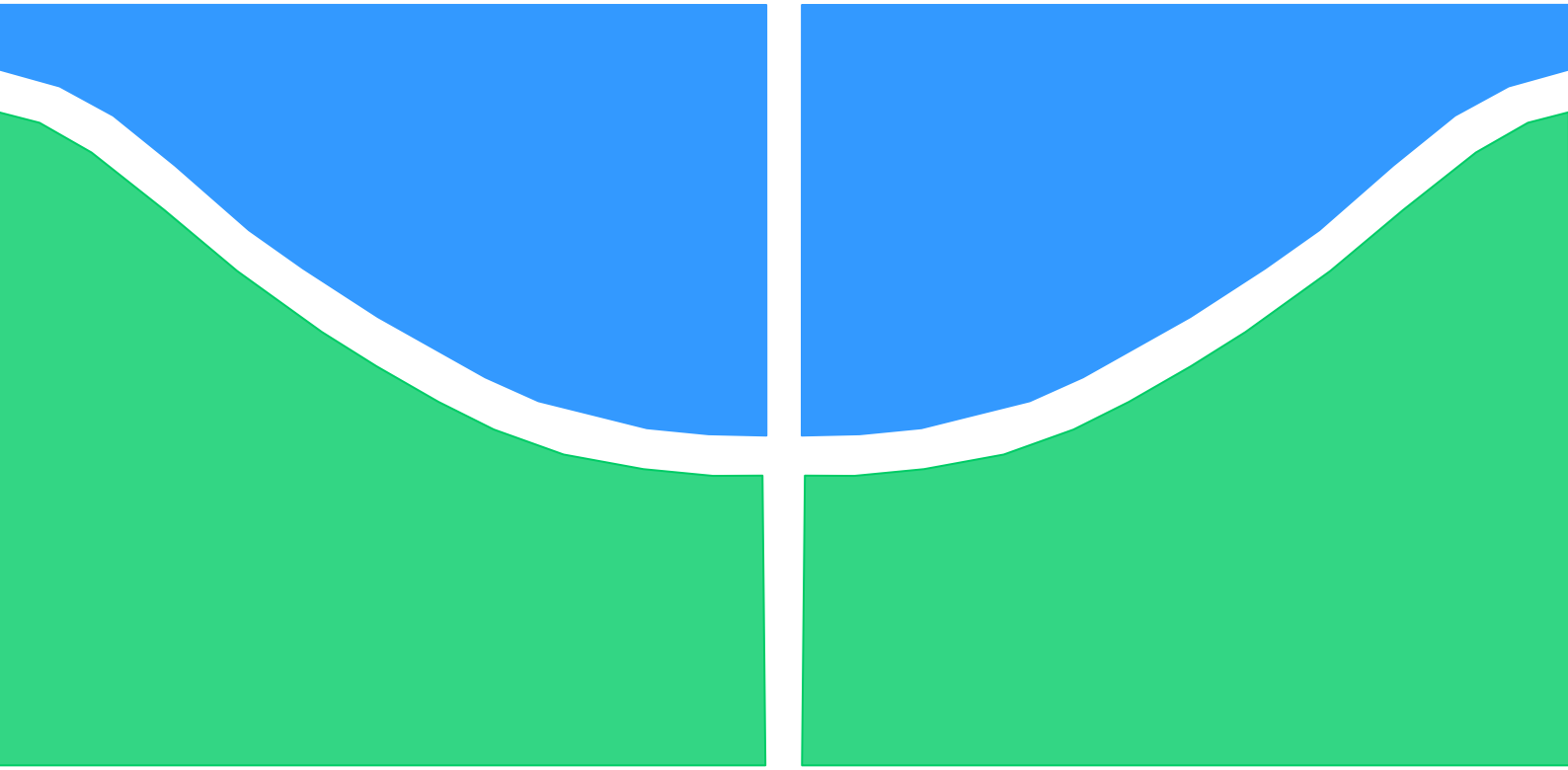


PROJETO DE GRADUAÇÃO 2

DESENVOLVIMENTO DE INTERFACE GRÁFICA PARA RESOLUÇÃO DE PROBLEMAS DE TRANSFÊNCIA DE CALOR

Por,

Diego de Carvalho Silva

Júlio Cesar Abade Dias Belisario

Brasília, 10 de Outubro de 2012

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA MECANICA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

PROJETO DE GRADUAÇÃO 2

DESENVOLVIMENTO DE INTERFACE GRÁFICA PARA RESOLUÇÃO DE PROBLEMAS DE TRANSFERÊNCIA DE CALOR

POR,

Diego de Carvalho Silva
Júlio César Abade Dias Belisário

Relatório submetido como requisito para obtenção
do grau de Engenheiro Mecânico.

Banca Examinadora

Prof. Antonio Francisco Fortes, UnB/ ENM (Orientador) _____

Prof. Armando de Azevedo Caldeira Pires, UnB/ ENM _____

Prof. Jones Yudi Mori Alves da Silva, UnB/ ENM _____

Brasília, 16 de Fevereiro de 2013

RESUMO

De uma maneira geral, a solução dos problemas de transferência de calor requer a solução de equações diferenciais que exprimem a conservação da massa, do momento e da energia. Nos problemas em que essas soluções são analíticas, é usual apresentá-las na forma gráfica, como o são os conhecidos diagramas de Heisler (Temperature Charts for Induction and Constant Temperature Heating, Trans. ASME, vol. 69, p.227-236, 1947) e de Schneider (Temperature Response Charts, John Wiley & Sons, 1963) para a condução de calor em sólidos. A apresentação gráfica é também muito utilizada para os fatores de forma na radiação térmica, como os diagramas de Siegel & Howell (Thermal Radiation Heat Transfer, McGraw-Hill, 1972). A indisputável utilidade desses gráficos e tabelas para a solução de problemas de transmissão de calor não elimina a necessidade dos valores numéricos precisos das grandezas representadas, cuja consulta gráfica está naturalmente sujeita às imprecisões das interpolações visuais, ou das interpolações numéricas nas tabelas correspondentes. O objetivo desse trabalho é desenvolver uma interface gráfica (GUI - Graphic User Interface) didática que permita ao usuário a determinação dos parâmetros para diversos problemas de condução de calor em sólidos e para os fatores de forma na radiação térmica.

ABSTRACT

Problems in heat transfer are generally posed as a set of partial differential equations expressing the conservation of mass, momentum and energy. In those situations where there are available closed analytical solutions, they are usually presented in graphical form, like the well-known diagrams of Heisler (Temperature Charts for Induction and Constant Temperature Heating, Trans. ASME, vol. 69, p.227-236, 1947) and of Schneider (Temperature Response Charts, John Wiley & Sons, 1963) for the conduction of heat in solids. By the same token, graphic solutions are also available in closed analytical form for the shape factors in radiation heat transfer, like the diagrams of Siegel & Howell (Thermal Radiation Heat Transfer, McGraw-Hill, 1972). The graphical representations as well as the tables of numerical values for the solutions are though subject to errors of interpolation, either visual in graphics or numerical in tables. The objective of this work is to develop a graphic user interface(GUI) that allows the user to determine precisely the needed parameters for different transient heat conduction problems in solids and the shape factor in thermal radiation heat transfer.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

A	Área	[m^2]
c_p	Calor especifico a pressão constante	[kJ/kg.K]
E	Poder emissivo	[W/ m^2]
$\dot{e}$	Geração de energia interna	[W/ m^3]
G	Irradiação	[W]
h	Coeficiente de transferência de calor por convecção	[W/ m^2. K]
k	Condutividade térmica	[W/m.K]
L	Comprimento característico	[m]
l	Comprimento	[m]
m	Massa	[Kg]
$\dot{m}$	Vazão mássica	[kg/s]
$\dot{q}$	Fluxo térmico	[W/ m^2]
R	Raio característico	[m]
r	Raio	[m]
T	Temperatura	[^oC]

Símbolos Gregos

α	Difusividade térmica	[m2/s]
σ	Constante de Stefan-Boltzmann	[W/ $m^2.K^4$]
ρ	Densidade	[kg/m^3]
υ	Volume	[m^3]
ρ	Densidade	[m^3/kg]
∇	Gradiente	--
∇^2	Laplaciano	--
ε	Emissividade	--
θ	Temperatura adimensional	--
χ	Comprimento adimensional	--
τ	Tempo adimensional	--

Grupos Adimensionais

Bi	Número de Biot
Fo	Número de Fourier

Subscritos

i	inicial
∞	infinito

Sobrescritos

$\bullet$	Variação temporal

Siglas

GUI	Interface Gráfica (Interface User Grafic)

1 INTRODUÇÃO

A transferência de calor é um fenômeno que ocorre a todo instante na natureza e possui uma importância fundamental nos sistemas de engenharia. Na área de geração e transporte de energia a transferência de calor é um aspecto dominante. Aviões, usinas, edifícios, motores, turbinas, computadores, aparelhos eletrônicos, eletrodomésticos são projetados e construídos com o uso dos princípios da transferência de calor.

Atualmente o desenvolvimento tecnológico em vários ramos depende de avanços nessa área do conhecimento. Por exemplo, a eficiência de células combustíveis de hidrogênio e turbinas a gás aumenta com a temperatura de operação, portanto o limite de eficiência desses dispositivos é proporcional à capacidade de receber e dissipar calor dos materiais que são feitos sem se danificar. Fato semelhante acontece com os computadores, onde microprocessadores e circuitos cada vez mais potentes e menores geram mais energia térmica e tornam suas necessidades de resfriamento um gargalo aos avanços. Progressos atuais na área de biomedicina também foram possíveis graças ao uso dos preceitos da transferência de calor.

Desde os primórdios da civilização o homem se encanta com o fogo e, portanto, com o calor, mas o entendimento dos princípios físicos da natureza do calor como aceito atualmente só veio com a teoria cinética, em meados do século XIX. Antes dessa época, era aceita a teoria do calórico, que segundo Cengel (2009), foi proposta em 1789 por Lavoisier, na qual existia um fluido invisível, chamado calórico, que fluía pelas substâncias e alterava suas temperaturas. Esse fluido não poderia ser criado ou destruído e a quantidade de calor transferido de ou para um objeto era proporcional à sua massa e à diferença de temperatura. Essa teoria recebeu diversas contestações até finalmente ser desbancada pelos trabalhos de James Joule, que realizou experimentos entre 1840 e 1849 mostrando ser possível gerar calor indefinidamente através de trabalho mecânico e logo o calor não poderia ser uma substância. Nessa obra foi demonstrada a equivalência entre calor e trabalho, ou seja, ambos são manifestações diferentes da mesma coisa, que é a energia. Pela teoria cinética, essa energia corresponde à energia cinética do movimento dos átomos e moléculas.

Sempre que houver uma diferença de temperatura entre corpos ou entre meios, irá haver um fluxo da energia térmica no sentido da maior temperatura para a menor. Esse fluxo que é conhecido como calor. A termodinâmica se preocupa com as quantidades totais de calor envolvidas nos processos e seus estados iniciais e finais. A maneira, porém, como essa interação acontece e os tempos e taxas em que os fluxos ocorrem são os objetos de estudo da transferência de calor.

O fluxo de calor irá cessar quando os corpos atingirem a mesma temperatura, ou seja, estiverem em equilíbrio térmico. Quando isso ocorre é possível mensurar a quantidade de calor que foi

transferido até o equilíbrio através da termodinâmica. Logo, essa análise trabalha com estados termodinâmicos de equilíbrio e suas passagens de um estado de equilíbrio para outro. Porém essa análise não é capaz de determinar o tempo em que esse equilíbrio é atingido. A transferência de calor estuda essa taxa de transferência, que apenas ocorre durante o estado de não-equilíbrio termodinâmico, ou seja, quando ainda existe o gradiente de temperatura.

Apesar de necessitar de outras ferramentas para cumprir seus objetivos, a ciência da transferência de calor utiliza as leis da termodinâmica como ferramenta essencial. A lei da conservação da energia (1ª Lei da Termodinâmica) é usada na forma de taxas, onde a taxa com que aumenta a energia de um sistema é igual a taxa de energia que entra nesse sistema. Essa taxa de transferência é proporcional ao grau de não-equilíbrio, ou seja, quanto maior for o gradiente de temperatura, maior será a taxa de transferência de calor.

Existem três mecanismos de transferência de calor: condução, convecção e radiação. Condução é a transferência de energia entre partículas com mais quantidade de energia para as partículas vizinhas com menos energia devido à interação entre elas, portanto são processos que acontecem em níveis atômicos e moleculares. A temperatura em um ponto de um corpo qualquer é associada à energia dos átomos e moléculas nas proximidades desse ponto e essa energia esta relacionada com seus movimentos de translação aleatório e movimentos internos de rotação e vibração. Em sólidos, ocorre apenas o movimento de vibração dos átomos e o movimento de translação dos elétrons livres. Em líquidos e gases o transporte de energia é feito pelas colisões moleculares (transferência da molécula de maior energia para a menor) e pela difusão molecular. Para que a transferência de calor em líquidos e gases ocorra por condução não pode haver movimento macroscópico do fluido, caso ocorra irá haver convecção.

A segunda forma de transferência de calor é a convecção. Nesse modo, que irá ocorrer quando houver um movimento macroscópico global de um gás ou líquido em relação a uma superfície sólida com temperaturas diferentes, além do efeito da condução existe transferência de energia através do movimento do fluido. Quanto maior a velocidade do fluido, maior será a transferência de calor. No caso limite, quando a velocidade for nula, o processo se da por condução pura.

