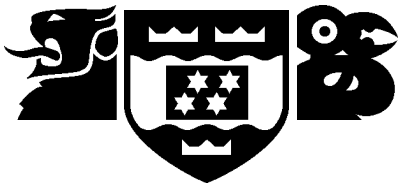
VICTORIA UNIVERSITY OF WELLINGTON



Department of Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 471 5328
Fax: +64 4 495 5232
Internet: Tech.Reports@comp.vuw.ac.nz

Simulating and animating fluid flow through a network of tubes

Peter Hall

Technical Report CS-TR-94/13
July 1994

Abstract

This paper introduces a method that simulates and animates fluid as it flows through a network of tubes, with arbitrary topology, in three dimensions. The simulation computes the flow through the network, given pressure functions at one or more places. This computation is based upon Poiseuille's equation for flow through a tube, and is fashioned to preserve volume. Animation interleaves clock-driven and event-driven mechanisms; clock-driven because flow is advanced from one frame to the next, event-driven because any significant events that occur between frames are detected, and acted upon. Primarily, these significant events occur when flow either converges or diverges at junctions of the network. Such events tend to mix the fluid as it flows. Animation correctly sequences such events. It is separate from the simulation calculations, and so provides a general mechanism. The method is illustrated via simulated x-rays of blood flow through vessels.

Author Information

Peter Hall is a lecturer at Victoria University. His principal research interests are scientific visualisation, and computer vision.

Simulating and animating fluid flow through a network of tubes

Peter Hall
Department of Computer Science
P.O. Box 600
Victoria University of Wellington
Wellington
New Zealand
Email: peter@comp.vuw.ac.nz

Abstract

This paper introduces a method that simulates and animates fluid as it flows through a network of tubes, with arbitrary topology, in three dimensions. The simulation computes the flow through the network, given pressure functions at one or more places. This computation is based upon Poiseuille's equation for flow through a tube, and is fashioned to preserve volume. Animation interleaves clock-driven and event-driven mechanisms; clock-driven because flow is advanced from one frame to the next, event-driven because any significant events that occur between frames are detected, and acted upon. Primarily, these significant events occur when flow either converges or diverges at junctions of the network. Such events tend to mix the fluid as it flows. Animation correctly sequences such events. It is separate from the simulation calculations, and so provides a general mechanism. The method is illustrated via simulated x-rays of blood flow through vessels.

1 Introduction

Simulating and animating fluid that flows through an arbitrary network of tubes is a challenging task. My motivation to address the problem was to simulate angiography, which is a particular kind of image acquisition technique used in medicine. In angiography, a dye that partially absorbs x-rays is injected into the blood stream. A time sequenced series of projection images is acquired as the dye flows through the blood vessels. Blood flow through a vessel network is a very complicated phenomena (Fung, 1984), so I made many simplifying assumptions. The simulation that is the result of these assumptions computes the laminar flow of incompressible fluid in an arbitrary network of rigid tubes; this is more like water in glassware than blood in vessels. However, the control mechanism used for animation is largely independent of the flow calculations themselves. Hence it provides a general mechanism and could find wider use in computer graphic or scientific visualisation contexts. Here, flow through vessels is used as an example application.

Animating fluid motion has been of interest to the graphics community in several ways. Ocean waves have been modelled and animated by Fournier and Reeves (1986), and by Peachey (1986). The primary concern of these researchers was to produce an animation that "looked good", so they made many simplifying assumptions. Their example in simplifying to obtain a solution that works in an acceptable time frame is followed here. While "looking good" is of concern here also, it is by no means the only one. In particular, the simulated fluid flow should be driven by pressure differences, and mass transport of dye particles must be accounted for. Kass and Miller (1990) solve a system of partial differential equations, and their solution accounts for mass transport, as

well as a variety of optical effects. However, the equations they use represent a height field and this makes them unsuitable for the current task. The effect of fluid flow in tubes is made visible by the presence of coloured particles, though here only the concentration of particles is of interest. Particle animation has been successful in computer graphics. For example, the gaseous surface of the planet Jupiter has been simulated and animated using particles (Yeager et al., 1986). More recently turbulent wind fields have been simulated and animated, made visible by the presence of objects such as leaves (Wejchert and Haumann, 1991), or directly as gas (Stam and Fiume, 1993). Particle animation has also found use in scientific visualisation, see Nagasawa and Kuwahara (1991), Briscolini and Santangelo (1991). None of these address the problem of animating fluid flow in a network.

Flow though a network of tubes have been investigated in the physical sciences, and the subject is discussed in some text books (Jeppson, 1977). Poiseuille's equation for flow through a tube is used as the basis of solution here. The equation states that the rate of flow, Q, through a tube is directly proportional to the pressure difference, Δp, that exists over the tube. That is

$$Q = C\Delta p \tag{1}$$

The quantity Q is a measure of the volume that passes through a cross-section of the tube, in unit time. The factor of proportionality, C, is called the conductance of the tube. For a fluid of constant viscosity, η, the conductance of a tube is directly computable from its geometry. For a straight tube of length L and constant radius, r, the conductance, is given by

$$C = \frac{\pi r^4}{8L\eta} \tag{2}$$

It is possible to derive Poiseuille's equation from the Navier-Stokes equation, see Fung (1984). Using Navier-Stokes equation would correctly be considered as more fundamental, but would add a level of complication to the system considered unnecessary here; Poiseuille's equation is sufficient. The principal assumptions made in deriving the Poiseuille equation are that tubes are rigid, straight, and of constant cross-section. The fluid is an incompressible Newtonian fluid, and its flow is purely laminar, there is no turbulent component. These simplifying assumptions have been made within the simulation.

Here, the fluid is partitioned into distinct regions. Adjacent regions are differentiated only by their optical properties, all other physical properties are assumed constant. A region with homogeneous optical properties will be called a bolus. As an example, distinct sections of water in a network may be dyed different colours. Each coloured section is a bolus, it is assumed that the dye does not alter density, viscosity, or any physical property apart from the fluid's reaction to light. The simulation also assumes that optical properties of interest can be derived from the concentration of dye, say, that is suspended in fluid. A bolus need not be visible, one that contains no optically active particles will not be.

Any boundary between adjacent bolii will be called a face. An assumption made is that each face provides a definitive partition between bolii; there is a discontinuity in concentration level at a face. Additionally, each face is considered to be flat. A parabolic face would better fit the velocity profile

for laminar flow, but a flat face is easier to deal with. In narrow tubes, this assumption make little difference to the images.

When starting a simulation, the faces may be given an initial position within the network, or they may be injected at specific locations. In the latter case the injection may last over several frames of animation. The simulation uses pressure values, given at certain well defined places in the network, to first compute a pressure distribution, and then the flow rate along each tube. By advecting faces, the fluid is made to appear as if it flows. Faces progress through the network from places of high pressure to places of low pressure. At junctions in the network flow will either converge or diverge. This mixes the flow and may lead to the creation of new faces, or the destruction of existing faces. Eventually, all faces will 'leak' from the ends of the network that are at low pressure. Such ends must exist because fluid cannot be forced to flow around a closed path - there can be no cycles in the flow.

Animation is responsible for producing a series of images in time sequence. It controls which faces are to be advected by the simulation, and through what time period they are to be advected. Even though animation and simulation may be separated logically, their mechanisms are interwoven. The mechanism to be described provides the potential for real-time animation. The animation halts after a given number of frames have been rendered, or when all faces have leaked out of the network, whichever is the sooner.

Results from the method are given, in section 4, using an x-ray simulation to affect a series of time sequenced angiograms. The paper concludes in section 5. Before then, the simulation and animation method is explained, in section 2, and mechanisms for rendering is briefly discussed in section 3.

2 Method

The method proposed consists of four sequential stages: first, a network of tubes must be constructed. A technique suitable for modelling is given in section 2.1 Second, the simulation should compute the rate of flow along each tube. For an incompressible fluid, these rates should such to preserve volume. Where fluid flow diverges or converges, changes in concentration should be computed. These calculations are the subject in section 2.2. Third, faces must be advected through the network, in a well defined order. The generation and destruction of faces must be controlled. An animation mechanism for this appears in section 2.3. Finally, the bolii in the network must be rendered to produce an animation frame.

If the pressure at a point changes with time then animation iterates over the last three of these stages, otherwise iteration is needed only over that last two. Each of the stages is now addressed in turn.

2.1 Modelling: tubes, pipes and junctions, networks, network points and faces

A tube is the basic primitive for modelling. Tubes have both geometric and physical properties. Tubes can be assembled end-to-end to form pipes. Alternatively, tubes may coincide at one end only, and so form a junction. A network is built from a collection of junctions and pipes; pipes connect junctions. The topological base for the network is an undirected graph. The simulation

makes use of the ends of tubes in junction, calling them network points, and faces exist with tubes of the network. These elements combine to form an animate model.

Tubes

The geometry of a tube is represented by a generalised cylinder. Generalised cylinders have been used in computer graphics contexts (Foley et al. 1990), and in computer vision contexts (Ballard and Brown, 1982). Bronsvoort and Klok (1985) define a generalised cylinder **G**, by its trajectory, **t**(u), and boundary (c_x(v), c_y(v)). The expression they give is

$$\mathbf{G}(u, v) = \mathbf{t}(u) + c_x(v)\,\mathbf{e}_2(u) + c_y(v)\,\mathbf{e}_3(u) \tag{3}$$

Where $\mathbf{e}_1$ denotes the unit vector that is tangential to **t**, which is the medial axis of the generalised cylinder. Unit vector $\mathbf{e}_1$ lies in the plane of instantaneous curvature of **t**. Unit vector $\mathbf{e}_2$ is perpendicular to $\mathbf{e}_1$ and also rests in the plane of instantaneous curvature of **t**. The unit vector $\mathbf{e}_3$ is the cross product of $\mathbf{e}_1$ and $\mathbf{e}_2$. Together, the vectors $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ form a Frenet frame. The functions c_x and c_y describe a contour in the plane defined by $\mathbf{e}_2$, and $\mathbf{e}_3$. This contour represents the cross section, and must be closed. The definition requires two parameters, u and v. The first of these specifies a particular position on the trajectory, the second specifies a particular position on the boundary. Tube geometry is depicted in figure 1, below.

