Modeling Thermochemical Biomass Pyrolysis in Python

Gavin Wiggins

PyKnoxville 4/8/2015

Biomass

- switchgrass, wood, crops, food waste
- this presentation refers to woody biomass

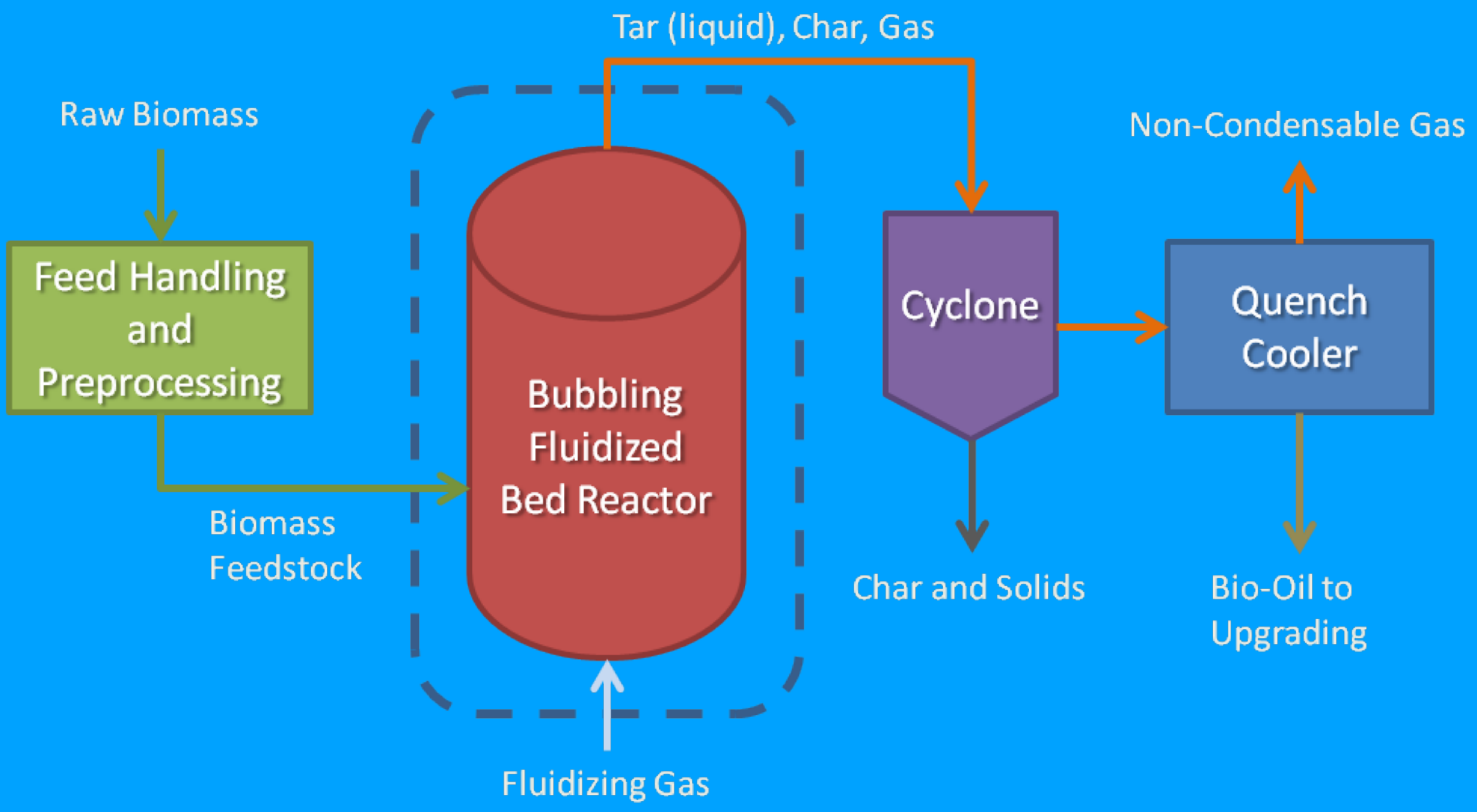
Pyrolysis

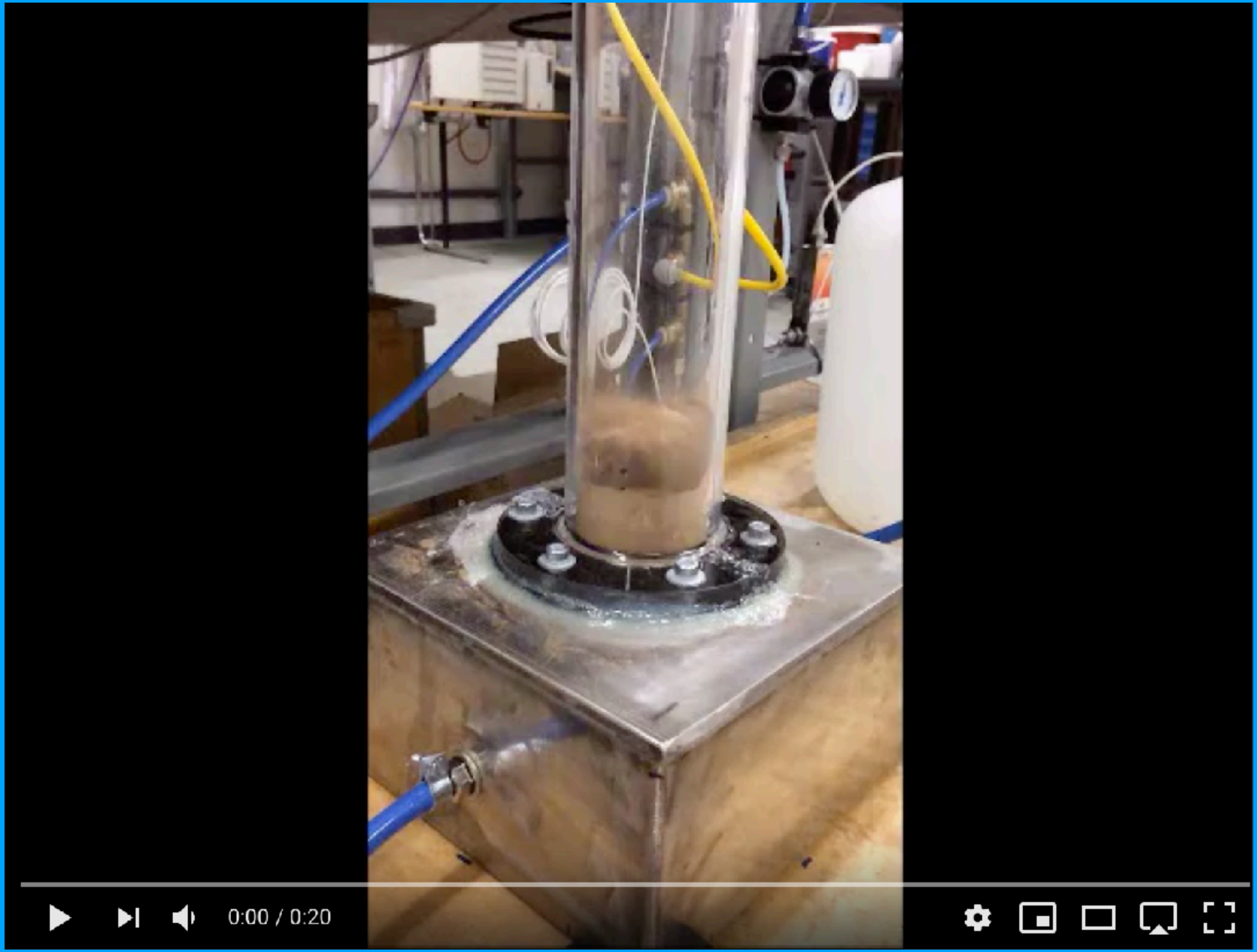
- thermochemical conversion of biomass in the absence of oxygen
- produces char, gas, liquids (condensable vapors)
- fast pyrolysis focuses on liquid (tar) production

Bio-oil

- liquid product from pyrolysis
- higher energy density than biomass
- upgraded to gasoline / diesel grade fuels
- high commodity chemicals

