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 II
Aufgabenblatt 6

In diesem Aufgabenblock geht es, wie schon im letzten, um die Berechnung von linearen Netzwerken. Dazu werden wir nochmal kurz die elektrotechnischen Begriffe wiederholen. Als Eingabedaten für die Algorithmen benutzen Sie bitte nochmals das Beispiel vom letzten Aufgabenblatt.

1 Zusammenfassung der elektrotechnischen Begriffe

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

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

Zweigimpedanzmatrix Z ist eine Diagonalmatrix mit

$$Z_{i,i} = R_i. \tag{1}$$

Fundamentalmaschenmatrix B enthält eine Zeile pro *Fundamentalmasche*, und eine Spalte pro Zweig. Die Einträge der Matrix sind $+1$ wenn die entsprechende Kante (der entsprechende Zweig) in der selben Richtung definiert ist, wie sie bei durchlaufen der Fundamentalmasche durchlaufen wird, -1, wenn sie entgegengerichtet ist, und 0 wenn diese Kante in der Fundmentalmasche nicht vorkommt.

Maschenanalyse: Auf diesem Aufgabenblatt soll eine Maschenanalyse durchgeführt werden. Das bedeutet hier, dass folgendes Gleichungssystem gelöst werden soll:

$$\begin{aligned} BZB^T i_l &= u_l \\ \text{mit } u_l &= -Bu \end{aligned}$$

mit den Zweigquellenspannungen u, Kreisströmen i_l, Maschenquellenspannungen u_l.

2 Maschenanalyse

Um Zweigströme i aus gegebenen aufgeprägten Spannungen u zu berechnen, muss man bei komplexeren Schaltungen eine Maschenanalyse durchführen und die Kirchhoff'sche Maschenregel anwenden. Bei der Suche nach diesen Maschen kann man eine Tiefensuche anwenden. Auf Basis dieser Tiefensuche kann man entscheiden, ob ein Graph zusammenhängend ist und einen Baum berechnen, der den Graphen aufspannt; ein sogenannter Spannbaum (siehe `http://de.wikipedia.org/wiki/Spannbaum`). Ein Spannbaum enthält alle von einem Referenzknoten aus erreichbaren Knoten. Wenn es einen Spannbaum gibt, der alle Knoten enthält, dann ist der Graph zusammenhängend. Dieser Baum wird benötigt, um Maschen im Graphen zu identifizieren.

Übung 1: *Tiefensuche*

Zum Konstruieren eines Baumes aus einem Graphen soll hier die Tiefensuche (depth first search (DFS)) angewendet werden, welche im folgenden Pseudocode gegeben ist. Dieser ist — wie viele Algorithmen aus der Graphentheorie — in Form einer Graphenfärbung formuliert. Um die Färbung des Graphen zu implementieren kann man wieder ein `enum`-Typ benutzen.

```
1  DFS(Graph G)
2  {
3    for each node n of G                // Color all nodes white
4    {
5      nodecol[n] = 'w';
6      pre[n] = nil;                     // set all predecessor to nil
7    }
8
9    for each edge e of G                // Color all edges white
10     edgecol[e] = 'w';
11
12   for each node n of G                // For each still white node
13     if nodecol[n] == 'w'
14       DFS-visit(n,0);                 //    : call DFS-visit
15 }
16
17 DFS-visit(u,depth)
18 {
19   nodecol[u] = 'g';                   // Color actual node grey
20   d[u] = depth;                       // set counter of actual node
21   for each node n of Adj[u]           // Adj[u] are all neighbors
22   {                                   // For each white neighbor node u
23     if nodecol[n] == 'w'
24     {
25       pre[n] = u;                     // set pred. of actual node
26       edgecol[id of edge(u,n)] = 'b'  // color all used edges black
27       DFS-visit(n,depth+1);           // call DFS-visit
28     }
29   }
30   col[u] = 'b';                       // Color actual node black
31 }
```

Listing 1: Pseudocode zur Tiefensuche

Bemerkung: Das Array `Adj[u]` in Zeile 21 enthält nicht nur die Nachbarn, für die eine Kante vom Knoten `u` aus erreichbar ist, sondern auch diejenigen, die u über eine Kante erreichen. Hier wird der eigentlich gerichtete Graph also als ungerichtet betrachtet.

