in section 2, a generalized sketch is a tuple $\boldsymbol{\mathcal{S}}=(\boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{T}})$, where the graph $\boldsymbol{\mathcal{S}}$, specifies the structure of a model and the category $\boldsymbol{\mathcal{T}}$, specifies the constraints imposed on the models by formulas.
In a categorical sketch, the graph $\boldsymbol{\mathcal{G}}$ has the role of classifying the concepts in a model, by specifying the types of concepts and the relationships between them. In the case of the Kripke frame model, we have only two concepts, namely; the world concept, which we denote with $\mathcal{w}$, and the relation concept, which we denote with $\boldsymbol{r}$. We also have a single diagram $d:\boldsymbol{\mathcal{P}}\rightarrow\boldsymbol{\mathcal{G}}$, defined as follows: $d(w)=\mathcal{w}$ and $d(\rho)=\boldsymbol{r}$.
In this context, the objects of category $\boldsymbol{\mathcal{T}}$ are generated by subsets of the set: {PT(d), PB(d), PD(d), P4(d), P5(d), PF(d), PL(d)}, in accordance with the definition of category $\boldsymbol{\Pi}$, from above. Of course, this set can be expanded with other formulas specific to the concrete model we want to build. Notice that the set of objects $\boldsymbol{\mathcal{T}}_0$ contains all well-formed formulas based on the constraints imposed on the model by the set $\boldsymbol{\Pi}_0$ of predicates. Therefore, there is a formula $P(d) \in \boldsymbol{\mathcal{T}}_0$, so that the constraints of the specific Kripke frame model can only be imposed by satisfying the formula P(d).
For example, if we want the relation R, of the model to be an equivalence relation, it is enough to impose the satisfaction of the formula $P(d)=PT(d)\wedge PB(d)\wedge P4(d)$, i.e. to be reflexive, symmetric and transitive.
But an instance $\boldsymbol{\mathcal{H}} \in \boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{G}})$, corresponding to the object $(\boldsymbol{\mathcal{H}},\tau)$, from the category Graph$\downarrow\boldsymbol{\mathcal{G}}$, satisfies the formula $P(d) \in \boldsymbol{\mathcal{T}}_0$, if there is another instance $\boldsymbol{\mathcal{H}}_1 \in \boldsymbol{\mathcal{I}}(\boldsymbol{\mathcal{G}})$, corresponding to the object $(\boldsymbol{\mathcal{H}}_1, \tau_1)$, from the Graph$\downarrow\boldsymbol{\mathcal{G}}$ category, so that $\boldsymbol{\mathcal{H}}=dom(PLB(\tau_1,d))$.

Next, we will show that any instance of the sketch is represented by a fixed point of an endofunctor of the Graph↓$\mathcal{G}$ category. For this we define an endofunctor Φ:Graph↓$\mathcal{G}$→Graph↓$\mathcal{G}$, like this: for each object $(\mathcal{H},\tau)\in$Graph↓$\mathcal{G}$, $\Phi((\mathcal{H},\tau))=$ (dom(PLB(τ,d), $\tau\circ\tau^*$), and the arrows between two objects $\Phi((\mathcal{H}_1, \tau_1)$ and $\Phi(\mathcal{H}_2, \tau_2)$ are the arrows from the Graph↓$\mathcal{G}$ category, between the objects dom(PLB(τ_1,d)), dom(PLB(τ_2,d)) $\in$ob(Graph↓$\mathcal{G}$).
Since the pullback never adds new components to a graph, but selects all the components that respect the constraints imposed on the model, it follows that if $\mathcal{H}$ is a model of the sketch, represented by the object $(\mathcal{H},\tau)\in$Graph↓$\mathcal{G}$, then $\Phi((\mathcal{H},\tau))=(\mathcal{H},\tau)$ that is, $\mathcal{H}$ is represented by a fixed point of the endofunctor Φ. Obviously, if the graph $\mathcal{H}$, has components that do not satisfy the constraints P(d), these components will be eliminated and therefore $(\mathcal{H}, \tau)$ is not a fixed point for the endofunctor Φ.
In the case of systems modelling, when the set of possible worlds overlaps with the set of states, the set of states is given by the possible evolutions of the system and therefore the set of possible worlds is fixed. Also, the set of transitions forms a minimal relation on the set of possible worlds that must be included in the relation of any instance of the Kripke frame of the model. From here it follows that any instance of the Kripke frame model does nothing but extend the relation R to a minimal relation that respects the constraints. We also note that the graph of the sketch $\mathcal{G}$ is a terminal object in the Graph category, from which it follows that each instance, of the sketch, is represented by a single object $(\mathcal{H},\tau)$, from the Graph↓$\mathcal{G}$ category. These observations lead us to the conclusion that, in the case of systems modelling, we can determine a minimal relationship on the set of system states that satisfies the constraints imposed on the model.

4 The reactive dimension of the model

When there are several plans, which lead to the achievement of the objectives, the agent must choose the optimal plan based on some values [3]. Therefore, the agent must be able to carry out reasoning in various logics, to fulfil the objectives, depending on the values it is pursuing.
The process of tracking the achievement of some objectives, through different types of reasoning, involves the dynamic adaptation of the Kripke frame in which the logical formula is evaluated. A Kripke model is reactive if the model instance changes as it is traversed to evaluate a formula. This means that the constraints applied to an instance change during the evaluation of a logical formula.
We consider a transition system that evolves through the actions of an agent. The actions of the agent are decided according to the realization of some logical formulas. Thus, if we have a finite set of propositional variables P={p,q,...}, a transition system is a construct of the form TS=(S,A,T,L), where S is the set of states, A is the set of actions that the agent can perform, T is a set of transitions, t$\in$S×A×S, between states by executing some actions, and L is an application L:S→2^P, which selects the propositions in P, which are satisfied in each state s$\in$S.
In this transition system, in each state, the choice of an action by the agent is conditioned by the realization of a formula from the modal logic. The evaluation of these formulas is done in a Kripke frame in which the set of possible worlds overlaps with the set S of states, and the accessibility relation is given by the transition relation.

In a classic Kripke model, the accessibility relationship does not change in the evaluation process, but in the reactive model, this relationship adapts dynamically, depending on the mode of reasoning that the agent will adopt in the respective context.

It is obvious that the set of possible worlds will overlap with the set of possible states of the transition system also in the reactive case and only the accessibility relation will be adapted. Also, the accessibility relation defined by the transitions of the system will be included in all instances of the reactive Kripke model, because the imposition of additional constraints involved the expansion of this relation. We will denote by $F_0=(\mathcal{W},\mathcal{R}_0)$, the Kripke frame defined by a transition system as follows: $\mathcal{W}$=S and for each pair u,v$\in\mathcal{W}$, we make $\mathcal{R}_0$(u,v)=true if there is a transition (u,a,v)$\in$T, a$\in$A and $\mathcal{R}_0$(u,v)=false, otherwise. Obviously, $F_0\in\in\mathcal{I}(\mathcal{G})$, where $\mathcal{G}$, is the graph of the sketch $\mathcal{S}$. We will call the relation $\mathcal{R}_0$, the initial relation of the Kripke frame instance.

Example 4.1. If we consider the transition system from Fig. 1, where P={p,q}, then the initial instance F_0, is defined as follows (Fig. 2): $\mathcal{W}=\{S_1,S_2,S_3,S_4\}$,

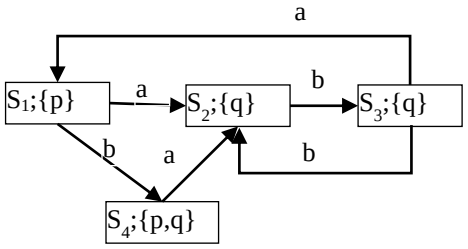


Fig. 1. Example, transition system

$\mathcal{R}_0(S_1,S_2)$=true;

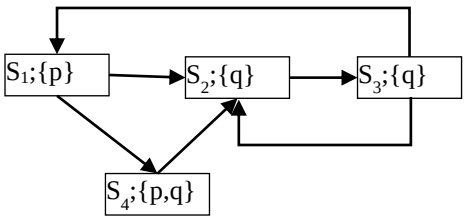


Fig. 2 Example, the initial instance

$\mathcal{R}_0(S_1,S_4)$=true; $\mathcal{R}_0(S_2,S_3)$=true; $\mathcal{R}_0(S_3,S_1)$=true; $\mathcal{R}_0(S_3,S_2)$=true; $\mathcal{R}_0(S_4,S_2)$=true and otherwise $\mathcal{R}_0(S_i,S_j)$=false.

Let's evaluate the formula $S_1|=\Box^2q$, in $F_0=(\mathcal{W},\mathcal{R}_0)$. In order for this formula to be satisfied in S_1, the subformula $\Box q$ will have to be satisfied in all accessible states from S_1, that is, the formulas: $S_2|=\Box q$, $S_4|=\Box q$ must be satisfied. In order for $S_2|=\Box q$ to be satisfied, q must be satisfied in state S_3, which is true, and for $S_4|=\Box q$ to be satisfied, q must be satisfied in S_2, which is true. Therefore, F_0 satisfies the formula $S_1|=\Box^2q$.

Let us now consider a set of predicates $P_1,P_2,\ldots P_n\in\Pi_0$, which specify constraints on a Kripke frame $F_0=(\mathcal{W},\mathcal{R}_0)$. Then for each predicate P_i, we have an instance $F_i=(\mathcal{W},\mathcal{R}_i)$, which satisfies the constraint imposed by this predicate. As we saw, in section 3, each instance $\mathcal{R}_i$ is represented by a fixed point of the functor Φ:Graph$\downarrow\mathcal{G}\rightarrow$Graph$\downarrow\mathcal{G}$.

In the Reactive Kripke frame model, we need to impose restrictions specified by a predicate P_i, i=1,n, or by a conjunction of such predicates, when we want to impose several such constraints simultaneously.

It is easy to prove that if the relation $\boldsymbol{\mathcal{R}}_i$, imposes the constraint P_i, and the relation $\boldsymbol{\mathcal{R}}_j$, imposes the constraint P_j, then the relation $\boldsymbol{\mathcal{R}}_{i,j}= \boldsymbol{\mathcal{R}}_i \vee \boldsymbol{\mathcal{R}}_j$, imposes the condition $P_i \wedge P_j$. Therefore, if we know the relations $\boldsymbol{\mathcal{R}}_i$, i=1,n, we can easily build any relation that is a disjunction of known relations. We will denote by $\boldsymbol{\mathcal{R}}$, the set of all possible relations in a Kripke reagent.
Therefore, the set $\boldsymbol{\mathcal{R}}$, of all possible relations in a reactive Kripke frame, is defined recursively as follows:
i) $\boldsymbol{\mathcal{R}}_0 \in \boldsymbol{\mathcal{R}}$ and if $P_i \in \boldsymbol{\Pi}_0$, then $\boldsymbol{\mathcal{R}}_i \in \boldsymbol{\mathcal{R}}$;
ii) If $\boldsymbol{\mathcal{R}}_i, \boldsymbol{\mathcal{R}}_j \in \boldsymbol{\mathcal{R}}$ then $\boldsymbol{\mathcal{R}}_i \vee \boldsymbol{\mathcal{R}}_j \in \boldsymbol{\mathcal{R}}$.

At each step of evaluating a modal formula, one of the relations from the set of relations $\boldsymbol{\mathcal{R}}$ will be selected, depending on the path travelled to the current world. The set of possible paths is a language on the set of possible worlds $\boldsymbol{\mathcal{L}} \subseteq \boldsymbol{\mathcal{W}}^*$.
Therefore, a Kripke frame instance is a construct of the form $F_R = (\boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{L}}, \Psi)$, where $\boldsymbol{\mathcal{W}}$ is the set of possible worlds, $\boldsymbol{\mathcal{R}}$ is the set of possible relations, $\boldsymbol{\mathcal{L}}$ is a language on the set $\boldsymbol{\mathcal{W}}$, and $\Psi: \boldsymbol{\mathcal{L}} \rightarrow \boldsymbol{\mathcal{R}}$ is an application that selects the appropriate relationship for each possible path. In this context, if we denote by δ the path through which the world s was reached, the evaluation of the modal operators will be done as follows:
„$s \Vdash \Box_\delta \psi$ iff, $\forall\ s_1 \in \boldsymbol{\mathcal{W}}$ with $\Psi(\delta)(s, s_1)$, it results $s_1 \Vdash \psi$;
$s \Vdash \Diamond_\delta \psi$ iff $\exists\ s_1 \in \boldsymbol{\mathcal{W}}$ with $\Psi(\delta)(s, s_1)$ and $s_1 \Vdash \psi$".

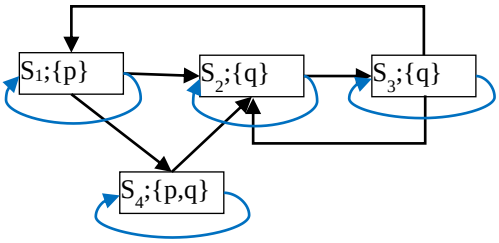


Fig. 3. Example, reflexive instance

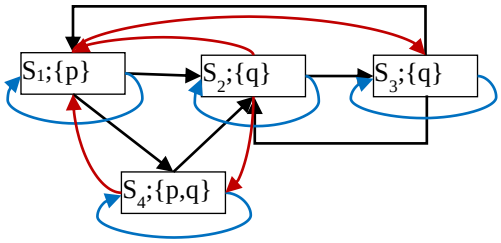


Fig. 4. Example, symmetrical instance

Example 4.2. If in example 4.1, we impose the restriction specified by the PT(u,v) predicate, then the resulting relation, which we denote by $\boldsymbol{\mathcal{R}}_1$, becomes a reflexive relation (Fig.3): $\boldsymbol{\mathcal{R}}_1(S_1,S_2)$=true; $\boldsymbol{\mathcal{R}}_1(S_1,S_4)$=true; $\boldsymbol{\mathcal{R}}_1(S_2,S_3)$=true; $\boldsymbol{\mathcal{R}}_1(S_3,S_1)$=true; $\boldsymbol{\mathcal{R}}_1(S_3,S_2)$=true; $\boldsymbol{\mathcal{R}}_1(S_4,S_2)$=true; $\boldsymbol{\mathcal{R}}_1(S_1,S_1)$=true; $\boldsymbol{\mathcal{R}}_1(S_2,S_2)$=true; $\boldsymbol{\mathcal{R}}_1(S_3,S_3)$=true; $\boldsymbol{\mathcal{R}}_1(S_4,S_4)$=true; and otherwise $\boldsymbol{\mathcal{R}}_1(S_i,S_j)$=false.
The relation corresponding to the predicate PB(u,v), which we denote by $\boldsymbol{\mathcal{R}}_2$, becomes a symmetrical relation (Fig.4): $\boldsymbol{\mathcal{R}}_2(S_1,S_2)$=true; $\boldsymbol{\mathcal{R}}_2(S_1,S_4)$=true; $\boldsymbol{\mathcal{R}}_2(S_2,S_3)$=true; $\boldsymbol{\mathcal{R}}_2(S_3,S_1)$=true; $\boldsymbol{\mathcal{R}}_2(S_3,S_2)$=true; $\boldsymbol{\mathcal{R}}_2(S_4,S_2)$=true; $\boldsymbol{\mathcal{R}}_2(S_1,S_1)$=true; $\boldsymbol{\mathcal{R}}_2(S_2,S_2)$=true; $\boldsymbol{\mathcal{R}}_2(S_3,S_3)$=true; $\boldsymbol{\mathcal{R}}_2(S_4,S_4)$=true; $\boldsymbol{\mathcal{R}}_2(S_2,S_1)$=true; $\boldsymbol{\mathcal{R}}_2(S_1,S_3)$=true; $\boldsymbol{\mathcal{R}}_2(S_4,S_1)$=true; $\boldsymbol{\mathcal{R}}_2(S_2,S_4)$=true; and otherwise $\boldsymbol{\mathcal{R}}_2(S_i,S_j)$=false.
By imposing the constraints specified by the predicates PT(u,v) and PB(u,v), together we obtain a symmetric and transitive relation $\boldsymbol{\mathcal{R}}_{1,2} = \boldsymbol{\mathcal{R}}_1 \vee \boldsymbol{\mathcal{R}}_2$, and therefore $\boldsymbol{\mathcal{R}}_= \{\boldsymbol{\mathcal{R}}_0, \boldsymbol{\mathcal{R}}_1, \boldsymbol{\mathcal{R}}_2, \boldsymbol{\mathcal{R}}_{1,2}\}$. Let's define the application Ψ, as follows: $\Psi(S_1)= \boldsymbol{\mathcal{R}}_0$; $\Psi(S_1S_2)= \Psi(S_1S_1)= \boldsymbol{\mathcal{R}}_1$; $\Psi(S_1S_4)= \boldsymbol{\mathcal{R}}_2$; $\Psi(\delta)= \boldsymbol{\mathcal{R}}_{1,2}$ otherwise.
We notice that if we evaluate the formula $S_1 \models \Box^2 q$, using the new Kripke frame $F_R=(\boldsymbol{\mathcal{W}}, \boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{L}}, \Psi)$, it is no longer satisfied, because reflexivity requires that the sentence q should also be satisfied in the state S_1.

5 Conclusions

The most important conclusion is that category theory offers all the necessary mechanisms for the specification and analysis of diagrammatic models, including classical or reactive Kripke models. We can find, from section 3, that based on some atomic predicates that impose constraints on Kripke structures, we can then easily construct any constraint, formulated in terms of these atomic constraints.
In the case of systems modeling, the set of possible worlds overlaps with the set of possible states, and therefore the set of possible worlds is also determined and the set of transitions forms a minimal relation on the set of possible worlds that must be included in the relation of any instance of the Kripke frame of the model.
Any instance of the Kripke frame model, extends the relationship given by the evolution of the system, to a minimal relationship that respects the constraints. These observations lead us to the conclusion that, in the case of systems modeling, we can determine a minimal relationship on the set of system states that satisfies the constraints imposed on the model. This subject will be treated in a future work where we intend to provide algorithms for the effective calculation of reactive Kripke frames for semantic models expressed in terms of transition systems.

References

[1] Michael Barr, Charles Wells, 2012. Category Theory For Computing Science, Reprints in Theory and Applications of Categories, No. 22.
[2] M. Huth, M. Ryan, 2004, Logic in Computer Science, Modelling and Reasoning about Systems, Published in the United States of America by Cambridge University Press, New York.
[3] Luo, Jieting; Liao, Beishui; Gabbay, Dov. 2022, Value-based Practical Reasoning: Modal Logic + Argumentation (In Press). In: COMMA - Computational Models of Argument. Conference Proceedings.
[4] D. Gabbay. 2004, Reactive Kripke semantics. In Proceedings of CompLog 2004, W. Carnielli, ed., pp. 7–20. Centre of Logic and Computation, University of Lisbon.
[5] H. Barringer and D. M. Gabbay. 2010, Modal and temporal argumentation networks. In Time for Verification. Essays in Memory of Amir Pnueli, D. Peled and Z. Manna, eds., pages 1–25. LNCS 6200, Springer, Berlin.
[6] M. Crochemore and D. M. Gabbay. 2011, Reactive Automata. Information and Computation, 209(4), 692–704. Published online: DOI: 10.1016/j.ic.2011.01.002.
[7] S. Modgil. 2009, Reasoning about preferences in argumentation frameworks. Artif. Intell., 173:901–934.
[8] Gabbay, D.M. 2013, Reactivity and Grammars: An Exploration. In: Reactive Kripke Semantics. Cognitive Technologies. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-41389-6_9.
[9] Olivier Gasquet , Andreas Herzig , Bilal Said ,François Schwarzentruber, 2014, Kripke's Worlds, An Introduction to Modal Logics via Tableaux, Springer Basel AG.
[10] Barringer, H., Rydeheard, D., Gabbay, D. 2014. Reactivity and Grammars: An Exploration. In: Dershowitz, N., Nissan, E. (eds) Language, Culture, Computation. Computing - Theory and Technology. Lecture Notes in Computer Science, vol 8001. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-45321-2_6
[11] Gabbay, D.M. 2008. Introducing Reactive Kripke Semantics and Arc Accessibility. In: Avron, A., Dershowitz, N., Rabinovich, A. (eds) Pillars of Computer Science. Lecture Notes in Computer Science, vol 4800. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-78127-1_17

[12] Zinovy Diskin, Uwe Wolter, 2008, A Diagrammatic Logic for Object-Oriented Visual Modeling, Electronic Notes in Theoretical Computer Science, Volume 203, Issue 6, 21 November 2008.
[13] Virginia Dignum, 2009, A Logic for Agent Organizations, Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models, Information Science Reference, by IGI Global.
[14] Y. Shoham, K. Leyton-Brown, 2009, Multiagent systems_ algorithmic, game-theoretic, and logical foundations -Cambridge University Press.
[15] Uwe Wolter, Zinovy Diskin, 2015, The Next Hundred Diagrammatic Specification Techniques, A Gentle Introduction to Generalized Sketches, 02 September 2015 : https://www.researchgate.net/publication /253963677.
[16] D.C. Crăciunean, D. Karagiannis, 2019, A categorical model of process cosimulation, Journal of Advanced Computer Science and Applications(IJACSA), 10(2).

DOI: 10.2478/ijasitels-2023-0002

Simulink implementation of the open-loop scalar command for a three-phase induction machine model

Gabriela Crăciunaş[1], *Alina Cristina Viorel*[1]

[1]*Computer Science and Electrical Engineering Department, "Lucian Blaga" University of Sibiu, Romania, E-Mail: gabriela.craciunas@ulbsibiu.ro; alina.viorel@ulbsibiu.ro*

Abstract

This paper describes step by step the modelling and simulation of the open loop scalar command for a three-phase induction machine with known parameters, which operates in motor regime. In order to change the electromagnetic torque and implicitly the rotor speed, in this paper the solution of applying a positive mechanical torque from outside the machine was chosen, without dynamically adjusting the supply voltage. The chosen mathematical model for the induction machine is the one described by Krause's equations, written in a stationary reference frame. The simulations were done on the Matlab/Simulink simulation environment and the behaviour of the machine was monitored under the working conditions listed above.

Keywords: three-phase induction machine, open-loop command, Simulink model

1 Introduction

The induction machine is an essential component in an electric drive, playing a crucial role in a variety of industrial, commercial, and residential applications. It is used in various fields due to its efficiency, reliability, and versatility. Constructively, they are similar to classic asynchronous motors, designed to operate at constant speed. What differs is the field of industrial applications, in which the machine is used as an electromechanical conversion element with adjustable speed, in which case the performance of the machine in dynamic mode is of particular interest, [1].

This paper describes the modelling and simulation of the open-loop scalar command for a three-phase induction machine operating in motor mode. The mathematical model is implemented on specialized electrical and electromechanical systems simulation software. One of the common tools used for this purpose is MATLAB/Simulink [2], but there are others such as PSCAD, EMTDC, or dedicated software for simulating electrical circuits and control systems.

Control is achieved by applying a constant voltage to the induction machine, without dynamically adjusting this voltage according to its operating conditions. In order to change the electromagnetic torque and implicitly the speed of the rotor, in this paper the solution of applying a mechanical torque from outside the machine was chosen. This type of control may be simpler to implement, but it does not provide the same precision and efficiency as a closed-loop command that uses feedback to adjust

parameters in real time. However, this command is preferred where stationary speeds are used for long periods of time or in undemanding drive systems, [3], [4].

2 Mathematical model of the three-phase induction machine

In this paper, the mathematical model of the three-phase induction machine chosen for implementation on the Matlab/Simulink simulation environment represents one of the multiple variants that can be found in the specialized literature. Thus, the Krause model was chosen for the equivalent circuit, [1],
Stator and rotor voltage equations in arbitrary reference frame are,

$$U_{ds} = R_s i_{ds} + p\lambda_{ds} - \omega_e \lambda_{qs} \tag{1}$$

$$U_{qs} = R_s i_{qs} + p\lambda_{qs} + \omega_e \lambda_{ds} \tag{2}$$

$$U_{os} = R_s i_{os} + p\lambda_{os} \tag{3}$$

$$U_{dr} = R_r i_{dr} + p\lambda_{dr} - (\omega_e - \omega_r)\lambda_{qr} \tag{4}$$

$$U_{qr} = R_r i_{qr} + p\lambda_{qr} + (\omega_e - \omega_r)\lambda_{dr} \tag{5}$$

$$U_{or} = R_s i_{or} + p\lambda_{or} \tag{6}$$

where, $U_{ds}, I_{ds}, \lambda_{ds}$ - d-axis components,

$U_{qs}, I_{qs}, \lambda_{qs}$ - q-axis components,

$U_{os}, I_{os}, \lambda_{os}$ - 0- axis and usually represent the unbalances in the system. In case of balanced voltages, the zero-axis currents, voltages and flux are zero under normal operating conditions.
Since, in electrical systems, certain parameters are represented by the unit of measure ohms, electric fluxes and inductances will be replaced by flux linkages per second and leakage reactance, thus,

$$\Psi = \omega_b \lambda \text{ and } X = \omega_b L \tag{7}$$

where, $\omega_b = 2\pi f_{rated}$ - motor angular electrical base frequency,

f_{rated} - the rated frequency.