Overview of Pyrolysis Process





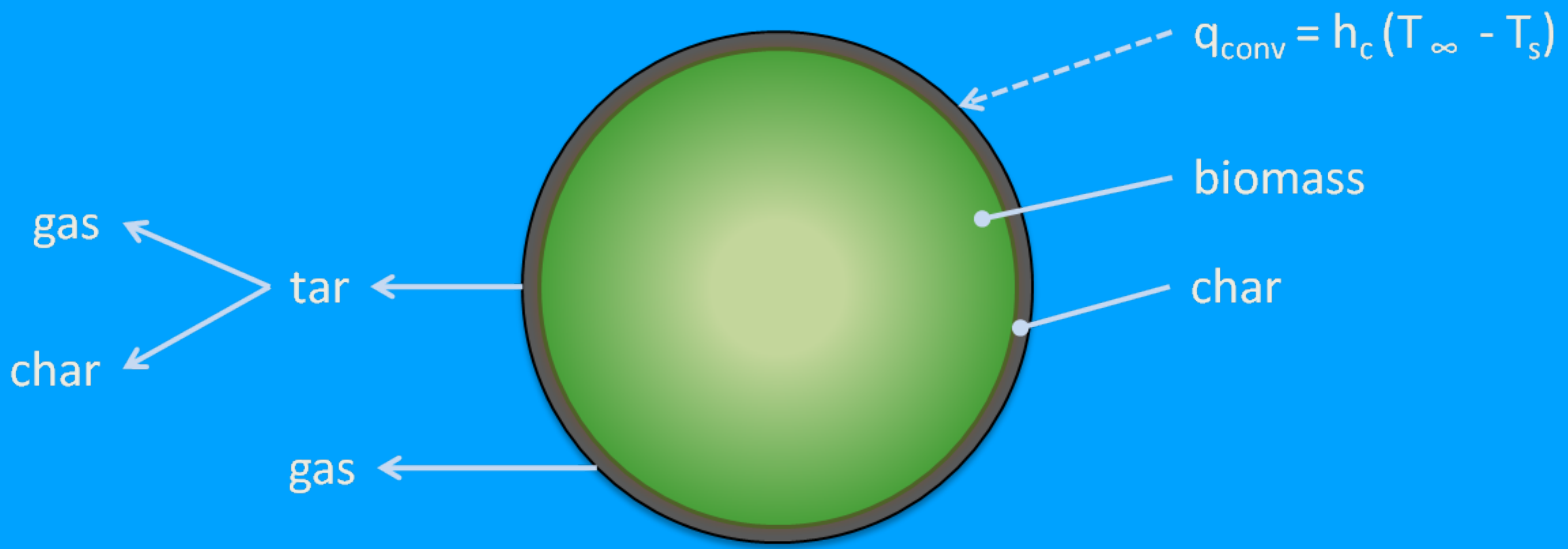
https://youtu.be/dcBAqLR65Hg

Pyrolysis of a biomass particle

Kinetic reactions

$$K_i = A_i + e^{-E_i/(RT)}$$

$$\frac{dC_i}{dt} = K_i C_i$$



T_∞ = 425-600°C (698-873K)

Transient heat conduction

$$\rho C_p \frac{\partial T}{\partial t} = \frac{1}{r^b} \frac{\partial}{\partial r} \left(k r^b \frac{\partial T}{\partial r} \right) + g$$

$$k \frac{\partial T}{\partial t} \bigg|_{r=R} = h \left(T_\infty - T_R \right)$$

