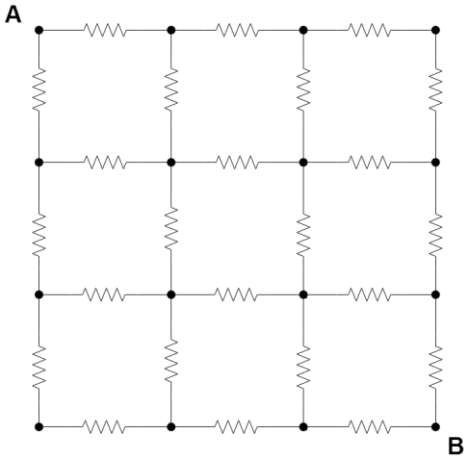
Sieć rezystorów

Wstęp

Aby stworzyć „inteligentny” materiał, chcemy wszyć w tkaninę przewodzące włókna. Uszkodzenie przerwie włókna co zmieni oporność tworzonej przez nie sieci. Zmiana oporności może być wykryta przez układ elektroniczny. Sieć w stanie idealnym, zanim zostanie uszkodzona, wygląda tak jak na rysunku 1. Liczba oczek pionowo i liczba oczek poziomo może być inna: nie 3x3, ale np. 6x7, a ogólnie $n-1$ wierszy i $m-1$ kolumn. Zakładamy że $2 \leq n \leq 100,\ 2 \leq m \leq 100$. Węzły oznaczone jako A i B służą do podłączenia do układu pomiarowego. Każdy z rezystorów ma, początkowo, taką samą oporność $R = 1\ [\Omega]$.

Zakładamy że 5% zmiana oporu zastępczego całej sieci jest możliwa do wykrycia. Jednak gdy będzie ona mniejsza, to cały pomysł wykrywania uszkodzeń będzie zbyt trudny do zrealizowania.



Rysunek 1. Sieć oporników 3x3 oczka.

Zadanie

Stworzyć program/skrypt w języku Matlab, który oblicza opór zastępczy pomiędzy punktami A i B tego rodzaju sieci oporników. Program ma najpierw obliczyć opór gdy wszystkie oporniki mają ten sam opór. Następnie ma przeliczyć wszystkie warianty scenariusza, w którym został uszkodzony jeden z oporników. Jako wyniki program ma podać: opór zastępczy przed uszkodzeniem oraz maksymalną i minimalną wartość oporu zastępczego po takim uszkodzeniu. Powinien też podać o ile procent zmieni się opór zastępczy w najgorszym przypadku, tj. wtedy zmiana oporności będzie najtrudniejsza do wykrycia. Dane wejściowe to liczby n i m oraz opór R.

Rozwiązanie

Przepływ prądu elektryczny w pojedynczym przewodniku opisuje prawo Ohma, złożone układy (takie jak mamy w zadaniu) można rozwiązać stosując pierwsze i drugie prawo Kirchhoffa. Bezpośrednie zastosowanie praw Kirchhoffa prowadzi do zbyt wielu równań i dlatego w praktyce lepiej jest zastosować albo metodę prądów oczkowych, albo metodę potencjałów węzłowych.

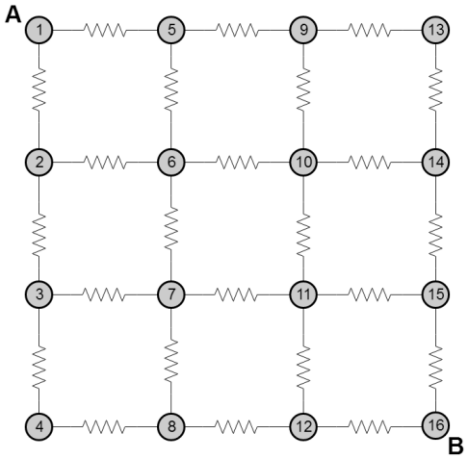
Ponieważ, niezależnie od uszkodzeń oporników, węzły będą zawsze istnieć to stosujemy metodę potencjałów węzłowych.[1] Aby obliczyć rozpływ prądów będziemy rozwiązujemy układ równań