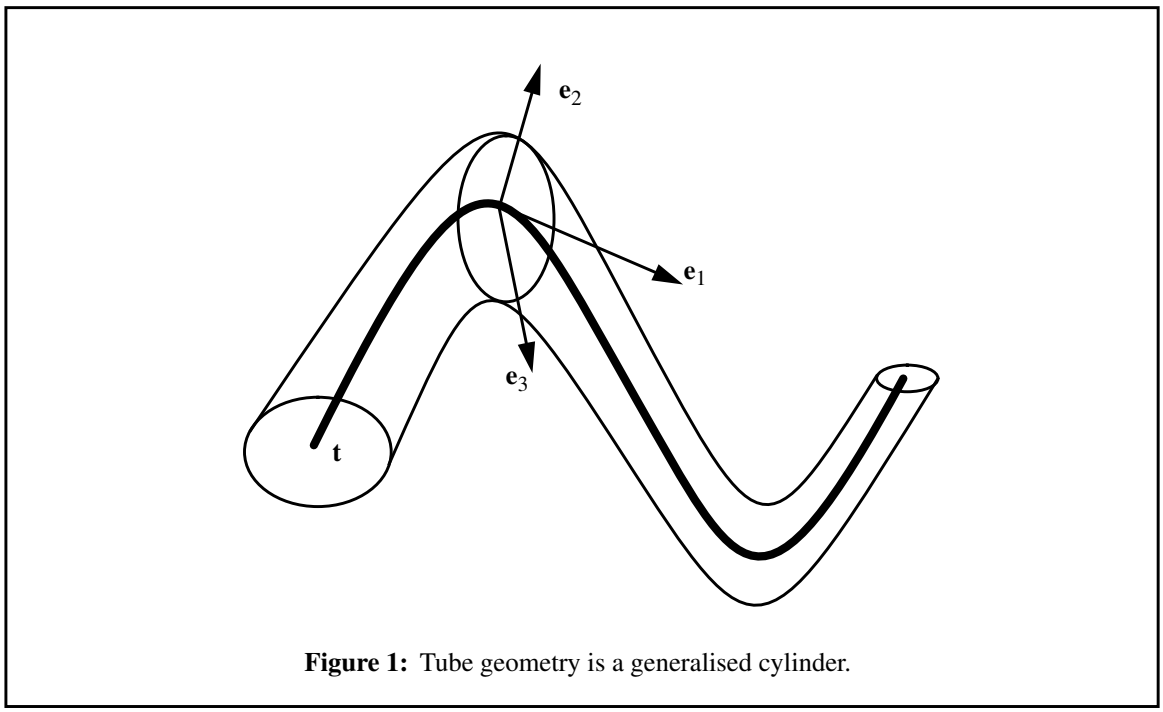


Figure 1: Tube geometry is a generalised cylinder.

For the purposes of this paper a circular cross-section is assumed, so that

$$\begin{aligned} c_x(v) &= r\sin(v) \\ c_y(v) &= r\cos(v) \end{aligned} \tag{4}$$

where r is the radius of the cross-section. Parameter v is an angle, measured in radians, in the range [0 , 2π). Typically, radius value is interpolated over the medial axis of the generalised cylinder. Thus

$$r = r_0 (1\text{-}f)+ r_1 f \tag{5}$$

where r_0 and r_1 are the radius value at the 'start' and 'end' of the trajectory, respectively, and f is a number in the range [0,1]. For reasons discussed below we define

$$f = \frac{s}{L} \tag{6}$$

where s is physical distance along the trajectory and L is the length of the trajectory.

Computing the length of a tube and computing the length of a trajectory $\mathbf{t}(u)$ are identical operations. To compute the length, L, of a trajectory a 'unit length', D, is chosen. Starting from one end, the trajectory is tracked until a unit length has been covered. The trajectory is then tracked again, this time from the end of the previous track. In this way, the whole trajectory is divided into N equal length sections, plus an 'end' section whose length, a, may be less than the unit length. Then

$$L = ND + a \tag{7}$$

This length is, of course, an estimate. Errors are introduced because the tracking procedure in inaccurate; analytic solutions for tracking are usually unavailable and iterative approaches are used in their place. A simple method to improve the estimate is to iteratively compute L at successively smaller values of D. When two estimates are sufficiently close, the length estimate can halt.

The mechanism just described does not produce an optimal decomposition of the trajectory in the sense that the minimum number of straight lines are used to describe it, Ihm and Naylor (1991) provide an algorithm for this. However, its advantage is that it may be used to reparameterise the trajectory in terms of arc length, s. Reparamterisation is important when interpolating radii over tube lengths and when advecting bolii through them. If radii were interpolated using the parameter u, then the distance along the trajectory at which a particular radius value would appear would depend on the specification of the trajectory. This undesirable effect is eliminated when physical distance is used as the interpolating variable, as demonstrated in Hall (1994). The reason that reparamterisation is an asset when advecting bolii is made clear in section 2.2.

To reparameterise, a bijection between parameter, u, and distance, s, needs to be defined. The computation of length, just described, defines the bijection at particular points on the trajectory because it associates a unique parameter value, u_i, with a unique physical distance, s_i. There will be M = N+2 such points, unless a = 0, then there will be M = N+1 such points. At the i^{th}point there is an ordered pair $< u_i, s_i >$. The M pairs form an ordered set, Z;

$$Z = < < u_0 = 0, s_0 = 0>, ..., < u_i, s_i>, ..., < u_M = 1, s_M = L> > \tag{8}$$

If $< u_i, s_i>$ and $< u_{i+1}, s_{i+1}>$ are consecutive entries then either $s_{i+1} = s_i + D$, or $s_{i+1} = s_i + a$, it true. Interpolation is used to fill in gaps between table entries. For example, the mapping from physical

distance to parameter can be defined as follows

$$u(s) = \begin{cases} u_i & \text{if } s \in s_i \\ u(s_{i-1})\,(1-f) + u(s_i)\,f & \text{otherwise} \end{cases} \tag{9}$$

where $s_{i-1}$is the largest value in the ordered set $< s_0, ..., s_i, ..., s_M>$ that is smaller than s, s_i is the smallest value in $< s_0, ..., s_i, ..., s_M>$ that is larger than s, and $f = (s_i - s)/(s_i - s_{i-1})$. The inverse relation, s(u), can be defined in an analogous manner. Because the points are close, the errors introduced by interpolation can be expected to be small. Also, because they are distributed over the trajectory at regular intervals, the expectation is that the error will be evenly spread across the trajectory (these expectations have yet to be tested).

As mentioned, the radius of a tube in the model may vary. So, conductance cannot be directly computed from the expression given in equation 2. Rather, conductance is approximated by breaking the tube into a series of smaller tubes, each of constant radius. The resistance of tubes in series is given by the sum of individual resistances, resistance is the reciprocal of conductance. To ensure a good approximation to the conductance an iterative approach is used. The tube is broken into smaller tubes at two scales, one scale half of the other. Conductance is estimated at each scale. In general, the conductances computed this way will differ. This provides the basis for an iteration that halts when the estimates are pleasingly close. Notice that conductance need be computed only once per simulation - unless the radius of the tube, the length of the tube, or the viscosity of the fluid changes. As a simplifying assumption we take it that the conductance of a tube is a constant of the simulation. Notice too that the conductance estimated is for a straight tube, curvature is not taken into account.

In summary, a single tube, T, in the network is defined as the set

$$T = \{\ \mathbf{t}(), r_0, r_1, L, Z, C\ \} \tag{10}$$

where $\mathbf{t}()$ is the trajectory of its medial axis, r_0, r_1 are start and end radii respectively, L is the length of the trajectory, Z is the ordered set associating distance, s, with parameter, u, and C is the conductance of the tube. We now turn attention to combining tubes to form pipes and junctions.

Pipes and junctions

It is usual in model building to construct complicated curves by concatenating simpler curves, the final curve being piecewise smooth. Often, the simpler curves are parametric cubics, and this approach is followed here. A pipe is formed from a sequence of tubes that abut end-to-end. Zeroth and first order continuity conditions are enforced upon the trajectories of the component tubes. The radii of abutting tubes must match. The component tubes of a pipe do not overlap. More formally, a pipe, P, is an ordered set of one or more tubes, $T_i = \{\ \mathbf{t}_i(), r_{i0}, r_{i1}, L_i, Z_i, C_i\ \}$, written

$$P = < T_1, ..., T_n > \tag{11}$$

such that $\mathbf{t}_i(L_i) = \mathbf{t}_{i+1}(0)$, $\mathbf{e}_i(L_i) = \mathbf{e}_{(i+1)}(0)$, $r_{i1} = r_{(i+1)0}$, and $\|T_i \cap T_j\| = 0$ for all i and j (provided i

and j are different); where $\mathbf{e}_i$ is the unit tangent to $\mathbf{t}_i$ at any point, $\|A\|$ is the cardinality of set A, and n is the number of tubes. Pipes may be regarded as an arrangement of tubes in series. The total resistance of a pipe may be estimated by summing the resistances of its component tubes. Alternatively, conductance may be directly computed in a manner similar to that for individual tubes.

A junction, also, is a collection of tubes. However, the trajectories of the tubes exhibit zeroth and first order continuity at one end only. The radius of cross section at the point that is common to all tubes in the set. More formally, a junction, J, is a set of one or more tubes, $T_i = \{ \mathbf{t}_i(), r_{i0}, r_{i1}, L_i, Z_i, C_i \}$;

$$J = \{ T_1, ..., T_n \} \tag{12}$$

such that $\mathbf{t}_i(0) = \mathbf{t}_1(0)$, $\mathbf{e}_i(0) = \mathbf{e}_1(0)$, $r_{i0} = r_{10}$, for all i; and $\|T_i \cap T_j\| \geq 0$ for all i, j. That end of a junction where tubes are coincident will be called the 'common base', the other end of each tube in the junction will be called a 'free end'. Junctions may be regarded an arrangement of tubes in parallel. However, tubes in a junctions will usually overlap to create a common volume. So flow along any tube of a junction is not independent of the flow along the other tubes in the junction. Consequently, any computation that is based on the tubes in a junction being a parallel arrangement of independent tubes will be subject to error. The magnitude of the error is related to the size of the common volume, as shown in section 7. In practice, the error was found to be negligible, so flow is treated as independent. A pipe, and a junction, are shown in figure 2.