In Figure 1, the d-q equivalent circuit for the induction machine is represented, and the equations (1) - (6) written in the stationary reference frame for a symmetrical three-phase induction machine are modified as follows,

$$\frac{d\Psi_{ds}}{dt} = \omega_b \left[U_{ds} + \frac{R_s}{x_{ls}} (\Psi_{md} - \Psi_{ds}) \right] \tag{8}$$

$$\frac{d\Psi_{qs}}{dt} = \omega_b \left[U_{qs} + \frac{R_s}{x_{ls}} (\Psi_{mq} - \Psi_{qs}) \right] \tag{9}$$

$$\frac{di_{os}}{dt} = \frac{\omega_b}{x_{ls}}\left[U_{os} - R_s i_{os}\right] \tag{10}$$

$$\frac{d\Psi_{dr}}{dt} = \omega_b\left[U_{dr} - \frac{\omega_r}{\omega_b}\Psi_{qr} + \frac{R_r}{x_{lr}}\left(\Psi_{md} - \Psi_{dr}\right)\right] \tag{11}$$

$$\frac{d\Psi_{qr}}{dt} = \omega_b\left[U_{qr} + \frac{\omega_r}{\omega_b}\Psi_{dr} + \frac{R_r}{x_{lr}}\left(\Psi_{mq} - \Psi_{qr}\right)\right] \tag{12}$$

$$\frac{di_{or}}{dt} = \frac{\omega_b}{x_{lr}}\left[U_{or} - R_r i_{or}\right] \tag{13}$$

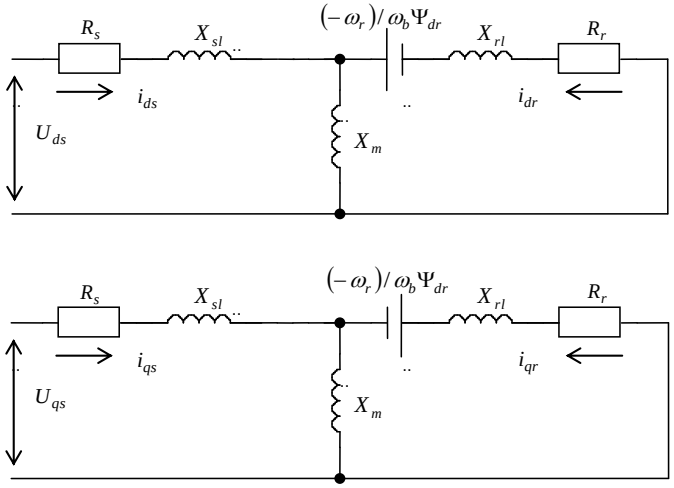


Figure 1. (d–q) equivalent circuit of the induction machine in stationary reference frame

The expressions of flux linkages and electrical currents from equations (8)-(13) are,

$$\Psi_{ds} = x_{ls} i_{ds} + \Psi_{md}, \quad i_{ds} = \frac{1}{x_{ls}}\left(\Psi_{ds} - \Psi_{md}\right) \tag{14}$$

$$\Psi_{qs} = x_{ls} i_{qs} + \Psi_{mq}, \quad i_{qs} = \frac{1}{x_{ls}}\left(\Psi_{qs} - \Psi_{mq}\right) \tag{15}$$

$$\Psi_{dr} = x_{lr} i_{dr} + \Psi_{md}, \quad i_{dr} = \frac{1}{x_{lr}}\left(\Psi_{dr} - \Psi_{md}\right) \tag{16}$$

$$\Psi_{qr} = x_{lr} i_{qr} + \Psi_{mq}, \quad i_{qr} = \frac{1}{x_{lr}}\left(\Psi_{qr} - \Psi_{mq}\right) \tag{17}$$

$$\Psi_{md} = x_M\left[\frac{\Psi_{ds}}{x_{ls}} + \frac{\Psi_{dr}}{x_{lr}}\right], \quad \Psi_{mq} = x_M\left[\frac{\Psi_{qs}}{x_{ls}} + \frac{\Psi_{qr}}{x_{lr}}\right] \tag{18}$$

$$x_M = \frac{1}{\frac{1}{x_m} + \frac{1}{x_{ls}} + \frac{1}{x_{lr}}} \tag{19}$$

where, Ψ_{ds}, Ψ_{qs} - d and q flux linkages,
Ψ_{md}, Ψ_{mq} - d and q magnetizing flux linkages,
x_{ls}, x_{lr} - stator and rotor leakage reactance,
ω_r - stator angular electrical frequency,
ω_r - rotor angular electrical speed.

The electromagnetic torque equation has the form,

$$T_e = \frac{3}{2}\frac{p}{2\omega_b}\left(\Psi_{ds} i_{qs} - \Psi_{qs} i_{ds}\right) \tag{20}$$

where, p – number of pole.
From the fundamental equation of the machine's motion, we get the rotor angular electrical speed, [4],

$$\omega_b \frac{d(\omega_r / \omega_b)}{dt} = \frac{P}{2J}\left(T_e + T_{mec} - T_L\right) \tag{21}$$

where, J - moment of inertia,
T_{mec} - mechanical torque,
T_L - load torque.
Another expression of the above equation, replacing J-moment of inertia with H-inertia constant,

$$H = \frac{J\omega_b^2}{2S_b} \tag{22}$$

where S_b - power in VA.

3 Simulink implementation

The implementation of the mathematical model will be done on the Matlab/Simulink simulation environment, [2], [5]. The machine used in the simulation is a three-phase induction machine with a short-circuited rotor, with known parameters, Table 1. The three-phase system of electric voltages and currents are transformed into a two-phase d-q system, and the general equations of the machine are adapted to the stationary reference frame, [1].

Table 1. Machine parameters

Parameter	Value
P	2
J	2,8 [kgm^2]
f_{rated}	50 [Hz]
R_s	0,206[Ω]
R_r	0,086[Ω]
x_{ls}	0,178[Ω]
x_{lr}	0,178[Ω]
x_m	4,861[Ω]

Fig. 2 shows the block diagram, where both input and output quantities are highlighted. Thus, the induction machine is powered by a three-phase voltage system, at nominal frequency and a mechanical torque act on it which is applied in the same direction of rotation as the rotor rotation. Also, it was considered that the neutral of the stator voltage system varies, and this shortcoming was solved by introducing a small capacity capacitor, between points s and g, Fig.3. As output quantities we have electromagnetic torque and rotor speed.

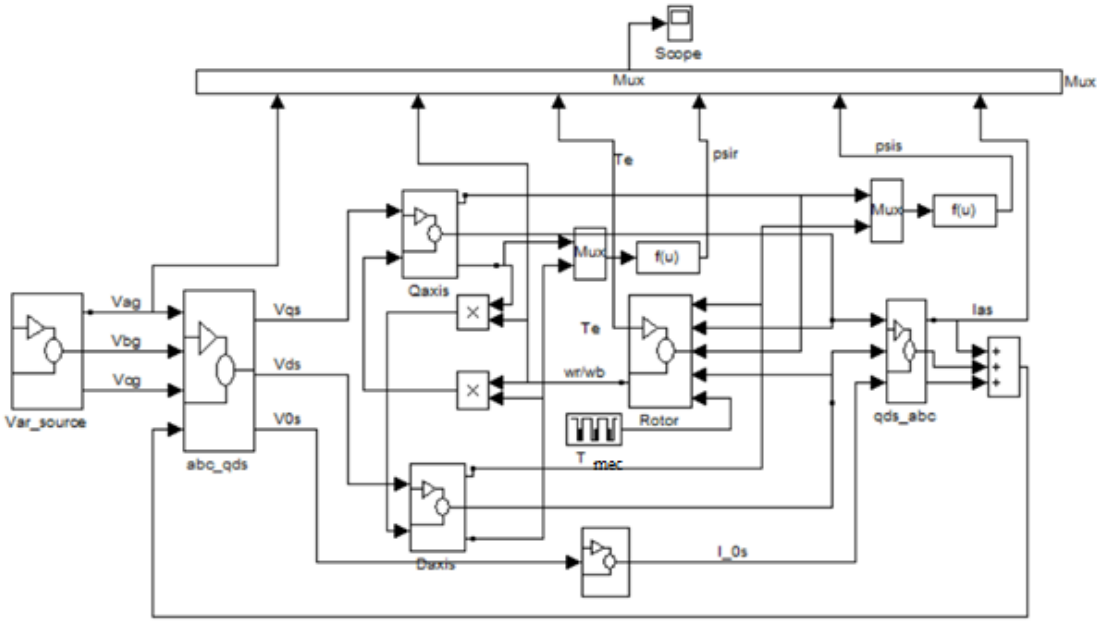


Figure 2. Block diagram of the studied model

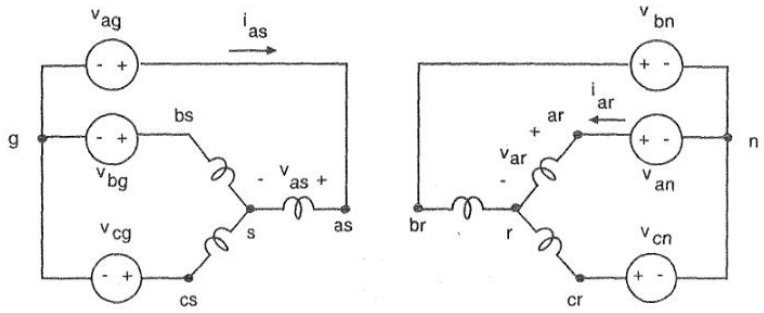


Figure 3. Connecting the machine's armatures

The block diagram in Fig. 2 contains four main blocks, as follows: the three voltages that supply the stator of the machine are determined, the transformation of the three-phase system into a two-phase d-q0 system in the stationary reference frame and vice versa, using the Clark and Park transformation equations and the block that models the induction machine in d-q stationary reference frame. Also, the block diagram includes a module that makes it possible to determine and visualize on the oscilloscope the real stator electrical currents.

The unit vectors $\cos\theta_e$ and $\sin\theta_e$ are calculated in the block in Fig. 4, using the integration formula,

$\theta_e = \int \omega_e dt$, where $\omega_e = 2\pi f_{rated}\ [rad/\sec]$ and $f_{rated} = 50Hz$.

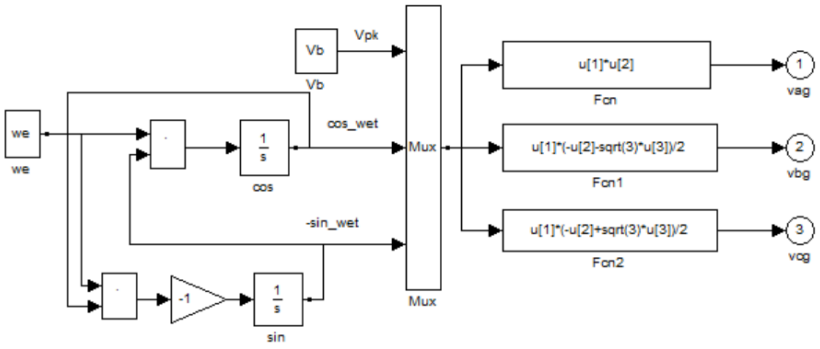


Figure 4. Frequency generator

Thus, the three-phase system of voltages is obtained, applied to the machine's stator, at the nominal frequency, which will later be converted into a two-phase stationary frame. In Fig. 5 is presented a block diagram where the calculation option of the fluxes and currents in the stator and rotor on the d axis was chosen, according to equations (8)-(13). Modeling and simulating the same quantities on the q axis is done in the same way.

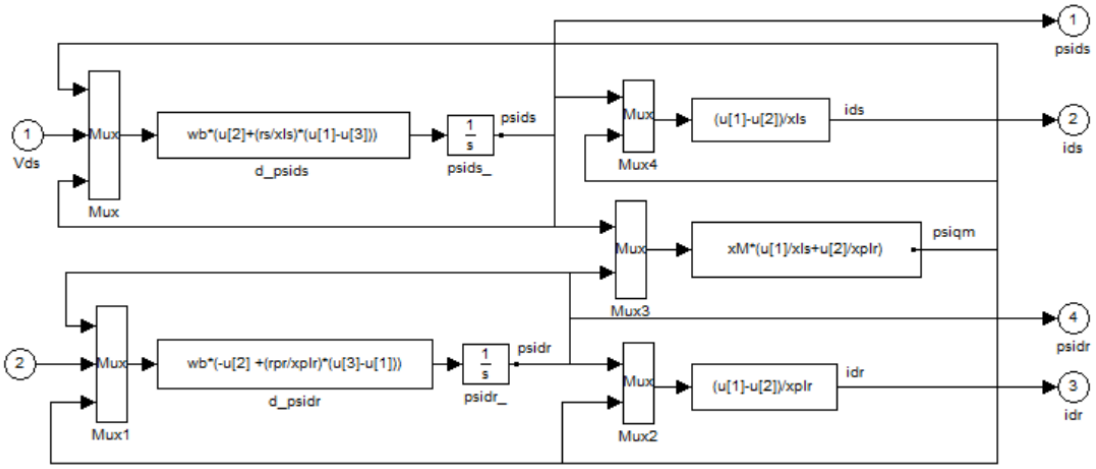


Figure 5. Block diagram d-axis elements

Once the fluxes and currents on the two axes d-q are calculated, according to equations (20)-(22), the electromagnetic torque and the rotor speed can be implemented and determined, Fig.6.

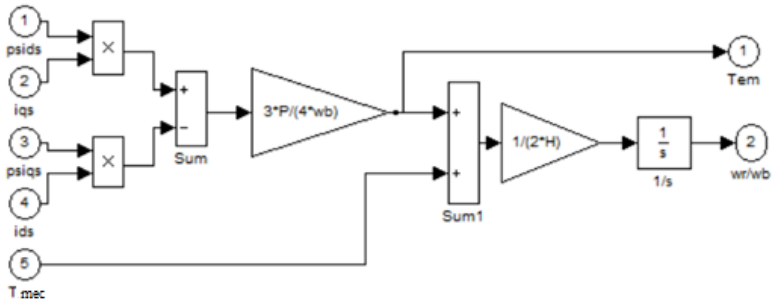


Figure 6. The fundamental equation of motion

The main block diagram, Fig. 2 is therefore built from modules whose sizes can be easily monitored by the Simulink tool „Scope", [6].

4 Simulation results

Once the component modules of the general block diagram are completed, the simulation on the Simulink simulation environment is initiated. The values of all electrical and mechanical parameters were established, considered for the program as input data. The system variables can also be monitored on the oscilloscope or are variables in the calculation of other quantities.
In this paper, we wanted to highlight the variation of the electromagnetic torque, respectively the rotor speed of by applying a mechanical torque from the outside, in the sense of the rotor rotation. Thus, it was opted for this method and not for the classical one, of varying the electric voltage supplying the induction machine.

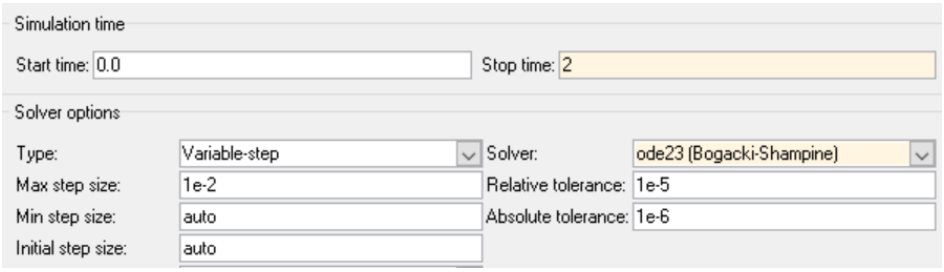
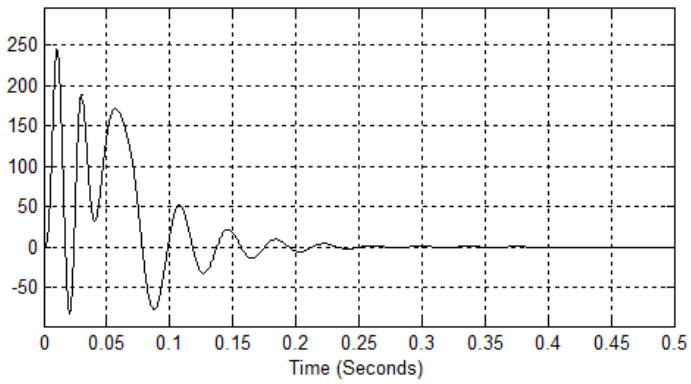
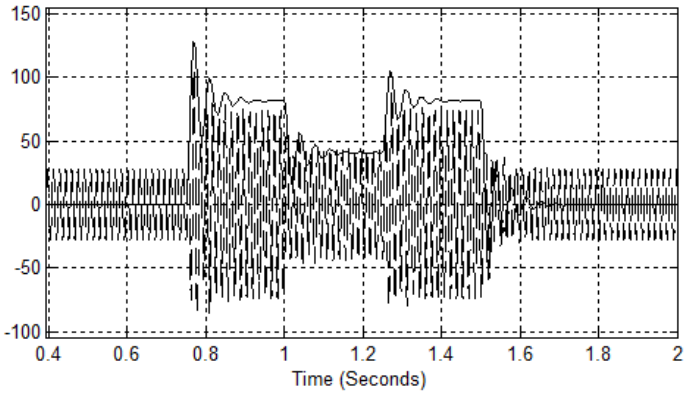


Figure 7. Simulink parameters configuration

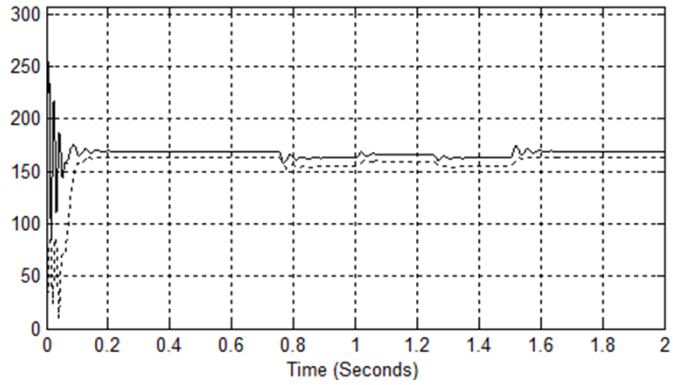
In Fig. 7, you can view the parameters that have been configured for Simulink, for the optimal functioning of the model proposed in this paper. Depending on the variable visualized on the oscilloscope, the simulation time will be adjusted. Thus, in Fig. 8 (a) shows the variation of the electromagnetic torque over time, where the mechanical torque is zero. If a positive mechanical torque is also applied, then it can be seen in Fig. 8 (b) how the two quantities are directly proportional. From the diagram, it can be seen how, when the mechanical torque stops, the waveform of the currents stabilizes at the steady-state value. In Fig. 8 (c) the steady-state values of the two flux linkages can be monitored, Ψ_s ___ and Ψ_r..... It can be observed the very small influence on the magnetic quantities when applying the mechanical torque, the machine operating stable.



(a) $T_e(t)$



(b) T_e __(t); i_s...(t)



(c) Ψ_s __(t); Ψ_r...(t)

Figure 8. Simulation results

Visualization of the rotor speed variation as a function of time is done on the graph in Fig. 9. In fact, what is observed is a ratio between the rotor speed ω_r and motor angular electrical base frequency ω_b. The unitary value of the variation of the speed ratio is over the entire simulation interval, the small distortions being noticed only when the mechanical torque appears.

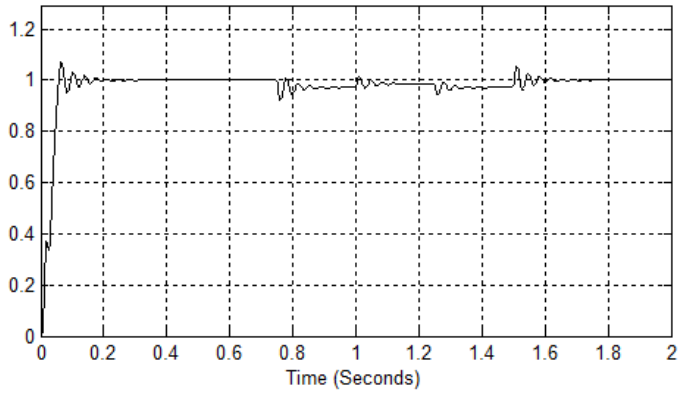


Figure 9. Characteristic $\omega_r/\omega_b(t)$

5 Conclusions

In this paper, the model of the three-phase induction machine written in the stationary reference frame was implemented, using the Matlab/Simulink simulation environment. The realization of a scalar command in the open loop by applying a positive mechanical torque and the behavior of the machine in these operating conditions was pursued. Analyzing each graph separately, Fig. 8 and Fig. 9, it can be concluded that the operation of the three-phase induction machine is stable, the variations being very small. Also, to create the mathematical model of the machine, implemented using Krause equations, the electric inductances and magnetic fluxes were modified, equations (8)-(13). The results of the simulations could be seen on the virtual oscilloscope of Simulink.

References

[1] P. C. Krause, *Analysis of Electric Machinery*. IEEE Press, ISBN 0-471-14326-X, 1995.
[2] Karris S.T., *Introduction to Simulink with Engineering Applications*, Orchard Publications, ISBN 0-9744239-8-X, 2006
[3] De Doncker R.W., Duco W.J., Veltman A., *Advanced Electrical Drives. Analysis, Modeling, Control*, Springer, ISBN 978-94-007-0179-3, 2011
[4] Ong C., M., *Dynamic Simulation of Electric Machinery Using Matlab Simulink*, Prentice Hall, ISBN 0-13-723785-5, 1997.
[5] Chiasson J., *Modeling and High-Performance Control of Electric Machines*, IEEE Press Series on Power Engineering, New York, ISBN 0-47 1-68449-X, 2005.
[6] Ozpineci B., Tolbert L.M., *Simulink Implementation of Induction Machine Model - a modular approach*, International Electric Machines and Drives Conference, USA, ISBN:0-7803-7817-2 2003

DOI: 10.2478/ijasitels-2023-0003

Different Chopper Types for Supply Separately Excited DC Motor

Alina Cristina VIOREL[1], *Gabriela CRĂCIUNAȘ*[1]

[1] *Computer Science and Electrical and Electronics Engineering Department, Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania*
{alina.viorel, gabriela.craciunas } @ulbsibiu.ro

Abstract

Nowadays saving the planet's resources plays a very important role in improving our life conditions. The purpose of the paper is to supply the separately excited dc motor by using different chopper circuits. A very sustainable solution is to replace the resistors from classic theory with dc choppers for better efficiency. The armature voltage can be controlled by using IGBT chopper and the PWM signal received from controller is done with a dedicated block created by MATLAB environment. The advantages of developing this technique are that the speed changes proportionally with armature voltage and inversely with field voltage by keeping field and armature voltage constant respectively. The chopper designed with IGBT and Diode modules offers good control at any frequency value and the speed, the armature current and electromagnetic torque of DC motor can be control. In this paper, different choppers had been simulating were working in four quadrants for supply an excited dc motor by using Matlab program.

Keywords: DC-DC converters, Matlab, open-loop control.

1 Introduction

Power electronic plays an important role in electric drives and control in the last decades. The semiconductor properties create high performance and high flexibility in electrical drives with many variable speed applications. DC drives are used in applications such as, good speed regulation adjustable speed control, braking and reversing frequent starting. The modern techniques of controlling the speed supply of a DC motor used different types of choppers instead of variables resistances placed in series with armature circuit.

The DC chopper system converts directly from DC to DC by turning the switch ON/OFF the voltage supply. The choppers are classified in different types, from A to E, according to the polarity of the output voltage and current. Thus, they can be of one quadrant (type A and B), two quadrants (type C, D), depending on the quadrant in which the output voltage and current lie.