A radiação térmica, que no âmbito da transferência de calor é chamada apenas radiação por simplificação, é a terceira forma de transferência de calor, sendo a energia emitida pela matéria em forma de ondas eletromagnéticas devido a alterações na configuração eletrônica dos átomos ou moléculas. A radiação térmica é devida a temperatura da matéria, sendo emitida por qualquer corpo que esteja em uma temperatura não-nula e assim difere dos outros tipos de radiação, como raios X, raios gama, ondas de rádio, pois estas não se relacionam com a temperatura. Diferentemente da condução e da convecção, a radiação não necessita de meio material para se propagar.

É comum que esses modos de transferência de calor ocorram simultaneamente em um meio e, nesse caso, cada parcela irá contribuir para o fluxo total de calor.

No caso da condução, o campo de temperaturas, ou seja, os valores de temperaturas em cada ponto de um corpo no tempo, pode ser determinado pela equação diferencial parcial adimensionalizada

$$\frac{\partial^2\theta}{\partial\chi^2} = \frac{\partial\theta}{\partial\tau} \tag{1.1}$$

juntamente com as condições de contorno, onde θ corresponde a temperatura, χ a distância e τ ao tempo, todos adimensionais. A vantagem de se trabalhar com sistemas adimensionais é que os resultados obtidos podem ser generalizados para sistemas de diferentes configurações.

Existem vários métodos numéricos e analíticos de resolução dessa equação diferencial, como elementos finitos, diferenças finitas, transformada de Laplace. O método adotado nesse trabalho é o de separação de variáveis, que usa séries de Fourier. A solução usando esse método envolve a soma de séries infinitas e equações implícitas. Isso faz com que essas soluções sejam de difícil análise e, portanto, existe uma motivação para usar soluções aproximadas dessas soluções analíticas em forma de gráficos e tabelas. Exemplos disso são os famosos gráficos de Heisler (Temperature Charts for Induction and Constant Temperature Heating, Trans. ASME, vol. 69, p.227-236, 1947) e de Schneider (Temperature Response Charts, John Wiley & Sons, 1963). Na Figura (1.1) é mostrada a versão desses gráficos para o caso da condução em uma parede plana. Existe também uma versão em forma de tabelas desses gráficos, mostrada na Tab (1.1).

Tanto os gráficos com as tabelas são ferramentas muito úteis na determinação de parâmetros de sistemas para resolução de problemas, porém, os processos de interpolação numérica carregam erros de aproximação. Na interpolação gráfica o erro é ainda maior, pois, por ser um processo visual, depende da habilidade e atenção do operador e consequentemente está sujeita a erros sistemáticos, como uma imperfeição da ferramenta usada para analisar o gráfico ou no próprio procedimento adotado, e a erros aleatórios, que decorrem da limitação da ferramenta ou do procedimento. Como agravante, os próprios gráficos e tabelas são feitos baseados em soluções aproximadas das soluções analíticas exatas. Esses erros todos, quando propagados nos cálculos podem mudar substancialmente os valores finais encontrados. Além disso, a leitura de gráficos e tabelas pode ser uma atividade desagradável e até desgastante quando feita repetitivamente.

O objetivo desse trabalho é desenvolver um programa computacional com propósitos didáticos que se apresente ao usuário através de uma interface gráfica (GUI) amigável e que o possibilite determinar parâmetros de problemas de transferência de calor e consequente resolução de exercícios sem a necessidade de interpolações gráficas e numéricas, possibilitando a obtenção de resultados mais precisos e tornando o trabalho mais fácil e confortável para estudantes da área de transferência de calor. Apesar de existirem alguns programas comerciais que possuem algumas dessas funcionalidades, como o EES e o Heat Transfer Tools, se tratam de softwares pagos. A interface desenvolvida aqui é disponibilizada gratuitamente, além de possibilitar a livre alteração do código fonte.

Nos problemas de radiação térmica também se costuma usar soluções numéricas aproximadas sob a forma gráfica. Dessa forma, o escopo acima apresentado irá abranger no fenômeno da condução os gráficos de condução transiente para uma grande parede plana, um longo cilindro e uma esfera, sólidos semi-infinitos. No fenômeno da radiação irá abranger também seus fatores de forma.

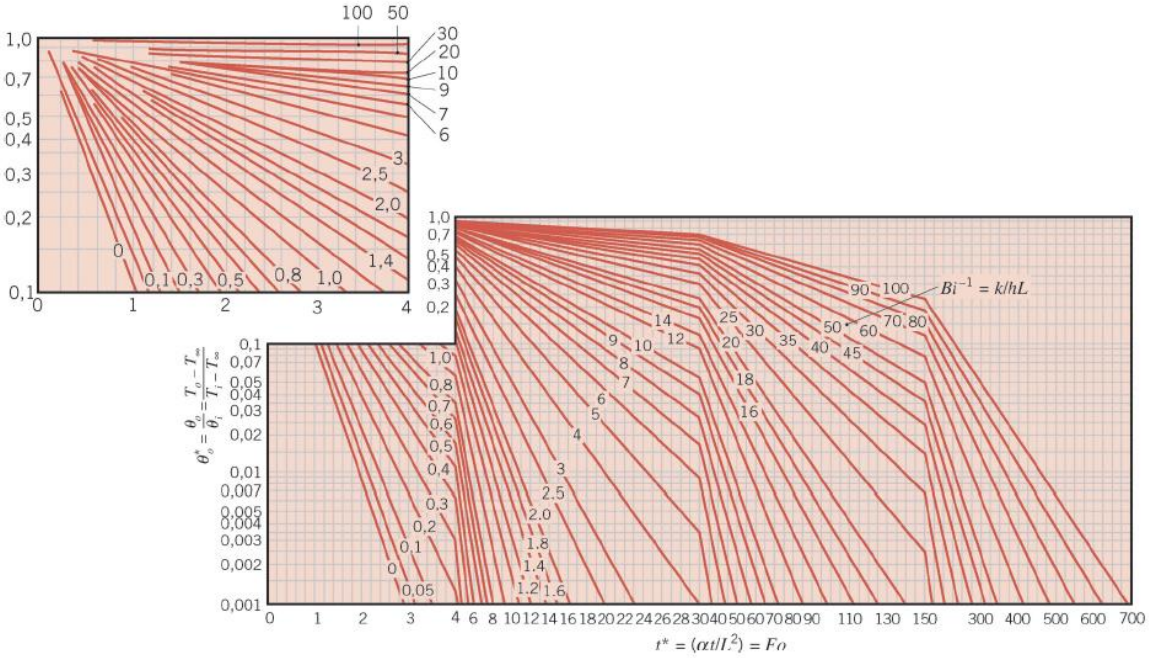


Figura 1.1 - Gráficos de Heisler de condução transiente para a parede. (fonte: Incropera, 2008)

O trabalho está divido em 4 capítulos. Na introdução são apresentados os aspectos principais do escopo do trabalho, mostrando qual sua motivação, abrangência e os objetivos propostos. No segundo capítulo é apresentada a formulação teórica dos problemas da condução de calor em algumas geometrias e da troca de calor entre superfícies por radiação. Esse embasamento teórico será necessário para o desenvolvimento da interface. No terceiro é mostrado o desenvolvimento da interface, juntamente com sua análise, mostrando seus aspectos essenciais, a forma de utilização de seus recursos bem como avaliação dos resultados produzidos. Na conclusão, quarto e último capítulo, são analisados os objetivos alcançados de uma maneira geral.

Tabela 1.1 – Coeficiente usados nas soluções de conduções transientes para parede, esfera e cilindro. (fonte: Cengel, 2003)

Coefficients used in the one-term approximate solution of transient one-dimensional heat conduction in plane walls, cylinders, and spheres (Bi = hL/k for a plane wall of thickness $2L$, and Bi = hr_o/k for a cylinder or sphere of radius r_o)

	Plane Wall		*Cylinder*		*Sphere*	
Bi	λ_1	A_1	λ_1	A_1	λ_1	A_1
0.01	0.0998	1.0017	0.1412	1.0025	0.1730	1.0030
0.02	0.1410	1.0033	0.1995	1.0050	0.2445	1.0060
0.04	0.1987	1.0066	0.2814	1.0099	0.3450	1.0120
0.06	0.2425	1.0098	0.3438	1.0148	0.4217	1.0179
0.08	0.2791	1.0130	0.3960	1.0197	0.4860	1.0239
0.1	0.3111	1.0161	0.4417	1.0246	0.5423	1.0298
0.2	0.4328	1.0311	0.6170	1.0483	0.7593	1.0592
0.3	0.5218	1.0450	0.7465	1.0712	0.9208	1.0880
0.4	0.5932	1.0580	0.8516	1.0931	1.0528	1.1164
0.5	0.6533	1.0701	0.9408	1.1143	1.1656	1.1441
0.6	0.7051	1.0814	1.0184	1.1345	1.2644	1.1713
0.7	0.7506	1.0918	1.0873	1.1539	1.3525	1.1978
0.8	0.7910	1.1016	1.1490	1.1724	1.4320	1.2236
0.9	0.8274	1.1107	1.2048	1.1902	1.5044	1.2488
1.0	0.8603	1.1191	1.2558	1.2071	1.5708	1.2732
2.0	1.0769	1.1785	1.5995	1.3384	2.0288	1.4793
3.0	1.1925	1.2102	1.7887	1.4191	2.2889	1.6227
4.0	1.2646	1.2287	1.9081	1.4698	2.4556	1.7202
5.0	1.3138	1.2403	1.9898	1.5029	2.5704	1.7870
6.0	1.3496	1.2479	2.0490	1.5253	2.6537	1.8338
7.0	1.3766	1.2532	2.0937	1.5411	2.7165	1.8673
8.0	1.3978	1.2570	2.1286	1.5526	2.7654	1.8920
9.0	1.4149	1.2598	2.1566	1.5611	2.8044	1.9106
10.0	1.4289	1.2620	2.1795	1.5677	2.8363	1.9249
20.0	1.4961	1.2699	2.2880	1.5919	2.9857	1.9781
30.0	1.5202	1.2717	2.3261	1.5973	3.0372	1.9898
40.0	1.5325	1.2723	2.3455	1.5993	3.0632	1.9942
50.0	1.5400	1.2727	2.3572	1.6002	3.0788	1.9962
100.0	1.5552	1.2731	2.3809	1.6015	3.1102	1.9990
∞	1.5708	1.2732	2.4048	1.6021	3.1416	2.0000

2 FORMULAÇÃO TEÓRICA

Nesse capítulo serão apresentados os conceitos básicos de transferência de calor. Será desenvolvida a formulação teórica do problema da condução de calor em parede plana. Depois essa formulação será expandida para os casos de um cilindro e uma esfera. Os resultados serão usados no próximo capítulo para a programação da interface.

2.1 MODOS DE TRANSFERÊNCIA DE CALOR

Conforme citado existem três modos no qual a energia térmica pode ser transferida de um meio para outro: condução, convecção e radiação. Na condução o calor é transferido através dos átomos e moléculas de um meio. No caso dos sólidos, esse transporte também é feito pelos elétrons livres. As taxas de calor transferido por condução podem ser quantificadas pela lei de Fourier. No caso de uma barra de comprimento L, com as superfícies laterais isoladas termicamente de modo a garantir que o fluxo de calor ocorra somente pelas extremidades mantidas a temperaturas diferentes T_1 e T_2, conforme Fig. (2.1), a lei de Fourier diz que a quantidade de calor que flui entre as extremidades por unidade de área e por unidade de tempo vale

$$\dot{q}=\frac{-k(T_1-T_2)}{L} \tag{2.1}$$

onde k é condutividade térmica do material (W/m.K) e representa sua capacidade de transportar calor.

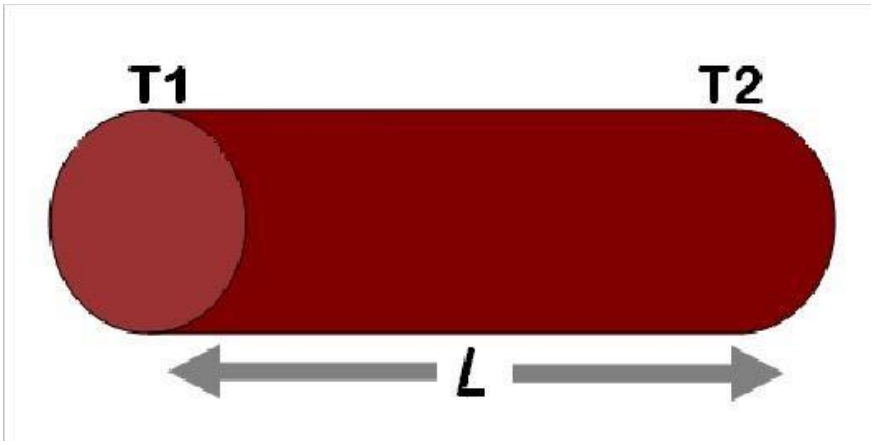


Figura 2.1 – Barra isolada lateralmente com extremidades mantidas à temperaturas T_1 e T_2.

