Solving Richards Equation with Newton's Method

```
clear, close all, clc
```

Consider a horizontal column filled with soil so that infiltration is driven only by capillary diffusion

PDE: $\frac{\partial \theta}{\partial t} + \nabla \cdot [D(\theta) \nabla \theta] = 0$

IC: $\theta(x, t = 0) = \theta_0$

BC: $\theta(x = 0, t) = \theta_b$

For simplicity we will consider the Brooks-Corey (BC) hydraulic diffusivity

$$D(\theta) = \frac{K_s h_b}{\lambda \Delta\theta} \left(\frac{\theta - \theta_r}{\Delta\theta} \right)^{2 + \frac{1}{\lambda}}$$

$\Delta\theta = \theta_s - \theta_r$ and we use the values for silty loam from above. Due to the dependence of the hydraulic diffusivity on the water content this problem is non-linear.

Solution with Newton's Method

For the fully coupled solution we dicretize the PDE and evaluate all θ's at the new time level, n+1. The residual of the discretized equation is given by

res = **theta** − **theta_old** − dt∗(**D**∗**Kd**(**theta**)∗**G**)∗**theta** − dt∗**fs**

where **theta** = θ^{n+1} and **theta_old** = θ^n. The diagonal matrix **Kd** contains $D_h(\theta)$ averaged to the call faces with an arithmetic average.

This residual is non-linear and hence cannot be written as a liner system in the form

L***theta** $\neq$ dt***fs** .

We must use Netwon's method and linearize the residual

$$\mathbf{r}(\boldsymbol{\theta}^{k+1}) \approx \mathbf{r}(\boldsymbol{\theta}^k) + \nabla \mathbf{r}(\boldsymbol{\theta}^k)\, \Delta\boldsymbol{\theta}^k = 0,$$

and iterate by finding the roots of successive residuals, k, for the update vector, $\Delta\theta$.

Determining the Jacobian Matrix

The Jacobian matrix $\mathbf{J} = \nabla \mathbf{r}(\boldsymbol{\theta}^k)$ can be determined by computing the directional derivative, $D_{\hat{\boldsymbol{\theta}}} \mathbf{r}(\bar{\boldsymbol{\theta}}) = \mathbf{J}(\bar{\boldsymbol{\theta}})\, \hat{\boldsymbol{\theta}}$, of the residual at $\bar{\theta} = \theta^k$ in the direction of the update $\hat{\theta} = \Delta\theta^k / \epsilon$ as follows

$$D_{\hat{\boldsymbol{\theta}}} \mathbf{r}(\bar{\boldsymbol{\theta}}) = \frac{\mathrm{d}}{\mathrm{d}\epsilon} \mathbf{r}\left(\bar{\boldsymbol{\theta}} + \epsilon\, \hat{\boldsymbol{\theta}}\right)\Big|_{\epsilon=0} = \frac{\mathrm{d}}{\mathrm{d}\epsilon} \bar{\boldsymbol{\theta}} + \epsilon\, \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^n - \Delta t\, \mathbf{D}\left[\mathbf{Kd}(\bar{\boldsymbol{\theta}} + \epsilon\, \hat{\boldsymbol{\theta}})\, \mathbf{G}(\bar{\boldsymbol{\theta}} + \epsilon\, \hat{\boldsymbol{\theta}})\right]\Big|_{\epsilon=0}.$$

computing this derivative (see class notes for details) we obtain the Jacobian as

`J(theta) = I - dt*D*(GU*dKd + Kd*G)`

where the matrices are given by

`GU = spdiags(G*theta,0,Nf,Nf)`

`Kd = spdiags(M*DH(theta),0,Nf,Nf)`

`dKd = spdiags(M*dDH(theta),0,Nf,Nf).`

Here the hydraulic diffusivity is DH and its derivative dDH, the latter is given by

$$\frac{\mathrm{d}D_H}{\mathrm{d}\theta} = \frac{K_s h_b}{\Delta\theta^2}\frac{1+2\lambda}{\lambda^2}\left(\frac{\theta-\theta_r}{\Delta\theta}\right)^{1+\frac{1}{\lambda}}$$

