How to determine which phase?

If T,p given: use 97
if T > Tsat(p) —> superheated
if T < Tsat(p) —> subcooled
if P < Psat(T) —> superheated
if P > Psat(T) —> subcooled

if v,u,h,s < vf,uf,hf,sf —> subcooled
if v,u,h,s > vg,ug,hg,sg —> superheated
if vf,uf,hf,sf < v,u,h,s < vg,ug,hg,sg —> two phase

if T>Tcrit and P>Pcrit —> supercritical
if T>Tcrit and P<Pcrit —> superheated
if T<Tcrit and P>Pcrit —> subcooled

```
//state
public double pressure;       //p //pascals
public double temperature;    //t //kalvin
public double volume;         //v //M^3/kg
public double internalenergy; //u //J/kg
public double entropy;        //s //J/kgK
public double enthalpy;       //h //J/kg
public double quality;        //q //%

//static properties of system
public double mass = 1; //kg
public double radius = 0.05; //M
//public double surfacearea = Math.Pow(3.141592*radius,2.0); //M^2 //hardcoded answer below
public double surfacearea = 0.024674011; //M^2 //hardcoded answer to eqn above
public double surfacearea_insqr = 38.2447935395871; //in^2 //hardcoded conversion from m^2 to in^2

//assume starting/ending point consistent for whole API!

public void add_heat_constant_p(double j)
{
  int region = ThermoMath.region_given_pvt(pressure,volume,temperature);

  //default guess
  double new_u = internalenergy;
  double new_v = volume;
  double new_t = temperature;

  switch(region)
  {
    case 1: //two-phase region
    {
      //temperature doesn't change!
      // ============ THERMO QUESTION ============
      //the current scenario is "in the two-phase region, we add heat with constant pressure"
      //TODO: I had previously written "no change between regions", and implemented all regions at once.
however, in 2-phase, we need to find quality- so how to do that?
      //TODO: how do we find everything else? (apparently requries quality?)
    }
    break;
    case 0: //subcooled liquid
    case 2: //superheated vapor
    {
      new_u = internalenergy + j;      Does j = heat*time/mass?
      { new_h = enthalpy + heat*time/mass
        //iterative process to find GUESS from MARK
        int MAX_ITERS = 100;
        double MAX_DELTA = 0.01;
        double step = 0.01;
        double guess = new_t;
        double mark = new_u;
        double delta = Math.Abs(ThermoMath.u_given_pt(pressure,guess)-mark);
        for(int i = 0; i < MAX_ITERS || delta < MAX_DELTA; i++)
```