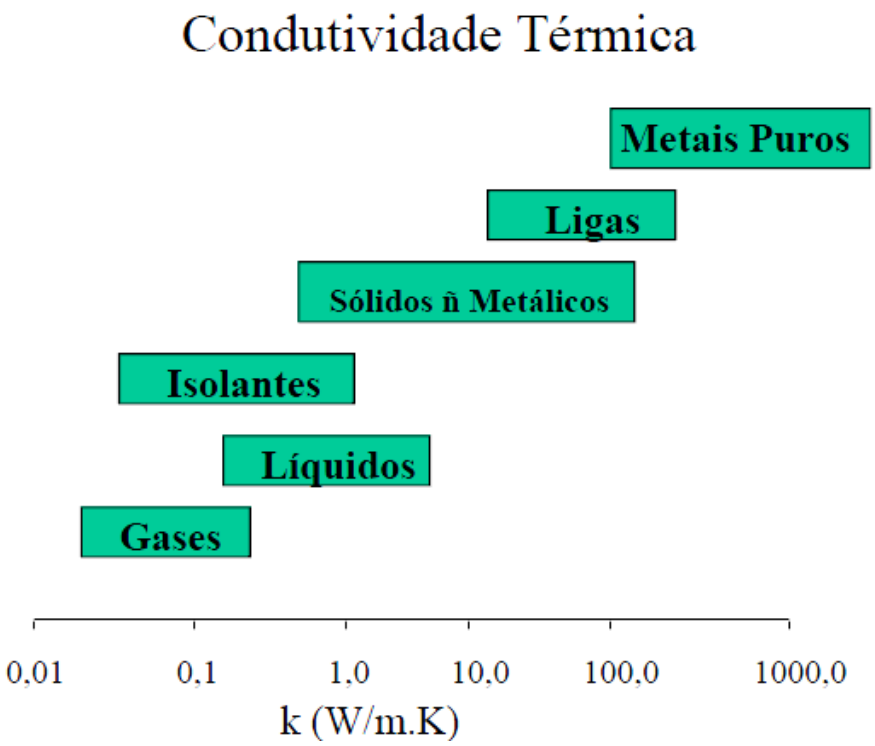
Alguns valores de k para diversos materiais são apresentados na Tab. (2.1). O sinal negativo é devido ao fato do calor ser transferido na direção da temperatura decrescente. Nessa situação o fluxo de calor ocorre em apenas uma dimensão. Essa lei generalizada para um fluxo de calor que ocorra nas 3 dimensões tem a forma

$$\dot{q} = -k\nabla T \tag{2.2}$$

onde ∇T é o gradiente de temperatura.

Tabela 2.1 – Faixas de valores de condutividade térmica para diversos materiais.

(fonte: CALDEIRA,Armando. Condução: Fundamentos. 2011. 19slides. NotasdeAula. ApresentaçãoMSPowerPoint.)



A convecção irá ocorrer quando houver movimento relativo de um fluido em relação a uma superfície, sendo modelada pela lei do resfriamento de Newton como

$$\dot{q} = h(T_s - T_\infty) \tag{2.3}$$

onde $\dot{q}$ é o fluxo térmico (W/m^2), T_s é a temperatura da superfície, T_∞ é a temperatura do fluido em um ponto distante da superfície, ambos dados em Kelvins, e h é o coeficiente de transferência de calor por convecção (W/m^2K) que depende de vários fatores como a geometria da superfície, propriedades do fluido, velocidade do escoamento. Na Figura (2.2) é mostrado um desenho esquemático de convecção em uma placa que possui temperatura maior que a temperatura do fluido.

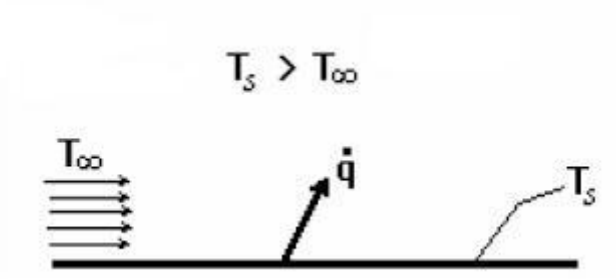


Figura 2.2 – Convecção em uma placa.

A radiação é a energia emitida pela matéria em forma de ondas eletromagnéticas ou fótons, que comportam-se, ao mesmo tempo, como onda e como partícula e deslocam-se com a velocidade da luz. Essa radiação é devida a mudanças nas configurações eletrônicas dos átomos. No espectro da radiação eletromagnética é bem definida a faixa do espectro da radiação térmica (Fig. 2.3), que engloba as faixas do infravermelho, da luz visível e parte do ultravioleta. Os demais comprimentos de ondas não se relacionam com a radiação térmica.

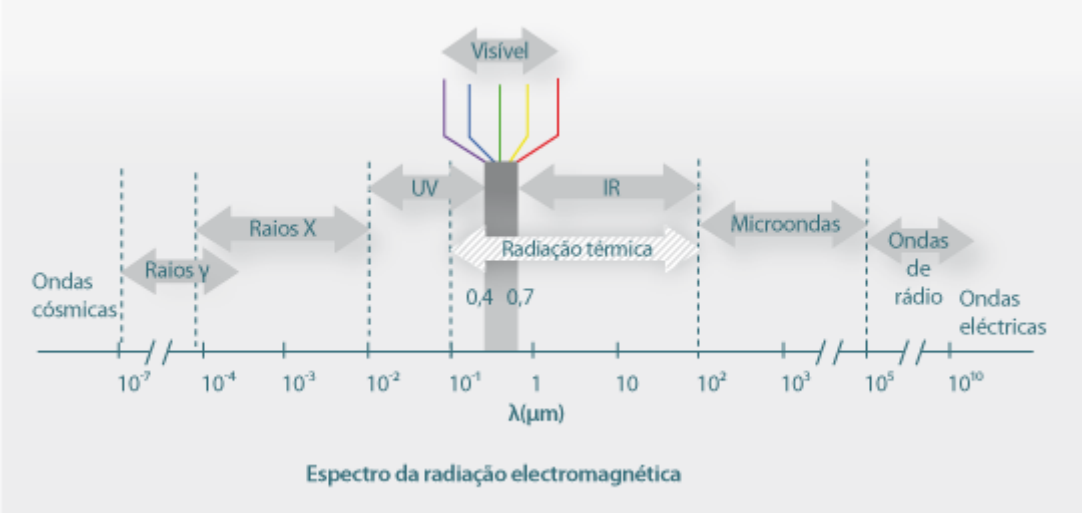


Figura 2.3 – Espectro da radiação térmica.

(fonte: http://labvirtual.eq.uc.pt/siteJoomla/index.php?option=com_content&task=view&id=248&Itemid=422, em 25/05/2012 as 17:00)

A taxa de energia de radiação liberada por uma superfície depende da temperatura que ela se encontra. Todavia, existe um limite máximo para esse poder emissivo, para cada temperatura, que é dado pela lei de Stefan-Boltzmann como

$$E = \sigma T_S^4 \tag{2.4}$$

onde E é o poder emissivo (W/m^2), T_S é a temperatura absoluta da superfície (K) e σ é a constante de Stefan-Boltzmann ($\sigma = 5{,}67.10^{-6}\, W/m^2\, K^4$).

Essa superfície idealizada é conhecida como corpo negro e as superfícies reais emitem menos radiação, para a mesma temperatura, de acordo com sua emissividade, na forma

$$E = \varepsilon \sigma T_S^4 \tag{2.5}$$

onde ε é a referida emissividade, que pode variar entre zero a 1. Outra característica importante de uma superfície é sua absortividade (α), que representa a taxa na qual a energia radiante incidente sobre essa superfície é absorvido por ela. Seu valor também varia de zero a 1 e um corpo negro tem uma absortividade igual a 1, ou seja, absorve toda radiação incidente. A irradiação (G) é toda a radiação

incidente sobre uma superfície (*W*), logo a parcela absorvida será de $\alpha.G$. Com isso a taxa líquida de calor entrando ou saindo de uma superfície será o balanço da sua emissão com sua absorção.

2.2 A CONDUÇÃO DE CALOR EM UMA PAREDE

Geralmente a temperatura no interior de um corpo varia em cada ponto, e também com o tempo. A equação geral da condução de calor para coordenadas retangulares, que considera fluxo de calor nas 3 dimensões, geração interna de calor, regime transiente e condutividade térmica constante é

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{\dot{e}}{k} = \frac{1}{\alpha}\frac{\partial T}{\partial t} \tag{2.8}$$

que também pode ser escrita na forma reduzida como

$$\nabla^2 T + \frac{\dot{e}}{k} = \frac{1}{\alpha}\frac{\partial T}{\partial t} \tag{2.9}$$

onde o primeiro termo da equação representa os gradientes de temperatura em cada direção, o segundo termo corresponde a geração de calor interna e o terceiro representa a variação temporal do campo de temperaturas.

Considere-se agora uma grande parede plana, que possui uma temperatura uniforme inicial T_i e que no instante *t=0* é colocada em um líquido com temperatura constante $T_i < T_\infty$. A parede irá perder calor para o meio por convecção em ambos os lados, com um coeficiente de convecção *h*, que será considerado constante e uniforme. Além disso, o fluxo de calor pode ser considerado unidimensional na direção χ, visto que esta sendo considerado que a parede é infinitamente longa pelo fato de sua espessura ser pequena quando comparada com a altura e a largura. Nesse caso também podem ser desconsiderados os efeitos das bordas que implicariam em não unidimensionalidade. Os perfis de temperatura no interior da parede irão variar com o tempo e essa variação é mostrada na Fig. (2.4).

Percebe-se que existe simetria dos perfis de temperatura em relação ao plano central vertical. Por isso é possível, para simplificar, resolver o problema para o domínio positivo do eixo *x*, e então aplicar o resultado para a outra metade. Logo, por se tratar de um problema unidimensional e que também não possui geração interna de calor, a equação geral para a condução para esse caso é simplificada para

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{\alpha}\frac{\partial T}{\partial t} \tag{2.10}$$

Para a resolução dessa equação diferencial é necessário especificar uma condição inicial e duas condições de contorno. A condição inicial é

$$T(x,0) = T_i \tag{2.11}$$

e as condições de contorno são

$$\frac{\partial T(0,t)}{\partial x} = 0 \tag{2.12}$$

que representa a condição de simetria do plano central da parede e

$$-k\frac{\partial T(L,t)}{\partial x} = h[T(L,t) - T_\infty] \tag{2.13}$$

que descreve a condição na superfície.

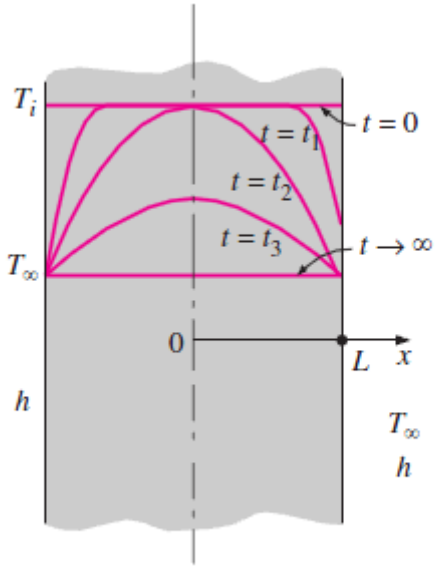


Figura 2.4 – Perfil de temperatura em uma parede submetida à convecção. (fonte: Cengel, 2003)

Nota-se que a determinação desse problema envolve 8 parâmetros e variáveis independentes (*x, t, Ti, T8, L, k, α, h*) para determinar a temperatura *T*. Uma das vantagens da adimensionalização é a diminuição da quantidade de parâmetros. Podemos definir uma temperatura adimensional *θ(x,t)* como a diferença de temperaturas dividida pela máxima diferença de temperatura possível, assim $\theta(x,t) = [\frac{T(x,t)-T_\infty}{T_i-T_\infty}]$. A coordenada espacial adimensional pode ser definida como uma distância do centro adimensional $\chi = x/L$ e um tempo adimensional pode ser definido como $\tau = \alpha t/L2$, também conhecido como número de Fourier. Esse número é uma medida do calor conduzido através de um corpo em relação ao calor armazenado. Substituindo essas equações adimensionais na equação diferencial (Eq. 2.10), na condição inicial (Eq. 2.11) e nas condições de contorno (Eq. 2.12 e 2.13) e rearranjando os termos teremos suas respectivas formas adimensionais

$$\frac{\partial^2\theta}{\partial\chi^2} = \frac{\partial\theta}{\partial\tau} \tag{2.14}$$

$$\theta(\chi, 0) = 1 \tag{2.15}$$

$$\frac{\partial\theta(0,\tau)}{\partial\chi} = 0 \tag{2.16}$$

$$\frac{\partial\theta(1,\tau)}{\partial\chi} = -Bi\,\theta(1,\tau) \tag{2.17}$$

onde *Bi = hL/k* é o número de Biot, e corresponde ao coeficiente de transferência de calor adimensional, uma relação entre a resistência interna de um corpo à condução e a sua resistência externa à convecção. Com todos esses arranjos, agora nosso problema possui apenas 3 parâmetros ou variáveis *(χ, τ, Bi)* para determinação da temperatura adimensional *θ*.

