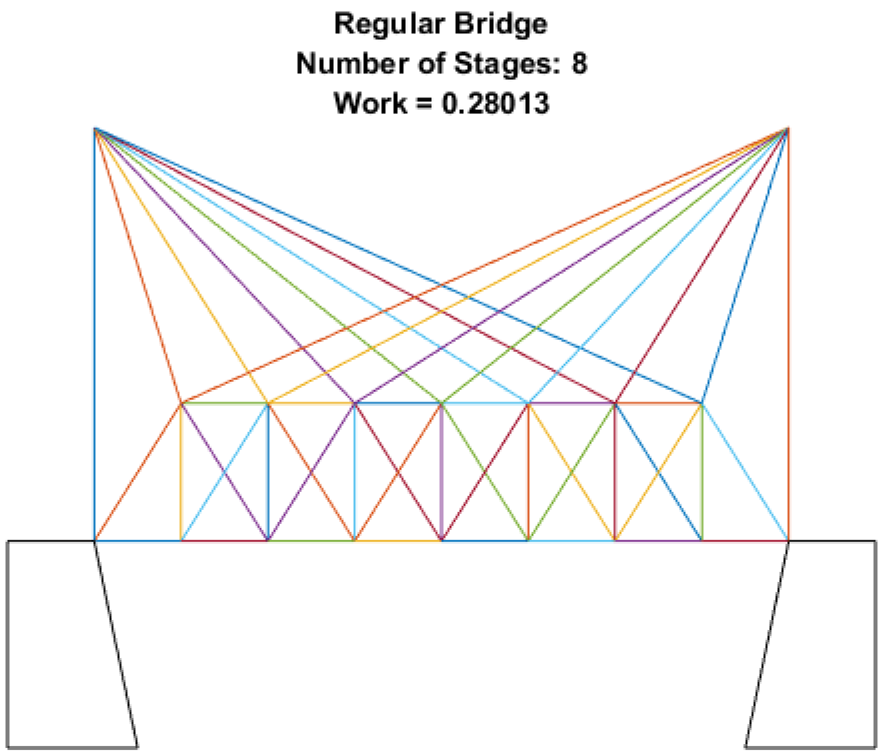
```
%Alison Felix de Araujo Maia, CAAM 210, Fall 2015, Lab 12
%bridge3.m
%Description: This function creates a bridge and simulates the optimal
 cross-sectional area for the minimum work done due a specific load.
%Example: bridge3(0.05, 8)
function bridge3(W, nos)
xc = zeros(7*nos-5,2);
yc = zeros(7*nos-5,2);
%The first 2 and last 3 fibers of initial bridge plus the first and
 last fiber of the additional part must be hardcoded
xc([1 2],[1 2]) = [0 1; 0 1]; yc([1 2],[1 2]) = [0 0; 0 1];
xc([5*nos-7 5*nos-6 5*nos-5],[1 2]) = [nos-1 nos-1 ; nos-1 nos; nos-1
 nos];
yc([5*nos-7 5*nos-6 5*nos-5],[1 2]) = [0 1; 1 0; 0 0];
xc(5*nos-4,[1 2]) = [0 0]; yc(5*nos-4,[1 2]) = [0 3];
xc(end,[1 2]) = [nos nos]; yc(end,[1 2]) = [0 3];
for j = 1:nos-2 %for-loop that establishes the repeatable part of the
 initial bridge without any load
    xc(5*j-2,[1 2]) = [j j];
    xc(5*j-1,[1 2]) = [j j+1];
    xc(5*j,[1 2]) = [j j+1];
    xc(5*j+1,[1 2]) = [j j+1];
    xc(5*j+2,[1 2]) = [j j+1];
    yc(5*j-2,[1 2]) = [0 1];
    yc(5*j-1,[1 2]) = [1 0];
    yc(5*j,[1 2]) = [1 1];
    yc(5*j+1,[1 2]) = [0 1];
    yc(5*j+2,[1 2]) = [0 0];
end
for j = 1:nos-1 %for-loop that creates the repeatable additional part
 of the bridge
    xc(5*nos-4+j,[1 2]) = [0 j];
    xc(end-j,[1 2]) = [nos-j nos];
    yc(5*nos-4+j,[1 2]) = [3 1];
    yc(end-j,[1 2]) = [1 3];
end
A = zeros(7*nos-5,nos*4);
%The effect of the interaction between nodes and not horizontal
 nor vertical fibers is decomposed in x and y values, where x =
 cos(theta), y = sin(theta) and theta is the right angle between the
 fiber and horizontal axis
d = 1/sqrt(2);
%The entries corresponding to the first 2 and last 3 fibers from the
 first bridge and first and last of the new part in matrix A must be
 hardcoded
A(1,1) = -1; A(2,[3 4]) = [-d -d];
A(5*nos-7,[end-7 end-6 end-5 end-4]) = [0 1 0 -1];
A(5*nos-6,[end-5 end-4]) = [d -d];
A(5*nos-5,[end-7 end-6]) = [1 0];
