A programozás alapjai 2.

Házi Feladat Dokumentáció

**Systems and Signals Solver**

Készítette: Magyar Gábor Balázs, ERYDAG

gabor.4441@gmail.com

Készítés féléve: 2023/24/2

Tartalomjegyzék

[Felhasználói dokumentáció 5](#_Toc167378671)

[Működtetés fájlok beolvasásával 5](#_Toc167378672)

[Manuális működtetés 6](#_Toc167378673)

[Osztályok statikus leírása 8](#_Toc167378674)

[matrix 8](#_Toc167378675)

[Felelőssége 8](#_Toc167378676)

[Ősosztályok 8](#_Toc167378677)

[Attribútumok 8](#_Toc167378678)

[Metódusok 8](#_Toc167378679)

[menu 10](#_Toc167378680)

[Felelőssége 10](#_Toc167378681)

[Ősosztályok 10](#_Toc167378682)

[Attribútumok 10](#_Toc167378683)

[Metódusok 10](#_Toc167378684)

[Verticle 11](#_Toc167378685)

[Felelőssége 11](#_Toc167378686)

[Ősosztályok 11](#_Toc167378687)

[Attribútumok 11](#_Toc167378688)

[Metódusok 11](#_Toc167378689)

[Edge 12](#_Toc167378690)

[Felelőssége 12](#_Toc167378691)

[Ősosztályok 12](#_Toc167378692)

[Attribútumok 12](#_Toc167378693)

[Verticle\* a; 12](#_Toc167378694)

[Verticle\* b; 12](#_Toc167378695)

[double I; Rajta folyó áram 12](#_Toc167378696)

[double U; Rajta lévő feszültség 12](#_Toc167378697)

[Metódusok 12](#_Toc167378698)

[Resistance 13](#_Toc167378699)

[Felelőssége 13](#_Toc167378700)

[Ősosztályok 13](#_Toc167378701)

[Attribútumok 13](#_Toc167378702)

[double R; Ellenállás 13](#_Toc167378703)

[static string type; 13](#_Toc167378704)

[Metódusok 13](#_Toc167378705)

[V\_src 14](#_Toc167378706)

[Felelőssége 14](#_Toc167378707)

[Ősosztályok 14](#_Toc167378708)

[Attribútumok 14](#_Toc167378709)

[double voltage; 14](#_Toc167378710)

[static string type; 14](#_Toc167378711)

[Metódusok 14](#_Toc167378712)

[I\_src 14](#_Toc167378713)

[Felelőssége 14](#_Toc167378714)

[Ősosztályok 14](#_Toc167378715)

[Attribútumok 14](#_Toc167378716)

[double current; 14](#_Toc167378717)

[static string type; 14](#_Toc167378718)

[Metódusok 14](#_Toc167378719)

[TwoPort 15](#_Toc167378720)

[Felelőssége 15](#_Toc167378721)

[Ősosztályok 15](#_Toc167378722)

[Attribútumok 15](#_Toc167378723)

[Edge\* Connected\_to; 15](#_Toc167378724)

[Metódusok 15](#_Toc167378725)

[Transformer 15](#_Toc167378726)

[Felelőssége 15](#_Toc167378727)

[Ősosztályok 15](#_Toc167378728)

[Attribútumok 15](#_Toc167378729)

[double n; Transzformátorra jellemző adat 15](#_Toc167378730)

[static string type; 15](#_Toc167378731)

[Metódusok 15](#_Toc167378732)

[Gyrator 16](#_Toc167378733)

[Felelőssége 16](#_Toc167378734)

[Ősosztályok 16](#_Toc167378735)

[Attribútumok 16](#_Toc167378736)

[double r; Girátorra jellemző adat 16](#_Toc167378737)

[static string type; 16](#_Toc167378738)

[Metódusok 16](#_Toc167378739)

[UML osztálydiagramm 17](#_Toc167378740)

[17](#_Toc167378741)

[Összegzés 18](#_Toc167378742)

[Mit sikerült és mit nem sikerült megvalósítani a specifikációból? 18](#_Toc167378743)

[Mit tanultál a megvalósítás során? 18](#_Toc167378744)

[Továbbfejlesztési lehetőségek 18](#_Toc167378745)

[Képernyőképek a futó alkalmazásról 19](#_Toc167378746)

# Felhasználói dokumentáció

A program egy menüben kezdődik, ahol kiválaszthatjuk, hogy fájlból, vagy manuálisan kívánjuk megadni a bemenetet.  
A menüben számokat kell megadni, kivéve, ha a program fájlnevet kér, ekkor „szöveg.txt” formájú bemenetet vár, a .txt is kötelező.

