

```

% courtesy of Dominik Edelmann
%%
% zunächst eine Funktion, die den Index mit minimalem Eintrag in r sucht
function [ i ] = minIndex( r )
i = 1;
for j=2:length(r)
    if (r(j)<r(i))
        i=j;
    end
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%% Dann der Simplex, der auch die Kosten im K-ten Schritt ausgibt
function [ x,K ] = simplex_kosten( A,b,c,ecke )
K = zeros(1,1);
iter = 1;

x = ecke;
[m,n] = size(A);

% Dieser Vektor speichert sämtliche Vertauschungen, die während dem
% Simplexalgorithmus passieren, damit zum Schluss wieder zurückgetauscht
% wird.
indice = [1:n]';

% Komponententausch: Falls Startecke nicht so beschaffen ist, dass letzten
% n-m Komponenten 0 sind.
nonzeroPos = m+1;
freePos = 1;
while (nonzeroPos <= n) % bis zum Ende durchsuchen
    while (x(freePos,1) ~= 0)
        freePos = freePos + 1;
    end
    while (x(nonzeroPos,1) == 0)
        nonzeroPos = nonzeroPos+1;
        if (nonzeroPos > n)
            break
        end
    end
    if (freePos <= m && nonzeroPos <= n)
        % Nur dann wird Tausch durchgeführt.
        x(freePos,1) = x(nonzeroPos,1);
        x(nonzeroPos,1) = 0;

        buf = indice(freePos);
        indice(freePos) = indice(nonzeroPos);
        indice(nonzeroPos) = buf;

        buf = c(freePos,1);
        c(freePos,1) = c(nonzeroPos,1);
        c(nonzeroPos,1) = buf;

        buf = A(:,freePos);
        A(:,freePos) = A(:,nonzeroPos);
        A(:,nonzeroPos) = buf;
    end
end

% Starte Simplexschritte
while (true)
    B = A(:,1:m);
    N = A(:,(m+1):n);
    hv = B' \ c(1:m,1);
    K(1,iter) = c'*x;
    iter = iter+1;
end

```

```

r = (c((m+1):n,1)'-hv'*N)';
if (r >= 0)
    disp('Optimale Loesung gefunden');

    % Tausche zurück
    y = x;
    for l=1:n
        x(indice(l))=y(l);

    end
    return;
else
    i = minIndex(r);
    v = B\N(:,i);
    if (v <= 0)
        disp('Keine Loesung');
        return;
    else
        hv = (v > 0); % Enthält 1 genau da, wo v > 0 ist

        for k=1:m
            if (hv(k,1)==1)
                minVal = x(k,1)/v(k,1);
                index = k;
                break;
            end
        end

        for j=(index+1):m
            if (hv(j,1) == 1 && x(j,1)/v(j,1) < minVal)
                index = j;
                minVal = x(j,1)/v(j,1);
            end
        end

        j = index;

        x(1:m,1) =x(1:m,1)-minVal*v;
        x(j,1) = minVal;

        buf = A(:,j);
        A(:,j) = A(:,m+i);
        A(:,m+i) = buf;

        buf = c(j,1);
        c(j,1) = c(m+i,1);
        c(m+i,1) = buf;

        buf = indice(j);
        indice(j) = indice(m+i);
        indice(m+i) = buf;
    end
end
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Dann das Hauptprogramm fuer das Problem von Klee und Minty
for N=3:5
    fprintf('Starte Groesse %d mit Klick \n',N);
    pause;

