Updates to the pressure code to enable systems with parallel channels

Astrid Boje

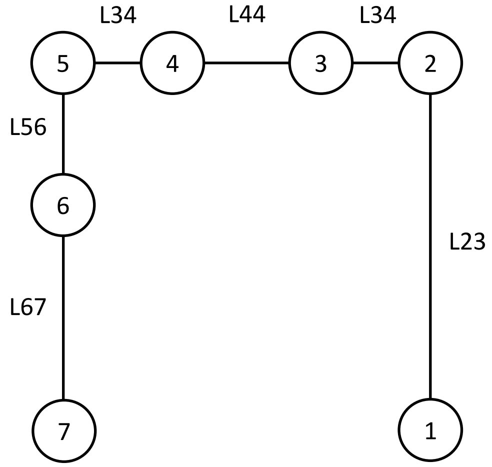
4 December 2020

Main modifications

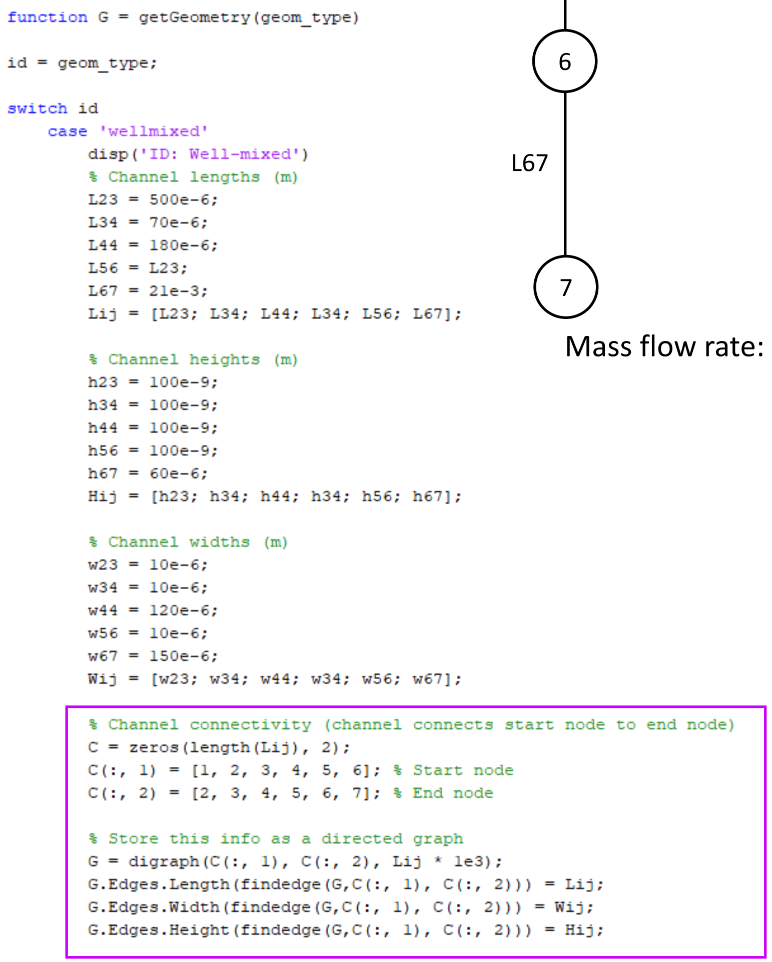
- Connectivity is specified in addition to dimensions in getGeometry function

- Graph tools are used to track connectivity and dimensions

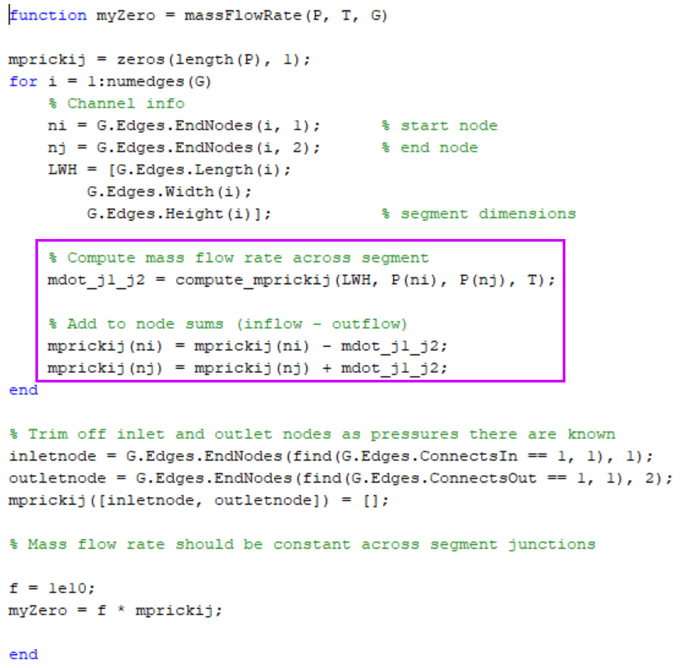
- Solves for consistent mass flow across (in/out) each node