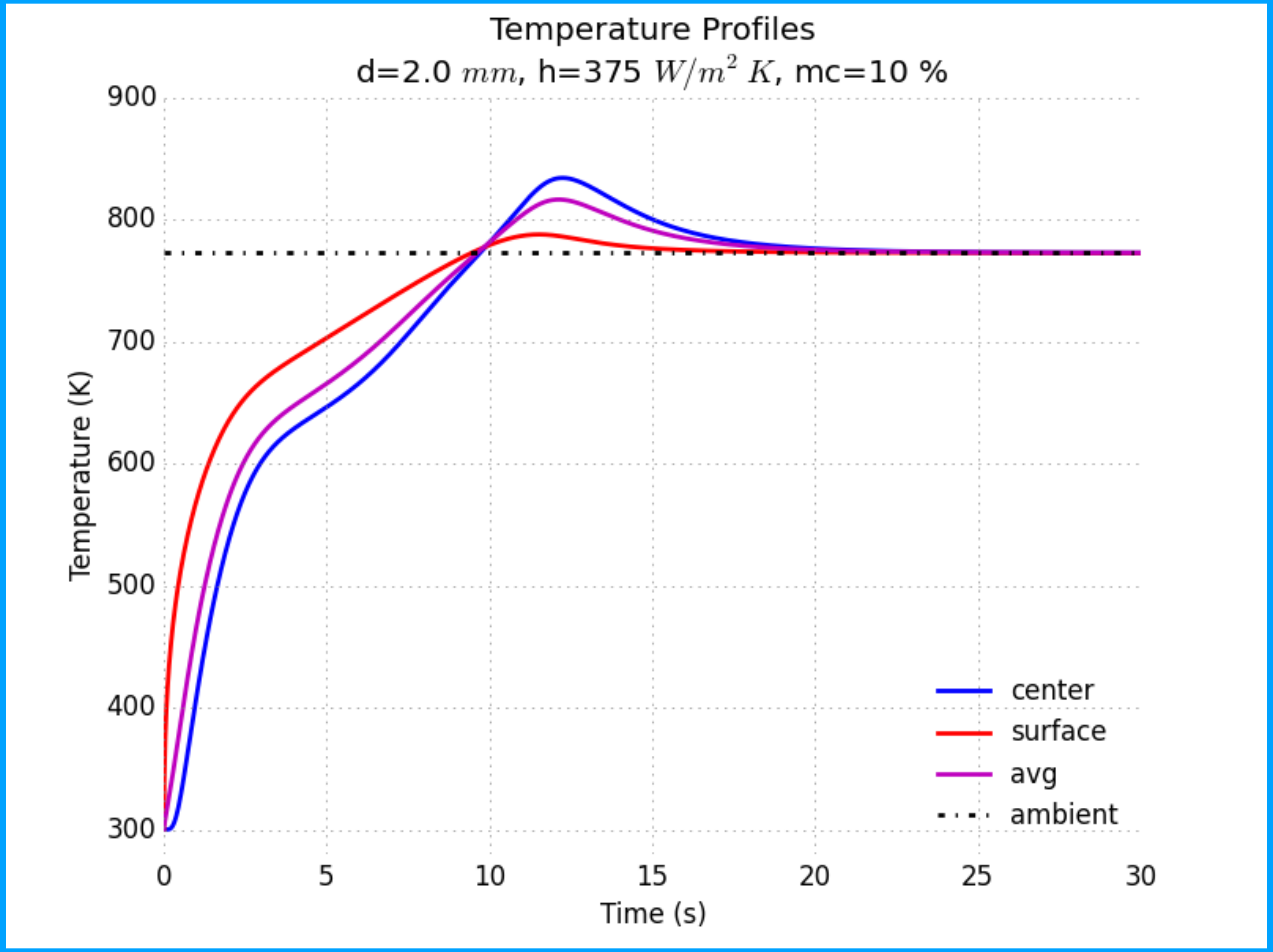
Scientific Python

- Python 3
- **SciPy** mathematical solvers
- **NumPy** n-dimensional array package
- **Matplotlib** comprehensive array package
- **Pandas** data structures and analysis
- **iPython** interactive console
- **Spyder** a Python IDE similar to Matlab

Anaconda

Completely free Python distribution for data processing, analytics, and scientific computing on Linux, Windows, and Mac. Download from Continuum Analytics.

1-D Transient Heat Conduction



```python
def hc(m, dr, b, dt, h, Tinf, g, T, i, r, pbar, cpbar, kbar, ab, bb, k, ri, rminus12, rplus12):

    v = dt / (pbar[0] * cpbar[0])

    # create internal terms
    kminus12 = (kbar[k] + kbar[k-1])/2
    kplus12  = (kbar[k] + kbar[k+1])/2
    w = dt / (pbar[k] * cpbar[k] * ri * (dr**2))
    z = dt / (pbar[k] * cpbar[k])

    # create surface terms
    ww = dt / (pbar[m-1] * cpbar[m-1])
    krminus12 = (kbar[m-1] + kbar[m-2])/2

    # upper diagonal
    ab[0, 1] = -(2 * v * kbar[0] * (1+b)) / (dr**2)      # center node T1
    ab[0, 2:] = -w * rplus12 * kplus12                   # internal nodes Tm+1

    # center diagonal
    ab[1, 0] = 1 + (2* v * kbar[0] * (1+b)) / (dr**2)                      # center node T0
    ab[1, 1:m-1] = 1 + w * rminus12 * kminus12 + w * rplus12 * kplus12     # internal nodes Tm
    ab[1, m-1] = 1 + (2*ww/(dr**2)) * krminus12 + ww * ((2/dr) + (b/r))*h   # surface node Tr

    # lower diagonal
    ab[2, 0:m-2] = -w * rminus12 * kminus12     # internal nodes Tm-1
    ab[2, m-2] = -(2*ww/(dr**2)) * krminus12    # surface node Tr-1

    # column vector
    bb[0] = T[i-1, 0] + v*g[0]                                   # center node T0
    bb[1:m-1] = T[i-1, k] + z*g[k]                               # internal nodes Tm
    bb[m-1] = T[i-1, m-1] + ww*((2/dr)+(b/r))*h*Tinf + ww*g[m-1]  # surface node Tr

    return ab, bb
```

