SIK2063 Transportprosesser, høst 2000

Problem: Unsteady diffusion into slab

Solve eq. 2.3-5 ($\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial z^2}$) in Cussler numerically using the discretization routines *dss020* and *dss042* and the differential equation solver *ode15s*.

Solve for both the boundary condition given in eqs. 2.3-6–8 and for the "infinite" BC given as $\frac{\partial C}{\partial z} = 0$.

Problem hints

Make a function that returns the time derivatives. this function must calculate the second order derivative. This is done by using *dss020* to obtain the first derivative and *dss042* to get the second order derivative. The time derivative is then just D times this.

The main program calls this function trough

```
[t,y] = ode15s('function_name',[t0 tend],C0)
```

where *t0* and *tend* are the starting and ending time, and *C0* is a vector containing the initial concentrations. The result is found in y, and t is the vector of integration time steps.

The treatment of the boundary conditions is a little tricky. If you use the boundary conditions given in Cussler, then the dependent variables that need to be integrated are only the points between the interface position and the bulk position. However, to obtain the numerical derivatives you need to incorporate the boundary conditions. This means that before you let *dss020* and *dss042* calculate the 1. and 2. order derivatives you must add the two boundary conditions to the vector of concentration variables, then get the derivatives, calculate the time derivative (also for the BC position) and then remove the first and last element of the time-derivative vector before it returns to *ode15s*.

When you use the BC of $\frac{\partial C}{\partial z} = 0$ at the bulk position, only the interface concentration must be treated in the way given above.

Example of using *dss* routines

The routines *dss020* and *dss042* estimate numerically the first and second order derivatives, respectively, for a vector of functional values. Say, you want numerically to determine the derivative of $\sin(x)$:

Code:

```
% Initializing constants and variabels
% ------------------------------------
m     = 20;                     % Number of discretisation points
r     = [0:(2*pi)/(m-1):2*pi];  % Vector of all discr. points
z     = sin(r);

% First derivative
% ----------------
derz  = dss020(0,2*pi,m,z,1);   % The last 1 indicates r-vector in flow direction
dera  = cos(r);                 % Analytic derivative of sin(r)

% Second derivative
% -----------------
derz2 = dss042(0,2*pi,m,z,derz,1,1);
                                % The last two 1,1 indicates type of BC.
                                % 1 indicates Dirichlet condition meaning that the
                                % first derivative is NOT used, 2 indicates that
                                % the derivative is used, i.e. Neuman condition
                                % see help ode042 for further help
dera2 = -sin(r);                % Analytic second derivative of sin(r)

% Plotting
% --------
plot(r,derz,'.',r,dera,'-')     % Plotting the first derivatives as function of r
hold on
plot(r,derz2,'o',r,dera2,'-')   % Plotting the second derivatives as function of r
xlabel('r')
ylabel('derivative')
legend('numerical 1','analytical 1','numerical 2','analytical 2')
```

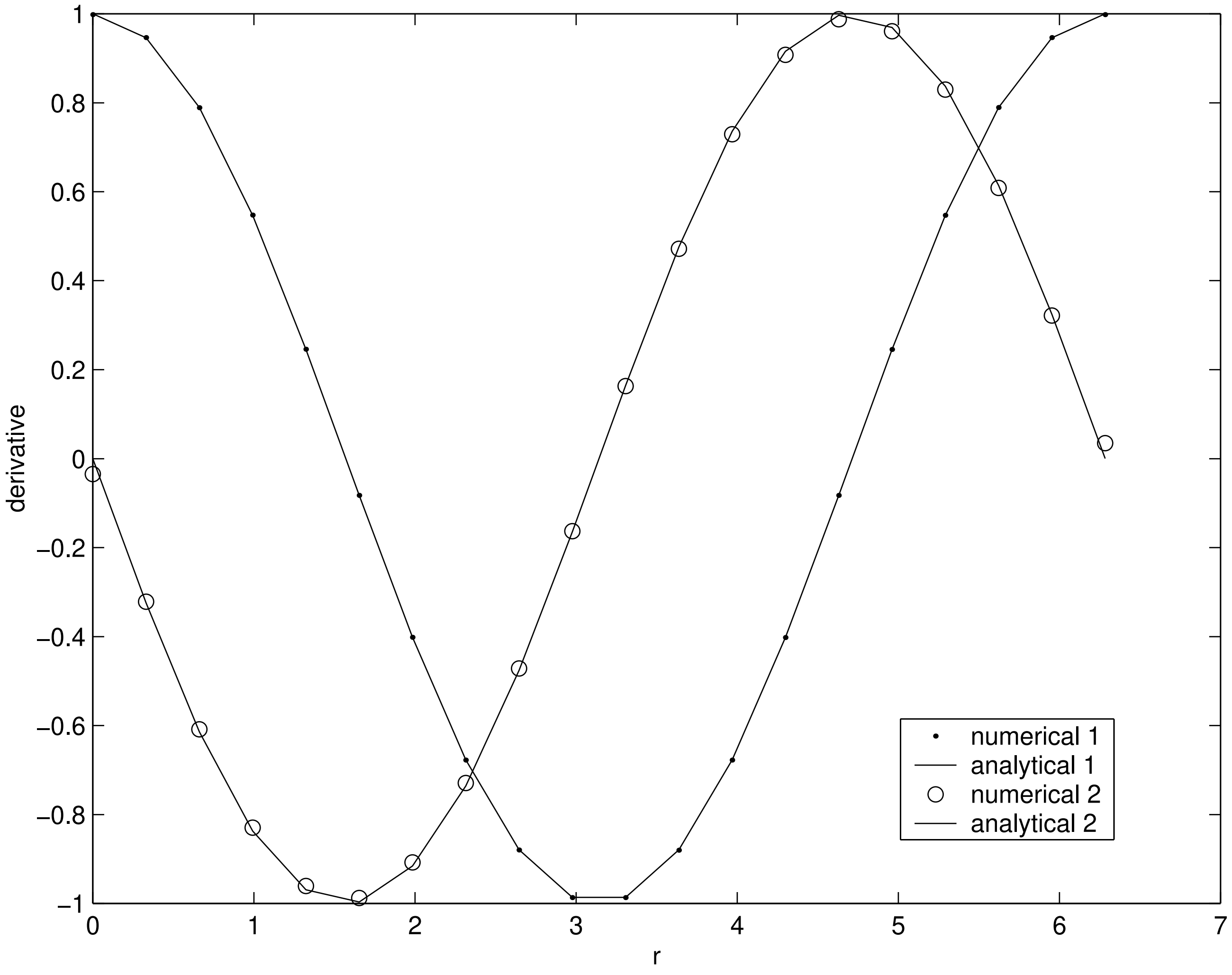

Figure 1: Resulting plot from the program.