In this paper, the control speed is done with different types of choppers which supply the armature circuit. The chopper is done with IGBT and Diode modules because offers a fast response, smooth control capability with good efficiency. The simulation of the model is done and analysed in MATLAB (Simulink) using different chopper types. The chopper control needs a required control signal for the IGBT which is done using the PWM Generator (DC-DC) block from SIMULINK library.

The speed of the DC machine could be observed by an outer speed loop using a PI controller.

2 Supply speed control of separately excited DC motor

The chopper converts the fixed DC voltage to variable DC voltage. Self-commutated devices (directly on or off devices via gate) like MOSFET, IGBT, power transistors, GTO are used for making choppers because they can be commutated by low power control signal and do not need commutation circuit. [1]
The open loop control voltage of separately excited DC motor is done in figure 1.

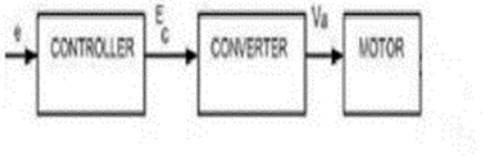


Figure 1 The block diagram for separately excited dc motor

To obtain a variable DC source an IGBT chopper controlled with a PWM signal is done and with this is possible to manage operating the excited dc motor in all four quadrants. Further, the paper presents converters for one quadrant equipped with choppers (A, B types) which supply the dc motor. The waveforms of current and voltage done by each type of chopper and the influence of them on dc motor speed and armature current values are presented. In this paper a dedicated block to generate PWM signals is used. The parameters of dc motor are given from Matlab blocks preset model with following values: 5HP 240V 1750 RPM Field: 300V [2].

2.1 Chopper Type – A

This chopper is representative of working in the first quadrant where the output voltage and current can be zero or positive as well as the power delivered to the motor. Thus, the power can flow only from the source to the load. The model created in Matlab/Simulink for supplying a dc motor is presented in figure 2. The waveforms obtained from type-A chopper are done in figure 3.

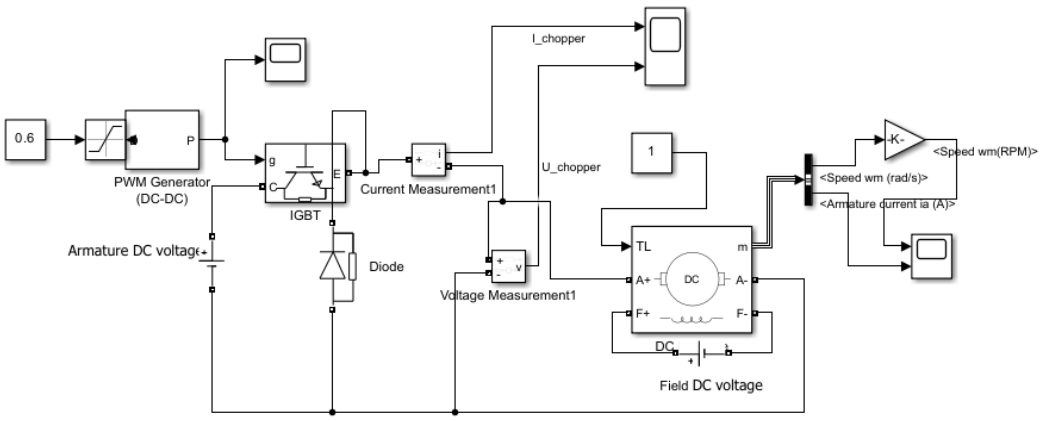


Figure 2 Simulink model of open loop model of chopper with dc machine (first quadrant).

The chopper contains an IGBT power electronics device for a very smooth transition between on/off period. [3]
The values of output electrical measures are according to the type-A chopper working principle. This chopper is attached to a dc motor. The results of speed motor and armature current obtained for dc motor fed by a type-A chopper are presented in figure 4.

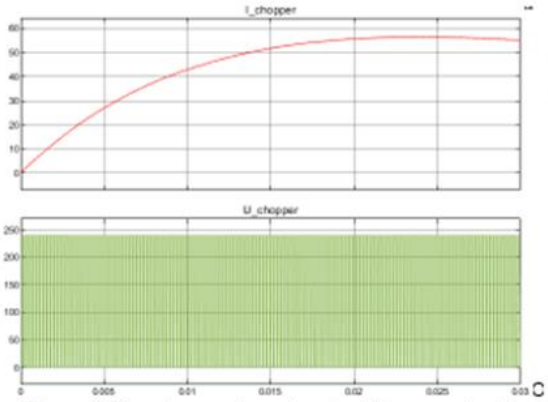


Figure 3 The chopper type-A output for current and voltage waveforms.

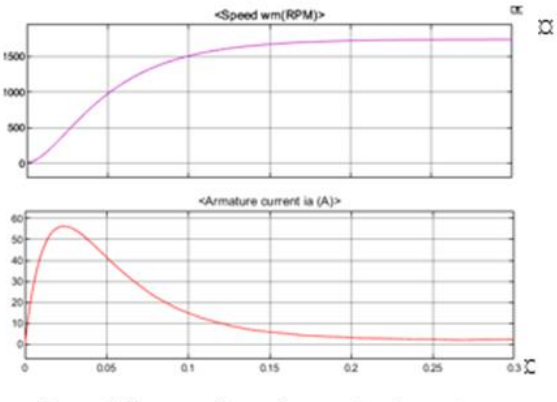


Figure 4 The waveforms for speed and armature current of dc machine.

In the open loop model of speed control of separately excited D.C. Motor for the machine the output speed is around 1750 RPM, and the armature current is observed to be above the rated value.

2.2 Chopper Type-B

Operation in the second quadrant corresponds to forward braking, so the sign of energy is changed because the output voltage is positive or zero and the output current is zero or negative. The simulation model is presented in figure 5. The dc motor in this case acts for a short duration as a generator and the kinetic energy stored in the system is returned to the input dc source.

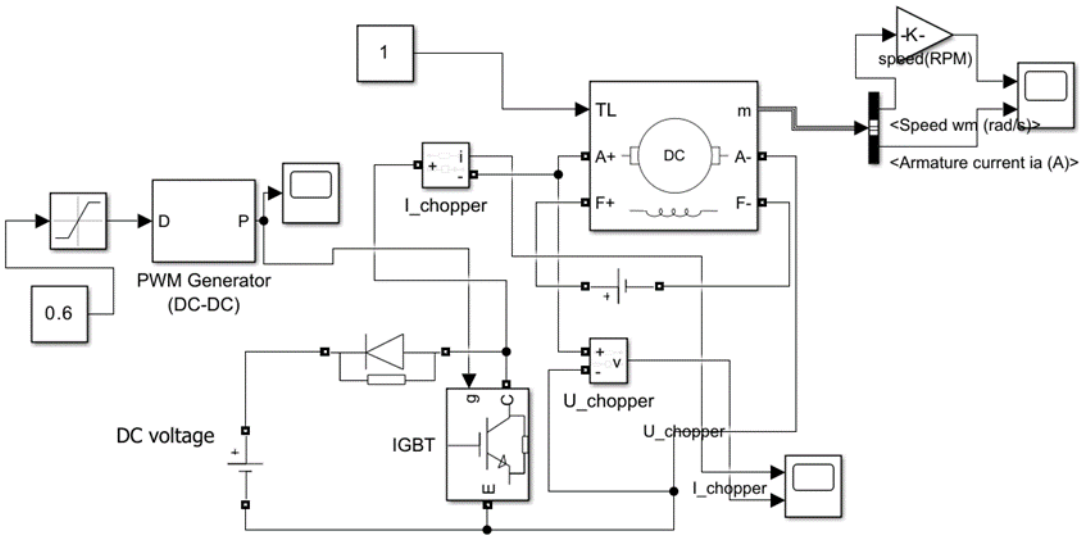


Figure 5 Simulink model of open loop model of chopper with dc machine (second quadrant).

The type-B chopper model has the waveforms obtained for current and voltage presented in figure 7. If the load is a dc motor and the voltage value described by (1) should be always greater than the dc source.

$$V_{(out)} = E + R_a i + L_a \frac{di}{dt} \qquad (1)$$

Where: - E is back emf, Ra – armature circuit resistance, La – armature circuit inductance. The results obtained by simulation are presented in figure 6. The power delivered to the load is negative, and the diode will allow the current to flow only from the load to source. The average output voltage of the chopper depends on the duty cycle of the switch. Due to the electrical loading the speed of the dc machine (now acting as generator) decreases Figure 7, and the armature current remains negative for a short period of time like in figure 8.

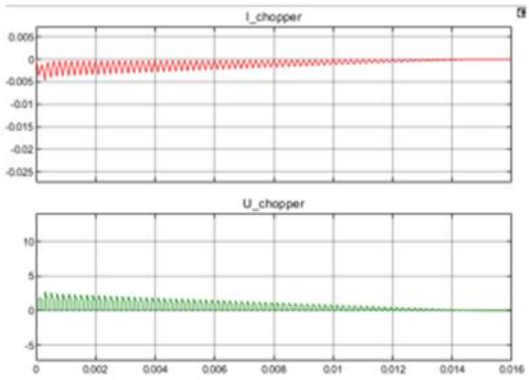


Figure 7 The chopper type-B output for current and voltage waveforms.

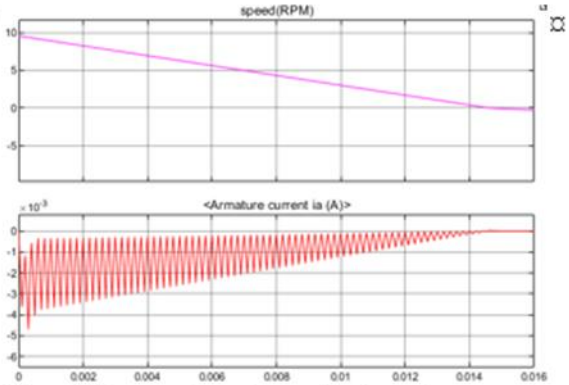


Figure 8 The waveforms for speed and armature current of dc machine.

An external inductance is added in most applications to increase the of dc motor armature inductance value.[4]

The other situations for converters build up with choppers for two and four quadrants a closed loop control is more adequately. A type-C chopper is also called a Two-Quadrant Class-A Chopper where the power can either flow from source to load or load to source and it can be used as a step-down or step-up chopper. Type-D chopper for example, is a circuit configuration of chopper in which power can flow in either direction, from source to load and load to source. The operation of this chopper is confined to the first and fourth quadrant. This type of chopper is also known as Two quadrant Type-B Chopper.

3 Conclusions

The paper presents a possibility to feed a separately excited DC motor by a chopper of one quadrant. The speed and the armature current of a dc motor has been successfully modelled by using chopper as a converter with IGBT electronic device and diode based on the opened loop model of DC motor. The simulation is done in MATLAB under the Simulink program which has a ready-to-use PWM generation block as well. The results are also studied and analysed under above mentioned conditions. The model shows good results under all conditions employed during simulation. Since, the simulation of speed supply in open loop of DC motor has been done.

The speed of the DC machine could be compared by an outer speed loop using a PI controller, that means a closed loop control which should be a future development of this paper.
Thus, the future aim is to develop the other types of choppers (C, D, E) to have total control on speed values in both situations: open loop and closed loop diagram.
The simulation results indicate that the proposed control schemes have good performance and robustness with appropriate safety for the DC drive system under steady and variable operating conditions, and in the presence of motor supply parameter variations.

References

[1] Anitha M, Pradeep M, Pruthviraj B.G., *MATLAB Simulation of Choppers*, International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering REPSE-17, Bengaloru, ISSN (online) 2278-1021, 2017.
[2] Farzin A, *Simulation of Power Electronics Circuits with MATLAB/SIMULINK, Design, Analyze, and Prototype Power Electronics*, https://doi.org/10.1007/978-1-4842-8220-5, ISBN-13 (electronic): 978-1-4842-8220-5, Maltepe University Istanbul, Turkey, 2022.
[3] Suryawanshi Monika, Pote V., Bhalerao A., *MATLAB Simulation on Chopper based Speed Control of DC Motor. A Review.*, International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395 -0056, 2017.
[4] Syed Abdul Rahman Kashif, *Chopper Fed DC Motor Drive -- Speed Control of DC motor* (https://www.mathworks.com/matlabcentral/fileexchange/35196-chopper-fed-dc-motor-drive-speed-control-of-dc-motor), MATLAB Central File Exchange. Retrieved November 20, 2023.

DOI: 10.2478/ijasitels-2023-0004

Fog Detection through Image Processing Methods

Teodor-Adrian Radescu[1], *Arpad Gellert*[1]

[1]*Computer Science and Electrical and Electronics Engineering Department, Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania*

{adrian.radescu, arpad.gellert}@ulbsibiu.ro

Abstract

This paper presents a fog detection algorithm, highlighting the significance of continued exploration in fog identification through image processing techniques. The advancement and application of this algorithm can significantly benefit various domains, including road safety, environmental monitoring, navigation, security, surveillance, and improving existing systems' performance. The evaluation performed on test images have shown an accuracy of 72%, a precision of 94%, a recall of 57% and an F1 score of 0.71. The proposed algorithm clearly outperformed some existing fog detection methods.

Keywords: image processing, computer vision, fog detection, fog density, visibility

1 Introduction

High image quality can be important in some computer vision methods, specific image restoration like denoising (see [1], [2] and [3]) or inpainting (see [4]) being helpful. Defogging is another image enhancement technique, whose first step is fog detection. Within the intricate landscape of transportation, adverse weather conditions such as fog, rain or snow pose formidable challenges, significantly impacting safety and efficiency. This article addresses these challenges, focusing specifically on the development of a fog detection algorithm grounded in the domain of image processing.

The unpredictability and severity of reduced visibility have long been a pressing concern for drivers, autonomous vehicles and traffic control systems. These natural phenomena not only obscure the visual landscape but also necessitate critical adjustments in speed and navigation. The degradation of contrast and color quality under foggy conditions further increases the risks associated with road transportation, emphasizing the need for a robust solution.

In response to these challenges, this article unveils a comprehensive algorithm designed to detect the adverse effects of fog. Leveraging image processing methods, this approach aims to identify the presence of fog in visual data and subsequently apply corrective measures. The implications of this work extend across various transportation sectors, offering potential enhancements in safety and operational efficacy. From its potential applications in autonomous vehicles to its integration within traffic control systems, this fog detection algorithm represents an advancement toward ensuring safer, more efficient travel in adverse weather conditions

The rest of this paper is organised as follows: Section 2 reviews the related work in fog detection, Section 3 presents the proposed fog detection algorithm, Section 4 discusses the experimental results, whereas Section 5 concludes the paper and establishes some further work directions.

2 Related work

This section concentrates on reviewing and analysing the diverse array of image processing techniques employed in fog detection systems. These methods aim to discern, quantify, and mitigate the impact of fog on visual data, thereby enhancing visibility and safety in various transportation domains. By examining the nuances and efficacy of these image processing approaches, this review aims to provide a comprehensive understanding and potential directions for advancing fog detection algorithms within the domain of image processing. A review of the solutions related to visibility enhancement and fog detection has been presented in [5].

The effects of a foggy environment on light are described by Dumont in [6]. In fog, visible light (with wavelengths from 380 up to 780 nanometers) passing through an aerosol containing a high quantity of water particles becomes dispersed. Along its path, the light beam from headlights is attenuated due to the dual phenomenon of absorption and diffusion, which characterize the fog based on an extinction coefficient K [m^{-1}] (the sum of the diffusion and absorption coefficients). However, in reality, the absorption is negligible, so diffusion remains predominant, causing light to deviate from its initial direction.

Koschmieder's law [7], formulated in 1924, defines how light diminishes through fog as distance or fog particle concentration increases. This law establishes an exponential relationship between light transmission and distance travelled within foggy conditions, where visibility decreases exponentially with increasing fog density or the distance through the fog. It highlights the attenuation of light due to scattering and absorption by fog particles, which is crucial knowledge for designing image processing algorithms aimed at detecting and quantifying fog in visual data.

Initially, the proposed algorithms used multiple images to gather the necessary information, thus forming an image unaffected by fog. The major drawback of this implementation is its dependency on weather conditions and limitations posed on using these systems in real-time scenarios. Polarization-based methods as described by Schechner and Narasimhan in [8] utilize two or more images captured with varying polarization degrees.

Pagani et al. presented an innovative work in [9], with a focus on leveraging neural networks for the automatic detection of fog in surveillance camera images. The study explored the application of advanced machine learning techniques to enhance fog detection capabilities, aiming to address challenges associated with adverse weather conditions in surveillance systems. Their work significantly contributed to the field by demonstrating the potential of neural networks in effectively identifying foggy conditions, thereby enhancing the reliability and efficiency of surveillance systems in varying weather environments.

The study of Liu et al. [10], delved into driving obstacle detection technology specifically designed for foggy weather conditions. The research proposed the utilization of GCANet (Generative Contextual Attention Network) along with feature fusion training methodologies. This approach aimed to improve obstacle

detection in challenging foggy environments, addressing safety concerns in driving scenarios affected by reduced visibility due to fog. Their work contributes to advancing obstacle detection systems by integrating innovative neural network architectures and feature fusion techniques to enhance performance and safety in foggy weather conditions.

Guo et al. introduced in [11] a novel haze image classification technique using the AlexNet network transfer model. Their study aimed to tackle the classification of haze images by leveraging transfer learning using the AlexNet neural network architecture. This approach sought to improve the classification accuracy of hazy images, addressing challenges related to haze distortion and visibility issues. By applying transfer learning techniques, the authors explored the adaptation of a pre-trained AlexNet model to effectively classify haze images, offering potential advancements in image classification methodologies in the context of varying atmospheric conditions. Their work contributes to enhancing image classification accuracy, particularly in scenarios affected by haze or atmospheric distortions.

Focusing on image processing techniques, the last decade has seen the development of computer vision methods to detect fog in images using various features such as color, texture, and contrast. However, these methods often face accuracy limitations due to variations in illumination and weather conditions. One initial approach to detect fog presence in an image, utilized in the algorithm to be presented in Section 3, involves analysing the brightness level within an image's environment. This process entails evaluating the overall luminance level within the scene depicted in the image, as described in [12] and [13]. In [14], Bronte et al. discuss the application of the Sobel operator for fog detection. The Sobel operator is used for edge or gradient detection in images. By evaluating the intensity gradient in various directions within an image, it highlights areas with abrupt changes in intensity, which can be useful in identifying regions affected by fog. As described by Bronte et al., foggy images have lower contrast and are blurrier than clear images. This means that, in foggy images, information in the higher frequencies is lower.

The algorithm described in this work is an amalgamation that draws from the distinctive, yet complementary methodologies discussed earlier. It intricately intertwines the concepts and practices derived from the evaluation of overall scene luminance, as presented in [12] and [13], with the fundamental principles intrinsic to edge and gradient detection techniques employed through the Sobel operator explained by Bronte et al. in [14].

By blending these different methods together, the goal of our proposed algorithm is to overcome the weakness of using only one method. The objective is to create a fog detection system that is detailed and flexible, not just solving the problems of each method alone, but also able to work well in various weather and lighting situations, while also being lightweight and easy to expand on.

3 Fog detection algorithm

The images obtained through analog or digital cameras can create the illusion of fog presence due to various reasons, including camera settings and influence from light sources. For instance, prolonged exposure might accumulate more noisy pixels, resulting in a blurry image. Additionally, ISO sensitivity can impact the noise level, leading to reduced quality. Exposure time also plays a significant role in creating the

illusion of fog, as object or camera movements can cause a blurred effect. Moreover, the light sources within the scene can significantly impact the images. Strong light can reflect off objects in front of the camera, creating the appearance of fog, while intense shadows from strong light can further enhance this effect. Similarly, weak light can equally influence the image, resulting in a dark and blurry picture that might be mistaken for a fog effect.

The presented method for fog detection is designed to consider these influences, aiming to offer precise detection of the fog effect in the input image. A workflow of this method can be observed in Fig. 1, providing a detailed breakdown of each step necessary to determine if an input image is genuinely affected by fog.

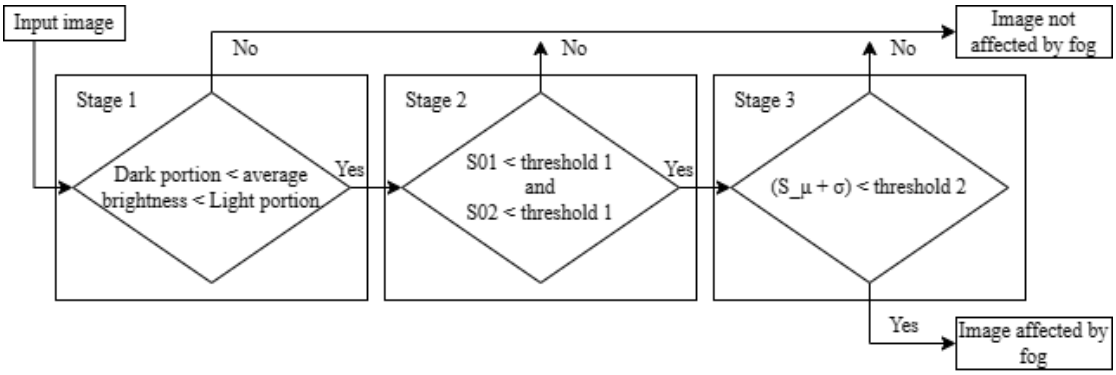


Figure 1. The workflow of the fog detection algorithm

Before applying the algorithm to the image, some specific preprocessing steps are essential for ensuring accurate detection. One of these steps involves converting the color image into a grayscale image, facilitating better detection of pixel intensity differences. In color images, colors can influence the visual perception of fog, potentially leading to inaccurate detection. Additionally, this conversion reduces the information load required for processing, potentially improving performance. Another preprocessing step involves calculating the image histogram. The histogram aids in assessing the distribution of pixels based on their intensity. In foggy images, a majority of pixels are expected to have a low intensity. Analysing the histogram provides insights into the brightness level of the image, crucial information influencing the decisions of the algorithm.

The first stage of the algorithm involves analysing the average brightness of the input image. As the average brightness of images produced in foggy conditions varies within certain limits, this analysis proves useful in determining whether an image is affected by fog. Typically, the average brightness of fog-affected images falls within a grayscale level ranging between 100 and 160, with histogram variations usually between 30 and 220. Based on this analysis, a condition is derived requiring the input image to possess an average brightness within a predefined range. Further analysis of the image's brightness reveals that the image histogram can be utilized to determine the distribution of grayscale levels from dark to light as seen below.

The second stage involves applying the Sobel operator to the input image, aiming to extract distinctive features of objects and structures present in the image, even in foggy conditions. This operator is commonly used in object recognition and scene analysis applications. In fog detection, this step is crucial to highlight edges and outlines of objects or structures visible despite foggy conditions. The detection algorithm

utilizes the edge gradient changes from the Sobel image to analyse differences between fog-affected and unaffected images. During this process, grayscale levels of edge pixels typically range between 250 and 255 on the histogram distribution. Analysing the probability density function, an increase in the average number of pixels with grayscale levels between 250 and 255 indicates the image is not affected by fog, whereas a decrease in this number suggests the opposite. Therefore, when the number of pixels within this range surpasses a certain threshold, the image is considered unaffected by fog. Conversely, when the number falls below the threshold, it indicates the image is affected by fog. Due to varying image size, normalization is necessary to ensure accurate comparison. The normalization applied by the Sobel algorithm ensures that the number of pixels contributing to detecting fog in an image is not influenced by its size. For instance, a larger image may have a higher number of fog-affected pixels, but this absolute value may be comparable to a smaller image with fewer fog-affected pixels. Normalization compares the proportions of fog-affected pixels in each image, resulting in a more objective and precise comparison. Additionally, normalization can help reduce uneven brightness effects or contrast differences that may appear in different images, potentially leading to false positives.

The third and final stage of the fog detection algorithm in an image involves analysing the standard deviation and mean of the pixels obtained from the Sobel-processed image in the previous stage. These measures provide significant information about the intensity of edges in the image, which can be utilized to evaluate the presence or absence of fog. In foggy environments, images often exhibit reduced visibility, with details hard to discern, resulting in a blurry and unclear appearance. From an image processing standpoint, a fog-affected image appears smoother and more uniform, with minimal variation in pixel intensity. Conversely, a clear image is characterized by numerous details, intricate textures, and significant pixel intensity variations. Thus, determining if the image is affected by fog involves using the information acquired from the previous stage to calculate the pixel mean and standard deviation of the Sobel-processed image. Using the Sobel operator yields a pixel mean as per the equation:

$$S_{\mu} = \frac{\sum_{i=1}^{n*m} S_i}{n*m} \quad (1)$$

where S_{μ} represents the Sobel image's mean, S_i denotes each pixel of the Sobel image, and $n * m$ denotes the image size.

