Chromatography with the Flux-corrected MacCormack method, pp. 202-204. Extra material for *Introduction to Chemical Engineering Computing,* 2nd ed., Bruce A. Finlayson, Wiley (2012).

The following discussion gives the details of the flux-corrected MacÇormack method, from Finlayson, pp. 337-339, 1992. The program (in MATLAB form) is in MacCormack_flux.txt. Separate that file into the m-files for MATLAB: initial.m, inlet.m, param.m, macflux.m, runcode.m. Then issue the command runcode and Figure 9.18b will be created.

Reference

Finlayson, B. A., *Numerical Methods for Problems for Moving Fronts*, Ravenna Park (1992).

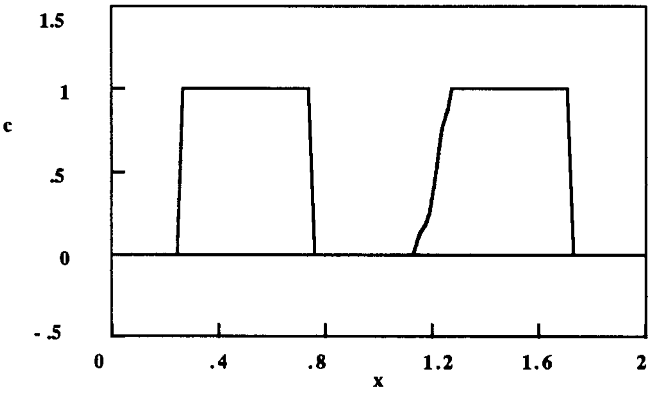

Figure 9.9b. Adsorption with Eq. (9.72), Random Choice Method, 100 nodes, $\Delta t = 0.01$

The speeds of the left-hand Riemann solution and right-hand Riemann solution are given by

$$S_L = \frac{1}{1 + \frac{1-\phi}{\phi}\frac{df}{dc}\Big|_{c_L}}, \quad S_R = \frac{1}{1 + \frac{1-\phi}{\phi}\frac{df}{dc}\Big|_{c_R}}. \tag{9.85}$$

Typical results are shown in Figure 9.9b; the concentration profile is quite sharp, retaining a shock on one side with a sloping, trailing edge, as exhibited by the exact solution.

MacCormack method. To apply the MacCormack method (with and without flux-correction), we rearrange Eq. (9.78) as

$$\frac{\partial c}{\partial \eta} + \frac{1}{P(c)}\frac{\partial c}{\partial \xi} = 0. \tag{9.86}$$

Then we define the quantity

$$M(c) = \int_0^c \frac{dc}{P(c)}. \tag{9.87}$$

The first derivative is then

$$\frac{\partial M(c)}{\partial \xi} = \frac{1}{P(c)}\frac{\partial c}{\partial \xi} \tag{9.88}$$

and the problem can be written as

$$\frac{\partial c}{\partial \eta} + \frac{\partial M(c)}{\partial \xi} = 0. \tag{9.89}$$

For the Langmuir adsorption we have

$$P(c) = 1 + \frac{1-\phi}{\phi}\frac{df}{dc}, \tag{9.90}$$

$$M(c) = \int_0^c \frac{dc}{1 + \frac{1-\phi}{\phi}\frac{df}{dc}}. \tag{9.91}$$

Expansion of this gives

$$M(c) = \int_0^c \frac{(1 + 2Kc + K^2c^2)\, dc}{1 + \frac{1-\phi}{\phi}\gamma + 2Kc + K^2c^2}. \tag{9.92}$$

This is cumbersome to use, but fortunately we do not have to. The MacCormack method without flux-correction applied to Eq. (9.86) is

$$\frac{c*_i^{n+1} - c_i^n}{\Delta\eta} = -\frac{M(c_{i+1}^n) - M(c_i^n)}{\Delta\xi}, \tag{9.93}$$

$$\frac{c_i^{n+1} - \frac{1}{2}(c_i^n + c*_i^{n+1})}{\Delta\eta} = -\frac{M(c*_i^{n+1}) - M(c*_{i-1}^{n+1})}{2\,\Delta\xi}, \tag{9.94}$$

