Muskingum routing: theory, example and a brief venture into linear algebra

Jan Verkade

December 11, 2016

Note: most of this is based on the slides downloaded from http://www.ce.utexas.edu/prof/maidment/CE374KSpring2011/Visual/HydrologicRouting.pptx

Introduction

Muskingum routing is a relatively simple form of *hydrologic* or *lumped* routing which is governed by the continuity equation (mass and, assuming constant density, volume) and a flow–storage relationship. Note that hydraulic or dynamic routing equations are governed by equations of continuity and momentum.

Governing equations

Continuity equation of mass and volume:

$$\frac{dS}{dt} = I\left(t\right) - Q\left(t\right) \tag{1}$$

Flow, storage relationship, assuming a linear reservoir:

$$S = kQ \tag{2}$$

Combining equations 1 and 2 yields:

$$\begin{aligned}
\frac{dS}{dt} &= I\left(t\right) - Q\left(t\right) \\
\frac{d\left(kQ\right)}{dt} &= I\left(t\right) - Q\left(t\right) \\
k\frac{dQ}{dt} + Q\left(t\right) &= I\left(t\right) \qquad (3)
\end{aligned}$$

This shows that the outflow hydrograph $Q\left(t\right)$ can be expressed as a function of the inflow hydrograph $I\left(t\right)$.

Storage in a reach

Total reach storage comprises prism and wedge storages and is thus calculated using values of inflow I and outflow Q (thus eliminating one unknown!):

$$S_{\text{prism}} = kQ \tag{4}$$

$$S_{\text{wedge}} = xk\left(I - Q\right) \tag{5}$$

Total storage S is then calculated as follows:

$$
\begin{aligned}
S &= S_{\text{prism}} + S_{\text{wedge}} \\
&= kQ + xk\left(I - Q\right) \\
&= k\left(Q + x\left(I - Q\right)\right) \\
&= k\left(Q + xI - xQ\right) \\
&= k\left(xI + Q - xQ\right) \\
&= k\left(xI + \left(1 - x\right)Q\right)
\end{aligned} \tag{6}
$$

Discretization of equations

Discretizing the equation 6 (storage) yields:

$$
\begin{aligned}
S_{\text{j}+1} - S_{\text{j}} &= k\left[xI + \left(1 - x\right)Q\right]_{\text{j}+1} - k\left[xI + \left(1 - x\right)Q\right]_{\text{j}} \\
&= k\{\left[xI_{\text{j}+1} + \left(1 - x\right)Q_{\text{j}+1}\right] - \left[xI_{\text{j}} + \left(1 - x\right)Q_{\text{j}}\right]\} \\
&= k\left(1 - x\right)Q_{\text{j}+1} + kxI_{\text{j}+1} - kxI_{\text{j}} - k\left(1 - x\right)Q_{\text{j}}
\end{aligned} \tag{7}
$$

Note that in the last step, the terms are ordered: $Q_{\text{j}+1}$ first, then $I_{\text{j}+1}$ followed by I_{j} and Q_{j}. We also discretize the equation 1 (the continuity equation, or mass/volume balance):

$$
\begin{aligned}
S_{\text{j}+1} - S_{\text{j}} &= \frac{I_{\text{j}+1} - I_{\text{j}}}{2}\Delta t - \frac{Q_{\text{j}+1} - Q_{\text{j}}}{2}\Delta t \\
&= \frac{1}{2}\Delta t\left(I_{\text{j}+1} - I_{\text{j}}\right) - \frac{1}{2}\Delta t\left(Q_{\text{j}+1} - Q_{\text{j}}\right) \\
&= -\frac{1}{2}\Delta tQ_{\text{j}+1} + \frac{1}{2}\Delta tI_{\text{j}+1} - \frac{1}{2}\Delta tI_{\text{j}} + \frac{1}{2}\Delta tQ_{\text{j}}
\end{aligned} \tag{8}
$$

Note that here, too, the terms are ordered in the last step: $Q_{\text{j}+1}$ first, then $I_{\text{j}+1}$ followed by I_{j} and Q_{j}.

