Institut für Wissenschaftliches Rechnen
Technische Universität Braunschweig
Prof. Hermann G. Matthies, Ph. D.
Dr. Elmar Zander

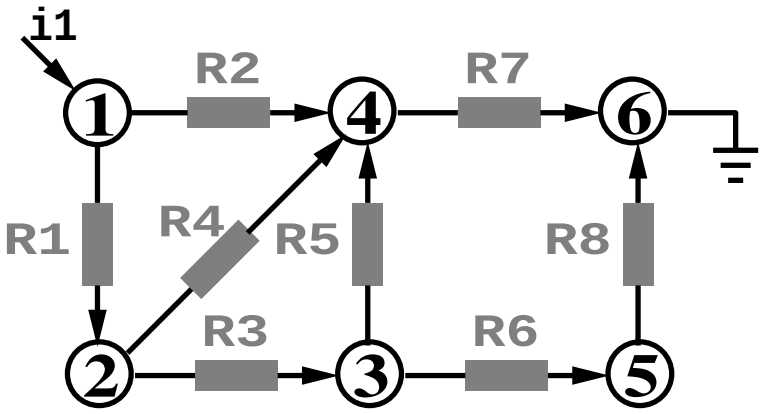
Sommersemester 2010
14. November 2014

Weiterführendes Programmieren

Lineare Widerstandsnetzwerke mit Matrixassemblierung

Aufgabenblatt 5

In den folgenden beiden Aufgabenblöcken sollen Sie eigenständig, ohne viele Vorgaben, verschiedene Löser für Probleme der linearen Schaltungsanalyse entwickeln. Dazu sollen die erforderlichen Gleichungssysteme auf zwei verschiedene Arten erzeugt werden. Auf dem 5. Aufgabenblatt, also dem Vorliegenden, wird die Systemmatrix mit der sogenannten Direct Stiffness Method (`http://en.wikipedia.org/wiki/Direct_stiffness_method`) erzeugt, auf dem 6. Aufgabenblatt wird die Systemmatrix über eine Maschenanalyse erzeugt.
Lineare Netzwerke können als Graphen aufgefasst werden. In linearen elektrischen Netzwerken haben die Kanten eine Richtung (die Stromzählrichtung) und einen Widerstand.




```
NODE(1)
NODE(2)
NODE(3)
NODE(4)
NODE(5)
NODE(6)
EDGE(1,2,10.0)
EDGE(1,4,10.0)
EDGE(2,3,10.0)
EDGE(2,4,10.0)
EDGE(3,4,100.0)
EDGE(3,5,10.0)
EDGE(4,6,10.0)
EDGE(5,6,10.0)
```

Abbildung 1: Schaltungsnetzwerk. Das Netzwerk aus der Abbildung wird durch die Eingabedatei auf der rechten Seite beschreiben.

Zusammenfassung der elektrotechnischen Begriffe

In diesem Aufgabenblock werden einige Begriffe verwendet, die aus der Elektrotechnik stammen und hier kurz erklärt werden.

Knoteninzidenzmatrix Beschreibt die Schaltungstopologie und enthält eine Zeile für jeden Knoten, eine Spalte für jede Kante.

$$A_{i,e} = +1 \text{ , wenn Kante } e \text{ von Knoten } i \text{ wegzeigt.} \quad (1)$$

$$A_{i,e} = -1 \text{ , wenn Kante } e \text{ zu Knoten } i \text{ hinzeigt.} \quad (2)$$

$$A_{i,e} = 0 \text{ , sonst.} \quad (3)$$

Zweig Die Begriffe "Zweig" und "Kante" sind synonym.

Zweigimpedanzmatrix Z ist eine Diagonalmatrix mit

$$Z_{i,i} = R_i. \quad (4)$$

Admittanz = (komplexer)Leitwert = $\frac{1}{\text{Impedanz}} \simeq \frac{1}{\text{Widerstand}}$. Dies gilt im reellen Fall, bei Gleichstrom; Wechselströme und nicht-lineare Bauteile werden in dieser Aufgabe allerdings nicht betrachtet. Die Matrix Y ist hier eine Diagonalmatrix mit $Y_{i,i} = 1/R_i$.

