Lecture 24: Implicit solution of Richards Equation

Logistics: - HW9 (last HW) is posted due Th 4/18

- HW7 last chance Th 4/18

Last time: Solving non-linear equations

⇒ Horizontal unsaturated flow

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

(θ^n inside $D(\theta)$, θ^{n+1} outside)

non-linear diffusion

- Explicit solution

  works surprisingly well

- Implicit solution

  ⇒ lagging non-linearity

- Fully non-linear

  ⇒ Newton-Raphson Iteration

  Scalar equation

Today: Newton-Raphson for systems of eqns

" " for Richards Equation

Newtons method for systems of Eqns

$\underline{r}(\underline{\theta})$ vector valued vector function

$r(\theta)$ scalar valued scalar function

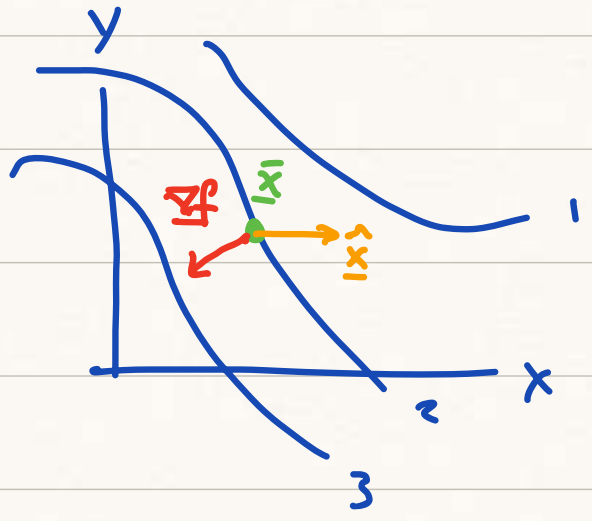
Linearize a function → directional derivative

Example scalar valued vector function:

$$f(\underline{x}) = x + y^2$$

What is change of f in direction $\hat{\underline{x}}$ at $\bar{\underline{x}}$.

$$D_{\hat{\underline{x}}} f(\bar{\underline{x}}) = \nabla f|_{\bar{\underline{x}}} \cdot \hat{\underline{x}}$$



$$\nabla f = \begin{bmatrix} 1 \\ 2y \end{bmatrix}$$

$$D_{\hat{x}} f(\bar{x}) = \begin{bmatrix} 1 & 2\bar{y} \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = \hat{x} + 2\bar{y}\,\hat{y}$$

$\bar{x} = 0 \quad \hat{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$$D_{\hat{x}} f(\bar{x}) = 1$$

Different definition of Directional derivative

$$D_{\hat{x}} f(\bar{x}) = \frac{d}{d\epsilon} f(\bar{x} + \epsilon \hat{x})\Big|_{\epsilon=0} \qquad f = x + y^2$$

$$= \frac{d}{d\epsilon} \bar{x} + \epsilon \hat{x} + (\bar{y} + \epsilon \hat{y})^2 \Big|_{\epsilon=0}$$

$$= \frac{d}{d\epsilon} \bar{x} + \epsilon \hat{x} + \bar{y}^2 + 2\epsilon \bar{y}\hat{y} + \epsilon^2 \hat{y}^2 \Big|_{\epsilon=0}$$

$$= \left(\hat{x} + 2\bar{y}\hat{y} + 2\epsilon \hat{y}^2 \right|_{\epsilon=0}$$

$$D_{\hat{x}} f(\bar{x}) = \hat{x} + 2\bar{y}\hat{y}$$

$$\boxed{D_{\hat{x}} f(\bar{x}) = \nabla f|_{\bar{x}} \cdot \hat{x} = \frac{d}{d\epsilon} f(\bar{x} + \epsilon \hat{x})\Big|_{\epsilon=0}}$$

more general

$f(\underline{x})$ is scalar valued vector function

$\underline{r}(\underline{x})$ is vector valued vector function.

$$D_{\hat{x}} \underline{r}(\bar{x}) = \underline{\nabla \underline{r}}|_{\bar{x}} \hat{x} = \frac{d}{d\epsilon} \underline{r}(\bar{x} + \epsilon \hat{x})\Big|_{\epsilon=0}$$

$\underline{\underline{J}}$ = Jacobian

Newtons method for vector valued vector fun.