```

```

% Zugehörige Matrix
A = eye(N);
for j=1:N
    for i=j+1:N
        A(i,j) = 2^(i-j+1);
    end
end
b = zeros(N,1);
for i=1:N
    b(i,1) = 5^i;
end
c = zeros(N,1);
for i=1:N
    c(i,1) = 2^(N-i);
end

% Habe Problem  $Ax \leq b$ ,  $x \geq 0$ ,  $c'x = \max$ . Definiere daher
B = eye(N);
B(:,(N+1):(2*N)) = A;
ecke = zeros(2*N,1);
for i=1:N
    ecke(i) = 5^i;
end
%b bleibt wie oben
new_c = zeros(2*N,1);
new_c((N+1):2*N,1) = -c;

[x,K]= simplex_kosten(B,b,new_c,ecke);
plot([1:length(K)],K);
y = x((N+1):2*N,1);
disp('Lösung des Problems:')
disp(y);
fprintf('Es wurden %d Iterationen benötigt \n',length(K));
disp('Matlab-Lösung')
z = linprog(-c,A,b,[],[],zeros(N,1));
disp(z);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%% Nochmal der Simplex, aber jetzt ohne die Kosten auszugeben
function [ x ] = simplex( A,b,c,ecke )
% Löst das Minimierungsproblem  $Ax=b$ ,  $x \geq 0$ ,  $c'x = \min!$  zu gegebener
% Startecke.

x = ecke;
[m,n] = size(A);

% Dieser Vektor speichert sämtliche Vertauschungen, die während dem
% Simplexalgorithmus passieren, damit zum Schluss wieder zurückgetauscht
% wird.
indice = [1:n]';

% Komponententausch: Falls Startecke nicht so beschaffen ist, dass letzten
% n-m Komponenten 0 sind.
nonzeroPos = m+1;
freePos = 1;
while (nonzeroPos <= n) % bis zum Ende durchsuchen
    while (x(freePos,1) ~= 0)
        freePos = freePos + 1;
    end
    while (x(nonzeroPos,1) == 0)
        nonzeroPos = nonzeroPos+1;
        if (nonzeroPos > n)
            break
        end
    end
end

```

```

end
if (freePos <= m && nonzeroPos <= n)
    % Nur dann wird Tausch durchgeführt.
    x(freePos,1) = x(nonzeroPos,1);
    x(nonzeroPos,1) = 0;

    buf = indice(freePos);
    indice(freePos) = indice(nonzeroPos);
    indice(nonzeroPos) = buf;

    buf = c(freePos,1);
    c(freePos,1) = c(nonzeroPos,1);
    c(nonzeroPos,1) = buf;

    buf = A(:,freePos);
    A(:,freePos) = A(:,nonzeroPos);
    A(:,nonzeroPos) = buf;
end
end

% Starte Simplexschritte
while (true)
    B = A(:,1:m);
    N = A(:,(m+1):n);
    hv = B' \ c(1:m,1);
    r = (c((m+1):n,1)' - hv'*N)';
    if (r >= 0)
        disp('Optimale Loesung gefunden');

        % Tausche zurück
        y = x;
        for l=1:n
            x(indice(l))=y(l);
        end
        return;
    else
        i = minIndex(r);
        v = B\N(:,i);
        if (v <= 0)
            disp('Keine Loesung');
            return;
        else
            hv = (v > 0); % Enthält 1 genau da, wo v > 0 ist

            for k=1:m
                if (hv(k,1)==1)
                    minVal = x(k,1)/v(k,1);
                    index = k;
                    break;
                end
            end

            for j=(index+1):m
                if (hv(j,1) == 1 && x(j,1)/v(j,1) < minVal)
                    index = j;
                    minVal = x(j,1)/v(j,1);
                end
            end

            j = index;

            x(1:m,1) =x(1:m,1)-minVal*v;
            x(j,1) = minVal;
        end
    end
end

```

```

    buf = A(:,j);
    A(:,j) = A(:,m+i);
    A(:,m+i) = buf;

    buf = c(j,1);
    c(j,1) = c(m+i,1);
    c(m+i,1) = buf;

    buf = indice(j);
    indice(j) = indice(m+i);
    indice(m+i) = buf;

```

```

end

```

```

end

```

```

end

```

```

end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

%

```

```

%% Dann das Programm, das die Startecke fuer ein Problem in Standardform sucht

```

```

function [ x ] = startecke( A,b )

```

```

[m,n] = size(A);

```

```

A(:,(n+1):(n+m)) = eye(m);

```

```

ecke = zeros(n+m,1);

```

```

ecke((n+1):(n+m),1) = b;

```

```

c = ones(n+m,1);

```

```

c(1:n,1) = zeros(n,1);

```

```

x = simplex(A,b,c,ecke);

```

```

x = x(1:n,1);

```

```

end

```