Benutzung des Graph-Interpreters

Der Graph-Interpreter, der auf den letzten Aufgabenblättern entwickelt wurde, kann auch in dieser Aufgabe genutzt werden, allerdings soll er hier nur zum Einlesen des gewichteten Graphen Verwendung finden. Dies kann mit einer Schleife wie in Listing 1 geschehen.

```
1  while((type=readline(ifile,&context))!=END)
2  {
3    interpret(stderr,context,type,&num_edge,edge_array,&num_node,node_array);
4  }
```

Listing 1: Schleife zum Einlesen eines Graphen

Übung 1: *Matrix-Bibliothek erweitern*
Listing 2 zeigt eine Datei `lin_ops.h`, welche die Deklarationszeilen der Matrix und Vektoroperationen angibt, die für die Lösung dieser Aufgaben benötigt werden. Die Allokation des Speichers für die Matrix kann vom letzten Aufgabenblatt übernommen werden (`lin_struct.h`).

```
1  #ifndef LINOPS__H
2  #define LINOPS__H
3    typedef enum error_type_enum {
4          SUCCESS=1,          GENERALERROR=-1,
5          WRONGDIMENSION=-2, NUMERICAL_ERROR=-3
6    } error_type;
7    typedef struct matrix_struct
8    {
9       int dim1;
10      int dim2;
11      double** value;
```

```
12   } matrix;
13   typedef struct vector_struct
14   {
15      int dim;
16      double* value;
17   } vector;
18
19   void create_matrix(matrix* mat, int _dim1, int _dim2);
20   void free_matrix(matrix* mat);
21   void copy_matrix(matrix* new_mat, matrix old_vec);
22   void create_vector(vector* vec, int _dim);
23   void free_vector(vector* vec);
24   void copy_vector(vector* new_vec, vector old_vec);
25
26   /* implement multiplication result=AB */
27   error_type mat_mat_mul(matrix result, matrix A, matrix B);
28
29   /* implement multiplication result=AB^T */
30   error_type mat_matT_mul(matrix result, matrix A, matrix B);
31
32   /* implement multiplication result=Ax */
33   error_type mat_vec_mul(vector result, matrix A, vector x);
34
35   /* delete row i of matrix*/
36   error_type delete_row(int i,matrix A);
37
38   /* delete column i of matrix*/
39   error_type delete_column(int i,matrix A);
40
41   /* solve Ax=b */
42   error_type solve(vector x, matrix A, vector b);
43 #endif
```

Listing 2: lin_ops.h

Die Funktion `solve` ist in dem Paket `http://www.wire.tu-bs.de/ADV/files/linops/linops.zip` enthalten und implementiert den aus Aufgabeblatt 2 bekannten Lösungsalgorithmus.
Aufgabe a) Implementieren Sie die in Listing 2 deklarierten Funktionen (außer der gegebenen Funktion `solve`). □

Übung 2: *Matrix Assemblierung nach der* Direct Stiffness Method

Um die resultierenden Spannungen bei gegebenen Knotenquellenströmen in einem Widerstandsnetzwerk zu bestimmen, muss folgendes Gleichungssystem gelöst werden:

$$\underbrace{\underline{\underline{A}}\,\underline{\underline{Y}}\,\underline{\underline{A}}^T}_{\underline{\underline{Y}}_n}\,\underline{u}_n = \underline{i}_n \tag{5}$$

Die *Direct Stiffness Method* (DSM) kann dazu verwendet werden, die Knotenadmittanzmatrix ($\underline{\underline{A}}\,\underline{\underline{Y}}\,\underline{\underline{A}}^T$) aus Gleichung 5 effizient aufzustellen. Die DSM kann an linearen Netzwerken sehr leicht hergeleitet und verstanden werden.

