Avril 2019

Grenoble, France

Étude de l'effet de la radiation sur un écoulement de sel fondu

Francisco Kovacevich

Sous la direction de Pablo Rubiolo

Sommaire

Ce rapport de stage condense un travail de six mois sur l'étude de l'effet de la radiation dans un écoulement de sel fondu et la conception d'une expérience pour pouvoir mesurer cet effet. Ce travail est fait dans le cadre d'un projet européen pour étudier l'utilisation des réacteurs de sel fondu pour la propulsion spatial. On a utilisé des simulations numériques en Ansys pour étudier la sensibilité de l'expérience conçue face à des changements des différentes conditionnes limites et propriétés du sel. On a finit la première étape de conception de l'expérience et on conclu que la radiation a un effet considérable et qui peut être mesuré.

Table des matières

Bibliographie

[1] Michael EADES. *Development of Molten Salt Reactor Technology for Space*. Ohio State University, 2012.

[2] Laurick HUGUET. *Options et choix de conception pour la propulsion nucléaire électrique dans l'espace*. 2018.

[3] Michael F. MODEST. *Radiative Heat Transfer, 3rd Edition*. 2013.

[4] NASA. *Kilopower*. 2018. URL: `https://www.nasa.gov/directorates/spacetech/kilopower`.

[5] Mauricio Tano PABLO RUBIOLO. *Design of close and open channel experiments to study molten salt flows*.

[6] Mauricio Tano PABLO RUBIOLO. *Design of Experiments for the Study of the Solidification of Molten Slats with Limited Influence of External Convection*. 2016.

[7] B. SUNDEN et D. ERIKSSON. *Advanced heat and mass transfer topics*. 2012.

[8] Mauricio TANO. *Modélisation multi-physique multi-échelle de caloporteurs sels fondus et validation expérimentale*. Université Grenoble Alpes, 2018.

A

Code pour les simulations avec Matlab

A.1 Fonction principal

Cette fonction "run" fait les calculs préliminaux et appelle les autres trois fonctions (présentées dessous) de manière itérative pour résoudre le problème de l'échange thermique par conduction y radiation dans un milieu de sel fondu (1D). Cette fonction prend la taille du domaine (X) et les températures limites (T1 et T2).

```
function [qrad0, qrad1, qcond0, qcond1] = run(X, T0, T1, plot_true)
  sigma = 5.6e-8;
  k = @(t) 0.36 + t*5.6E-4;

  % Band Model
  lambda1 = 5;
  lambda2 = 11;
  alfa1 = 2; %alfa1 = 0.02;
  alfa2 = 3000; %alfa2 = 30;
  alfa3 = 50000; %alfa3 = 500;

  % Problem constants
  %X = 0.005;
```

```
n = 100;
%T0 = 800;
%T1 = 700;

constants = [0, sigma, T0, T1, lambda1, lambda2];

x = linspace(0,X,n);
T = (T0 + x*(T1-T0)/X)';
G = 4 * sigma * T.^4;
error = 1;
iterations = 0;

while(error>1e-3)
  Told = T;
  Gold = G;

  G1 = SolveSurfaceToSurface(x,T,alfa1,constants);
  G2 = SolveP1(x,T,alfa2,constants);
  G3 = SolveRosseland(x,T,alfa3,constants);

  T = SolveT(x,[G1 G2 G3],[alfa1 alfa2 alfa3],T,constants,k);

  G = G1 + G2 + G3;
  error = max(norm(Told-T),norm(Gold-G));
  iterations = iterations + 1;

  if(plot_true)
    subplot(1,2,1);plot(x,T);xlabel('x');ylabel('T (K)');
    title(sprintf('Tmin: %d\nTmax: %d\nX: %d',T0,T1,X))
          subplot(1,2,2);plot(x,G);xlabel('x');ylabel('G (W/m2)');
    title(sprintf('error: %f',error))
    pause(0.01)
  end