To evaluate the difference on the right-hand side, we write it as

$$M(c_{i+1}^n) - M(c_i^n) = \int_{c_i^n}^{c_{i+1}^n} \frac{dc}{P(c)}. \tag{9.95}$$

Next we evaluate the integral, using the trapezoid rule, as

$$\int_{c_i^n}^{c_i^{n+1}} \frac{dc}{P(c)} = \frac{1}{2}\left(\frac{1}{P(c_{i+1}^n)} + \frac{1}{P(c_i^n)}\right)(c_{i+1}^n - c_i^n). \tag{9.96}$$

The MacCormack method is then

$$\frac{c*_i^{n+1} - c_i^n}{\Delta\eta} = -\frac{1}{2\Delta\xi}\left(\frac{1}{P(c_{i+1}^n)} + \frac{1}{P(c_i^n)}\right)(c_{i+1}^n - c_i^n), \tag{9.97}$$

$$\frac{c_i^{n+1} - \frac{1}{2}(c*_i^{n+1} + c_i^n)}{\Delta\eta} = -\frac{1}{4\Delta\xi}\left(\frac{1}{P(c*_i^{n+1})} + \frac{1}{P(c*_{i-1}^{n+1})}\right)(c*_i^{n+1} - c*_{i-1}^{n+1}). \tag{9.98}$$

Oscillations in the solution would be disastrous if they cause the coefficient P(c) to be evaluated for a negative c value because this could give P a value of zero. In this case that will not happen since $df/dc \geq 0$ for all c values, whether positive or negative.

Results for the MacCormack method without flux-correction are shown in Figure 9.9c. The solution contains significant oscillations. Indeed, the solution looks similar to Figure 4.9b for the advection equation, except that the trailing

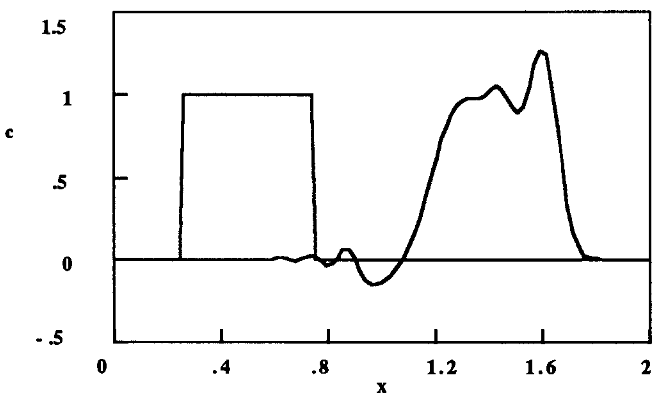

Figure 9.9c. Adsorption with Eq. (9.72), MacCormack Method, 100 nodes, $\Delta t = 0.01$

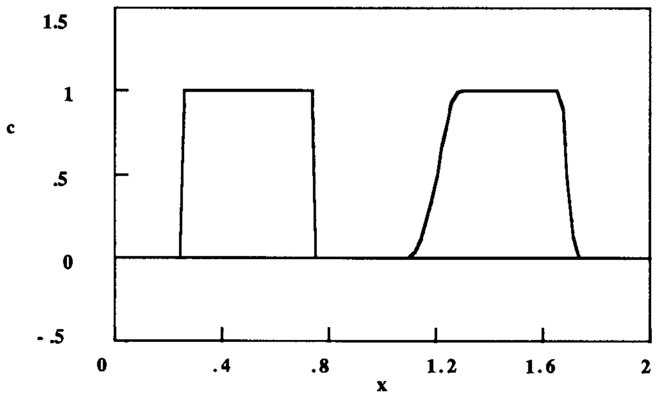

Figure 9.9d. Adsorption with Eq. (9.72), MacCormack Method with Flux-correction 100 nodes, $\Delta t = 0.01$

edge is more smoothed. The method is improved significantly when the flux-correction step is added: the oscillations disappear and the leading edge is quite sharp (see Figure 9.9d). The trailing edge is more diffuse than the leading edge, as expected.

