We choose to solve the NS equations w/r/t different state variables:

$$\underset{\sim}{U} = \begin{bmatrix} \rho \\ \rho u_i \\ \rho e \end{bmatrix} \qquad \underset{\sim}{Y} = \begin{bmatrix} p \\ u_i \\ T \end{bmatrix}$$

Regardless of your choice, we need to calculate the same quantities to evaluate the residual:

- Advective Flux (inviscid)
- Diffusive flux (stress)

To allow us to write one QFunction for different state variables, we define QFs using a generic State struct:

$$\text{State} = \begin{cases} \underset{\sim}{U} \\ \underset{\sim}{Y} \end{cases}$$

In C (and other languages) a struct is a collection of data into a single object

We have functions that create these State structs from raw state vectors called "StateFromQi". Other helper functions use State structs as inputs.

In addition to solution state, we need solution derivatives. To convert derivatives of specific state variables to derivatives of our generic State, we define derivative versions of our conversion functions. These functions are denoted with a "_fwd" at the end.

ex. "StateFromQi_fwd"

Total Derivatives (NOT = material derivative, but related)

Derivatives in libCEED-Fluids use total derivative notation. Inspired from automatic differentiation, but found in thermodynamics and (indirectly) in fluid mechanics.

Def. We calculate the total change of a quantity w/rt the change of its inputs using chain rule. This general form can be used to calculate more specific quantities.

Example: Given $f(x_i)$, the total derivative is

$$df(x_j) = f_{,x_i}(x_j)\, dx_i$$

So the change in f (df) is equal to the change of its inputs (dx_i) contracted with the Jacobian of f ($f_{,x_i}$).

Functions of functions

Given $f(g(x))$, we'd write

$$df = f_{,g}(g(x))\, dg \qquad \Rightarrow \qquad df = f_{,g}(g(x))\, \underbrace{g_{,x}(x)\, dx}_{dg}$$

$$dg = g_{,x}(x)\, dx$$

Given $f(x_i)$, what is df/dx_1.

substitute dx_i with $dx_i/dx_1 = \delta_{i1}$

$$\frac{df}{dx_1} = f_{,x_i}(x_j)\frac{dx_i}{dx_1} = f_{,x_i}(x_j)\,\delta_{i1} = f_{,x_1}(x_j)$$

Fluids example: We have $\underset{\sim}{U}$, want u_i (inside StateFromU)

$$u_i = f(\underset{\sim}{U}) = m_i/\rho$$

Derivative: $du_i(\underset{\sim}{U}, d\underset{\sim}{U}) = du_i(m_i, \rho, dm_i, d\rho)$

$$du_i = f_{,m_i}(m_i,\rho)\, dm_i + f_{,\rho}(m_i,\rho)\, d\rho$$

$$= \left[1/\rho\right] dm_i + \left[-m_i/\rho^2\right] d\rho$$

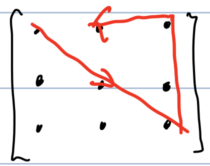
$$\Rightarrow \quad du_i = \frac{dm_i - u_i\, d\rho}{\rho}$$

To compute velocity gradient, $d\underset{\sim}{U} = \underset{\sim}{U}_{,j}$ $\quad (dm_i = m_{i,j} \quad d\rho = \rho_{,j})$. Plugging in

$$u_{i,j} = \frac{m_{i,j} - u_i \rho_{,j}}{\rho}$$

Kelvin-Mandel Notation:

Way to write symmetric matrices as a vector.



$$\{u_{11},\ u_{22},\ u_{33},\ \sqrt{2}\,u_{32},\ \sqrt{2}\,u_{31},\ \sqrt{2}\,u_{21}\}$$

QFunctions have similar form:

1. Sort inputs and outputs into arrays
2. Quadrature point loop
   a) Process inputs per quadrature point
   b) Convert input solution (state) vector into State struct
   c) Calculate our residual quantities
   d) Populate outputs with partial residuals