In fog-affected images, due to airborne particles, the image appears disrupted, and objects seem less defined, making edge highlighting by the Sobel operator more challenging. Comparing the gradient change of a Sobel image for a fog-free and fog-affected image reveals that for the latter, the change appears relatively disordered and diffuse, while for the fog-free image, the change is more evident and well-defined. This difference arises because fog-affected images' disturbances hinder edge detection.

The standard deviation measures the variation or dispersion of data concerning their mean. In image processing, the standard deviation indicates how much the pixel values in the image deviate from the image's mean. A higher standard deviation implies greater variation in pixel values around the mean, suggesting increased contrast. Conversely, a lower standard deviation indicates minimal variation in pixel values concerning the image's mean, suggesting reduced contrast.

$$\mu = \frac{\sum_{i=1}^{n*m} P_i}{n*m} \quad (2)$$

The following equation can be used to calculate the mean μ, where P_i represents a pixel of the original image, and $n * m$ denotes the image size. Using the mean μ, the subsequent equation can be employed to obtain the standard deviation.

4 Experimental results

This section thoroughly examines the performance of the proposed fog detection algorithm, showcasing its effectiveness in identifying fog in images. It presents and analyses results obtained from applying the algorithm to a diverse dataset of 462 images, capturing various scenarios of fog density and lighting conditions. These experiments evaluate the algorithm's accuracy across different settings. The computing system used for these evaluations comprises an Intel Core i5 processor (1.5 GHz, 16 GB RAM, and Intel Iris Xe Graphics). The hardware significantly influences the algorithm's processing speed and capacity. Results are system-specific, with potential variations in performance on other systems. A snapshot of these findings is outlined in Table 1, displaying selected images from the dataset and the algorithm's outcomes based on Sobel image mean, standard deviation, and average brightness levels.

Table 1. Experimental results for the fog detection algorithm

NAME	MEAN	DEVIATION	BRIGHTNESS	DETECTION
0001-0.jpg	60	85.6	126	CORRECT
0001-1.jpg	35	54.3	107.7	CORRECT
0001-2.jpg	38	59.8	123.9	WRONG
0002-0.jpg	32	47.8	108.9	CORRECT
0002-1.jpg	16	19.2	112.6	CORRECT
0002-2.jpg	17	19.4	114.5	CORRECT
0002-3.jpg	17	16.1	135.8	CORRECT
0002-4.jpg	14	13.7	120.8	CORRECT
0007-0.jpg	49	77.8	112.3	CORRECT
0007-1.jpg	31	58.5	109.4	CORRECT
0007-2.jpg	28	49.8	120.4	WRONG
0007-3.jpg	20	41	136.6	WRONG
0007-4.jpg	14	30.6	134.1	CORRECT
0011-0.jpg	46	76.2	152.5	CORRECT
0011-1.jpg	29	54.9	149.8	CORRECT
0011-2.jpg	15	31.4	141.6	WRONG
0011-3.jpg	8	13.8	142.7	CORRECT
0011-4.jpg	8	10.7	145.9	CORRECT
0012-0.jpg	89	116.3	137.2	CORRECT
0012-1.jpg	35	51.6	126.5	CORRECT

There are several evaluation metrics used to measure the accuracy of fog detection algorithms in images. Among the most critical metrics are the following: accuracy, precision, recall and F1 score. The accuracy determines the ratio of correct classifications to the total classifications made by the algorithm. For the presented algorithm in this work, the accuracy for the dataset used is approximately 72%. The precision reflects the proportion of correctly identified clear images among all images classified as clear. After applying the algorithm to the dataset, the precision is approximately 94%. The recall quantifies the algorithm's ability to correctly identify foggy images among all foggy images in the dataset. The algorithm's recall value for the dataset is approximately 57%. The F1 Score combines precision and recall to provide a balanced assessment of fog detection. The calculated F1 score for this algorithm is approximately 0.71, indicating its ability to distinguish between foggy and clear images.

Within the experiment, attention was also given to the execution time of the fog detection algorithm. For each processed image, the time interval required for the complete execution of the algorithm was measured. Assessing execution time is crucial to understand the efficiency and performance of the algorithm in handling a large volume of data. This evaluation provides relevant information about the algorithm's speed and effectiveness in practical applications, aiding in identifying potential weaknesses and subsequent optimizations. The execution time can directly impact user experience and the practical applicability of the algorithm in real-time scenarios or applications with strict time requirements.

The experimental results revealed that the algorithm's execution time can vary based on the input image's size and complexity. Images with higher resolution or complex characteristics may require more time to process compared to smaller or simpler images. Fig. 2 represents a valuable resource in understanding the relationship between image resolution and processing time. This information can be useful in taking decisions regarding image selection and the feasibility of the algorithm in different scenarios.

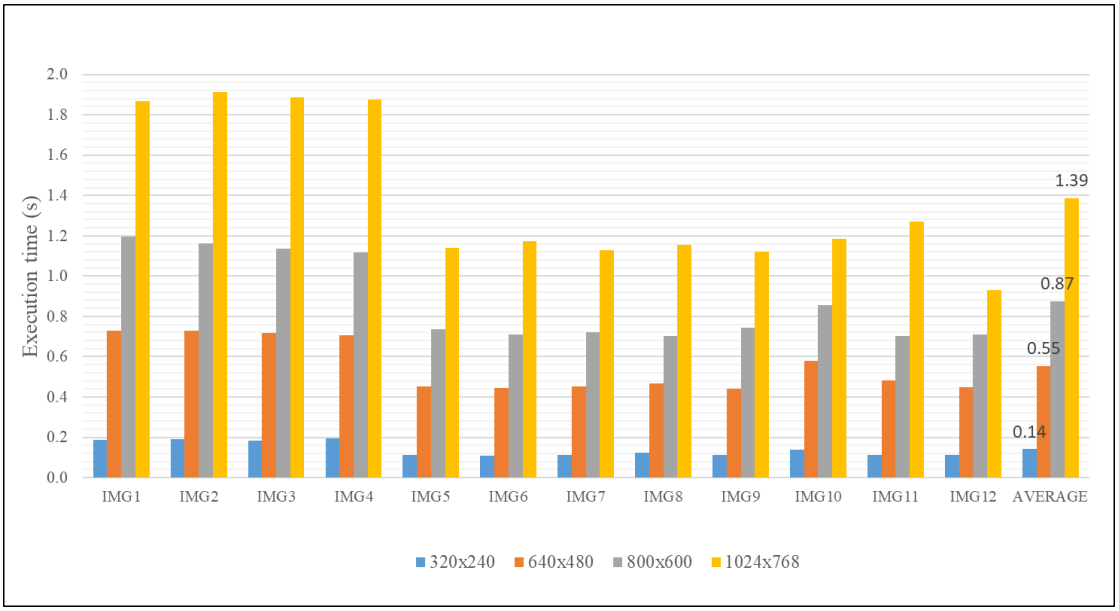


Figure 2. Execution time based on image resolution

Additionally, from the analysis depicted in Fig. 2 it can be inferred that the fog detection algorithm demonstrates significantly faster performance when applied to images with a resolution of 320×240 pixels, yielding consecutive and predictable results. As image resolution increases, execution time significantly extends and becomes less consistent in delivering results. Understanding the execution time associated with each image resolution allows for a quick evaluation of the algorithm's performance based on specific needs. For instance, in real-time applications where processing time is critical, opting for lower-resolution image captures or implementing image resizing techniques may be preferable. Conversely, if maximum accuracy in fog detection is required and processing time is not a major concern, higher-resolution images can be utilized, even though they require more execution time.

Comparative analysis of fog detection algorithms plays a pivotal role in understanding their strengths, weaknesses, and overall performance. As the field of fog detection continues to evolve, researchers and practitioners strive to develop robust and precise algorithms capable of efficiently detecting fog in images. Making informed decisions regarding the algorithm to use in a specific application or scenario necessitates a comparative analysis evaluating their performance based on well-defined criteria. Table 2 presents comparative results for the algorithm discussed alongside three other relevant algorithms. The table data reflect the relative performance of the algorithm discussed in this work compared to the three selected algorithms for analysis. These metrics provide insight into the effectiveness and reliability of each algorithm concerning fog detection in images.

Table 2. Metrics for various fog detection algorithms

Algorithm	Accuracy	Precision	Recall	F1 score
Proposed algorithm	72%	94%	57%	0.71
Deep Neural Network [9]	99%	60%	70%	0.65
GCANet	47%	62%	68%	0.64
Alexnet Network Transfer Model	67%	71%	67%	0.64

It is essential to note that these results are specific to the dataset and experimental conditions used in the studies from which they were obtained. The performance of the algorithms may vary depending on different datasets and configuration parameters.

Analyzing Table 2, the algorithm proposed in this paper demonstrates remarkable accuracy – the algorithm's capability to correctly classify images, and higher accuracy values indicate a greater ability to detect fog in images. In comparison with the algorithm proposed by Pagani and colleagues in [9], the proposed algorithm shows moderate accuracy but highlights significantly higher accuracy than the other two algorithms.

However, the experiment reveals that the proposed algorithm achieves high precision values. Precision refers to the correct proportion of positive detections relative

to the total images classified as positive and is crucial in evaluating the algorithm's ability to avoid misclassifications. The proposed algorithm exhibits significantly higher precision compared to the other cited algorithms, indicating a superior ability to distinguish correctly between images with and without fog.

On the other hand, the analysis of the algorithm's sensitivity shows comparable results to the other algorithms. Sensitivity, also known as "recall" or the true positive rate, measures the algorithm's ability to detect all positive cases. Similar sensitivity outcomes indicate that the algorithm doesn't present significant advantages or disadvantages regarding the detection of positive fog instances compared to the other selected algorithms for the study.

The F1 score, an aggregated measure that combines precision and sensitivity, provides an overall perspective on the algorithm's performance in detecting positive fog instances. Evaluating the proposed algorithm in terms of F1 score indicates a favorable outcome, demonstrating its ability to achieve an optimal balance between correct fog detection and minimizing misclassifications.

The experimental results obtained in this study highlight the efficacy and performance of the proposed algorithm in fog detection in images. The analysis of evaluation metrics showcases the algorithm's proficiency in producing precise results while minimizing misclassifications. Comparing the algorithm with other existing approaches has demonstrated its consistency and reliability. Thus, the proposed algorithm represents an efficient and promising solution in fog detection in images, with potential applications across various previously discussed domains.

5 Conclusions and further work

In this work, fog detection in images was addressed using an image processing-based algorithm, a justified choice considering its adaptability and automation potential. The domain of image processing, focusing on visual information analysis and manipulation, is relevant due to the fog's impact on image visibility and quality.

Algorithms based on image processing techniques offer flexibility and adaptability, being customizable as per application-specific needs and can be integrated into existing systems, allowing efficient real-time fog detection or handling a large volume of images. The fog detection algorithm employed three conditions, each addressing distinct aspects of detection, resulting in a comprehensive and reliable fog detection system. These conditions analyze ambient luminosity, compute thresholds based on image characteristics, and consider Sobel image mean and standard deviation. Their integration enhances fog detection reliability and efficiency across various scenarios, contributing to a more robust fog detection algorithm. The proposed algorithm presented an accuracy of 72%, a precision of 94%, a recall of 57% and an F1 score of 0.71, clearly outperforming some existing fog detection methods.

Future developments aim to optimize the fog detection algorithm, especially in variable lighting conditions or when dealing with other atmospheric phenomena related to fog. Enhanced fog level estimation methods integrating factors like visibility distance and environmental properties are proposed. Furthermore, a fog removal algorithm utilizing machine learning and neural networks for a better understanding of fog in images could be explored. The adaptation of the algorithm for real-time video fog detection is also considered. These directions intend to achieve more precise and practical outcomes suitable for real-world applications.

References

[1] A. Gellert and B. Remus, "Context-Based Prediction Filtering of Impulse Noise Images," *IET Image Processing,* vol. 10, no. 6, 2016.

[2] A. Gellert and R. Brad, "Studying the influence of search rule and context shape in filtering impulse noise images with Markov chains," *Signal, Image and Video Processing,* vol. 12, no. 2, 2018.

[3] A. Gellert, R. Brad, D. Morariu and M. Neghina, "Filtering Random Valued Impulse Noise from Grayscale Images through Support Vector Machine and Markov Chain," *International Journal of Advanced Statistics and IT&C for Economics and Life Sciences,* vol. 11, no. 1, 2021.

[4] A. Gellert and R. Brad, "Image Inpainting with Markov Chains," *Signal, Image and Video Processing,* vol. 14, no. 7, 2020.

[5] R. Miclea, V. Ungureanu, F. Sandru and I. Silea, "Visibility Enhancement and Fog Detection: Solutions Presented in Recent Scientific Papers with Potential for Application to Mobile Systems," *Sensors,* vol. 21, no. 10, 2021.

[6] E. Dumont and V. Cavallo, "Extended Photometric Model of Fog Effects on Road Vision," *Transportation Research Record Journal of the Transportation Research Board,* vol. 1862, no. 1, pp. 77-81, 2004.

[7] Z. Lee and S. Shang, "Visibility: How Applicable is the Century-Old Koschmieder Model?," *Journal of the Atmospheric Sciences,* vol. 73, no. 11, p. 4573–4581, 2016.

[8] Y. Y. Schechner and S. G. Narasimhan, "Instant dehazing of images using polarization," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[9] G. Pagani, J. W. Noteboom and W. Wauben, "Neural network approach for automatic fog detection using surveillance camera images," in *EGU General Assembly*, 2018.

[10] Z. Liu, S. Zhao and X. Wang, "Research on Driving Obstacle Detection Technology in Foggy Weather Based on GCANet and Feature Fusion Training," *Sensors (Basel),* vol. 23, no. 5, 2023.

[11] L. Guo, X. Song and H. Huang, "Haze Image Classification Method Based on Alexnet Network Transfer Model," *Journal of Physics: Conference Series,* vol. 1176, no. 3, 2019.

[12] R. T. Tan, "Visibility in bad weather from a single image," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Anchorage, Alaksa, USA, 2008.

[13] W. Sun, H. Wang, C. Sun, B. Guo, W. Jia and M. Sun, "Fast single image haze removal via local atmospheric light veil estimation," *Computers & Electrical Engineering,* vol. 46, pp. 371-383, 2015.

[14] S. Bronte, L. M. Bergasa and P. F. Alcantarilla, "Fog Detection System Based on Computer Vision Techniques," in *Intelligent Transportation Systems*, 2009.

DOI: 10.2478/ijasitels-2023-0005

Implementation of a weather station to monitor agricultural crops

Ionuț Claudiu Udrescu[1], *Alina Cristina Viorel* [2], *Gabriela Crăciunaș*[2], *Mihai Bogdan*[2]

[1]*Vitesco Technologies Engineering Romania SRL, Lyon Str., No. 2, 550018 Sibiu, Romania, E-Mail: ionut-claudiu.udrescu@vitesco.com*

[2]*Computer Science and Electrical Engineering Department, "Lucian Blaga" University of Sibiu, Romania, E-Mail: {alina.viorel, gabriela.craciunas, mihai.bogdan}@ulbsibiu.ro*

Abstract

Various studies on local crop monitoring are presented in the literature, however there is insufficient research on low-cost IoT weather stations that can obtain weather information over a long period of time. To overcome the difficulties described above, this paper proposes the development of an IoT weather station for low-cost, sustainable and multi-functional agricultural crop monitoring.

The objective of the work is to develop a local weather observation device for agricultural crops, to set up a server for storing and providing meteorological information and to perform an analysis of the acquired weather data.

The weather station architecture developed and proposed in this project consists of four main blocks: sensor and controller module, power module, local data storage module and data communication module (gateway).

Keywords: Weather station, Internet of Things, Thing Speak, Matlab, Histogram.

1. Introduction

Crop growth is normally affected by 'internal' factors represented by plant genetics and 'external' factors represented by climatic, soil, biotic, physiological, and socio-economic factors [6].

Climatic factors are the amount of precipitation, air humidity and temperature, solar radiation, wind speed and direction and the concentration of gases in the atmosphere, and soil factors are soil moisture, soil temperature, soil mineral content, organic matter content, soil pH and soil organisms. No matter how many innovations happen in the agricultural industry plant growth will always be influenced by the weather. Weather monitoring is a valuable tool to help farmers maximize production and minimize losses due to environmental factors.

A weather station for agricultural crop monitoring is essentially a data acquisition system that can collect weather information followed by processing and displaying it on a web server.

The main architectural elements of an IoT weather station for agricultural crop monitoring are sensors, controller, and data transmission network. The sensors are

responsible for acquiring weather data (air temperature and humidity, wind speed, rainfall, etc.). The station controller is usually a microcontroller and is responsible for controlling the station. The data transmission network is responsible for retrieving information from the acquisition layer and transmitting the data over the internet.
The operation of the weather station for the monitoring of the agricultural crops weeded and proposed in this project is as follows: weather information is acquired with the help of 7 sensors, the controller will process this information and transform it into data that will be written at a certain time interval on the local data storage SD frame and on the online server.
In this way, the farmer will be able to access in real time all the information about the weather conditions of the crop he is monitoring, as well as the weather history which will be available either on the SD card or on the server. With this information the farmer will be able to make important decisions on certain actions to be taken in the monitored crop. From the information provided by the crop monitoring station in this project the farmer can draw several conclusions.

2. Weather station wiring diagram

The electrical layout of the weather station for monitoring agricultural crops was designed in Proteus 7 Professional design and simulation software.
Since a weather station for monitoring agricultural crops is usually placed on farmland at a great distance from the electricity grid, we opted to power the station via a solar panel-battery assembly.

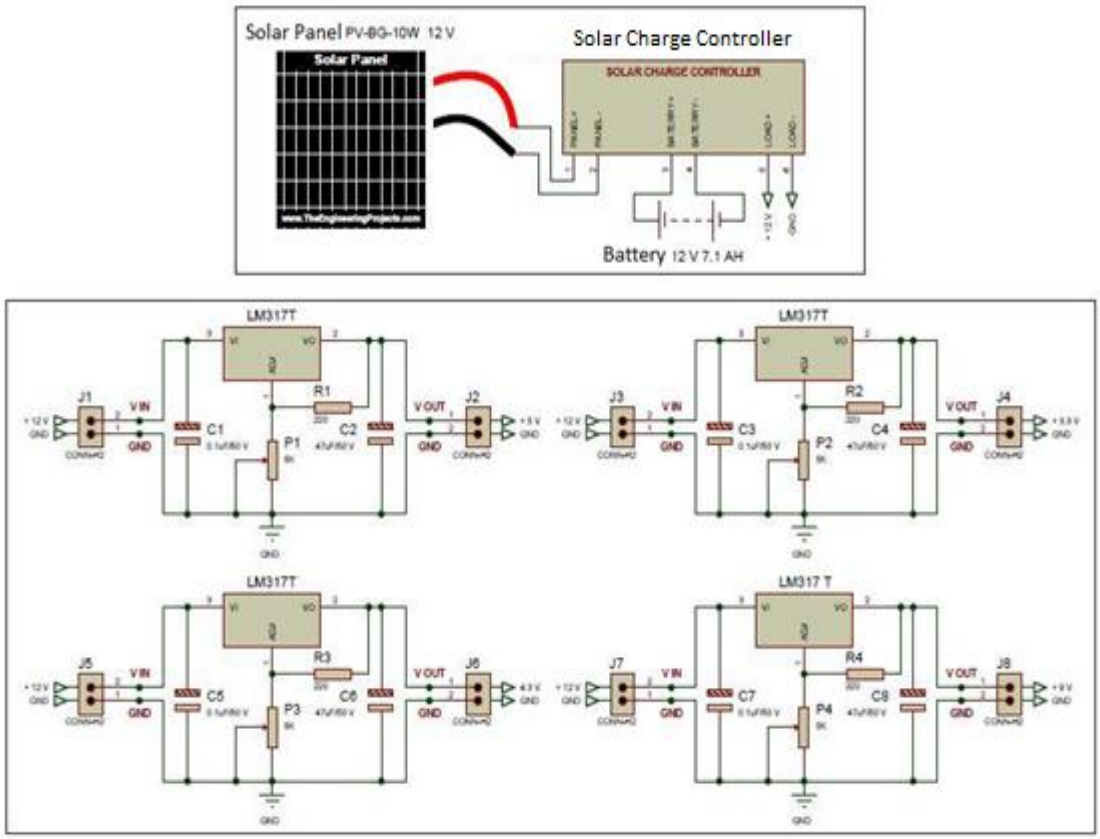


Figure 1 Weather station power supply wiring diagram

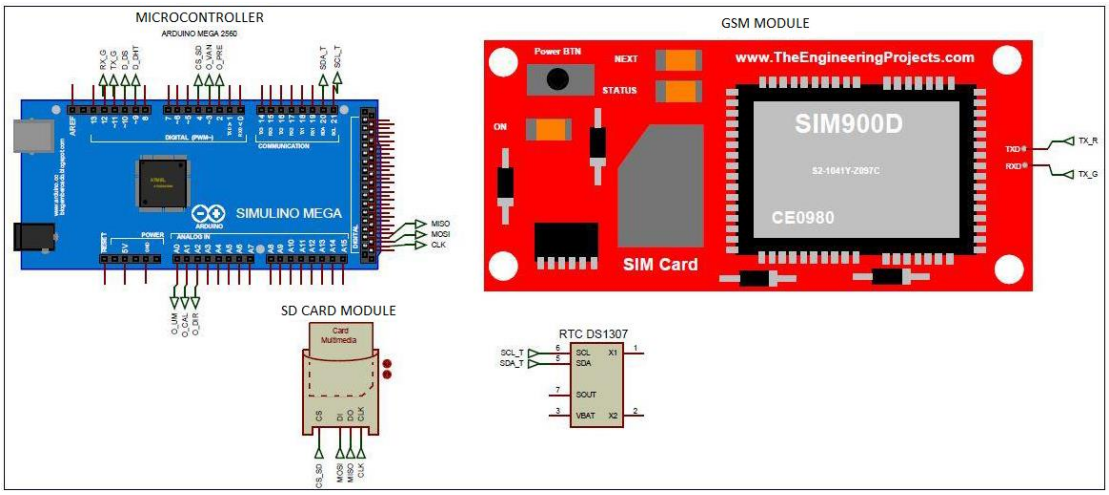


Figure 2 Controller wiring diagram and station modules

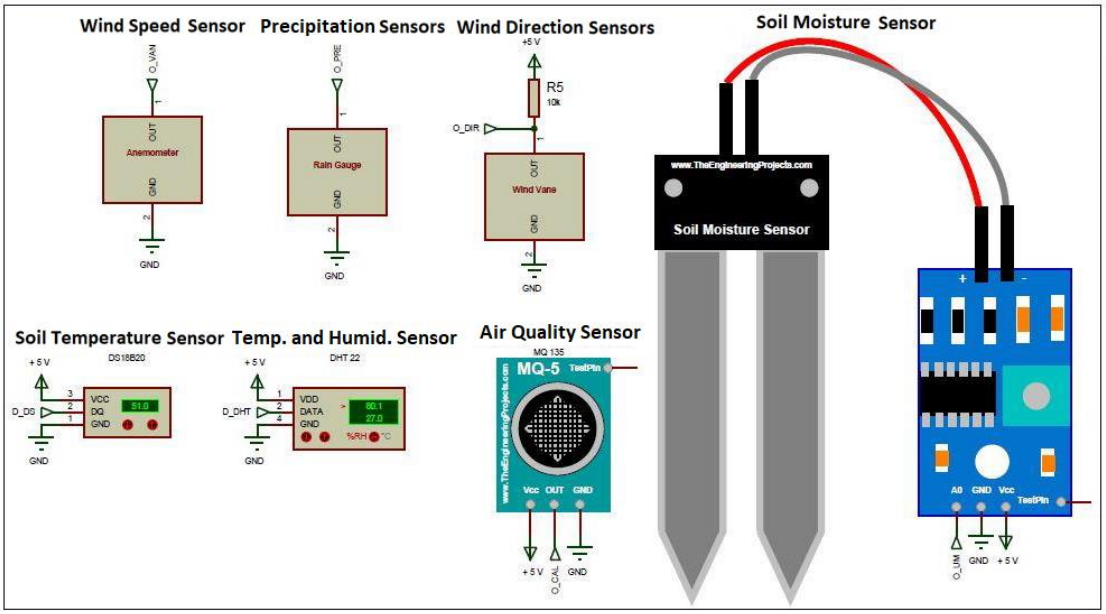


Figure 3 Wiring diagram of sensors

3. Experimental results

Weather information will be sent to a server where the farmer can view it in real time using a phone, computer, etc. As a server for sending the information, we opted for the ThingSpeak platform developed by MathWorks, which is a platform designed mainly for IoT projects that collects sensor data in the cloud and allows the development of complex analyses of the data provided in Matlab. How the ThingSpeack platform works is shown in Figure 4.