$$\begin{cases} G_{11}V_1 + G_{12}V_2 + \cdots G_{1l}V_k = I_1 \\ G_{21}V_1 + G_{22}V_2 + \cdots G_{2l}V_k = I_2 \\ \cdots \\ G_{l1}V_1 + G_{l2}V_2 + \cdots G_{ll}V_l = I_l \end{cases}$$

gdzie $l = nm$, G_{ii} obliczamy jako sumy przewodności wszystkich gałęzi bezpośrednio dołączonych do i-tego węzła, G_{ij} dla $i \neq j$ jest zanegowaną sumą przewodności gałęzi łączących węzeł i-ty z węzłem j-tym, prądy gałęziowe I_i obliczamy dzieląc SEM w danej gałęzi przez jej oporność. Taki układ równań możemy, zmieniając trochę oznaczenia, łatwo zapisać w postaci macierzowej :

$$\mathbf{Gv} = \mathbf{b}\,, \quad \mathbf{G} = \begin{pmatrix} G_{11} & G_{12} & \cdots & G_{1l} \\ G_{21} & G_{22} & \cdots & G_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ G_{l1} & G_{l2} & \cdots & G_{ll} \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_l \end{pmatrix} \quad \text{oraz} \quad \mathbf{b} = \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_l \end{pmatrix}$$

W naszej sieci oporników węzły numerujemy od góry do dołu, kolejnymi kolumnami, tak samo jak robi to Matlab z elementami macierzy (rysunek 2.). Dla danych n i m, węzeł w i-tym wierszu j-tej kolumny ma numer $n(j-1)+i$.



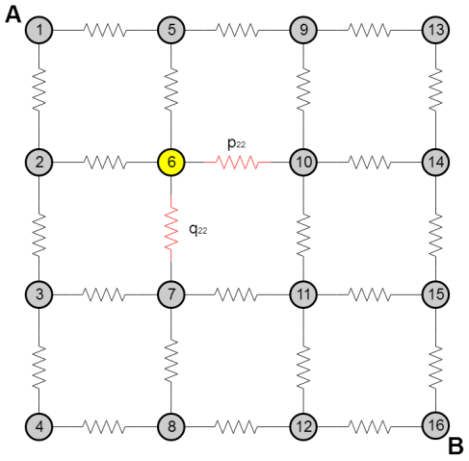
Rysunek 2. Numeracja węzłów.

Niestety, tak skonstruowana macierz $\mathbf{G}$ jest macierzą osobliwą. Dlaczego? Z fizyki pola elektrycznego wiemy że potencjał elektryczny można określić z dokładnością do stałej całkowania. Stałą tą możemy przyjąć dowolnie, więc potencjał nie jest określony jednoznacznie. Co nie przeszkadza: gdy obliczymy napięcie różnicę potencjałów to stała całkowania w odjemnej i odjemniku wzajemnie się znoszą. Można oczywiście użyć *pinv(G)* dla obliczenia macierzy odwrotnej (w nowszych wersjach programu Octave lub Matlab; Matlab 2010b nie potrafi obliczać macierzy pseudoodwrotnej z osobliwej macierzy rzadkiej). My jednak użyjemy prostej sztuczki: skoro równania nie są liniowo niezależne, to można usunąć z nich ostatnie i zastąpić je przez równanie $V_l = 0$, czyli wybrać taką

[1] *Nodal analysis of finite square resistive grids and the teaching effectiveness of students' projects*, P. Zegarmistrz et al., 2nd World Conference on Technology and Engineering Education, Ljubljana, Slovenia, 5-8 September 2011, pp. 84-87.

stałą całkowania, aby ostatni węzeł miał potencjał zero. Czyli modyfikujemy: macierz $\mathbf{G}$, aby $G_{lk} = 0$ dla każdego $k \neq l$ oraz $G_{ll} = 1$; macierz $\mathbf{b}$ tak, aby $b_l = 0$.