Mass flow rate: 1.2e-12 kg/s

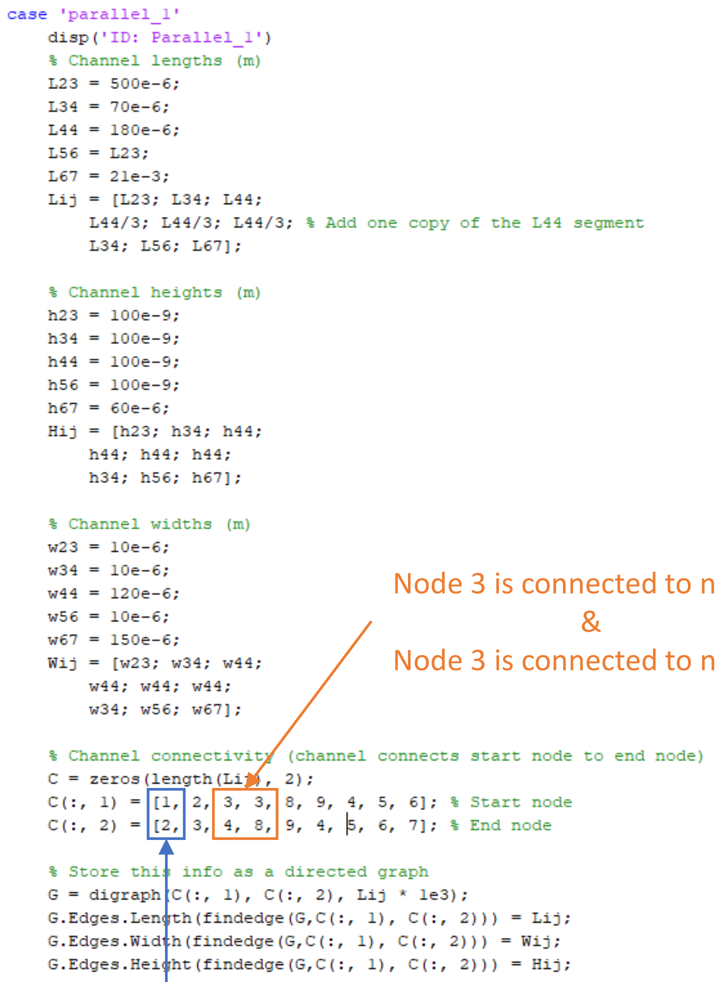


```
function G = getGeometry(geom_type)

id = geom_type;

switch id
    case 'wellmixed'
        disp('ID: Well-mixed')
        % Channel lengths (m)
        L23 = 500e-6;
        L34 = 70e-6;
        L44 = 180e-6;
        L56 = L23;
        L67 = 21e-3;
        Lij = [L23; L34; L44; L34; L56; L67];

        % Channel heights (m)
        h23 = 100e-9;
        h34 = 100e-9;
        h44 = 100e-9;
        h56 = 100e-9;
        h67 = 60e-6;
        Hij = [h23; h34; h44; h34; h56; h67];

        % Channel widths (m)
        w23 = 10e-6;
        w34 = 10e-6;
        w44 = 120e-6;
        w56 = 10e-6;
        w67 = 150e-6;
        Wij = [w23; w34; w44; w34; w56; w67];

        % Channel connectivity (channel connects start node to end node)
        C = zeros(length(Lij), 2);
        C(:, 1) = [1, 2, 3, 4, 5, 6]; % Start node
        C(:, 2) = [2, 3, 4, 5, 6, 7]; % End node

        % Store this info as a directed graph
        G = digraph(C(:, 1), C(:, 2), Lij * 1e3);
        G.Edges.Length(findedge(G,C(:, 1), C(:, 2))) = Lij;
        G.Edges.Width(findedge(G,C(:, 1), C(:, 2))) = Wij;
        G.Edges.Height(findedge(G,C(:, 1), C(:, 2))) = Hij;
```



```
function myZero = massFlowRate(P, T, G)

mprickij = zeros(length(P), 1);
for i = 1:numedges(G)
    % Channel info
    ni = G.Edges.EndNodes(i, 1);      % start node
    nj = G.Edges.EndNodes(i, 2);      % end node
    LWH = [G.Edges.Length(i);
        G.Edges.Width(i);
        G.Edges.Height(i)];           % segment dimensions

    % Compute mass flow rate across segment
    mdot_j1_j2 = compute_mprickij(LWH, P(ni), P(nj), T);

    % Add to node sums (inflow - outflow)
    mprickij(ni) = mprickij(ni) - mdot_j1_j2;
    mprickij(nj) = mprickij(nj) + mdot_j1_j2;
end

% Trim off inlet and outlet nodes as pressures there are known
inletnode = G.Edges.EndNodes(find(G.Edges.ConnectsIn == 1, 1), 1);
outletnode = G.Edges.EndNodes(find(G.Edges.ConnectsOut == 1, 1), 2);
mprickij([inletnode, outletnode]) = [];

% Mass flow rate should be constant across segment junctions

f = 1e10;
myZero = f * mprickij;

end
```