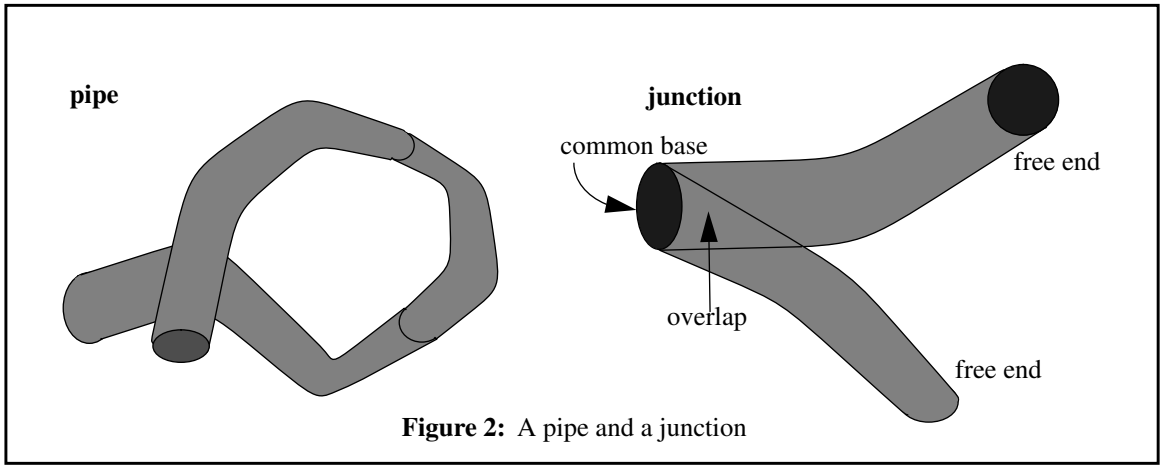


Figure 2: A pipe and a junction

Networks

A labelled graph, G, provides a convenient topological framework for an arbitrary network. We define G = (V, E), where V is a set of vertices, and E is a set of edges. Graph vertices are labelled by junctions, graph arcs are labelled by pipes. If {J} is a set of junctions then $f_V: V \rightarrow \{J\}$ is a vertex labelling, and if {P} is a set of pipes then $f_E: E \rightarrow \{P\}$ is an edge labelling. Typically, the labelling functions may be defined by forcing corresponding elements, in appropriate sets, to share a subscript. The model, Φ, can then be written as

$$\Phi = (G, \{J\}, \{P\}) \tag{13}$$

The graph an undirected graph. This is despite the fact that flow along any given tube has a specific direction. The reason for using an undirected graph, rather than a directed graph, is that the direction of flow depends only upon pressure differentials that exist over the network. These may change from one animation to the next, or in time for a particular animation. So the direction of flow along any particular tube cannot be defined without reference to the pressure gradient over it.

Every pipe must connect two junctions. The pipe will abut with either the common base of a junction, or one of its free ends. The continuity conditions enforced, between the tube belonging to the junction and that belonging to the pipe, are exactly the conditions given for tubes abutting in a pipe. This means that the network forms a continuous, smooth interior along which fluid can flow. Neither the common base, nor any of the free ends in a junction, need have an abutting pipe. Fluid will flow either into or out of the network at such places. An example of a network is seen in figure 3.

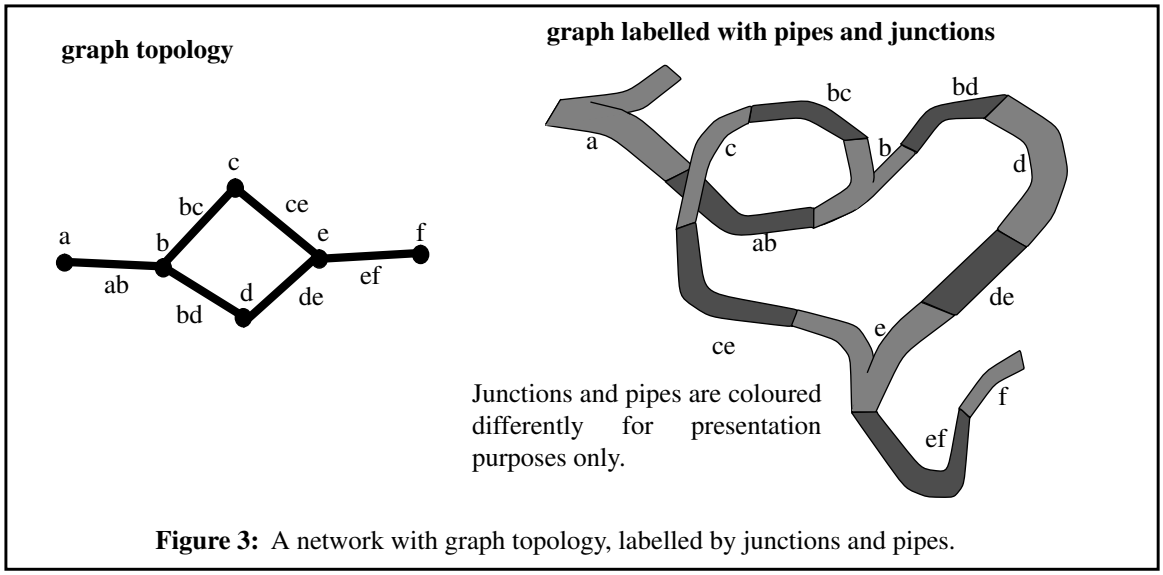


Figure 3: A network with graph topology, labelled by junctions and pipes.

This model is sufficient for static rendering, but not for animation. Two additional structures are useful there; network points, and faces.

Network points and faces

Fluid flows into or out of the network via tubes in junctions, and pipes abut with tubes in a junction. So, the ends of tubes in junctions are very important places, to both the simulation and the animation. These places are characterised by the points at the centre of the cross-section. Such points will be called network points, and will be referred to extensively in the remainder of the text. Network points can be uniquely identified by reference to the junction, J_{in}, they are in and the junction that the abutting pipe connects it to, J_{con}. If there is no abutting pipe then some arbitrary, but unique, identifier, J_{arb}, say, must replace the reference to second junction. Instantaneous pressure, p, and concentration as a function of time, $\kappa(t)$, is recorded in a network point. There will be one such function, $\kappa_i(t)$, for each of the tubes that abuts at the point, Π. Hence

$$\Pi = \{ < J_{in}, J_{uniq}>, p, \{ \kappa_i(t) \} \} \tag{14}$$

where J_{uniq} is either J_{con} or J_{arb}. It may seem that having multiple concentration functions in a network point leads to an ambiguous definition of concentration at the point. This would be true, but the concentration functions are not defined at the same point, rather they are defined pleasingly close to the point without ever touching it, and inside the particular tube the correspond to. Hence they represent concentration at distinct points, see figure 4, below. The reason that a distinct $\kappa_i(t)$ is needed for each tube that abuts at network point is so that flow in each tube of the junction can be considered independently. This simplifies the animation control mechanism.

A face, F, separates a pair of bolii, and is defined by

$$F = \{ T, s, t, dt, \kappa_1, \kappa_2 \} \tag{15}$$

in which T is the tube that the face resides in, s is the physical position of the face in the tube, t is a time instant, dt is a time period, and κ_1, κ_2 are concentration of dye, say, on each side of the face. A face is depicted in figure 4. Concentration, κ, is the number, M, of dye particles per unit volume, V, thus

$$\kappa = M / V \tag{16}$$

The position, time instant, and time period in a face are used during simulation and animation, their meaning will be explained below.

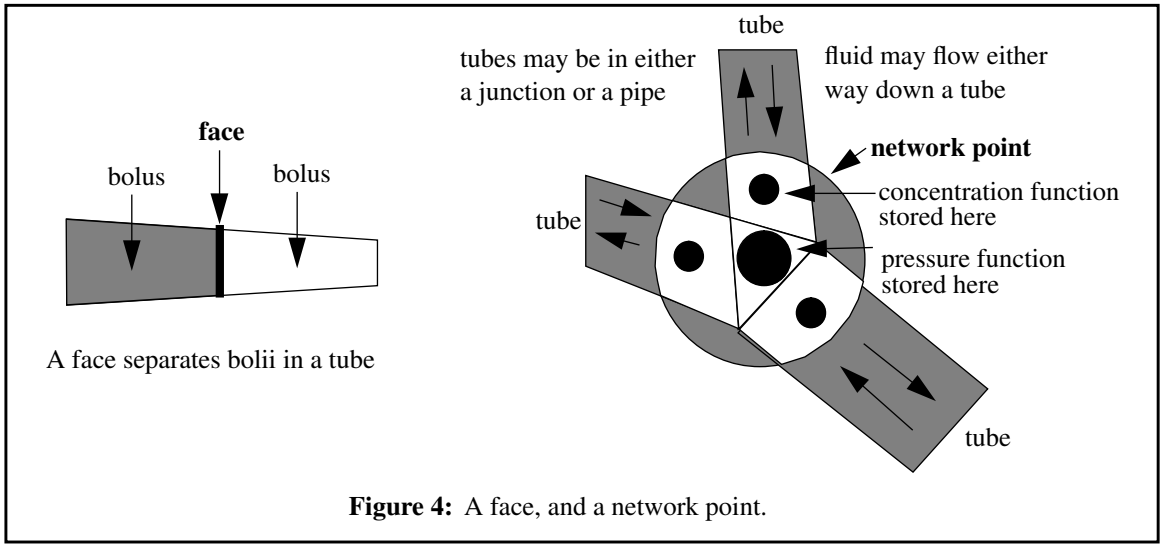


Figure 4: A face, and a network point.

2.2 Simulation: pressure distribution and flow rates, advection and spills, concentration changes

The most useful places for defining pressure values are the network points. The reason for this is twofold: (1) the pressure over any tube or pipe is given directly, and (2) given pressure values at a few network points, the pressure at other network points is readily computed, to yield a pressure distribution. The given pressures may be functions of time, but only instantaneous pressure values are stored. Also, because the pressure is sampled at discrete time instances, temporal aliasing is likely. The simplest approach to this problem is to raise the sample rate. The given pressure

functions may be cyclic, transient, or a mixture of both. An assumption made is that any pressure changes that occur are instantaneously effective over the whole network.

