


Universitatea Politehnica Bucureşti
Facultatea de Automatică si Calculatoare
Departamentul de Automatică şi Ingineria Sistemelor

# LUCRARE DE LICENŢĂ

# Clasificare defecte într-o rețea de apă de mari dimensiuni

Absolvent
Cazan Cristian-Claudiu

Coordonator
Conf. dr. ing. Florin Stoican

București, 2018

# Cuprins

# Listă de figuri

# Listă de tabele

# Listă de algoritmi

# 1. Introducere

## 1.1. Motivația alegerii temei

Transportul și distribuția apei reprezintă una dintre cele mai vechi preocupări inginerești de proporții, existând de mai mult de 4000 de ani. Civilizația minoică, localizată în insula Creta, este considerată a fi prima care a construit apeducte - structuri pentru transportul apei de la sursă către orașe - în 2500 î.Hr.

Deși majoritatea popoarelor din antichitate care s-au ocupat cu construcția apeductelor întrebuințau aceste sisteme pentru irigația pământului - ocupațiile de bază de atunci fiind în strânsă legătură cu agricultura - romanii au văzut în sistemele de provizionare a apei și un potențial imens în dezvoltarea civilizației, astfel ei sunt ei care aduc cele mai mari contribuții inginerești, apeductele construite de aceștia impresionând și astăzi prin grandoarea și iscusința cu care au fost construite.

Inovațiile în acest domeniu au suferit salturi bruște și puternice în momentul descoperirii unei noi relații matematice care transformă un parametru despre care se puteau face doar niște estimări grosiere într-o mărime bine definită și bine controlată. Istoric vorbind incipitul dezvoltării științei hidraulice s-a bazat pe relația descoperită de Arhimede din Siracuza în sec III î.Hr. $F = V_{obiect} * \rho_{lichid} * g$. O altă contribuție care are o deosebită importanță în domeniul tehnologiei de distribuiție a apei și nu numai o reprezintă tubul lui Pitot folosit la măsurarea vitezei fluidului, inventat de Henri Pitot în sec XVII. Din punct de vedere constructiv tubul are o formă de **L**, scufundarea acestuia într-un fluid (apă sau gaz) va determina creșterea nivelului și a presiunii până la o anumită limită ;[1], ecuația care guvernează depedența nivel - viteză este:

$$u = \sqrt{\frac{2(p_t - p_s)}{\rho}} \tag{1.1}$$

unde:

- u reprezintă viteza fluidului
- $p_t$ reprezintă presiunea de stagnare
- $p_s$ reprezintă presiunea statică
- $\rho$ reprezintă densitatea fluidului

Mergând mai departe, alte contribuții importante apar din partea marilor matematicieni precum Daniel Bernoulli și Leonhard Euler, care au mărit spectrul mecanicii lui

Newton și Leibniz spre aria hidraulicii și a termodinamicii. Fluidele considerate sunt incompresibile și au densitatea constantă în timp și uniform distribuită în spațiu. Bernoulli afirmă despre lichidele incompresibile că o creștere în viteză a lichidului este însoțită de o scădere a energiei potențiale a lichidului (i.e. presiune):

$$\frac{v^2}{2} + gz + \frac{p}{\rho} = c \tag{1.2}$$

unde:

- v reprezintă viteza fluidului
- g reprezintă accelerația la care e supus fluidul
- z reprezintă elevația ștragulației conductei
- p reprezintă presiunea într-un anumit punct
- $\rho$ reprezintă densitatea fluidului

În contextul în care se dorește analiza unui caz real este important ca toate diferențele între cazurile ideale și cazurile reale trebuie puse în evidență în mod matematic, astfel se particularizează ecuația generală Navier-Stokes pentru cazuri în care se cunosc anumiți parametrii ai sistemului de analizat. Spre exemplu ecuația Poisuille care modelează începutul fluxului de apă într-o conductă este ;[2]:

$$\frac{\partial u}{\partial t} = \frac{G}{\rho} + \nu \left( \frac{\partial^2 u}{\partial t^2} + \frac{1}{r} \frac{\partial u}{\partial r} \right) \tag{1.3}$$

unde:

- $u$ reprezintă viteza lichidului prin conductă
- $t$ reprezintă timpul
- $G$ reprezintă diferența de presiune
- $\rho$ reprezintă densitatea lichidului
- $\nu$ reprezintă vâscozitatea cinematică
- $r$ reprezintă poziția

Se poate observa că pe măsură ce modelul matematic se apropie de realitate, complexitatea acestuia crește și pentru fiecare situație specială - spre exemplu analiza presiunii la introducerea apei într-o conductă vs. analiza presiunii când conducta este încărcată cu apă - are nevoie de o ecuație specială sau de o particularizare a ecuației Navier-Stokes, pentru care încă nu se conoaște dacă există soluții pentru cazul cu 3 dimensiuni și dacă soluțiile acestea sunt netede.

Ținând cont de importanța apei în desfășurarea activităților cotidiene atât pentru oameni cât și pentru actorii importanți ai industriei, este o condiție sine-qua-non ca un

oraș să aibă un sistem performant și rezistent la defecte pentru distribuția apei. În contextul actual al dezvoltării tehnologiei este natural să folosim tehnici moderne de monitorizare a diferiților parametrii din cadrul unei rețele pentru a putea face o analiză riguroasă și eficientă cu referire nu numai la mentenanță ci și la consumul global și local în ideea îmbunătățirii și reducerii pierderilor.

## 1.2. Expunerea problemei

În această lucrare se va aborda problematica identificării prezenței unui defect - *Fault detection* și izolarea defectului *Fault isolation* într-o regiune a rețelei.

O rețea de apă poate fi privită ca un graf neorientat $G = (V, E)$ unde $V$ este mulțimea nodurilor rețelei - acestea reprezentând o abstractizare asupra componentelor precum:

- rezervoare
- tancuri de apă
- puncte de distribuție

$E$ este mulțimea muchiilor reprezentând de fapt țevile care fac legătura între noduri.

Mergând mai departe cu abstractizarea se pot considera rețele de apă active și rețele de apă pasive. Diferența între cele două făcându-se în baza pompelor de apă amplasate în zonele unde presiunea sau elevația vin în detrimentul distribuției apei.

Rețelele de apă care vor fi tratate în această lucrare fac parte din categoria pasivă, astfel putem diviza mulțimea nodurilor $V$ în $V^t$ și în $V^j$ reprezentând mulțimea nodurilor de tip tanc și mulțimea nodurilor joncțiune, cu proprietatea că $V = V^t \cup V^j$. Tancurile și rezervoarele dintr-o rețea de apă au proprietatea că nivelul de apă din acestea se va menține la un nivel oarecum staționar, astfel simulările din capitolele viitoare se vor axa pe nodurile simple de tip joncțiune, deci mulțimea de interes în acest caz va fi $V^j$ pentru care cunoaștem cardinalul.

Caracteristicile care se pot recolta dintr-o rețea de apă pot varia în funcție de elementul inspectat și de senzorii dispuși în rețea, astfel pentru fiecare nod $n_i \in V^j$ putem defini la fiecare moment de timp

- presiunea $p_i(t)$ - măsurată în metri coloană de apă $mH2O$, mărime influențată puternic de presiunea interioară a nodului și de eventualele perturbații exterioare i.e. scurgeri de apă prin țevi
- 'cererea' $d_i(t)$ - măsurată $L/s$, mărime ce caracterizează profilul de utilizare al utilizatorilor de-a lungul unei zile i.e. debitul de apă care ajunge la consumatori. Acest debit poate varia de-a lungul zilei, putem distinge de exemplu intervale de timp în care cererea este foarte mică și rețeaua intră în regim staționar

De asemenea pentru fiecare conductă a rețelei $e_{ij} \in E$ putem măsura viteza lichidului $v_{ij}(t)$.

Pentru a putea rezolva problema de *Fault Detection and Isolation* este importantă găsirea unei modalități eficente de selecție și prelucrare a datelor de la rețea.Mai mult, punând în lumină aspectul ingineresc al problemei, trebuie găsită o submulțime $V_{opt} \subset V^j$ ai cărei elemente pot aduce informații necesare și suficiente pentru a detecta un defect într-o acoperire destul de mare a rețelei.

## 1.3. Exemplul de lucru

În următoarele capitole și în implementarea lucrării consider rețeaua din Hanoi iar pentru simularea scenariilor propuse voi folosi biblioteca și suita de funcții **EPANET** - Environmental Protection Agency NETwork

Rețeaua Hanoi constă într-o mulțime de noduri de tip joncțiune $V^j$ cu $|V^j| = 31$ și mulțimea $V^t$ cu $|V^t| = 1$, ilustrată în figura de mai jos:

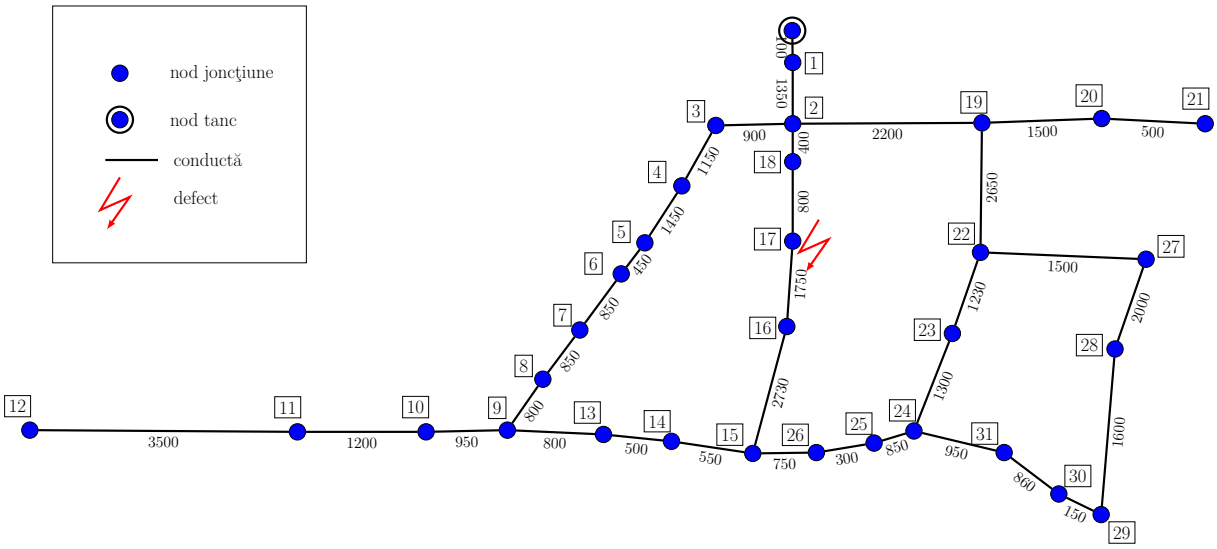



Figura 1.1.: Graful rețelei de apă din Hanoi [3]

După cum se poate observa în figura de mai sus au fost reprezentate două tipuri de node pasive, anume tancurile și joncțiunile. În cazul apariției unui defect în rețea, este important de luat în considerație modalitatea în care acesta va influența dinamica rețelei, spre exemplu este de la sine înțeles că dacă se cosideră un defect în nodul cu indicele 17 - i.e. în acest nod au apărut anumite scurgeri care afectează fluxul de apă către consumatori - nodurile în care se va observa o modificare puternică a caracteristicilor (presiune și debit) vor face parte din mulțimea nodurilor adiacente rețelei $S = V_{16}, V_{18}$, deși pare o concluzie naturală, o modelare matematică riguroasă din care să se tragă această nu este o problemă foarte ușor de rezolvat, anumiți parametrii fiind extrem de greu de estimat chiar și în cazul în care se consideră un regim staționar.

# 2. Simulări și software folosit

## 2.1. Dificultatea simulării unei rețele de apă

Găsirea unui set de ecuații al cărei soluție să conducă la o estimare îndeajuns de bună pentru control este o condiție sine qua non pentru detecția unui defect și izolarea acestuia în cadrul nodurilor rețelei. Astfel după cum a fost expus în capitolul 1 ecuațiile care guvernează relațiile intre viteza prin conducte și presiune dintr-un anumit punct sunt particularizări ale ecuațiilor Bernoulli-Euler sau Navier-Stokes. În cadrul unei rețele de apă a unui oraș, complexitatea rezolvării problemei crește semnificativ din varii motive precum:

- ansamblul de coduncte și noduri interconectate dă naștere unui sistem fizic greu de modelat matematic
- parametrii care pot influența calitatea soluțiilor precum: tipul materialului conductei și al nodului, elevația fiecărui nod, rugozitatea fiecărei conducte și depunerile de pe aceasta
- apariția unor factori exogeni care pot fi uneori greu de estimat - tiparul de utilizare al rețelei de către consumatori poate varia puternic
- apariția defectelor precum scurgerile în proximitatea unui nod

Ținând cont de complexitatea problemei în regim dinamic pentru a putea obține o soluție de regim staționar a rețelei este necesar să ignorăm evenimentele imprevizibile precum apariția unei scurgeri sau variațiile bruște ale consumului.

Ecuațiile de regim staționar includ condiții de conservare fluxului de apă:

$$\sum_{j=1}^{n} \mathbf{B}_{ij}\mathbf{q}_j = \mathbf{d}_i \tag{2.1}$$

Unde $q_i$ reprezintă debitul prin fiecare conductă iar $\mathbf{B}$ reprezintă matricea de adiacență a rețelei la echilibru, definită astfel:

$$\mathbf{B}_{ij} = \begin{cases} 1, & \text{conducta j intră în nodul i} \\ 0, & \text{conducta j nu este conectată la nodul i} \\ -1, & \text{conducta j iese din nodul i} \end{cases} \tag{2.2}$$

Partea de estimare a diferenței de presiuni (în engl. "Head-Flow differential") între două noduri interconectate se face utilizând formula Hazen-Williams ;[4]:

$$\mathbf{h}_i - \mathbf{h}_j = \frac{10.67 \cdot L_\ell}{C_\ell^{1.852} \cdot D_\ell^{4.87}} \cdot \mathbf{q}_\ell \cdot |\mathbf{q}_\ell|^{0.852} \tag{2.3}$$

unde:

- $\mathbf{h}$ reprezintă presiunea - măsurată de obicei în metru coloană de apă
- $C_l$ reprezintă coeficientul de rugozitate al conductei
- $D_l$ reprezintă diametrul conductei
- $L_l$ reprezintă lungimea conductei
- $q_l$ reprezintă debitul

Din ecuația empirică (2.1) termenul $R_{ij} = \frac{10.67 \cdot L_\ell}{C_\ell^{1.852} \cdot D_\ell^{4.87}}$ reprezintă rezistența conductei $ij$ iar dual, putem obține conductivitatea conductei $G_{ij} = \frac{1}{R_{ij}}$.

Având la dispoziție (2.1) și (2.3) putem exprima dependența debit presiune în regim staționar sub o formă matriceală compactă și cu o structură neliniară:

$$\mathbf{BG}\left[\left(-\mathbf{B}^\top \mathbf{h} + \mathbf{B}_f^\top \mathbf{h}_f\right) \times \left|-\mathbf{B}^\top \mathbf{h} + \mathbf{B}_f^\top \mathbf{h}_f\right|^{-0.46}\right] = \mathbf{d} \tag{2.4}$$

unde s-au luat în calcul și nodurile care au variații de presiune foarte mici - spre exemplu nodurile de tip tanc și nodurile de tip rezervor - termenul $\mathbf{B}_f^\top \mathbf{h}_f$ reprezintă contribuția acestor noduri la starea de echilibru a rețelei.