```

```
  end

  fprintf('iterations: %g',iterations);

  dT_dx0 = (T(2)-T(1))/(x(2)-x(1));
  dT_dx1 = (T(n)-T(n-1))/(x(n)-x(n-1));

  dG1_dx_0 = (G1(2)-G1(1))/(x(2)-x(1));
  dG1_dx_1 = (G1(n)-G1(n-1))/(x(n)-x(n-1));
  dG2_dx_0 = (G2(2)-G2(1))/(x(2)-x(1));
  dG2_dx_1 = (G2(n)-G2(n-1))/(x(n)-x(n-1));
  dG3_dx_0 = (G3(2)-G3(1))/(x(2)-x(1));
  dG3_dx_1 = (G3(n)-G3(n-1))/(x(n)-x(n-1));

  qrad0 = sigma*(T0^4*(F_blackbody(lambda1,T0)-0)-T1^4*(F_blackbody(
    lambda1,T1)-0));
  qrad0 = qrad0 + dG2_dx_0/(3*alfa2);
  qrad0 = qrad0 + (16/3)*(sigma/alfa3)*T0^3*dT_dx0*(1-F_blackbody(
    lambda2,T1));

  qrad1 = sigma*(T0^4*(F_blackbody(lambda1,T0)-0)-T1^4*(F_blackbody(
    lambda1,T1)-0));
  qrad1 = qrad1 + dG2_dx_1/(3*alfa2);
  qrad1 = qrad1 + (16/3)*(sigma/alfa3)*T0^3*dT_dx1*(1-F_blackbody(
    lambda2,T1));

  qcond0 = - k(T0) * dT_dx0;
  qcond1 = - k(T1) * dT_dx1;
end
```

A.2 Résolution de la température

```
function T = SolveT(x,G,alfa,T,constants,k)
```

```

sigma = constants(2);
T0 = constants(3);
T1 = constants(4);

lambda1 = constants(5);
lambda2 = constants(6);

n = length(x);
dx = x(2)-x(1);

error = 1;
max_it = 100;
it = 0;

level=3;
relax_it=2;
relax_para=0.1;
post_smoothing=1;
max_iter=200;
tol=1e-06;
pc_type=2;
connection_threshold=0.25;

while(error>1e-3 && it < max_it)

  A = zeros(n,n);
  for i = 1:n
    for j=1:n
      if i==j
        A(i,j) = -2;
        if j>1
          A(i,i-1) = 1;
        end
```

```
          if j<n
            A(i,i+1) = 1;
          end
        end
      end
    end

    for i=1:n
      %[T(i)*0.000001 ...
      %(4/k(T(i)))*alfa(1)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda1,T(i))-0) ...
      %(4/k(T(i)))*alfa(2)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda2,T(i))-F_blackbody(lambda1,T(i))) ...
      %(4/k(T(i)))*alfa(3)*dx^2*sigma*T(i)^3 * (1-F_blackbody(lambda2,T(i)))]

      A(i,i) = A(i,i) - (4/k(T(i)))*alfa(1)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda1,T(i))-0);
      A(i,i) = A(i,i) - (4/k(T(i)))*alfa(2)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda2,T(i))-F_blackbody(lambda1,T(i)));
      A(i,i) = A(i,i) - (4/k(T(i)))*alfa(3)*dx^2*sigma*T(i)^3 * (1-F_blackbody(lambda2,T(i)));
    end

    A(1,1) = 1;
    A(1,2) = 0;
    A(n,n) = 1;
    A(n,n-1) = 0;

    b = zeros(n,1);
    for i = 2:n-1
      b(i) = -(1/k(T(i)))*dx^2*(alfa(1)*G(i,1) + alfa(2)*G(i,2) + alfa(3)*G(i,3));
    end
```

```
    b(1) = T0;
    b(n) = T1;

    Told = T;
    T = linsolve(A,b);

    %%[T,errorAMG,iter,flag]=AMG(A, b, T, ...
    %%            level,   relax_it, relax_para, ...
        %%                    post_smoothing, max_iter, tol,   ...
        %%                    pc_type, connection_threshold);

    error = norm(Told-T);
    it = it + 1;
  end
end
```

A.3 Résolution de la equation de Rosseland

```
function G = SolveRosseland(x,T,alfa,constants)

  sigma = constants(2);
  T0 = constants(3);
  T1 = constants(4);

  lambda1 = constants(5);
  lambda2 = constants(6);

  n = length(x);
  X = x(n);
  dx = x(2)-x(1);
  G = zeros(n,1);

  for i = 2:n-1
    dT_dx = (T(i+1) - T(i-1))/(2*dx);
```

```
    d2T_dx2 = (T(i+1) - 2*T(i) + T(i-1))/(dx*dx);

    G(i) = 4*sigma*T(i)^4 + (16/3)*(sigma/alfa^2) * (3*(T(i)^2)*(dT_dx)^2 + (T(i)^3)*(d2T_dx2));
    G(i) = G(i) * (1-F_blackbody(lambda2,T(i)));

    %d2T_dx2 = (T(i+1) - 2*T(i) + T(i-1))/(x(i+1)-x(i)^2);
    %G(i) = 4*sigma*T(i)^4 * (F_blackbody(lambda1, T(i)) - F_blackbody(lambda2, T(i))) - k * d2T_dx2;

  end
  G(1) = (G(2) + 2*alfa*sigma*dx*T0^4*(1-F_blackbody(lambda2,T0))) / (1+0.5*alfa*dx);
  G(n) = (G(n-1) + 2*alfa*sigma*dx*T1^4*(1-F_blackbody(lambda2,T1))) / (1+0.5*alfa*dx);

  %G(1) = ( 2*sigma*T(1)^4 + G(2)/(alfa*dx) ) / (1/(alfa*dx)+0.5);
  %G(n-1) = ( 2*sigma*T(n)^4 + G(n) * (1/(alfa*X)+1/2) ) / (1/(alfa*X));
end
```