### Működtetés fájlok beolvasásával

* Kezelői oldalról válasszuk az 1-es módot a menüben, majd adjuk meg a fájlok nevét.
* programozói oldalról lehetséges úgy megnyitni a menüt, hogy az alapértelmezetten egy fájlt olvasson be, majd írjon ki valahová. Ehhez egy olyan menüt lehet létrehozni, ami nem jelenik meg a standard kimeneten, és úgy kezeli a fájlunkat. A main-ben erre példa a file\_in menü.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírásA képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírásA képen szöveg, képernyőkép, szám, dokumentum látható

Automatikusan generált leírásA bemenetre található minta a csatolt fájlok közt a programnál.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

### Manuális működtetés

3 funkciót használhatunk:

* Mátrix számológép
* Eredő ellenállás
* Lineáris Nem Dinamikus hálózat megoldó (LINNDIN)

Ezeknek igencsak különböző bemenetei vannak

|  |  |  |  |
| --- | --- | --- | --- |
| Mátrix | Ellenállás | LINNDIN | |
| Bemenet: | | | |
| 2  1  2 2 1 2 3 4  2 2 419 67 0.1415 2  +  E | 2  2  2  1  0  7  -1  2  0 1 | 2 3 3  0 1 4 5  1 0 3 3  2 0  0  2  1  7  -1 | 1  2  0 1  2  2 3  3 |
| Kimenet | | | |
| Welcome to Systems and Signals Solver!  Choose from the following modes:  0. Exit  1. Read from File  2. Interactive Problem Solver  2  Choose from the following functions:  0. Exit  1. Matrix Calculator  2. Resistence Calculator  3. Lin-NDin Solver  1  Warning! The following function is only able to work with 3 matrices  use T for transpose G for Gauss elimination, D for determinant  and i for inverse: (only affecting Matrix2)  Press E to return, any to continue  a  Matrix3: f(Matrix1,matrix2)  Matrix1: (n), (m), (data)  2 2 1 2 3 4  row: 2  column: 2  1, 3,  2, 4,  Matrix2: (n), (m), (data)  2 2 419 67  0.1415 2  row: 2  column: 2  419, 0.1415,  67, 2,  Now what should we do with them? Remember:  use T for transpose and i for inverse(only affecting Matrix2)  +  Solution:  row: 2  column: 2  420, 3.1415,  69, 6,  E | Welcome to Systems and Signals Solver!  Choose from the following modes:  0. Exit  1. Read from File  2. Interactive Problem Solver  2  Choose from the following functions:  0. Exit  1. Matrix Calculator  2. Resistence Calculator  3. Lin-NDin Solver  2  Number of Verticles:  2  Now you can add the dipoles(edges), note, to first give the verticle where the  voltage points from to another  Index of the first verticle(from):  (-1 to exit)  1  Index of the second verticle(to):  0  Resistence value:  7  Index of the first verticle(from):  (-1 to exit)  -1  n: 2  b: 2  l: 1  Give the index of edges to make 1 loops  Loop 1. :  How many branches?  2  Indexes of the branches:  0 1  The Resultant Resistence: 7 | Welcome to Systems and Signals Solver!  Choose from the following modes:  0. Exit  1. Read from File  2. Interactive Problem Solver  2 3 3  Choose from the following functions:  0. Exit  1. Matrix Calculator  2. Resistence Calculator  3. Lin-NDin Solver  Number of Verticles:  Now you can add the dipoles(edges), note, to first give the verticle where the  voltage points from to another  Index of the first verticle(from):  (-1 to exit)  0 1 4 5  Index of the second verticle(to):  Cathegory:  Resistance:1  Transformer:2  Gyrator:3  Voltage Source:4  Current Source:5  Voltage value:  Index of the first verticle(from):  (-1 to exit)  1 0 3 3  Index of the second verticle(to):  Cathegory:  Resistance:1  Transformer:2  Gyrator:3  Voltage Source:4  Current Source:5  Gyrator resistence:  The index of the first verticle of the other Gyrator:  2 0  Index of the second verticle of the other Gyrator:  Index of the first verticle(from):  (-1 to exit)  0  Index of the second verticle(to):  2  Cathegory:  Resistance:1  Transformer:2  Gyrator:3  Voltage Source:4  Current Source:5  1  Resistence value:  7  Index of the first verticle(from):  (-1 to exit)  -1  Choose from the following answers:  0.Current 1. Voltage  1  n: 3  b: 4  l: 2  Give the index of edges to make 2 loops  Loop 1. :  How many branches?  2  Indexes of the branches:  0 1  Loop 2. :  How many branches?  2  Indexes of the branches:  2 3  The index of the edge, with the voltage in question:  3  The voltage in question: 11.6667 | |