Parametrul $\mathbf{d}$ reprezintă cosumul pentru noduri $(L/s)$ și reprezintă un factor de incertitudine pentru întregul model. În cadrul unei simulări acesta poate fi estimat sau ales empiric, dar nu poate fi cunoscut cu precizie în fiecare moment. Problema aflării vectorului de consum a nodurilor intră în categoria problemelor legate de serii de timp și a estimării valorilor viitoare. Simulatoarele software transpun în general o variație a parametrilor de rețea (emitter-demand) în modificarea acestui parametru d.

Din cauza dificultății rezolvării unei ecuații matriceale neliniare, software-ul specializat trebuie să folosească diferite metode de optimizare ("Solver") pentru a putea obține o diferență cât mai mică între cazul estimat și rezultatul real al ecuației. Este important de reținut faptul că rezolvarea problemelor de programare neliniară cu constrângeri poate generea de fapt o problemă NP-completă, sau în unele cazuri chiar NP-dură ;[5].

## 2.2. Simulări folosind biblioteca EPANET

Dezvoltat la începutul anilor 90' de către USEPA (United States Environmental Protection Agency), EPANET a fost inițial privit ca un instrument pentru cercetare, acesta a devenit un standard de industrie la capitolul simulărilor software robuste pentru rețele de

apă, foarte multe pachete software proprietare de simulare hidraulică se bazează masiv pe EPANET, diferențele apărând la design-ul interfeței grafice și a manipulării datelor. Aceast program de simulare oferă utilizatorului posibilitatea de a-și defini într-un mod interactiv o rețea de apă configurând tipul de nod, legătura între oricare două noduri și posibilitatea de a adăuga și elemente active în rețea, pompe. Pentru a simula utilizatorul trebuie să își definească pentru fiecare nod un anumit debit cerut de utilizatori, o elevație, și o legătură cu alte noduri. Simularea se va desfășura pe o perioadă de timp definită cu pasul de eșantionare cât mai convenabil ;[6].

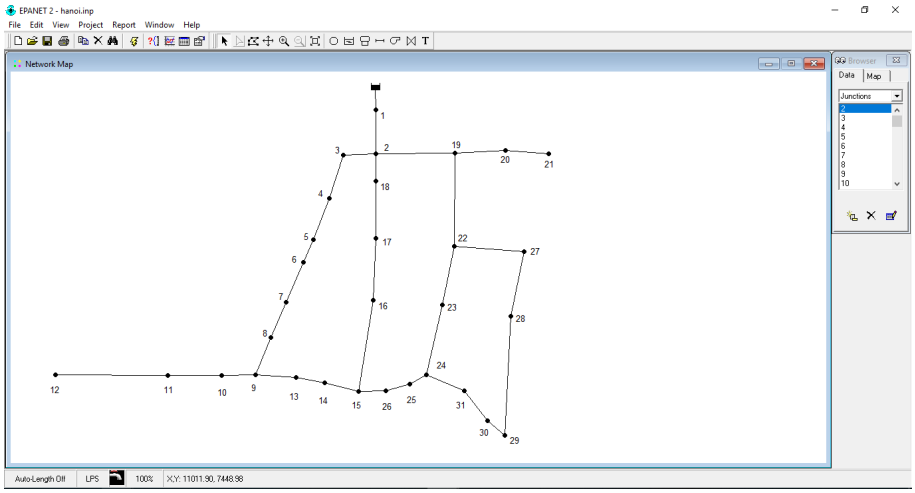



Figura 2.1.: Simulatorul EPANET

În urma execuției simulării EPANET va stoca toate datele în memorie și va putea reliza grafice și alte interogări complexe.

Modalitatea în care simulatorul EPANET reușește să obțină datele de simulare este prin implementarea eficientă a ecuațiilor Hazen-Williams (2.1) Darcy-Weisbach și Chezy-Manning, la fiecare perioadă de eșantionare algoritmul bazat de metoda gradientului rezolvă ecuațiile matriceale neliniare.

Pe lângă posibilitatea de a simula în cadrul unui program de sine stătător toate scenariile dorite, USEPA pune la dispoziție și un API (Application Programming Interface) pentru a putea realiza programatic simulările și modificările aferente fiecărui scenariu. Folosind în acest fel interfața pusă la dispoziție scrisă în limbajul C și oferită sub forma unei biblioteci dinamice (în Windows fișier .dll, în Linux fișier .so) programatorul are posibilitatea de a realiza propriul software specializat pentru simularea rețelelor de apă.

Particularizările programatice includ Demand-ul ( debitul de ieșire din nod către utilizatori L/s) fiecărui nod la orice moment de timp, proporționalitatea Demand-ului pentru a putea emula modul în care rețeaua este folosită de utilizatori de-a unei zile de lucru și unul din aspectele importante pe care EPANET le pune la dispoziție programatorilor este posiblitatea de a simula o scurgere în rețea emulată prin intermediul unui Emitter. Emitterul din biblioteca epanet este modelat ca un orificiu (perforație) prin care se poate scurge apa, fie din motive de a elibera presiunea sau din cauza unui defect. Ecuația care guvernează scurgerea prin acest orificu este:

$$q = Cp^{\gamma} \tag{2.5}$$

unde:

- q reprezintă debitul prin emitter
- C reprezintă o constantă de proporționalitate
- p reprezintă presiunea din joncțiune
- $\gamma$ reprzintă exponentul de presiune

Astfel apariția unui defect într-un anumit nod determină apriția unei relații liniare în funcție de presiunea din nod. Coeficientul $\gamma$ definde de tipul de joncțiune și de dimensiunea orificiului prin care se va scurge apa, pentru orificii relativ mici acesta ia valori de aproximativ 0.5 ;[6].

### 2.2.1. Schema bibliotecii ANSI C - EPANET

API-ul pus la dispoziție se bazează pe funcții C scrise într-o manieră modularizată, în ideea de a separa partea de analiză a rețelei și de modificare a parametrilor de partea de simulare hidraulică și calitativă. Astfel în imaginea de mai jos este prezentată modalitatea în care datele ajung și sunt folosite în cadrul simulatorului:

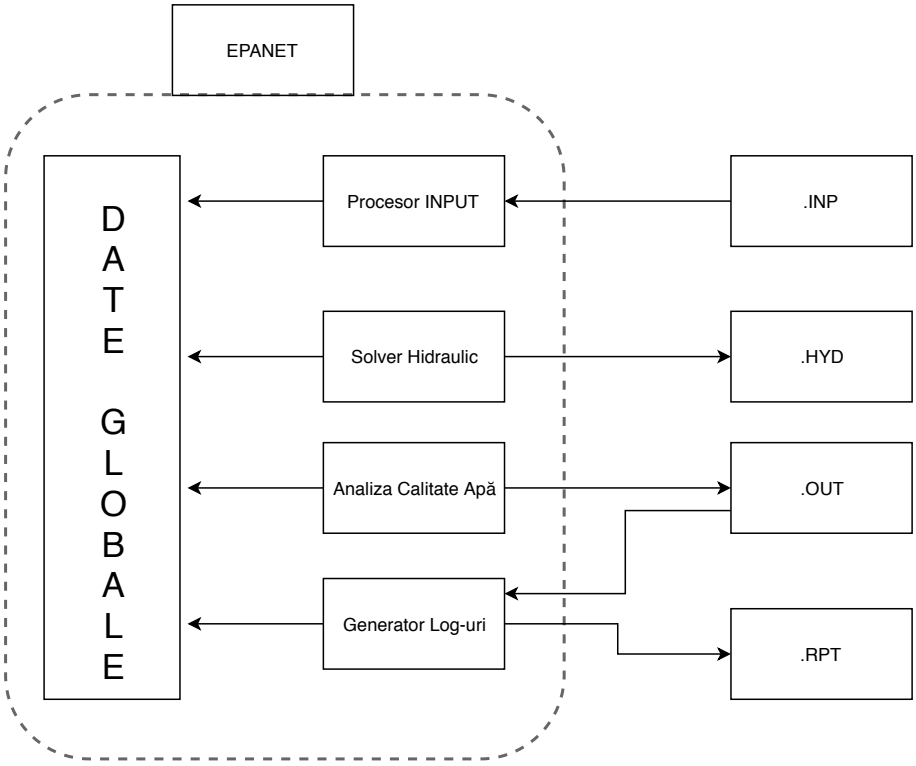



Figura 2.2.: Fluxul de date în biblioteca EPANET

Fișierele prezentate în schema de mai sus în partea dreaptă au semnificația:

- .INP - fișierul în care sunt definite topologia rețelei, tiparele de utilizare, caracteristicile elementelor active, valorile de Emitter și reguli complexe pentru funcționarea conductelor și pompelor

- .HYD - fișerul în care sunt stocate rezultatele simulării hidraulice
- .OUT - fișierul în care sunt stocate datele analizei calitative a apei
- .RPT - jurnalul în care biblioteca trece toate mesajele de eroare și atenționările cu privire la rețea

### 2.2.2. Structura fișierului de intrare .INP

Ceea ce stă la baza unei simulări este definirea input-ului rețelei de apă, anume fișierul INP. Fișierul INP - este un fișier text care conține anumite directive referitoare la componentele rețelei, tabelul de mai jos descrie toate aceste cuvintele cheie recunoscute de interpretor, împărțite în 4 categorii funcționale:

- Componente - se definesc toate componentele fizice ale sistemului noduri, conducte, valve, emitters
- Simulare - se definesc caracteristicile care trebuie urmate de solver în procesul de simulare
- Calitate - se definesc parametrii relevanți analizei de apă - Cantitatea de substanță dizolvabilă din fiecare nod, constantele de timp pentru reacțiile de ordin I și II
- Jurnalizare - secțiune responsabilă cu fixarea parametrilor de execuție a rețelei - pasul de eșantionare, unitățile de măsură, parametrii impliciți și modul de raportare a simulării

| Componente | Simulare | Calitate | Jurnalizare |
|---|---|---|---|
| [TITLE] | [CURVES] | [QUALITY] | [OPTIONS] |
| [JUNCTIONS] | [PATTERNS] | [REACTIONS] | [TIMES] |
| [RESERVOIRS] | [ENERGY] | [SOURCES] | [REPORT] |
| [TANKS] | [STATUS] | [MIXING] | |
| [PIPES] | [CONTROLS] | | |
| [PUMPS] | [RULES] | | |
| [VALVES] | [DEMANDS] | | |
| [EMITTERS] | | | |

Tabela 2.1.: Structura fișierului INP

Cele mai importante directive au fost trecute în tabelul 2.1 în culoarea albastru. Urmând ca fiecare să fie explicată mai în detaliu.

#### [JUNCTIONS]

Reprezintă structurile care definesc obiectele de tip nod din rețea, acestea vor avea ca atribute setate:

- ID - indicele nodului
- Elevation - înălțimea nodului (m)

- Base demand - debitul inițial către utilizatori (s-a folosit termenul demand pentru a desemna cerința de apă din rețea)
- Demand pattern ID - pentru fiecare nod se poate fixa un anumit profil de utilizare prin legarea acestuia cu un anumit obiect PATTERN

**[PIPES]**
Pipes reprezintă structurile care modelează conductele din rețea, pentru care atributele sunt:

- ID - indexul fiecărei conducte
- Start node ID - indexul nodului de plecare
- End node ID - indexul nodului destinație
- Length - lungimea conductei (m)
- Diameter - diametrul conductei (mm)
- Roughness coef - coeficientul de rugozitate al conductei
- Minor loss coef - coeficientul de pierdere al conductei
- Status - conducta poate fi în 3 stări , deschisă - OPEN, închisă - CLOSED sau polarizată - CV (lasă să treacă fluxul de apă într-o singură direcție)

**[EMITTERS]**
În lucrarea de față folosesc EMITTERS ca modalitate de a modela o scurgere de apă într-un anumit nod. Caracteristicile structurii sunt:

- Junction ID - indexul nodului unde se află scurgerea
- Flow coeff - coeficientul C din ecuația (2.5)

**[PATTERNS]**
Prin intermediul structurilor PATTERN se definesc profilurile de utilizare ale rețelei. Modalitatea de funcționare a demand-ului este prin multiplicarea valorii PATTERN cu valoarea de Base Demand a fiecărui nod. Astfel motorul de simulare va diviza întreaga perioadă de simulare astfel încât fiecare multiplicitate să primească un interval de timp egal.

**[DEMANDS]**
Structură care definește gradul de utilizare pentru fiecare nod din rețea. Atributele care definesc utilizarea sunt:

- Junction ID - indexul nodului unde se dorește modificarea debitului de utilizare
- Base Demand - magnitudinea debitului de utilizare

## 2.3. Integrarea EPANET cu Python

Pentru a ușura mecanismul de procesare și interpretare am decis să folosesc funcțiile de bibliotecă EPANET adaptate din C la limbajul Python de proiectul open-source [7]. Motivele pentru care am ales limbajul Python sunt rapiditatea dezvoltării aplicației, robustețea soluției și ușurința tratării erorilor. Python este un limbaj interpretat - deci tipurile variabilelor sunt decise dinamic în momentul rulării aplicației în schimb acesta nu permite conversia implicită a variabilelor la run-time, eliminând astfel foarte multe erori greu de depistat.

Prin utilizarea unei clase de adaptare, **??**, la API-ul de Python pus la dispoziție de [7] doresc să realizez o interfață mai ușoară pentru definirea, rularea simulărilor dar și pentru savlarea datelor într-un format care permite operabilitatea între mai multe programe, de exemplu Python-Matlab. Structura clasei de adaptare numită în engleză Wrapper (i.e. ambalaj al bibliotecii de funcții scrise în C) se bazează pe instanțierea unui obiect prin care se pot apela diferite funcții din biblioteca dinamică astfel încât să se obțină datele necesare. Metoda clasei prin care se face interogarea rețelei este:

```
1    def query_network(self, sim_dict):
```

Listing 2.1: Funcția de query

Parametrul *sim_dict* primit de funcția 2.1 reprezintă un fișier JSON (Java Script Object Notation) care are o structură ce permite interogarea rețelei în legătură cu diferite mărimi dorite.

```
1        {
             simulation_name : "name",
3            simulation_type: "H" or "Q"
             emitter_values : [ (node_index, emitter_value) ]
5            query : {
                 nodes : [ "EN_PRESSURE"
7                ]
                 links : [ "EN_VELOCITY"
9                ]
             }
11       }
```

Listing 2.2: Structură JSON intrare

Unde câmpurile reprezintă:

- simulation_name - numele simulării
- simulation_type - tipul simulării H - hidraulic, Q - calitate a apei
- emitter_values - un vector de cupluri (node_index, emitter_value), node_index reprezintă indexul joncțiunii iar emitter_value reprezintă magnitudinea scurgerii din acel nod
- query - reprezintă valorile care se doresc a fi interogate pentru fiecare componentă:
  - EN_PRESSURE - întoarcerea valorilor presiunilor pentru fiecare nod
  - EN_DEMAND - întoarcerea valorilor demand-ului pentru fiecare nod

– EN_VELOCITY - întoarcerea valorilor vitezei apei prin fiecare conductă

La apelarea funcției query_network se va returna de asemenea un dicționar, care poate fi convertit la format-ul JSON, pentru a putea fi folosit în alte procese de simulare și validare, spre exemplu într-un mediu de dezvoltare Matlab/Octave. Structura fișierului de ieșire este:

```
{
    SIM_NAME = simulation-name
    NODE_VALUES = [
        {
            "EN_PRESSURE" : [ [values for each node]]
            "EMITTER_VAL" :
            "EMITTER_NODE" :
        }
    ]

}
```

Listing 2.3: Structură JSON ieșire

Câmpul NODE_VALUES reprezintă un vector JSON-uri, fiecare element al acestui vector fiind în fapt rezultatul numeric al unei simulări pentru câmpurile specificate la fișierul de intrare 2.2.