x = quality
x = (u - uf)/(ug - uf)
use 97: uf = uliq_p(p), ug = uvap_p(p)
x = (v - vf)/(vg - vf)
use 97: vf = vliq_p(p), vg = vvap_p(p)
x = (h - hf) / (hg - hf)
use 97: hf = hliq_p(p), hg = hvap_p(p)
x = (s - sf) / (sg - sf)
use 97: sf = sliq_p(p), sg = svap_p(p)

```
        {
         double delta_a = Math.Abs(ThermoMath.u_given_pt(pressure,guess+ step     )-mark);
         double delta_b = Math.Abs(ThermoMath.u_given_pt(pressure,guess-(step/2.0))-mark);
         if(delta_a < delta_b)
         {
          delta = delta_a;
         }
         else
         {
          delta = delta_b;
          step = step/-2.0;
         }
         guess += step;
        }
        new_t = guess;
       }
       new_v = ThermoMath.v_given_pt(pressure,new_t);
     }
     break;
   }

   //at this point, we have enough internal state to derive the rest
   volume = new_v;
   internalenergy = new_u;
   temperature = new_t;
   enthalpy = ThermoMath.h_given_vt(volume,temperature);
   entropy = ThermoMath.s_given_vt(volume,temperature);

   transform_to_state();
 }

 public void add_heat_constant_v(double j)
 {
   //no difference between regions
   double new_t = temperature;
   double new_u = internalenergy;

   new_u = internalenergy+(j/mass);    need to be consistent with eq on first page

   {
    //iterative process to find GUESS from MARK
    int MAX_ITERS = 100;
    double MAX_DELTA = 0.01;
    double step = 0.01;
    double guess = new_t;
    double mark = new_u;
    double delta = ThermoMath.u_given_vt(volume,guess)-mark;
    for(int i = 0; i < MAX_ITERS || delta < MAX_DELTA; i++)
    {
     double delta_a = Math.Abs(ThermoMath.u_given_vt(volume,guess+ step     )-mark);
     double delta_b = Math.Abs(ThermoMath.u_given_vt(volume,guess-(step/2.0))-mark);
     if(delta_a < delta_b)
```

```
      {
       delta = delta_a;
      }
      else
      {
       delta = delta_b;
       step = step/-2.0;
      }
      guess += step;
     }
     new_t = guess;
    }

    //at this point, we have enough internal state to derive the rest
    internalenergy = new_u;
    temperature = new_t;
    pressure = ThermoMath.p_given_vt(volume,temperature);
    enthalpy = ThermoMath.h_given_vt(volume,temperature);
    entropy = ThermoMath.s_given_vt(volume,temperature);
    // ============ THERMO QUESTION ============
    //TODO: how do we find quality? [ONLY RELEVANT FOR 2-PHASE REGION!]

    transform_to_state();
  }

  public void add_pressure_uninsulated(double p)
  {
    int region = ThermoMath.region_given_pvt(pressure,volume,temperature);
    double new_p = pressure+p;
    double new_u = ThermoMath.u_given_pt(new_p,temperature)

    switch(region)
    {
     case 1: //two-phase region
     {
      // ============ THERMO QUESTION ============
      //TODO: ensure this was calculated correctly
      quality = (new_u-internalenergy) + pressure*volume*Math.Ln(new_v/volume);
     }                    new_u = u + heat*time/mass - pressure*volume*Math.Ln(new_v/volume)
     break;
     case 0: //subcooled liquid
     case 2: //superheated vapor
     {
      //already done!
     }
     break;
    }

    // ============ THERMO QUESTION ============
    //TODO: what is the vBck[2] = ... line in Scenario 5.pdf?
    //aren't all variables accounted for?        You can ignore this statement - I was checking something for myself
```

```
  //at this point, we have enough internal state to derive the rest
  pressure = new_p;
  internalenergy = new_u;
  volume = ThermoMath.v_given_pt(pressure,temperature);
  enthalpy = ThermoMath.h_given_vt(volume,temperature);
  entropy = ThermoMath.s_given_vt(volume,temperature);

  transform_to_state();
 }

 public void add_pressure_insulated(double p)
 {
  int region = ThermoMath.region_given_pvt(pressure,volume,temperature);
  double new_p = pressure+p;

  switch(region)
  {
   case 0: //subcooled liquid
   {
    //at this point, we have enough internal state to derive the rest
    pressure = new_p;
    //temperature/volume insignificantly changed
    // ============ THERMO QUESTION ============
    //TODO: any change to internalenergy?
    volume = ThermoMath.v_given_pt(pressure,temperature);
    enthalpy = ThermoMath.h_given_vt(volume,temperature);
    entropy = ThermoMath.s_given_vt(volume,temperature);
   }
   break;
   case 1: //two-phase region
   {
    //IGNORE BECAUSE UNSURE HOW TO CALCULATE!
    //PROBLEM: BOTH T AND V CHANGE
    pressure = new_p;
    // ============ THERMO QUESTION ============
    //TODO: any ideas how to stub with something functional-but-imperfect?
    //  idea: solve once for V holding T constant, then reset, solve for T holding V constant at 1/2 way between
previous value and value calculated assuming T was constant?
    //  assuming we had V and T- how to find quality?
   }
   break;
   case 2: //superheated vapor
   {
    //default guess
    double new_t = temperature;
    double new_u = internalenergy;
    double new_v = volume;

    {
     double k = 1.27;
     new_u = internalenergy-((new_p*new_v-pressure*volume)/(1-k));
```

```
      {
       //iterative process to find GUESS from MARK
       int MAX_ITERS = 100;
       double MAX_DELTA = 0.01;
       double step = 0.01;
       double guess = new_t;
       double mark = new_u;
       double delta = Math.Abs(ThermoMath.u_given_pt(pressure,guess)-mark);
       for(int i = 0; i < MAX_ITERS || delta < MAX_DELTA; i++)
       {
        double delta_a = Math.Abs(ThermoMath.u_given_pt(pressure,guess+ step     )-mark);
        double delta_b = Math.Abs(ThermoMath.u_given_pt(pressure,guess-(step/2.0))-mark);
        if(delta_a < delta_b)
        {
         delta = delta_a;
        }
        else
        {
         delta = delta_b;
         step = step/-2.0;
        }
        guess += step;
       }
       new_t = guess;
      }

      new_v = ThermoMath.v_given_pt(new_p, new_t);
     }

     //at this point, we have enough internal state to derive the rest
     pressure = new_p;
     volume = new_v;
     temperature = new_t;
     internalenergy = new_u;
     enthalpy = ThermoMath.h_given_vt(volume,temperature);
     entropy = ThermoMath.s_given_vt(volume,temperature);
    }
    break;
   }

  transform_to_state();
}
```