Przy założonych rozmiarach naszej sieci oporników macierz $\mathbf{G}$ może być dość duża by sensowne było użycie macierzy rzadkich. Dla $n = 100$ i $m = 100$ macierz $\mathbf{G}$ ma rozmiar 10000 na 10000, ale ponieważ każdy węzeł łączy się tylko z dwoma, trzema lub czterema sąsiednimi, więc niezerowych elementów jest nie więcej niż 500, czyli 5%. Dane o przewodności gałęzi trzymać będziemy w dwóch macierzach $\mathbf{p}$ i $\mathbf{q}$ mających po nm elementów każda (rysunek 3.). Wartości p_{ij} będą to przewodności gałęzi poziomych pomiędzy węzłem w i-tym wierszu i j-tej kolumnie a węzłem w tym samym wierszu i kolumnie $j + 1$. Wartości q_{ij} to przewodności pomiędzy węzłem i-tym wierszu i j-tej kolumnie a węzłem w $(i + 1)$ wierszu i kolumnie j-tej. Elementy o przewodnościach p_{im} i p_{jn} nie będą używane, nie ma ich na schematach. Zerowa przewodność oznacza przerwane/nieistniejące połączenie, czyli nieskończoną oporność.



Rysunek 3. Węzeł 6 jest w drugim wierszu i drugiej kolumnie – dlatego p_{22} i q_{22} dla oporników w prawo i w dół .

Dla mniej złożonego problemu, samego obliczenia oporu zastępczego sieci jednakowych oporników, można znaleźć kody źródłowe opublikowane na *Rosetta Code* (niestety generowane są macierze osobliwe).[2] Napisanie poprawnego programu w Matlabie, rozwiązującego konkretnie nasz problem, jest względnie proste (patrz listing 1.).

Nasz program jest kompromisem pomiędzy szybkością działania, wysiłkiem włożonym w jego napisanie i czytelnością kodu. Mógłby działać szybciej gdyby uniknąć wielokrotnego tworzenia macierzy $\mathbf{G}$, unikać instrukcji *if* wywoływanych w pętlach *for* oraz usprawnić modyfikowanie wektorów $\mathbf{p}$ i $\mathbf{q}$, a być może także gdyby użyć szybszego schematu rozwiązywania układu równań. Jednak zysk z oszczędzenia pracy komputera nie jest obecnie wart dodatkowej pracy człowieka. Ważniejsza jest czytelność kodu.

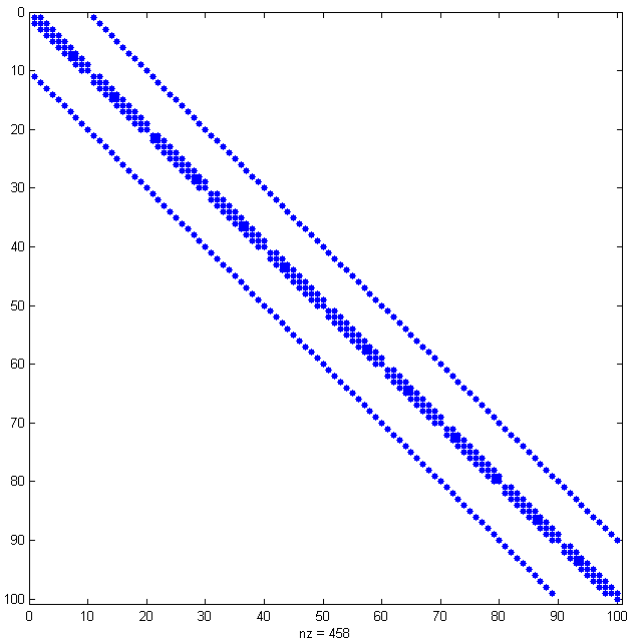
Program dawał wyniki, dla małych sieci, zgodne z otrzymanymi z *PartSim* – wersji on-line programu *PSpice*.[3] Nota bene: *PartSim* jest doskonałym programem do symulacji obwodów elektronicznych (także nieliniowych), ale dla sieci 10x10 jest 100 węzłów i 180 oporników, więc rysowanie 180 razy takiego obwodu myszką na ekranie byłoby pracochłonne – program w Matlabie jest znacznie bardziej efektywny.

