Jacobian for transient unconfined flow

Consider the transient unconfined equation

$$\phi_0 h^m \frac{\partial h}{\partial t} - \nabla \cdot \left[\frac{K_0}{n+1} h^{n+1} \nabla h \right] = f_s$$

lets write this as a general non-linear diffusion eqn

$$\boxed{s(u) \frac{\partial u}{\partial t} - \nabla \cdot [f(u) \nabla u] = f_s}$$

where $s(u)$ and $f(u)$ are arbitrary differentiable functions. In the case of unconfined flow $s = \phi_0 u^m$ and $f = \frac{K_0}{n+1} u^{n+1}$.

Discretization with Backward Euler

$$\{\underline{s}(\underline{u}^{n+1})\}_c (\underline{u}^{n+1} - \underline{u}^n) - \Delta t \, \underline{\underline{D}} \left[\{\underline{\underline{M}} \, \underline{f}(\underline{u}^{n+1})\}_f \underline{\underline{G}} \, \underline{u}^{n+1} \right] = \Delta t \, \underline{f_s}$$

Discrete residual: $\underline{u}^{n+1} \equiv \underline{u}$

$$\boxed{\underline{r}(\underline{u}, \underline{u}^n) = \{\underline{s}(\underline{u})\}_c (\underline{u} - \underline{u}^n) - \Delta t \, \underline{\underline{D}} \left[\{\underline{\underline{M}} \, \underline{f}(\underline{u})\}_f \underline{\underline{G}} \, \underline{u} \right] - \Delta t \, \underline{f_s}}$$

Now we have to solve the non-linear problem for $\underline{u} = \underline{u}^{n+1}$ using Newton-Raphson method. We have to be careful to distinguish the k superscripts for the iteration from the n superscripts for the timestep.

<u>Directional derivative</u> $D_{\hat{\underline{u}}}\,\underline{r}(\bar{\underline{u}})$

$$\frac{d}{d\epsilon}\underline{r}(\bar{\underline{u}}+\epsilon\hat{\underline{u}})\Big|_{\epsilon=0} = \frac{d}{d\epsilon}\underbrace{\{s(\bar{\underline{u}}+\epsilon\hat{\underline{u}})\}_c(\bar{\underline{u}}+\epsilon\hat{\underline{u}}-\underline{u}^n)}_{acc} - \underbrace{\Delta t\,\underline{\underline{D}}\left[\{\underline{\underline{M}}f(\bar{\underline{u}}+\epsilon\hat{\underline{u}})\}_f\,\underline{\underline{G}}(\bar{\underline{u}}+\epsilon\hat{\underline{u}})\right]}_{flux} - \cancel{\Delta t f_s}\Big|_{\epsilon=0}$$

acc:

$$\left\{\frac{ds}{du}(\bar{\underline{u}}+\cancel{\epsilon\hat{\underline{u}}})\,\hat{\underline{u}}\right\}_c(\bar{\underline{u}}+\cancel{\epsilon\hat{\underline{u}}}-\underline{u}^n) + \{\underline{s}(\bar{\underline{u}}+\cancel{\epsilon\hat{\underline{u}}})\}_c\,\hat{\underline{u}}\Big|_{\epsilon=0}$$

$$\left\{\frac{ds}{du}(\bar{\underline{u}})\,\hat{\underline{u}}\right\}_c(\bar{\underline{u}}-\underline{u}^n) + \{\underline{s}(\bar{\underline{u}})\}_c\,\hat{\underline{u}} = \underbrace{\left\{\frac{ds}{du}\right\}_c}_{\underline{\underline{dS}}}\{\bar{\underline{u}}-\underline{u}^n\}_c\,\hat{\underline{u}} + \underbrace{\{\underline{s}(\bar{\underline{u}})\}_c}_{\underline{\underline{S}}}\,\hat{\underline{u}}$$

$$acc = \left[\underline{\underline{dS}}\,\{\bar{\underline{u}}-\underline{u}^n\}_c + \underline{\underline{S}}\right]\hat{\underline{u}}$$

flux:

$$\Delta t\,\underline{\underline{D}}\left[\left\{\underline{\underline{M}}\frac{df}{du}(\bar{\underline{u}}+\cancel{\epsilon\hat{\underline{u}}})\,\hat{\underline{u}}\right\}_f\underline{\underline{G}}(\bar{\underline{u}}+\cancel{\epsilon\hat{\underline{u}}}) + \{\underline{\underline{M}}\,\underline{f}(\bar{\underline{u}}+\cancel{\epsilon\hat{\underline{u}}})\}_f\,\underline{\underline{G}}\,\hat{\underline{u}}\right]\Big|_{\epsilon=0}$$

$$\Delta t\,\underline{\underline{D}}\left[\underbrace{\{\underline{\underline{G}}\,\bar{\underline{u}}\}_f}_{\underline{\underline{Gu}}}\underbrace{\left\{\underline{\underline{M}}\frac{df}{du}(\bar{\underline{u}})\,\hat{\underline{u}}\right\}_f}_{?} + \underbrace{\{\underline{\underline{M}}\,\underline{f}(\bar{\underline{u}})\}_f}_{\underline{\underline{F}}}\,\underline{\underline{G}}\,\hat{\underline{u}}\right]$$

$$\left\{\underline{\underline{M}}\frac{df}{du}(\bar{\underline{u}})\,\hat{\underline{u}}\right\}_f = \underline{\underline{M}}\left\{\frac{df}{du}(\bar{\underline{u}})\,\hat{\underline{u}}\right\}_c = \underline{\underline{M}}\left\{\frac{df}{du}(\bar{\underline{u}})\right\}_c\hat{u} = \underline{\underline{M}}\,\underline{\underline{dF}}\,\underline{u}$$

$$flux = \Delta t\,\underline{\underline{D}}\left[\underline{\underline{Gu}}\,\underline{\underline{M}}\,\underline{\underline{dF}} + \underline{\underline{F}}\,\underline{\underline{G}}\right]\hat{\underline{u}}$$

So that we have:

$$D_{\underline{\hat{u}}}\underline{r}(\underline{\bar{u}}) = \Big\langle \underbrace{\underline{\underline{dS}}\{\underline{\bar{u}} - \underline{u}^n\}_c + \underline{\underline{S}} - \Delta t\, \underline{\underline{D}}\left[\underline{\underline{G\bar{U}}}\,\underline{\underline{M}}\,\underline{\underline{dF}} + \underline{\underline{F}}\,\underline{\underline{G}}\right]}_{\text{Jacobian}} \Big\rangle\, \underline{\hat{u}}$$

Hence the Jacobian for unconfined flow is

$$\boxed{\underline{\underline{J}}(\underline{\bar{u}}, \underline{u}^n) = \underline{\underline{dS}}\{\underline{\bar{u}} - \underline{u}^n\}_c + \underline{\underline{S}} - \Delta t\, \underline{\underline{D}}\left[\underline{\underline{dF}}\{\underline{\underline{G}}\,\underline{\bar{u}}\}_f + \underline{\underline{F}}\,\underline{\underline{G}}\right]}$$

Implementation pitfalls:

In a transient problem we have <u>three</u> versions of the unknown vector $\underline{u}$. Both residual and Jacobian are functions of

$$\left.\begin{array}{l} \underline{r} = \underline{r}(\underline{u}, \underline{u}^n) \\[2ex] \underline{\underline{J}} = \underline{\underline{J}}(\underline{u}, \underline{u}^n) \end{array}\right\} \quad \begin{array}{l} \underline{u}^n = \text{solution at last time step} \\ \underline{u} = \underline{u}^k = \text{current iterate of solution at new} \\ \qquad\qquad \text{time step} \end{array}$$

These two variables are connected in that $\underline{u}^k$ is initalized as $\underline{u}^0 = \underline{u}^n$ at the beginning of each Newton-Raphson iteration

$\Rightarrow$ common error is to confuse $\underline{u}^n$ and $\underline{u}^k$

General code outline for transient non-linear problem:

```
for  n = 1 : N          % time stepping loop

     uold = u ;         % save soln from last time step  uold = u^n

     while  nres > tol || ndu > tol ||  k < kmax

            du  = - J(u , uold)^-1 res(u, uold) ;

            u = u + du ;

     end

end
```