Streeter and Wylie (1975) present three conditions that flow in a network must satisfy. These conditions are:

1. The algebraic sum of pressure drops around any closed path must be zero, (this means flow around a closed path is impossible).
2. Flow into any junction must equal flow out of the junction, (for rigid walled containers).
3. The Darcy-Weisbach equation, or equivalent, must be satisfied for each tube (energy losses must be accounted for).

The Darcy-Weisbach equation measures the energy lost from the flow into non-recoverable forms of energy, such as heat. The equation, given in Jeppson (1977), is

$$h_f = f\frac{L}{D}\frac{V^2}{2g} \tag{17}$$

In which L is tube length and D is its diameter, V is the fluid velocity, and g is acceleration due to gravity. f is a dimensionless friction factor. For laminar flow, this factor is depends only the inverse of the Reynolds number, R_e, and is given by $f = 64 / R_e$, see Jeppson (1977). For turbulent flow the form of the friction factor is considerably more complex and not of interest here, refer to texts such as Jeppson (1977) for details.

Accounting for energy losses when computing pressure distribution usually requires an iterative procedure to be used, such as the Hardy-Cross method (see Streeter and Wylie, 1975, or Jeppson, 1977). Here, these losses are disregarded and this leads to a method for computing the pressure distribution via a system of simultaneous linear equations. The calculation yields flow that obeys conditions (1) and (2) stated above, and is described next.

Pressure distribution and flow rates

Suppose there are N_Π network points, Π_i, in a structure, each with pressure value, p_i. The pressure values are ordered in a column vector, **P**. Pressure is given at n_1 of these points, where $n_1 > 0$, and we can write

$$p_i = h_i \tag{18}$$

At network points that have no abutting pipe, and no pressure explicitly given, the pressure value is assumed to be zero, so there $h_i = 0$. There will be n_2 such points, so $n = n_1 + n_2$ pressures are either given or assumed. At the remaining N_Π-n network points, appeal to conservation of volume gives the expression

$$\Sigma_j (p_j - p_i)C_{ij} = 0 \tag{19}$$

in which C_{ij} is the conductance of the tube or pipe that connects point i to point j, and Σ_j runs over all $j \in [1, N_\Pi]$. If there is no connecting pipe then $C_{ij} = 0$.

A suitable matrix to represent the above equations can be set up as follows. Define a row vector $\mathbf{C}_i$ for the i^{th} pressure point. The j^{th} component of this vector is C_{ij}. The i^{th} component of this vector is the negative sum of all its other components. For network points where pressure is unknown, the inner product

$$\mathbf{C}_i\mathbf{P} = 0 \tag{20}$$

is the expression for the conservation of volume at such points. For network points where pressure is given or assumed we set the i^{th} component of $\mathbf{C}_i$ to unity and all other components to zero. We then write

$$\mathbf{C}_i\mathbf{P} = h_i \tag{21}$$

There are N_Π such inner products. These can be assembled into a matrix product,

$$\mathbf{CP} = \mathbf{H} \tag{22}$$

in which the i^{th} row of the matrix $\mathbf{C}$ is just $\mathbf{C}_i$. This system of equations can be solved for $\mathbf{P}$ by inversion of $\mathbf{C}$,

$$\mathbf{P} = \mathbf{C}^{-1}\mathbf{H} \tag{23}$$

This yields a value for pressure at each network point. Notice that if the conductivity of each tube is a constant of the simulation, then matrix inversion needs to be carried out only once. Should pressures change with time then it is necessary to compute a new column $\mathbf{H}$ and its matrix product with $\mathbf{C}^{-1}$, as in equation 24.

Having computed the pressure distribution, the rate of flow along each tube can be computed via the Poiseuille equation, given in equation 1, section 1. Here, the expression has the form

$$Q_{ij} = C_{ij}(p_j - p_i) \tag{24}$$

where Q_{ij} is the average rate of flow along any tube that connects networks points Π_i and Π_j. If the network points are at either end of a pipe, the computed flow rate must be the flow rate in every tube of that pipe. Notice that this flow rate is positive if the flow is from Π_j to Π_i, and negative if flow is in the other direction. Typically, the flow rate can be stored with the tube model, after a suitable modification to the definition of a tube. The flow rate along each tube needs to be computed each time the vector of pressures is computed.

Advection and spills

The simulation advects each face in the network, according to the flow rate of the tube it is in. Recall that the tubes are permitted to taper. This makes advecting a face by forward integration a very difficult task. Instead, preservation of volume is used directly, as follows. The volume of fluid, $V_{required}$, that a face sweeps in a tube in a time, dt, is just

$$V_{required} = Qdt \tag{25}$$

where Q is average rate of flow. For a tube that tapers linearly the volume captured between positions s_{old} and s_{new} is

$$V_{captured} = \pi\left(r^2 + \frac{(\Delta r)^2}{3}\right)|s_{new} - s_{old}| \tag{26}$$

where r is the minimum radius at the old at new positions, and Δr is the absolute change in radius. A solution to advection requires a value for s_{new} be computed. This is done by iterating positions of s_{new} until $V_{captured}$ is pleasingly close to $V_{required}$. Depending on the direction of flow, starting values for s_{new} will be either 0 or L, the length of the tube. Notice the use of physical distance in this formulation; implementing this is made easier by reparameterisation of the trajectory (see section 2.1).

This same computation can be used to determine whether a spill has occurred. A spill occurs whenever an advected face exits the tube it is currently in. If the initial estimate of $V_{captured}$ is less than $V_{required}$ then there is no solution; the face has spilled from the tube. The time taken, δt, to reach the spill is computable. This is just

$$\delta t = \frac{V_{captured}}{Q} \tag{27}$$

this time period is used by the animation control strategy.

Concentration changes

Spills are most interesting at network points that correspond to the common base in some junction. It is at such points that concentration may change. Appeal to preservation of number of dye particles leads yields the governing equation for concentration change;

$$\kappa_{out} = \frac{\sum_{in} |\kappa_{in} Q_{in}|}{\sum_{out} |Q_{out}|} \tag{28}$$

κ_{out} is the concentration in all tubes whose flow is directed away from the spill point; $\Sigma_{out} |Q_{out}|$ is the sum of the outward flow rates. κ_{in} is the concentration in a tube whose flow, rate Q_{in}, is directed in to the spill point, Σ_{in} sums over all input tubes. All of the terms in this equation have a time dependency. Computation of flow rates that change in time has already been discussed, but computation of concentration as a function of time has not. The simulation records the concentration behind the face at the time of its spill, at the place of the spill, in the tube it spilled from. This record shows concentration as a function of time, making computation of κ_{out} possible, and is one reason why a set of such functions are held in a network point. These concentration functions are constructed as the animation progresses. Animation control is discussed next.

2.3 Animation: three lists, two cycles, one strategy

Animating fluid flow means advecting the faces of the flow. The bolii appear to move because the

faces move. At junctions, faces may split to form two or more new faces, or merge together to form a single face. This splitting and merging of faces may be accompanied by concentration changes. Such events need to be properly scheduled, and animation frames must be captured at regular time intervals. Controlling the simulation is the task of the animation. In this sense, animation in not separate from the simulation. However, animation uses a control strategy that is independent of any calculation made by the simulation, and in that sense is separate. This strategy makes use of three lists, and two cycles, as explained next.

Three lists

The animation strategy uses three lists. These will be called the 'active list', the 'part-done' list, and the 'done list'. The active list is a list of faces that require advection. Faces on the active list are ordered by time, earlier faces first. The done list is a list of faces whose advection is complete, for a given frame. The part-done list is a list of faces whose advection is only partially complete, for a given frame. The advection of a face may be partially complete if the face spills from its current junction or pipe. Spills from one tube to another within the same pipe are not handled by the control strategy being outlined, rather, the simulation continues to advect the face in its new tube.

To begin the animation, an active list is constructed. One way to do this is simply to specify the faces on the list; that is supply an initial state for the network. In such a case, the time period of each face will be set to be the time interval between frames. Also, in such a case the time location, t, of each face is set to zero, say. Another way to generate the list is to simulate injection of fluid. To inject is rather more difficult, and is discussed below. For the immediate discussion, we assume that a set of faces exist within the network, and that these faces are recorded on the active list.

Animation control uses two cycles that move faces back and forth between lists. The basic function of these cycles will now be outlined.

Two cycles

One of the cycles accounts for the clock events needed to regulate animation. During this cycle faces are transferred from the active list to the done list. Transfer occurs only after they have been advected without spill. This phase of the cycle stops when there are no more faces on the active list, the simulation has then been moved forward from one frame to the next. If the done list is empty as well, then all faces will have spilled out of the network, and the animation halts. Otherwise, all faces on the done list are in position for a frame of animation to be captured. Once a frame is captured, the done list is emptied into the active list, and the cycle begins again.

The second cycle handles spill events. During this cycle faces are removed form the active list, and faces appear on the part done list. However, the transfer is not one-to-one because a face that spills generates new faces, and it is these that appear on the part done list. In the reverse phase of the cycle, faces are removed from the part done list and merged into the active list, the time ordering of the active list is maintained. Care is taken to ensure that faces that appear in the same place of the network, at the same time, are not duplicated; this process destroys faces.

The overall animation interweaves this pair of cycles, with the clock-driven cycle being dominant.

Details regarding this interweaving are now given.

One strategy

The face at the front of the active list is removed. The time instant associated with that face is deemed to be the current clock time. An attempt is made to advect this face for the period, dt, associated with it. Advection may cease for two reasons: either the full time period has been consumed,; or the face has spilled. In the former case the face's time instant component is incremented by dt, its period set to the time between frames, and the face is placed on the done list. In the latter case the face's time constant is incremented by the time consumed to the spill, δt, and its time period decremented by the same. The time consumed is given in equation 28. Any new faces generated by the spill inherit these time values, and each face is placed on the part done list.