A legtöbb menüpont leírja, hogy mit vár, viszont a dipólusok iránya nem triviális: úgy kell megadni az éleket, hogy a később megadott hurkok irányított körök legyenek, jelenleg ezt manuálisan kell még megadni, de már készül a DFS algoritmus a program fejlesztéséhez. Valamint a Resistance calculator -nál is számolni kell, hogy az utolsó és első csúcs közé a program behelyettesít egy dipólust, emiatt úgy kell megadni a köröket.

# Osztályok statikus leírása

## matrix

### Felelőssége

A matrixok különböző matematikai tulajdonságai teszik lehetővé a késöbbi matematikai műveletek megoldását.

### Ősosztályok

-

### Attribútumok

#### Privát

int n, m; sor, oszlop

vector<double>\* data; adat

### Metódusok

#### Publikus

**Konstruktorok:**matrix();   
matrix(int sqr);  
matrix(const matrix& ref);  
matrix(vector<double> v);  
matrix(const int n, const int m, vector<double>\* data);

**Destruktor:**~matrix();

**Getterek:**int getn() const;  
int getm() const;

vector<double>\* getData() const;

**Feltöltők:**bool fill(int n, int m, vector<double> data);  
bool fill(int n, int m, vector<double>\* data);  
bool fill(int n, int m, double value);

**Nullázók:**bool null();  
bool null(int n, int m);

**Sor/oszlop törlő**matrix del\_row(int ezt = 0) const;  
matrix del\_col(int ezt = 0) const;

**Diagnosztikai kiírató (egyszerű megtalálni ctrl F -fel)**  
void print(std::ostream & = std::cout);

**Gauss elimináció**

double gauss();

**Transzponált**  
matrix T() const;

**Precíz és gyors determináns:**  
double determinant() const;

double qdeterminant() const;

**Inverz**  
matrix inverse() const;

**Hozzáadja d-t a megadott indexű helyhez**  
void plusat(const double d, const unsigned i, const unsigned j);

**Összeadás**  
matrix operator+(const matrix& theother) const;

**Szorzás mátrixszal**  
matrix operator\*(const matrix& theother) const;

**Szorzás vektorral**  
matrix operator\*(const vector<double>& theother) const;

**Szorzás skalárral**  
matrix operator\*(const double scalar) const;

**Egyébb operátorok:**  
void operator=(const matrix& theother);

vector<double> operator[](const unsigned i) const;

friend ostream& operator<<(ostream& out, const matrix& theother);

**Egymás után fűz két mátrixot**  
matrix operator&(const matrix& theother) const;

## menu

### Felelőssége

Kezeli és futtatja az adott beállítást.

### Ősosztályok

### Attribútumok

#### Privát

bool Createdanewifofstreamsocodeleaks; Enélkül egy beállítás szivárog ☹

ostream\* out; Kimenet

istream\* in; Bemenet

unsigned mode; mód

unsigned Function; funkció

graph\* G; Hálózat gráfja (upgrade idea: gráfjai?)

string dump; File beolvasáskor ide dobja ki a szemét sorokat a program

### Metódusok

#### Privát

void Mode(); Mód választó

void Func(); Funkció választó

void matrixcalc(); Mátrix számológép funkció

void Build(); Gráf építő

void Tester(); Teszt funkció

#### Publikus

**Konstruktorok:**  
menu(istream& InputConsole = std::cin, ostream& OutputConsole = std::cout)  
menu(ifstream& InputConsole, ofstream& OutputConsole)  
menu(menu& other) :out(other.out), in(other.in), mode(other.getmode()), Function(other.getfunc()) { G = new graph(\*other.G); }

