

**Background material for the Scilab script "CompHydraulicsIV\_Script.sce" based on Chapter 5 - from Vreugdenhil, C.B., 1989, "Computational Hydraulics - An Introduction," Springer-Verlag, Berlin.**

*Material from Chapter 5 - The Leap-Frog Method*

The leap-frog method for equation (4.3) is given by Equation (5.6).

This scheme is called the leap-frog method because of its use of a staggered grid for calculating the derivatives. It is a three-level method that requires the knowledge of two previous levels. For the starting step, therefore, a two-level method is used to generate the second level before being able to use the leap-frog method for time level number three. The following function, *leapfrog.m*, is used to solve the contaminant transport equation using the leap-frog method:

```
function [x,t,c,sigma] = leapfrog(a,b,Dx,t0,tm,Dt,u,alpha,ci,cb)
% =====
% This function calculates the concentration of contaminant
% c(j,n) = c(x(j),t(n)), j = 1,2,...,J; n = 1,2,...,N, as
% well as the positions x(j) and time levels t(n) for the
% case of contaminant injection into a one-dimensional flow
% domain. The solution is found by solving the partial differential
% equation:      dc/dt + u*(dc/dx) = 0,
% where u is the flow velocity. The solution requires
% function ci(x) which provides the initial conditions
% c(x(j),t0) = ci(x(j)), and function cb(t) which provides
% the upstream boundary condition c(a,t(n)) = cb(t(n)).
% The solution is found in the x-domain = [a,b], and time
% domain = [t0,tm]. The x-increment is Dx, and the time
% increment is Dt. The method used is described by equation
% (5.6) in the book: Vreugdenhil, C.B., 1989, "Computational
% Hydraulics," Springer-Verlag, Berlin - Chapter 5.
% The first time step calculation is performed by using the
% explicit method from Chapter 5 in the same book.
% The alpha parameter for explicit method is such that:
% sigma^2 < alpha < 1 for stability
% =====

% Calculate the Courant number, sigma
sigma = u*Dt/Dx;

% Calculate the number of x positions (J) and the number of
% time steps (N), and create vectors x, t, and matrix c:

J = fix((b-a)/Dx) + 1;
N = fix((tm-t0)/Dt) + 1;
x = [a:Dx:b];
t = [t0:Dt:tm];
c = zeros(J,N);

% Load initial conditions
c(:,1) = feval(ci,x);

% Calculation of concentrations for first time step

% Load boundary conditions
c(1,2) = feval(cb,t(2)); % u/s
c(J,2) = c(J,1) + sigma*(c(J,1)-c(J-2,1)); % d/s (leap frog)

% Calculate concentrations in interior points - first time step
for j = 2:J-1
    c(j,2) = 0.5*(alpha-sigma)*c(j+1,1)+ (1-alpha)*c(j,1)+0.5*(alpha+sigma)*c(j-1,1);
end

% Calculations for time steps 2 through n
```

```

for n = 2:N-1
    % Load boundary conditions
    c(1,n+1) = feval(cb,t(n+1));           % u/s
    c(J,n+1) = c(J,n-1) + sigma*(c(J,n)-c(J-2,n)); % d/s

    % Calculate concentrations in fixerior pofixs
    for j = 2:J-1
        c(j,n+1) = c(j,n-1)-sigma*(c(j+1,n)-c(j-1,n));
    end
end

% End function

```

The following script, *CompHydExIV\_Script.m* was put together to drive the function *leapfrog* to reproduce the examples of Chapter 5 in Vreugdenhil's book:

```

% Chapter 5 - Explicit Finite-Difference Methods
% 5.2. The Leap Frog method, eq. (5.6)s
% initial conditions function - create m-file
c0 = 100;
myFile = fopen('ci.m','w');
fprintf(myFile,'function [cci] = ci(x)\r');
fprintf(myFile,'\nif x == 0 \r');
fprintf(myFile,'\n    cci = %s; \r',num2str(c0));
fprintf(myFile,'\nelse \r');
fprintf(myFile,'\n    cci = 0; \r');
fprintf(myFile,'\nend ');
fclose(myFile);
% boundary conditions function
cb = inline(num2str(c0),'t');
% Solution with alpha = 1.0 and sigma = 0.5
a = 0; b = 20000; t0 = 0; tm = 22000;
Dx = 500; Dt = 500; u = 0.5; alpha=1.0;
disp('Solution with alpha = 1.0 and sigma = 0.5');
disp(['a      = ',num2str(a)]);
disp(['b      = ',num2str(b)]);
disp(['t0     = ',num2str(t0)]);
disp(['tm     = ',num2str(tm)]);
disp(['Dx    = ',num2str(Dx)]);
disp(['Dt    = ',num2str(Dt)]);
disp(['u     = ',num2str(u)]);
disp(['alpha = ',num2str(alpha)]);
pause
% Calculate concentration
[x,t,c,sigma]=leapfrog(a,b,Dx,t0,tm,Dt,u,alpha,'ci',cb);
% Plot 3-d solution
figure(1);clf;surf(x,t,c');
xlabel('x');ylabel('t');zlabel('c');
pause
% Show plots x-vs-t
for n=4:4:44
    figure(1);clf;plot(x,c(:,n));axis([0 20000 0 150]);
    title(['contaminant transport - n = ',num2str(n)]);xlabel('x(m)');ylabel('c(mg/l)');
    pause
end
pause

% Graph of solution at selected times - Fig. 5.1(a), p.24
figure(2);clf;subplot(1,2,1);
plot(x,c(:,15),'r',x,c(:,30),'m',x,c(:,44),'b');

```

```

legend('t=7500 s','t=15000 s','t = 22000 s');
axis([0 20000 0 150]);
title('alpha = 1.0 - sigma = 0.5');xlabel('x(m)');ylabel('c(mg/l)');
pause

% Solution with alpha = 1.0 and sigma = 1.0
c0 = 100;
a = 0; b = 20000; t0 = 0; tm = 22000;
Dx = 500; Dt = 1000; u = 0.5; alpha=1.0;
disp('Solution with alpha = 1.0 and sigma = 1.0');
disp(['a      = ',num2str(a)]);
disp(['b      = ',num2str(b)]);
disp(['t0     = ',num2str(t0)]);
disp(['tm     = ',num2str(tm)]);
disp(['Dx     = ',num2str(Dx)]);
disp(['Dt     = ',num2str(Dt)]);
disp(['u      = ',num2str(u)]);
disp(['alpha = ',num2str(alpha)]);
% Calculate concentration
[x,t,c,sigma]=leapfrog(a,b,Dx,t0,tm,Dt,u,alpha,'ci',cb);
% Plot 3-d solution
figure(3);clf;surf(x,t,c');
xlabel('x');ylabel('t');zlabel('c');
pause
% Show plots x-vs-t
for n=2:2:22
    figure(4);clf;plot(x,c(:,n));axis([0 20000 0 150]);
    title(['contaminant transport - n = ',num2str(n)]);xlabel('x(m)');ylabel('c(mg/l)');
    pause
end
pause

% Graph of solution at selected times - Fig. 5.1(a), p.24
figure(2);subplot(1,2,2);
plot(x,c(:,8),'r',x,c(:,15),'m',x,c(:,22),'b');
legend('t=8000 s','t=15000 s','t = 22000 s');
axis([0 20000 0 150]);
title('alpha = 1.0 - sigma = 1.0');xlabel('x(m)');ylabel('c(mg/l)');
% End script

```