When a spill occurs at a network point two actions are triggered. The first of these is that the appropriate concentration function is updated. The appropriate concentration function is stored in the network point where the spill occurs, and corresponds to the tube that the original face was in. Updating means inserting a semi-infinite step function into the function. The edge of the this step function occurs at the time instant of the spill. Before then, the concentration function is unchanged, after then the concentration is taken to be the height of the step; which is the concentration value. This remains at a constant level, until the next change that arises from a spill. Which of the two concentration values stored in a face to use in this update depends on the direction of flow. The update should use the concentration function that lays behind the face, relative to the flow. This is illustrated in figure 5.

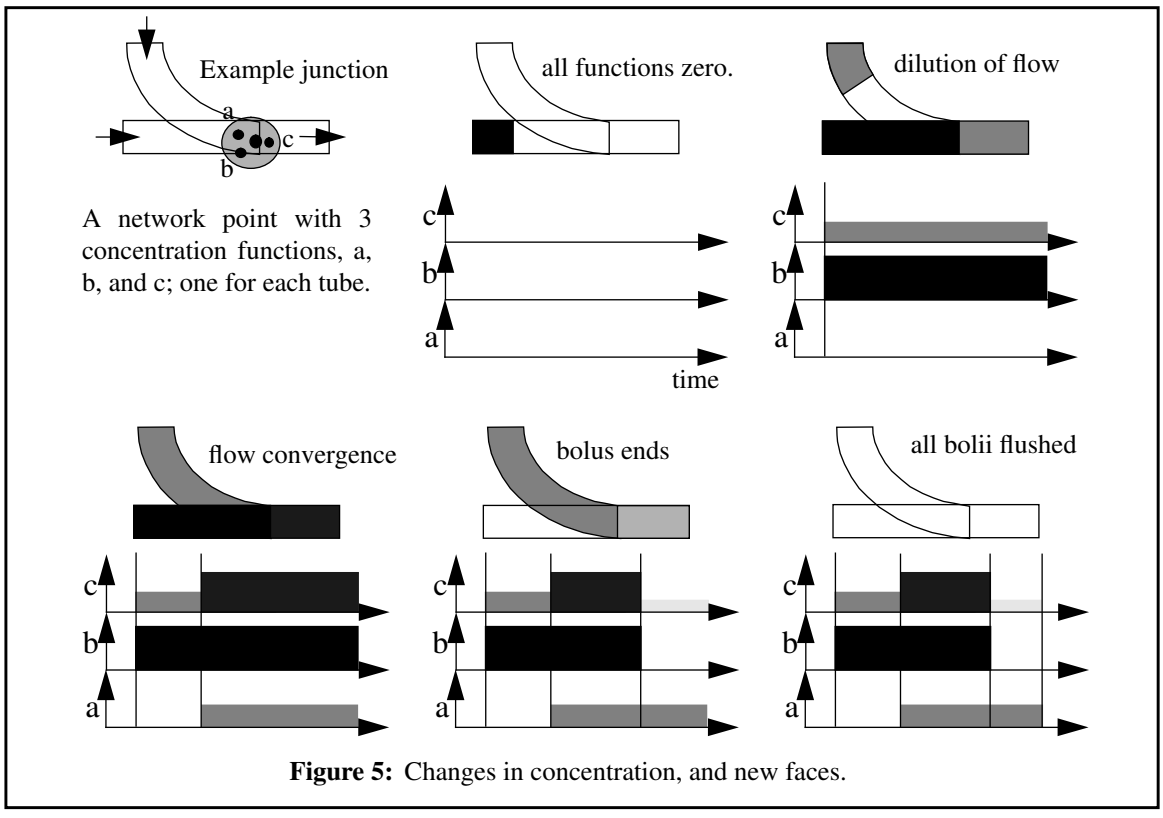


Figure 5: Changes in concentration, and new faces.

The second action triggered by a spill is the generation of new faces. For this, the local topology and flow of the network are examined, and a new face is created in each tube that flows away from the network point. These tubes may belong to either the junction, any abutting pipe, or both. If there are no such tubes, as at a leak, then no new faces are created. Clearly, each new face is created at the network point, but the concentration of these faces cannot be computed. This is because the concentration functions for tubes that flow into the network point may not be complete up to the time of the spill. However, the concentration of at a face can always be properly computed when it is removed from the active list. This is because all concentration functions will be complete up to current clock time, so that any relevant spill is a well defined historical event. This computation provides the reason why concentration functions are stored in network points.

A synopsis of the strategy is presented in figure 6.

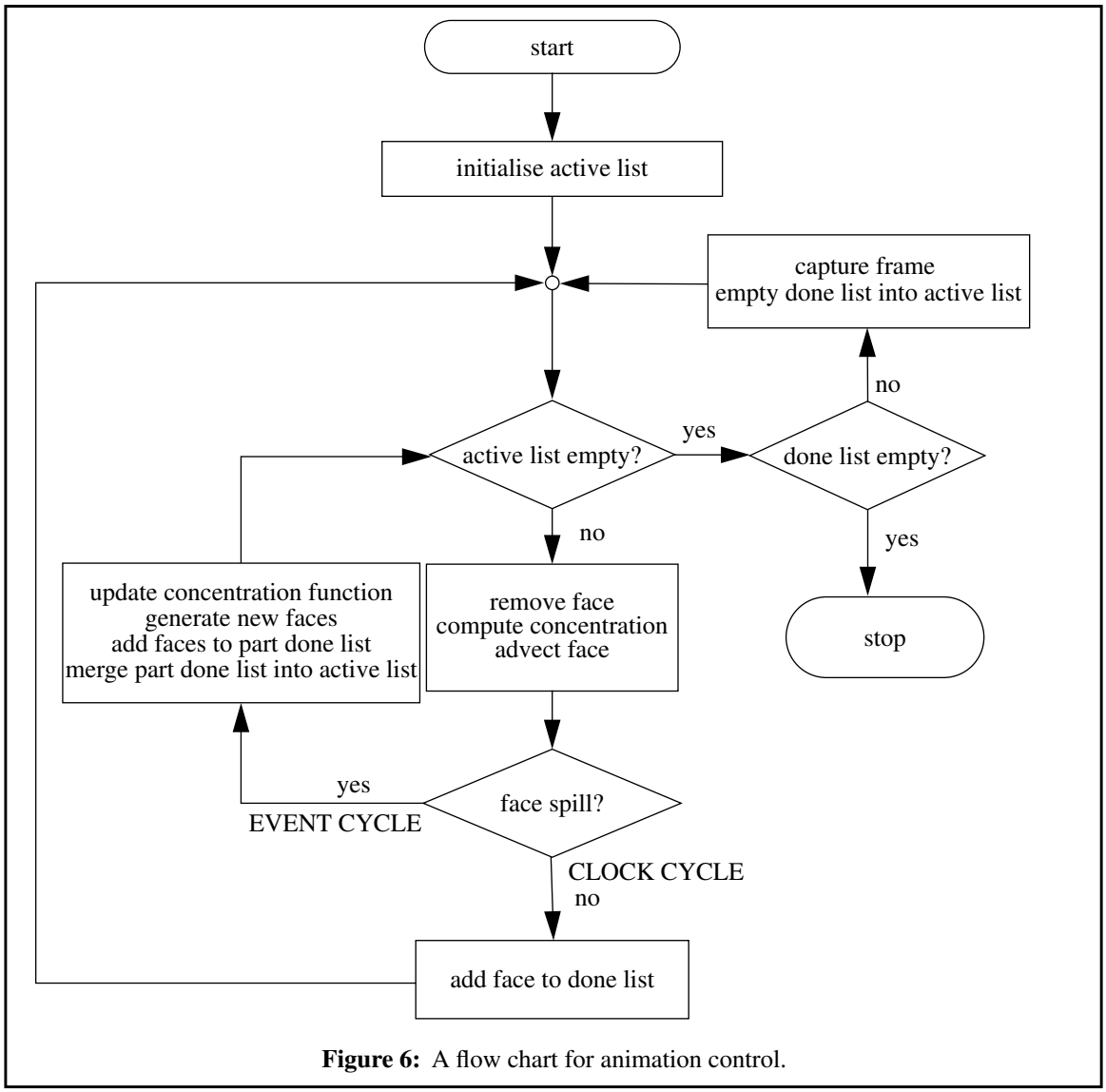


Figure 6: A flow chart for animation control.

Before turning to discuss rendering issues, it is pertinent to briefly explain injection. *Injection* is the simulation of a syringe injection, and is conveniently defined to take place at a network point.

The volume and concentration of dye injected must be given, this is the syringe. Any transitory pressure associated with the injection must be given also. Two sets of faces are released into the fluid, one to mark the start of the injection of dye, and one set to mark its completion.

Initially, the 'starting' faces are injected. The volume of dye that can be injected in the time leading to the next frame is computed. If this amount is less than the volume currently in the syringe then that amount is deducted from the syringe volume. This deduction of volume in the syringe continues at the start of each new frame until the amount of dye that needed to be injected is greater than the volume currently in the syringe. Then the time to the next frame must be divided into two parts, that for which dye is injected, and that for which it is not. The animation proceeds as normal over the first time period, the trailing faces are injected, and the animation proceeds once more. So, two complete iterations of the clock cycle are required before next frame is rendered.

3 Rendering a frame

Whenever a frame is to be captured, the distribution of dye throughout the network can be discerned from the faces on the done list and from the concentration functions built up in each tube at network points. The faces partition the network into regions of homogeneous concentration. The concentration functions are useful where a tube contains no face, the concentration within that tube is specified by the concentration values in the function; this is a second use for them. In a junction tubes will usually overlap. If the concentration function has different values in each tube at the common base then some ambiguity arises as to what the concentration should be. This can resolved by choosing either the highest concentration, or the concentration that flows away from the common base. If the projected image of the overlapping volume is small, this problem may be ignored.

The are many mechanisms that may be used for rendering the network. Ray tracing in an obvious method, Bronsvoort and Klok (1985) show how to ray trace generalised cylinders. However, ray tracing is an expensive option, and Bronsvoort (1992) provides a method for rendering by point sampling the surface of generalised cylinders. Hall (1994) provides a mechanism for rendering by point sampling the interior volume of tubes. It produces x-ray simulations, and will now be briefly described.

