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

Ta	ble c	les matières	3
1	Introduction		5
	1.1	Réacteurs au sel fondu	5
	1.2	Le sel fondu	7
	1.3	Le problème de la radiation	7
2	Intr	oduction au problème de la radiation	11
	2.1	Description générale	13
	2.2	Le modèle des bandes	14
	2.3	Le flux de chaleur	17
	2.4	Modèle surface to surface, l'approximation de Rosseland et l'approxi-	
		mation P1	17
	2.5	Simulation en Matlab	19
	2.6	Le transfert dans un écoulement	19
3	Conception d'une manip expérimentale		22
	3.1	Facilité expérimentale SWATH	22
	3.2	Dispositif expérimental	23
	3.3	Conditions limites	26
4	Rés	ultats des simulations avec Ansys	28

	4.1	Vitesse à l'entrée	29		
	4.2	Évolution du profil de température	30		
	4.3	Variation avec la température de la paroi inférieure	31		
	4.4	Sensibilité à la conductivité thermique	31		
	4.5	Effet de l'émissivité des parois internes	31		
	4.6	Variation de la vitesse à l'entrée	32		
	4.7	Chute de pression	32		
5	Con	clusions	37		
Bi	Bibliographie				
A	Cod	e pour les simulations avec Matlab	40		
	A.1	Fonction principal	40		
	A.2	Résolution de la température	42		
	A.3	Résolution de la equation de Rosseland	45		
	A.4	Résolution de la equation de diffusion (PI)	46		
	A.5	Résolution de la transferance "Surface to Surface"	48		

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 fon-dus et validation expérimentale*. Université Grenoble Alpes, 2018.



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).

```
1 function [qrad0, qrad1, qcond0, qcond1] = run(X, T0, T1, plot_true)
    sigma = 5.6e-8;
    k = @(t) 0.36 + t*5.6E-4;
3
   % Band Model
   lambda1 = 5;
   lambda2 = 11;
    alfa1 = 2; %alfa1 = 0.02;
    alfa2 = 3000; %alfa2 = 30;
    alfa3 = 50000; %alfa3 = 500;
10
11
   % Problem constants
12
   %X = 0.005;
13
```

```
n = 100;
    %T0 = 800;
15
    %T1 = 700;
16
17
    constants = [0, sigma, T0, T1, lambda1, lambda2];
18
    x = linspace(0, X, n);
20
    T = (T0 + x*(T1-T0)/X)';
21
22
    G = 4 * sigma * T.^4;
    error = 1;
23
    iterations = 0;
25
    while (error > 1e - 3)
26
      Told = T;
27
      Gold = G;
28
      G1 = SolveSurfaceToSurface(x,T,alfa1,constants);
30
      G2 = SolveP1(x,T,alfa2,constants);
31
      G3 = SolveRosseland(x,T,alfa3,constants);
32
33
      T = SolveT(x, [G1 G2 G3], [alfa1 alfa2 alfa3], T, constants, k);
35
      G = G1 + G2 + G3;
36
      error = max(norm(Told-T), norm(Gold-G));
37
       iterations = iterations + 1;
38
      if (plot_true)
40
         subplot(1,2,1); plot(x,T); xlabel('x'); ylabel('T (K)');
41
         title(sprintf('Tmin: %d\nTmax: %d\nX: %d',T0,T1,X))
42
               subplot(1,2,2); plot(x,G); xlabel('x'); ylabel('G (W/m2)');
43
         title(sprintf('error: %f',error))
         pause (0.01)
45
      end
46
47
```

```
end
49
    fprintf('iterations: %g',iterations);
50
51
52
    dT_dx0 = (T(2)-T(1))/(x(2)-x(1));
    dT_dx1 = (T(n)-T(n-1))/(x(n)-x(n-1));
54
    dG1_dx_0 = (G1(2)-G1(1))/(x(2)-x(1));
55
    dG1_dx_1 = (G1(n)-G1(n-1))/(x(n)-x(n-1));
56
    dG2_dx_0 = (G2(2)-G2(1))/(x(2)-x(1));
57
    dG2_dx_1 = (G2(n)-G2(n-1))/(x(n)-x(n-1));
    dG3_dx_0 = (G3(2)-G3(1))/(x(2)-x(1));
59
    dG3_dx_1 = (G3(n)-G3(n-1))/(x(n)-x(n-1));
60
61
    qrad0 = sigma*(T0^4*(F_blackbody(lambda1,T0)-0)-T1^4*(F_blackbody(
62
     lambda1, T1) - 0));
    qrad0 = qrad0 + dG2_dx_0/(3*alfa2);
63
    qrad0 = qrad0 + (16/3)*(sigma/alfa3)*T0^3*dT_dx0*(1-F_blackbody(
64
     lambda2,T1));
65
    qrad1 = sigma*(T0^4*(F_blackbody(lambda1,T0)-0)-T1^4*(F_blackbody(
     lambda1,T1)-0));
    qrad1 = qrad1 + dG2_dx_1/(3*alfa2);
67
    qrad1 = qrad1 + (16/3)*(sigma/alfa3)*T0^3*dT_dx1*(1-F_blackbody(
     lambda2,T1));
    qcond0 = - k(T0) * dT_dx0;
70
    qcond1 = - k(T1) * dT_dx1;
71
72 end
```