Para determinar a solução desse problema de valores inicial e de fronteira (Eq. 2.14 a 2.17), pode-se usar o método de separação de variáveis (método de Fourier). Primeiramente iremos expressar a temperatura adimensional *θ* como produto de duas funções na forma

$$\theta(\chi, \tau)=F(\chi)G(\tau) \tag{2.18}$$

que substituindo na Eq. (2.12) fica

$$\frac{1}{F}\frac{d^2F}{d\chi^2} = \frac{1}{G}\frac{dG}{d\tau} \tag{2.19}$$

Como existe uma igualdade entre funções que dependem de variáveis independentes diferentes, ambos os lados da equação precisam ser iguais a uma constante. Assim

$$\frac{1}{F}\frac{d^2F}{d\chi^2} = -\lambda^2 \tag{2.20}$$

$$\frac{1}{G}\frac{dG}{d\tau} = -\lambda^2 \tag{2.21}$$

onde $-\lambda^2$ é uma constante independente de χ e de τ, e foi definida com esse formato por conveniência. De fato a constante não pode assumir valores positivos pois faria com que a função *G(τ)* crescesse indefinidamente e não pode ser nula pois implicaria não ter dependência com o tempo.

Dessa forma transformamos a solução de uma equação diferencial parcial (Eq. 2.14), na solução de duas equações diferenciais ordinárias (Eq. 2.20 e 2.21), cujas soluções gerais são respectivamente

$$F = C_1\cos(\lambda\chi) + C_2\text{sen}(\lambda\chi) \tag{2.22}$$

$$G = C_3 e^{-\lambda^2\tau} \tag{2.23}$$

e assim

$$\theta = F(\chi)G(\tau) = e^{-\lambda^2\tau}[A\cos(\lambda\chi) + B\text{sen}(\lambda\chi)] \tag{2.24}$$

onde $A = C_1C_3$ e $B = C_2\,C_3$ são constantes.

Substituindo essa função θ nas condições de contorno encontramos que

$$\theta = Ae^{-\lambda^2\tau}\cos(\lambda\chi) \tag{2.25}$$

$$\lambda\tan\lambda = Bi \tag{2.26}$$

A Equação (2.26) é uma autofunção com infinitas raízes (autovalores), e por ser uma equação implícita os autovalores precisam ser determinados numericamente. Logo, a Equação (2.25) irá possuir um número infinito de soluções. Por ser um problema linear, a combinação linear de quaisquer soluções também irá ser uma solução, assim

$$\theta = \sum_{n=1}^{\infty} A_n e^{-\lambda_n^2\tau} cos(\lambda_n\chi) \tag{2.27}$$

Usando a condição inicial (Eq. 2.15) pode-se determinar as constantes A_n, assim

$$\sum_{n=1}^{\infty} A_n cos(\lambda_n\chi) = 1 \tag{2.28}$$

e multiplicando os dois lados da equação por *cos($\lambda_n\chi$)* e integrando em χ de 0 a 1 tem-se

$$\int_0^1 \cos(\lambda_n\,\chi)d\chi = A_n \int_0^1 \cos^2(\lambda_n\,\chi)d\chi \tag{2.29}$$

e então

$$A_n = \frac{4sen\lambda_n}{2\lambda_n} + sen(2\lambda_n). \tag{2.30}$$

Com isso a determinação das temperaturas adimensionais transientes na parede é dada pela soma da série infinita da Eq. (2.27) e os respectivos termos A_n são dados pela Eq. (2.30). Os termos do somatório diminuem rapidamente por isso normalmente se usa apenas os primeiros como uma aproximação razoável. Os gráficos de Heisler (Fig. 1.1) apresentam essa solução na forma gráfica e usam apenas o primeiro termo da série (Eq. 2.31), possuindo, segundo Incropera (2008), com essa aproximação um erro menor que 2%.

$$\theta = A_1 e^{-\lambda_1^2\tau} cos(\lambda_1 x/L) \tag{2.31}$$

Para determinar a temperatura no centro da parede faz-se x=0 na Eq. (2.31), obtendo-se

$$\theta(0,\tau) = A_1 e^{-\lambda_1^2\tau} \tag{2.32}$$

e fazendo a razão entre as Eq. (2.31) e (2.32) chega-se a relação

$$\frac{\theta}{\theta(0,\tau)} = cos(\lambda_1 x/L) \tag{2.33}$$

o que mostra a temperatura adimensional em qualquer ponto irá variar na mesma taxa. A solução gráfica aproximada da relação da Eq. 2.32 completa os gráficos de Heisler (Fig. 2.5).

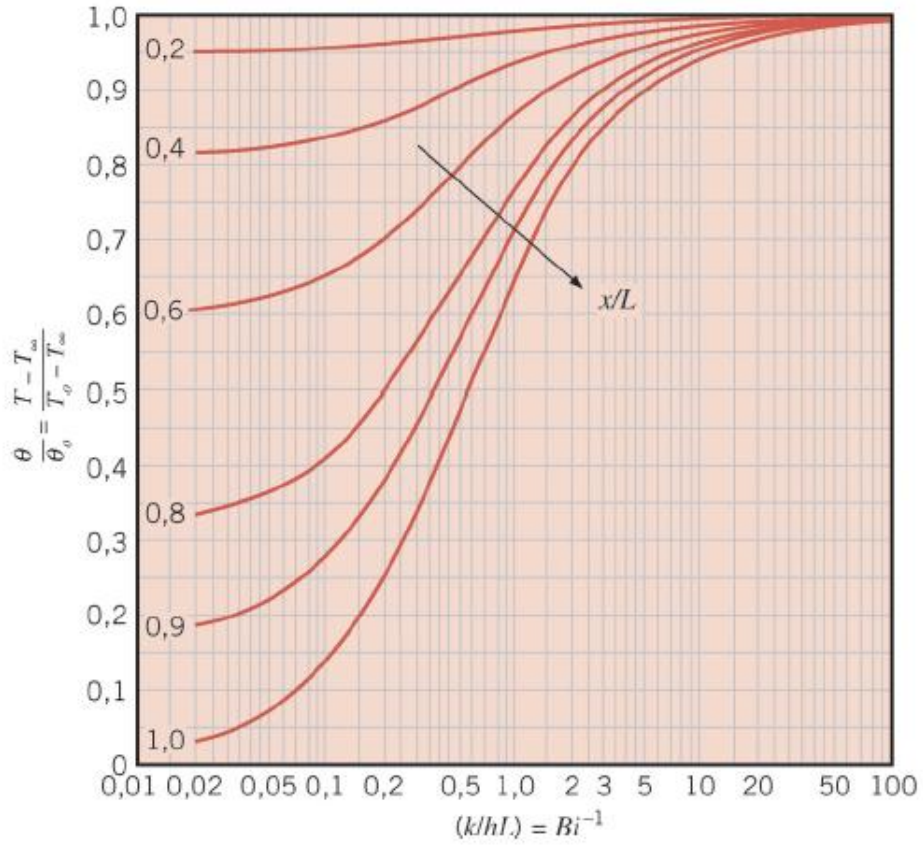


Figura 2.5 – Gráficos de Heisler para a temperatura adimensional em algum ponto da parede. (fonte: Incropera, 2008)

A terceira parte desse gráfico corresponde ao gráfico de transferência de calor transiente de H. Grober, que serve para determinação da quantidade total de calor transferido até o tempo *t*. Da termodinâmica sabe-se que a quantidade total de energia $Q_{MÁX}$ que flui de um corpo de massa *m* e calor específico c_p com temperatura inicial T_i para um ambiente a temperatura T_∞ até o equilíbrio térmico é

$$Q_{MÁX} = mc_p(T_\infty - T_i) \tag{2.34}$$

Para calcular a quantidade de calor transferido *Q* até um determinado tempo *t* pode-se substituir *m* por *ρυ* (densidade vezes volume) e integrar sobre toda a geometria do corpo

$$Q = \int_\upsilon \rho c_p[T(x,t) - T_i]d\upsilon \tag{2.35}$$

e fazer a relação entre Q/QMÁX

$$\frac{Q}{Q_{MÁX}} = \frac{\int_\upsilon \rho c_p[T(x,t) - T_i]d\upsilon}{\rho \upsilon c_p(T_\infty - T_i)} = \frac{1}{\upsilon}\int_\upsilon (1-\theta)d\upsilon \tag{2.36}$$

e então, usando novamente a aproximação de um termo, e efetuando as integrais tem-se

$$\frac{Q}{Q_{MÁX}} = 1 - \theta(O,\tau)\frac{sen\,\lambda_1}{\lambda_1} \tag{2.37}$$

portanto o gráfico de H. Grober (Fig. 2.6) corresponde a solução gráfica aproximada dessa relação.

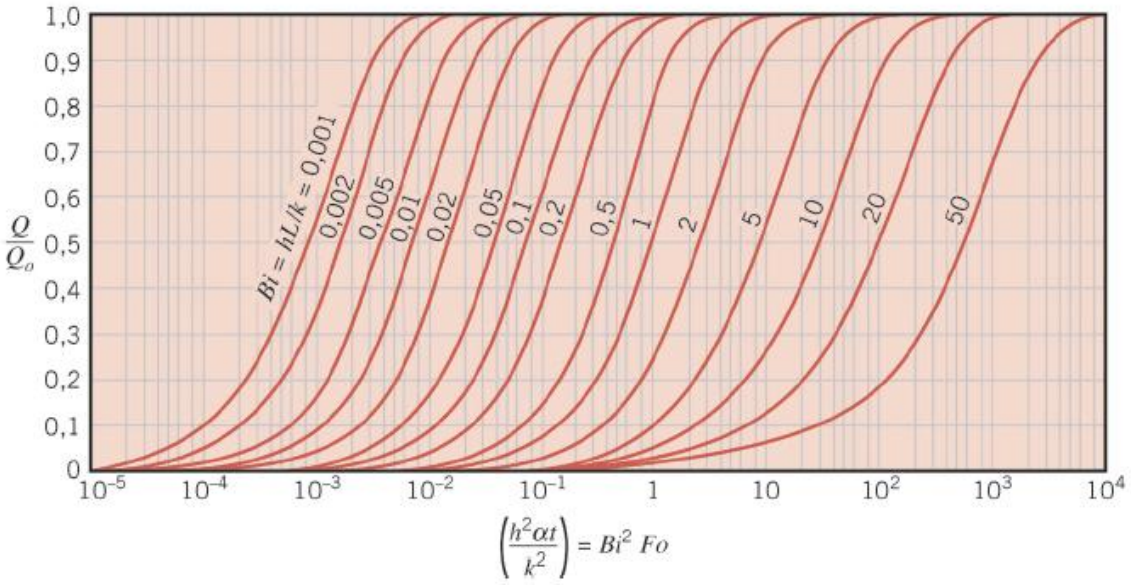


Figura 2.6 – Gráficos de Grober do fluxo de calor para condução transiente para a parede. (fonte: Incropera, 2008)

Para condições de contorno especificadas nesse problema foi considerado as duas faces da parede submetidas à convecção. Contudo, os resultados obtidos podem ser aplicados para o caso de uma superfície isolada e outra sob ação da convecção, visto que a formulação será a mesma. Toda formulação matemática desenvolvida até aqui será usada na programação da interface.

2.3 CONDUÇÃO EM CILINDRO E ESFERA

Pode-se generalizar a formulação do problema de condução em uma parede longa para os casos de um longo cilindro e uma longa esfera, pois a análise será a mesma. O restante dos gráficos de Heisler/Grober para o cilindro e a esfera, Fig. (2.7) e (2.8), irão surgir da solução gráfica das equações obtidas.

No caso da parede, a variável espacial é a coordenada x e o comprimento característico é a metade de sua espessura (L/2). Para o cilindro e a esfera a variável espacial será o raio r do ponto considerado e o comprimento característico será o raio total r0. As equações da temperatura adimensional transiente, da temperatura no centro da geometria e da quantidade de energia transferida, para o cilindro, analogicamente as equações da parede (Eq. 2.26, 2.27, 2.30, 2.31, 2.32, 2.33) são

$$\theta = \sum_{n=1}^{\infty} A_n e^{-\lambda_n^2 \tau} J_0(\lambda_n r/r_0) \tag{2.38}$$

$$A_n = \frac{2\,J_1(\lambda)}{\lambda\,J_0(\lambda)^2 + J_1(\lambda)^2} \tag{2.39}$$

$$\lambda_n \frac{J_1(\lambda)}{J_0(\lambda)} = Bi \tag{2.40}$$

$$\theta = A_1 e^{-\lambda_1^2 \tau} J_0(\lambda_1 r/r_0) \tag{2.41}$$

$$\theta(0,\tau) = A_1 e^{-\lambda_1^2 \tau} \tag{2.42}$$

$$\frac{Q}{Q_{MÁX}} = 1 - 2\theta(O,\tau)\frac{J_1(\lambda_1)}{\lambda_1} \tag{2.43}$$