$\underline{r}(\underline{u})$ non-linear

Linearize

$$L_{\underline{\bar{u}}}\,\underline{r}(\underline{\hat{u}}) = \underline{r}(\underline{\bar{u}}) + \epsilon D_{\underline{\hat{u}}}\,\underline{r}(\underline{\hat{u}})$$

$$= \underline{r}(\underline{\bar{u}}) + \epsilon \nabla \underline{r}|_{\underline{\bar{u}}}\,\underline{\hat{u}}$$

$$= \underline{r}(\underline{\bar{u}}) + \nabla \underline{r}|_{\underline{\bar{u}}}\,\underline{\Delta u} + \text{h.o.t} \qquad \underline{\Delta u} = \epsilon\,\underline{\hat{u}} \quad (\hat{u}: \text{unit})$$

$$= \underline{r}(\underline{\bar{u}}) + \underline{\underline{J}}(\underline{\bar{u}})\,\underline{\Delta u}$$

Find root of linearization and iterate

$$L_{\underline{u}}\,\underline{r}(\underline{\hat{u}}) = 0$$

$$\Rightarrow \boxed{\underline{\underline{J}}(\underline{\bar{u}})\,\underline{\Delta u} = -\underline{r}(\underline{u})}$$

solve linear system for update $\underline{\Delta u}$

$$\underline{u}^{k+1} = \underline{u}^k + \underline{\Delta u}^k$$

→ Live script example

Newtons method for Richard's Eqn

Horizontal Direction

$$\frac{\partial \theta}{\partial t} - \nabla \cdot [D_H(\theta) \nabla \theta] = f_s$$

Discretize

$$\underline{r} = \underline{\theta}^{n+1} - \underline{\theta}^{n} - \Delta t \, \underline{\underline{D}} * [\underbrace{\underline{\underline{K_d}}(\underline{\theta}^{n+1}) * \underline{\underline{G}} \, \underline{\theta}^{n+1}}_{\text{non-linear} \neq \underline{\underline{L}} \, \underline{\theta}^{n+1}}] - \Delta t \, f_s = 0$$

Jacobian: $\nabla \underline{r} = \frac{d \underline{r}}{d \underline{\theta}} = \underline{\underline{J}}$

$$J_{ij} = \frac{\partial r_i}{\partial \theta_j} = \begin{bmatrix} \frac{\partial r_1}{\partial \theta_1} & \frac{\partial r_1}{\partial \theta_2} & \cdots & \frac{\partial r_1}{\partial \theta_N} \\ \frac{\partial r_2}{\partial \theta_1} & \frac{\partial r_2}{\partial \theta_2} & & \frac{\partial r_2}{\partial \theta_N} \\ \frac{\partial r_3}{\partial \theta_1} & \frac{\partial r_3}{\partial \theta_2} & & \frac{\partial r_3}{\partial \theta_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial r_N}{\partial \theta_1} & \frac{\partial r_N}{\partial \theta_2} & & \frac{\partial r_N}{\partial \theta_n} \end{bmatrix}$$

$$\underline{\underline{J}} = \begin{bmatrix} \frac{d \underline{r}}{d u_1} & \frac{d \underline{r}}{d u_2} & \cdots & \frac{d \underline{r}}{d u_N} \end{bmatrix}$$

Calculating Jacobian is main challenge !