Un pas important în dezvoltarea Wrapper-ului peste EPANET a reprezentat tratarea erorilor returnate de funcțiile de C. În momentul apelării unei funcții de C, EPANET returnează o pereche (a, b) unde a reprezintă valoarea efectivă interogată iar b reprezintă codul erorii. Tratarea excepțiilor s-a făcut cu ajutorul funcției:

```
    @staticmethod
    def _getncheck(ret_val):

        # check the return code
        if isinstance(ret_val, list):
            if ret_val[0] == 0:
                # everything OK
                return ret_val[1]
            else:
                err_msg = ENgeterror(ret_val[0], 100)
                raise EpanetError(err_msg)
        else:
            if ret_val is not 0:
                err_msg = ENgeterror(ret_val, 100)
                raise EpanetError(err_msg)
```

Listing 2.4: Cod pentru tratarea erorilor

și prin definirea unei excepții customizate pentru clasa ENWrapper scrisă

```
class EpanetError(Exception):

    def __init__(self, err_msg):
        super().__init__(err_msg)
```

Listing 2.5: Definirea Excepției

## 2.4. Simulări pe rețeaua de apă din Hanoi

Având ca susținere fișierul pentru rețeaua de apă din Hanoi 1.1 și biblioteca EPANET cu codul de adaptare Python, pot realiza simulări asupra acestei rețele de apă. Inițial voi explicita profilurile de utilizare a rețelei de apă urmând ca mai apoi să arăt modul în care rețeaua funcționează în mod nominal (fără nici un defect).

Parametrii funcționali rețelei hanoi.inp sunt următorii:

- pasul de eșantionare al simulării 15min
- pasul de eșantionare al pattern-ului 15min
- Perioada de simulare este de 24h

Graficul de utilizare al rețelei are o distribuție centrată în jurul orelor dimineții - acest lucru fiind corelat cu faptul că atunci este momentul în care cei mai mulți utilizatori își încep activitatea și consumă apă. Astfel rețeaua este supusă unui stres mai mare în timpul orel matinale față de exemplu de perioada nopții unde valorile multiplicității ajung la valori aproximativ de două ori mai mici.

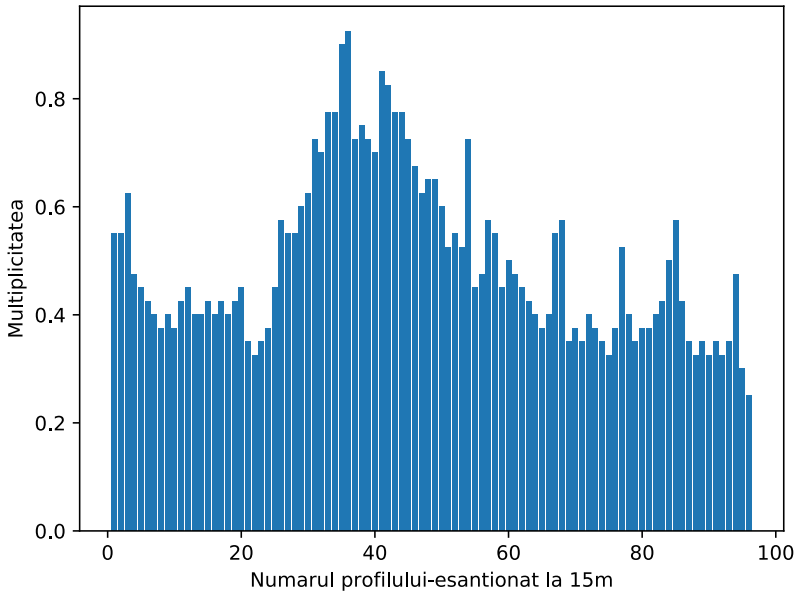



Figura 2.3.: Profilul normal de utilizare

Pentru rularea unei simulări nominale fișierul JSON de intrare poate avea câmpul de emitter_values gol, obținându-se asftel caracteristica tranzitorie a presiunii, a debitului din noduri și a vitezei fluxului de apă din conducte.

## 2.5. Prezentarea măsurilor de interes preluate din simulări

Problema care se pune în continuare alegera unui subset din mărimile prelevate din simulator astfel încât discernerea între un caz de utilizare normal și unul defectuos să fie cât mai ușoară.

### 2.5.1. Simulare dinamică pentru încărcare nominală

În regim nominal ansamblul rețelei de apă nu este supus la nici un defect, funcționarea fiind corespunzătoare profilului de utilizare normal 2.3. Datele care sunt extrase din simularea nominală și prezintă interes pentru analiză sunt corespunzătoare presiunii din fiecare nod. De asemenea vor fi prezentate și celelalte măsurători - viteza apei prin conducte și debitul către utilizatori.

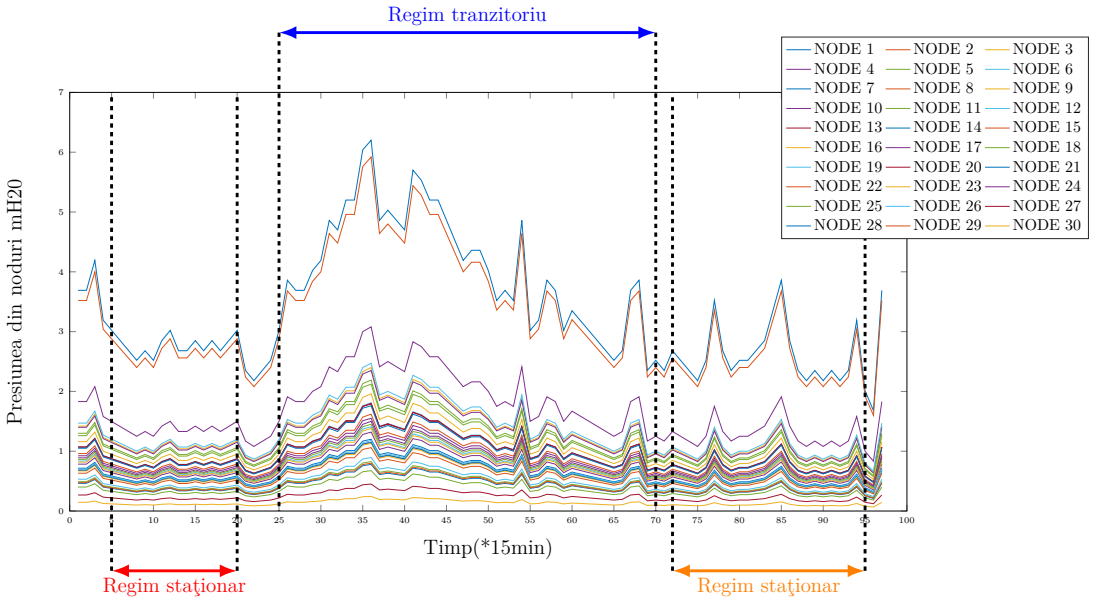
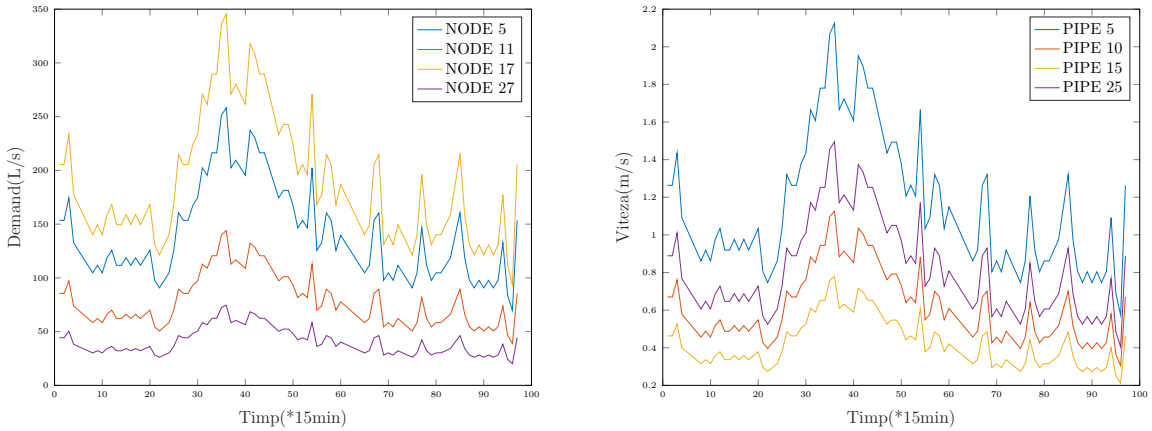



Figura 2.4.: Profilul nominal al presiunii

După cum se poate observa în 2.4 profilul nominal al presiunii intră în tiparul profilului de cerere prezentat în 2.3, unde în jurul orelor dimineții, când activitatea utilizatorilor este asociată unui regim dinamic al rețelei, iar orele nopții sunt caracterizate de marimi lent variabile. Astfel pe acest grafic s-au putut evidenția 3 zone importante, anume, cele două zone de regim staționar în intervalele $[5, 20] * 15min$ și $[72, 95] * 15min$ și zona de regim dinamic în intervalul $[23, 70] * 15min$. Este de menționat că aceste intervale au fost selectate empiric în baza unei analize grafice asupra caracteristicilor presiune-viteză-debit.

### 2.5.2. Alte mărimi de interes din simulatorul EPANET

Caracteristicile de viteză și debit urmează o caracteristică asemănătoare cu caracteristica de presiune din fiecare nod, o explicație naturală provenind de la faptul că presiunea este considerată ca fiind înălțimea nivelului de apă dintr-un nod.




(a) Profil nominal demand

(b) Profil nominal viteza

Figura 2.5.: Rezultate simulări nominale

După cum se observă în 2.5 zonele de regim staționar și de regim dinamic se păstrează în continuare și pentru caracteristicile de viteză și debit. De asemenea este important de menționat că profilurile nominale ale celoralte noduri decât cele ilustrate în figurile de mai sus urmează aceeași caracteristică dinamică, alegerea nodurilor și conductelor respective a fost făcută pentru a păstra figura curată.

# 3. Scenarii pentru defecte și simulări

## 3.1. Definirea defectelor

Defectele sunt simulate modificând parametrul $C$ din ecuația emitter-ului (2.5). Modalitatea prin care se execută în cod simularea unui defect este prin apelarea metodei:

```
    def set_emitter(self, node_index, emitter_val):
2       if self.network.nodes[node_index].node_type is EN_JUNCTION:
            ENSim._getncheck(self.ENsetnodevalue(node_index, EN_EMITTER,
    emitter_val))
```

Listing 3.1: Funcție pentru simularea defectelor

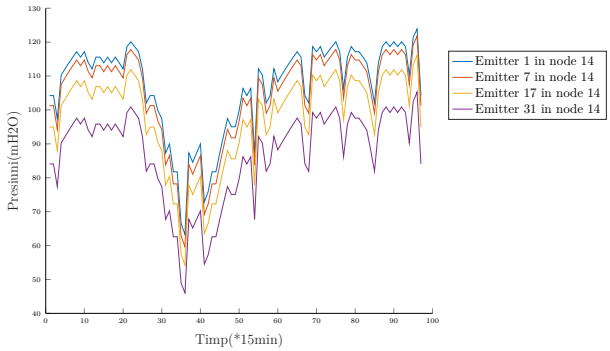
parametrii funcției *set_emitter* sunt:

- node_inde - indexul nodului în care se simulează defectul
- emitter_val - magnitudinea defectului

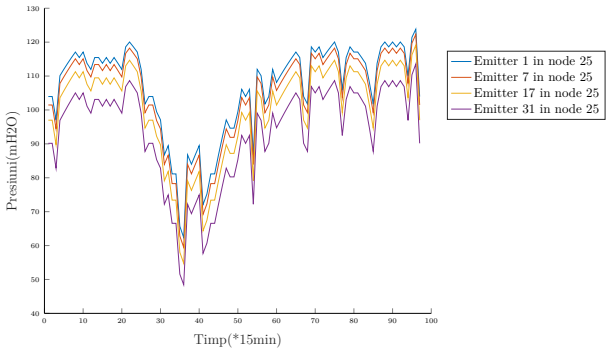
Metoda mai întâi verifică dacă nodul cu indexul $node_{index}$ reprezintă doar o joncțiune apoi setează magnitudinea defectului în nodul primit cu ajutorul funcției de bibliotecă **ENsetnodevalue**

## 3.2. Simulare dinamică pentru defecte în diferite noduri

În continuare vom considera un scenariu de defect pentru rețea care constă în modificarea succesivă a parametrului de proporționalitate din relația de calcul a debitului de emitter (2.5). În imaginile următoare voi considera mai multe magnitudini de defect într-un anumit nod și voi reprezenta grafic răspunsul în timp al rețelei în același nod.




(a) Profile cu defect în nodul 14



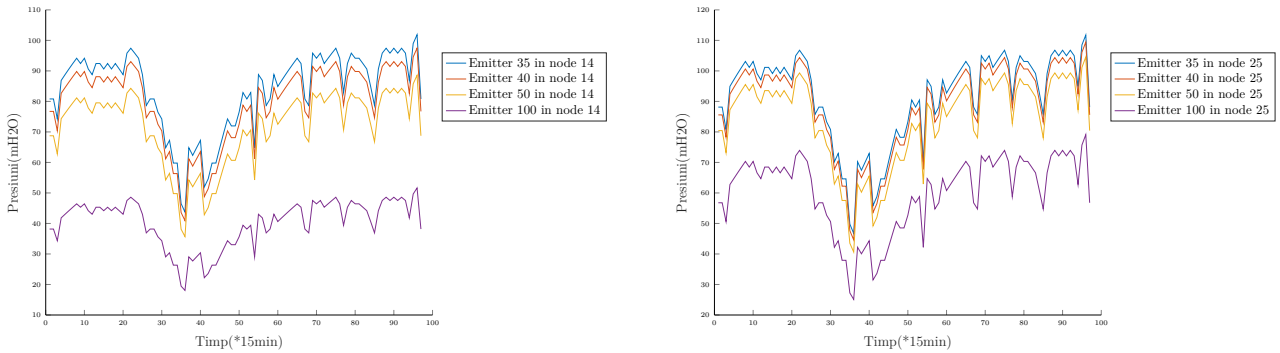

(b) Profil cu defect în nodul 25

Figura 3.1.: Rezultate simulări defecte ușoare

După cum se poate observa în imaginile 3.1 variația emitter-ului într-un nod produce în mod evident o modificarea a modului comun al caracteristicii $timp - presiune$. Din punctul de vedere al magnitudinilor de simulare pentru defecte, am considerat 2 clase de defecte, anume:

- defecte ușoare (soft faults) - cu valorile coeficientului de emitter mai mici de 35
- defecte puternice (hard faults) - cu valorile emitter mai mari de 35

Cele din urmă produc și modificări ale caracteristicii dinamice, introducând distorsiuni sau aplatizări ale mărimilor măsurate. Reprezentarea defectelor hard este reprezentată în figurile de mai jos:




(a) Profile cu defect puternic în nodul 14

(b) Profil cu defect puternic în nodul 25

Figura 3.2.: Rezultate simulări defecte puternice

Se observă de exemplu că pentru o valoare a emitter-ului de 100 caracteristica dinamică este deja modificată din cauza scurgerilor puternice din nod.