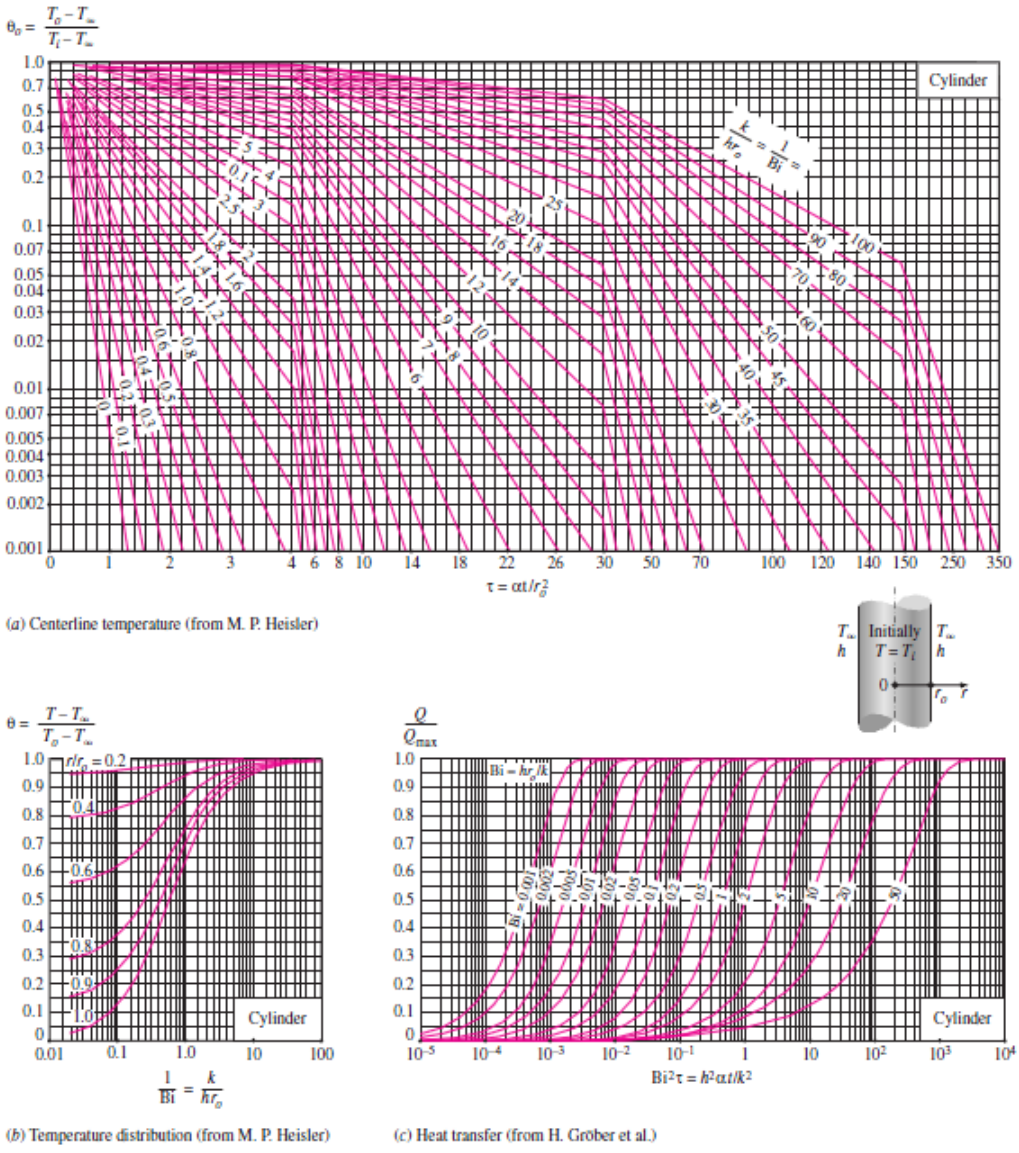


Figura 2.7 – Gráficos de Heisler/Grober de condução transiente para o cilindro. (fonte: Cengel, 2003)

Para a esfera tem-se

$$\theta = \sum_{n=1}^{\infty} A_n e^{-\lambda_n^2 \tau} \frac{sen\,(\lambda_n r/r_0)}{\lambda_n r/r_0} \tag{2.44}$$

$$A_n = \frac{4\,(\mathrm{sen}\lambda_n - \lambda_n cos\lambda_n)}{2\lambda_n - sen(2\lambda_n)} \tag{2.45}$$

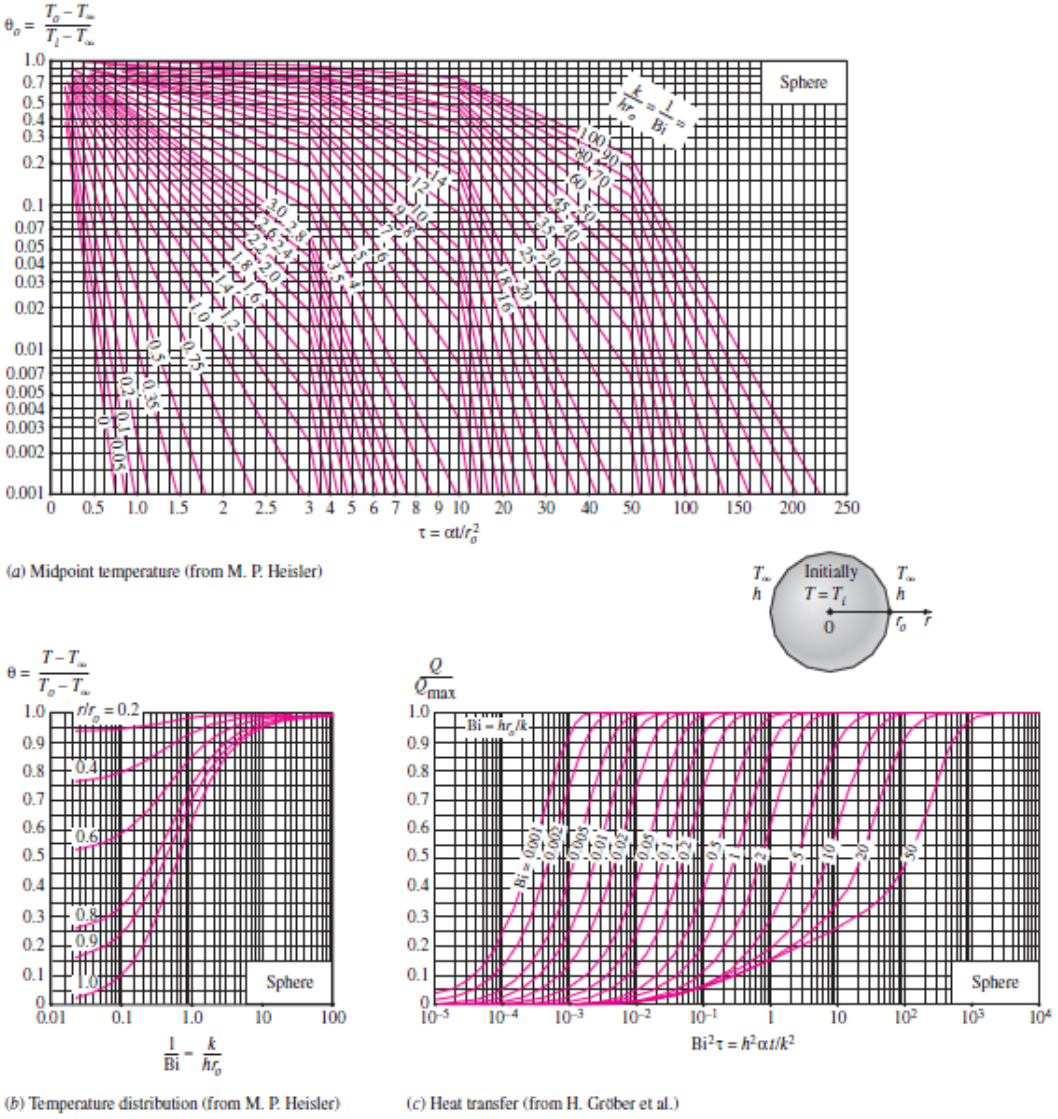
$$1 - \lambda_n cot\lambda_n = Bi \tag{2.46}$$

$$\theta = A_1 e^{-\lambda_1^2 \tau} \frac{sen\,(\lambda_1 r/r_0)}{\lambda_1 r/r_0} \tag{2.47}$$

$$\theta(0,\tau) = A_1 e^{-\lambda_1^2 \tau} \tag{2.48}$$

$$\frac{Q}{Q_{MÁX}} = 1 - 3\theta(O,\tau) \frac{sen\,\lambda_1 - \lambda_1 \cos\lambda_1}{\lambda_1^3} \tag{2.49}$$

onde J_0 e J_1 são as funções de Bessel do primeiro tipo de ordem zero e de primeira ordem respectivamente.



(a) Midpoint temperature (from M. P. Heisler)

(b) Temperature distribution (from M. P. Heisler)

(c) Heat transfer (from H. Gröber et al.)

Figura 2.8 - Gráficos de Heisler/Grober de condução transiente para a esfera. (fonte: Cengel, 2003)

2.4 CONDUÇÃO EM SÓLIDOS SEMI-INFINITOS

Um sólido semi-infinito é um corpo idealizado que possui uma superfície plana e se estende em todas as direções até o infinito, conforme montrado na Fig (2.9). Nesse caso, as condições da superfície irão determinar as mudanças de temperatura próximas a ela, que é a região de interesse. Uma parede muito grossa pode ser modelada como um sólido semi-infinito ao se considerar a variação de temperatura próximo à superfície. A Terra também pode ser considerada como um sólido semi-infinito.

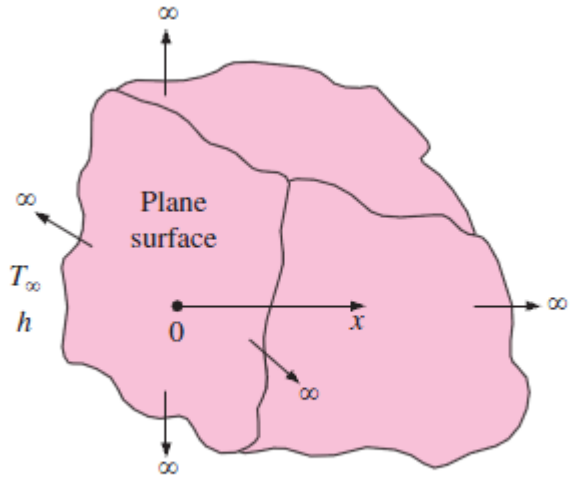


Figura 2.9 – Sólido semi-infinito.

Na verdade, qualquer corpo pode ser modelado como um sólido semi-infinito por um período de tempo pequeno, visto que o calor não tem tempo suficiente para ao penetrar no corpo e, dessa forma, a espessura não entra na análise.

Para a presente análise considera-se que o solido possui propriedades constantes, não existe geração interna de calor e possui uma temperatura inicial uniforme. Nesse caso a transferência de calor ocorre apenas na direção normal à superfície (unidimensional). A equação diferencial que modela essa transferência é a mesma da parede plana, Eq (2.10)

$$\frac{\partial^2 \mathrm{T}}{\partial \mathrm{x}^2} = \frac{1}{\alpha}\frac{\partial \mathrm{T}}{\partial \mathrm{t}} \qquad 2.50$$

Se considerarmos que a temperatura da superfície é alterada para T_s e mantida constante e que a profundidade do sólido é grande em comparação com a profundidade que o calor pode penetrar, temos as condições de contorno

$$T(0,t) = T_s \qquad (2.51)$$

$$T(x \to \infty, 0) = T_i \qquad (2.52)$$

E a condição inicial

$$T(x,0) = T_i \tag{2.53}$$

O método de separação de variáveis não pode ser usado nesse caso pois o meio é infinito. Uma outra solução é definir uma variável de similaridade η que combine as variáveis *x* e *t*. A variável é definida como $\eta = \frac{x}{\sqrt{4\alpha t}}$. Substituindo nas Eq. (2.50), (2.51) e (2.52) temos

$$\frac{\partial^2 T}{\partial \eta^2} = -2\eta \frac{\partial T}{\partial \eta} \tag{2.54}$$

$$T(0) = T_s \tag{2.55}$$

$$T(\eta \to \infty, 0) = T_i \tag{2.56}$$

Agora define-se um nova variável $w = dT/d\eta$. Substituindo na Eq. (2.54), resolvendo a equação diferencial de primeira ordem resultante e voltando para a variável η tem-se

$$T = C_1 \int_0^\eta e^{-u^2} du + C_2 \tag{2.57}$$

onde as constantes C_1 e C_2 são dadas pelas condições de contorno. Calculando as constantes e substituindo na Eq. (2.57) temos que

$$\frac{T - T_s}{T_i - T_s} = \frac{2}{\sqrt{\pi}} \int_0^\eta e^{-u^2}\, du \tag{2.58}$$

A função $\int_0^\eta e^{-u^2}\, du$, também chamada de função erro, não pode ser resolvida analiticamente. Por isso, na literatura, são usados tabelas ou gráficos resultados da avaliação numérica da função erro, como apresentado na Tab. Xx.

O fluxo de calor na superfície pode então ser determinado pela lei de Fourier como

$$q_s = \frac{\dot{k}(T_s - T_i)}{\sqrt{\pi \alpha t}} \tag{2.59}$$

2.3 TRANSFERÊNCIA DE CALOR POR RADIAÇÃO

A radiação eletromagnética envolve a emissão de energia interna de um corpo devido a mudanças nas configurações eletrônicas dos átomos ou moléculas, e se da através de ondas eletromagnéticas, que são campos elétricos e magnéticos em movimento.