We chose the ThingSpeak platform because it allows writing up to 3 million data records per year (free non-commercial version), is robust and can be easily interfaced with Matlab where complex weather analysis can be performed using sensor data. Also ThingSpeak which has real-time data acquisition capability, allows data to be visualised as very intuitive graphs and has collaboration capabilities with various web services.

The data sent by the sensors is stored in a channel, called the ThingSpeak channel. A ThingSpeak channel consists of 8 fields where sensor data is stored, 3 fields for storing the station location and a status field where the channel status is shown.

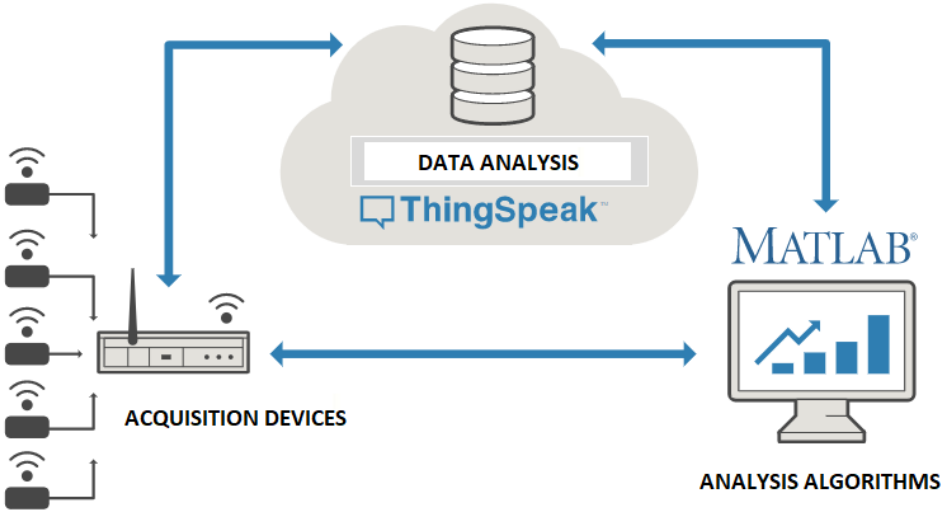


Figure 4 How ThingSpeak platform works

Figure 5 shows the configuration of the fields for the IoT weather station for agricultural crop monitoring. Figure 5 shows on which field the data from each sensor used will be displayed, for example on field number 5 the data from sensor MQ 135 sensor measuring carbon dioxide concentration is displayed.

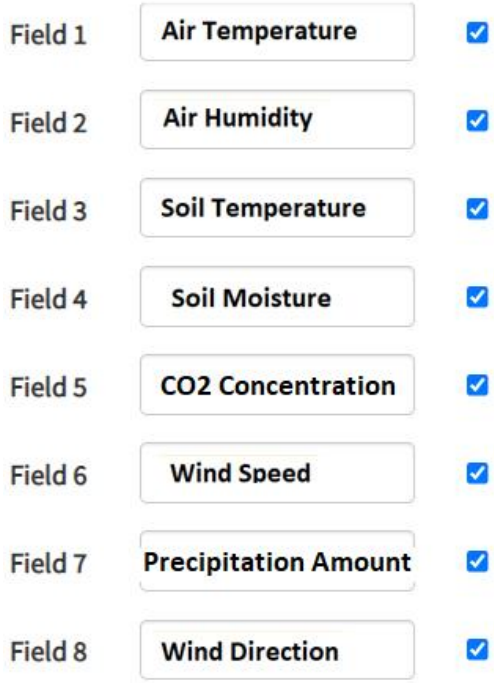


Figure 5 Setting up the weather station fields.

Data acquired by the ThingSpeak channel can be exported to MATLAB where complex analyses can be performed. In the case of agricultural crops, the most important values of a phenomenon are the minimum, maximum and average values over a given time interval.

The definition of the function for calculating the air temperature of the last 10 days (average, minimum and maximum) consisted in setting the ThingSpeak channel from which the air temperature reading is desired (defining the ID and APIkey of the channel of the IoT weather station for monitoring agricultural crops). The next step in defining the function was to read the air temperature of the last 10 days from the specific air temperature field (field 1 of the ThingSpeak channel of the IoT weather station for agricultural crop monitoring).
The last step in defining the function for calculating the average, minimum and maximum air temperature over the last 10 days consisted of calculating the minimum, maximum and average values using specific MATLAB functions (min, max, mean) followed by displaying them.

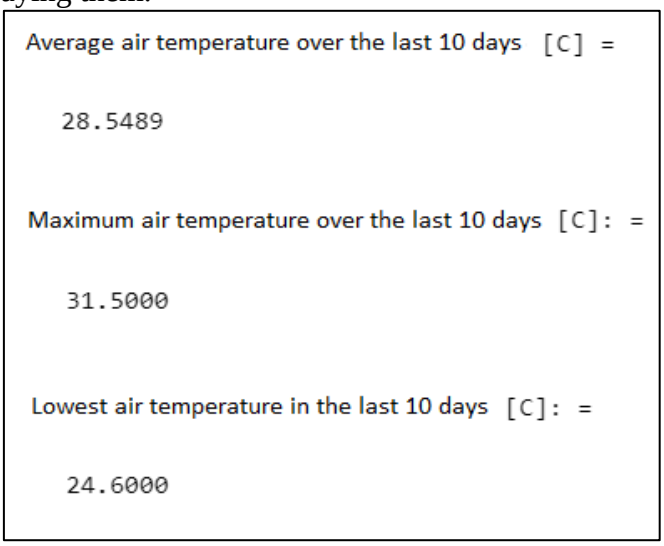


Fig. 6. Average, maximum, and minimum air temperature, over the last 10 days

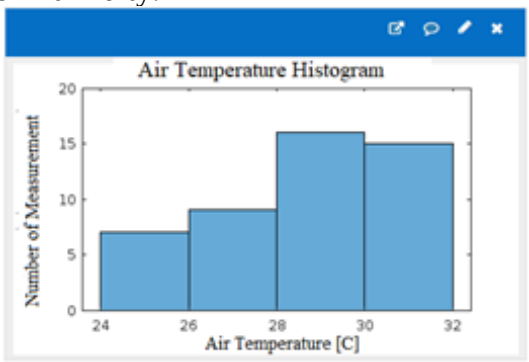
The results of mean, minimum and maximum values of air temperature, air humidity, soil temperature, soil moisture, CO2 concentration and precipitation amount for the last 10 days can be seen in Table I.

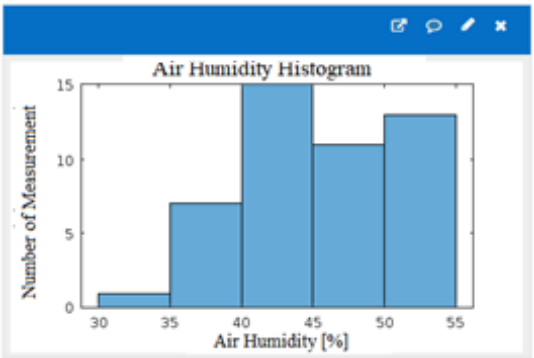
Table I

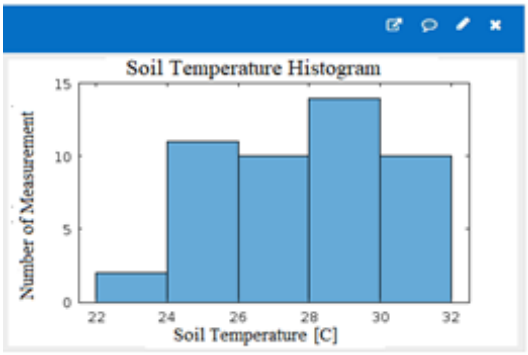
Physical phenomenon	Average value	Maximum value	Minimum value
Air temperature [°C]	28.54	31.5	24.6
Air humidity [%]	45.42	53.1	34.9
Soil temperature [°C]	27.56	31	22.5
Soil humidity [%]	68.12	82	1
CO2 concentration [ppm]	463.7	473	452
Precipitation amount [l/mp]	3.25	8	0

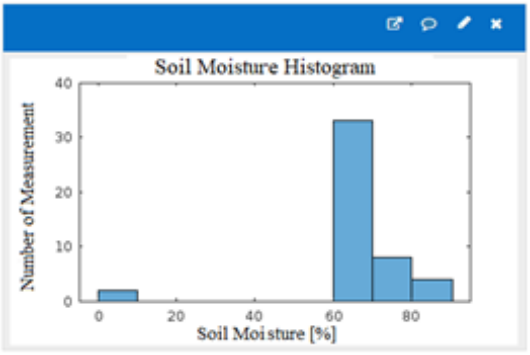
Another important analysis of the data provided by a weather station for monitoring agricultural crops is the display of the number of data that are within a certain range, to determine this we made a MATLAB code that makes a histogram of each magnitude acquired by the weather station proposed and developed in the project in the last 10 days.
Defining the function for displaying the air temperature histogram of the last 10 days consisted of setting the ThingSpeak channel from which the air temperature reading is desired (defining the ID and APIkey of the channel of the IoT weather station for agricultural crop monitoring). The next step in defining the function was to read the air

temperature of the last 10 days from the specific air temperature field (field 1 of the ThingSpeak channel of the IoT weather station for agricultural crop monitoring).
The last step in defining the function for displaying the histogram of the air temperature of the last 10 days was to call the specific Matlab function (histogram) that performs the histogram of the read values, i.e. the display of the histogram performed.
Below you can see the histogram of air temperature, air humidity, soil temperature and soil humidity.

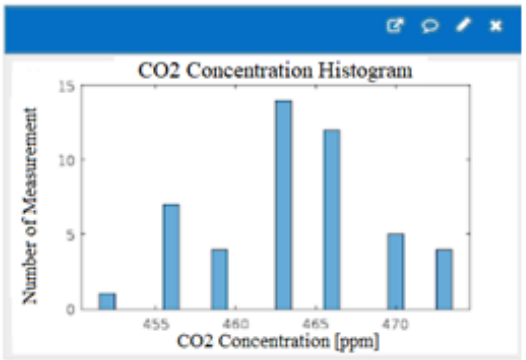


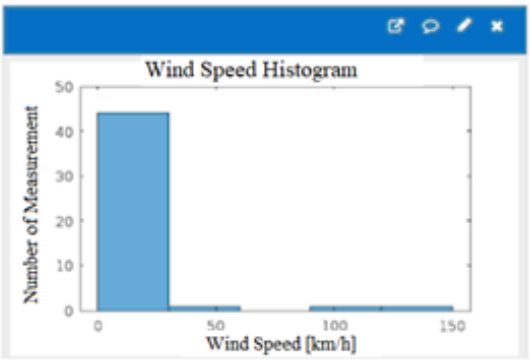


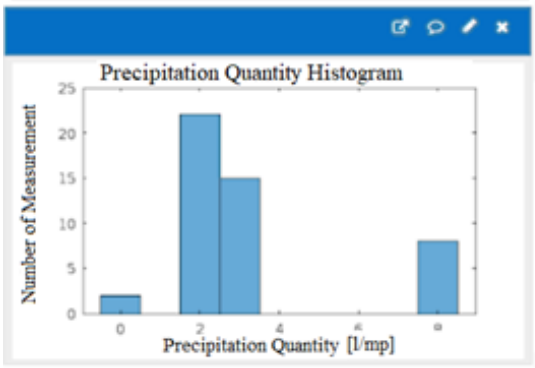


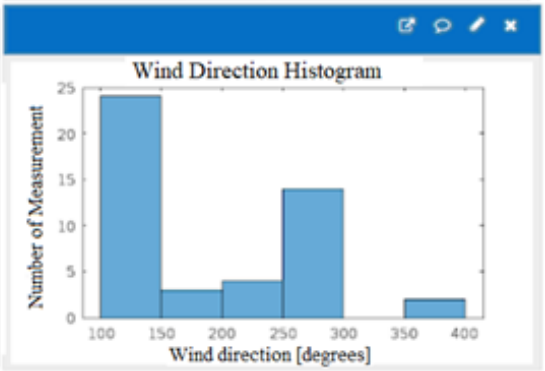


Histograms of CO2 concentration, wind speed, precipitation quantities, and wind direction can be seen below.









4. Conclusions

In this paper, we proposed and developed a low-cost, sustainable, and multi-functional IoT weather station variant for agricultural crop monitoring.
Possible further developments of the IoT weather station for agricultural crop monitoring developed and proposed in this work are:
- Adding more sensors to measure atmospheric pressure, solar radiation and leaf humidity;
- Improving the application code so that the station is able to provide weather forecasts for a number of days;
- Comparing the results provided by the weather station with an accurate weather station, so that the output of the station can be made more efficient;
- Energy efficiency of the weather station (i.e. the weather station remains in sleep mode when not transmitting data);
- Sending alerts to the farmer if a certain phenomenon has an extreme value.

References

[1] B. Srinivas Rao, K. Srinivasa Rao and N. Ome, " Internet of Things (IOT) Based Weather Monitor system", International Journal of Advanced Research in Computer and Communication Engineering, vol. 5, no. 9, pp. 312–319, September 2016.
[2] Bogdan, M. „About the Smart Weather Station", ACTA Universitatis Cibiniensis Technical Series, ISSN 583-7149, DOI: 10.1515/aucts-2016-0006, p. 26-29, published by De Gruyter Open, December 2016.
[3] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," IEEE Commun. Surv. Tutorials, vol. 16, no. 1, pp. 414–454, 2014
[4] Daoliang Li, Yingy Chen, "Computer and Computing Technologies in Agriculture", Springer, 2010
[5] Jeyeon Kim, Daichi Minagawa, Daiki Saito, Shinichiro Hoshina and Kazuya Kanda, "Development of KOSEN Weather Station and Provision of Weather Information to Farmers", Journal Sensors, 2022
[6] https://www.pakbelgium.com/importance-weatherstations/

DOI: 10.2478/ijasitels-2023-0006

An Overview on Sound Features in Time and Frequency Domain

Constantin CONSTANTINESCU[1], *Remus BRAD*[1]

[1]*Computer Science and Electrical and Electronics Engineering Department, Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania*

{ctin.constantinescu, remus.brad}@ulbsibiu.ro

Abstract

Sound is the result of mechanical vibrations that set air molecules in motion, causing variations in air pressure that propagate as pressure waves. Represented as waveforms, these visual snapshots of sound reveal some of its characteristics. While waveform analysis offers limited insights, audio features provide a quantitative and structured way to describe sound, enabling data-driven analysis and interpretation. Different audio features capture various aspects of sound, facilitating a comprehensive understanding of the audio data. By leveraging audio features, machine learning models can be trained to recognize patterns, classify sounds, or make predictions, enabling the development of intelligent audio systems. Time-domain features, e.g., amplitude envelope, capture events from raw audio waveforms. Frequency domain features, like band energy ratio and spectral centroid, focus on frequency components, providing distinct information. In this paper, we will describe three time-domain and three frequency-domain features that we consider crucial and widely used. We will illustrate the suitability of each feature for specific tasks and draw general conclusions regarding the significance of sound features in the context of machine learning.

Keywords: sound, feature extraction, machine learning

1 Introduction

A sound is the result of mechanical vibrations originating from an object. When an object vibrates, it imparts kinetic energy to the surrounding air molecules. These molecules, influenced by the vibrating object, undergo oscillations, moving back and forth. As a result, localized variations in air pressure occur, manifesting as compressions and rarefactions, which propagate through the air as a pressure wave. A sound wave is a type of mechanical wave, representing the movement or vibration that travels through the air due to objects vibrating. This vibration sets the air molecules in motion, causing changes in air pressure that ripple through the air, carrying energy from one point to another. In simple terms, a sound wave is the means by which energy from vibrating objects is transported through the air, perceived by us as sound.

The most common representation of sound is the waveform. A waveform is a visual representation of sound, showing how air pressure changes over time. Think of it as a line graph where time is on the horizontal axis, and the vertical axis represents air pressure. When a sound occurs, this line wiggles up and down, corresponding to variations in air pressure caused by the sound. The shape of the line, or waveform,

provides information about the sound's characteristics. For example, if the line on the graph repeats in a regular and smooth pattern, it indicates a periodic sound, like a musical note. On the other hand, if the line is more jagged and irregular, it represents an aperiodic sound, such as noise. The waveform serves as visual snapshot of the sound, allowing us to study and understand its properties, like frequency, amplitude, and duration. It is essentially a fingerprint for sounds, aiding in identification and analysis.

Amplitude quantifies how strong or loud the sound or signal is. In a waveform representation, it's often observed in the height of the peaks or the depth of the troughs from the central axis. Larger amplitude signifies a stronger, louder signal, while a smaller amplitude represents a weaker, quieter signal.
An audio signal is a representation of sound that encodes all the information necessary to faithfully reproduce the sound it represents. Analog signals have continuous values for both time (on the x-axis) and amplitude (on the y-axis), while digital signals represent data in the form of a sequence of discrete values. These discrete values are finite in number and are used to encode information. In the context of digital signals, both the timing (time) and the amplitude of the signal are quantized into discrete, specific values.

By looking at the sound waveform we can only have a basic idea about the sound with very limited information. Audio features provide a quantitative and structured way to describe sound, enabling data-driven analysis and interpretation. Different audio features capture various aspects of sound, such as pitch, timbre, rhythm, and intensity, facilitating a comprehensive understanding of the audio data. By leveraging audio features, machine learning models can be trained to recognize patterns, classify sounds, or make predictions, enabling the development of intelligent audio systems for applications like speech recognition, music genre classification, and sound analysis.

Audio features are categorized by the domain they operate in, such as time-domain, frequency domain, or specialized domains like the cepstral domain. This is the most important strategy that we have for categorizing different audio features.

There are certain audio features that are in the time-domain. Some of these are amplitude envelope, root-mean square energy and zero crossing rate. They are extracted from a waveform, from the basic raw audio, capturing events over time.

The sound is also characterized by frequency. The frequency is an extremely important descriptor of sound. So, there are other features that go under the name of frequency domain features, which focus on the frequency components of sound. Some of these are band energy ratio, spectral centroid and spectral flux, frequency domain features providing information not readily available in the time-domain.

2 Time domain features

2.1 Understanding time domain features

The core concept in the time domain revolves around waveform analysis, where the waveform visually illustrates the changing amplitude of a signal as it progresses through

time. In a waveform, the x-axis represents time, and the y-axis represents amplitude. Each point on the waveform corresponds to a specific moment in time, providing a direct visualization of the signal's behaviour. This allows us to observe all the events that occurred in a sound over time. Features that extract information from this representation are called time-domain audio features.
For the extraction pipeline, we will consider an analog sound, such as the sound of a violin or ambient noise. First, we want to convert that sound using the ADC (Analog digital conversion) process. This involves sampling and quantizing the analog sound to obtain a digital signal. Once we have the digitalized version of that sound, the next step will be the framing. Framing means that we want to bundle together a bunch of samples. Frames in audio processing serve as fundamental units of analysis, breaking down continuous audio signals into manageable segments. These segments are essential for various tasks, including audio analysis, feature extraction, and signal processing. In other words, frames represent chunks of audio that are perceptible to the human ear. These chunks are chosen to be of a duration that makes sense in the context of the analysis, such as a fraction of a second.
If we take a look at one sample at the sampling rate of 44.1 KHz (sample rate for the CD ROM), we find out that that sample has a duration (inverse of the sample rate) of 0.0227ms. The duration of this single sample is below the threshold of the time resolution of the human hearing (around 10ms). All the things that are below 10ms, the human ear cannot appreciate them as acoustic events. So with frames, we will have enough duration of an audio signal so that we can appreciate that.
To optimize computational efficiency, frame sizes often adhere to a power of 2 for the number of audio samples within each frame. The size of frames can vary depending on the specific application and analysis requirements. Common frame sizes typically range from 256 to 8192 samples. Smaller frame sizes, such as 256 samples, are suitable for capturing rapid changes in audio, while larger frame sizes, such as 8192 samples, are more appropriate for analyzing longer-term audio characteristics.

2.2 Amplitude envelope

The amplitude envelope of a sound frame is defined as the maximum amplitude value among all the individual samples within that frame. Figure 1 [1] illustrates a segment of a sound represented as a waveform. The waveform is divided in 6 frames while the amplitude envelope is marked with the green small rectangles. Only those values are used in this case to represent this sound. It is worth mentioning that, for clarity, in this case the waveform is divided into this six frames. In reality, frames are much smaller so we would have a lot more values for the amplitude envelope.

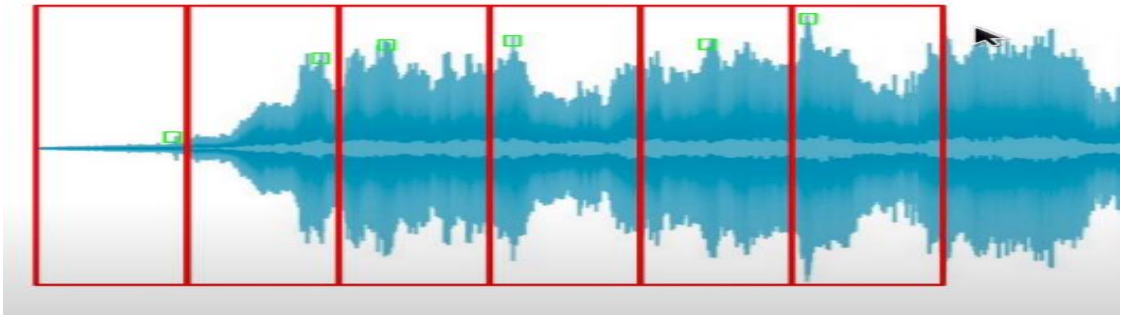


Figure 1. Visual representation of the amplitude envelope of a sound [1]

The formula for calculating the amplitude envelope for one single frame looks like this:

$$\mathrm{AE_t} = max_{k=t\cdot K}^{(t+1)\cdot K-1} s(k) \quad (1)$$

Where:

- t is the frame
- AE_t represents the amplitude envelope at frame t
- K is the frame size (number of samples we have in a frame)
- $k=t*K$ is the first sample of frame t
- $(t+1)*K-1$ is the last sample of the frame t
- $s(k)$ represents the amplitude of the k-th sample

As expected, the formula (1) is a maximum function to determine the highest value within the range of samples that belong to that frame.
The biggest problem of this feature is that it tends to be influenced by extreme values or outliers, because we only consider one value (the maximum), one sample from every frame that might not represent the sound correctly. Imagine a frame where all the values are close to zero, with only one spike at a considerably higher value. The amplitude envelope would only consider that spike, while all other small values from that frame would not have any influence on it. In this case, the value of that spike is not representative for the entire frame.

Applications of AE

The amplitude envelope provides a general sense of loudness, since the amplitude is directly related to intensity of a sound. Its primary application lies in onset detection, where it is utilized to identify the moment a particular note or acoustic event begins, by capturing sudden changes in the signal [2]. Figure 2 demonstrates the efficacy of the amplitude envelope in event detection, showcasing the waveform of a recording featuring a functioning valve in an industrial setting. In this instance, the amplitude envelope (red) serves to identify the moment when the valve is opened or closed, clearly discernible through the spikes in the envelope.