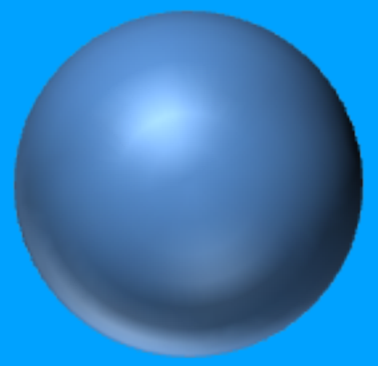
Kinetic Chemical Reactions

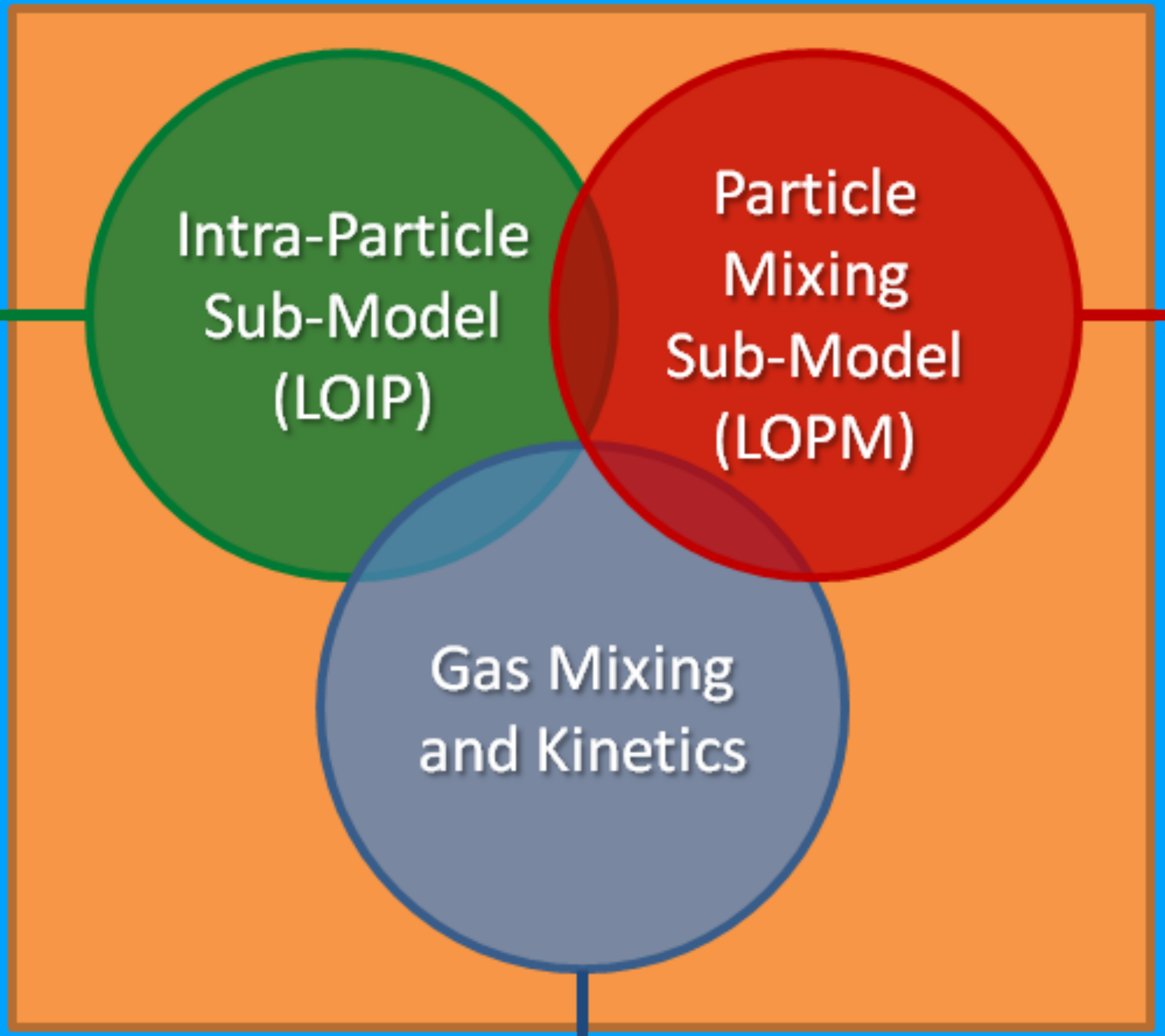
```python
def kn4(T, pw, pc, pg, pt, pwa, pva, dt, i, H):
    R = 0.008314 # universal gas constant, kJ/mol*K

    # A = pre-factor (1/s) and E = activation energy (kJ/mol)
    A1 = 1.3e8;      E1 = 140    # wood -> gas
    A2 = 2e8;        E2 = 133    # wood -> tar
    A3 = 1.08e7;     E3 = 121    # wood -> char
    A4 = 4.28e6;     E4 = 108    # tar -> gas
    A5 = 1e6;        E5 = 108    # tar -> char
    Aw = 5.13e6;     Ew = 87.9   # water -> vapor

    # evaluate reaction rate constant for each reaction, 1/s
    K1 = A1 * np.exp(-E1 / (R * T[i]))  # wood -> gas
    K2 = A2 * np.exp(-E2 / (R * T[i]))  # wood -> tar
    K3 = A3 * np.exp(-E3 / (R * T[i]))  # wood -> char
    K4 = A4 * np.exp(-E4 / (R * T[i]))  # tar -> gas
    K5 = A5 * np.exp(-E5 / (R * T[i]))  # tar -> char
    Kw = Aw * np.exp(-Ew / (R * T[i]))  # water -> vapor

    # determine reaction rate for each reaction, rho/s
    rww = -(K1+K2+K3) * pw[i-1]     # wood rate
    rwg = K1 * pw[i-1]              # wood -> gas rate
    rwt = K2 * pw[i-1]              # wood -> tar rate
    rwc = K3 * pw[i-1]              # wood -> char rate
    rtg = K4 * pt[i-1]              # tar -> gas rate
    rtc = K5 * pt[i-1]              # tar -> char rate
    rwa = -Kw * pwa[i-1]            # rate of water vaporization
    rva = Kw * pwa[i-1]             # rate of water -> vapor

    # update wood, char, gas concentration as a density, kg/m^3
    pww = pw[i-1] + rww*dt                  # wood
    pgg = pg[i-1] + (rwg + rtg)*dt          # gas
    ptt = pt[i-1] + (rwt - rtg - rtc)*dt    # tar
    pcc = pc[i-1] + (rwc + rtc)*dt          # char
    pwwa = pwa[i-1] + rwa*dt                # water
    pvva = pva[i-1] + rva*dt                # vapor

    # calculate heat of generation term
    Hv = 2260000        # heat of vaporization, J/kg
    g = H*rww + Hv*rwa  # heat generation, W/m^3

    # return the wood, char, gas, tar concentration and the heat generation
    return pww, pcc, pgg, ptt, pwwa, pvva, g
```