Taylor-Galerkin method. The Taylor-Galerkin method can be obtained most easily by using Eq. (9.89) rather than Eq. (9.78). This is the form used in Chapter 5 when the Taylor-Galerkin method was applied to Burger's equation and the general flux equation, Eq. (5.25). The Taylor-Galerkin method is obtained by inspection from Eq. (5.53):

$$\frac{1}{6}\frac{c_{i+1}^{n+1}-c_{i+1}^{n}}{\Delta\eta}+\frac{2}{3}\frac{c_{i}^{n+1}-c_{i}^{n}}{\Delta\eta}+\frac{1}{6}\frac{c_{i-1}^{n+1}-c_{i-1}^{n}}{\Delta\eta}=-\frac{M_{i+1}^{n}-M_{i-1}^{n}}{2\Delta\xi}+$$
$$+\frac{\Delta\eta}{4\Delta\xi^{2}}\left\{\left[\frac{1}{P^{2}(c_{i+1}^{n})}+\frac{1}{P^{2}(c_{i}^{n})}\right][c_{i+1}^{n}-c_{i}^{n}]-\left[\frac{1}{P^{2}(c_{i}^{n})}+\frac{1}{P^{2}(c_{i-1}^{n})}\right][c_{i}^{n}-c_{i-1}^{n}]\right\}. \quad (9.99)$$

```
% initial.m
% M-file to set the initial conditions, and the numerical parameters
delt = 0.01
tfinal = 1
nplot = 1
nstep = tfinal/(nplot*delt)
%revise tfinal in case the there is round-off error
tfinal = nstep*nplot*delt
delx = 0.02
nx = 101
x(1)=0;
for ix=1:nx-1
        x(ix+1) = x(ix)+delx;
end
cinit(1)=inlet(0.);
for ix=2:nx
%       cinit(ix)=0;
        if x(ix)<0.25
                cinit(ix) = 0.;
        elseif x(ix)>0.75
                cinit(ix) = 0.;
        else
                cinit(ix) = 1.;
        end
end
cinit
display('Initial condition is shown above and plotted; press any key to continue.')
plot(x,cinit)
axis([x(1) x(nx) -.5 1.5])
xlabel('x')
ylabel('c')
title('Initial Concentration')
pause
for it=1:nstep*nplot+1
        time=delt*(it-1);
        y(it) = inlet(time);
        xd(it) = time;
end
display('Inlet concentration vs. time is plotted; press any key to continue.')
plot(xd,y)
xlabel('time')
ylabel('c')
title('Inlet Concentration versus time')
pause
%****************************************************************
```

```
% inlet.m
function y=inlet(t)
% enter with t = time
% exit with inlet = inlet concentration at that time
y = 0;
%**************************************************************


% param.m
% Set the parameters for Langmuir adsorption
phi = 0.485
Kads = 2
gamma = 0.1
%gamma = 2
%**************************************************************


% macflux.m
% MacCormack method with flux correction
% Adsorption (equilibrium) with Langmuir isotherm
% enter with
        % x(i), i = 1:nx, delx and nx
        % c(i), i = 1:nx, concentration at time t1
        % nstep, the number of time steps to compute for
        % delt, the timestep
        % parameters phi, K, and gamma
% exit with c(i), i = 1:nx for time t1+nstep*delt

for j=2:nx
        csoln(j) = c(j);
end
% add the ficticious point nx+1
csoln(nx+1)=csoln(nx);
para=delt/delx;
eps=delt*(1-phi)/phi;
for ii=1:nstep
        time = time + delt;