[2] http://rosettacode.org/wiki/Resistor_mesh#Octave, 1 kwietnia 2017
[3] http://www.partsim.com/, 1 kwietnia 2017

Patrząc na wyniki dla sieci 10x10 (Octave 3.6.4):

```
r0 =  3.0117
drmax =  0.82707
drmin =  0.0011220
drmax_relative =  0.27462
drmin_relative = 3.7254e-004
ans =  3.0117
```

widzimy że względna zmiana oporu to poniżej 0.04% w najgorszym przypadku, co jest w praktyce niemożliwe w prosty sposób do wykrycia. Dla sieci o większej liczbie węzłów (w obu kierunkach) będzie jeszcze gorzej. Uruchamiając program z odpowiednimi danymi możemy dowiedzieć się że sieć 4x10 to pesymistycznie około 0.3% zmiana, sieć 4x4 daje (w niektórych przypadkach) mniej niż 2.6% wzrost oporności. Czyli poniżej założonych 5%.



Rysunek 4. Nieuszkodzona sieć 10x10, wywołanie spy(G), program Matlab 2010b.

Można oczywiście funkcję *ResistorMesh* wywoływać z innego skryptu Matlaba, np. po to aby sporządzić wykres zależności *drmin_relative* od n i m. Można zróżnicować oporności poszczególnych elementów itp. itd. Warto zauważyć (rysunek 4.) że macierz $\mathbf{G}$ jest macierzą pasmową, można więc próbować różnych metod rozwiązywania odpowiednich w takich przypadkach. Można w rozmaity sposób udoskonalić obliczenia, aby były szybsze, dokładniejsze i dostarczały więcej danych.

Wnioski

Czy tkanina z zaszytym drutem oporowym w formie siatki jest dobrym pomysłem na materiał wykrywający uszkodzenia jako zmianę oporu? Jak wynika z naszych obliczeń: nie jest to dobry pomysł jeżeli połączenia będą według schematu z rysunku 1. Zmiany oporności mogą być zbyt małe, aby skutecznie je wykrywać.