A(5*nos-4,[end-3 end-2]) = [0 -1];
A(end,[end-1 end]) = [0 -1];
```

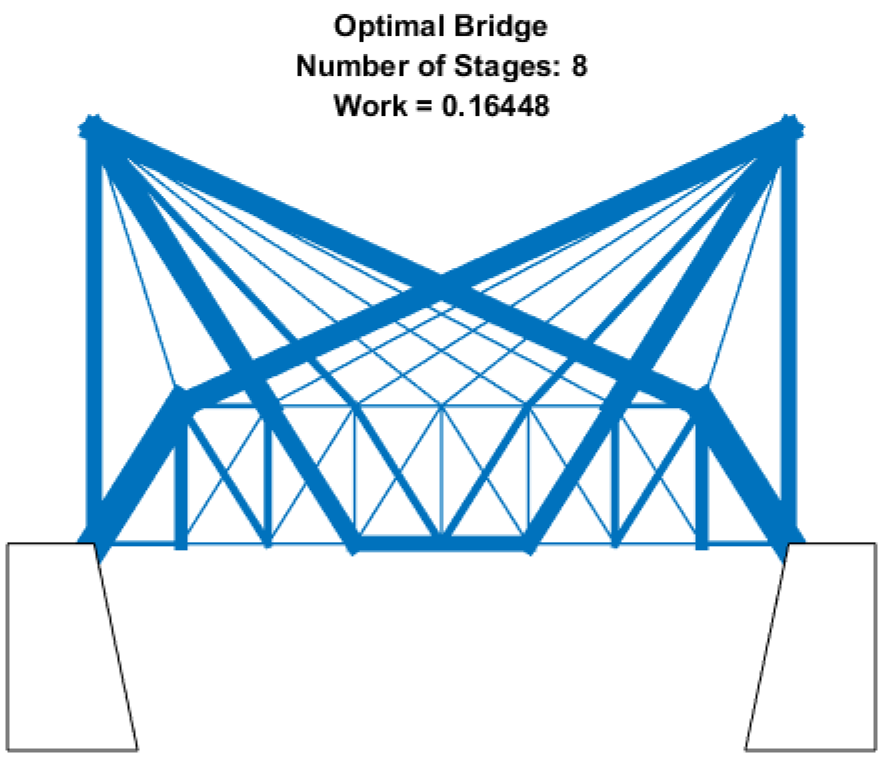
```
for j = 1:nos-2 %for-loop to fulfill the first part of adjacency
 matrix A
    A(5*j-2,[4*j-3 4*j-2 4*j-1 4*j]) = [0 1 0 -1];
    A(5*j-1,[4*j-1 4*j 4*j+1 4*j+2]) = [d -d -d d];
    A(5*j,[4*j-1 4*j 4*j+3 4*j+4]) = [1 0 -1 0];
    A(5*j+1,[4*j-3 4*j-2 4*j+3 4*j+4]) = [d d -d -d];
    A(5*j+2,[4*j-3 4*j-2 4*j+1 4*j+2]) = [1 0 -1 0];
end
for j = 1:nos-1 %for-loop to fulfill the new part of adjacency matrix
 A
    h = sqrt(j^2+4);
    A(5*nos-4+j,[4*j-1 4*j end-3 end-2]) = [-j/h 2/h j/h -2/h];
    A(end-j,[end-4*j-1 end-4*j end-1 end]) = [j/h 2/h -j/h -2/h];
end
%Definition of the length of each fiber
L = ones(1,7*nos-5);
L([2 5*nos-6]) = sqrt(2);
L([5*nos-4 end]) = 3;
for j = 1:nos-2
    L([5*j-1 5*j+1]) = sqrt(2);
end
for j = 1:nos-1
    h = sqrt(j^2+4);
    L([5*nos-4+j end-j]) = h;
end
%Definition of the fixed bridge volume and initial guess for fiber
 cross-sectional area
V = sum(L);
a0 = ones(1, 7*nos-5);
[work1, ~] = work(a0,A,L,W); %Work done on the regular bridge
figure(1)%Plot of regular bridge
line(xc',yc');
hold on
title({'Regular Bridge',['Number of Stages: ',num2str(nos)],['Work =
 ',num2str(work1)]});
fill([-1 0 0.5 -1],[0 0 -1.5 -1.5],'w');
fill([nos nos+1 nos+1 nos-0.5],[0 0 -1.5 -1.5],'w');
axis off
hold off
%Definition of lower and upper bounds on the permissible cross-
sectional area
LB = 0.1*ones(1, 7*nos-5);
UB = 10*ones(1, 7*nos-5);
options = optimset('gradobj','on'); %Switching on the gradient as
 input in fmincon
[optma,work2] = fmincon(@(a) work(a,A,L,W), a0, [], [], L, V, LB, UB,
 [], options); %fmincon will return the optimal cross-sectional area
 and the corresponding work done on the bridge
figure(2)%Plot of optimal bridge
for j = 1:7*nos-5
line(xc(j,:)',yc(j,:)','linewidth',ceil(3*optma(j)));
hold on
end
```

```
title({'Optimal Bridge',['Number of Stages: ',num2str(nos)],['Work =
 ',num2str(work2)]});