Aufgabe a) Die Matrix $\underline{\underline{Y}}_n$ kann man als Summe von "Elementmatrizen" berechnen. Um dies zu sehen, stellen Sie die Matrix $\underline{\underline{Y}}_n$ für die folgenden Graphen von Hand auf (siehe dazu Gleichung 5). □

Zunächst ein Graph mit nur einer Kante und zwei Knoten:

```
Node(1)
Node(2)
Edge(1, 2, 10.0)
```

Dann ein Graph mit nur einer Kante aber vielen Knoten:

```
Node(1)
...
Node(8)
Edge(3, 6, 10.0)
```

Wieder ein Graph mit nur einer Kante aber vielen Knoten:

```
Node(1)
...
Node(8)
Edge(6, 2, 20.0)
```

und nun die Kombination der letzten beiden Graphen:

```
Node(1)
...
Node(8)
Edge(3, 6, 10.0)
Edge(6, 2, 20.0)
```

Sie sehen, dass die Knotenadmittanzmatrix des kombinierten Graphen als Summe der Matrizen der Graphen mit nur einer Kante geschrieben werden kann. Die "Elementmatrizen" können also sehr leicht direkt auf Grund der Kanteninformationen aufgestellt werden.

Aufgabe b) Implementieren sie eine Funktion, die einen gewichteten Graphen als Eingabe nimmt und die Knotenadmittanzmatrix zu dem Graphen berechnet. Dazu soll die *Assemblierung* mit der folgenden Berechnungsvorschrift umgesetzt werden. Diese Berechnungsvorschrift setzt die DSM für den hier behandelten Fall um.

```
for all Edges e
  A[e.source][e.source] += ( 1 / e.weight )
  A[e.source][e.target] -= ( 1 / e.weight )
  A[e.target][e.source] -= ( 1 / e.weight )
  A[e.target][e.target] += ( 1 / e.weight )
```

Listing 3: Pseudocode zur Erzeugung der Knotenadmittanzmatrix

□

Übung 3: *Gleichungssystem lösen*

Aufgabe a) Schreiben Sie zunächst ein Programm, welches den am Anfang beschriebenen Graphen einliest und die passende Knotenadmittanzmatrix aufstellt. Lösen Sie das Gleichungssystem dann mit der Funktion `solve`, wobei Sie als rechte Seite die externen Ströme $i_1 = 0.1$ und $i_{2..6} = 0$ verwenden. □

Sie sehen, dass der Algorithmus das Gleichungssystem nicht lösen kann. Das liegt daran, dass es für das Problem keine eindeutige Lösung gibt. Das lässt sich sehr schön an einem kleinen Beispiel (siehe Abbildung 2) verstehen: Hier ist der Spannungsabfall über dem Widerstand zwar $1V$, man kann aber keine

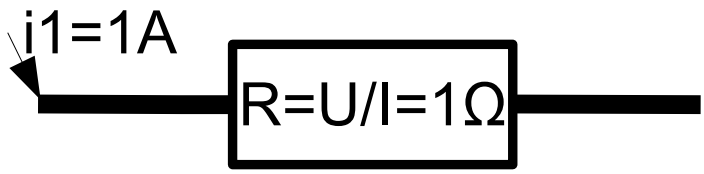


Abbildung 2: Einfaches Beispiel

Aussage über die absolute Spannung an einem der beiden Enden der Schaltung machen. Mit anderen Worten gibt es unendlich viele Lösungen für das Problem. Um die Lösung eindeutig zu machen, kann man eine der Knoten erden. In dem Gleichungssystem muss dazu eine unbekannte eliminiert werde.

Aufgabe b) Benutzen sie die Funktionen:

```
delete_row(int i,matrix A)
delete_column(int i,matrix A)
```

um eine Erdung am Knoten 6 einzuführen. Und lösen Sie danach erneut. □

Das Beispielnetzwerk von oben, mit geerdetem Knoten 6, hat relativ zu Knoten 6 die folgenden Knotenspannungen:

$$u_1 = 1.284, u_2 = 0.8581, u_3 = ???, u_4 = 0.7097, u_5 = 0.2903$$

Aufgabe c) Verifizieren Sie mit Hilfe Ihres Programms diese Potentiale und geben Sie das Potential u_3 an. □