One option is a numerical Jacobian using finite differences
compute one column of $\underline{\underline{J}}$ at a time

General code:

```
[J] = comp_jacobian(r, u, ε)
  n = length(u);  % Grid.N
  u_perturb = u;
  % loop over columns of J
  for i = 1:n
      u_perturb(i) = u_perturb(i) + ε;
      J(:,i) = (r(u_perturb) - r(u)) / ε;  % FD
      u_perturb(i) = u(i);
  end
```

This works for any vectorvalued residual! ∇

If $\underline{r}(\underline{u})$ is complicated it can be defined as separate function.

For time dependent problems need to avoid confusing time level $(\underline{u}^n, \underline{u}^{n+1})$ with the iterates of Newtons method! $(\underline{u}^k, \underline{u}^{k+1})$

- u^n is not involved in Newton
- Newton is iterating to find $\underline{u}^{n+1}$ →

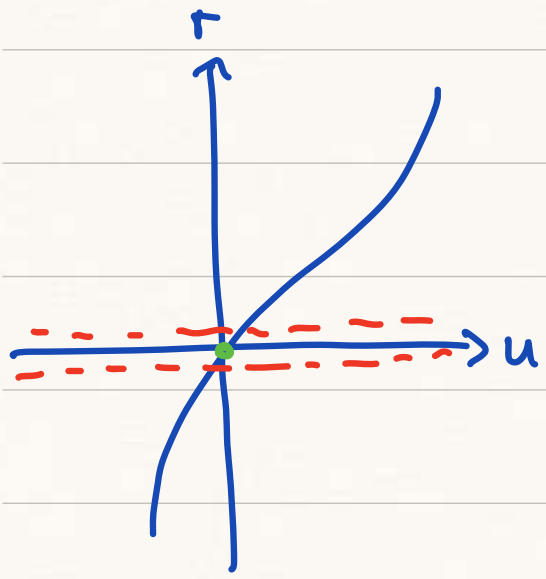
$$\underline{u}^0 = \underline{u}^n \rightarrow \underline{u}^1 \rightarrow \underline{u}^2 \rightarrow \underline{u}^3 \rightarrow \dots \rightarrow \underline{u}^{n+1}$$

General outline for a transient problem

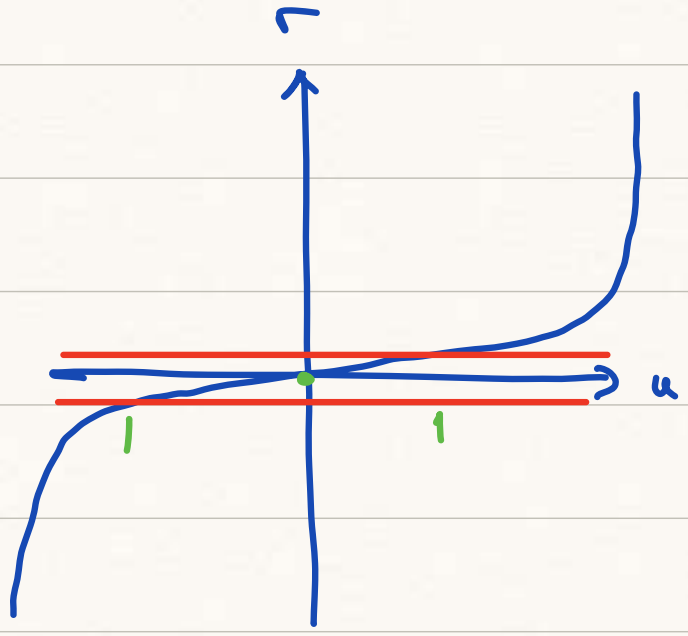
```
for n = 1 : N                    % timestepping loop
    θold = θ;                    % θold = θ^n
    while nres > tol || ndθ > tol && k < kmax      } Newton
        dθ = - J(θ, θold) \ r(θ, θold);
        θ = θ + dθ
    end
end
```

Note: θold is not updated in Newton loop.

Convergence of Newton



$|r| < tol$



$|du| < tol$