Eg1: One parallel channel

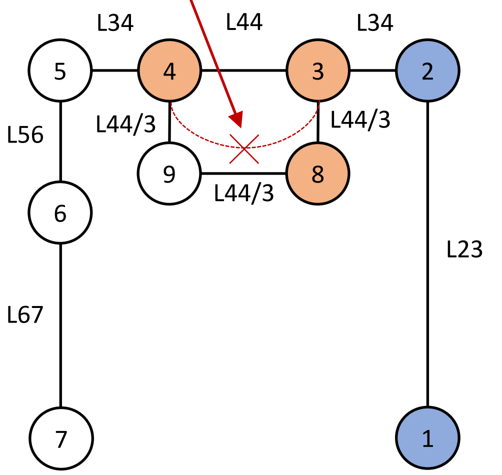


```
case 'parallel_1'
    disp('ID: Parallel_1')
    % Channel lengths (m)
    L23 = 500e-6;
    L34 = 70e-6;
    L44 = 180e-6;
    L56 = L23;
    L67 = 21e-3;
    Lij = [L23; L34; L44;
        L44/3; L44/3; L44/3; % Add one copy of the L44 segment
        L34; L56; L67];

    % Channel heights (m)
    h23 = 100e-9;
    h34 = 100e-9;
    h44 = 100e-9;
    h56 = 100e-9;
    h67 = 60e-6;
    Hij = [h23; h34; h44;
        h44; h44; h44;
        h34; h56; h67];

    % Channel widths (m)
    w23 = 10e-6;
    w34 = 10e-6;
    w44 = 120e-6;
    w56 = 10e-6;
    w67 = 150e-6;
    Wij = [w23; w34; w44;
        w44; w44; w44;
        w34; w56; w67];

    % Channel connectivity (channel connects start node to end node)
    C = zeros(length(Lij), 2);
    C(:, 1) = [1, 2, 3, 3, 8, 9, 4, 5, 6]; % Start node
    C(:, 2) = [2, 3, 4, 8, 9, 4, 5, 6, 7]; % End node

    % Store this info as a directed graph
    G = digraph(C(:, 1), C(:, 2), Lij * 1e3);
    G.Edges.Length(findedge(G,C(:, 1), C(:, 2))) = Lij;
    G.Edges.Width(findedge(G,C(:, 1), C(:, 2))) = Wij;
    G.Edges.Height(findedge(G,C(:, 1), C(:, 2))) = Hij;
```

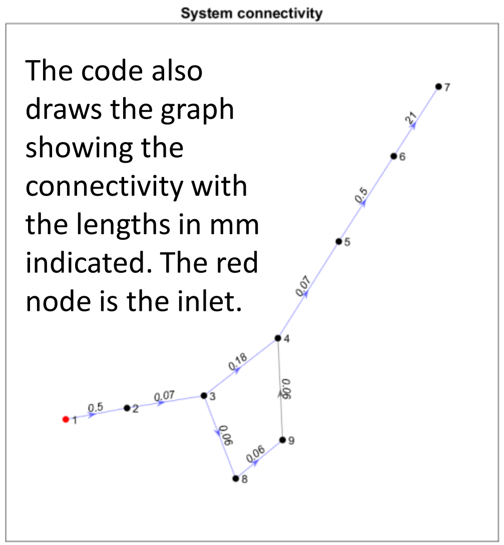
Node 3 is connected to node 4
&
Node 3 is connected to node 8