        csoln(1)=inlet(time);
        for ix=1:nx+1
                pp(ix)=1+gamma*(1-phi)/(phi*(1+Kads*csoln(ix))^2);
        end
        for ix=2:nx
                cnew(ix)=csoln(ix)-para*.5*(csoln(ix+1)-csoln(ix))*(1/pp(ix+1)+1/pp(ix));
        end
        cnew(1)=csoln(1);
```

```
	cnew(nx+1) = cnew(nx);
	for ix=1:nx
		pp(ix)=1+gamma*(1-phi)/(phi*(1+Kads*cnew(ix))^2);
	end
	for ix=2:nx
		ctemp(ix)=.5*(csoln(ix)+cnew(ix))-para*(cnew(ix)-cnew(ix-
1))*(1/pp(ix)+1/pp(ix-1))/4;
	end
	ctemp(1) = csoln(1);

	% The following are the Flux-corrected transport steps.  If you don't want them
	% just put ctemp into csoln as in the next three statements.
%	for ix=1:nx
%		csoln(ix)=ctemp(ix)
%	end

	% csoln is c at n, ctemp is c-squiggle,calculate cnew to be c-hat
	for ix=2:nx
	% use eta = 1/8
		cnew(ix)=ctemp(ix)+(csoln(ix+1)-2*csoln(ix)+csoln(ix-1))/8;
	end
	cnew(1)=csoln(1);
	% anti-diffusion step, calculate the deltas
	for ix=1:nx
		fcdel(ix)=cnew(ix+1)-cnew(ix);
	end
	fcdel;
	% calcualte the fluxes
	for ix = 1:nx-1
		delhat=(ctemp(ix+1)-ctemp(ix))/8;
		% use eta = 1/8 again
		if (delhat>=0)
			sign=1;
		else
			sign=-1;
		end
		if ix>1
			amin= min([sign*fcdel(ix-1),abs(delhat),sign*fcdel(ix+1)]);
		else
			% effectively sets fcdel(0) = 0
			amin = 0;
		end
		fc(ix) = sign * max ([0 amin]);
	end
	fc(nx)=0;
	for ix=2:nx
```

```
                csoln(ix)=cnew(ix)-(fc(ix)-fc(ix-1));
        end
        csoln(nx+1) = csoln(nx);
% end of flux correction step

end
% return answer to c(j), j=1:n
for j=1:nx
        c(j) = csoln(j);
end
%*************************************************************


% runcode.m

% find initial conditions and set numerical parameters
initial
param

% put initial conditions in the working array and the first plotting array
for i=1:nx
        c(i) = cinit(i);
        cplot(1,i)=c(i);
end
time = 0.;

% calculate for each plot
for i=1:nplot
        macflux
        % put solution into next plotting array
        for j=1:nx
                cplot(i+1,j)=c(j);
        end
end

% plot solutions
if nplot==1
                plot(x,cplot(1,:),'r',x,cplot(2,:),'b')
        elseif nplot==2
                plot(x,cplot(1,:),'r',x,cplot(2,:),'b',x,cplot(3,:),'g')
        elseif nplot==3
                plot(x,cplot(1,:),'r',x,cplot(2,:),'b',x,cplot(3,:),'g',x,cplot(4,:),'k')
        elseif nplot==4
                plot(x,cplot(1,:),'r',x,cplot(2,:),'b',x,cplot(3,:),'g',x,cplot(4,:),'k',x,cplot(5,:),'m')
        else
```

```
			plot(x,cplot(1,:),'r',x,cplot(nplot-4,:),'b',x,cplot(nplot-3,:),'g',x,cplot(nplot-
2,:),'k',x,cplot(nplot-1,:),'m',x,cplot(nplot,:),'c')
end

axis([x(1) x(nx) -.5 1.5])
xlabel('x')
ylabel('c')
title (['Solution with phi = ',num2str(phi),', K = ',num2str(Kads),', gamma =
',num2str(gamma)])
```