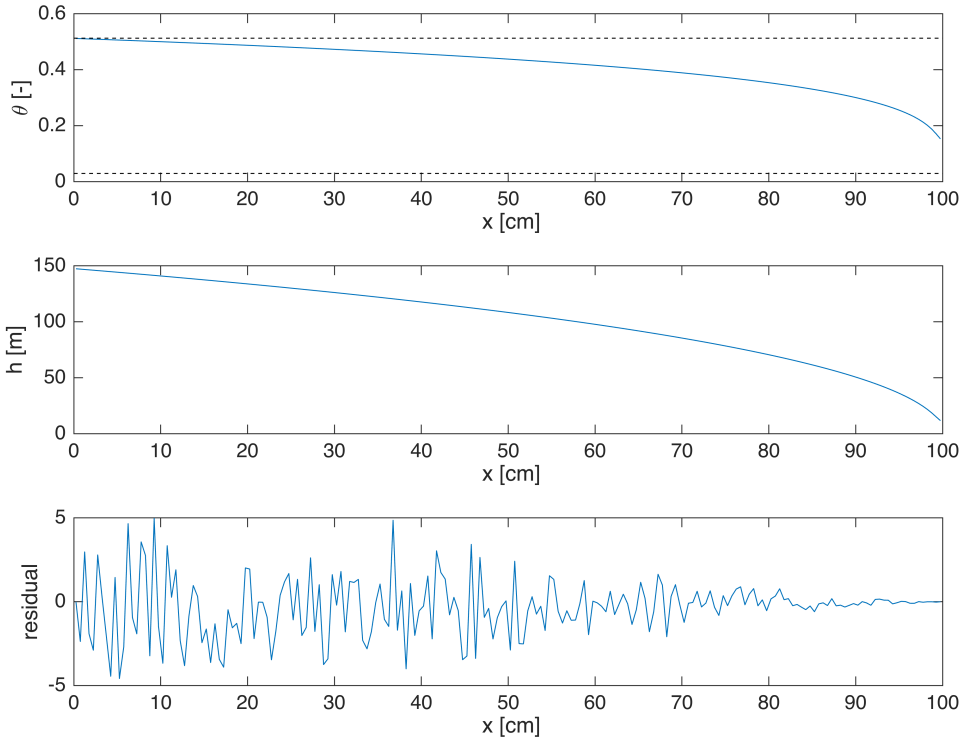
for the Brooks-Corey constitutive function. Hence the Jacobian can easili be evaluated analytically from our discrete operators. This generally speeds up the evaluation of the Jacobian significantly and is also avoids errors due to the finite difference approximation.