A.4 Résolution de la equation de diffusion (PI)

```
function G = SolveP1(x,T,alpha,constants)

  sigma = constants(2);
  T0 = constants(3);
  T1 = constants(4);

  lambda1 = constants(5);
  lambda2 = constants(6);

  n = length(x);
  dx = x(2)-x(1);

```

```
  A = zeros(n,n);
  for i = 1:n
    for j=1:n
      if i==j
        A(i,j) = -2-3*alpha^2*dx^2;
        if j>1
          A(i,i-1) = 1;
        end
        if j<n
          A(i,i+1) = 1;
        end
      end
    end
  end

  A(1,1) = -(dx*alpha*0.5+1);
  A(1,2) = 1;
  A(n,n) = -(dx*alpha*0.5+1);
  A(n,n-1) = 1;

  b = zeros(n,1);
  for i = 2:n-1
    b(i) = -3 * alpha^2 * dx^2 * 4 * sigma * T(i)^4 * (F_blackbody(
     lambda2, T(i)) - F_blackbody(lambda1, T(i)));
  end
  b(1) = -alpha*dx*2*sigma*T0^4 * (F_blackbody(lambda2, T0) -
     F_blackbody(lambda1, T0));
  b(n) = -alpha*dx*2*sigma*T1^4 * (F_blackbody(lambda2, T1) -
     F_blackbody(lambda1, T1));

  G = linsolve(A,b);
end
```

A.5 Résolution de la transferance "Surface to Surface"

```
function G = SolveSurfaceToSurface(x,T,alfa,constants)

  sigma = constants(2);
  T0 = constants(3);
  T1 = constants(4);

  n = length(x);

  G = sigma * (T1^4 - T0^4) * ones(n,1);

end
```

A.6 Fonction auxiliaire pour calculer le spectre de corps noir

```
function F = F_blackbody(lambda, T)
  LT = lambda * T;

  %% TAKEN FROM INCROPERA PAG. 740, (lambda*T, F(lambda1 -> lambda2))
  %% WHERE LAMBDA IS IN um AND T in K
  F_data = [0,0;
     200,0;
     400,0;
     600,0;
     800,0.000016;
     1000,0.000321;
     1200,0.002134;
     1400,0.00779;
     1600,0.019718;
     1800,0.039341;
     2000,0.066728;
     2200,0.100888;
```

```
    2400,0.140256;
    2600,0.18312;
    2800,0.227897;
    2898,0.250108;
    3000,0.273232;
    3200,0.318102;
    3400,0.361735;
    3600,0.403607;
    3800,0.443382;
    4000,0.480877;
    4200,0.516014;
    4400,0.548796;
    4600,0.57928;
    4800,0.607559;
    5000,0.633747;
    5200,0.65897;
    5400,0.68036;
    5600,0.701046;
    5800,0.720158;
    6000,0.737818;
    6200,0.75414;
    6400,0.769234;
    6600,0.783199;
    6800,0.796129;
    7000,0.808109;
    7200,0.819217;
    7400,0.829527;
    7600,0.839102;
    7800,0.848005;
    8000,0.856288;
    8500,0.874608;
    9000,0.890029;
    9500,0.903085;
    10000,0.914199;
```

```
    10500,0.92371;
    11000,0.93189;
    11500,0.939959;
    12000,0.945098;
    13000,0.955139;
    14000,0.962898;
    15000,0.969981;
    16000,0.973814;
    18000,0.98086;
    20000,0.985602;
    25000,0.992215;
    30000,0.99534;
    40000,0.997967;
    50000,0.998953;
    75000,0.999713;
    100000,0.999905];

  if(LT > 100000)
    F = 1;
    return
  end

  % Find matching value
  F = interp1(F_data(:,1),F_data(:,2),LT);
  return
end
```