Este relevantă împărțirea defectelor în mai multe clase de magnitudini pentru a putea valida un model de clasificare. Spre exemplu este normal să se întrebe dacă un model antrenat pe baza unui set date corespunzător unor magnitudini normale $C \in (0, 35)$ poate da rezultate semnificative pentru un set de date cu magnitudini ale emitter-ului puternice $C \geqslant 35$.

## 3.3. Preprocesarea datelor

În urma extragerii datelor din rețea este extrem de importantă etapa de prelucrare și preprocesare a datelor. Domeniul de preprocesare a datelor este unul extrem de vast și important în domeniul de învățare automată (engl. Machine Learning) și procesare de semnal. Preprocesarea datelor este etapa în care datele de intrare pentru un algoritm sunt aduse la o formă optimă pentru desfășurarea procesului impus, de exemplu în domeniul clasificării este important ca algoritmii să primească date care să fie scalate într-un anumit domeniu, pentru a asigura convergența[8], [9]. Alegerea metodei de preprocesare este strâns legată de tipul de date disponibile și de starea acestora. În cazul rețelelor de apă, unde am ales caracteristica presiunii ca mărime de intrare pentru algoritm și ținând cont de răspunsul în timp al rețelei am considerat ca fiind necesare următoarele operații:

- eliminarea frontului comun și extragerea diferenței dintre semnalul nominal și cel măsurat în rețea
- filtrarea semnalului obținut anterior

## 3.4. Nomenclatura mărimilor alese

Pentru a menține rigurozitatea și eleganța metodelor folosite este nevoie de o definire matematică pentru toate mărimile și metodele de filtrare folosite.

### 3.4.1. Presiunea în regim dinamic

Reprezintă o funcție de timp:

$$p_i : \mathbb{R} \longrightarrow \mathbb{R}^n, i \in V \tag{3.1}$$

unde $n$ reprezintă numărul de noduri al rețelei, iar $i$ reprezintă indexul nodului. Deoarece cazurile tratate în această lucrare reprezintă momente discrete de timp este important să definim presiunea măsurată în intervalele discrete în care este simulat procesul:

$$\mathbf{p}_i \in \mathbb{R}^{n \times p_{sim}} \tag{3.2}$$

unde $p_{sim}$ reprezintă numărul de eșantioane pentru fiecare măsurătoare. Mergând mai departe în analiza simulării este de asemenea important să definim mărimea afectată de un defect în nodul $j$, de magnitudine $m$ și măsurată în nodul $i$:

$$\mathbf{p}_i^{j,m} \in \mathbb{R}^{n \times p_{sim}} \tag{3.3}$$

Pentru cazul în care magnitudinea $m$ ia valori nule, atunci vom considera notația mărimii nominale:

$$\mathbf{p}_i^{j,0} = \mathbf{p}_i^{nom}, \forall j \in V \tag{3.4}$$

Pentru valorile presiunii recoltate din rețea în nodul $i$ despre care nu se cunoaște nici o informație, vom considera notația

$$\widehat{\mathbf{p}}_i \tag{3.5}$$

### 3.4.2. Presiunea în regim static

Considerând o plajă de momente de timp situate între indicii $rs_1 : rs_2$ unde se afla valorile de regim staționar ale procesului, putem defini o medie a regimului static în felul următor:

$$\overline{\mathbf{p}}_i^{j,m} = \frac{1}{rs_1 - rs_2 + 1} \sum_{k=rs_1}^{rs_2} \mathbf{p}_i^{j,m}[k] \tag{3.6}$$

În aceeași manieră definim și media presiunii nominale în regim static:

$$\overline{\mathbf{p}}_i^{j,0} = \overline{\mathbf{p}_i}^{nom}, \forall j \in V \tag{3.7}$$

Media presiunii măsurată în nodul $i$ și despre care nu se cunosc informații în legătură cu valoarea și poziția defectului:

$$\overline{\widehat{\mathbf{p}}}_i = \frac{1}{rs_1 - rs_2 + 1} \sum_{k=rs_1}^{rs_2} \widehat{\mathbf{p}}_i[k] \tag{3.8}$$

### 3.4.3. Reziduuri

Așa cum a fost discutat în secțiunea 3.3, preprocesarea datelor are un rol important iar în cazul analizei și clasificării defectelor în rețelele cu apă, este nevoie să definim caracteristica prelucrată care va fi folosită mai apoi în procesul de izolare a defectelor. Reziduul absolut reprezintă diferența dintre valoarea măsurată în rețea și valoarea nominală, aici putem discerne două cazuri: Reziduu temporal:

$$\mathbf{r}_i^{j,m} = \mathbf{p}_i^{j,m} - \mathbf{p}_i^{nom} \tag{3.9}$$

Reziduu atemporal, calculat ca diferența dintre cele două valori mediate pe intervalul staționar al caracteristicii:

$$r_i^{j,m} = \overline{\mathbf{p}}_i^{j,m} - \overline{\mathbf{p}}_i^{nom} \tag{3.10}$$

iar pentru valorile reziduului despre care nu se cunosc încă lucruri folosim notația din stilul anterior:

$$\widehat{r}_i = \overline{\widehat{\mathbf{p}}}_i - \overline{\mathbf{p}}_i^{nom} \tag{3.11}$$

Alte tipuri de reziduuri pre-procesate sunt relative:

$$rrelativ_i^{j,m} = \frac{r_i^{j,m}}{\overline{\mathbf{p}}_i^{nom}} \tag{3.12}$$

Reziduurile normate:

$$rnorm_i^{j,m} = \frac{r_i^{j,m}}{\left\| r_{1:n}^{j,m} \right\|} \tag{3.13}$$

Reziduurile scalate:

$$rscal_i^{j,m} = \frac{r_i^{j,m} - \min r_{1:n}^{j,m}}{\max r_{1:n}^{j,m} - \min r_{1:n}^{j,m}} \tag{3.14}$$

Ca semnificație notațiile prezentate în 3.4 care conțin simbolul $\widehat{\ }$ fac referire la datele folosite pentru validarea modelului iar valorile unde se specifică nodul defectului și magnitudinea acestuia sunt considerate ca fiind date de antrenare și testare. Astfel în contextul definirii setului de dat pe care vom aplica algoritmii de clasificare va trebui să definim matricea:

$$\mathbf{R} < tip >^{j,m} \in \mathbb{R}^{n_d \times n} \tag{3.15}$$

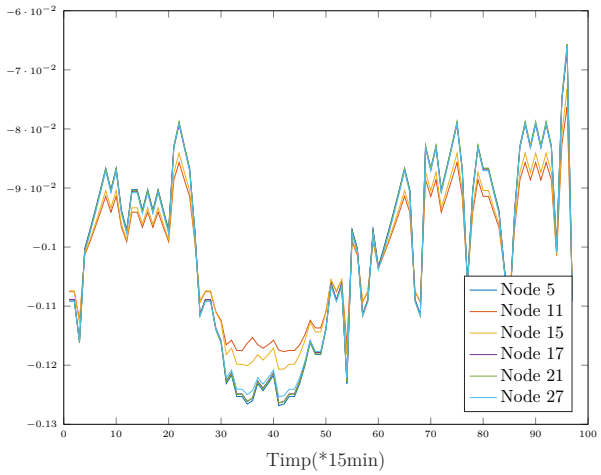
Unde croșetele din formulă reprezintă un înlocuitor pentru metoda de reziduu folosită iar $n_d$ reprezintă numărul de defecte tratate în setul de date. De asemenea pentru fiecare linie a matricei (3.15) putem defini perechea

$$(\mathbf{R} < tip > (d, :), y_d) \tag{3.16}$$

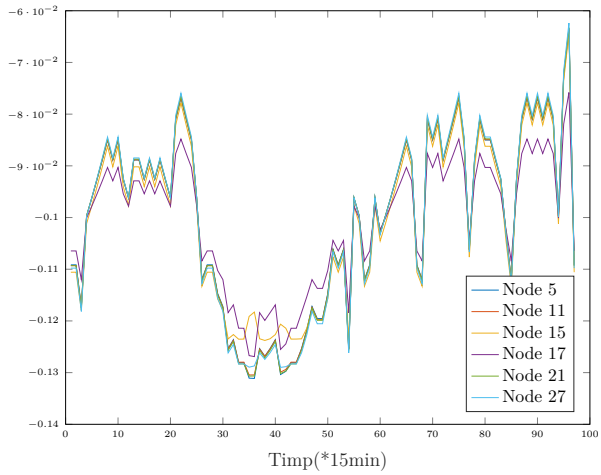
Unde $\mathbf{R} < tip > (d, :)$ reprezintă răspunsul rețelei prin reziduuri la defectul $d$. Iar $y_d$ reprezintă eticheta pentru acest set de date, anume, nodul în care a avut loc defectul.

## 3.5. Calcul și prezentare reziduuri

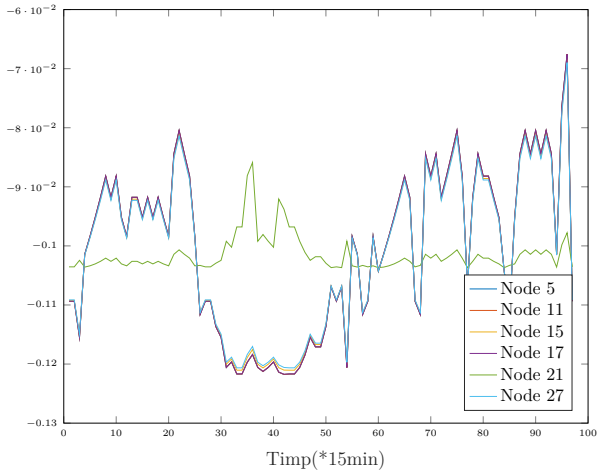
În continuare vom prezenta grafic reziduurile temporale normalizate care apar în rețea pentru diferite scenarii ale defectelor definite anterior. Astfel vom considera nodurile de măsurătoare ca o submulțime a lui $V' \subset V$ și $V = \{5, 11, 15, 17, 21, 27\}$, nodurile în care s-au simulat defecte sunt $V_d = \{11, 17, 27, 29\}$.



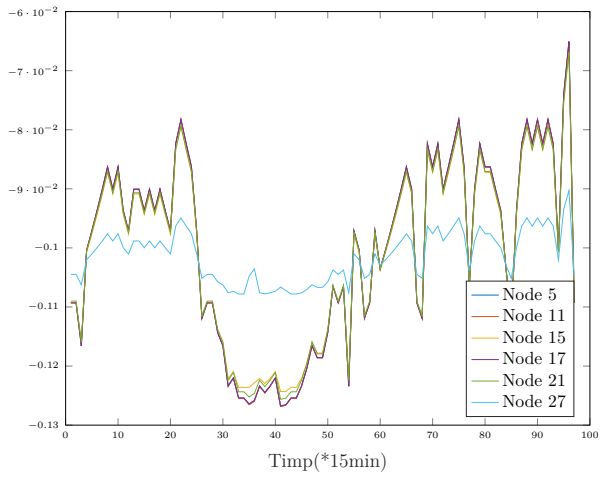

(a) Reziduuri pentru defect în nodul 11, magnitudine 29




(b) Reziduuri pentru defect în nodul 17, magnitudine 29




(c) Reziduuri pentru defect în nodul 21, magnitudine 29

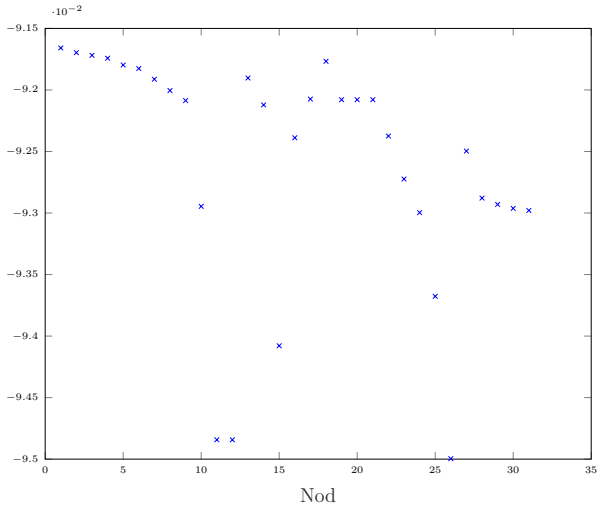



(d) Reziduuri pentru defect în nodul 27, magnitudine 29

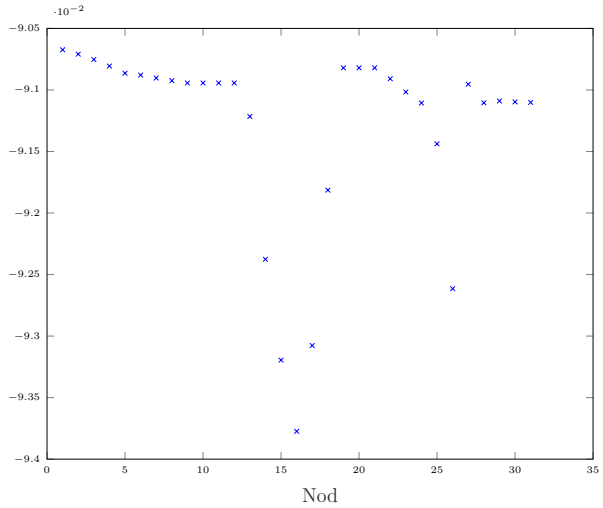
Figura 3.3.: Reziduuri rețea

Se poate observa că în figurile 3.3 reziduul cel mai pronunțat ca funcție de timp se găsește în nodul în care se simulează și defectul - lucru natural și de așteptat. O caracteristică importantă a acestei rețele de apă este faptul că există o dependență între diferitele răspunsuri în timp ale caracteristicii de presiune, fapt care ne permite să exploatăm redundanțele din rețea și să prezicem cu o acuratețe relativ ridicată defectele.