Node 1 is connected to node 2

NB: no two channels can share the same two nodes so there must be "right-angled" bends

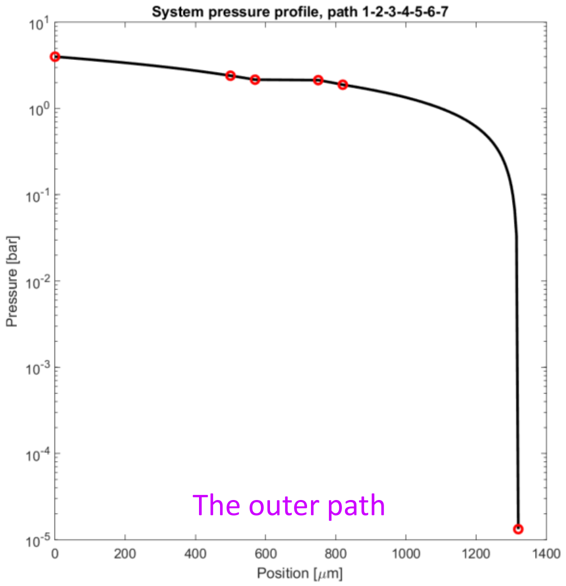


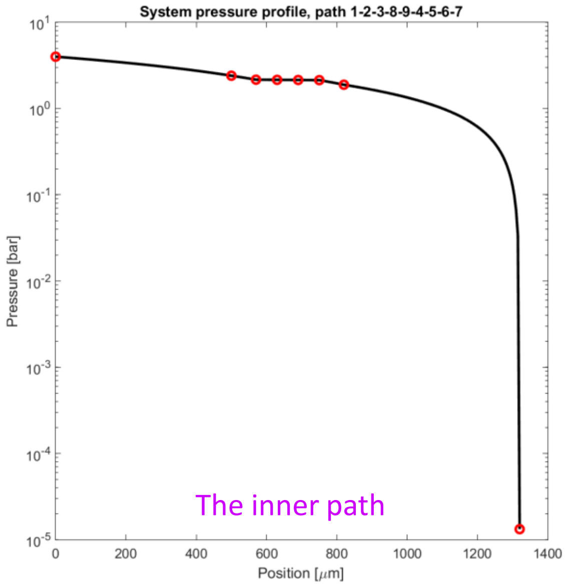
Mass flow rate: 1.2e-12 kg/s w 50/50 split in parallel part



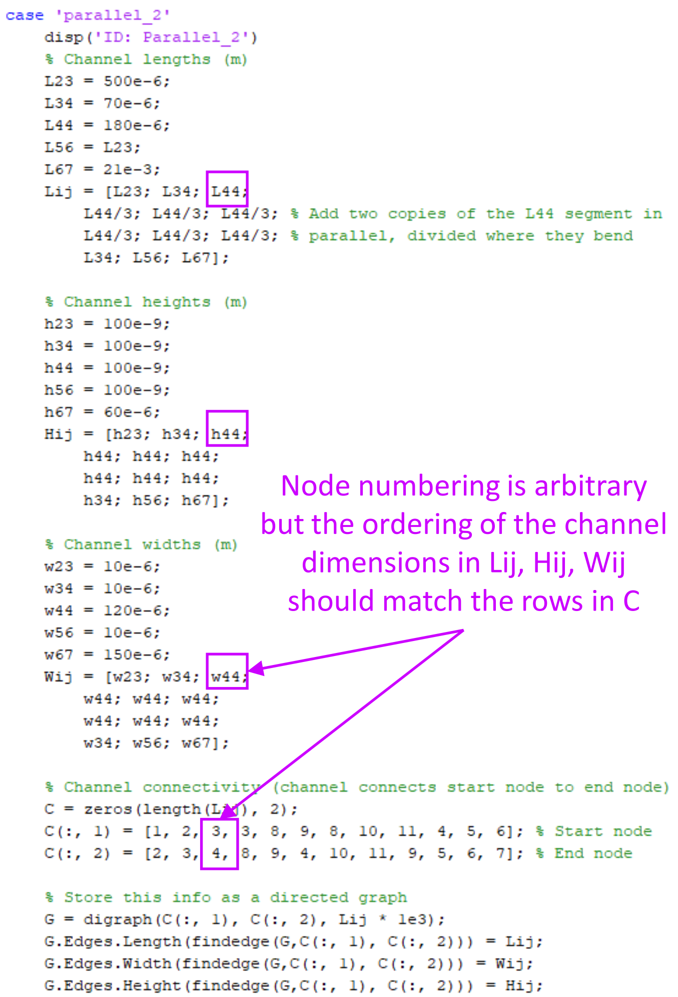
The code also draws the graph showing the connectivity with the lengths in mm indicated. The red node is the inlet.