**Destruktor**  
~menu();

**Getterek:**  
unsigned getfunc() const { return Function; }  
unsigned getmode() const { return mode; }

**Start menü**  
void start();

**Futtatás**  
void Run();

## Verticle

### Felelőssége

Reprezentálja a hálózat csomópontját

### Ősosztályok

### Attribútumok

#### Privát

double potential; A csomópont potenciálja (fejlesztés alatt)

double inflow; Befolyó áram (fejlesztés alatt)

bool status; Elért- e már a csúcs (fejlesztés alatt)

unsigned degree; fokszám

Edge \*\*edges; csatlakozó élek

### Metódusok

#### Publikus

**Konstruktorok**  
Verticle(double potential = 0, double inflow = 0, unsigned d = 0, Edge\*\* tmb = nullptr)  
Verticle(const Verticle& other)

**Destruktor**  
~Verticle();

**Setterek:**  
bool setPot(double potential);  
bool setStatus(bool status);

**Hozzá köt egy élt**  
bool addEdge(Edge\* DBG\_newedge);

**Igaz, ha az él csatlakozik hozzá**  
bool isEdge(Edge\* edge);

**Igaz, ha ez az adott csomópont**  
bool isMe(Verticle\* verticle);

**Getterek**  
double getPot() const;  
bool getStatus() const;  
unsigned getd() const;  
Edge\*\* getEdgeArray() const;

## Edge

### Felelőssége

Tisztán virtuális osztály a kétpólusokhoz.

### Ősosztályok

### Attribútumok

#### Privát

### Verticle\* a;

### Verticle\* b;

### double I; Rajta folyó áram

### double U; Rajta lévő feszültség

### Metódusok

#### Publikus

**Konstruktor**  
Edge(Verticle\* a = nullptr, Verticle\* b = nullptr) : a(a), b(b) , U(0), I(0)

**Setterek és getterek:**bool setU(const double u);  
ouble getU();  
bool setI(const double I);  
virtual double getI() const;

**Karakterisztikával tér vissza (kód)**  
virtual unsigned charteristics() const = 0;

**Csúcspont setterek és getterek**bool seta(Verticle\* a);bool setb(Verticle\* b);Verticle\* geta() const;Verticle\* getb() const;

**Megcseréli a-t b-vel**  
void switch\_direction();

**Visszatér a dipólus tipusával (pl Transformer) std::string formátumban**  
virtual string getType() const = 0;

## Resistance

### Felelőssége

Ellenállást reprezentál

### Ősosztályok

Edge

### Attribútumok

#### Privát

### double R; Ellenállás

### static string type;

### Metódusok

#### Publikus

**Konstruktor**  
Resistance(Verticle\* a = nullptr, Verticle\* b = nullptr, double R = 0)

**U/I**  
bool calcI();

**I\*R**  
bool calcU();

**getterek**  
double getR() const;  
double getP() const; **(teljesítmény: fejlesztés alatt)**

**Operátorok**  
void operator=(Resistance& other);  
Resistance& operator+(Resistance& other);

**Virtuális felülírások**  
string getType() const

unsigned charteristics() const;

## V\_src

### Felelőssége

Feszültség forrást reprezentál

### Ősosztályok

Edge

### Attribútumok

#### Privát

### double voltage;

### static string type;

### Metódusok

#### Publikus

**Konstruktor**  
V\_src(Verticle\* a = nullptr, Verticle\* b = nullptr, double U = 0)

**Setter feszültséghez**bool setV(const double U);

**Getter feszültséghez**  
double getV() const;

**Virtuális felülírások**  
string getType() const

unsigned charteristics() const;

## I\_src

### Felelőssége

Áram forrást reprezentál

### Ősosztályok

Edge

### Attribútumok

#### Privát

### double current;

### static string type;

### Metódusok

#### Publikus

**Konstruktor**  
I\_src(Verticle\* a = nullptr, Verticle\* b = nullptr, double I = 0)

**Setter áramhoz**bool setI(const double i);

**Getter áramhoz**  
double getI() const;

**Virtuális felülírások**  
string getType() const

unsigned charteristics() const;

## TwoPort

### Felelőssége

Csatolt kétpólusok virtuális ősosztálya

### Ősosztályok

Edge

### Attribútumok

#### Privát

### Edge\* Connected\_to;

### Metódusok

#### Publikus

**Konstruktor**  
TwoPort(Verticle\* a = nullptr, Verticle\* b = nullptr, Edge\* that = nullptr) : Connected\_to(that), Edge(a, b) {};

**Setter a csatoláshoz**  
bool setoth(Edge\* other);

**Getter a csatoláshoz**  
Edge\* getoth() const;

## Transformer

### Felelőssége

Ideális transzformátort reprezentál.