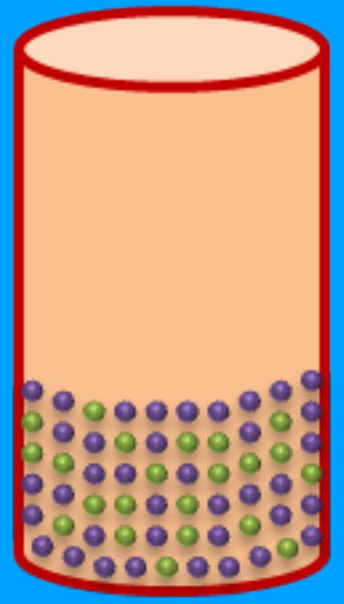
Evaluate intra-particle heat conduction and kinetic reactions at each time step

```python
for i in range(1, nt+1):

    # heat transfer
    ab, bb = hc(m, dr, b, dt, h, Tinf, g, T, i, r, pbar, cpbar, kbar, ab, bb, k, ri, rminus12, rplus12)
    T[i] = sp.solve_banded((1, 1), ab, bb)

    # kinetic reactions
    pw[i], pc[i], pg[i], pt[i], pwa[i], pva[i], g = kn4(T, pw, pc, pg, pt, pwa, pva, dt, i, H)

    # update thermal properties
    cpw = 1112.0 + 4.85 * (T[i] - 273.15)
    kw = 0.13 + (3e-4) * (T[i] - 273.15)
    cpc = 1003.2 + 2.09 * (T[i] - 273.15)
    kc = 0.08 - (1e-4) * (T[i] - 273.15)

    # update mass fraction vector
    Yw = pw[i] / rhow
    Ywa = pwa[i] / rhowa
    cpbar = Yw*cpw + (1-Yw)*cpc + Ywa*cpwa
    kbar = Yw*kw + (1-Yw)*kc + Ywa*kwa
    pbar = pw[i] + pc[i]
    Ys[i] = np.mean(pbar) / rhow
```