Aufgabe a) Implementieren Sie diesen Algorithmus und testen Sie ihn. □

Übung 2: *Maschenidentifikation*

Ein Kreis ist ein Pfad in einem Graphen, dessen Startknoten und Endknoten übereinstimmen. Ein zu-

sammenhängender Graph, der keine Kreise enthält, ist ein Baum.
Ein Spannbaum (engl. spanning tree) eines ungerichteten zusammenhängenden Graphen ist ein Teilgraph, der ein Baum ist und alle Knoten des betrachteten Graphen enthält. Jede Kante eines ungerichteten zusammenhängenden Graphen, die nicht zu einem Spannbaum des Graphen gehört, definiert einen Kreis. Wir nennen eine solche Kante im folgenden Text *Verbindungskante*. Dass jede Verbindungskante einen Kreis definiert ist nachvollziehbar, wenn man beachtet, dass im Spannbaum eines ungerichteten zusammenhängenden Graphen bereits alle Knoten des Graphen enthalten sind. Eine zusätzliche Kante muss also bereits über einen Pfad verbundene Knoten miteinander verbinden und damit einen Kreis erzeugen. Für die Kreisidentifikation gibt es eine Vereinfachung, wenn der Spannbaum durch eine Tiefensuche (*DFS*), wie in unserem Fall, entstanden ist: Dann muss nämlich jede *Verbindungskante* eine Kante zu einem anderen Knoten auf dem gleichen Ast des Baumes sein. Da Kreise aus der Graphentheorie in der Elektrotechnik den Maschen in der Schaltungsanalyse entsprechen, definiert eine *Verbindungskante* daher genau eine Masche.
Ein Pseudocode, um alle Maschen (Kreise — engl.: Cycles) in einem Graphen zu finden, ist im folgenden Listing angegeben.

```
getCycles(Graph G)
{
  for each edge e of G
  {
    if edgecol[e] = 'w'
    {
      if(d[e.source] < d[e.target])
        cycle[e]=buildCycle(e.source,e.target)
      else
        cycle[e]=buildCycle(e.target,e.source)
    }
  }
}

buildCycle(source,target)
{
  circle.append(target) // target is first node of cycle
  p = source;
  while p!=target       // go branch upwards towards node u
  {
    cycle.append(p)     // add nodes in branch to cycle
    p=pre[p]
  }
  return cycle
}
```

Listing 2: Pseudocode zur Maschenidentifikation

Aufgabe a) Implementieren Sie den Algorithmus zur Identifikation der Maschen. □

Übung 3: *Maschenanalyse*
Um nun eine Maschenanalyse auf dem Netzwerk durchzuführen, müssen Sie die Fundamentalmaschenmatrix B aufstellen.

Aufgabe a) Erzeugen Sie die Fundamentalmaschenmatrix, indem Sie die Fundamentalmaschen durchlaufen und die entsprechenden Zahlen in die Matrix eintragen. □

Um die Zweigströme in einem Widerstandsnetzwerk zu bestimmen, muss folgendes Gleichungssystem gelöst werden:

$$\begin{aligned} BZB^T i_l &= u_l \\ \text{mit } u_l &= -Bu \end{aligned}$$

Dabei ist Z die Zweigimpedanzmatrix, die im Fall eines Widerstandsnetzwerkes eine Diagonalmatrix ist. Es geht nun darum, aus einem gegebenen Zweigquellenspannungsvektor u die resultierenden Kreisströme i_l zu berechnen, aus denen dann die Zweigströme i mit der Beziehung

$$i_l = -B^T i$$

abgeleitet werden können.

Bemerkung: Beachten Sie den Unterschied zwischen u, dem gegebenen Vektor der Zweigquellenspannungen, und u_l, dem (unbekannten) Vektor der Maschenquellenspannungen, der die maschenweise Spannungsverteilung bei angelegter Zweigquellenspannung u angibt.

Aufgabe b) Schreiben Sie eine Funktion, welche aus einem gegebenen (mit Widerstandswerten gewichteten) Graphen und angegebenem Zweigspannungsvektor über eine Maschenanalyse die gesuchten Größen i und u_l berechnet. □

Aufgabe c) Führen Sie eine Maschenanalyse in dem Beispielnetzwerk aus Abbildung 1 vom letzten Aufgabenblatt durch (Eingabevektor $u = (5, 0, 0, 0, 0, 0, 0, 0)^T$) und beantworten Sie folgende Frage: Wenn zwischen Knoten 1 und Knoten 2, in Serie mit R1, eine ideale Spannungsquelle mit 5 Volt geschaltet ist, welche Ströme fließen dann durch die einzelnen Widerstände? Ein Teil der Antwort lautet: $i_2 = -0.1806A$, $i_5 = 0.009677A$. Berechnen Sie die restlichen Ströme mit dem Programm. □