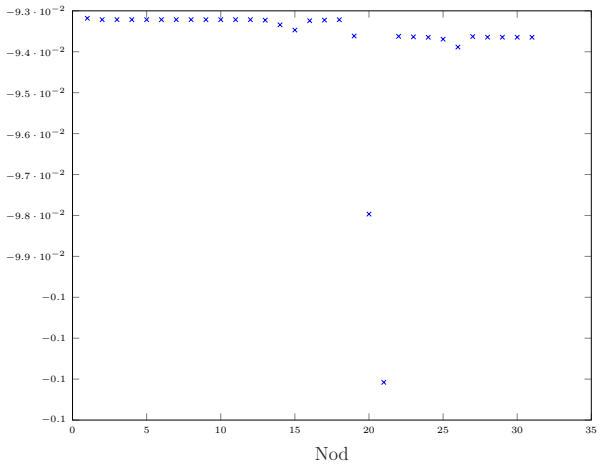
Este necesar acum să prezentăm profilurile reziduurilor atemporale, care în final vor reprezenta caracteristicile de intrare pentru algoritmul de clasificare și selecție de senzori.



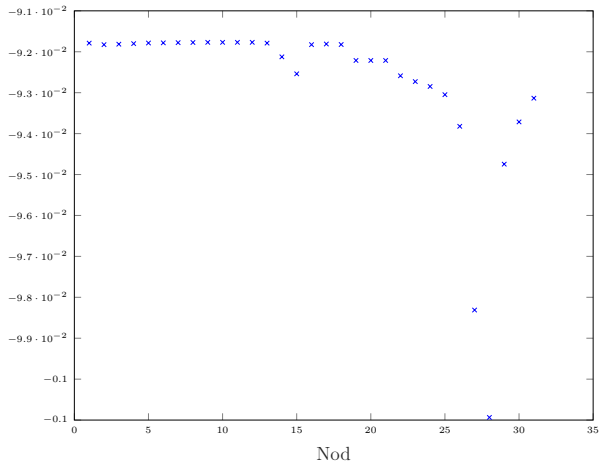

(a) Reziduuri pentru defect în nodul 11, magnitudine 25




(b) Reziduuri pentru defect în nodul 17, magnitudine 25




(c) Reziduuri pentru defect în nodul 21, magnitudine 25




(d) Reziduuri pentru defect în nodul 27, magnitudine 25

Figura 3.4.: Reziduuri atemporale rețea

Asemenea reziduurilor de la 3.3 se poate observa că simularea unui emitter într-un nod va determina un răspuns puternic în nodul respectiv și în vecinătatea nodului afectat

## 3.6. Metodă preliminară de selecție a senzorilor

O metodă prin care se poate decide și evalua importanța senzorilor într-o rețea este prin construirea matricei de reziduuri (3.15), asupra căreia aplicăm o operație de scalare pe coloane, pentru a aduce valorile acesteia în intervalul $[0, 1]$. Pentru experimentul în care simulăm în fiecare nod un emitter de 25 obținem grafic o matrice $\mathbf{R}scaled \in \mathbb{R}^{31\times31}$:

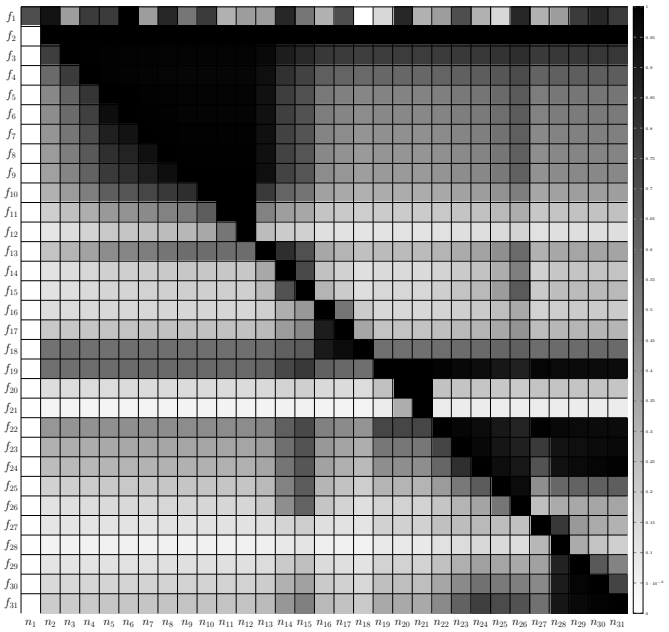


Figura 3.5.: Matricea de reziduuri scalate

După cum se observă în 3.5 fiecare coloană a matricei $\mathbf{R}scaled$ reprezintă răspunsul prelucrat - scalat în intervalul $[0, 1]$ - al unui nod, valorile care tind spre o culoare mai închisă de negru sunt răspunsuri mai pronunțate iar valorile care se apropie de alb reprezintă valori mai puțin evidențiate ale reziduului în nod(i.e. coloana de apă nu a scăzut atât de mult în nodul respectiv). Analizând matricea se poate observa care noduri răspund mai bine la anumite răspunsuri, o caracteristică importantă a acestei matrice este că, o alură diagonală - semnificația elementelor diagonale fiind răspunsul nodurilor afectate de defecte aplicate în ele însuși. Dacă matricea de reziduuri ar fi avut o caracteristică diagonală perfectă - elementele nenule s-ar fi regăsit doar pe aceasta - problema de clasificare ar fi fost una trivială și ar fi implicat amplasarea unui senzor în fiecare nod. Cazul real impune dependențe între presiunile, debitele din fiecare nod, dar și vitezele din fiecare conductă, astfel, fiecare defect separat va avea o "amprentă" unică - prin care se diferențiază de celelalte și una comună. Scopul este găsirea unui subset al mulțimii de noduri care să poată identifica defectele cu pierderi minime de performanțe.

### 3.6.1. Binarizarea matricei de reziduuri

Pentru a putea reține un răspuns binar în cadrul matricei este important să considerăm o limită $l$ peste care răspunsul senzorului este luat în considerare sau nu. Astfel definim matricea binară $\mathbf{M}$ de aceeași dimensiune ca și $\mathbf{R}scaled$ dar peste care aplicăm o operație de binarizare:

$$\mathbf{M}_{i,j} = \begin{cases} 1 & \text{dacă } \mathbf{R}scaled_{i,j} \geq l \\ 0 & \text{dacă } \mathbf{R}scaled_{i,j} < l \end{cases} \tag{3.17}$$

În continuare vom defini notațiile matematice pentru mulțimile defectelor și a nodurilor. Este important să considerăm astfel mulțimea răspunsurilor la defecte $F = \{\mathbf{f}_i | i \in V\}$ iar fiecare element $\mathbf{f}_i$ al mulțimii este definit ca un vector format din valori binare cu însemnătatea:

$$\mathbf{f}_i[k] = \begin{cases} 1 & \text{dacă nodul k răspunde la defectul i} \\ 0 & \text{dacă nodul k nu răspunde la defectul i} \end{cases} \tag{3.18}$$

Echivalent putem defini mulțimea răspunsurilor nodurilor $N = \{\mathbf{n}_i | i \in V\}$, pentru fiecare element $\mathbf{n}_i$ avem:

$$\mathbf{n}_i[k] = \begin{cases} 1 & \text{dacă defectul k este detectat de nodul k} \\ 0 & \text{dacă defectul k nu este detectat de nodul k} \end{cases} \tag{3.19}$$

#### Exemplificare

Considerând o limită $l = 0.65$ pentru matricea de răspunsuri scalate putem obține o matrice binară calculată după formula (3.17). Forma matricei $\mathbf{M}$ este:

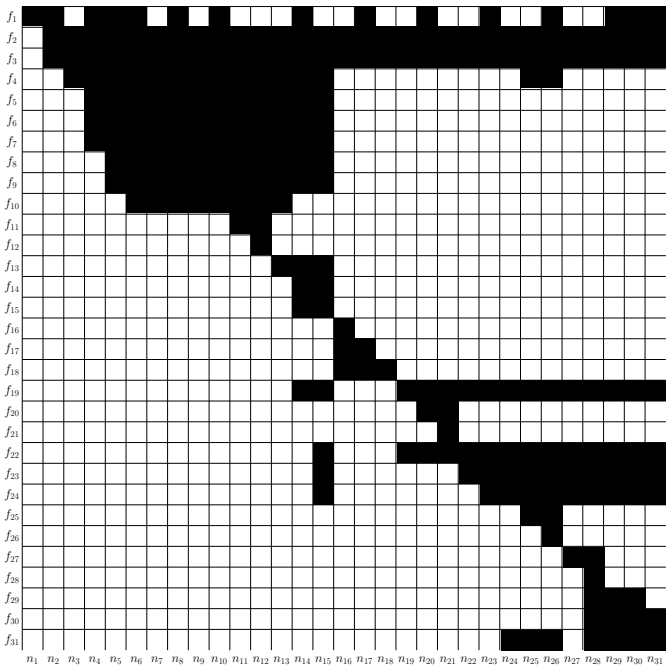



Figura 3.6.: Matricea de răspunsuri binarizate

## 3.6.2. Selecția senzorilor

Având definite elementele pentru interpretarea decizională a datelor preluate de la senzori, putem formula o metodă prin care să selectăm acei senzori care oferă informațiile

cele mai importante. După cum se observă și în matricea reziduurilor 3.5 există foarte multă redundanță în sistemul complex al rețelei de apă care întâmpină perturbații precum scurgeri. Ținând cont că de faptul că fiecare defect are un răspuns diferit, putem să decidem care este subsetul de senzori care reușește să ofere informații îndeajuns de relevante în legătură cu evenimentele din rețea, ca exemplu trivial putem afirma că un nod care răspunde la toate defectele oferă informații relevante pentru detecția de defecte dar nu și pentru izolarea defectelor.

Pentru problema selecției de senzori dorim să obținem o submulțime de noduri $V_s \subset V^j$ cu proprietatea că aceasta va avea un cardinal cât mai mic și defectele detectate de senzorii selectați sunt cât mai multe. Problema aceasta reprezintă de fapt o abordare a unei probleme NP-complete anume Minimum Set Cover - problema acoperirii minime a mulțimilor [10].

### Problema MSC

Fiind dată o mulțime univers $U$ cu $|U| = n$ și o mulțime cu cardinal $m$ $S = \{S_i | S_i \subset U\}$ având proprietatea că $\bigcup_{i=1}^{m} S_i = U$ se dorește găsirea celei mai mici partiții a mulțimii $S$ a cărei reuniuni va fi egală cu $U$ [11].

În cazul rețelelor de apă mulțimea $U$ reprezintă de fapt mulțimea defectelor pentru care se dorește găsirea partiției cu cardinal minim a mulțimii de noduri care să fie sensibilă la toate defectele [10].

Această problemă poate fi pusă sub forma unei probleme de optimizare matematică, având la dispoziție matricea binară $\mathbf{M}$ și un vector de selecție al senzorilor $\alpha$:

$$\min_{\alpha \in \mathbb{R}^n} \quad \sum_j \alpha_j \tag{3.20a}$$

$$\text{s.l.:} \quad \sum_j M_{i,j}\alpha_j \geq 1 \tag{3.20b}$$

Minimizarea sumei vectorului $\alpha$ din 3.20a se referă la selectarea unui număr cât mai mic de senzori pentru a fi plasați în rețea. Constrângerea atașată acestei probleme de programare liniară (4.12b) are rolul de a impune ca pentru fiecare defect să existe cel puțin un senzor care să îl detecteze.

### Modalitatea de rezolvare a problemei MSC

Având la dispoziție toate datele de intrare pentru problema MSC, am ales să folosesc toolbox-ul **YALMIP - MATLAB** pentru a afla nodurile cu importanța cea mai mare, rezolvând problema de optimizare (4.12). **YALMIP** este un toolbox specializat pentru probleme de optimizare și a fost dezvoltat așa fel încât programatorii să poată rezolva

aceste ecuații într-un mod mult mai ușor - transpunând în cod ecuațiile de minimizare/-maximizare și inegalitățile - fără să mai fie nevoie să aducă problema inițială la o formă intermediară [12].

Codul **MATLAB** care se rulează pentru a rezolva acestă problemă este:

```
thr = 0.65; % setarea limitei de sensibilitate a senzorilor
M = double(R > thr); % binarizarea senzorilor
alpha = binvar(size(M,2), 1); % definirea variabilei binare alpha

constrangere = [M * alpha >=1];
obiectiv = sum(alpha); % criteriul de minimizare

optimize(constrangere, obiectiv)
alpha = value(alpha);
sum(alpha); % numarul de noduri care algoritmul de optimizare numerica a
    selectat
sum(M(alpha == 1, :),2) >= 1; % verif daca se indeplineste conditia de set
    covering
find(alpha) % nodurile selectate de algoritm
```

Listing 3.2: Rezolvare problemă MSC

Nodurile alese de această configurație a problemei sunt $V_s = \{11, 13, 15, 20, 26, 27\}$ prezentate pe graful rețelei în figura următoare:

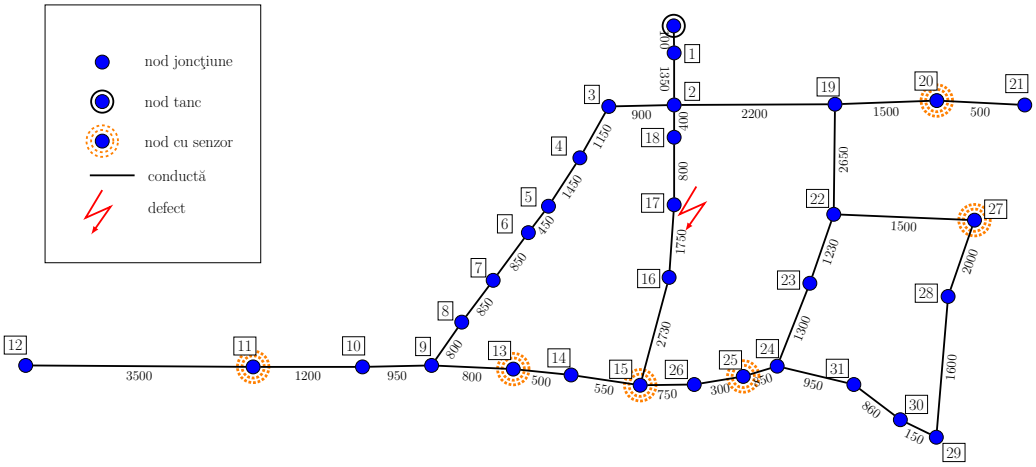



Figura 3.7.: Rețeaua cu senzorii plasați

# 4. Clasificarea defectelor folosind tehnici de învățare automată

## 4.1. Problematica domeniului de învățare automată

Învățarea automată reprezintă un subdomeniu al inteligenței artificiale, stiință care se ocupă cu dezvoltarea de algoritmi care să transpună în domeniul mașinilor acele sarcini care sunt ușor de făcut pentru om, spre exemplu: interpretarea limbajului natural și a imaginilor, recunoașterea obiectelor, menținerea echilibrului sau luarea de decizii. Deși aceste acțiunui par destul banale pentru un om, transpunerea acestei probleme pentru un calculator este mult mai grea din cauza faptului că însuși definirea matematică este anevoioasă - majoritatea formulându-se ca probleme de optimizare, dorindu-se minimizarea erorii de predicție a estimatorului.