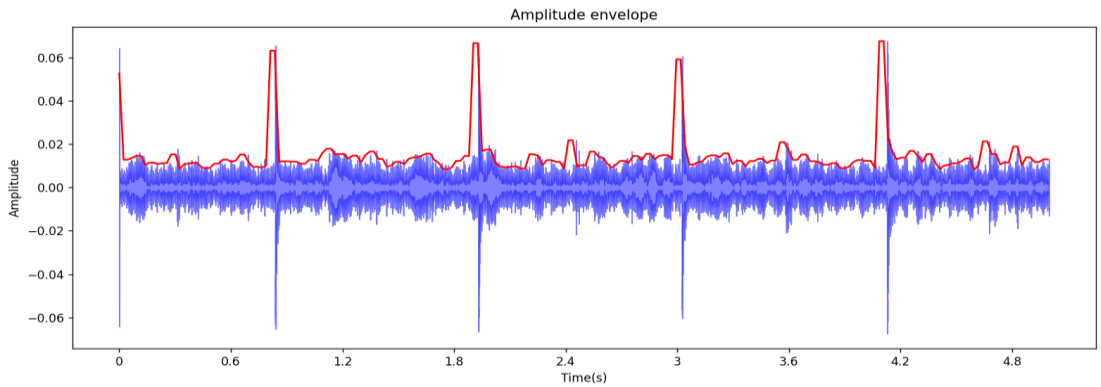


Figure 2. Visual representation of the sound of a working valve and its AE

A big part of the interpretation of sound events rests on duration, a property that provides important information about the materials and actions involved in it. Using amplitude envelope we can estimate the duration of the event and also determine if it had a natural ending or an abrupt one [3]. The difference between percussive and flat envelopes has been extensively researched as they occur not only in duration estimation, but also in tasks like audio-visual integration [4] or associative memory [5]. Percussive sounds, usually heard when two objects collide, carry a lot of information about the event. Flat sounds in contrast are characterized by a period of sustained amplitude with abrupt onsets and offsets. This indicates, that flat sounds usually do not come from nature, but are rather synthetically generated (for example alarms) [6].
Apart from onset/offset detection this feature can be used in a lot of other tasks as well, like for example music genre classification.

2.3 Root-mean square energy

The Root-mean-square energy is the root mean square (RMS) of all samples in a frame. In other words, root-mean-square energy is calculated as the square root of the mean of the squared values of all samples within a frame.
The formula (2) for calculating the root-mean-square energy for one single frame:

$$\mathrm{RMS_t} = \sqrt{\frac{1}{K} \cdot \sum_{K=t\cdot K}^{(t+1)\cdot K-1} s(k)^2} \tag{2}$$

Where:

- *s(k)* is the amplitude at sample *k* (the square of the amplitude is the energy)
- We sum the energy for all the samples in frame *t*
- We take the mean of the sum of the energy

So, the root-mean-square energy is the square root of the mean of the sum of energy.
Root-mean-square (RMS) energy is a measure commonly used as an indicator of loudness in audio processing. Unlike the Amplitude Envelope (AE), RMS energy is less sensitive to extreme values or outliers in the signal. It calculates the square root of the mean of the squared values of all samples within a frame, as opposed to the AE where only one value was taken into consideration. This is why RMS is providing a more stable representation of the signal's energy.

Application of the RMSE

RMS energy is particularly useful in tasks such as audio segmentation, where identifying different sections or events in an audio signal is crucial. For example, when we want to decide whether someone is talking or not, and we have that change in the sound signal and want to segment who is talking. In figure 3, we can observe how the RMS tends to zero when there is a pause in speech.

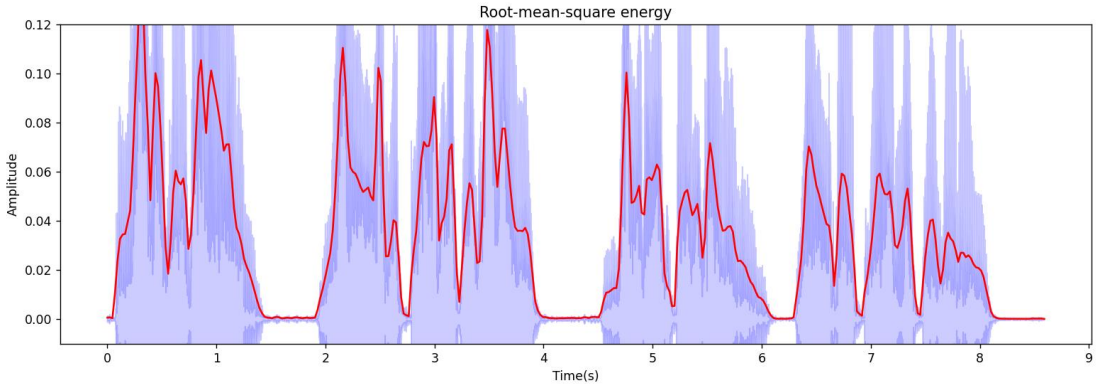


Figure 3. Visual representation of the RMSE (red) of a speech recording

Specifically, RMS of sound has been used in the underwater detection of spiny lobster [7] to establish sound pressure levels and generally in research about underwater animals in relation to sound. In medicine, RMS was also used in combination with other features in automatic diagnosis of COVID-19 from respiratory sound data [8].
Additionally, it plays a role in music genre classification, helping to characterize and distinguish the overall loudness characteristics of different music genres.

2.4 Zero crossing rate

The zero crossing rate provides information about the number of times a signal crosses the horizontal axis.
In Figure 5 [1] we have the visualization of a signal wave of a frame. The zero crossing rate is equal to the count of the green dots. For each of these green dots, we have a crossing of the horizontal axis. For this signal, the zero crossing rate is equal to 6.

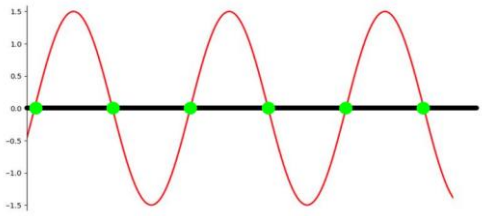


Figure 4. Visual representation of the zero crossing [1]

How do we calculate the zero crossing rate mathematically? The formula (3) for calculating the zero crossing rate:

$$\mathrm{ZCR_t}=\frac{1}{2}\sum_{k=t\cdot K}^{(t+1)\cdot K-1}\left|sgn\big(s(k)\big)-sgn\big(s(k+1)\big)\right| \tag{3}$$

The basic idea here is that we compare the amplitude value for consecutive pairs of samples and we look whether there are differences in the signs of those amplitudes for those consecutive samples.
For this we are going to use the sign function. The sign function *sgn(s(k))* gives back the sign of a given value. In this case, if the amplitude *s(k)* is greater than zero, the sign function gives us back the value "+1" because the amplitude is positive. If the amplitude is negative, we get back the value "-1" and if the amplitude is equal to zero, then we get back the value "0".

But how do we calculate the zero crossing rate for each pair of samples? We take the sign for the amplitude at sample k and then we subtract the sign for the amplitude at sample *k+1*. Basically we are comparing the two consecutive samples.
If both amplitude values have the same sign, we get 0 on that side, so we don't get any value that is going to increase the zero crossing rate.
But if we have alternate opposite signs, for example at sample k we have a negative amplitude and at sample *k+1* we have a positive amplitude, we will get a value of 2.
In this formula (3), we sum over all the samples that we have in a frame so that we are going to get the values for all the zero crossings that we have in a frame.

Applications of the ZCR

Zero Crossing Rate (ZCR) holds significance in diverse applications. It aids in distinguishing between percussive and pitched sounds, contributing to the recognition of distinct auditory characteristics. This is because percussive sounds usually tend to have a quite random zero crossing rate, so they tend to change the zero crossing rate a lot, whereas pitch sounds tend to be way more stable in zero crossing rates.
In the field of audio signal processing, ZCR is employed for monophonic pitch estimation, offering insights into the fundamental frequency of a single musical note or sound. There is a relationship between the number of zero crossings and the pitch. And if we have a monophonic pitch, we can observe that the higher the number of zero crossings that we have, the higher the pitch is going to be.
In the context of speech recognition, the zero crossing rate can be utilized to distinguish between signals containing voice and those that are unvoiced [9]. Typically, voice signals exhibit a lower zero crossing rate compared to unvoiced signals. This distinction may be attributed to the fact that unvoiced segments tend to be noisier, resulting in a higher zero crossing rate [10].

3 Frequency domain features

3.1 Understanding frequency domain features

The initial steps for obtaining a frequency domain representation of the sound mirror those found in the time-domain feature pipeline. We begin with an analogue sound, apply sampling, followed by quantization to obtain the digitized version of the sound (audio signal). Next, we frame the signal, resulting in a series of frames. At this stage, we have a waveform divided into multiple frames. However, how do we transition from this time-domain representation to the frequency domain? The solution lies in applying the Fourier transform, which translates the signal from the time-domain to the frequency domain. Unfortunately, there is a problem we encounter when we do that, and this is the spectral leakage.
In signal processing, spectral leakage arises when processing a signal that lacks an integer number of periods. In real life, most sounds are not periodic so the endpoints of the frames are discontinuous. This leads to additional high-frequency components in the signal's spectrum, introduced by the abrupt changes or discontinuities. Windowing, a technique in signal processing, addresses this issue by applying a specific windowing function to each frame of an audio signal. The Hann window, a widely used windowing

function, is a bell-shaped curve that tapers the signal, reducing spectral leakage by smoothing transitions at the frame edges.
The application of the Hann window involves multiplying it by the original signal at each corresponding sample, effectively eliminating discontinuities. However, we encounter some signal loss when gluing multiple windowed signals together.
To address this, overlapping frames are introduced. In non-overlapping frames, a frame size is applied to the signal without any overlap, so the current frame just follows the previous one as shown in figure 1 [1]. Overlapping frames, on the other hand, involve applying a frame size to the signal with an overlap. The hop length, representing the number of samples shifted to the right for each new frame, allows for minimizing spectral leakage in the Fourier Transform of the windowed signal.
Following this transformation, we acquire a spectrum with minimized spectral leakage, where the x-axis represents frequency and the y-axis represents magnitude.
However, this representation now lacks information about time. To capture both time and frequency information, we need to apply the Short-Time Fourier Transform (STFT) on the waveform, which produces a spectrogram—a representation of sound that incorporates details about both time and frequency.

3.2 Band energy ratio (BER)

This feature provides us information about the relation between the energy in the lower and higher frequency bands. We can consider it as a measure of how dominant the lower frequencies are. To understand how it works, we need to understand the math behind it. The formula (4) for the band energy ratio is as follows:

$$\mathrm{BER_t} = \frac{\sum_{n=1}^{F-1} m_t(n)^2}{\sum_{n=F}^{N} m_t(n)^2} \tag{4}$$

We can observe that the formula is a fraction, which is expected since we are dealing with a ratio of two elements. In both the numerator and denominator, we have the power of the signal at time *t* and frequency bin *n*. The power is essentially the squared magnitude of the signal. The capital *F* in the sum function represents the split frequency, serving as a threshold that distinguishes higher frequencies from lower frequencies. The choice of the split frequency is arbitrary and depends on the application, with a common value being 2000 Hz.
So, in the upper part of this fraction we have the power in the lower frequency bands at a specific point in time, whereas at the denominator we have the opposite of that, meaning the power in the higher frequency bands. It is worth mentioning that the result will be the band energy ratio at a specific frame, so the formula has to be applied to each frame.
The band energy ratio can be used in all sorts of things in music and speech processing, but specifically it has been extensively used to discriminate music from speech, for certain music classification problems, like music genre classification or mood classification.

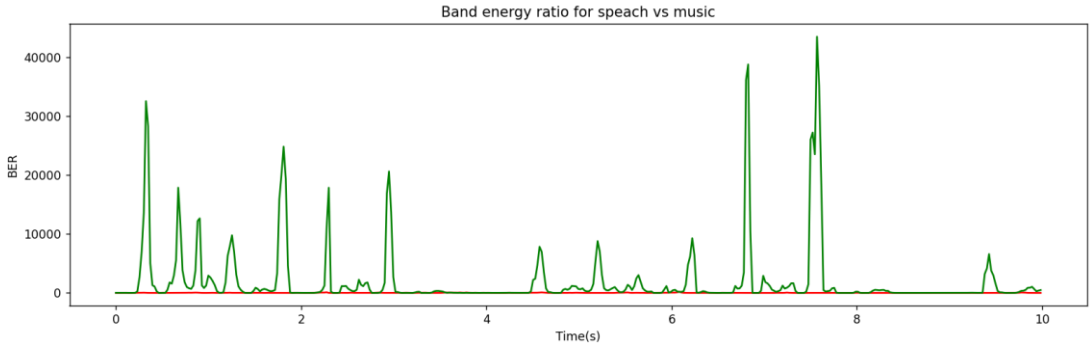


Figure 5. Visual representation of BER of a speech (green) versus music (red)

In Figure 5, we can observe that the band energy ratio (BER) for speech, indicated by the red line, exhibits significantly lower values compared to the BER for a musical piece, represented by the green line. This illustrates one of the most commonly applied uses of the band energy ratio in sound processing.
But music and voices are only 2 classes of sounds and the band energy ratio showed very good results in environmental sound recognition and classification experiments with more than ten classes [11] that include for example the inside of a running car or the sound of a forest. [12]

3.3 Spectral centroid (SC)

A very common and famous feature is the spectral centroid. It is going to provide us the center of gravity of the magnitude spectrum [13]. In other words, it will give us the frequency band where most of the energy is concentrated. It maps onto a very prominent timbral characteristic of the sound, being a measure of "brightness", meaning it will tell us how open or dull a certain sound is. The spectral centroid is essentially the weighted mean of the frequencies.

$$SC_t = \frac{\sum_{n=1}^{N} m_t(n) \cdot n}{\sum_{n=1}^{N} m_t(n)} \tag{5}$$

The weights are represented by the magnitude at a specific frequency bin, whereas the rest of the formula(5) is a basic mean. Like the band energy ratio, we calculate the spectral centroid for each frame.
Figure 6 highlights the limitations of information obtained solely from a waveform, underscoring the increased utility of frequency domain features, such as the spectral centroid. Specifically, we examine a recorded lung sound captured using an electronic stethoscope. The spectral centroid provides a clearer view of the patient's respiratory cycles, a distinction that is challenging or nearly impossible to discern in the waveform alone. This insight proves valuable, particularly for tasks like segmentation. Instead of blindly partitioning the sound into arbitrary sections, leveraging the spectral centroid allows for precise division into complete respiratory cycles. This approach ensures that essential information about the patient's respiration is preserved, enhancing the accuracy and relevance of the analysis.

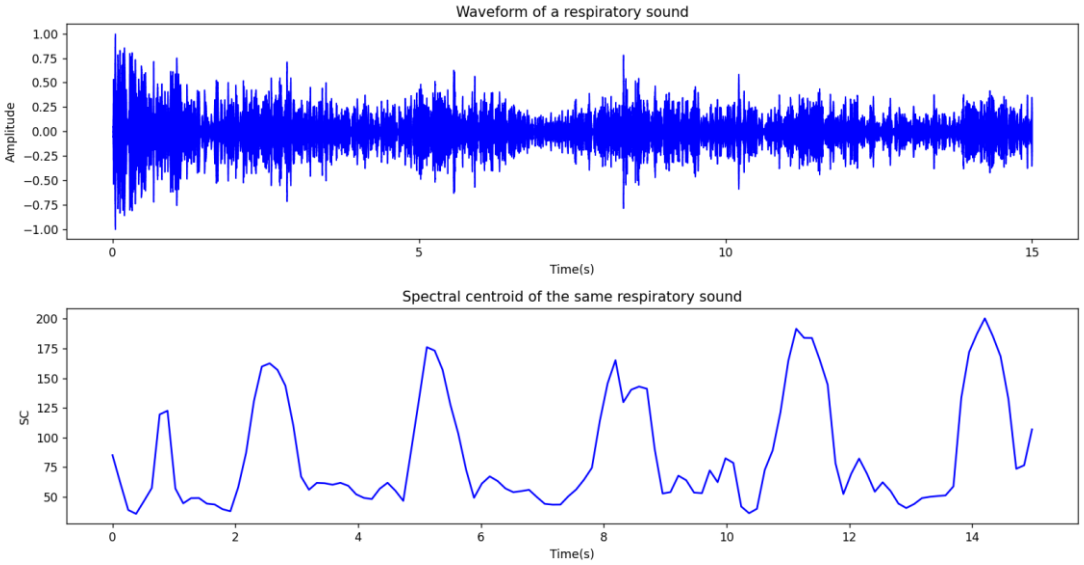


Figure 6. Waveform representation and spectral centroid of a lung sound recording

During the pandemic, it was used in the exploring of automatic diagnosis of COVID-19 from sound data [8] but also generally in medicine to process sounds like cough [14], voice and breathing [15]. The spectral centroid is also very much used in audio classification or music classification problems, usually with classic machine learning techniques. It is one of the key frequency domain audio features.

3.4 Bandwidth (BW)

The bandwidth is somewhat related to the spectral centroid meaning that we can think of the bandwidth as that spectral range, which is of interest and is around the centroid. In simpler terms, the bandwidth represents the variance from the spectral centroid and has a correlation with the perceived timbre. Similar to the spectral centroid, it is a weighted mean. However, in this case, it is not a weighted mean of the frequencies but rather a weighted mean of the distances of frequency bands from the spectral centroid.

$$BW_t = \frac{\sum_{n=1}^{N} |n - SC_t| \cdot m_t(n)}{\sum_{n=1}^{N} m_t(n)} \quad (6)$$

The weights are once again the magnitude for the signal at the specific time frame *t* and the specific frequency band *n*, while the distance between the frequency band and the spectral centroid is represented by the absolute value in the formula(6). Consequently, the bandwidth varies based on how energy is distributed across the different frequency bands. If the energy is dispersed across the frequency bands, the bandwidth value decreases. On the opposite, if the energy is concentrated in only a few frequency bands, the bandwidth value decreases. It is evident that it measures how much the energy is spread, and for this reason, the bandwidth is also referred to as spectral spread.
Bandwidth has been extensively employed in music processing, including applications like music genre classification or music mood classification.

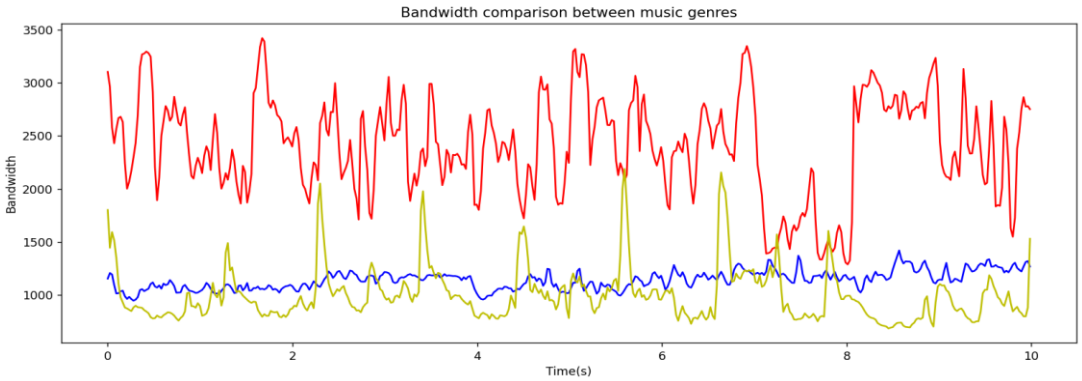


Figure 7. Visual representation of the BW for rock music(red), classical music(blue), jazz music(yellow)

Figure 7 illustrates the bandwidth for 3 music genres: rock, classical and jazz. We can observe that the bandwidth is much higher for rock music than the other genres, proving its utility in music genre classification.
It has also applications in medicine, for example in pre-processing voice sounds, to prepare them for anomaly detection like intoxication [16] or various medical conditions.

4 Experimental results

In order to demonstrate some of the theoretical notions explained and some of the conclusions we drew in this paper, we conducted an experiment. Given that most features find applications in sound classification, we utilized each of these features to classify industrial machine sounds. The sounds were sourced from the MIMII Dataset [17], which contains labeled sounds from four types of machines: fans, pumps, valves and sliders. While the dataset was primarily constructed for anomaly detection, our focus involved approximately 1000 normal sounds (without anomalies) from each machine type, at a signal-to-noise ratio of 6 dB. Each sound was saved as a WAV file and lasted for 10 seconds. In terms of pre-processing, we resampled each sound to 16 kHz and divided it into frames of 1024 samples with an overlap of 512 samples.
With the data prepared, we extracted every feature presented in this paper. As all features were calculated for each frame, the result was six arrays (one for each feature), each containing another 4000 arrays, one for each particular sound. Since all the sound files were 10 seconds long, each array had the same length.
For the classification, we chose a supervised approach, using the classic machine learning algorithm KNN [18] with a *k*-value of 3 neighbours [19]. The data was split into training and testing samples, with the test samples constituting 30% of the entire dataset. We evaluated the algorithm using three metrics as shown in Table 1 with their corresponding results.
In this particular case, the algorithm was able to better distinguish between fan, pump, slider and valve when using the time domain features, with the RMS standing out. This can be explained by the fact that these industrial machines all operated in low frequency but varied in amplitude. We ran the algorithm multiple times with various *k* values, resulting in slightly different outcomes, but the proportion between features remained consistent.

	Accuracy (%)	Precision (%)	F1-Score (%)
AE	79,07	87,30	74,55
RMSE	96,40	96,46	96,46
ZCR	70,89	75,22	63,29
BER	50,61	56,63	41,96
SC	70,32	79,19	70,02
BW	59,04	52,90	52,32

Table 1. KNN Algorithm evaluation results

5 Conclusions

Classic machine learning algorithms, such as decision trees, support vector machines, and k-nearest neighbors, heavily rely on sound features for tasks like audio classification and genre recognition. In traditional machine learning, feature extraction is a critical process, with audio features encompassing Amplitude Envelope, Root-Mean-Square Energy, Zero Crossing Rate, Band Energy Ratio, Spectral Centroid, Spectral Flux, Spectral Spread, Spectral Roll-Off, and more. The selection of features depends on their suitability for the specific problem at hand.

For sound classification, one might choose features like amplitude envelope, zero-crossing rate, and spectral flux. These selected features are isolated, extracted from audio files, and fed into traditional machine learning algorithms like support vector machines for training.

Deep learning techniques, including neural networks like CNNs and RNNs, are increasingly employed for complex tasks such as speech recognition, music generation, and audio synthesis. In deep learning, unstructured audio representations, such as raw audio or spectrogram-like features, can be passed to systems that leverage neural networks to automatically extract relevant features from the audio data. While deep learning eliminates the mandatory feature extraction step, audio features can still be valuable in the preprocessing phase, so the utility of these features should not be overlooked.

In the context of sound anomaly detection, we do not always have the necessary amount of data for the deep learning approach, especially since anomalous sounds occur rarely and unexpectedly, which makes them hard to record. For this reason, we may have to employ classic ML algorithms to perform the necessary tasks. Depending on the task, extracting the appropriate feature or combining some of the features presented in this paper will be necessary and will improve the results. Furthermore, regardless of the approach, the raw sound has to go through some preprocessing steps, like segmentation, smoothing or filtering, steps where these features prove their utility.

Sound is part of every aspect and moments of our lives so the need to process it automatically is undeniable. Regardless if we need to detect anomalies in sound of an industrial machine or in the respiratory auscultation of a patient, these audio features are valuable tools to achieve our goals.

References

[1] V. Velardo, "https://github.com/musikalkemist/AudioSignalProcessingForML," 10 10 2020. [Online]. Available: https://github.com/musikalkemist/AudioSignalProcessingForML. [Accessed 27 11 2023].

[2] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies and M. B. Sandler, "A tutorial on onset detection in music signals," *IEEE Transactions on Speech and Audio Processing,* pp. 1035-1047, 2005.

[3] G. T. Vallet, D. I. Shore and M. Schutz, "Exploring the role of the amplitude envelope in duration estimation," *Perception,* vol. 43, no. 7, pp. 616-630, 2014.

[4] . L. Chuen and M. Schutz, "The unity assumption facilitates cross-modal binding of musical, non-speech stimuli: The role of spectral and amplitude envelope cues," *Attention, Perception, and Psychophysics,* pp. 1512-1528, 2016.

[5] M. Schutz , J. Stefanucci, S. Baum and A. Roth, "Name that percussive tune: Associative memory and amplitude envelope," *Quarterly Journal of Experimental Psychology,* pp. 1323-1343, 2017.

[6] S. Sreetharan, J. Schlesinger and M. Schutz, "Decaying amplitude envelopes reduce alarm annoyance: Exploring new approaches to improving auditory interfaces," *Applied Ergonomics,* 2021.

[7] Y. Jézéquel, L. Chauvaud and J. Bonnel, "Spiny lobster sounds can be detectable over kilometres underwater," *Sci Rep 10,* 2020.

[8] C. Brown, J. Chauhan, A. Grammenos, J. Han, A. Hasthanasombat, D. Spathis, T. Xia, P. Cicuta and C. Mascolo, "Exploring Automatic Diagnosis of COVID-19 from Crowdsourced Respiratory Sound Data," *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20),* pp. 3474-3484, 2020.

[9] G. Sharma, K. Umapathy and S. Krishnan, "Trends in audio signal feature extraction methods," *Applied Acoustics,* vol. 158, 2020.

