Research Meeting

2024.02.06

Subin Kim

01 Febuary 1st

Thanks for Minseo Kang

- Code study with paper - Sina Amini Niaki, PINN for modeling thermochemical curing process of composite-tool systems during manufacture(2021)

02 Paper - Introduction

Sina Amini Niaki, PINN for modeling thermochemical curing process of composite-tool systems during manufacture(2021)

- Key words: PINN, Deep learning, Composites processing …

Manufacturing polymer composite materials → Thermochemical process in coupled nonlinear PDEs (heat conduction & resin cure kinetics) → Solution needs a computational estimation

Not available in closed form

- FEM: each point in time, iteration
> computational cost ↑↑

To computational efficiency↑
= To fixed computational cost

↓

Needs

① Flexible PDE
② Loss objection

① DNN
: needs parallerlized hardware, automatic differenriation and stoachastic optimization

② Need separate high fidelity model
- FEM: needs high fidelity data
- PINN: with PDE don't need other things / advances of DNNs

03 Paper – Exothermic heat transfer in curing composite materials

- Heat transfer governing PDE

- $\frac{\partial}{\partial t}(\rho C_P T) = \frac{\partial}{\partial x}\left(k_{xx}\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_{yy}\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k_{zz}\frac{\partial T}{\partial z}\right) + \dot{Q}$

- $\dot{Q} = \upsilon_r \rho_r H_r \frac{d\alpha}{dt}$

- $\frac{\mathrm{d}\alpha}{\mathrm{dt}} = \mathrm{g}(\alpha, T)$

- $\frac{\partial T}{\partial t} = a\frac{\partial^2 T}{\partial x^2} + b\frac{d\alpha}{dt}$ where $a = \frac{k}{\rho C_P}$ and $\mathrm{b} = \frac{\upsilon_r \rho_r H_r}{\rho C_P}$

① Considering one-dimensional system

② Homogeneous material $\mathrm{i.e.}\ \frac{\partial \mathrm{k}}{\partial \mathrm{x}} = 0$

③ Temperature-independent physical properties $\mathrm{i.e.}\ \frac{\partial \rho}{\partial \mathrm{t}} = \frac{\partial C_P}{\partial \mathrm{t}} = 0$

T: temperature
$\mathrm{C_p}$, k, ρ: specific heat capacity, conductivity, density
$\dot{Q}$: internal heat generation
α: resin degree of cure
----
r: represents resin
υ : volume fraction
H: heat of reaction gererated per unit mass

03 Paper – Exothermic heat transfer in curing composite materials

```python
# heat transfer PDE loss
def pde_func_ct(LambdaList):
    x_inp = LambdaList[0]
    t_inp = LambdaList[1]
    out_T = LambdaList[2]
    pred_alpha = LambdaList[3]

    # gradients
    Tt = tf.gradients(out_T, t_inp)
    Tx = tf.gradients(out_T, x_inp)
    Txx = tf.gradients(Tx[0], x_inp)

    # XX=1 where loss is applicable, XX=0 where loss is not applicable
    XX = 0.5 * (1.0+tf.math.sign(t_inp-scale_min2-1e-15)) \
       * 0.5 * (1.0+tf.math.sign(x_inp-scale_min1-1e-15)) \
       * 0.5 * (1.0-tf.math.sign(x_inp-scale_max1+1e-15))

    # correcting based on the nonzero number of points in the batch
    batch_size = tf.cast(tf.size(XX), tf.float64)
    nonzero_in_batch = tf.cast(tf.math.count_nonzero(XX), tf.float64)
    chk = tf.cast(tf.math.count_nonzero(nonzero_in_batch), tf.float64)
    nonzero_in_batch = nonzero_in_batch * chk + coll_num * (1.0-chk)
    batch_size = batch_size * chk + tot_num * (1.0-chk)
    correction =tf.math.sqrt(batch_size/nonzero_in_batch)

    # RHS function in ODE of degree of cure
    alpha_RHSTOT = alpha_RHSTOT_func(out_T, pred_alpha, A_normalized,
                                 dE_normalized,
                                 M, N, ALCT_normalized, ALC,
                                 R_normalized)
```

Input data

Gradients

Why?? Essential??

If $xx = 1$

t and x have to '$t > 0, 1 > x > 0$' → Checking scaling?? Essential??