Uma superfície plana emite radiação em todas as suas partes e em todas as direções no hemisfério acima da superfície. Essa distribuição direcional da radiação emitida geralmente não é uniforme, ela varia com a direção considerada. A intensidade de radiação I $[\frac{W}{m^2 sr}]$ é uma quantidade que descreve essa amplitude da radiação emitida em uma determinada direção no espaço. Um corpo

negro emite radiação com mesma intensidade em todas as direções. A unidade *sr* (steradian) se refere ao ângulo sólido referente a emissão. Para poder quantificar a espaço abrangido por uma determinada emissão usa-se o ângulo sólido *w* que delimita a área de parte de uma superfície de uma esfera onde a emissão se propaga.

A quantidade de calor que é trocado entre duas superfícies qualquer depende, além das propriedades de radiação e temperatura das superfícies, de suas orientações. Uma pessoa perto de uma fogueira irá receber maior quantidade de calor por radiação se ficar de frente do que de lado. O fator de forma (F_{ij}) é uma propriedade geométrica que mensura o efeito da orientação entre superfícies, ou seja, F_{ij} significa a parcela de radiação deixando a superfície i que atinge a superfície j.

O fator de forma de uma superfície 1 em relação a uma superfície 2 é dado por

$$F_{12} = \frac{1}{A_1}\int_{A_2}\int_{A_1}\frac{cos\theta_1\, cos\theta_2}{\pi r^2}\, dA_1 dA_2 \tag{2.60}$$

onde A_1 e A_2 são as áreas das superfícies, θ_1 e θ_2 são os ângulos de inclinação das superfícies uma em relação à outra, a partir da direção normal de cada superfície e *r* é a distância entre ambas.

Porém, mesmo para geometrias simples, as integrações resultantes são geralmente muito complexas e difíceis de resolver. Em diversas literaturas e publicações existem resultados dessas soluções para várias geometrias comuns apresentados na forma gráfica e de tabelas, como na Fig (2.10), que mostra o gráfico dos fatores de forma para duas placas perpendiculares com uma aresta em comum.

Devido as soluções serem apresentadas na forma gráfica e de tabelas tem-se, assim como no caso da condução, os problemas relacionados à aproximação e propagação de erros na análise gráfica ou de interpolação. Dessa forma, a interface gráfica desenvolvida também engloba a determinação dos fatores de forma da radiação para diversas geometrias mais usais, através da resolução da Eq. (2.60).

Para essa análise considera-se que a intesidade de radiação *I* é constante, ou seja, as superfícies emitem radiação uniformemente em todas as direções (superfícies difusas) em toda a superfície (isotérmicas). Outra consideração é que o meio entre as superfícies é não participante, ou seja, não absorve nem emite radiação, como o vácuo ou o ar.

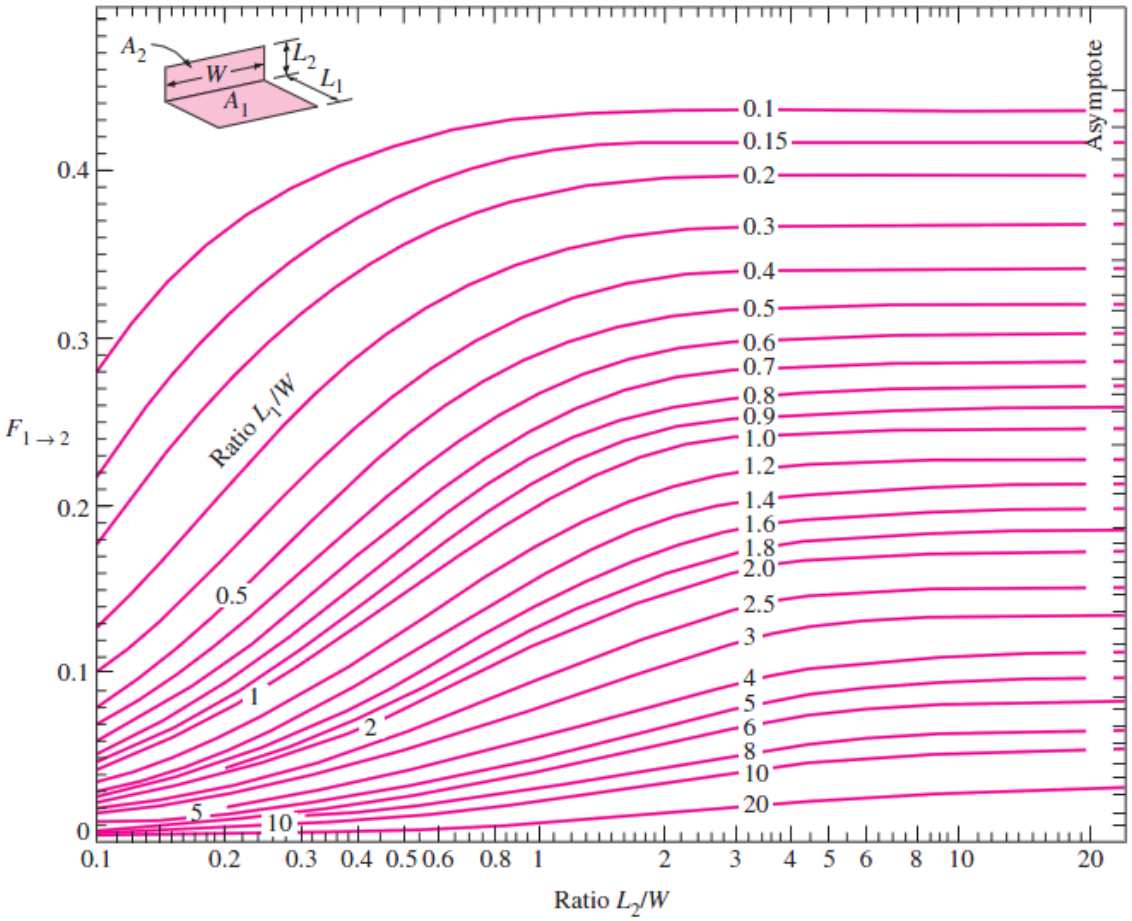


Figura 2.10 – Fatores de forma para duas placas perpendiculares com uma aresta em comum. (Fonte Cengel 2003)

...

3 DESENVOLVIMENTO E ANÁLISE DA INTERFACE

Nesse capítulo é feito o desenvolvimento da interface juntamente com a análise dos resultados obtidos.

3.1 O MATLAB

Existem várias linguagens de programação capazes de realizar cálculos numéricos e desenvolver interfaces gráficas, como Matlab, Fortran, Básic, Phython, C. O Matlab é uma linguagem didática, utilizada em diversas disciplinas do curso de engenharia de mecânica da UnB, pois possui muitas funcionalidades matemáticas e físicas, além de cálculo matricial, processamento de sinais e construção de gráficos, apresentando um ambiente fácil de programação quando comparado a outras linguagens.

A interface proposta pode ser construída em outras linguagens, como as mencionadas acima. Um dos fatores que fizeram o Matlab ser adotado foi a boa familiaridade com a linguagem que os autores do projeto possuiam. Assim, apesar do Matlab não ser um software de alta performance quando comparado a outras linguagens citadas, sua capacidade se mostrou suficiente para cumprir os objetivos do presente trabalho, somado ao fato de possuir diversos recursos e bibliotecas de cálculo.

3.2 CONCEPÇÃO DA INTERFACE

A primeira janela apresentada ao usuário serve para selecionar qual das 5 opções será executada, apresentada na Fig. (3.1). Existe um grupo de seleções para a "condução em regime transiente" composto de 4 opções mais a opção para "radiação".

A proposta do programa elaborado para a parte de condução é obter os valores de temperatura no sólido para um tempo especificado, ou o tempo necessário para atingir uma dada temperatura, e a quantidade de calor transferida durante o processo, dados os parâmetros do material e do meio, e o ponto de interesse no sólido.

Foi proposto ainda exibir o perfil de temperatura e uma animação para ilustrar o fenômeno. O arquivo que apresentará a solução é composto de programas distintos, cada um responsável pela análise do problema aplicado as formas da esfera, do cilindro infinito, da parede plana infinita e do sólido semi-infinito. Os cálculos numéricos e a plataforma da interface foram ambos desenvolvidos no Matlab.

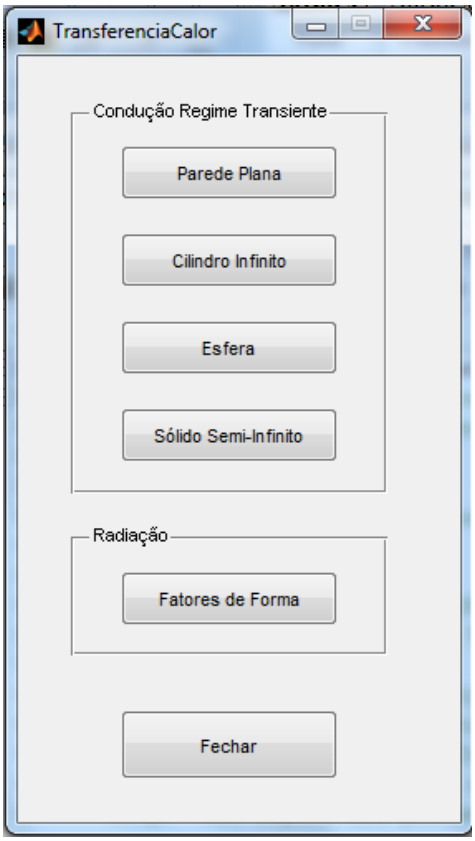


Figura 3.1 – Tela inicial da interface.

Ao escolher a geometria desejada, a interface que aparecerá será similar para os três programas, com as exceções de que o comprimento característico é identificado como L para a parede plana e R para a esfera e o cilindro, bem como o comprimento que localiza o ponto de interesse é l e r respectivamente, conforme as equações desenvolvidas no Cap. 2.

As Figuras (3.2), (3.3), (3.4) e (3.5) mostram a interface proposta em caso de escolha da opção para a parede, esfera, cilindro e semi-infinito respectivamente.

Nos campos de propriedades do material, o usuário deverá fornecer os valores dos parâmetros do material constituinte da geometria: condutividade térmica (K), calor específico (C_p), comprimento característico (L ou R), massa específica (ρ) e temperatura inicial (T_i). Nos campos de propriedades do meio o usuário deverá incluir os valores do coeficiente de convecção (h) e da temperatura do meio externo (T).

Definidos os parâmetros do sistema o usuário deverá escolher qual tipo de solução que deseja no campo “Tipo de Solução”. Nesse campo terá duas opções.

"Resolver para a Temperatura": ao escolher essa opção deverá preencher os campos "Variáveis no Ponto de Interesse" com a localização do ponto onde deseja conhecer a temperatura (definido como a variável "l" para a parede e "r" para o cilindro e esfera, dados em metros) e preencher para qual tempo deseja conhecer essa temperatura (variável t, dado em segundos). Como resultado aparecerá o valor da temperatura para o ponto e para o tempo escolhido na parte inferior da interface. Nessa região também aparecerá os valores no número do biot e do número de fourrier para o problema. Um exemplo dessa resolução para a esfera e mostrada na fig. (3.6), com os campos preenchidos com determinados valores e os resultados obtidos nos devidos campos. Na interface também é apresentada uma figura onde mostra o perfil em cores da temperatura na esfera no instante considerado. À direita da figura existe a legenda das cores. Abaixo da figura existe um gráfico em coordenadas cartesianas da temperatura pelo tempo com a indicação do ponto considerado.

"Resolver para o Tempo": nessa opção o usuário entrará novamente com a localização do ponto porém agora o outro campo será preenchido com o valor da temperatura desejada (variável *T*, dada em graus celsius). Como resultado terá o tempo necessário para que o ponto escolhido atinja essa temperatura especificada. Na figura (3.7) é apresentado um exemplo para essa opção com os mesmos valores dos parâmetros do sistema usados no exemplo anterior. O gráfico cartesiano apresentado nesse caso é do tempo pela temperatura.

No canto inferior central da interface existe um comando de animação. Esse comando simula a mudança do perfil de temperatura no tempo. Ao ser executada a simulação irá começar do tempo zero até o atingir o equilíbrio térmico.

A quinta opção da tela inicial é a que leva à interface dos fatores de forma de radiação, mostrada Fig. (3.8). Nessa tela o usuário terá a opção de escolher entre geometrias bidimensionais e tridimensionais mais usuais. Ao escolher a geometria irão aparecer os respectivos campos para preenchimento das dimensões e as temperaturas das superfícies.

Os valores das dimensões podem ser colocados em qualquer sistema de unidade, visto que o valor adimensional do fator de forma será o mesmo. Já os campos com os valores das temperaturas devem ser colocados em Kelvins. O valores do fator de forma F_{ij} e da troca líquida de calor Q_{ij} são calculados e apresentados.