Each tube is partitioned into its component bolii. Each bolus is partitioned into small sections that are sufficiently straight to be regarded as a truncated cone, but one whose end planes are not perpendicular to its central axis. Each cone is taken in turn, and is sampled by points in three dimensional space at some arbitrary resolution. Care is taken to ensure that the sample points for all cones are elements of the same sample space. Any point found to be within the cone is tested against a list of previously rendered cones. If the test fails, the point is rendered, otherwise the point is not rendered. The cone is then added to the list of cones. The computation can be improved by restricting the list of cones that are searched to those in tubes that are known to overlap.

The simulated x-rays produce shadow-like images of tubes. The shadow is caused by the amount of material the x-ray has traversed. So, darker shadows are generated where tubes appear to cross. Depth of shadow is also a function of the angle of the tube to the image plane, the image of a given, straight, tube becomes darker as its trajectory approaches the perpendicular of the image plane. Also, wider tubes appear darker than narrower tubes, and all tubes appear darker in their middle

than at their edges.

The point sampling just described accumulates the effective amount of material that an x-ray passes through on the way from its source to the image plane. The ‘effective amount’ is the product of the actual amount and the absorption coefficient of the material. The absorption coefficient is directly proportional to the concentration of dye particles in the cone being sampled. To accumulate, each sample point is treated as a small but finite drop of absorbent material. This droplet is projected, and contributes to any pixels it may cover by addition. Only after all tubes have been rendered is this accumulated value used to compute a final intensity at each pixel; the negative of the accumulated value is the logarithm of the fraction of radiation that is transmitted.

The are two advantages of this method; first resolution is not restricted by available memory and, second, a point can be rendered simultaneously in more than one image with little extra overhead.

4 Results

The results presented have been obtained by injecting dye into a network, and simulating x-rays to capture animation frames. Pressures distributions, effects relating to changes in concentration at junctions, and how flow through only part of a network can be achieved, are all demonstrated

An animation is shown in figure 7, it demonstrates the change in concentration at a junction. The network is a parallel arrangement of pipes, each pipe acts as a delay line. Notice that there is no change in concentration where a single tube splits into many tubes. Where the pipes recombine there is a change in concentration - because of the delay. Fluid that carries dye at a high concentration mixes with fluid at a low concentration to yield a flow with some intermediate concentration. Later, when additional flows reach the junction the concentration in the output flow rises, only to fall again as bolii sweep through the junctions and out of the network.

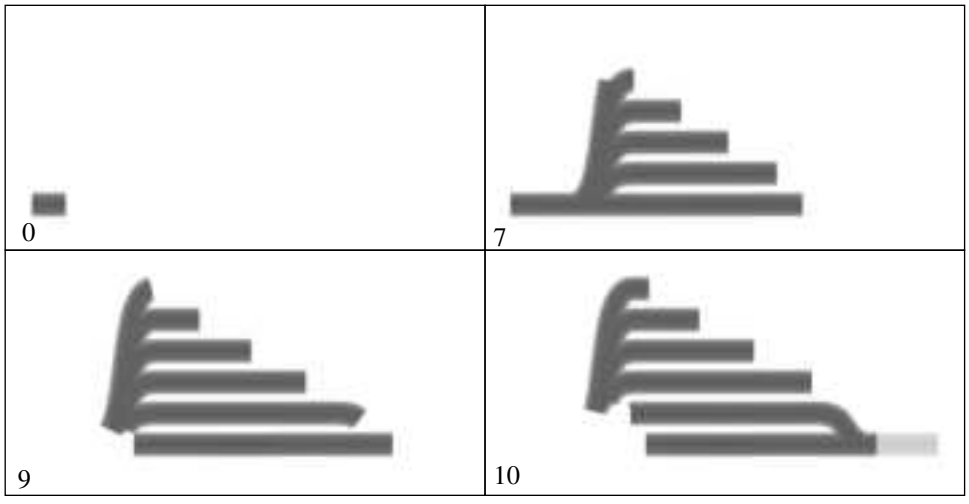


Figure 7: Animation showing change in concentration at a junction. Notice that the frames presented do not form a regular sequence in time. Rather, frames have been selected because they show significant event such as a change in concentration. Frame numbers are given. The figure continues...

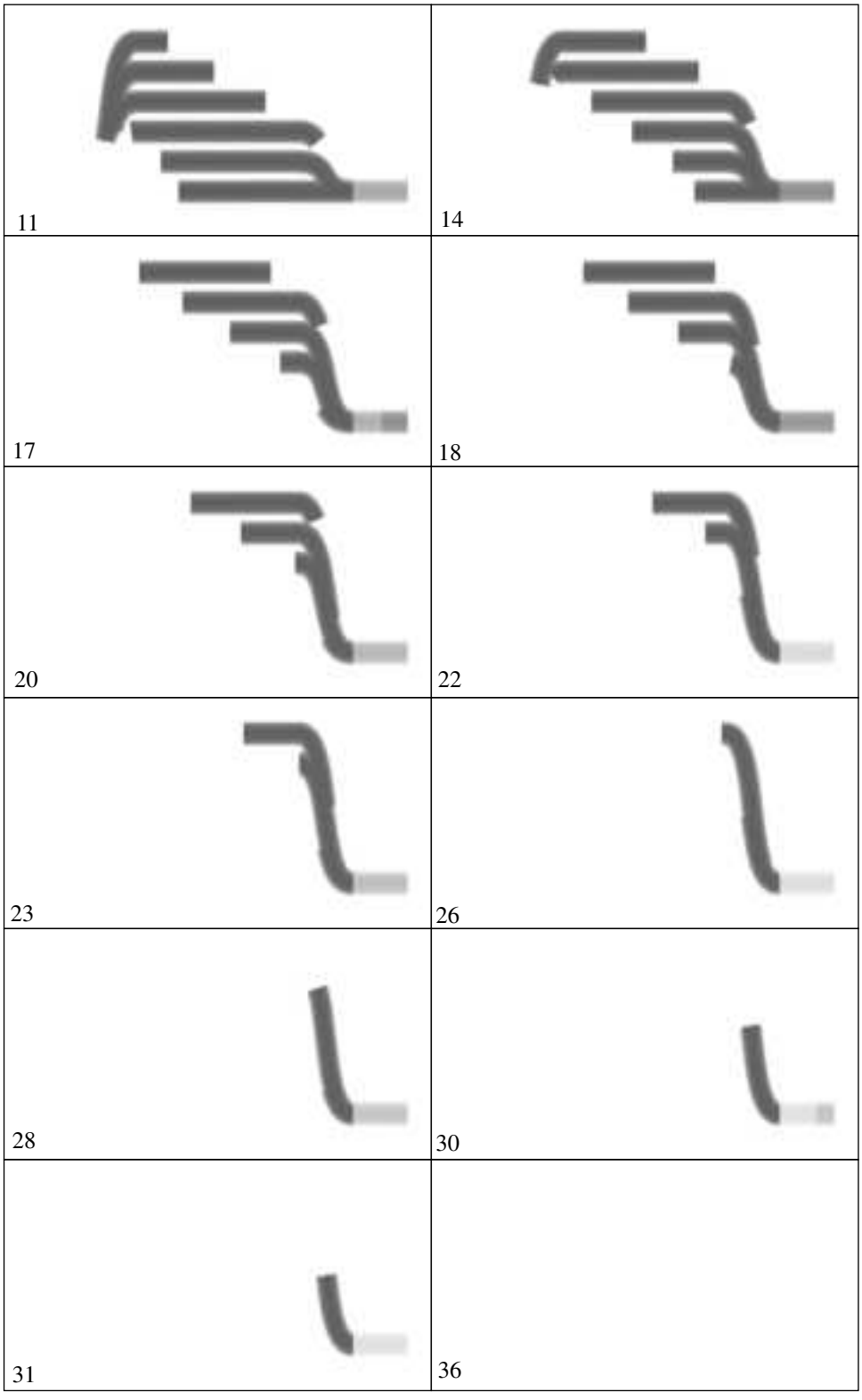
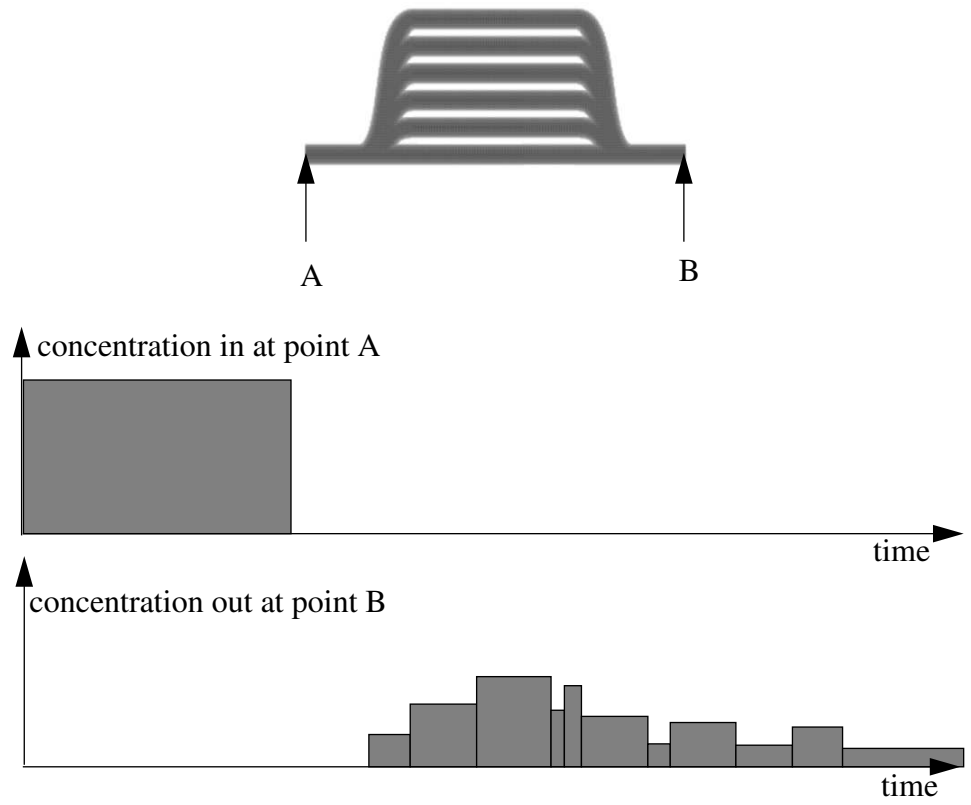


figure 7 - continued

Concentration as a function of time at the input point, and at the output point of this network is shown in figure 8. The general form of the output curve is that expected from a network of delay pipes. The area under these curves is the same, to within a few percent. This demonstrates conservation of number of dye particles, and hence of volume also.