[10] Y. A. Ibrahim, J. C. Odiketa and T. S. Ibiyemi, "Preprocessing technique in automatic speech recognition for human computer interaction: an overview.," *Ann Comput Sci Ser,* vol. 15, no. 1, pp. 186-191, 2017.

[11] S. Chu, S. Narayanan and C.-C. J. Kuo, "Environmental Sound Recognition With Time–Frequency Audio Features," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 17, no. 6, pp. 1142-1158, 2009.

[12] S. Sivasankaran and K. Prabhu, "Robust features for environmental sound classification," *IEEE International Conference on Electronics, Computing and Communication Technologies,* pp. 1-6, 2013.

[13] F. Alías, . J. C. Socoró and X. Sevillano, "A Review of Physical and Perceptual Feature Extraction Techniques for Speech, Music and Environmental Sounds," *Applied Sciences,* vol. 6, no. 5, 2016.

[14] R. Islam, E. Abdel-Raheem and M. Tarique, "A study of using cough sounds and deep neural networks for the early detection of Covid-19," *Biomedical Engineering Advances,* vol. 3, 2022.

[15] A. Hassan, I. Shahin and M. B. Alsabek, "COVID-19 Detection System using Recurrent Neural Networks," *International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI),* pp. 1-5, 2020.

[16] A. B S, S. R. Shetty, S. Srinivas, V. Mantri and V. R. B. Prasad, "Intoxication Detection using Audio," *2023 IEEE 8th International Conference for Convergence in Technology (I2CT),* pp. 1-6, 2023.

[17] H. Purohit, R. Tanabe, K. Ichige, T. Endo, Y. Nikaido, K. Suefusa and Y. Kawaguchi, "MIMII Dataset: Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection," in *Proc. 4th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2009.

[18] R. Cretulescu and D. Morariu , Tehnici de clasificare si clustering al documentelor, Cluj Napoca: Editura Albastra, 2012.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

DOI: 10.2478/ijasitels-2023-0007

Sentiment Analysis Using Bert Model

Dorca Manuel-Ilie[1], Pitic Antoniu Gabriel[1], Crețulescu Radu George[1]
[1]Computer Science and Electrical and Electronics Engineering Department, Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania
{manuel.dorca, antoniu.pitic, radu.cretulescu} @ulbsibiu.ro

Abstract

The topic of this presentation entails a comprehensive investigation of our sentiment analysis algorithm. The document provides a thorough examination of its theoretical underpinnings, meticulous assessment criteria, consequential findings, and an enlightening comparative analysis. Our system makes a substantial contribution to the field of sentiment analysis by using advanced techniques based on deep learning and state-of-the-art architectures.

Keywords: Sentiment analysis, Sentiment polarity, Emotion prediction, BERT model

1. Introduction

Sentiment analysis, an essential aspect of natural language processing, is becoming increasingly important in understanding human emotions expressed through text data. The increasing amounts of social media content, consumer feedback, and various textual datasets highlight the need for advanced sentiment analysis algorithms. We address this need by presenting a methodology that utilizes advancements in machine learning and deep learning.
Our program aims to analyze feelings with a different approach, beyond traditional methods. The increasing abundance of text data on social platforms and other digital spaces necessitates accurate sentiment interpretation, which is crucial for various applications such as market research and social media monitoring.

2. Main Work

2.1. Theoretical Fundamentals of the Proposed Method for Sentiment Analysis

Our sentiment analysis method was developed by creating a theoretical framework that combines ideas from deep learning with advanced technology approaches. Our process goes beyond just combining existing components; instead, it involves integration of several methods, taking inspiration from the most recent breakthroughs in deep learning technology. Our approach to sentiment analysis includes new elements such as dynamic lexicon development and creative management of subjects and emotions. The

DOI: 10.2478/ijasitels-2023-0007

theoretical foundation is to provide a versatile understanding of emotions in written content, surpassing the constraints of conventional methods.

The effectiveness of our solution is based on the use of a versatile dictionary expansion mechanism, which allows the algorithm to adjust its understanding of emotional subtleties. When combined with the neural network structure explained in the paper, this collaboration creates a strong basis for accurately identifying emotions. Our program goes beyond simply copying current methods; it incorporates insights obtained from the complexities described in the study, thus advancing sentiment analysis into a more practical realm. This approach represents a significant shift in the way sentiment analysis is conducted in the current field of machine learning, utilizing advanced techniques to effectively handle the changing requirements.

2.2. Analysis of the Kaggle Sentiment140 Dataset

2.2.1. Summary of the Dataset

The Kaggle Sentiment140 dataset [7] is a highly utilized resource in sentiment analysis, comprising of tweets sourced from Twitter, with each tweet being categorized as either positive or negative. The binary sentiment classification sets a definitive goal for the development and evaluation of sentiment analysis methods.

2.2.2. Dataset Source and Preprocessing

The dataset was acquired using Twitter's public API, assuring diversity across time periods and user demographics. It was curated by Stanford University. Tweets were recovered by identifying specific emoticons that indicate either good or negative attitudes. The preprocessing stage encompassed the elimination of irrelevant information, the segmentation of the text into individual tokens for further analysis, and the organization of the dataset to facilitate sentiment analysis.

2.2.3. Analysis of Sentiment Distribution and Text Length

Examining the distribution of sentiment allows for a deeper understanding of the dataset's accuracy in representing various viewpoints and the potential difficulties in classifying them. Furthermore, analyzing the lengths of tweets helps to discover limitations that could affect the effectiveness of a model, enabling researchers to adjust their approaches accordingly.

2.3. Algorithm Details

Our sentiment analysis method consists of meticulously developed steps to capture and analyze sentiment information effectively and precisely. The following sections outlines the key steps of our approach.

DOI: 10.2478/ijasitels-2023-0007

2.3.1. Topic Annotation

Our algorithm begins by performing topic annotation, which is a method aimed at identifying and isolating the main subjects or topics present in the input text. By utilizing natural language processing techniques, particularly part-of-speech tagging with the NLTK toolkit [1], our implementation guarantees a focused examination of emotional tone. We exemplify the concept by providing code snippets that showcase the process of tokenizing sentences, assigning parts of speech to words, and extracting the key themes. To address phrases that encompass many themes, we utilize the TF-IDF method, which offers a strong and effective method for identifying the most pertinent subject matter.

```
Sentence: @stellargirl I loooooooovvvvvveee my Kindle2. Not that the DX is cool
_____________________________________________________________________________
                    Topic: Kindle2
_____________________________________________________________________________

Sentence: Reading my kindle2... Love it... Lee childs is good read.
_____________________________________________________________________________
                    Topic: kindle2, Lee childs
_____________________________________________________________________________

Sentence: "Ok, first assessment of the #kindle2 ...it fucking rocks!!!"

_____________________________________________________________________________
                    Topic:kindle2
_____________________________________________________________________________
```

Figure 1 Output of topic anotation step

2.3.2. Emotion Annotation with Dynamic Lexicon Expansion

The second stage centers on emotion annotation, augmenting the comprehension of natural language and sentiment analysis. We apply a predetermined lexicon to assign emotional scores to words, utilizing the Levenshtein distance [8] technique for word comparison. The algorithm progressively enlarges the lexicon as time goes on, consistently enhancing the process of emotion annotation. We explore the intricacies of lexicon-based emotion annotation, Levenshtein distance word matching, sentiment score calculation, and sentiment mapping. The method of dynamic lexicon expansion guarantees the ability to adapt to changing language patterns, which is crucial for appropriately assigning emotion scores to words that have not been encountered before. After the algorithm has determined the word that closely corresponds, it next proceeds to allocate the emotion score linked to that word. This approach ensures that even if a term is not found in the lexicon, it can still be assigned an emotional value by evaluating its similarity to existing words. [1]

2.3.3. Emotion Categorization

Utilizing the sentiment scores calculated in the preceding step, the third phase categorizes each sentence into positive, negative, or neutral sentiment groups according to the dataset. As our objective is to predict sentiment, the general emotion of the Kaggle dataset [7] has been standardized to a scale of -1 for negative, 0 for neutral, and 1 for positive. The algorithm discerns the most pertinent emotions associated with

DOI: 10.2478/ijasitels-2023-0007

negative, neutral, and positive phrases, so offering a nuanced comprehension of the emotional context. We analyze the approach used for classifying sentiments, identifying emotions, and managing discrepancies between given categories and sentiment scores. This phase guarantees a thorough sentiment analysis, capturing the entire emotional tone of every line.

The algorithm begins by annotating the topic and then proceeds to detect and categorize the main subjects present in the textual data. This establishes a fundamental comprehension of the surrounding circumstances, setting the basis for further examination of emotions.

By annotating the subjects and emotions, the computer can ascertain the overall sentiment of the text. This stage enables a thorough sentiment analysis by combining specified subjects and emotions. The program considers the interaction between themes and emotions, guaranteeing a sophisticated evaluation.
Our method uses a sentiment mapping dictionary to establish a connection between sentiment scores and distinct emotion categories. The purpose of this dictionary is to provide clear definitions of different types of emotions and the specific ranges of scores associated with each category. For example, the sentiment category labeled as "Positive" encompasses feelings such as "*Happy*," "*Excited*," and "*Amazed*," each of which is linked to specific score ranges.

The system categorizes the emotion of a given sentence by comparing its sentiment score to predetermined ranges. When the score is within a defined range, the statement is associated with the corresponding emotion category.

2.4. Neural Network Configuration

During the fourth stage of our method, we concentrate on customizing the neural network by utilizing the BERT architecture specifically designed for sentiment analysis. The method requires evaluating several hyperparameter configurations, such as training epochs, learning rate, and batch size, in order to improve model performance. The BERT model, which is built on transformers, effectively collects contextual information in both directions and has demonstrated good performance in tasks related to natural language processing.[3][4]

The training epochs significantly impact the model's exposure to the dataset, striking a balance between acquiring knowledge and achieving generalization. Similarly, the learning rate, which determines how the model's parameters are adjusted during optimization, affects how quickly the model converges and the likelihood of overfitting. The batch size, which refers to the number of training instances processed at once, has an effect on both memory usage and computational efficiency.

Hyperparameter tuning entails a methodical examination utilizing techniques such as grid search or random search to determine the ideal configuration. The utilization of regularization strategies, which aim to mitigate overfitting, is explored. This includes the incorporation of dropout rates and weight decay. Model evaluation is based on parameters like as accuracy, precision, recall, F1 score, and AUC-ROC[5]. In order to

DOI: 10.2478/ijasitels-2023-0007

guarantee impartial assessment, the dataset is partitioned into training, validation, and test sets.

Configuring the neural network is crucial for utilizing BERT's capabilities to provide successful sentiment analysis. This involves capturing contextual nuances and accurately classifying sentiment in textual input. The method entails a thorough and systematic approach of conducting experiments, validating results, and fine-tuning in order to achieve an optimal configuration that improves the performance of the model.

Data preparation and normalization play a crucial role in preparing data for sentiment analysis models, and their importance should not be underestimated. During this pivotal phase, the algorithm converts annotated data into tensors that are compatible with the BERT neural network design. The procedure entails tokenization via the DistilBertTokenizer, encoding with padding and truncation to ensure consistent sequence length, and converting sentiment labels and subcategories into numerical representations.

Subsentiment analysis offers an extra training feature that enables the replacement of labels with appropriate subcategories, enhancing the evaluation of sentiment with more nuanced precision. The processed data is structured as dictionaries nested within a list. Each dictionary contains 'input_ids' and 'attention_mask' tensors, which represent tokenized and encoded texts. Additionally, the dictionaries include 'labels' for sentiment labels and 'topics' for supplementary topic information. The utilization of this organized structure guarantees the BERT model's ability to process and comprehend the data during both training and evaluation.

In order to enhance efficiency, the code utilizes caching mechanisms by using the *functools* library, specifically the *lru_cache* decorator. This improves the computing efficiency for repetitive tasks, particularly when dealing with large datasets.

The technique systematically investigates various configurations for the BERT neural network, simulating a wide range of parameters such as the number of epochs, learning rates, and batch sizes. The number of epochs ranges from 4 to 16, the learning rates consist of values such as 0.0002, 0.00002, 0.000002, and 0.0000002, and the batch sizes span from 8 to 256 in powers of 2. The purpose of this systematic experimentation is to determine the most effective hyperparameter settings, obtaining a good combination of training efficiency and model performance.

3. Results

3.1. Evaluation Metrics

For evaluating our sentiment analysis algorithm we used as metrics the accuracy, precision, recall, and F1 score. These metrics served as crucial indicators of the algorithm's performance in sentiment classification across diverse datasets and domains.

DOI: 10.2478/ijasitels-2023-0007

3.2. Performance Analysis

The algorithm demonstrated robustness in handling challenges such as sarcasm and sentiment shifts, showcasing its adaptability to complex sentiment analysis tasks. This performance analysis provided valuable insights into the algorithm's overall efficiency.

3.3. Comparative Analysis

Comparing our algorithm to prior research revealed accuracies ranging from 58% to 82%. While our accuracy aligns with [3], the inclusion of additional metrics such as precision and recall offered a more comprehensive evaluation.

The sentiment polarity prediction results (Table 1) show an overall mean accuracy of 67.4%, with a standard deviation of 7.4%. This indicates a moderate level of consistency in sentiment prediction. The F1 score, which balances precision and recall, is consistently high across epochs, suggesting a well-rounded performance in sentiment classification.

Table 1 Sentiment Polarity Prediction Results

	Accuracy	Precision	Recall	F1 Score
Epoch count	16	16	16	16
mean	0.67389	0.81571	0.82534	0.81833
std	0.07396	0.07346	0.05935	0.0505
min	0.57137	0.7088	0.7434	0.75164
25%	0.60813	0.75752	0.7752	0.7691
50%	0.66926	0.81826	0.81433	0.81834
75%	0.73423	0.87628	0.87819	0.86176
max	0.80001	0.91967	0.93312	0.8956

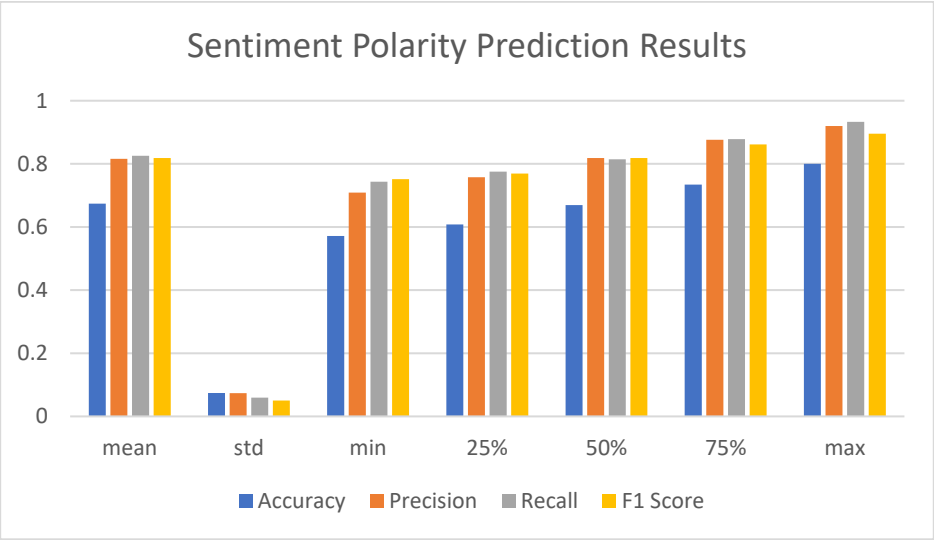


Figure 2 Sentiment Polarity Prediction Results

DOI: 10.2478/ijasitels-2023-0007

In the base emotion prediction results (Table 2), the mean accuracy of 70.2% with a standard deviation of 6.6% indicates a stable performance. The F1 score, emphasizing a balance between precision and recall, maintains a consistently high level, further reinforcing the algorithm's proficiency in emotion prediction.

Table 2 Base Emotion Prediction Results

	Accuracy	Precision	Recall	F1 Score
Epoch count	128	128	128	128
mean	0.70186	0.82147	0.84365	0.83037
std	0.06584	0.05873	0.0552	0.03977
min	0.57333	0.68264	0.72509	0.71752
25%	0.64774	0.77471	0.79864	0.80316
50%	0.70532	0.82421	0.8469	0.83147
75%	0.75617	0.87068	0.88863	0.85867
max	0.81999	0.9199	0.93967	0.92575

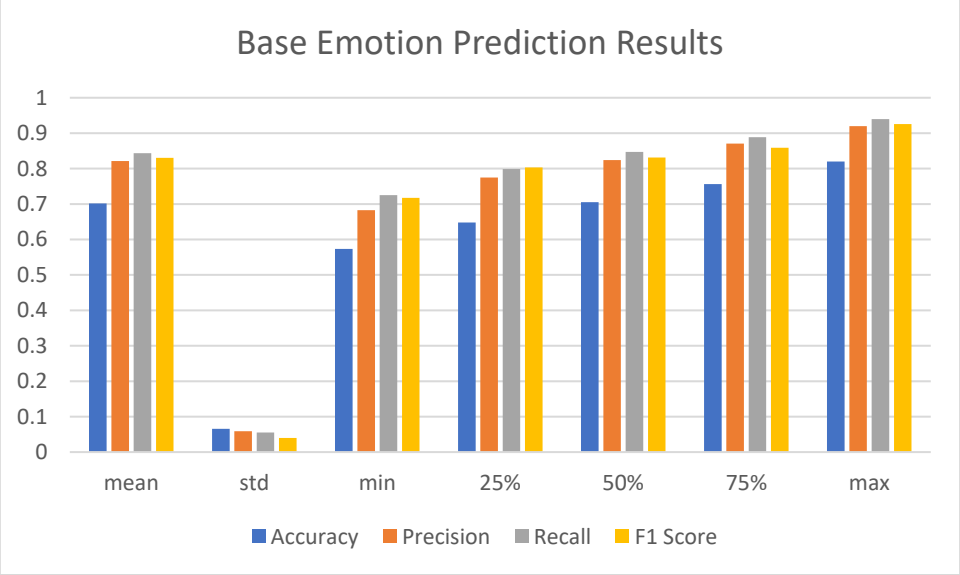


Figure 3 Base Emotion Prediction Results

3.4. Discussion of Results

Our sentiment analysis algorithm, incorporating topic and emotion annotation with a pre-trained BERT model [6], demonstrated a improved sentiment classification. The results showcase its effectiveness, particularly in capturing nuanced emotional

DOI: 10.2478/ijasitels-2023-0007

subtleties within the text. Areas for improvement were identified, such as refining preprocessing techniques and exploring advanced feature extraction methods.

3.5. Future Research Directions

Future research could prioritize refining preprocessing techniques, exploring sophisticated feature extraction methods, and considering ensemble learning approaches to further enhance performance. Addressing challenges specific to certain domains or languages will be crucial for the algorithm's continued adaptability and efficacy.

3.6 Conclusion

In conclusion, the results and discussion chapter provide a concise yet comprehensive analysis of our sentiment analysis algorithm's outcomes. The findings contribute to the broader understanding of sentiment analysis capabilities and guide future research in this dynamic field.

4. References

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding*. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (pp. 4171-4186).

[2] LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning. Nature*, 521(7553), 436-444.

[3] Pang, B., & Lee, L. (2008). *Opinion Mining and Sentiment Analysis*. Foundations and Trends in Information Retrieval, 2(1-2), 1-135.

[4] Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). *Recursive deep models for semantic compositionality over a sentiment treebank*. In Proceedings of the conference on empirical methods in natural language processing (EMNLP) (Vol. 1631, p. 1642).

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017). *Attention is all you need*. Advances in neural information processing systems, 30.

[6] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270 (last visit december 2023)

[7] https://www.kaggle.com/datasets/rajugc/kaggle-dataset (last visit december 2023)

[8] Kaufmann, M., Principles of Data Integration, ISBN 9780124160446, 2012

DOI: 10.2478/ijasitels-2023-0008

Methods for data validation using QR codes

Oleksik Vlad Andrei[1], Pitic Elena Alina[2], Crețulescu Radu George[1]
[1]Computer Science and Electrical and Electronics Engineering Department, Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania
[2]Department of Mathematics and Informatics, Faculty of Sciences, "Lucian Blaga" University of Sibiu, Romania
{vladandrei.oleksik, alina.pitic, radu.cretulescu} @ulbsibiu.ro

Abstract

This article aims to elaborate on a few strategies designed for validating data encoded as a QR code. As the use-cases for such validation schemes vary widely as the demand for such practices is continuously increasing, the present article will represent a comparative study of checksum, hashing, as well as asymmetric encryption algorithms for the offline validation of claims transmitted through the payload of the exchanged data. Results are evaluated based on different criteria impacting the reliability, such as the size of the QR code and the resulting security.

Keywords: Data validation, Data encoding, Checksum algorithms, Hashing Algorithms, Signature algorithms

1. Introduction

In today's context of constant technological evolution and free access to digital communication tools, data is one of the main resources for research, development, innovation, and economic growth [8]. Open access to technological solutions and communication networks contributes to the increasing amounts of data generated and transmitted daily.

Each owner of an internet-enabled device can access dozens of information resources, which in turn collect and transmit data that we consciously and unconsciously expose, such as location including place of work and/or residence, race, gender, age, contact details, personal interests, marital status, or religious beliefs [5].

This data is collected and stored by organisations for the purpose of further use to identify opportunities or threats, thus preparing their decision-making strategy and adapting to current or future trends.

As mentioned, we all expose huge amounts of data, and organisations don't need all the data we expose, whether it is real or not. What data collectors need to ensure is that this information meets their collection requirements. This is where ***data validation*** comes in.

DOI: 10.2478/ijasitels-2023-0008

Data validation is a crucial practice in ensuring the integrity, accuracy, and structural soundness of data before it is used in various business operations [6]. It encompasses a range of validation types, including data type, constraint, structure, consistency, and code validation, each of which is tailored to check that the data meets the necessary criteria for further processing [10].

At the same time, data validation can be defined as a form of data cleansing, which involves examining source data for accuracy and quality before any processing, import or use takes place [24]. This definition aligns with the perspective presented above, where the importance of ensuring that data meets specific standards and objectives within the constraints of the destination was also emphasized.

Given the enormous amounts of data that are transported daily, securing this data is one of the priorities of organisations that collect this data. This is an obligation for the organisations concerned. For example, under GDPR rules, companies and organisations that collect, store and process personal data are obliged to provide safeguards for the protection of that data [20]. Thus, the organisations concerned must consider securing data in the 2 known forms: *data in motion* and *stored data*. To this end, organisations usually use data ***checksum algorithms***, ***hash algorithms*** and ***encryption algorithms*** [15] [3]. Each of these categories of algorithms plays a significant role in ensuring data integrity and confidentiality.

Checksum algorithms are used for ensuring data integrity. These algorithms generate fixed-size values, known as checksums, which uniquely represent the contents of a given data set. These algorithms aim to identify errors or changes in the data during transmission or storage. A checksum is calculated at the source and sent or saved with the data when it is transferred or stored. When receiving or retrieving data, the checksum is recalculated and if it differs from the original transmitted or stored checksum, this indicates a possible error or corruption of the data [15] [4]. In our research, we used three types of checksum algorithms**.** The first algorithm was ***CRC32*** (*Cyclic Redundancy Check 32*) is a checksum algorithm based on polynomial division, that hashes byte sequences to 32-bit values [2].

Another algorithm used is the Fletcher algorithm, which was devised by John G. Fletcher. There are three popular Fletcher checksum algorithms, of which I used Flecher32 which divides the data word into 16-bit blocks, computes two 16-bit checksums, and adds them together to form a 32-bit Fletcher checksum [13].

The last checksum algorithm used is Adler-32, which is an algorithm written by Mark Adler by modifying Fletcher's checksum. Compared to Flecher's algorithm, Adler32 is mainly known for its simplicity and speed, while Fletcher32 is chosen for applications where efficiency and simplicity are prioritized over cryptographic strength [9].

Hashing algorithms are one-way or irreversible, so the text cannot be deciphered and decoded by anyone else. Thus, hashing protects data at rest, so that even if someone gains access to the server where it is stored, the items remain unreadable [11][26]. In this study, we used SHA-1-HMAC, a keyed hash algorithm based on the SHA1 hash function and used as a hash-based message authentication code (HMAC). The HMAC process mixes a secret key with the message data, hashes the result using the hash function, mixes the hash value with the secret key again, and then applies the hash function a second time [25].