Listing 1.

```
function [r0, drmin_relative ] = ResistorMesh(n,m,R)
% Obliczanie rezystancji inteligentnej tkaniny, funkcja dla programu MATLAB.
%
% Dane:
%
%   n - liczba węzłów poziomo
%   m - liczba węzłów pionowo
%   R - opór pojedynczej gałęzi (w omach)
%
% Wyniki:
%
%   r0             - opór zastępczy układu (w omach)
%   drmin_relative - najmniejsza możliwa względna zmiana oporu wywołana
%                    zniszczeniem (przerwaniem) jednego z oporów;
%                    np. wartość drmin_relative = 0.05 oznacza iż możliwe jest
%                    uszkodzenie które da wzrost oporu o jedynie 5%.

    % Jeżeli nie podano n, m, R to przyjmowane są wartości domyślne.
    %
    if nargin < 1, n = 10; end
    if nargin < 2, m = 10; end
    if nargin < 3, R =  1; end

    % Sieć oporników jest zapamiętana jako dwa wektory przewodności:
    % p poziomo,q to te ułożone pionowo. Czyli z węzła o współrzędnych (i,j),
    % który ma numer n*(j-1) + i, wychodzić mogą gałęzie p(i,j-1), p(i,j),
    % q(i-1,j), q(i,j) jeżeli nie jest on położony na brzegu.
    %
    % Wektory p0, q0 przechowują pełną siatkę połączeń.
    %
    p0 = ones(n,m-1) / R;
    q0 = ones(n-1,m) / R;

    % Obliczanie oporu zastępczego dla pełnej sieci oporników.
    %
    r0 = resistance(n,m,p0,q0);

    % Obliczanie oporu zastępczego dla usuniętych pojedynczych oporników
    % - wybierane są tylko poziome. Wyniki są wpisywane do tablicy rp.
    %
    rp = zeros(size(p0));
    for k = 1:length(p0);
        p = p0;
        p(k) = 0;
        rp(k) = resistance(n,m,p,q0);
    end

    % Obliczanie oporu zastępczego dla usuniętych pojedynczych oporników
    % - wybierane są tylko pionowe. Wyniki są wpisywane do tablicy rq.
    %
    rq = zeros(size(q0));
    for k = 1:length(q0);
        q = q0;
        q(k) = 0;
        rq(k) = resistance(n,m,p0,q);
```

```
    end

    % Scalanie danych, z powstającego wektora usuwane są wszystkie powtarzające
    % się wyniki.
    %
    r = unique( [ rp(:) ; rq(:) ] );

    % Wypisywanie, w bardzo uproszczony sposób, wyników na konsoli:
    % drmax jest maksymalną wartością bezwzględnej różnicy pomiędzy oporem
    % sieci oporników po uszkodzeniu (jednego oporu) i przed uszkodzeniem;
    % drmin jest minimalną wartością bezwzględnej różnicy pomiędzy oporem
    % sieci po i przed uszkodzeniem.
    %
    r0
    drmax = max( abs( r - r0 ) )
    drmin = min( abs( r - r0 ) )
    drmax_relative = drmax / r0
    drmin_relative = drmin / r0

end


function equivalent_resistance = resistance(n,m,p,q)
% Obliczanie oporu zastępczego dla liczby węzłów n, m oraz wektorów przewodności
% p,q. Ponieważ jest to funkcja lokalna, to nie będzie (bo nie może) wywoływana
% inaczej niż z funkcji ResistorMesh. Nie musimy więc zastanawiać się nad tym
% gdzie i jak mogłaby być użyta, czy podano wszystkie dane wejściowe itd. itp.
%
% Jest w tym samym pliku co funkcja ResistorMesh - czyli trudniej o pomyłkowe
% użycie nie tej co trzeba funkcji resistance w ResistorMesh.
%
% UWAGA: obliczenia mogłyby przebiegać szybciej, gdyby nie tworzyć za każdym
% razem całej macierzy G. Jednak zysk na czasie nie jest aż tak wielki jakby
% mogło się wydawać - tworzenie jest szybkie w porównaniu z odwracaniem.

    % Utworzenie macierzy przewodności G.
    %
    nm = n*m;
    G = sparse(nm,nm); % utworzenie zerowej macierzy rzadkiej
    for i = 1:n
        for j = 1:m
            k = n * (j - 1) + i;  % numer węzła
            if j > 1 % nie jest to lewy brzeg, więc jest sąsiad z lewej
                k1 = n * (j - 1 - 1) + i;
                G(k,k) = G(k,k) + p(i,j-1);
                G(k,k1) = G(k,k1) - p(i,j-1);
            end
            if j < m % nie jest to prawy brzeg, więc jest sąsiad z prawej
                k1 = n * (j - 1 + 1) + i;
                G(k,k) = G(k,k) + p(i,j);
                G(k,k1) = G(k,k1) - p(i,j);
            end
            if i > 1 % nie jest to pierwszy wiersz, więc jest  sąsiad powyżej
                k1 = n * (j - 1) + i - 1;
                G(k,k) = G(k,k) + q(i-1,j);
                G(k,k1) = G(k,k1) - q(i-1,j);
            end
```

```
            if i < n % nie jest to ostatni wiersz, więc jest sąsiad powyżej
                k1 = n * (j - 1) + i + 1;
                G(k,k) = G(k,k) + q(i,j);
                G(k,k1) = G(k,k1) - q(i,j);
            end
        end
    end

    % Wektor prądów b. Wszystkie niepodłączone węzły mają prądy zero (tyle samo
    % wpływa co wypływa). Dwa węzły są podłączone do źródła prądowego.
    % UWAGA: oznaczenie b zamiast I, i, j itp. aby uniknąć nieporozumień takich
    % jak pomylenie z macierzą jednostkową i/lub liczbami zespolonymi.
    %
    node_A = 1;
    node_B = nm;
    b = sparse(nm,1);
    b(node_A) =  1.0; % prąd o natężeniu  1 ampera
    b(node_B) = -1.0; % prąd o natężeniu -1 ampera

    % Modyfikujemy układ równań tak, aby ostatni węzeł miał potencjał zero.
    % Ok, moglibyśmy po prostu zrobić pętlę for i = 1:(n-1), bo i tak wyliczony
    % w tej pętli wiersz macierzy G zostanie teraz nadpisany na nowo.
    %
    G(nm,:)  = 0;
    G(nm,nm) = 1;
    b(nm)    = 0;

    % Właściwe obliczenia - rozwiązanie układu równań.
    %
    v = G \ b;

    % Obliczenie oporności zastępczej. Ponieważ prąd z założenia wynosił 1 amper,
    % to do obliczenia oporu zastępczego dzielenie przez 1 może być pominięte.
    %
    equivalent_resistance = full(v(node_A) - v(node_B));
end
```