The area under each curve is 19.09 +/- 0.25. This demonstrates preservation of number of dye particles, and of volume, to within a few percent.

Figure 8: Concentration as a function of time at input and output.

The general form of the output curve is that expected from empirical measurements (see Ruch and Patton, 1974, for example). There is a rapid rise to a peak as most of the dye moves rapidly through; followed by a longer decay that is produced by dye that has been delayed by the network. Usually, this function is continuous. Here, the function is step-like in nature because the faces form a definitive partition, and flow is laminar. Also, there are peaks and troughs in the tail that may be present in measurements from real systems because of re-circulation. This is not an adequate explanation here. Rather, these artefacts are caused by that part of the flow that arrives at the convergence point after previous bolii have left. These events can be witnessed in figure 7.

Pressure distribution, and flow rate distribution, can also be measured. These distributions are easiest to show on a test network that is symmetric, such that in figure 9. The network can be seen, with junctions and pipes clearly marked by colour. Pressure and flow rate are constants in any tube that intersects a line drawn vertically through the network. The radius of each tube in the network is a constant, but the total cross-sectional area rises, as shown in the figure.

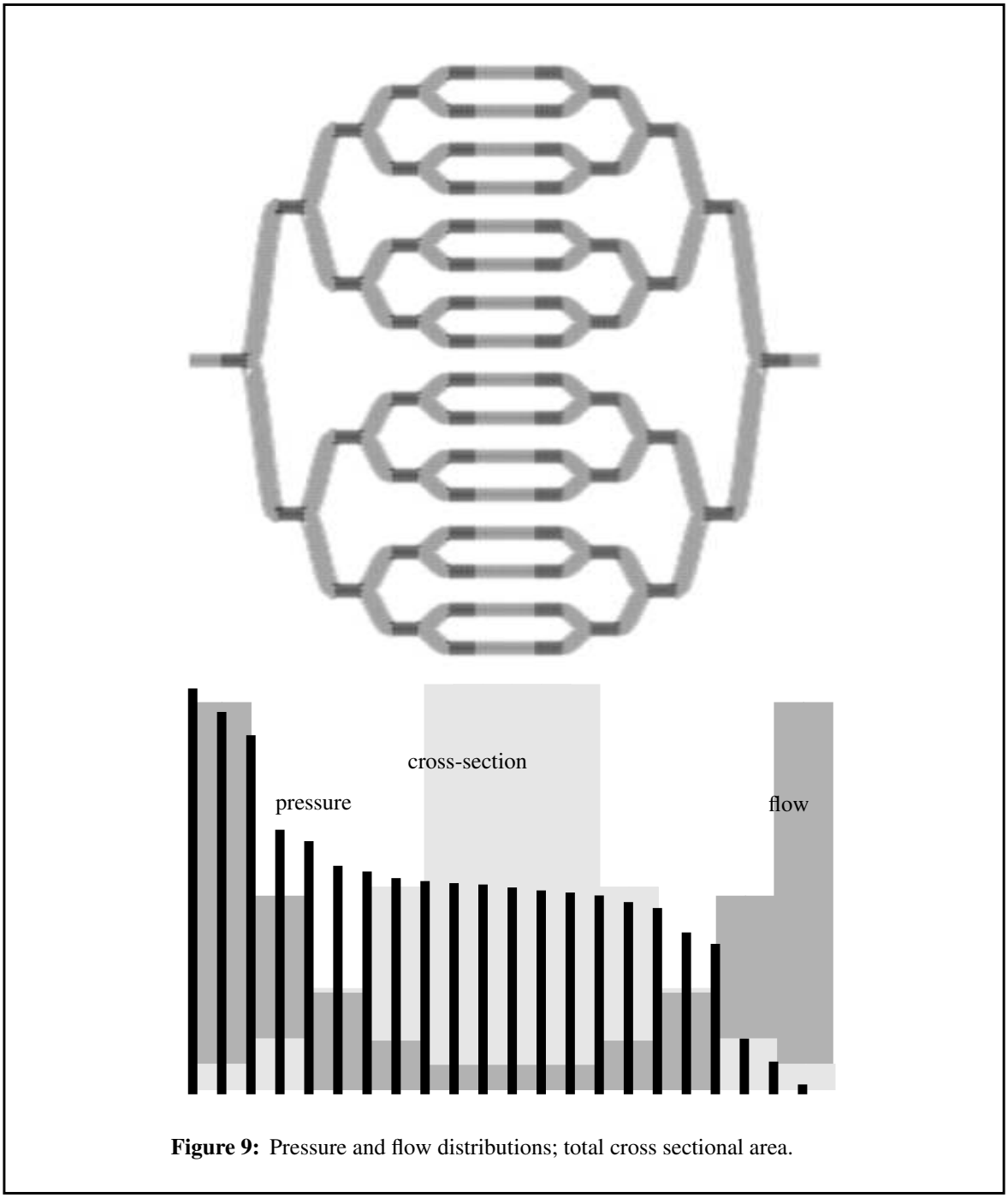


Figure 9: Pressure and flow distributions; total cross sectional area.

The actual pressure distribution is strongly effected by the radii of the tubes making up the network.

Pressure distribution is greatly affected by geometry. This can be used to pick out particular subsystems in a network. Such flow is common in blood vessel networks, and its effect is reproduced in the simulated network shown in figure 10.

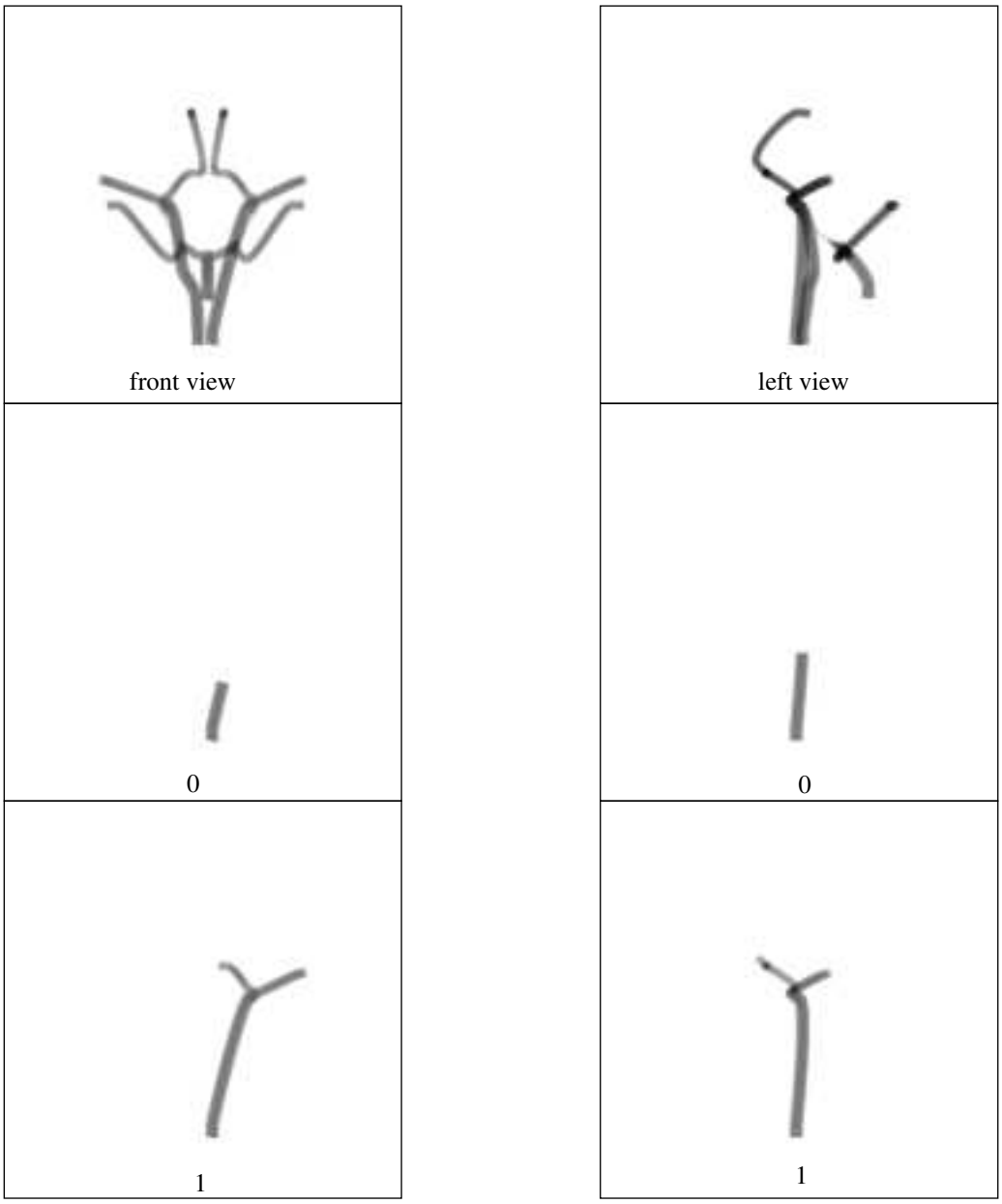


Figure 10: Flow through a subsystem in simulated network of blood vessels. Two sets on frames are shown, each depicting the flow in a subsystem from a different point of view. These sets are arranged in columns, with an image of the complete network at the top; the fluid in these top images is static. The sets of frames were not captured simultaneously, but two separate injection events were used. Frames are chosen for their content, rather than on a regular division of time. Frame numbers for are given. Frames in different sets that share the same frame number only loosely correspond. The figure continues...