Pressure profiles are plotted for each path through the system. Pressures at the junctions will be the same as both paths are solved together. Flow rates can be different depending on the channel dimensions



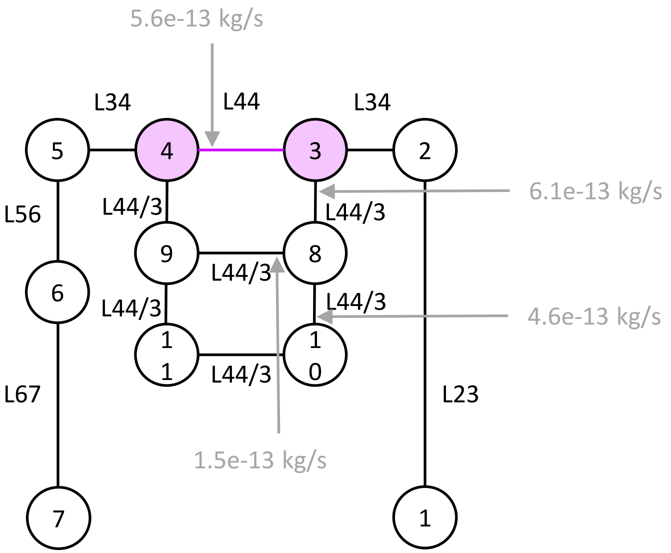


Eg2: Two parallel channels

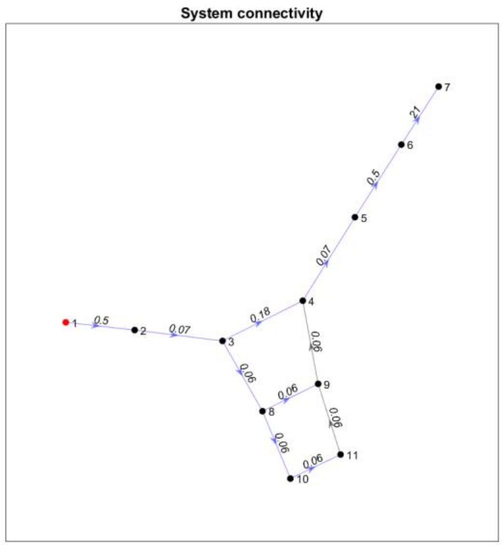


```
case 'parallel_2'
    disp('ID: Parallel_2')
    % Channel lengths (m)
    L23 = 500e-6;
    L34 = 70e-6;
    L44 = 180e-6;
    L56 = L23;
    L67 = 21e-3;
    Lij = [L23; L34; L44;
        L44/3; L44/3; L44/3; % Add two copies of the L44 segment in
        L44/3; L44/3; L44/3; % parallel, divided where they bend
        L34; L56; L67];

    % Channel heights (m)
    h23 = 100e-9;
    h34 = 100e-9;
    h44 = 100e-9;
    h56 = 100e-9;
    h67 = 60e-6;
    Hij = [h23; h34; h44;
        h44; h44; h44;
        h44; h44; h44;
        h34; h56; h67];

    % Channel widths (m)
    w23 = 10e-6;
    w34 = 10e-6;
    w44 = 120e-6;
    w56 = 10e-6;
    w67 = 150e-6;
    Wij = [w23; w34; w44;
        w44; w44; w44;
        w44; w44; w44;
        w34; w56; w67];

    % Channel connectivity (channel connects start node to end node)
    C = zeros(length(Lij), 2);
    C(:, 1) = [1, 2, 3, 3, 8, 9, 8, 10, 11, 4, 5, 6]; % Start node
    C(:, 2) = [2, 3, 4, 8, 9, 4, 10, 11, 9, 5, 6, 7]; % End node

    % Store this info as a directed graph
    G = digraph(C(:, 1), C(:, 2), Lij * 1e3);
    G.Edges.Length(findedge(G,C(:, 1), C(:, 2))) = Lij;
    G.Edges.Width(findedge(G,C(:, 1), C(:, 2))) = Wij;
    G.Edges.Height(findedge(G,C(:, 1), C(:, 2))) = Hij;
```