Istoria învățării automate (*Machine Learning*) a pornit la mijlocul secolului XX din dorința de a depăși stadiul programelor "statice", care doar aduc blocuri de memorie, le prelucrează și afișează informația într-un mod relevenant, și de a crea programe "dinamice" care folosind datele furnizate, să descopere și să învețe singure anumite reguli prin care să extragă informațiile de interes. Una din primele aplicații ale domeniului de machine learning a fost în teoria jocurilor pentru jocul de dame [13], unde cercetătorii s-au ocupat de dezvoltarea unui model care să devină un jucător mai bun odată cu jucarea jocurilor. Jocurile precum șahul, damele, și recent jocul go [14], reprezintă provocări foarte mari pentru cercetătorii în inteligența artificială din simplul motiv că oferă o interfață foarte simplă între mașină și mediul cu care interacționează, dar, din punct de vedere computațional sunt extrem de complexe, spre exemplu, jocul de șah și go a fost dovedit ca fiind în clasa de complexitate EXPTIME, probleme rezolvabile în timp exponențial.

În domeniul recunoașterii de modele unul dintre roadele acestui câmp, luat ca a tare în momentul de față, este conceptul de OCR *Optical Character Recognition* care reprezintă o colecție de algoritmi și metode de prelucrare care au ca scop final transformarea unei imagini cu text într-un șir de caractere, folosit cu succes încă de la începutul anilor 90 în sistemul poștal american pentru recunoașterea codurilor poștale [15].

### Tipuri de algoritmi de învățare automată

Deși descrierea scopului învățării automate poate părea destul de generalistă, metodele aparținând acestui câmp de studiu sunt extrem de variate, unele modele apărând din câmpuri adiacente informaticii și ingineriei, de exemplu psihologia (modelul acțiune recompensă) și medicina (modelarea neuronului și a rețelelor neurale). O împărțire a algoritmilor de machine learning bazată pe tipul setului de date este următoarea

- învățare supervizată - seturi de date etichetate
- învățare nesupervizată - seturi de date neetichetate
- învățare semi-supervizată - combinație între exemple etichetate și neetichetate
- învățare ranforsată *reinforcement learning* - modele bazate pe interacțiunea unui agent cu mediul

Dintre aceste patru subclase le vom detalia pe primele două, algoritmii de interes pentru această lucrare făcând parte din ele.

### 4.1.1. Învățarea supervizată

Învățarea supervizată presupune folosirea unui set de date de tipul $S = \{(x_i, y_i)|i = \overline{1, N}\}$ unde $x_i$ este vectorul de caracteristici, $y_i$ reprezintă clasa din care face parte acest exemplu iar $N$ reprezintă numărul de exemple din $S$ pentru a găsi parametrii optimi ai unui estimator $f : X \to Y$ astfel încât predicțiile acestuia să fie cât mai apropiate de etichetele setului de date *ground truth* [16]. Considerând definiția de mai sus putem trage concluzia că problemele de învățare supervizată se reduc la probleme de optimizare:

$$\min_{\gamma} \sum_{i}^{N} \mathcal{L}(f(x_i|\gamma), y_i) \tag{4.1}$$

Unde

- $\mathcal{L}$ reprezintă funcția de cost pentru un exemplu individual al setului de date
- $\gamma$ reprezintă vectorul cu ponderile funcției estimator $f$

Funcția de cost $\mathcal{L}$ poate să difere de la algoritm la algoritm, ea având rolul să furnizeze o distanță cât mai favorabilă între predicția funcției $f$ și valoarea de ground truth $y_i$. Detaliind modul în care se calculează costul dintre predicția clasificatorului și eticheta pentru exemplul $i$, putem enumera costul pătratic care este folosit intensiv în antrenarea clasificatoarelor bazate pe regresie liniară, polinomială, și rețele neurale:

$$\mathcal{L}(f(x_i|\gamma), y_i) = (f(x_i|\gamma) - y_i)^2 \tag{4.2}$$

costul absolut, folosit atunci când se dorește ușurarea efortului de calcul, pune probleme din cauza faptului că nu e diferențiabil în origine:

$$\mathcal{L}(f(x_i|\gamma), y_i) = |f(x_i|\gamma) - y_i| \tag{4.3}$$

și costul pivotant *hinge loss*, folosit pentru clasificatoare de margine maximă de separație:

$$\mathcal{L}(f(x_i|\gamma), y_i) = max(0, 1 - f(x_i|\gamma) * y_i) \tag{4.4}$$

Din punctul de vedere al clasei $y_i$ putem să discernem între:

- clasificare unde se dorește maparea lui $x_i$ într-un spațiu discret de clase
- regresie unde se dorește maparea lui $x_i$ într-un spațiu continuu, problemă asemănătoare cu interpolarea datelor sau cu calcularea unui scor

### 4.1.2. Învățarea nesupervizată

Învățarea nesupervizată reprezintă categoria algoritmilor de *machine learing* unde se dorește reprezentarea și transformarea datelor într-o modalitate care să confere structură și înțeles unui set de date neetichetat. În contrast cu învățarea supervizată metodele nesupervizate pot învăța să clasifice datele de intrare în *clustere*, sau să găsească dependența între variabilele observate și anumite variabile latente - metodă folosită intensiv în analiza de limbaj natural pentru detecția temei unui text.

În cadrul acestui tip de învățare ne interesează cu precădere algoritmii de reducere a dimensionalității setului de date, algoritmul *T distributed stochatical neighbours embedding* și analiza componentelor principale, bazată pe algoritmul de descompunere în valori singulare.

#### T-SNE

Reprezintă un algoritm important în zona de analiză a datelor și este folosit pentru a aduce vectori de dimensiuni foarte mari la dimensiuni unde pot fi reprezentați grafic (2D și 3D). Având la dispoziție un set de $N$ vectori $\mathbf{x}_i \in \mathbb{R}^n$, mai întâi se calculează probabilitățile ca doi vectori diferiți să fie proporționali, algoritmul se bazează pe convertirea distanței euclidiene în într-o probabilitatea ca cei doi vectori să fie similari. Așa cum a spus și autorul lucrării [17] "Similaritatea dintre $x_i$ și $x_j$ este probabilitatea condiționată $p_{i|j}$ ca $x_i$ să aleagă pe $x_j$ ca vecin considerând că vecinii sunt aleși în conformitate cu distribuția de probabilitate gausiană centrată în $x_i$:

$$p_{i|j} = \frac{e^{-\frac{||x_j - x_i||^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{||x_k - x_i||^2}{2\sigma_i^2}}} \tag{4.5}$$

Următorul pas este ca algoritmul să găsească un nou set de vectori $y_i$ de dimensiune $d$ cu $d < n$, astfel încât similaritățile $p_{j|i}$ dintre vectorii $n$ dimensionali să fie cât se poate mai apropiate de similaritățile $q_{j|i}$ dintre vectorii $d$ dimensionali.

$$q_{i|j} = \frac{-e^{||y_i - y_j||^2}}{\sum_{k \neq i} e^{-||y_i - y_k||^2}} \tag{4.6}$$

Vectorii $\mathbf{y} \in \mathbb{R}^d$ sunt învățați prin minimizarea variantei simetrice a divergenței Kullback-Leibler:

$$\mathbf{KL}(P,Q) = \sum_i \sum_j p_{ij} ln \frac{p_{ij}}{q_{ij}} \tag{4.7}$$

unde

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \tag{4.8}$$

și

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1})} \tag{4.9}$$

## 4.2. Mașini cu vectori suport

Algoritmii cu vector suport, *Support Vector Machines SVM*, reprezintă o clasă de modele supervizate folosite cu succes atât pentru regresie cât și pentru clasificare. Principiul de bază pe care se bazează un SVM este ca plecând de la un set de exemple fiecare aparținând unei clase, algoritmul va găsi hiperplanul optim de separație între cele două exemple. Prin hiperplan optim de separație se înțelege acel hiperplan care asigură marginea cea mai largă de separare [18].

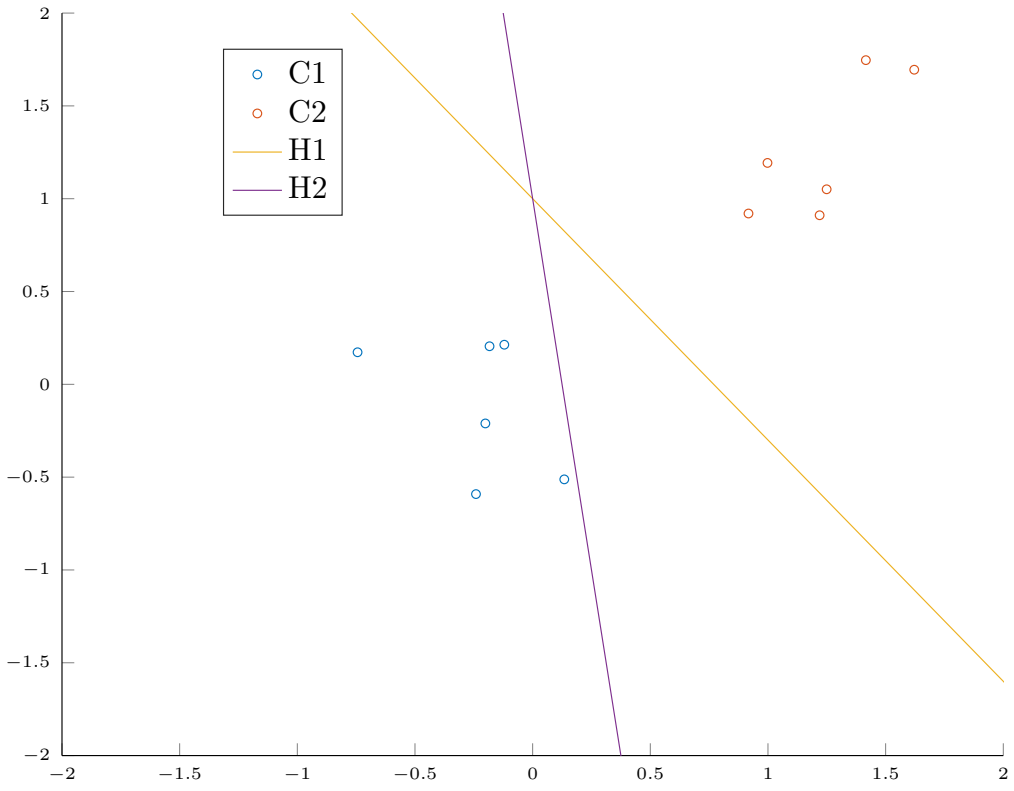



Figura 4.1.: Exemplu de hiperplane de decizie

Așa cum se poate observa în figura de mai sus setul de date este format din vectori bidimensionali structurați în două clase C1 și C2. H1 reprezintă hiperplanul optim de separație între cele două clase deoarece oferă cel mai mult spațiu între vectorii de suport iar H2 reprezintă un hiperplan de separație care neoptim din cauza aproprierii prea mari dintre el și exemplele de la clasa C1.

După cum se poate observa problema găsirii hiperplanului de separație constrânge ca datele ce se doresc a fi clasificate trebuie să fie liniar separabile [18]. În schimb există de asemenea modalități de a putea reprezenta în mod eficient și seturi de date cu neliniarități puternice, anume folosirea unui kernel i.e. o funcție care să transforme spațiul datelor de intrare într-un nou spațiu care să confere mai multe informații pentru clasificare. În abordarea lucrării de față vom folosi algoritmul SVM liniar.

### 4.2.1. Problema de optimizare SVM

Având un set de date $S = (\mathbf{x}_i, y_i)$ unde $y_i \in \{-1, 1\}$ reprezintă clasa în care se află vectorul $mathbfx_i \in \mathbb{R}^n$, se dorește să se găsească hiperplanul de separație de "margine maximă":

$$w \cdot \mathbf{x} - b = 0 \tag{4.10}$$

unde $w$ reprezintă vectorul normal care definește hiperplanul de separație iar $b$ reprezintă coeficientul de compensare

Clasificarea cu SVM se realizează analizând semnul rezultatului $p = \mathbf{w} \cdot \mathbf{x} - b$

- dacă $p < 0$ atunci exemplul $x$ aparține clasei -1
- dacă $p \geq 0$ atunci exemplul $x$ aparține clasei 1

Pentru seturi de date care nu sunt liniar separabile se folosește funcția de cost (4.4), scopul ei fiind să asigure distanța cât mai mare de separație între cele două clase.

Astfel problema de optimizare pentru SVM este dată de minimizarea costurilor pentru fiecare exemplu dat:

$$\min_w \frac{1}{n} \sum_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i - b)) + C||w||^2 \tag{4.11}$$

Introducând variabila $\zeta = max(0, 1 - y_i(w \cdot x_i - b))$ putem rescrie problema de optimizare:

$$\min_w \quad \frac{1}{n} \sum_{i=1}^{n} \zeta_i + C||w||^2 \tag{4.12a}$$

$$\text{s.l.: } y_i(w \cdot x_i - b) \geq 1 - \zeta_i \tag{4.12b}$$

Avantajul este că pentru rezolvarea acestei probleme există solvere de programare pătratică foarte performante, care întrebuințează diferite metode ale metodei gradientului.

### 4.2.2. Motivația alegerii SVM

Unul dintre motivele pentru care am ales acest algoritm a fost faptul că există numeroase pachete de Python care implementează atât solvere eficiente pentru problema de optimizare cât și o bibliotecă bogată și extrem de bine documentată pentru extrem de multe metode de preprocesare și algoritmi de învățare automată, precum SciKit-Learn [19] [20].