Deciding loss function can applicable or not

Correction

Number of zero value ↑ ~ correction ↑ ~ loss value ↑

```python
L = (Tt[0] - a_ct * Txx[0]  - b_ct * alpha_RHSTOT)* XX  * correction
```

Computting $\frac{d\alpha}{dt}$

03 Paper – Exothermic heat transfer in curing composite materials

```
# RHS function in ODE of degree of cure
    alpha_RHSTOT = alpha_RHSTOT_func(out_T, pred_alpha, A_normalized,
                                     dE_normalized,
                                     M, N, ALCT_normalized, ALC,
                                     R_normalized)


    a_ct = a_c_normalized *  0.5 * (1.0 + tf.math.sign(x_inp-(L_t_bc_t_normalized+1e-15))) + a_t_normalized *  0.5 * (1.0 - tf.math.sign(x_inp-
(L_t_bc_t_normalized+1e-15)))
    b_ct = b_normalized *  0.5 * (1.0 + tf.math.sign(x_inp-(L_t_bc_t_normalized+1e-15)))



    L = (Tt[0] - a_ct * Txx[0]  - b_ct * alpha_RHSTOT) \
         * XX  * correction

    return L
```

???

- $a = \dfrac{k}{\rho C_P}$ and $b = \dfrac{v_r \rho_r H_r}{\rho C_P}$
- a, b is value but a_ct, b_ct is about sign

03 Paper – Exothermic heat transfer in curing composite materials

- Governing PDE

$$\frac{\partial}{\partial t}(\rho C_P T) = \frac{\partial}{\partial x}\left(k_{xx}\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_{yy}\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k_{zz}\frac{\partial T}{\partial z}\right) + \dot{Q}$$

$$\dot{Q} = v_r \rho_r H_r \frac{d\alpha}{dt}$$

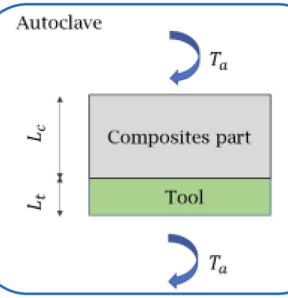
$$\frac{\mathrm{d}\alpha}{\mathrm{dt}} = \mathrm{g}(\alpha, T)$$

$$\frac{\partial T}{\partial t} = a\frac{\partial^2 T}{\partial x^2} + b\frac{d\alpha}{dt} \quad \text{where} \quad a = \frac{k}{\rho C_P} \quad \text{and} \quad \mathrm{b} = \frac{v_r \rho_r H_r}{\rho C_P}$$

$\mathrm{C_p}$, k, ρ: specific heat capacity, conductivity, density

Leads to coupled system of differential equation↓

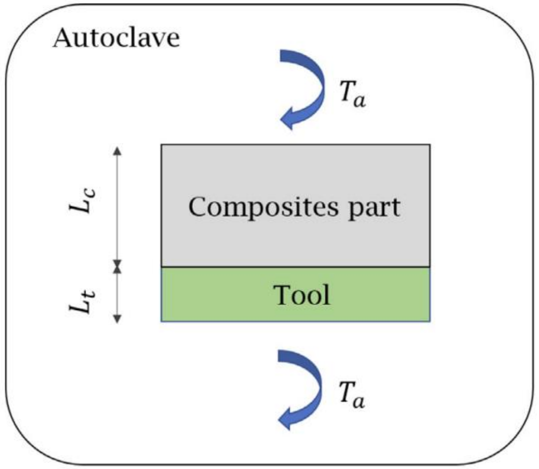
Composite-too system→

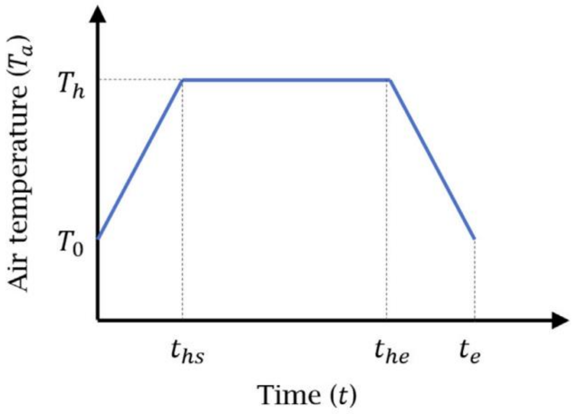


$$a = \begin{cases} a_t & \text{for } 0 < x < L_t \\ a_c & \text{for } L_t < x < L_t + L_c \end{cases}$$

$$b = \begin{cases} 0 & \text{for } 0 < x < L_t \\ b_c & \text{for } L_t < x < L_t + L_c \end{cases}$$

03 Paper – Exothermic heat transfer in curing composite materials





- Boundary condition

  Air temperature in autoclave

  - $T|_{x=0} = T_a(t)$
  - $T|_{x=L_t+L_c} = T_a(t)$

More realistic

Convective heat transfer coefficient

- $h_t(T|_{x=0} - T_a(t)) = k_t \frac{\partial T}{\partial x}|_{x=0}$
- $h_c(T_a(t) - T|_{x=L_t+L_c}) = k_c \frac{\partial T}{\partial x}|_{x=L_t+L_c}$