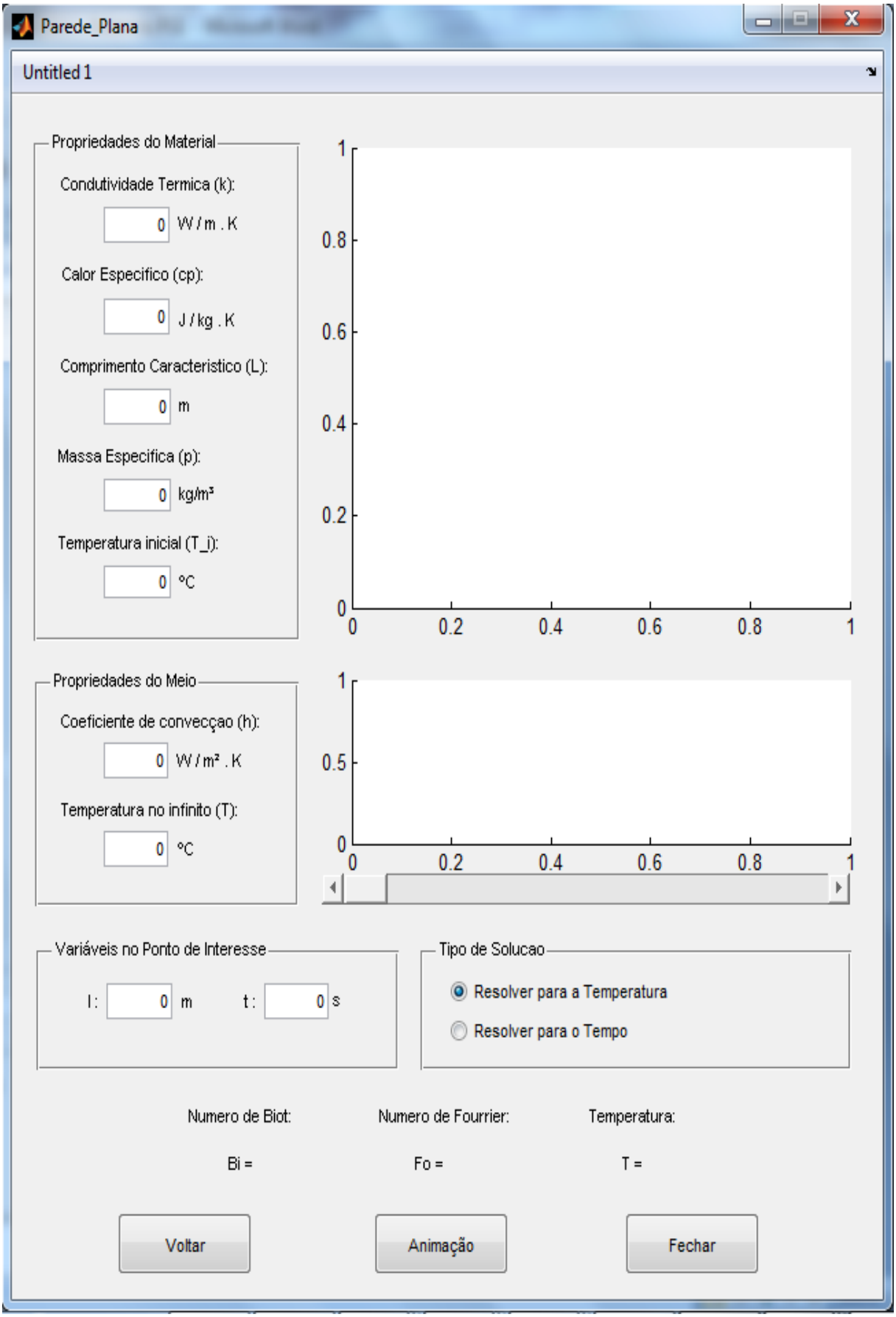


Figura 3.2 – Interface de resolução da condução transiente para a parede.

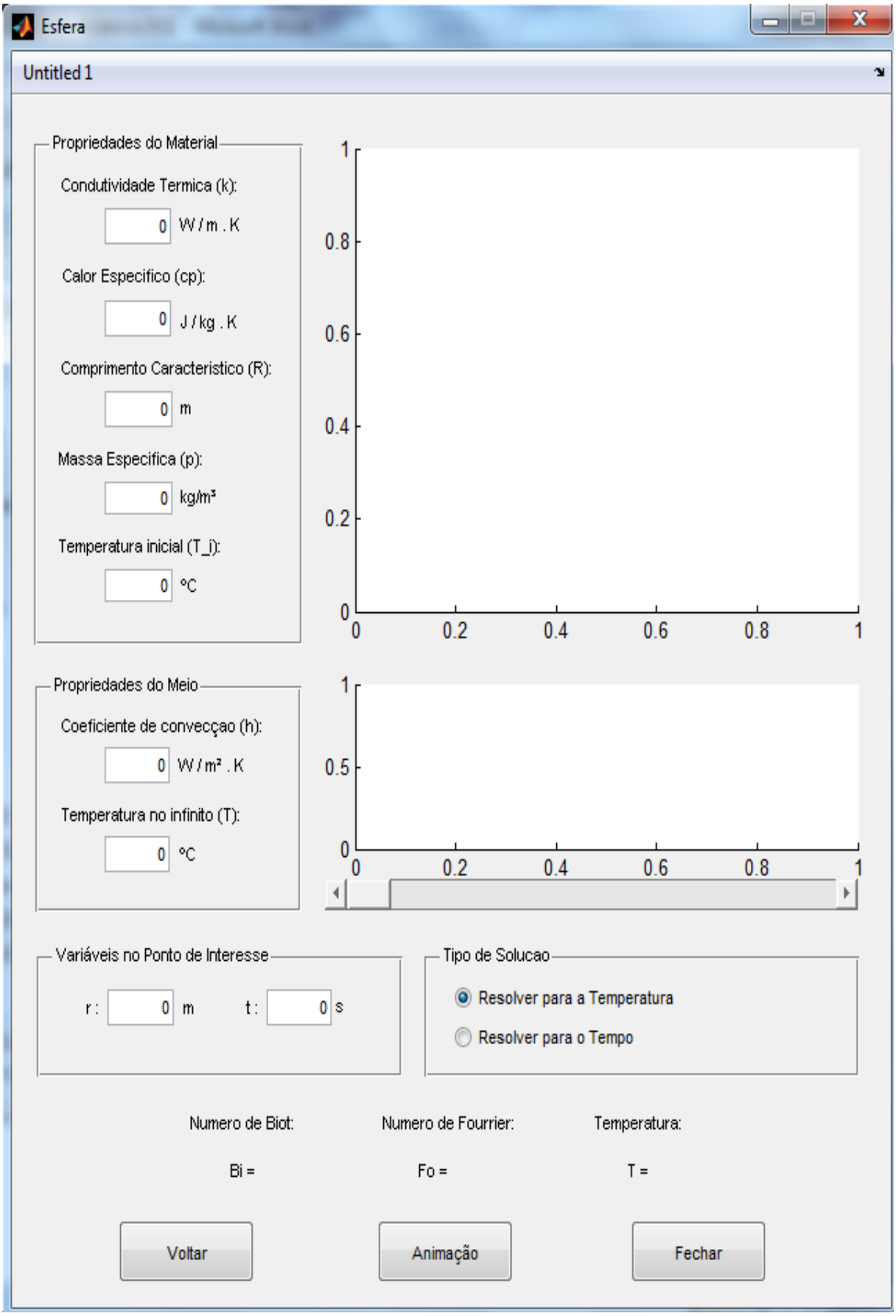


Figura 3.3 - Interface de resolução da condução transiente para a esfera.

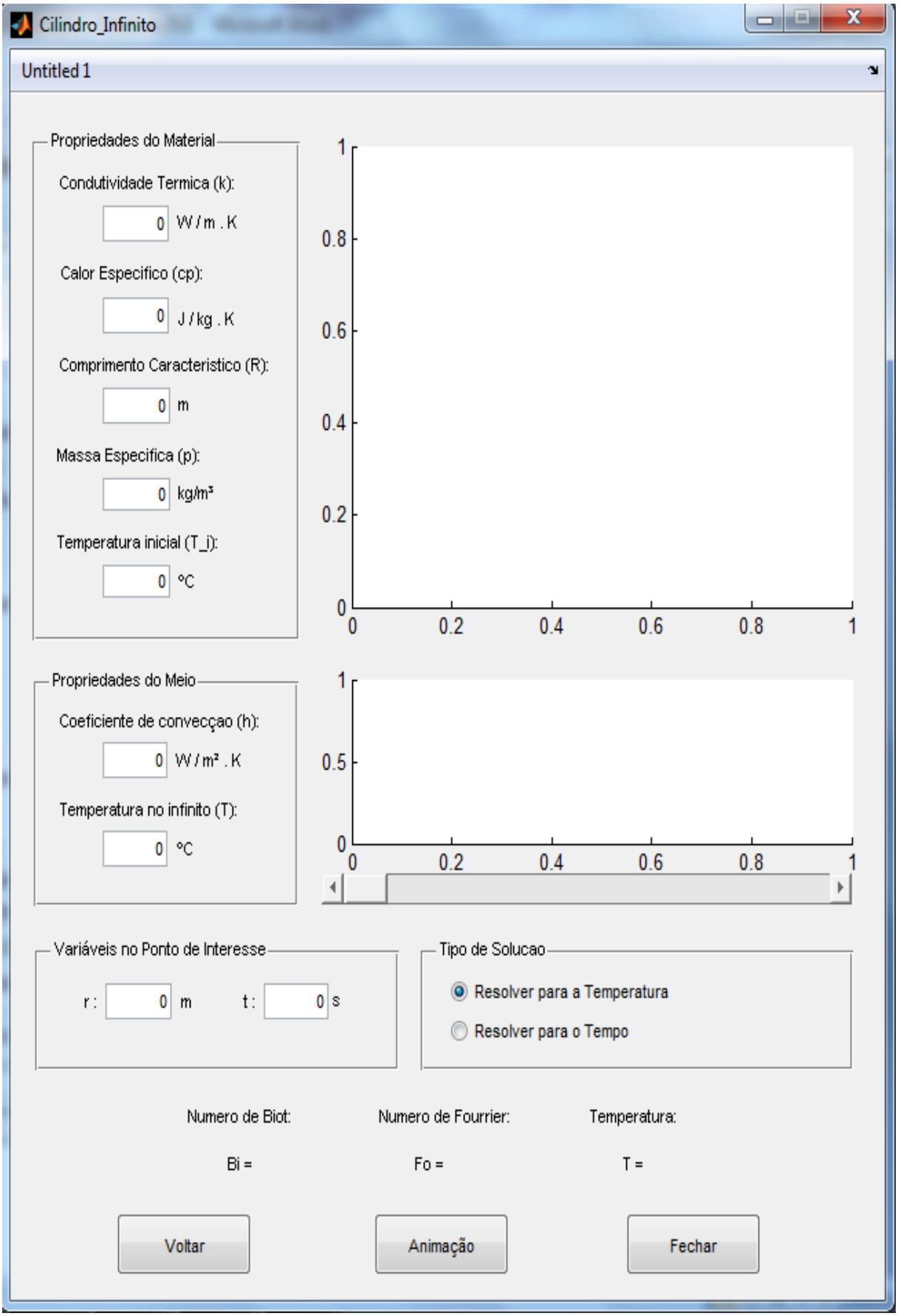


Figura 3.4 - Interface de resolução da condução transiente para o cilindro.

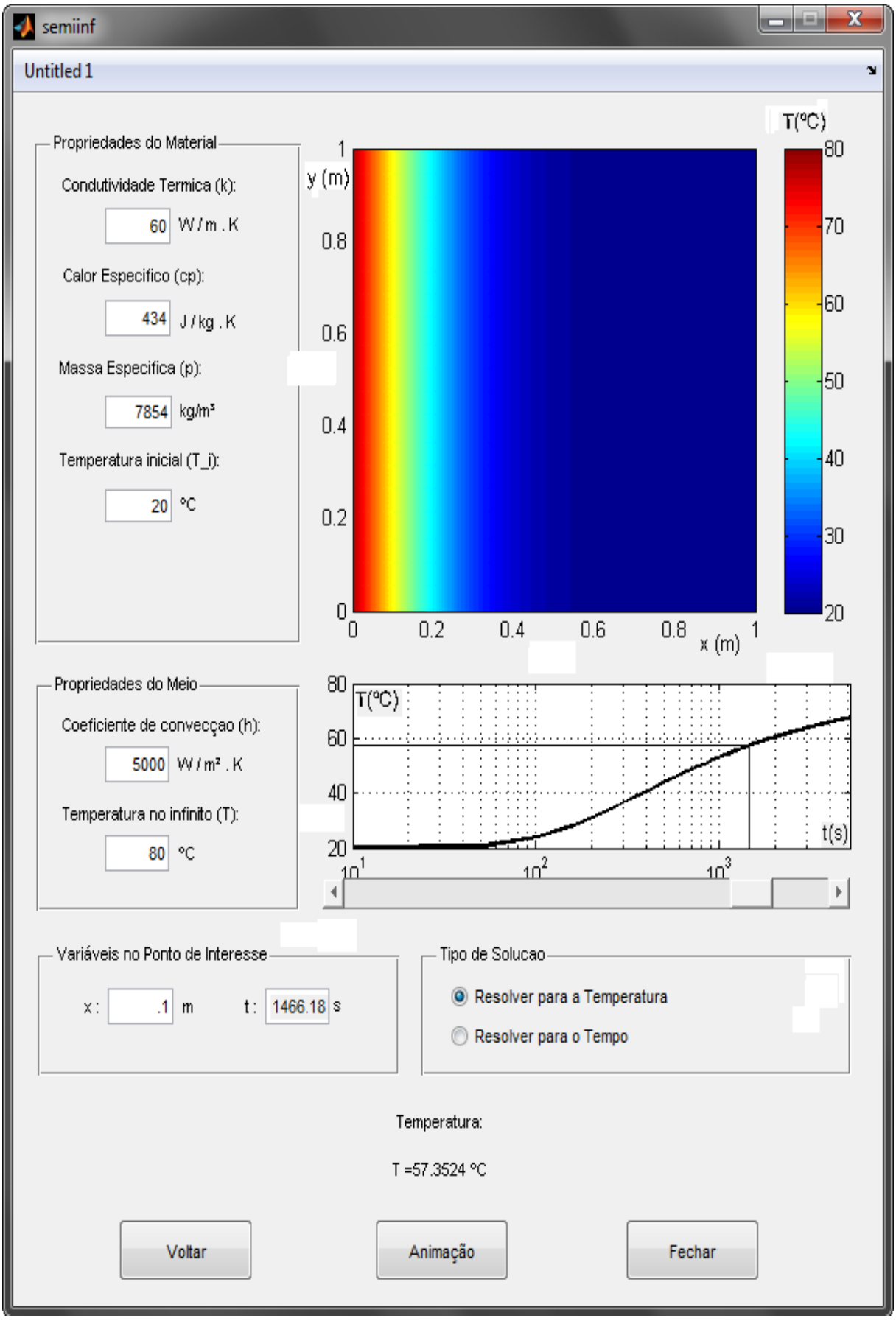


Figura 3.5 – Exemplo de resolução do solido semi-infinito para a temperatura.