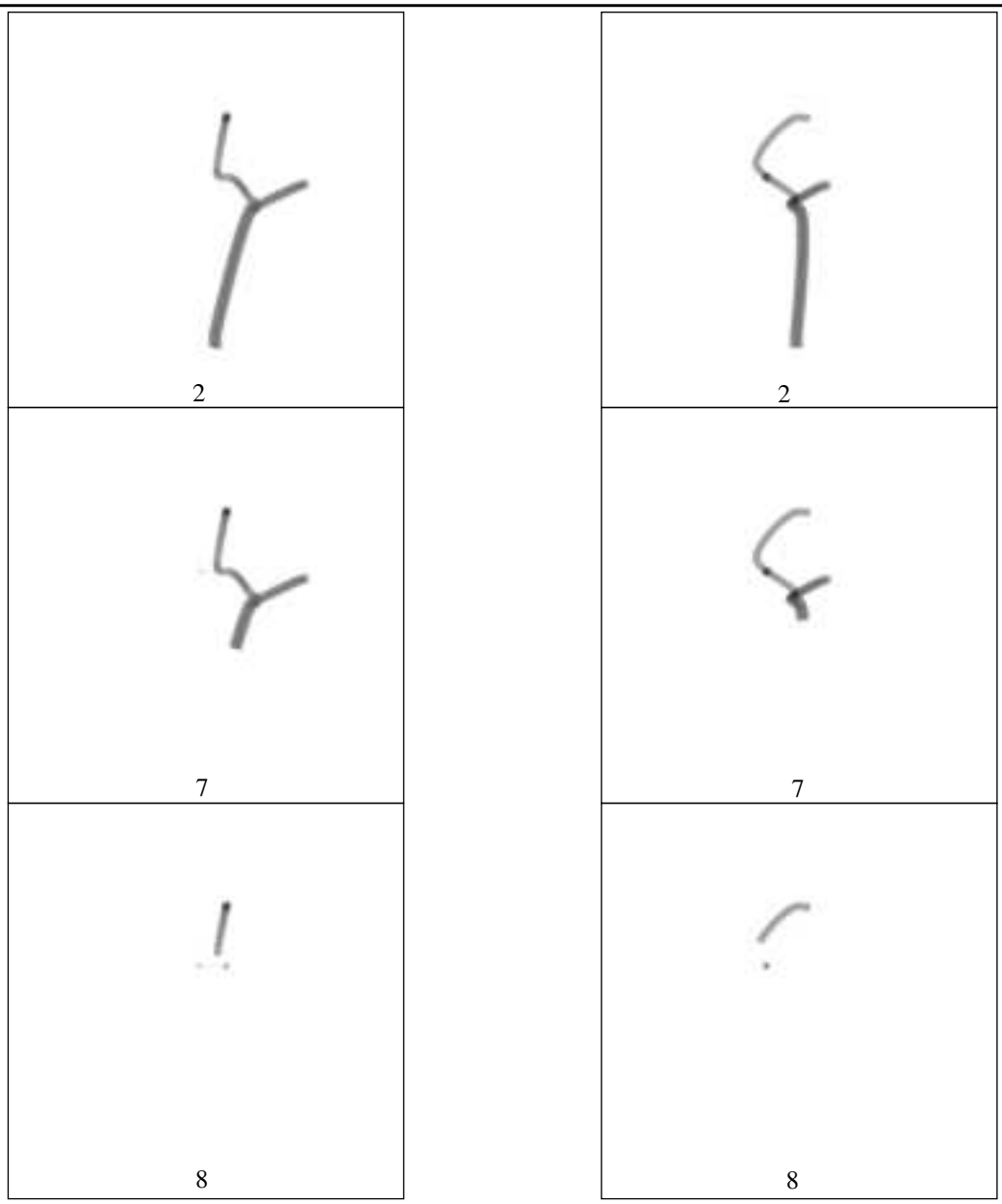


figure 10 - continued

5 Conclusion

A mechanism for simulating and animating fluid as is flows through a network of tubes has been presented. Many assumptions are made, including but not limited to; incompressible fluid, rigid wall containers, laminar flow, instantaneous re-distribution of pressure changes, dye agents the same density as fluid. These assumptions invalidate the simulation for many applications. For example, it would not serve as a physiological model of blood flow in vessel networks, nor does it handle turbulent flows. Measurements taken from the simulation show that general trends are reproduced, albeit in a crude form; the model may provide a suitable starting point for more advanced simulations.

The animation control mechanism can be separated from the simulation computations. So, it can be regarded as a general framework that may find use in computer graphics or scientific visualisation. It advects fluid from frame to frame and has proven to be very efficient, it may possibly be used in a real-time situation. Experience with using the system has shown that it is rendering that spends most computer time (no formal measurements have been made to quantify this). Rendering only the changes in a frame may provide a way forward to real-time animation of laminar flow in networks.

6 References

Ballard, D.H., and Brown, C.M. (1982) Computer Vision. *Prentice-Hall, Englewood Cliffs, NJ, USA*.

Briscolini M.,and P. Santangelo (1991) Animation of computer simulations of two-dimensional turbulence and three dimensional flows. *IBM J Res Develop 35(1/2)*: 119-139

Bronsvoort, W., and Klok, F. (1985) Ray tracing generalised cylinders. *ACM Trans. Graph 4(4):* pp 291-303

Bronsvoort, W. (1992) A surface-scanning algorithm for displaying generalised cylinders. *The Visual Computer 8(3)*: pp 162-170

Foley, J., van Dam, A., Feiner, S., and Hughes, J. (1990) *Computer graphics principles and practice, second edition.* Addison-Wesley, Reading, MA, USA.

Fournier, A., and Reeves, W.T. (1986) A simple model of ocean waves. *Comput. Graph. 24(4) (Proc. SIGGRAPH)*: 75-84

Fung, Y.C. (1984) *Biodynamics: circulation*: Springer-Verlag, New York, USA.

Hall, P.M. (1994) Implicit volume rendering of generalised cylinders. Tech. Report CS-TR-94-10. Department of Computer Science, Victoria University of Wellington, Wellington, New Zealand.

Ihm, I. and Naylor, B. (1991) Piecewise linear approximations of digitized space curves with applications. I*n Scientific visualization of physical phenomena, Patrikalakis N.M. (Ed.) Proc. Comp. Graph. Intl., MIT, June 1991*: 545-569

Jeppson, R.W. (1977) *Analysis of flow in pipe networks*. Ann Arbor Science.

Kass, M., and Miller, G. (1990) Rapid, stable fluid dynamics for computer graphics. *Comput. Graph 24(4) (Proc. SIGGRAPH)*: 305-309

Nagasawa, M. and Kuwahara, K. (1991) Smoothed particle rendering for fluid visualisation in astrophysics. I*n Scientific visualization of physical phenomena, Patrikalakis N.M. (Ed.) Proc. Comp. Graph. Intl., MIT, June 1991*: 589-605

Peachey, D.R. (1986) Modelling waves and surf. *Comput. Graph. 24(4) (Proc. SIGGRAPH)*: 65-74

Ruch,T.C., and Patton, H.D. (1974) *Physiology and biophysics II: circulation, respiration, and fluid balance* W.B. Saunders, Philadelphia.

Stam, J. and Fiume, E. (1993) Turbulent wind fields for gaseous phenomena. *Proc. SIGGRAPH '93*: 369-376

Streeter, V.L., and Wylie, E.B. (1975) *Fluid mechanics (sixth edition)* McGraw-Hill, Tokyo

Wejchert, J., and Haumann, D. (1991) Animation aerodynamics. *Comput Graph 25(4) (Proc. SIGGRAPH)*: 19-22

Yeager, L., Upson, C., and Myers, R. (1986) Combining physical and visual simulation - creation of the planet Jupiter for the film 2010. *Comput. Graph. 24(4) (Proc. SIGGRAPH)*: 85-93

7 Appendix: accounting for the interdependency of flow in a junction

The computation of flow rate through a junction depends on the extent to which tubes in the junction overlap. To see this consider the computation of flow through the junction. Let Q_i be the rate of flow through the i^{th} tube of the junction. Then, the volume, V_i, consumed by that tube in a time, t, is

$$V_i = tQ_i \tag{29}$$

and the total volume consumed by the independent tubes is

$$V_{ind} = \Sigma_i V_i \tag{30}$$

Because the tubes are not independent this measure of volume is in error, the actual volume consumed, V_{act}, is

$$V_{act} = V_{ind} - V_{overlap} \tag{31}$$

The equation serves to define $V_{overlap}$ as the difference between the actual volume and the volume measured by summation of individual volumes, V_i. Define the effective conductance of the junction when modelled as independent tubes, C_{ind} as

$$C_{ind} = V_{ind} / \Delta p \tag{32}$$

where Δp is a pressure gradient. The actual conductance of the junction, C_{act}, is defined in analogous fashion;

$$C_{act} = V_{act} / \Delta p \tag{33}$$

Hence

$$C_{act} = V_{act}\, C_{ind} / V_{ind} = (V_{ind} - V_{overlap})C_{ind} / V_{ind} \tag{34}$$

which shows that conductance depends of the volume of common overlap. To compute $V_{overlap}$ in

a junction of n tubes, each permutation of possible overlap needs to be accounted for. An expression for $V_{overlap}$ can be derived by partitioning the overlapping volumes into equivalence classes, each distinguished by the number of tubes common to the overlap. A given equivalence class will be an overlap produced by r tubes, say. Such a class will have $m_{r,n} = n!/r!(n-r)!$ members. Let $V_{j,r}$ be the volume of the j^{th} member of the class. Then

$$V_{overlap} = \Sigma_r (r-1)\Sigma_j V_{j,r} \tag{35}$$

where Σ_j runs over each volume in the r^{th} equivalence class, and Σ_r runs over all equivalence classes for which $r > 1$.

Notice that $V_{overlap}$ need not be computed, the important factor to discover when compensating for the overlap is

$$k = (V_{ind} - V_{overlap}) / V_{ind} = V_{act} / V_{ind} \tag{36}$$

This requires that V_{act} be measured. Even so, surfaces that bound V_{act} are very complicated, and numeric methods should be used. One way to measure the volume is to divide the volume into cubes at some arbitrary resolution. The vertices at each cube can be sampled, and a volume estimate id derived that is based on the number of them that reside within the volume. The problem with this is resolution; acceptably accurate results require a very high resolution, but time complexity rises with its cube. Fortunately, a typical value of k turned out to be very close to 1, indicating that overlap may be neglected.