De asemenea fiind un algoritm liniar simplu există șanse foarte mici să apară fenomenul de supra estimare *overfitting* prin care modelul ales nu numai că învață trendul setului de date, dar și zgomotul de care acesta este afectat. O exemplificare a acestui fenomen este dată în figura următoare unde se compară un model simplu de gradul 1 cu un model mai complex de gradul 15:

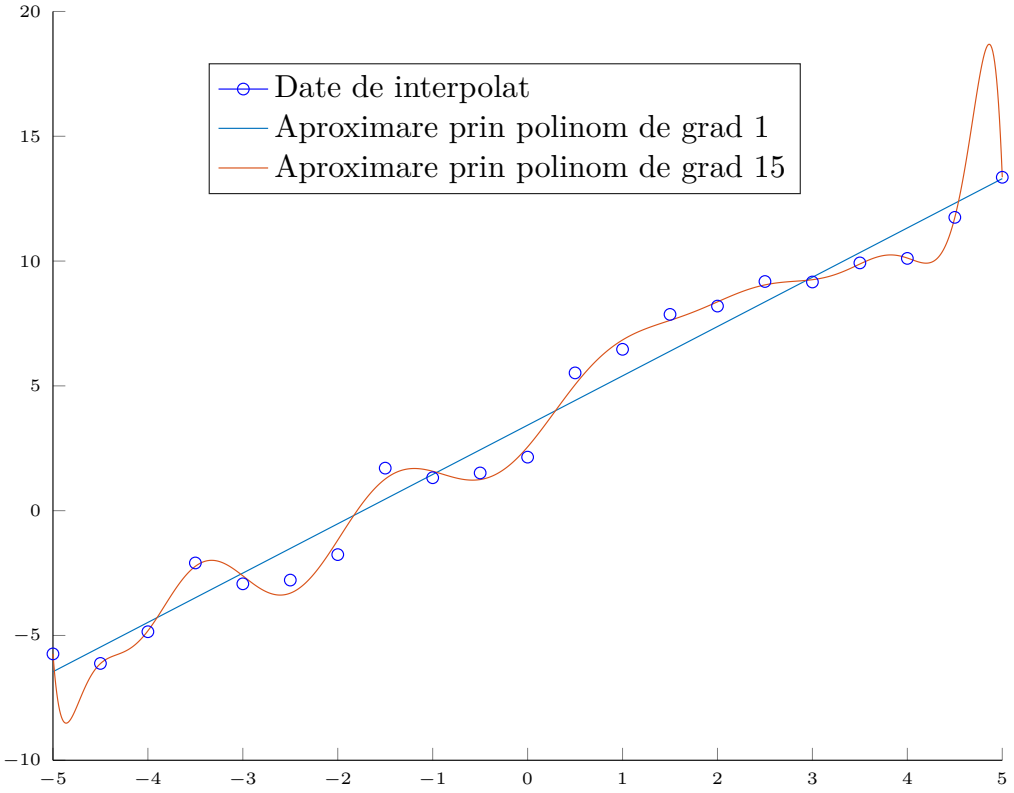



Figura 4.2.: Aproximări prin regresie liniară

## 4.3. Definirea performanțelor

Pentru un set de date $S = \{(x_i, y_i) | i = \overline{1,n}\}$ putem să evaluăm performanțele clasificatorului comparând rezultatele adevărate $y_i$ cu predicția acestuia $y_{pi}$. Ca limbaj vom considera că un exemplu selectat de clasificator va reprezenta un nod în care este prezis un defect. Astfel putem să definim următoarele:

- adevărat pozitiv (TP) $y_i = y_{pi} = 1$
- adevărat negativ (TN) $y_i = y_{pi} = -1$

- fals pozitiv (FP) $y_i = -1$ și $y_{pi} = 1$
- fals negativ (FN) $y_i = 1$ și $y_{pi} = -1$

Iar pe baza acestora putem defini metrici precum:

Acuratețea reprezintă raportul de obiecte prezise corect din totalul de exemple:

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.13}$$

Precizia, arată câte din rezultatele selectate sunt relevante:

$$P = \frac{TP}{TP + FP} \tag{4.14}$$

Sensibilitatea, denumită și rata de retragere, raportul dintre nodurile selectate și totalul nodurilor în care există un defect:

$$S = \frac{TP}{TP + FN} \tag{4.15}$$

## 4.4. Rezultate preliminare folosind toți senzorii

Considerând acum setul de date pentru un algoritm liniar SVM vectorii $x_i$ vor fi reprezentați de mediile reziduurilor pe intervalul staționar, deci $x_i \in \mathbb{R}^{31}$, iar $y_i = \overline{1..31}$ reprezintă clasa în care poate fi situat un defect.

Setul de date a fost obținut prin considerarea a două mulțimi de defecte [3]:

- pentru setul de antrenare avem valori impare pentru emițător între 1 și 31
- pentru setul de testare avem valori pare pentru emițător între 1 și 31

Astfel rulând câte o simulare pentru fiecare nod și valoare a emițătorului ajungel a un set de antrenare $\mathbf{X}_{train} \in \mathbb{R}^{496\times31}$ și un set de testare $\mathbf{X}_{test} \in \mathbb{R}^{465\times31}$ pentru fiecare dintre acestea vând seturile de clase $y_{train} \in \mathbb{N}^{496}$ și $y_{test} \in \mathbb{N}^{465}$

Partea de antrenare a modelului se realizează prin folosirea unui obiect de tip SVC *Support Vector Classifier* din toolkit-ul SciKit-Learn. Având ca parametru de regularizare C = 10 și kernelul liniar am obținut următoarele rezultate pe setul de testare:

| Acuratețe | Precizie | Sensibilitate |
|---|---|---|
| 0.94 | 0.98 | 0.94 |

Tabela 4.1.: Metrici medii pe setul de testare plin

## 4.5. Selecția de senzori folosind Eliminarea recursivă de caracteristici

Amintind de metoda de selecție a senzorilor de la secțiunea 3.6 putem să folosim mașinile cu vectori suport pentru a ordona nodurile în ordinea importanțelor la procesul de clasificare. Aici descriem algoritmul de eliminare recursivă a caracteristicilor *Recursive Feature Ellimination*. Procesul se bazează pe faptul că cele mai importante caracteristici din vectorii de intrare ale SVM-ului au ponderile cele mai mari. Astfel pentru un clasificator binar format din perechea $(w, b)$ cea mai puțin importantă caracteristică este dată de relația [21]:

$$arg\min_{i} \; w \tag{4.16}$$

Pentru a putea clasifica cele 31 de defecte, vom avea nevoie de 31 de clasificatoare de tipul *One vs All*, iar ponderile pentru o astfel de problemă vor fi o matrice $mathbf\, W \in \mathbb{R}^{n_c-1,n_j}$, unde $n_c$ reprezintă numărul de defecte iar $n_j$ reprezintă numărul de noduri folosiți pentru clasificare.

Pentru a afla scorul fiecărui nod în procesul de clasificare vom suma absolut matricea **W** pe coloane, deci:

$$scor(k) = \sum_{i=1}^{n_c} |W_{ik}| \tag{4.17}$$

### Algoritmul RFE

Algoritmul de eliminare recursivă a caracteristicilor este descris în continuare:

**Algoritm 4.1:** Eliminarea recursivă a caracteristicilor

**Input:** $S_{train} = \{(x_i, y_i) | i = \overline{1, n}\}$

$C \leftarrow$ mulțimea curentă de caracteristici;
$N_{cp} \leftarrow$ număr de caracteristici de păstrat;
**while** $|C| > N_{cp}$ **do**
    antrenează clasificatorul cu caracteristicile din $C$;
    calculează scorurile cu formula (4.17);
    elimină din mulțimea $C$ caracteristica cu scorul cel mai mic
**end**

La finalul acestui proces mulțimea $C$ va conține $N_{cp}$ noduri care au cea mai puternică contribuție la procesul de clasificare.

## 4.6. Rezultate folosind senzorii selectați

Pentru a testa metoda de selecție a senzorilor am fixat un spectru pentru parametrul $N_{cp} = \{1, 4, 6, 10\}$

| Nr senzori selectați | Senzori selectați | Acuratețe |
|---|---|---|
| 1 | 11 | 64.4% |
| 4 | 10, 11, 25, 27 | 89.7% |
| 6 | 10, 11, 16, 25, 27, 28 | 93.8% |
| 10 | 9, 10, 11, 12, 16, 20, 25, 27, 28, 29 | 94.1% |

Tabela 4.2.: Performanțele clasificării SVM cu senzorii selectați de RFE

În figura următoare este reprezentată performanța fiecărui nod în procesul de clasificare:

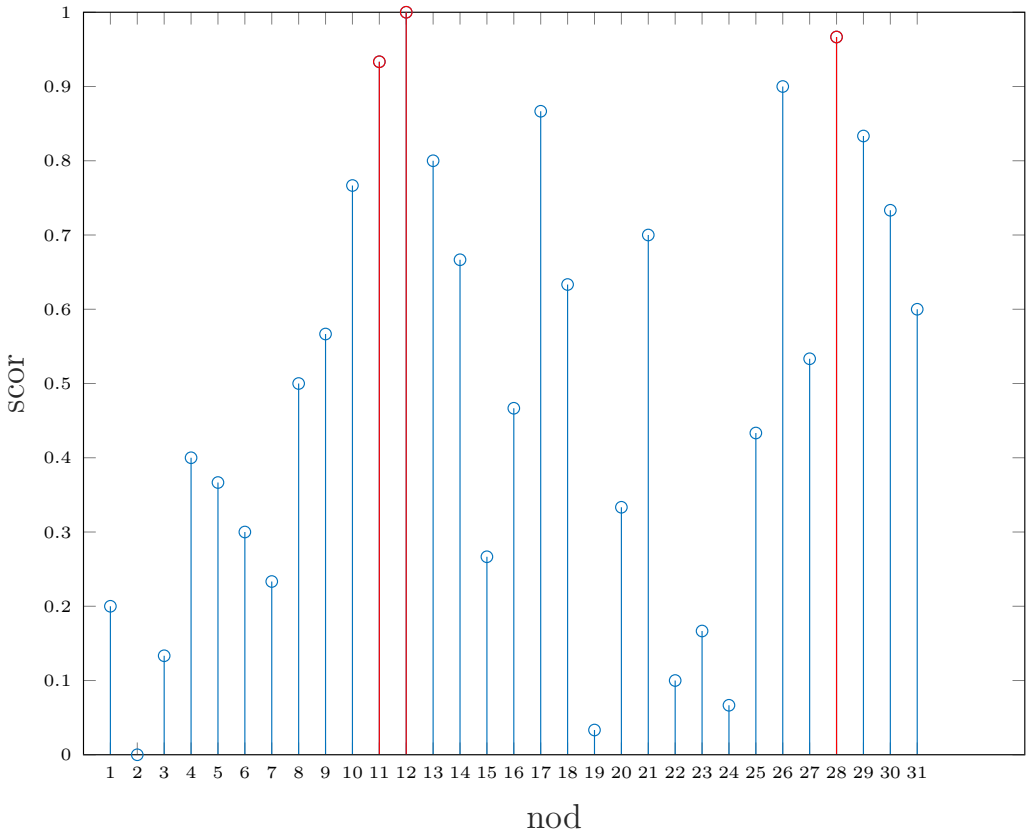



Figura 4.3.: Scorul RFE

După cum se vede în figura 4.3 și în tabelul 4.2 cei mai importanți senzori sunt cei reprezentați cu roșu și aduc cea mai puternică contribuție la performanțele procesului de clasificare.

### Comparație între MSC și RFE

Având în vedere că algoritmul RFE poate fi parametrizat pentru a putea selecta un anumit număr de senzori $N_{cp}$ problema MSC nu impune o constrângere explicită asupra numărului de senzori selectați, pentru a putea modifica acest număr este nevoie să considerăm modificarea limitei de sensibilitate a senzorilor pentru calcularea matricei $\mathbf{M}$ (3.17)

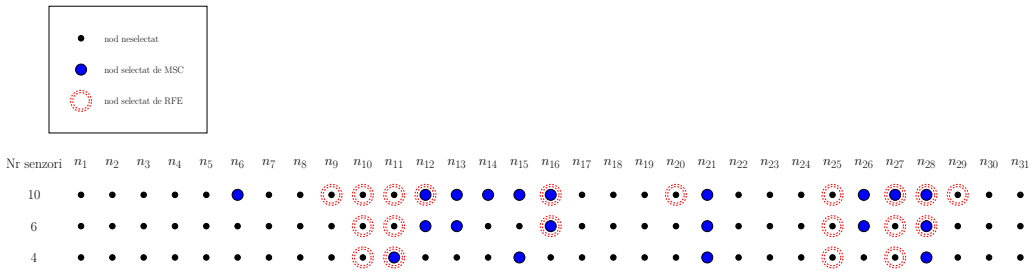



Figura 4.4.: Comparație între senzorii aleși de RFE și MSC

Figura 4.4 prezintă similaritățile între senzorii aleși de cele două metode, după cum se poate observa pentru cazul în care se doresc 4 senzori, atât MSC cât și RFE aleg senzorul 11. O particulartiate a MSC este că nu asignează o importanță a nodurilor ci doar alege acele noduri care asigură îndeplinirea condițiilor impuse de problema de minimizare. De asemenea metoda eliminării de caracteristici oferă o consistență în caracteristicile alese - pe măsură ce parametrul $N_{cp}$ crește, mulțimea de senzori selectați nu pierde elemente ci este doar completată.

| Nr senzori selectați | Senzori selectați | Acuratețe |
|---|---|---|
| 4 | $11, 15, 21, 28$ | 92.1% |
| 6 | $12, 13, 16, 21, 26, 28$ | 93.3% |
| 10 | $6, 12, 13, 14, 15, 16, 21, 26, 27, 28$ | 94% |

Tabela 4.3.: Performanțele clasificării SVM cu senzorii selectați de MSC

# 5. Clasificarea defectelor folosind metoda învățării de dicționare rare

## 5.1. Aspecte teoretice

Problema învățării dicționarelor se clasează în domeniul problemelor de învățare nesupervizată - setul de date nu trebuie să fie structurat pe clase - dar există și variațiuni ale acesteia prin care modifică criteriul de optimizat astfel încât să se poată învăța dicționare specializate pentru clasificare.

Având ca date de intrare un set de date $\mathbf{Y} \in \mathbb{R}^{n \times n_s}$, unde $n$ reprezintă dimensiunea semnalelor și $n_s$ reprezintă numărul de semnale folosite la antrenare, dorim să găsim acele matrice $\mathbf{D} \in \mathbb{R}^{n \times m}$ și $\mathbf{X} \in \mathbb{R}^{m \times n_s}$ astfel încât să rezolvăm problema de optimizare[22, Capitol 2]:

$$\min_{\mathbf{D},\mathbf{R}} \quad ||\mathbf{Y} - \mathbf{D}\mathbf{X}||_F^2 \tag{5.1a}$$

$$\text{s.l.:} \quad ||x_l||_0 \leq s, l = 1 : N \tag{5.1b}$$

$$||d_j|| = 1, j = 1 : n \tag{5.1c}$$

Unde:

- $\mathbf{D}$ reprezintă matricea dicționarului pe baza căruia se va calcula reprezentarea, coloanele acesteia se numesc atomi
- $\mathbf{X}$ este reprezentarea rară a setului de date $\mathbf{Y}$

Constrângerea (5.1b) se referă la raritatea vectorului de reprezentare iar (5.1c) la normalizarea atomilor pentru dicționar.

Problema de găsire a dicționarului și a reprezentării semnalelor de antrenare $\mathbf{Y}$ conține neliniarități puternice din cauza condiției de sparsitate impuse. Cazul interesant și cel mai abordat în literatură îl reprezintă acela în care dicționarul este supracomplet [22, Capitolul 1], această proprietate poate aduce numeroase beneficii diferitelor procese de clasificare, anume:

- stocarea matricelor sparse se face mult mai eficient decât cele pline
- din punct de vedere computațional există foarte multe multiplicări care nu se vor mai efectua

Modul în care am descris problema duce cu gândul la o metodă sofisticată de extragere a caracteristicilor din setul de date. Astfel dacă extragem o reprezentare sparsă $X = \{x_i\}$, putem să folosim vectorul rar $x_i$ ca exemplu de antrenare pentru alți algoritmi de antrenare sau clasificare.