- Initial condition
  - $T|_{t=0} = T_0(x)$
  - $\alpha|_{t=0} = \alpha_0(x)$

03 Paper – Exothermic heat transfer in curing composite materials

```
# first boundary condition loss
def bc_b_func(LambdaList):

    x_inp = LambdaList[0]
    t_inp = LambdaList[1]
    output_T = LambdaList[2]

    T_bc_b = T_bc(temptypeb, t_inp, T_rate_b, T_s_b, T_hold_b, th1_b, th2_b, scaler2)
    T_bc_b = T_bc_b / T_scale

    # XX=1 where loss is applicable, XX=0 where loss is not applicable
    XX = 0.5 * (1.0-tf.math.sign(x_inp-scale_min1-1e-15))

    # correcting based on the nonzero number of points in the batch
    batch_size = tf.cast(tf.size(XX), tf.float64)
    nonzero_in_batch = tf.cast(tf.math.count_nonzero(XX), tf.float64)
    chk = tf.cast(tf.math.count_nonzero(nonzero_in_batch), tf.float64)
    nonzero_in_batch = nonzero_in_batch * chk + bc_num * (1.0-chk)
    batch_size = batch_size * chk + tot_num * (1.0-chk)
    correction =tf.math.sqrt(batch_size/nonzero_in_batch)

    if bc_typeb == 'prescribed':
        L = (output_T - T_bc_b) * XX * correction
    else:
        Tx = tf.gradients(output_T, x_inp)
        L = ((T_bc_b - output_T) + k_t / h_t * Tx[0]  * scaler1[0,0]) * XX * correction


    return L
```

Call ‘T_bc’ function and show calculated temperature

Deciding loss function can applicable or not

Correction

‘prescribed’ calculate loss with real temperature and predicted temperature

Else calculate loss with a partial differential and fixed function

03 Paper – Exothermic heat transfer in curing composite materials

```python
# second boundary condition loss
def bc_t_func(LambdaList):

    x_inp = LambdaList[0]
    t_inp = LambdaList[1]
    output_T = LambdaList[2]

    T_bc_t = T_bc(temptypet, t_inp, T_rate_t, T_s_t, T_hold_t, th1_t, th2_t, scaler2)
    T_bc_t = T_bc_t / T_scale

    # XX=1 where loss is applicable, XX=0 where loss is not applicable
    XX = 0.5 * (1.0+tf.math.sign(x_inp-scale_max1+1e-15))

    # correcting based on the nonzero number of points in the batch
    batch_size = tf.cast(tf.size(XX), tf.float64)
    nonzero_in_batch = tf.cast(tf.math.count_nonzero(XX), tf.float64)
    chk = tf.cast(tf.math.count_nonzero(nonzero_in_batch), tf.float64)
    nonzero_in_batch = nonzero_in_batch * chk + bc_num * (1.0-chk)
    batch_size = batch_size * chk + tot_num * (1.0-chk)
    correction =tf.math.sqrt(batch_size/nonzero_in_batch)

    if bc_typet == 'prescribed':
        L = (output_T - T_bc_t) * XX * correction
    else:
        Tx = tf.gradients(output_T, x_inp)
        L = ((T_bc_t - output_T) - k_c / h_c * Tx[0]  * scaler1[0,0])  * XX * correction


    return L
```

Call ‘T_bc’ function and show calculated temperature

Deciding loss function can applicable or not

Correction

‘prescribed’ calculate loss with real temperature and predicted temperature

Else calculate loss with a partial differential and fixed function

03 Paper – Exothermic heat transfer in curing composite materials

```
# temperature initial condition loss
def iniT_func(LambdaList):

    x_inp = LambdaList[0]
    t_inp = LambdaList[1]
    output_T = LambdaList[2]

    T_ini_arr = t_inp * 0 + T_ini / T_scale

    # XX=1 where loss is applicable, XX=0 where loss is not applicable
    XX = 0.5 * (1.0-tf.math.sign(t_inp-scale_min2-1e-10)) \
       * 0.5 * (1.0+tf.math.sign(x_inp-scale_min1-1e-10))  \
       * 0.5 * (1.0 - tf.math.sign(x_inp-scale_max1+1e-10))

    # correcting based on the nonzero number of points in the batch
    batch_size = tf.cast(tf.size(XX), tf.float64)
    nonzero_in_batch = tf.cast(tf.math.count_nonzero(XX), tf.float64)
    chk = tf.cast(tf.math.count_nonzero(nonzero_in_batch), tf.float64)
    nonzero_in_batch = nonzero_in_batch * chk + ini_num * (1.0-chk)
    batch_size = batch_size * chk + tot_num * (1.0-chk)
    correction =tf.math.sqrt(batch_size/nonzero_in_batch)

    L = (output_T - T_ini_arr) * XX * correction

    return L
```

Deciding loss function can applicable or not

Correction

calculate loss with real temperature and predicted temperature

04 To-do List

- Study about ‘heat flux continuity loss’, ‘temperature continuity loss’, ‘Alpha losses’, Networks, training …