### Ősosztályok

Edge, TwoPort

### Attribútumok

#### Privát

### double n; Transzformátorra jellemző adat

### static string type;

### Metódusok

#### Publikus

**Konstruktor**  
Transformer(Verticle\* a = nullptr, Verticle\* b = nullptr,const double n = 1.0, Transformer\* that = nullptr)

**Setter n-hez**  
bool setn(const double n);

**Getter n-hez**  
double getn() const;

**Virtuális felülírások**  
string getType() const

unsigned charteristics() const;

## Gyrator

### Felelőssége

Girátort reprezentál.

### Ősosztályok

Edge, TwoPort

### Attribútumok

#### Privát

### double r; Girátorra jellemző adat

### static string type;

### Metódusok

#### Publikus

**Konstruktor**  
Transformer(Verticle\* a = nullptr, Verticle\* b = nullptr,const double r = 1.0, Gyrator\* that = nullptr)

**Setter r-hez**  
bool setr(const double r);

**Getter r-hez**  
double getr() const;

**Virtuális felülírások**  
string getType() const

unsigned charteristics() const;

## graph

### Felelőssége

Girátort reprezentál.

### Attribútumok

#### Privát

vector<Verticle\*> V; Csúcsok

Edge\*\* E; Élek (Edges)

unsigned n; Csúcsok száma

unsigned b; Élek száma

unsigned l; Hurkok száma

### Metódusok

#### Privát

void setl()

#### Publikus

**Konstruktorok:**  
graph(Verticle V[] = nullptr, Edge\*\* E = nullptr, unsigned n = 0, unsigned b = 0)  
graph(unsigned n);  
graph(graph& other);

**Destruktor**  
~graph();

**Felvesz egy csúcsot vagy élt a gráfba, illetve eltávolít egy élt**  
bool addv(Verticle\* v);  
bool adde(Edge\* e);  
void deledge(unsigned i);

**Getterek:**  
unsigned getn() const;  
unsigned getb() const;  
unsigned getl();  
Verticle\* getv(unsigned i) const;  
Edge\* getedge(unsigned i) const;

**Igazzal tér vissza, ha a csúcs vagy él benne van a gráfban**  
unsigned isverticle(Verticle \*v);  
unsigned isedge(Edge \*e);

**Megoldja a gráf minden élére a Kirchoff-egyenleteket és karakterisztikákat**  
bool solve(ostream& out = cout, istream& in = cin);

# UML osztálydiagramm

# A képen szöveg, képernyőkép, tervezés látható Automatikusan generált leírás

# Összegzés

## Mit sikerült és mit nem sikerült megvalósítani a specifikációból?

Minden kitűzött célt elértem, viszont a formátumon némileg hibakezelési megfontolásokból módosítottam.  
Van néhány csatolt kétpólus amelyeket még nem implementáltam, de ezeknek már csak az idő szabott határt.

## Mit tanultál a megvalósítás során?

A kivételkezelések jelentősségét elsősorban. A polimorfizmus és a heterogén kollekciók hasznát is szívesen ismertem meg. A rekurzió okozta problémákkal is jobban megismerkedtem. Template osztályt vagy függvényt magamnak nem állítottam elő, viszont használtam az std::vector<T> osztályt. A kimenetet ostream-mel és ofstream-mel kezeltem ami szintén újdonság volt, ugyanez igaz a bemenetre is.

## Továbbfejlesztési lehetőségek

A kódomban a kommentek között feltűntettem rengetek fejlesztési ötletet, közülük néhány:

* Automatikus hurok keresés
* Csomópontok potenciáljának meghatározása
* Még több csatolt kétpólus
* Még több funkció (pl. impedancia kalkulátor)
* Komplex mátrixok
* Néhány dinamikus probléma megoldása

# Képernyőképek a futó alkalmazásról

A képen szöveg, képernyőkép, képernyő, szoftver látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, szoftver, képernyő látható

Automatikusan generált leírás