DOI: 10.2478/ijasitels-2023-0008

Data encryption is a security measure that involves the transformation of readable information, called plaintext, into an unreadable format, called ciphertext, using algorithms and cryptographic keys [23] [3]. The process serves to secure sensitive data from unauthorised access while maintaining confidentiality and privacy. In encryption, a mathematical technique uses a cryptographic key to alter the original data, rendering it unreadable without the appropriate decryption key. This cryptographic approach is critical for protecting information during transmission or storage, preventing cybercriminals, unauthorised users, and other harmful entities from posing a threat.

Encryption plays an essential role in securing various digital communications and transactions, such as online banking, e-commerce transactions and the transfer of sensitive information [21][7]. It serves as a fundamental component of cybersecurity strategies, preventing unauthorised parties from interpreting and exploiting confidential data [22].

There are two types of encryption algorithms, namely symmetric encryption algorithms, in which a single key is used for both encryption and decryption of data, and asymmetric encryption algorithms, which involve the use of two keys: the public key, used for encrypting data, and the private key, used for decrypting data [21] [7].

As encryption algorithms we used two asymmetric algorithms: *RSA* and *ECDSA*.

RSA, or ***Rivest-Shamir-Adleman***, is an encryption algorithm that provides a versatile approach for data securing through asymmetric cryptography, offering two distinct methods [14] [28]. One of the methods involves encrypting sensitive information with the recipient's public key, ensuring only the owner of the corresponding private key can decrypt the data. This method is ideal for secure transmission of data across networks, as the sender uses the recipient's public key for encryption. Other method involves encrypting a message with the sender's private key and sending both the encrypted data and the sender's public key to the recipient. Here, the recipient can decrypt the data using the sender's public key, verifying the sender's identity [18].

ECDSA, or ***Elliptic Curve Digital Signature Algorithm***, is an asymmetric cryptographic primitive which stands out for its digital signature efficiency compared to other algorithms, achieved by using smaller keys while maintaining a high level of security. *ECDSA* generates certificates, electronic documents for authenticating the certificate owner, containing information about the key, owner details, and the issuer's signature. The elliptic curve analysis forms the basis of *ECDSA*'s security, making it resistant to current encryption cracking methods. Despite being standardized in 2005, newer protocols favour *ECDSA* due to its relative novelty, reducing the time hackers have had to crack it (What is ECDSA Encryption? 2020). However, *RSA* remains widely used, primarily due to its longer presence since standardization in 1995, even though attackers have had more time to attempt to break it. While *ECDSA* offers enhanced security and is preferred in certain contexts, its drawback lies in complexity during implementation compared to the more straightforward setup of *RSA*, making proper implementation crucial for avoiding vulnerabilities [12].

DOI: 10.2478/ijasitels-2023-0008

2. Methodology

In this section, various QR code-based data validation schemes, each based on a subset of the algorithms described in section 1, will be presented in detail. Along with the specific data encoding scheme, for each case, an emphasis will be placed on certain theoretic remarks regarding the time complexity of the algorithms involved in the process, the physical space occupied by the code used for the data transfer, or the way each of these schemes would integrate into the modern data handling paradigm.

To analyse the behaviour of different data validation strategies, first and foremost, it is required to establish some sample data on which to apply each encoding. As such, a set of sample data (claims) representing a student's current enrolment status, based on the widely used JSON format [17] in data serialization, is presented below:

```
{
 "data": {
  "no": 309,
  "scope": "Internal",
  "universityYear": "2023-2024",
  "date": "12/12/2023",
  "holder": {
   "givenName": "Vlad-Andrei",
   "familyName": "Oleksik",
   "CNP": "1230405065000",
   "studyYear": "I",
   "programme": "Calculatoare"
  },
  "issuer": {
   "university": "ULBS",
   "faculty": "FING"
  }
 }
}
```

In this example, the JSON data contains information about the personal details of a student, as well as a context describing the issuer and intended receiver of the claims. This structure is in accordance with the most used data or authentication flows in information systems [27].

However, this format would apply for serializing any "object" modelled under the OOP paradigm. As it can be noticed, the format does not inherently optimize the space occupied by all the data, nor does the format describe each field efficiently. Thus, the total space required to handle the entire sample would amount to approximately 260 B.

For the data validation, several uses of QR codes will be detailed below. Depending on how one intends to use the QR-encoded data in the validation process, there are two main categories of validation schemes. Thus, a first concept would be the encoding of the data in its entirety, for it to then be validated and processed as-is, without the need for further communication to obtain relevant data. Another approach is only encoding an identifier in the data to be validated, while invariably keeping a copy of the complete claim's server-side. Thus, trusting a remote connection would still be a prerequisite for using data validated using QR codes. This latter approach will be first presented.

sciendo
DOI: 10.2478/ijasitels-2023-0008

2.1. Basic data validation

This strategy relies on the use of an identifier to locate the resources that need to be validated. The integrity of the identifier itself is ensured by using either a checksum or a hash-based message authentication code.

2.1.1. Checksum-enabled basic validation

This first approach is centred on the use of the basic data required for identification, along with a simple checksum based on the Cyclic Redundancy Check algorithm, as described above. Thus, the payload will have the structure presented below:

1230405065000-309-12/12/2023-46B6D55B

The size of the checksum used in this case is 4 bytes. Since both the identification data and the hexadecimal encoded checksum only contain alphanumeric data, the size of the QR code can be easily reduced to a small size as compared to a QR code encoding the same data with the binary encoding.

As mentioned in the previous chapter, were the checksum computation speed of particular concern, the Fletcher32 or Adler32 algorithms could be used instead. However, as this is typically not the case, only the CRC32 will be considered for the purpose of this paper.

It is worth mentioning that, as all the algorithms involved are publicly known and reversible, this method is generally not recommended when the code is used for sensitive identification purposes. Thus, this encoding scheme only ensures resistance against very basic tampering, while being very effective space-wise.

2.1.2. HMAC-enabled basic validation

If a higher security level is expected for validating some basic identification information, while the encoded size represents a concern, then the use of a Hash-based Message Authentication Code primitive instead of the checksum is desirable.

Briefly, this algorithm uses a one-way function which takes in both the payload and a secret key, outputting a value unique (considering the collision probability to be negligible) to each payload and secret key. While the SHA-1 algorithm is not recommended anymore for the purpose of concealing secrets, it is still considered safe for the purpose of HMAC. This HMAC algorithm provides an output of size 20 bytes. Thus, the encoded data will resemble the following format:

1230405065000-309-12/12/2023-C62600020DC4A288A4A1EC411C494AA5B275C1F9

When validating the data, a validator knowing the secret key can compute the same HMAC value for the payload, the data being considered valid if it matches the one provided within the QR code.

The use of this algorithm enables for reasonable security, while maintaining a relatively small size of the encoding data.

DOI: 10.2478/ijasitels-2023-0008

The two validation schemes described above cover the case of validating basic identification data, referencing claims stored server-side. As such, the verification process typically continues online, certifying the accuracy of the data. The next subchapter will present schemes that could be used for validating the entire data based solely on the QR-encoded data.

2.2. Complete data validation

This strategy is focused on encoding the entire data in the QR code. As this scheme would not necessarily require an online validation unless claims dynamically change and the space used will inevitably increase, the use of stronger algorithms for ensuring the integrity of the data is recommended.

Another area of concern when including the whole sample data in the QR code is the format to be used for handling it. Since the data to be ultimately encoded is structured as a payload and a code or signature used for validation, the JSON Web Token standard is a desirable choice. It is widely used in sensitive authentication data handling [27] and encompasses a header describing the validation algorithm to be used, as well as payload and validation fields.

2.2.1. Complete symmetric (HMAC-based) data validation

This first approach keeps relying on the HMAC algorithm, although, for this case, where the offline validation should suffice, this article will treat the HS256 (SHA-256-HMAC) algorithm. As opposed to the SHA-1-HMAC algorithm, the size of the output is 64 bytes. The final JWT is as presented below:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRhIjp7Im5vIjozMDksInNjb3BlIjoiSW50ZXJuYWwiLCJ1bml2ZXJzaXR5WWVhciI6IjIwMjMtMjAyNCIsImRhdGUiOiIxMi8xMy8yMDIzIiwiaG9sZGVyIjp7ImdpdmVuTmFtZSI6IlZsYWQtQW5kcmVpIiwiZmFtaWx5TmFtZSI6Ik9sZWtzaWsiLCJDTlAiOiIxMjMwNDA1MDY1MDAwIiwic3R1ZHlZZWFyIjoiSSIsInByb2dyYW1tZSI6IkNhbGN1bGF0b2FyZSJ9LCJpc3N1ZXIiOnsidW5pdmVyc2l0eSI6IlVMQlMiLCJmYWN1bHR5IjoiRklORyJ9fX0.qWeK7SYOZidksAt8js0UAQyVRYw_1lzoHPd5WjZNYHw

Since the payload, like the validation data, is of considerably larger size, the JSON Web Token will have them encoded in base 64. As such, the alphanumeric encoding mode of the QR code can no longer be used and the binary encoding will be used instead. As only a subset of the characters in the binary character space are used, the efficiency of the space usage in the QR code will decrease unless the resulting token is compressed and converted to another base.

As for the scheme presented in subsection 2.1.2., a validator knowing the secret key convened upon would have to compute the HMAC output for the header-payload combination and the secret, an output corresponding to the last field of the JWT certifying the integrity of the data.

DOI: 10.2478/ijasitels-2023-0008

2.2.2. Complete asymmetric (RSA-based) data validation

This approach switches to the asymmetric RSA algorithm as described in the first chapter. The issuer passes the protected data in the JWT through the private function of the RS256 primitive, the output, of size 256 B, being appended to the encoded data, which would resemble the following structure:

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJkYXRhIjp7Im5vIjozMDksInNjb3BlIjoiSW50ZXJuYWwiLCJ1bml2ZXJzaXR5WWVhciI6IjIwMjMtMjAyNCIsImRhdGUiOiIxMi8xMy8yMDIzIiwiaG9sZGVyIjp7ImdpdmVuTmFtZSI6IlZsYWQtQW5kcmVpIiwiZmFtaWx5TmFtZSI6Ik9sZWtzaWsiLCJDTlAiOiIxMjMwNDA1MDY1MDAwIiwic3R1ZHlZZWFyIjoiSSIsInByb2dyYW1tZSI6IkNhbGN1bGF0b2FyZSJ9LCJpc3N1ZXIiOnsidW5pdmVyc2l0eSI6IlVMQlMiLCJmYWN1bHR5IjoiRklORyJ9fX0.auauRueb4p2kbVxLcEyCldoGQD6AIU9YbL-RpDqEk60ffcqYGO919rjphbeh0ZJybovKVRTpWL8wKXFNFuWFfZEyzR3X5d6_6y9fxZoBIC46dxikuHkTCXauY0XE1Ytnh5qxkQdQSA5w_A-xD-hlzlR-Ah4U5qc4Hm8A6uTG1RAvlb3-R1ckkv4USl7kz2WC1MCQzP-bySsw6JaXNPI0mBGkbpMnt9ms2V4uBAi8CMPzTUgbUHYpcRvvcTYOIiIchTxmns-hRPbKCjA6N9pnkGT8RogSyjzxyOL1WL8Ea2R_KxXf-ULYhxgVnNG6cUj0JByFq3AhUJ5Vdxx8N43D7w

For processing the data, the validator must use the public key of the issuer to reverse the signature process back to the data to be protected. If it matches the claims provided in the QR code, they are assumed to be valid, with a very high degree of certainty.

The security is greatly enhanced for this scheme in that the validator is no longer required to know the private key used to generate other valid tokens, but only its public component that can be used exclusively for checking the integrity of the data. Thus, security threats of the system are reduced.

2.2.3. Complete asymmetric (ECDSA-based) data validation

This approach uses the asymmetric ECDSA algorithm as described in the first chapter instead of RSA. There are few high-level functional differences between the two algorithms, but ECDSA is generally preferred in that the size of the signature is lower, for a security equivalent to that provided by RSA.

The issuer passes the protected data in the JWT through the private function of the ES256 primitive, the output, of size just 128 B, being appended to the data to be included in the QR code:

eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiJ9.eyJkYXRhIjp7Im5vIjozMDksInNjb3BlIjoiSW50ZXJuYWwiLCJ1bml2ZXJzaXR5WWVhciI6IjIwMjMtMjAyNCIsImRhdGUiOiIxMi8xMi8yMDIzIiwiaG9sZGVyIjp7ImdpdmVuTmFtZSI6IlZsYWQtQW5kcmVpIiwiZmFtaWx5TmFtZSI6Ik9sZWtzaWsiLCJDTlAiOiIxMjMwNDA1MDY1MDAwIiwic3R1ZHlZZWFyIjoiSSIsInByb2dyYW1tZSI6IkNhbGN1bGF0b2FyZSJ9LCJpc3N1ZXIiOnsidW5pdmVyc2l0eSI6IlVMQlMiLCJmYWN1bHR5IjoiRklORyJ9fX0.AICvWh5P1bDrOXtdyY2FcicjoR5CVHN5xtHM62-9dStgAy1RO1mnEw9ODFKdjG6QlKMbS4zxSfzbGG9LJh9Iel04AFDhen6uLZLvQFQ79iSev3EE7HFYpcRP1DxYLNf1YpaxdQiA1QRZtsEBeCn0Ha2H6rgd8L_bUWY1lUnQuOo3hcBw

The other considerations stay the same as for RSA.

DOI: 10.2478/ijasitels-2023-0008

3. Results

To assess the behaviour and performance of each of the approaches described above, sample codes have been generated using the sample data and the schemes presented in section 2. For all cases, the medium level of error correction was used.

For each code, several parameters have been recorded, including the physical space occupied by each code (determined by the QR version needed to accommodate for each of the strings to be encoded) and the bits of security provided by each of the validation algorithms and other specifications.

Using an implementation of the checksum algorithm and the cryptographic primitives previously described, codes were generated in accordance with each of the validation schemes presented above. The results can be found in Figures 1-5.

Figure 1 Minimal data encoded for checksum validation

Figure 2 Minimal data encoded for HMAC validation

DOI: 10.2478/ijasitels-2023-0008

Figure 3 Complete data encoded for HMAC validation

Figure 4 Complete data encoded for RSA validation

Figure 5 Complete data encoded for ECDSA validation

DOI: 10.2478/ijasitels-2023-0008

As it can be seen, the size of the QR codes varies widely depending on the validation scheme to be used. In accordance with this and given the increasing trend towards using QR codes on printable documents, the approach to be preferably used is largely determined by the physical space available. As such, a comparison of the space required by each approach for the data to be encoded is presented in Table 1. For the minimum printed size, the standard ratio of 12 modules/cm [16]

Table 1 Size characteristics of the resulting codes

Approach	Data size (B)	Signature relative size (%)	QR code version	Module number	Minimum printed size (cm^2)
1.	25	21.6	2	25 x 25	2.08 x 2.08
2.	48	60.0	4	33 x 33	2.75 x 2.75
3.	428	10.0	16	81 x 81	6.75 x 6.75
4.	727	47.2	22	105 x 105	8.75 x 8.75
5.	561	31.4	19	93 x 93	7.75 x 7.75

As it can be noticed, for encoding the entire data to be transmitted, the size required increases boldly, a compromise thus existing between the reliability and the physical space occupied to ensure readability of data.

The other relevant factor for such a data validation scheme is the security that it provides. The most widespread way to quantify this security would be represented by the number of bits of entropy in the key that would be required to obtain the correct key. The security inherent to each approach, along with the other defining factors of each of them, are presented in Table 2. For the checksum algorithm, since it is not designed for use with a key, a value of 0 bits of security was considered.

Table 2 Security of the resulting codes

Approach	Data included	Data size (B)	Security (b)	Algorithm type
1.	minimal	25	0	-
2.	minimal	48	~160*	symmetric
3.	complete	428	~256*	symmetric
4.	complete	727	128	asymmetric
5.	complete	561	128	asymmetric

**The level of security of HMAC is influenced by an implementation detail, making it difficult to estimate the exact information needed to break it [1] The level of security for SHA-1-HMAC can often be as low as 160 bits.*

As it can be seen in the table above, apart from the unsecure first approach, which is based on checksums, the levels of security are sufficient for all the other validation schemes, above the current, agreed upon, security threshold of 100 bits of security.

While the theoretical entropy of the keys used in symmetric algorithms is higher, their security is otherwise hindered by the need for disclosing the key used for issuing new data to be validated.

DOI: 10.2478/ijasitels-2023-0008

An additional factor to consider when evaluating these approaches is the time elapsed for each process (generation and validation, respectively). As such, the generation and the validation rounds were each timed with a microsecond-precision timer, the average results for each algorithm being presented in Table 3.

Table 3 Time elapsed for processing the resulting codes

Approach	Algorithm	Generation time (µs)	Verification time (µs)
1.	CRC32	<1	<1
2.	SHA-1-HMAC	<1	<1
3.	SHA-256-HMAC	1	1
4.	RSA	42823	644
5.	ECDSA	3098	2184

These results are represented in the logarithmic-linear chart in Figure 6. As it can be seen from the table above and the chart, the only significant delays are produced by the asymmetric signature algorithms.

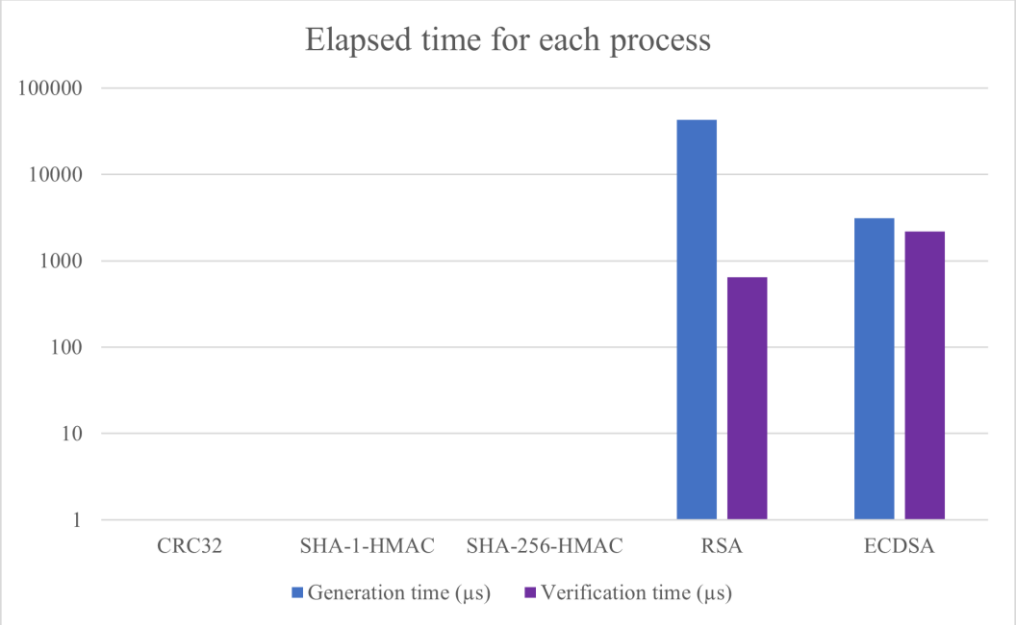


Figure 5 Complete data encoded for ECDSA validation

Notably, while the generation of the signature takes a very long time for RSA, ECDSA has a much higher signature speed, although the verification process is slightly slower than for RSA. However, as these two algorithms considerably enhance security, the time cost using asymmetric algorithms is justified.

4. Conclusions

The present paper has presented multiple alternatives for streamlined data handling using QR code-based data validation.

Although there are many factors whose influence on the optimum approach highly depends on the circumstances that describe the intended use of the validation flow, these approaches generally cover most practical use-cases.

DOI: 10.2478/ijasitels-2023-0008

For encoding only minimal data, the HMAC-based approach should not be used as the last verification layer, it is very effective in ensuring the integrity of data even against relatively qualified, but not large-scale tampering or attacks.

On the other hand, where the physical space available for the QR code is not a very important constraint, the ECDSA-signed JSON Web Token represents an asset in that it allows for both privacy (the data is stored with the holder) and transparency (anyone can verify the authenticity of the data). Thus, this scheme is currently being introduced in an increasing number of official documents in the worldwide process of digitalisation [20]

5. References

[1] Bynens, M, PBKDF2+HMAC hash collisions explained. https://mathiasbynens.be/notes/pbkdf2-hmac (March 25, 2014).

[2] Davies, J. Understanding CRC32. https://commandlinefanatic.com/cgi-bin/showarticle.cgi?article=art008 (December 10, 2023).

[3] De Groot, J., What Is Data Encryption? (Definition, Best Practices & More). https://www.digitalguardian.com/blog/what-data-encryption (December 10, 2023).

[4] Fisher T., What Is a Checksum? See a Definition, Examples, and More. Lifewire. https://www.lifewire.com/what-does-checksum-mean-2625825 (December 10, 2023).

[5] Hetler, A, Common Social Media Privacy Issues. WhatIs.com. https://www.techtarget.com/whatis/feature/6-common-social-media-privacy-issues (October 14, 2023).

[6] Kerner, S., M., What Is Data Validation? Data Management. https://www.techtarget.com/searchdatamanagement/definition/data-validation (December 9, 2023).

[7] Loshin, P., What Is Encryption and How Does It Work? Security. https://www.techtarget.com/searchsecurity/definition/encryption (December 10, 2023).

[8] Neri, A , 'We Should Treat Data as a Natural Resource. Here's Why'. World Economic Forum. https://www.weforum.org/agenda/2020/03/we-should-treat-data-as-a-natural-resource-heres-why/ (December 10, 2023).

[9] 'https://en.wikipedia.org/w/index.php?title=Adler-32&oldid=1146207638 (December 10, 2023).

[10] What Is Data Validation?' Astera. https://www.astera.com/knowledge-center/what-is-data-validation/ (December 9, 2023).

[11] Fundamental Difference Between Hashing and Encryption Algorithms | Baeldung on Computer Science. https://www.baeldung.com/cs/hashing-vs-encryption (December 10, 2023).

[12] Encryption: ECDSA vs. RSA Keys | Baeldung on Computer Science. https://www.baeldung.com/cs/encryption-asymmetric-algorithms (December 10, 2023).

[13] Fletcher's Checksum. https://www.tutorialspoint.com/fletcher-s-checksum (December 10, 2023).

DOI: 10.2478/ijasitels-2023-0008

[14] https://clarvision.ro/referinte/arsat/ (December 27, 2022).
[15] What Is a Checksum? An Easy-to-Understand Checksum Definition. Code Signing Store. https://codesigningstore.com/what-is-checksum-how-it-works (December 10, 2023).
[16] QR Code Model 1 and Model 2: Point for setting the module size. https://www.qrcode.com/en/howto/cell.html (May 4, 2013).
[17] The JSON Data Interchange Syntax. https://www.ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf (December, 2017)
[18] What Is RSA? How Does an RSA Work? | Encryption Consulting. https://www.encryptionconsulting.com/education-center/what-is-rsa/ (December 10, 2023).
[19] Data Protection under GDPR. https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_en.htm (December 10, 2023).
[20] eHealth Network Guidelines on Technical Specifications for EU Digital COVID Certificates | Volume 2'. https://health.ec.europa.eu/system/files/2022-07/digital-covid-certificates_v2_en.pdf (June 15, 2022).
[21] What Is Data Encryption? https://www.forcepoint.com/cyber-edu/data-encryption (December 10, 2023).
[22] What Is Encryption and How Does It Work?' Google Cloud. https://cloud.google.com/learn/what-is-encryption (December 10, 2023).
[23] What Is Encryption? Data Encryption Defined. https://www.ibm.com/topics/encryption (December 10, 2023).
[24] What Is Data Validation: Definition. https://www.informatica.com/services-and-training/glossary-of-terms/data-validation-definition.html (December 9, 2023).
[25] 'HMACSHA1 Class (System.Security.Cryptography)'. https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.hmacsha1?view=net-8.0 (December 10, 2023).
[26] Hashing Algorithm Overview: Types, Methodologies & Usage. https://www.okta.com/identity-101/hashing-algorithms/ (December 10, 2023).
[27] OpenID Connect Basic Client Implementer's Guide 1.0 - draft 46'. https://openid.net/specs/openid-connect-basic-1_0.html (October 28, 2023).
[28] veritas.com. 'What Is RSA Encryption?' https://www.veritas.com/information-center/rsa-encryption (December 10, 2023).

International Journal of Advanced Statistics and IT&C
for
Economics and Life Sciences

VOLUME 13 Number 1 CONTENTS 2023

ISSN 2067 – 354X
eISSN 2067 - 354X