### 5.1.1. Găsirea reprezentării sparse

Reprezentarea sparsă se ocupă de reconstruirea unui semnal $y \in \mathbf{R}^n$ având la dispoziție un dicționar $\mathbf{D} \in \mathbb{R}^{n \times m}, \mathbf{D} = [d_1, d_2, ..., d_m]$ cu $m > n$, adică, găsirea unui vector $\mathbf{x} \in \mathbb{R}^m$ astfel încât $y \approx \mathbf{Dx}$. Reziduul aproximării sparse este definit ca:

$$e = y - \mathbf{Dx} \tag{5.2}$$

Un algoritm greedy care rezolvă această problemă este OMP *Orthogonal Mathching Pursuit*. Având la un moment dat mulțimea $\mathcal{S}$ a atomilor din dicționar selectați, algoritmul dorește să atingă un anumit nivel de sparsitate $s = |\mathcal{S}|$ și îndeplinirea unui criteriu de eroare $||y - \mathbf{D}_{\mathcal{S}}\mathbf{x}|| < tol$ [22, Capitolul 1]. Algoritmul va completa mulțimea $\mathcal{S}$ cu acel atom $d_k$ care nu se află în ea și care va fi cel mai bine corelat cu reziduul actual, deci $k = arg\max_{j \notin \mathcal{S}} |e^T x_j|$.

Pentru a putea obține reprezentarea semnalului $y$ folosind mulțimea de atomi $\mathcal{S}$ este nevoie să rezolvăm sistemul de ecuații supradeterminat:

$$\mathbf{Dx} = \mathbf{y} \tag{5.3}$$

Prin folosirea soluției ecuațiilor normale:

$$\mathbf{x} = (\mathbf{D}\mathbf{D}^T)^{-1}\mathbf{D}\mathbf{y} \tag{5.4}$$

## 5.2. Adaptarea la problema rețelelor de apă

Pentru a putea clasifica nodurile unde s-au produs defecte trebuie extinsă problema de optimizare (5.1) astfel încât să se ia în calcul o structură eficientă a dicționarului pentru reprezentarea sparsă a reziduurilor, dar și apariția unei noi matrici $\mathbf{W}$ numită clasificator [22]:

$$\min_{\mathbf{W}} ||\mathbf{H} - \mathbf{W}\mathbf{X}||_F^2 + \gamma||\mathbf{W}||_F^2 \tag{5.5}$$

matricea $\mathbf{H}$ reprezintă colecția de etichete pentru fiecare dintre reziduurile din $\mathbf{Y}$ forma lui fiind:

$$\mathbf{H}_i = e_k, \text{dacă pentru exemplul i s-a produs un defect în nodul k} \tag{5.6}$$

parametrul $\gamma$ are rolul de a condiționa mai bine matricea și a evita fenomenul de *overfitting*. Matricea $\mathbf{W}$ reprezintă o matrice de transformare care în mod ideal încearcă să transforme spațiul rar $\mathbf{X}$ într-un spațiu categoric de tipul versorilor $e_k \in \mathbb{R}^{n_{clase}}$. Real vorbind, ceea ce se întâmplă este că rezultatul înmulțirii $\mathbf{WX}$ este un vector dens, unde indexul celui mai mare element denotă de fapt clasa din care face parte reziduul transformat $p_{dl} = argmax(\mathbf{Wx})$ [22, Capitolul 8]. Agregând cele două probleme de optimizare obținem învățarea de dicționare discriminative:

$$\min_{\mathbf{D},\mathbf{X},\mathbf{W}} ||\mathbf{Y} - \mathbf{DX}||_F^2 + \alpha||\mathbf{H} - \mathbf{WX}||_F^2 \tag{5.7}$$

O altă modalitate e a spori performanțele de clasificare este prin obligarea problemei de optimizare să aloce anumiți atomi ai dicționarului $\mathbf{D}$ unor anumite clase de defecte - învățarea de dicționare cu consistență de clasă *label consistent dictionary learning* [22]. Problema de optimizare la care se reduce este:

$$\min_{\mathbf{D},\mathbf{W},\mathbf{A},\mathbf{X}} ||\mathbf{Y} - \mathbf{DX}||_F^2 + \alpha||\mathbf{H} - \mathbf{WX}||_F^2 + \beta||\mathbf{Q} - \mathbf{AX}||_F^2 \tag{5.8}$$

matricea $\mathbf{Q}$ fiind de fapt parametrul care alocă fiecare atom unui anumit defect și contribuie la sporirea calității clasificării. Liniile matricei $\mathbf{Q}$ sunt asociate fiecărui atom cu proprietatea

$$q_{ij} = \begin{cases} 1, & \text{atomul i este asociat clasei j} \\ 0, & \text{în rest} \end{cases} \tag{5.9}$$

Problema (5.10) poate fi redusă la problema de optimizare [22]:

$$\min_{\mathbf{D},\mathbf{W},\mathbf{A},\mathbf{X}} \left\| \begin{bmatrix} \mathbf{Y} \\ \sqrt{\alpha}\mathbf{H} \\ \sqrt{\beta}\mathbf{Q} \end{bmatrix} - \begin{bmatrix} \mathbf{D} \\ \sqrt{\alpha}\mathbf{W} \\ \sqrt{\beta}\mathbf{A} \end{bmatrix} \mathbf{X} \right\|_F^2 \tag{5.10}$$

și poate fi rezolvată cu ajutorul algoritmului K-DVS care funcționează prin alternarea fixării matricelor $\mathbf{D}$ și $\mathbf{X}$, prima dată dicționarul este fixat și se găsește cea mai bună reprezentare $\mathbf{X}$, apoi se recalculează dicționarul $\mathbf{D}$. Metoda se bazează pe aproximarea unei matrice de rang n printr-o sumă de matrice de rang 1.

## 5.3. Rezultate și metrici de clasificare

Pentru testarea metodelor de DL am folosit setul de date de la capitolul 4, iar metodele care calculează dicționarele și reprezentările sparse sunt cele de la [23]. Asemănător capitolului de clasificare cu SVM voi considera nodurile alese prin metoda MSC și prin metoda RFE, și voi arăta pentru fiecare dintre acestea acuratețea obținută cu ajutorul metodelor de DL cu consistență de clasă și DL discriminativ. Astfel avem pentru senzorii selectați de algoritmul RFE: Iar pentru metoda de plasare a senzorilor cu problema MSC

| Nr senzori | Senzori selectați | Acuratețe DL discriminativ | Acuratețe LC-DL |
|---|---|---|---|
| 4 | $10, 11, 25, 27$ | 46.5% | 82.4% |
| 6 | $10, 11, 16, 25, 27, 28$ | 53.1% | 86.2% |
| 10 | $9, 10, 11, 12, 16, 20, 25, 27, 28, 29$ | 57.2% | 93.3% |

Tabela 5.1.: Performanțele clasificării DL cu senzorii selectați de RFE

am obținut rezultatele:

| Nr senzori | Senzori selectați | Acuratețe DL discriminativ | Acuratețe LC-DL |
|---|---|---|---|
| 4 | $11, 15, 21, 28$ | 58.7% | 89.7% |
| 6 | $12, 13, 16, 21, 25, 26$ | 51.4% | 86.9% |
| 10 | $6, 12, 13, 14, 15, 16, 21, 26, 27, 28$ | 65.59% | 96.7% |

Tabela 5.2.: Performanțele clasificării DL cu senzorii selectați de MSC

Luând în calcul cei 10 senzori selectați de metoda MSC clasificatorul obținut de metoda învățarii de dicționare cu consistență de clase dă un rezultat al acurateții de 96.7%, cu 2.3 % mai performant decât abordarea cu SVM. Astfel putem afirma că învățare dicționarelor rare poate fi aplicată cu succes în problema clasificării defectelor într-o rețea cu apă și în principiu oferă o așa zisă robustețe, deoarece atomii asignați unei anumite clase vor reprezenta într-un mod prost profilurile defectelor din altă clasă.

# 6. Concluzii și direcții viitoare

În această lucrare am prezentat problematica identificării defectelor într-o rețea de apă, prin analiza profilurilor reziduurilor într-o perioadă de timp corespunzătoare regimului staționar. Abordarea s-a axat pe alegerea unei metode empirice și statistice în detrimentul unei metode analitice, așa cum a fost prezentat și în capitolul 2, o interpretare exactă a rețelelor de apă este zadarnică, procesul care stă în spatele dependenței presiune - debit - viteză în conducte fiind unul puternic neliniar și influențat de mulți factori exogeni.

Abordarea propusă în această lucrare este una bine documentată [3], [10], [6] asupra căreia am aplicat două metode de învățare automată supervizată - mașini cu vectori suport și învățarea de dicționare rare. De asemenea o preocupare a acestei lucrări a fost centrată asupra optimizării procesului de clasificare, anume reducerea numărului de senzori prezenți la clasificare. Motivația reducerii numărului de senzori este ajungerea la un compromis ingineresc între investiția în senzorii plasați în nodurile rețelei de apă și cheltuielile pentru combaterea defectelor. Rezultatul reducerii numărului de senzori prin metoda 4.1 și prin rezolvarea problemei NP-Complete 4.12, a arătat prezența redundanței puternice între informațiile provenite de la senzori, în mod natural, deoarece nodurile vecine într-un graf sunt supuse unor solicitări foarte apropiate.

Rezultatele clasificării folosind metodele LC-DL și SVM au dat rezultate apreciabile și relevante din punct de vedere statistic, antrenarea și testarea fiind făcută pe două seturi de date echilibrate (i.e. același număr de obiecte pentru fiecare clasă) și variate - fiecare defect pentru care se încerca clasificarea avea ca exemple de clasă profiluri cu mai multe magnitudini - metricile de acuratețe atingând un procent foarte ridicat - 96.7% pentru LC-DL folosind 10 senzori aleși cu MSC și 94.1% pentru SVM folosind 10 senzori aleși cu RFE. De asemenea algoritmul SVM poate fi folosit într-un alt algoritm 4.1 pentru a putea alege nodurile din rețea cu cea mai importantă contribuție.

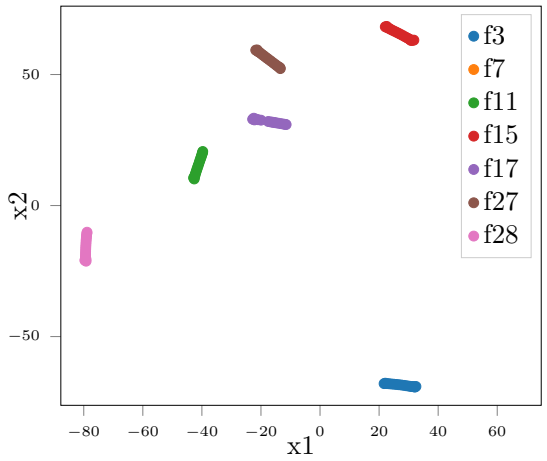



Figura 6.1.: Proiecția în 2D a profilurilor defectelor

Ca ilustrație grafică bidimensională asupra profilurilor defectelor am realizat în figura 6.1 proiecția pe 2 axe a vectorului de caracteristici al unui defect format din 31 de caracteristici, folosind algoritmul TSNE. După cum se poate observa aceste defecte nu numai ca sunt liniar separabile, dar în funcție de magnitudinea emițătorului în nodul respectiv proiecțiile $(x1, x2)$ se vor situa pe o dreaptă. Deși folosirea unei metode de clasificarea bazate pe extragerea proiecțiilor în 2D sau 3D prin algoritmul TSNE poate părea o idee atractivă, problema apare în momentul în care se dorește transformarea unui set de date nou - fiind un algoritm stocastic TSNE va da rezultate diferite la fiecare rulare și există riscul ca aceleași defecte să fie coagulate în alte zone ale planului bidimensional, compromițând astfel procesul de clasificare.

## Direcții viitoare

Ca abordare diferită putem să considerăm partiționarea rețelei în mai multe subgrafuri [3] cu scopul de a relaxa problema și de a crește performanțele clasificării. O modalitate de a vedea care sunt nodurile care oferă cea mai mare apropiere în sensul partiționării este folosirea unui algoritm de clusterizare asupra reziduurilor fiecărui defect, de exemplu K-Means. Rularea acestui algoritm cu 4 centroide dă următoarea cofigurație de clustere:

- $C_1 = \{22, 23, 24, 27, 28, 29, 30, 31\}$
- $C_2 = \{1, 2, 3, 16, 17, 18, 19\}$
- $C_3 = \{20, 21\}$
- $C_4 = \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}$
- $C_5 = \{25, 26, 14, 15\}$

Rulând algoritmul de clasificare de la capitolul 4 obținem o acuratețe de 95.2% cu ajutorul a suportului de 4 senzori selectați $C = \{12, 19, 25, 27\}$

# Anexe

# A. Fișiere sursă

# Bibliografie


[1] Rex Klopfenstein Jr. „Air velocity and flow measurement using a Pitot tube“. In: *ISA transactions* 37.4 (1998), pp. 257–263.

[2] Donald F Elger and John A Roberson. *Engineering fluid mechanics*. Wiley Hoboken (NJ), 2016.

[3] Paul Irofti and Florin Stoican. „Dictionary learning strategies for sensor placement and leakage isolation in water networks“. In: *IFAC-PapersOnLine* 50.1 (2017), pp. 1553–1558.

[4] Gerard Sanz Estapé. „Demand modeling for water networks calibration and leak localization“. In: (2016).

[5] Richard M Karp. „On the computational complexity of combinatorial problems“. In: *Networks* 5.1 (1975), pp. 45–68.

[6] Lewis A Rossman et al. „EPANET 2: users manual“. In: (2000).

[7] Assela Pathirana. *EPANET calling API for python.* 2016. URL: https://github.com/asselapathirana/epanettools.

[8] SB Kotsiantis, D Kanellopoulos, and PE Pintelas. „Data preprocessing for supervised leaning“. In: *International Journal of Computer Science* 1.2 (2006), pp. 111–117.

[9] *Garbage in Garbage out.* 2016. URL: https://en.wikipedia.org/wiki/Garbage_in,_garbage_out.

[10] Lina Sela Perelman et al. „Sensor placement for fault location identification in water networks: A minimum test cover approach“. In: *Automatica* 72 (2016), pp. 166–176.

[11] Thomas H. Cormen et al. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009. ISBN: 0262033844, 9780262033848.

[12] J. Löfberg. „YALMIP : A Toolbox for Modeling and Optimization in MATLAB“. In: *In Proceedings of the CACSD Conference*. Taipei, Taiwan, 2004.

[13] Arthur L Samuel. „Some studies in machine learning using the game of checkers“. In: *IBM Journal of research and development* 3.3 (1959), pp. 210–229.

[14] Elizabeth Gibney. „Google AI algorithm masters ancient game of Go“. In: *Nature News* 529.7587 (2016), p. 445.

[15] Yann LeCun et al. „Backpropagation applied to handwritten zip code recognition“. In: *Neural computation* 1.4 (1989), pp. 541–551.

[16] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[17] Laurens van der Maaten and Geoffrey Hinton. „Visualizing data using t-SNE“. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[18] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[19] F. Pedregosa et al. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[20] Lars Buitinck et al. „API design for machine learning software: experiences from the scikit-learn project“. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[21] Vladimir Svetnik et al. „Application of Breiman's Random Forest to Modeling Structure-Activity Relationships of Pharmaceutical Molecules“. In: *Multiple Classifier Systems*. Ed. by Fabio Roli, Josef Kittler, and Terry Windeatt. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 334–343. ISBN: 978-3-540-25966-4.

[22] B. Dumitrescu and P. Irofti. *Dictionary Learning Algorithms and Applications*. Springer, 2018.

[23] Paul Irofti Bogdan Dumitrescu. *Dictionary Learning Book*. 2018. URL: https://github.com/dl-book/dl-book.