fill([-1 0 0.5 -1],[0 0 -1.5 -1.5],'w');
fill([nos nos+1 nos+1 nos-0.5],[0 0 -1.5 -1.5],'w');
axis off
hold off
end

%Function that applies gaussian elimination. For this project it will
%receive the adjacency matrix and the vector of forces applied to the
%bridge and return the vector with displacements of each node
function newpos = gauss(M,F)
n = length(F);
M(:,n+1)=F; %Augment M with F
for j = 1:n-1
    [v, r] = max(abs(M(j:n,j))); %Get the row with highest absolute
 value for the specified column
    r = r + j - 1;
    if v < eps
        disp('Warning! The pivot is too small.');
    end
    M([j r],:) = M([r j],:); %Swap row r and row j
    for k = j+1:n %for-loop that nullifies M(k,j)
        M(k,:) = M(k,:) + (-M(k,j)/M(j,j)) .* M(j,:);
    end
end
if M(n,n) < eps
    disp('Warning! The last number of the diagonal is too small.');
end
F = M(:,n+1); %Get the new values of F
newpos = trisolve(M,F);
end

%This function uses backsubstitution to solve the system M*(newpos) =
 F when M is triangular
function newpos = trisolve(M,F)
n = length(F);
newpos = zeros(n,1);
newpos(n) = F(n)/M(n,n);
for j = n-1:-1:1
    tmp = 0;
    for k = j+1:n
        tmp = tmp + M(j,k)*newpos(k);
    end
    newpos(j) = (F(j) - tmp) / M(j,j);
end
end

%Function that calculates the work done on the bridge and returns it
 together with the gradient of the compliance
function [val,grad] = work(a,A,L,W)
M = A'*diag(a./L)*A; %Stiffness matrix
F = zeros(1,size(A,2));
for j = 2:4:size(A,2)-4
```

```
    F(j) = -W; %Force applied on each node
end
x = gauss(M,F');
vwork = x.*F';
val = sum(vwork); %Total work done on the bridge
grad = -(A*x)'.^2./L; %Gradient of compliance
end
```

```
Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-
decreasing in
feasible directions, to within the default value of the function
 tolerance,
and constraints are satisfied to within the default value of the
 constraint tolerance.
```





Published with MATLAB® R2015a