Finally, equations 7 and 8 are combined and the items containing $Q_{\text{j}+1}$ are transferred to the left:

$$
\begin{aligned}
k\left(1 - x\right)Q_{\text{j}+1} + kxI_{\text{j}+1} - kxI_{\text{j}} - k\left(1 - x\right)Q_{\text{j}} &= -\frac{1}{2}\Delta tQ_{\text{j}+1} + \frac{1}{2}\Delta tI_{\text{j}+1} - \frac{1}{2}\Delta tI_{\text{j}} + \frac{1}{2}\Delta tQ_{\text{j}} \\
k\left(1 - x\right)Q_{\text{j}+1} + \frac{1}{2}\Delta tQ_{\text{j}+1} &= -kxI_{\text{j}+1} + \frac{1}{2}\Delta tI_{\text{j}+1} + kxI_{\text{j}} - \frac{1}{2}\Delta tI_{\text{j}} + k\left(1 - x\right)Q_{\text{j}} + \frac{1}{2}\Delta tQ_{\text{j}} \\
Q_{\text{j}+1}\left[k\left(1 - x\right) + \frac{1}{2}\Delta t\right] &= I_{\text{j}+1}\left[\frac{1}{2}\Delta t - kx\right] + I_{\text{j}}\left[kx - \frac{1}{2}\Delta t\right] + Q_{\text{j}}\left[k\left(1 - x\right) + \frac{1}{2}\Delta t\right]
\end{aligned} \tag{9}
$$

This can be rewritten:

$$
Q_{\text{j}+1} = C_1 I_{\text{j}+1} + C_2 I_{\text{j}} + C_3 Q_{\text{j}} \tag{10}
$$

where

$$
\begin{aligned}
C_1 &= \frac{\Delta t - 2kx}{2k\left(1 - x\right) + \Delta t} \\
C_2 &= \frac{\Delta t + 2kx}{2k\left(1 - x\right) + \Delta t} \\
C_3 &= \frac{2k\left(1 - x\right) - \Delta t}{2k\left(1 - x\right) + \Delta t}
\end{aligned} \tag{11}
$$

Note that the denominator is identical across all three equations.

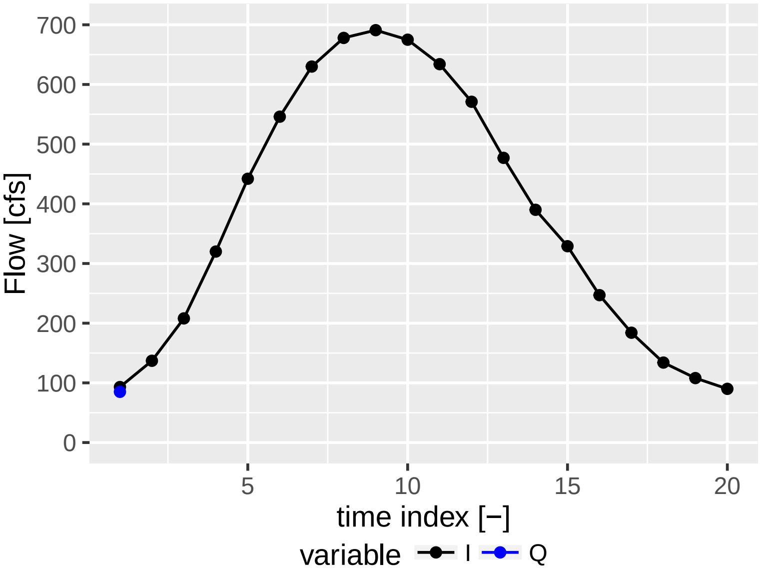
Example

Given:

- the inflow hydrograph $I\left(t\right)$ (see below);
- $k = 2.3$ hours;
- $x = 0.15\left[-\right]$;
- $\Delta t = 1$ hour;
- initial flow $Q\left(\mathrm{t}=1\right) = 85$ cfs.

```
I <- c(93,137,208,320,442,546,630,678,691,675,634,571,477,390,329,247,184,134,108,90)
Q <- vector("numeric",length=length(I)); Q[1] <- 85; Q[2:length(Q)] <- NA
k <- 2.3; x <- 0.15; dt <- 1
```

The inflow hydrograph then looks like:



Constants C_1 through C_3 are calculated as follows:

```
C1 <- (dt-2*k*x)/(2*k*(1-x)+dt)
C2 <- (dt+2*k*x)/(2*k*(1-x)+dt)
C3 <- (2*k*(1-x)-dt)/(2*k*(1-x)+dt)
```

which yields the following values, respectively:

```
## [1] 0.06313646
```

```
## [1] 0.3441955
```

```
## [1] 0.592668
```

Note that the sum of these constants should equal 1:

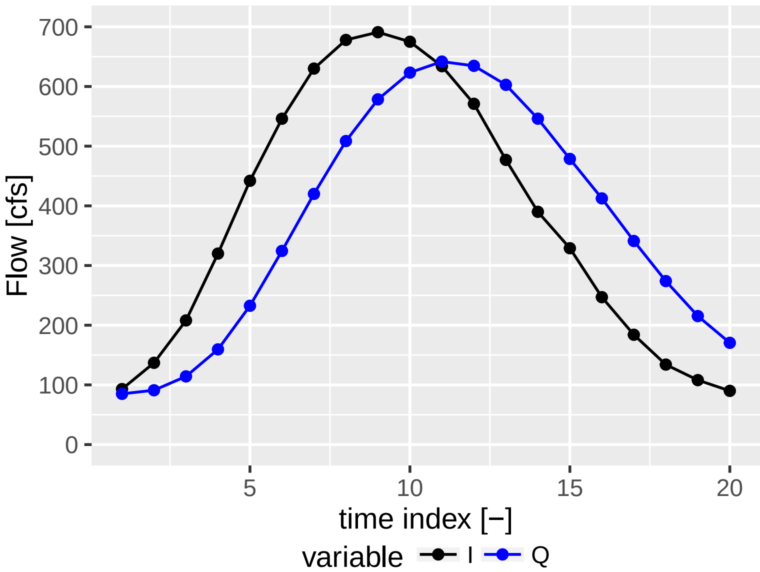
```
C1+C2+C3
```

```
## [1] 1
```

Then we calculate the components of equation 10:

```
for (t in 1:(length(I)-1)) {
  Q[t+1] <- C1*I[t+1] + C2*I[t] + C3*Q[t]
}
```

which we then show graphically:



Solution using linear algebra[1]

We can re-write the Muskingum equation $Q_{\text{j+1}} = C_1 I_{\text{j+1}} + C_2 I_{\text{j}} + C_3 Q_{\text{j}}$ or $Q_{\text{j}} = C_1 I_{\text{j}} + C_2 I_{\text{j-1}} + C_3 Q_{\text{j-1}}$ using matrices and vectors:

$$\begin{bmatrix} Q_{\text{t1}} \\ Q_{\text{t2}} \\ Q_{\text{t3}} \\ \vdots \\ Q_{\text{T}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ C2 & C1 & 0 & \cdots & 0 & 0 \\ 0 & C2 & C1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & C2 & C1 \end{bmatrix} \begin{bmatrix} I_{\text{t1}} \\ I_{\text{t2}} \\ I_{\text{t3}} \\ \vdots \\ I_{\text{T}} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ C3 & 0 & 0 & \cdots & 0 & 0 \\ 0 & C3 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & C3 & 0 \end{bmatrix} \begin{bmatrix} Q_{\text{t1}} \\ Q_{\text{t2}} \\ Q_{\text{t3}} \\ \vdots \\ Q_{\text{T}} \end{bmatrix} \tag{12}$$

which we'll simplify:

$$\vec{Q} = \bar{\bar{D}}\vec{I} + \bar{\bar{A}}\vec{Q} \tag{13}$$

Note that $\vec{I}$, Q_{t1} and all values of $C1$, $C2$ and $C3$ are known. We need to solve equation 13 for Q_{t} with $\text{t} > \text{t1}$. We'll do this by re-writing slightly as to allow solvers to find their answers quickly:

$$-\bar{\bar{D}}\vec{I} = (\bar{\bar{A}} - \bar{\bar{E}})\vec{Q} \tag{14}$$

with $\bar{\bar{E}}$ being the identity matrix (usually denoted as $\bar{\bar{I}}$ but as this variable was already taken, using the Dutch notation $\bar{\bar{E}}$).

Implementation in R

First, the matrices $\bar{\bar{A}}$ and $\bar{\bar{D}}$ have to be constructed. Matrix $\bar{\bar{D}}$ has two patterns that are very diagonal-like, but not quite. However, I have used the **diag()** function to construct the matrix – in two steps:

[1]Many thanks to Jorn Baayen at Deltares for help in constructing Muskingum as a linear algebra problem.

```r
T <- length(I)
D1 <- matrix(data=0, nrow=T,ncol=T); D2 <- D1
D1[2:T,2:T]   <- diag(T-1,x=C1)
D2[2:T,1:T-1] <- diag(T-1,x=C2)
D <- D1+D2; rm(D1,D2)
```

Likewise for matrix $\bar{\bar{A}}$:

```r
A <- matrix(data=0,nrow=T,ncol=T)
A[2:T,1:T-1] <- diag(T-1,x=C3)
```

The function **solve()** will be used to solve the equation $Ax = b$ for x. First, a and b are constructed (the reason for this is that an initial value will need to be imposed on b).

```r
a <- A-diag(T); b <- -D%*%I
```

The Muskingum problem is an initial value problem. Hence, I am changing the first equation to be solved in such a way that it immediately knows the value of outlow at the first time step. There may be a more elegant way to do this, but I am not aware of it.

```r
b[1] <- -Q[1]
```

After that, the matrix system can be actually solved:

```r
q <- solve(a=a,b=b)
```

And a comparison between Q (calculated by a loop) and q (calculated using matrix algebra) reveals identical values. While I was hoping that the linear algebra approach would be a lot faster, the converse was true: for large timeseries (~10,000 values), the matrix approach was approx. 7,000 times slower!