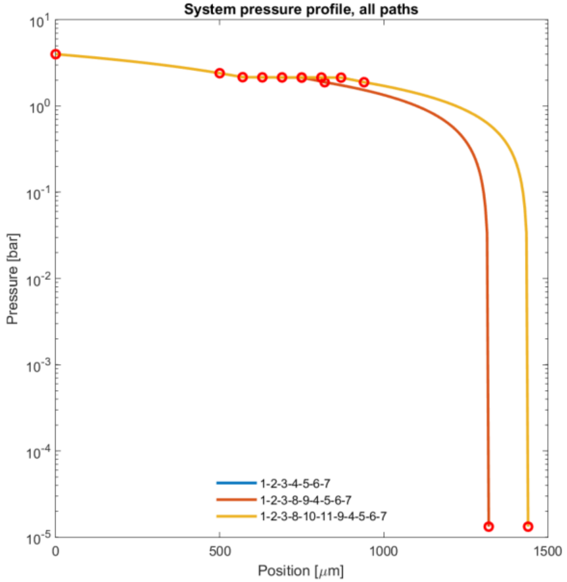
Node numbering is arbitrary but the ordering of the channel dimensions in Lij, Hij, Wij should match the rows in C



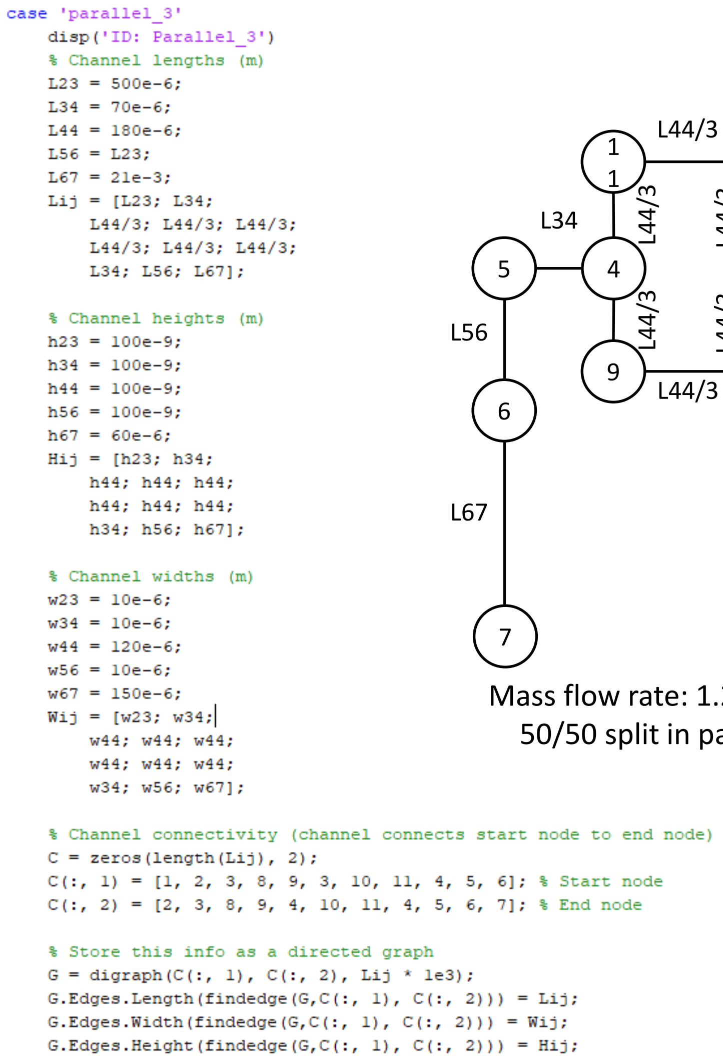
Mass flow rate: 1.2e-12 kg/s w split as indicated