A.2 Résolution de la température

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

```
2
    sigma = constants(2);
3
    T0 = constants(3);
4
5
    T1 = constants(4);
6
7
    lambda1 = constants(5);
    lambda2 = constants(6);
8
10
    n = length(x);
    dx = x(2)-x(1);
11
    error = 1;
13
    max_it = 100;
14
    it = 0;
15
16
    level=3;
17
    relax_it=2;
18
19
    relax_para = 0.1;
    post_smoothing=1;
20
    max_iter = 200;
21
    tol=1e-06;
23
    pc_type=2;
    connection_threshold = 0.25;
24
25
    while(error>1e-3 && it < max_it)</pre>
26
      A = zeros(n,n);
28
      for i = 1:n
29
         for j=1:n
30
           if i == j
31
             A(i,j) = -2;
             if j > 1
33
               A(i, i-1) = 1;
34
35
             end
```

```
if j<n
                                                                                                                       A(i, i+1) = 1;
 37
                                                                                                        end
38
                                                                                        end
 39
 40
                                                                       end
                                                     end
 42
                                                     for i=1:n
 43
 44
                                                                   %[T(i)*0.000001 ...
                                                                     (4/k(T(i)))*alfa(1)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda1,T(i)))*alfa(1)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda1,T(i)))*alfa(1)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda1,T(i)))*alfa(1)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda1,T(i)))*alfa(1)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda1,T(i)))*alfa(1)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda1,T(i)))*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*alfa(1)*
 45
                                                )-0) ...
                                                                   %(4/k(T(i)))*alfa(2)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda2,T(i)))*alfa(2)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda2,T(i))*alfa(2)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda2,T(i))*alfa(2)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda2,T(i))*alfa(2)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda2,T(i))*alfa(2)*dx^2*sigma*T(i)^3 * (F_blackbody(lambda2,T(i))*alfa(2)*dx^2*sigma*T(i)*alfa(2)*dx^2*sigma*T(i)*alfa(2)*dx^2*sigma*T(i)*alfa(2)*dx^2*sigma*T(i)*alfa(2)*alfa(2)*dx^2*sigma*T(i)*alfa(2)*alfa(2)*alfa(2)*alfa(2)*alfa(2)*alfa(2)*alfa(2)*alfa(2)*alfa(2)*alfa(2)*alfa(2)*alfa(2)*alfa(2)*alfa(2)*alfa
 46
                                                )-F_blackbody(lambda1,T(i))) ...
                                                                   %(4/k(T(i)))*alfa(3)*dx^2*sigma*T(i)^3*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda2,T(i)^3)*(1-F_blackbody(lambda
                                                i)))]
                                                                     A(i,i) = A(i,i) - (4/k(T(i)))*alfa(1)*dx^2*sigma*T(i)^3 * (
 49
                                                F_blackbody(lambda1,T(i))-0);
                                                                     A(i,i) = A(i,i) - (4/k(T(i)))*alfa(2)*dx^2*sigma*T(i)^3 * (
 50
                                                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-i)^4
                                                F_blackbody(lambda2,T(i)));
                                                    end
 52
                                                  A(1,1) = 1;
                                                  A(1,2) = 0;
56
                                                  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
 61
                                                  (3)*G(i,3));
 62
                                                    end
```

```
b(1) = T0;
      b(n) = T1;
65
      Told = T;
66
      T = linsolve(A,b);
67
      \%[T,errorAMG,iter,flag]=AMG(A, b, T, ...
      %%
                   level, relax_it, relax_para, ...
70
          %%
                                post_smoothing, max_iter, tol, ...
71
          %%
72
                                pc_type , connection_threshold);
73
      error = norm(Told-T);
74
      it = it + 1;
75
76
77 end
```