Integrated Reactor Model

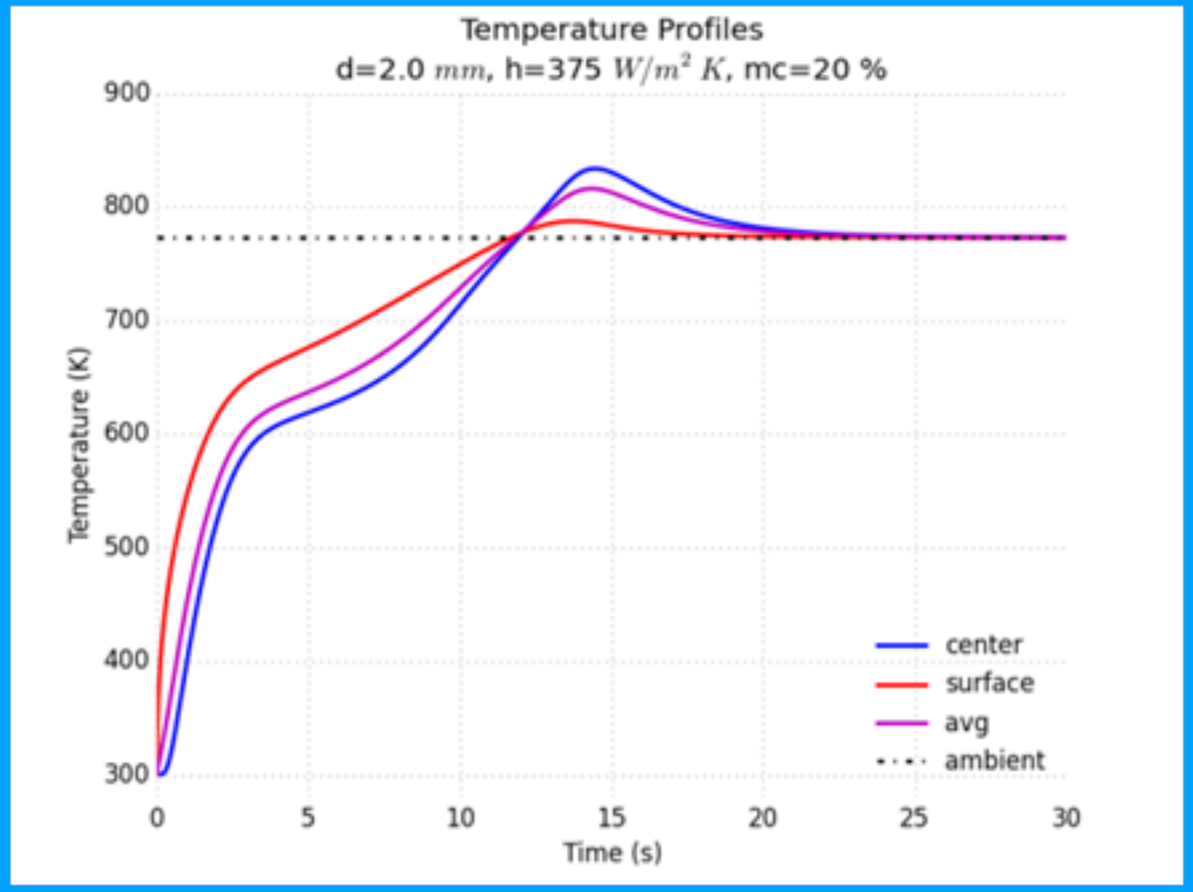




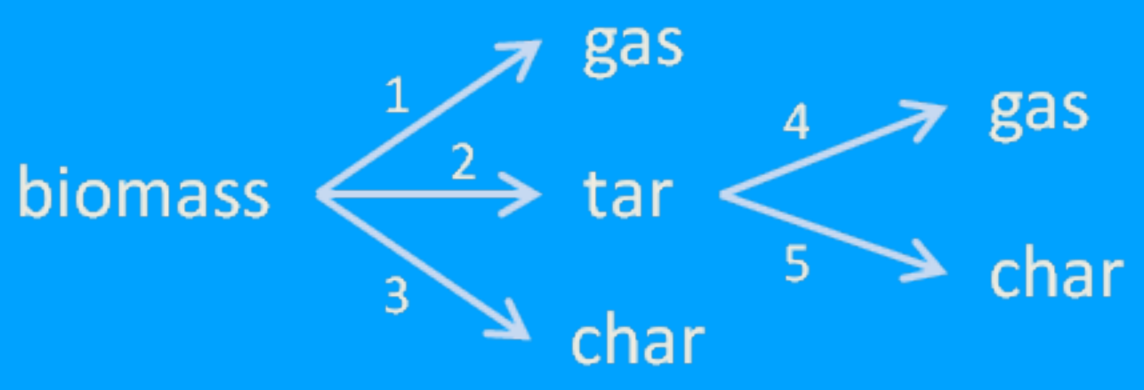


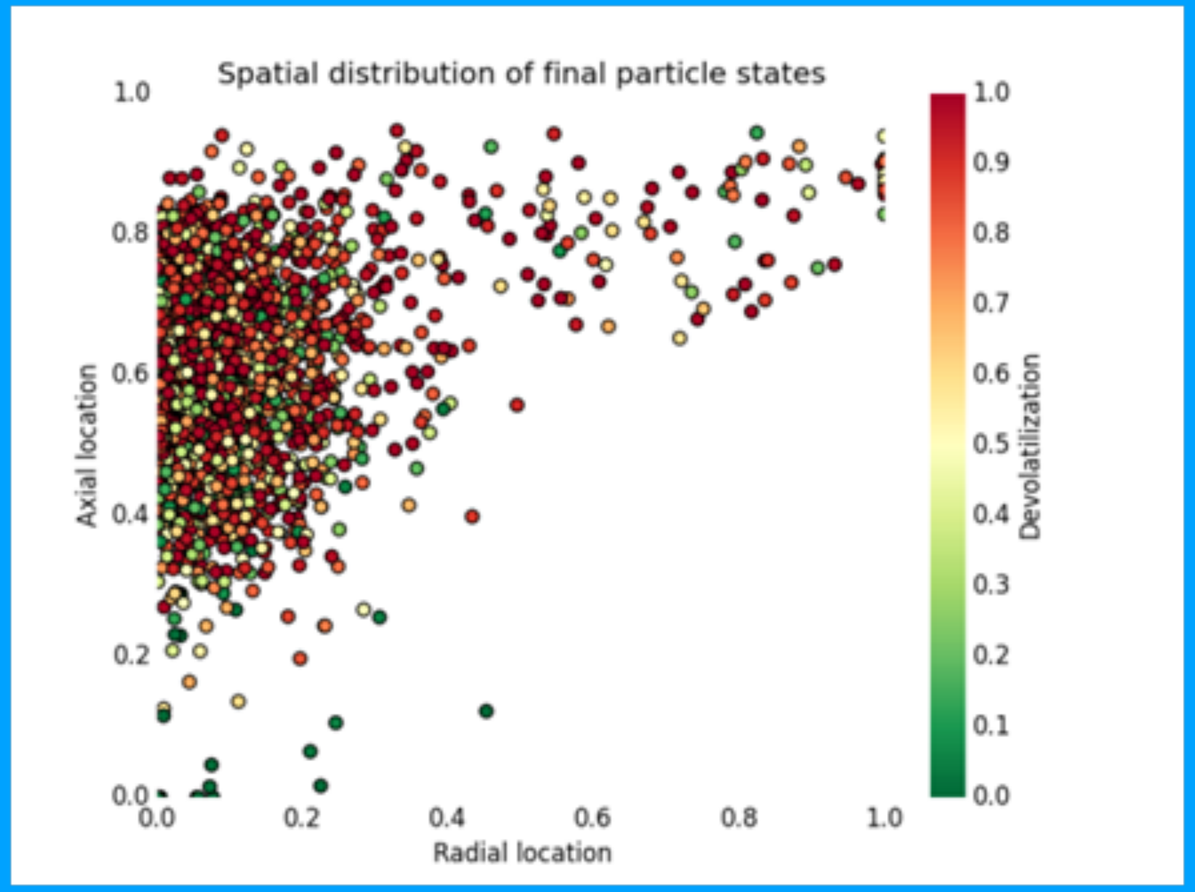
- heat transfer
- mass transport
- thermal properties
- physical characteristics
- moisture content

- fluidization scheme
- residence times
- reactor height / diameter
- bed height
- feedstock injection



- chemical species
- devolatilization rate
- liquid yield
- secondary reactions





Computational Pyrolysis Consortium (CPC)

http://cpcbiomass.org

CPC on GitHub

https://github.com/pyrolysis

Knoxville CocoaHeads

http://knoxcocoa.github.io