Numerical solution with Newtons Method

```
% Rock properties (silt loam - Brooks Corey)
rock.Ks = 30.5; % [cm/d]
rock.theta_s = 0.513;
rock.lambda = 0.54;
rock.theta_r = 0.03;
rock.hb = 1.48*100; % [cm]

% Constitutive functions
sat = @(theta,rock) (theta-rock.theta_r)/(rock.theta_s-rock.theta_r);
DBC = @(theta,rock) rock.Ks*rock.hb/(rock.lambda*(rock.theta_s-
rock.theta_r))*sat(theta,rock).^(2+1/rock.lambda);
% Analytic Drivative of the Hydraulic Diffusivity
dDBC = @(theta,rock) rock.Ks*rock.hb/(rock.theta_s-
rock.theta_r)^2*(1+2*rock.lambda)/rock.lambda^2*sat(theta,rock).^(1+1/
rock.lambda);

%% Build Grid and Operators
theta_0 = rock.theta_r+0.001;
theta_b = rock.theta_s-0.001;

Grid.xmin = 0; Grid.xmax = 100; Grid.Nx = 200;
Grid = build_grid(Grid);

% Build operatror
[D,G,~,I,M] = build_ops(Grid);
Kd = @(theta) comp_mean(DBC(theta,rock),M,1,Grid,1);
L = @(theta) -D*Kd(theta)*G;
IM = @(theta,dt) I + dt*L(theta);
EX = @(theta,dt) I ;
fs = spalloc(Grid.N,1,0);
```

```
% Build BC's
BC.dof_dir   = Grid.dof_xmin;
BC.dof_f_dir = Grid.dof_f_xmin;
BC.g = 0; % set BC in initial guess and then don't update bnd!
BC.dof_neu   = [];
BC.dof_f_neu = [];
BC.qb        = [];
[B,N,fn] = build_bnd(BC,Grid,I);

% IC
theta = theta_0*ones(Grid.N,1);
theta(BC.dof_dir) = theta_b;

%% Time integration
tmax = 1;
Nt = 3e2;
dt = tmax/Nt;

%% Newton iteration
tol = 1e-6; % convergence tolerance
kmax = 20;   % maximum number of iterations
epsilon = 1e-4;
figure
tic
for n = 1:Nt % timestepping loop
    theta_old = theta;
    % Newton-Raphson Iteration
    nres = 1; ndtheta = 1; k = 0;
    while (nres > tol || ndtheta > tol) && k < kmax
        % Compute residual vector and Jacobian matrix
        res = comp_residual_richards(theta,theta_old,D,Kd,G,fs,dt,BC);
        Jac =
comp_jacobian_richards_ana(theta,Grid,D,G,I,M,DBC,dDBC,dt,rock);

        % Solve for update and compute new iterate
        dtheta = solve_lbvp(Jac,-res,B,BC.g,N);
        theta = theta + dtheta;
        res = comp_residual_richards(theta,theta_old,D,Kd,G,fs,dt,BC);
        nres = norm(N'*res); ndtheta = norm(N'*dtheta);

        % Update convergence criteria
        k = k+1;
        % fprintf('it = %d: nres = %3.2e  ndtheta = %3.2e\n',k,nres,ndtheta)
        if k == 1; ndtheta = 0; end % to allow exit on first iteration
    end
    % if k<kmax+1;
    %     fprintf('Newton converged after %d iterations.\n\n',k)
    % else
    %     fprintf('Newton did not converge\n\n')
```

```
    % end
    if mod(n,1) == 0
        subplot 311
        plot(Grid.xc,theta), hold on
        plot([0 100],rock.theta_r*[1 1],'k--')
        plot([0 100],rock.theta_s*[1 1],'k--')
        xlabel ' x [cm]'
        ylabel '\theta [-]'
        hold off

        subplot 312
        plot(Grid.xc,rock.hb*sat(theta,rock).^(1/rock.lambda)),
        xlabel ' x [cm]'
        ylabel 'h [m]'

        subplot 313
        plot(Grid.xc,res)
        xlabel ' x [cm]'
        ylabel 'residual'
        drawnow
    end
end
```




```
toc
```

```
Elapsed time is 58.087400 seconds.
```

Auxillary functions

Function that computes the residual

```
%% Residual Richards Equation (horizontal - no gravity)
function [res] = comp_residual_richards(theta,theta_old,D,Kd,G,fs,dt,BC)
    res = theta - theta_old - dt*(D*Kd(theta)*G)*theta - dt*fs;
    res(BC.dof_dir) = 0;
end
```

Function that computes the Jacobian

```
%% Analytical Jacobian Richards Equation
function [Jac] =
comp_jacobian_richards_ana(theta,Grid,D,G,I,M,Diff,dDiff,dt,rock)
GU = spdiags(G*theta,0,Grid.Nf,Grid.Nf); % Gradient on diagonal
Kd = comp_mean(Diff(theta,rock),M,1,Grid,1);
dKd = spdiags(dDiff(theta,rock),0,Grid.N, Grid.N);
Jac = I - dt*D*(GU*M*dKd + Kd*G);
end
```