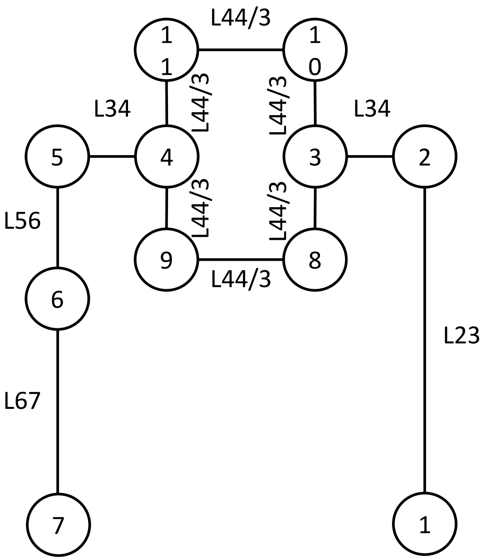
Plotting all profiles on the same axis shows that the outer path (connecting nodes 10 and 11) is longer



Eg3: Two parallel channels

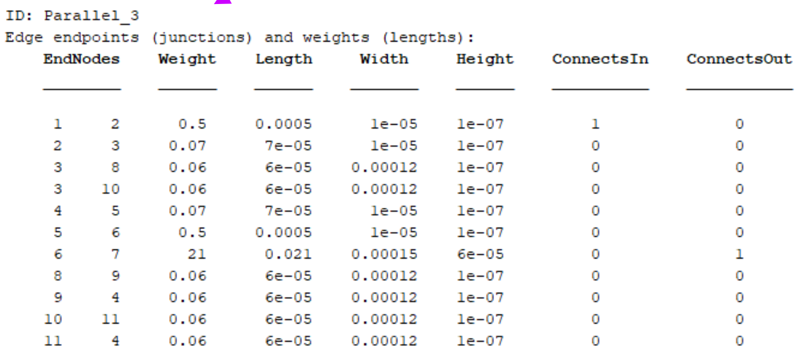


```
case 'parallel_3'
    disp('ID: Parallel_3')
    % Channel lengths (m)
    L23 = 500e-6;
    L34 = 70e-6;
    L44 = 180e-6;
    L56 = L23;
    L67 = 21e-3;
    Lij = [L23; L34;
        L44/3; L44/3; L44/3;
        L44/3; L44/3; L44/3;
        L34; L56; L67];

    % Channel heights (m)
    h23 = 100e-9;
    h34 = 100e-9;
    h44 = 100e-9;
    h56 = 100e-9;
    h67 = 60e-6;
    Hij = [h23; h34;
        h44; h44; h44;
        h44; h44; h44;
        h34; h56; h67];

    % Channel widths (m)
    w23 = 10e-6;
    w34 = 10e-6;
    w44 = 120e-6;
    w56 = 10e-6;
    w67 = 150e-6;
    Wij = [w23; w34;
        w44; w44; w44;
        w44; w44; w44;
        w34; w56; w67];

    % Channel connectivity (channel connects start node to end node)
    C = zeros(length(Lij), 2);
    C(:, 1) = [1, 2, 3, 8, 9, 3, 10, 11, 4, 5, 6]; % Start node
    C(:, 2) = [2, 3, 8, 9, 4, 10, 11, 4, 5, 6, 7]; % End node

    % Store this info as a directed graph
    G = digraph(C(:, 1), C(:, 2), Lij * 1e3);
    G.Edges.Length(findedge(G,C(:, 1), C(:, 2))) = Lij;
    G.Edges.Width(findedge(G,C(:, 1), C(:, 2))) = Wij;
    G.Edges.Height(findedge(G,C(:, 1), C(:, 2))) = Hij;
```