A.3 Résolution de la equation de Rosseland

```
1 function G = SolveRosseland(x,T,alfa,constants)
2
    sigma = constants(2);
3
    T0 = constants(3);
    T1 = constants(4);
    lambda1 = constants(5);
7
8
    lambda2 = constants(6);
    n = length(x);
    X = x(n);
11
    dx = x(2) - x(1);
12
13
    G = zeros(n,1);
14
    for i = 2:n-1
15
      dT_dx = (T(i+1) - T(i-1))/(2*dx);
16
```

```
d2T_dx^2 = (T(i+1) - 2*T(i) + T(i-1))/(dx*dx);
18
      G(i) = 4*sigma*T(i)^4 + (16/3)*(sigma/alfa^2) * (3*(T(i)^2)*(dT_dx)
19
     ^2 + (T(i)^3)*(d2T_dx2));
20
      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);
22
      G(i) = 4*sigma*T(i)^4 * (F_blackbody(lambda1, T(i)) - F_blackbody(
     lambda2, T(i)) - k * d2T_dx2;
24
    end
    G(1) = (G(2) + 2*alfa*sigma*dx*T0^4*(1-F_blackbody(lambda2,T0))) /
26
     (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);
29
   %G(n-1) = (2*sigma*T(n)^4 + G(n) * (1/(alfa*X)+1/2)) / (1/(alfa*X));
31 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
14
      for j=1:n
15
        if i == j
16
17
          A(i,j) = -2-3*alpha^2*dx^2;
           if j > 1
18
            A(i, i-1) = 1;
19
           end
20
           if j<n
21
            A(i, i+1) = 1;
22
          end
        end
24
      end
25
    end
26
27
    A(1,1) = -(dx*alpha*0.5+1);
28
    A(1,2) = 1;
29
    A(n,n) = -(dx*alpha*0.5+1);
30
    A(n,n-1) = 1;
31
32
    b = zeros(n,1);
33
    for i = 2:n-1
34
      b(i) = -3 * alpha^2 * dx^2 * 4 * sigma * T(i)^4 * (F_blackbody(
35
     lambda2, T(i)) - F_blackbody(lambda1, T(i)));
    end
36
    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) -
38
      F_blackbody(lambda1, T1));
39
    G = linsolve(A,b);
41 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

```
1 function F = F_blackbody(lambda, T)
    LT = lambda * T;
2
3
    % TAKEN FROM INCROPERA PAG. 740, (lambda*T, F(lambda1 -> lambda2))
4
    %% WHERE LAMBDA IS IN um AND T in K
    F_{data} = [0,0;
      200,0;
7
      400,0;
8
9
      600,0;
      800,0.000016;
      1000,0.000321;
11
      1200,0.002134;
12
      1400,0.00779;
13
      1600,0.019718;
14
      1800,0.039341;
15
      2000,0.066728;
16
      2200,0.100888;
17
```

```
2400,0.140256;
      2600,0.18312;
19
      2800,0.227897;
20
21
      2898,0.250108;
      3000,0.273232;
22
23
      3200,0.318102;
24
      3400,0.361735;
25
      3600,0.403607;
26
      3800,0.443382;
      4000,0.480877;
27
      4200,0.516014;
      4400,0.548796;
29
30
      4600,0.57928;
31
      4800,0.607559;
32
      5000,0.633747;
      5200,0.65897;
33
      5400,0.68036;
34
35
      5600,0.701046;
      5800,0.720158;
36
      6000,0.737818;
37
      6200,0.75414;
      6400,0.769234;
39
      6600,0.783199;
40
      6800,0.796129;
41
      7000,0.808109;
42
      7200,0.819217;
43
      7400,0.829527;
44
      7600,0.839102;
45
      7800,0.848005;
46
      8000,0.856288;
      8500,0.874608;
48
      9000,0.890029;
49
      9500,0.903085;
50
51
      10000,0.914199;
```

```
10500,0.92371;
      11000,0.93189;
53
      11500,0.939959;
54
55
      12000,0.945098;
      13000,0.955139;
56
      14000,0.962898;
      15000,0.969981;
58
      16000,0.973814;
59
      18000,0.98086;
60
      20000,0.985602;
61
      25000,0.992215;
      30000,0.99534;
63
      40000,0.997967;
64
      50000,0.998953;
65
      75000,0.999713;
66
      100000,0.999905];
67
68
69
    if(LT > 100000)
      F = 1;
70
      return
71
    end
72
73
    % Find matching value
74
    F = interp1(F_data(:,1),F_data(:,2),LT);
75
    return
77 end
```