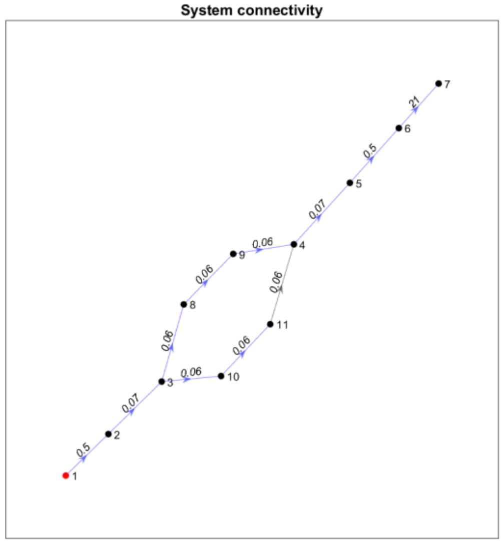


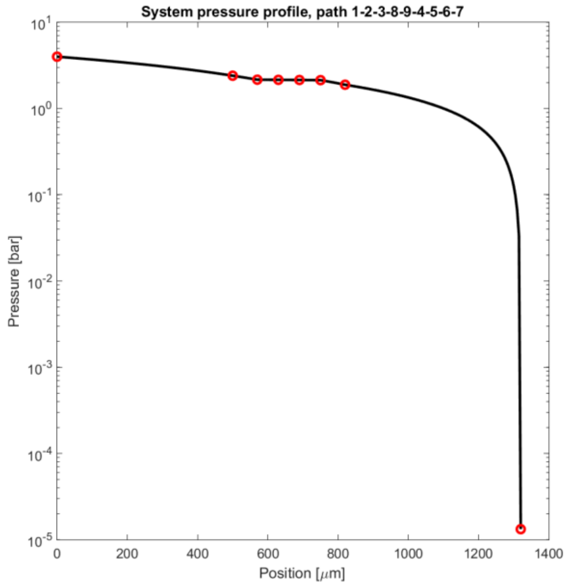
Mass flow rate: 1.2e-12 kg/s w 50/50 split in parallel part

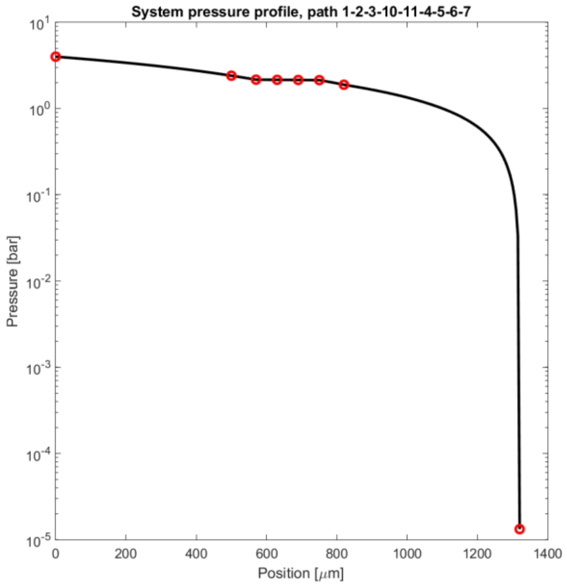
This table is printed out so you can check the dimensions are correct. Every row corresponds to one channel.



```
ID: Parallel_3
Edge endpoints (junctions) and weights (lengths):
    EndNodes    Weight    Length     Width     Height    ConnectsIn    ConnectsOut
    ________    ______    ______    _______    ______    __________    ___________

     1     2      0.5     0.0005      1e-05     1e-07        1              0
     2     3     0.07      7e-05      1e-05     1e-07        0              0
     3     8     0.06      6e-05    0.00012     1e-07        0              0
     3    10     0.06      6e-05    0.00012     1e-07        0              0
     4     5     0.07      7e-05      1e-05     1e-07        0              0
     5     6      0.5     0.0005      1e-05     1e-07        0              0
     6     7       21      0.021    0.00015     6e-05        0              1
     8     9     0.06      6e-05    0.00012     1e-07        0              0
     9     4     0.06      6e-05    0.00012     1e-07        0              0
    10    11     0.06      6e-05    0.00012     1e-07        0              0
    11     4     0.06      6e-05    0.00012     1e-07        0              0
```







Comments

Pressures

- It is important that the initial guess for the pressure profile is reasonable. This is more difficult than in the linear case, because the pressure needs to drop between successive nodes. I added a function called *setPressures* to interpolate between the inlet and outlet pressures across each path and then average for nodes that occur in multiple paths.
- Since the ordering of the nodes is arbitrary, the inlet and outlet nodes do not need to have the smallest and largest node numbers (see examples on the previous slides where I chose to number the outer path first and the outlet node is node 7/11). I added the function *reorder* to put the specified initial and final pressures in the correct places in the pressure vector to match their node numbers.

Numerical stability

- I have tested the examples in the previous slides, and I believe the code should also work for a wider range of configurations (e.g., all the channels don’t have to diverge and reconverge at the same pair of nodes). However, this assumption is untested so please let me know if it breaks under any configuration. I can imagine cases where it finds a negative mass flow or pressure solves the equations.
- I noticed in testing it that sometimes there is a tiny (order 1e-15) imaginary component in the pressures. I have clipped that off, but I added a warning in case it gets significant. Please let me know if you ever see that happening.