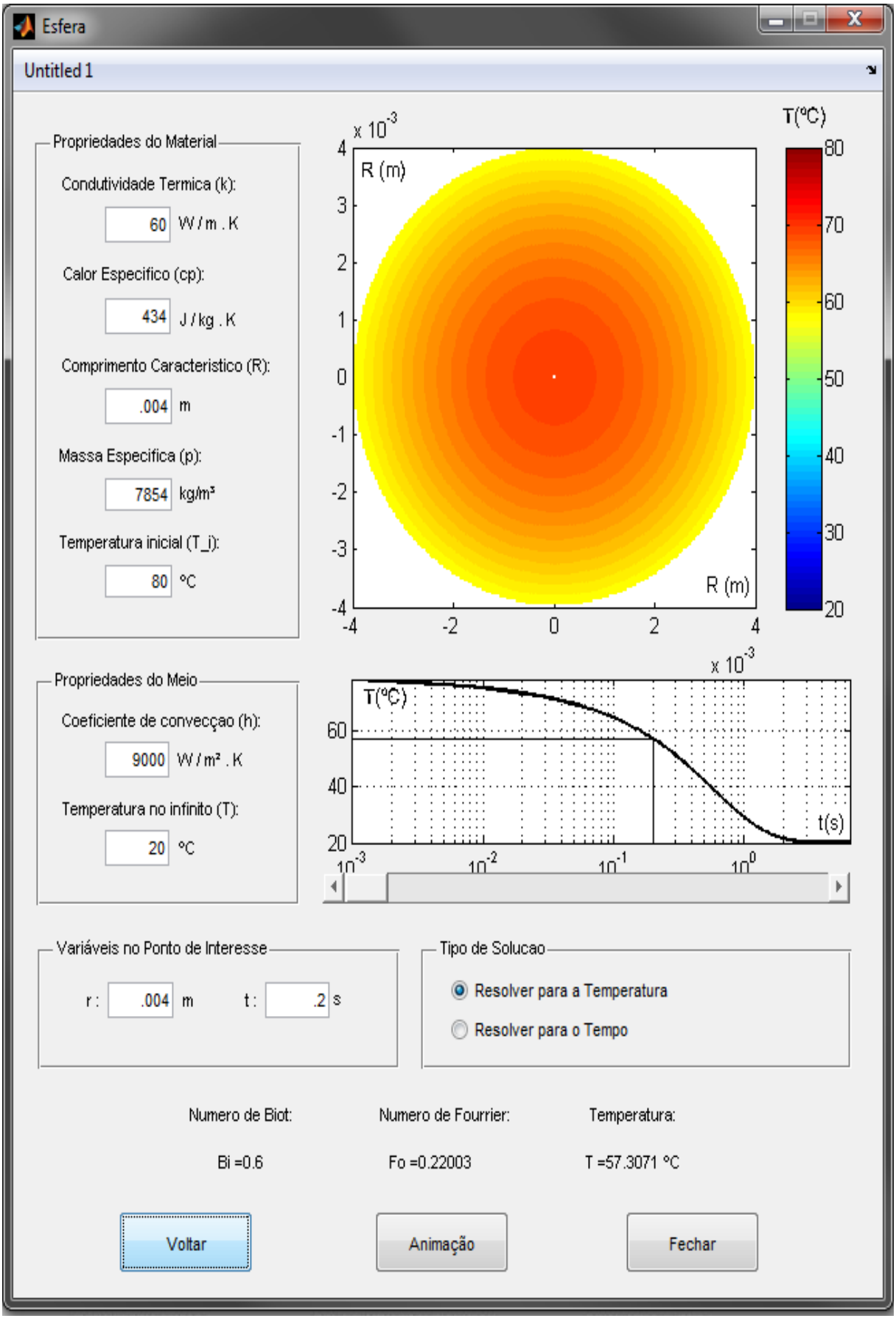


Figura 3.6 – Exemplo de resolução da esfera para a temperatura.

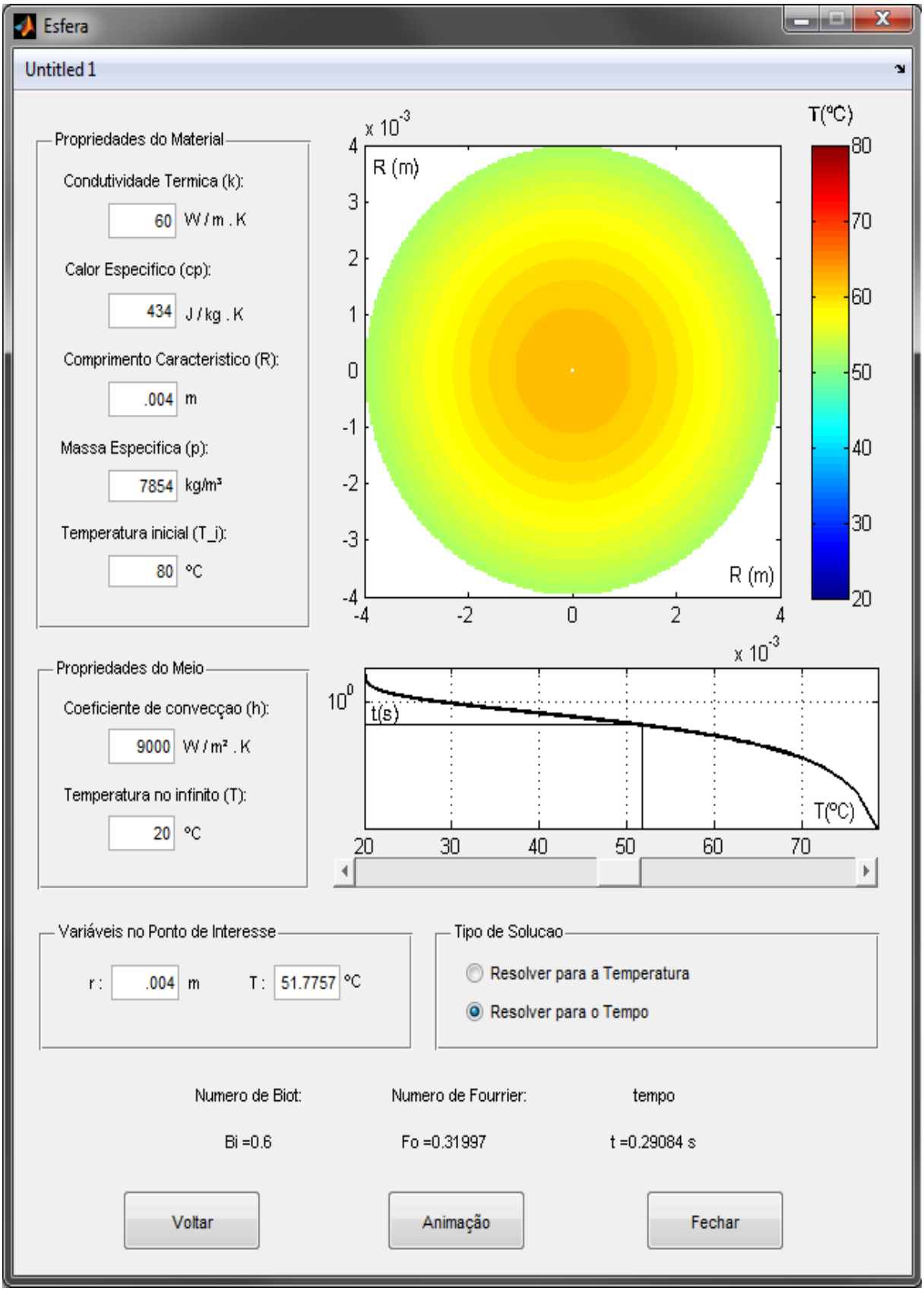


Figura 3.7 - Exemplo de resolução da esfera para o tempo.

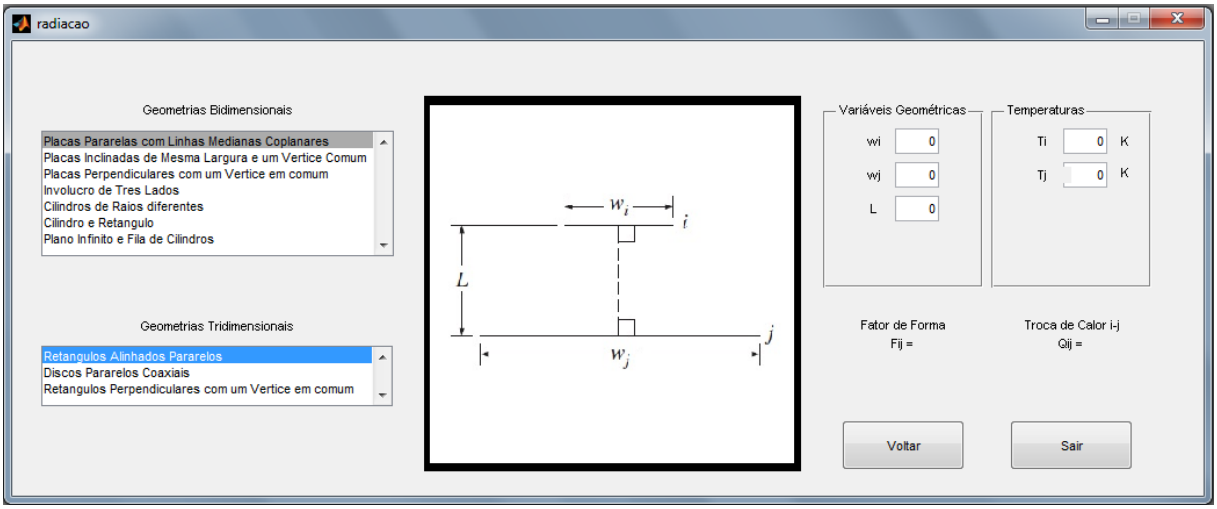


Fig. 3.8 – Interface de resolução dos fatores de forma de radiação.

3.3 ESTRUTURA NUMÉRICA

A estrutura numérica da interface segue métodos diferentes, dependendo do problema abordado. Para a condução transitente dos sólidos finitos foi feita uma dividido em três programas similares, diferindo-se apenas nas equações diferenciais usadas para definir o problema e nos nomes usados para identificar suas funções, mas seguem a mesma lógica computacional que será detalhada a seguir, e que pode ser constatada no código fonte da interface constante no anexo I desse relatório.

Para facilitar a consulta ao código fonte, se usará a designação %{ } ao longo do relatório com o número da referência entre as chaves que poderá ser consultado diretamente no código.

Para a estimativa dos autovalores foram usadas as equações que descrevem o campo de temperatura dos sólidos no regime transiente mostradas no capitulo 2.

- Para a parede:

$$\theta = \sum_{n=1}^{\infty} A_n e^{-\lambda_n^2 \tau} cos(\lambda_n \chi) \quad (2.27)$$

$$A_n = \frac{4 sen\lambda_n}{2\lambda_n} + sen(2\lambda_n) \quad (2.30)$$

$$\lambda_n \, tan\lambda_n = Bi \quad (2.26)$$

- Para o cilindro:

$$\theta = \sum_{n=1}^{\infty} A_n e^{-\lambda_n^2 \tau} J_0(\lambda_n r/r_0) \quad (2.38)$$

$$A_n = \frac{2\,J_1(\lambda)}{\lambda\,J_0(\lambda)^2 + J_1(\lambda)^2} \quad (2.39)$$

$$\lambda_n \frac{J_1(\lambda)}{J_0(\lambda)} = Bi \quad (2.40)$$

- Para a esfera:

$$\theta = \sum_{n=1}^{\infty} A_n e^{-\lambda_n^2 \tau} \frac{sen\,(\lambda_n r/r_0)}{\lambda_n r/r_0} \quad (2.44)$$

$$A_n = \frac{4\,(\mathrm{sen}\lambda_n - \lambda_n cos\lambda_n)}{2\lambda_n - sen(2\lambda_n)} \quad (2.45)$$

$$1 - \lambda_n cot\lambda_n = Bi \quad (2.46)$$

As Equações (2.27), (2.38) e (2.44) envolvem somas de séries infinitas sendo que os seus termos são obtidos a partir dos índices dados pelas Eq. (2.30), (2.39) e (2.45), que por sua vez dependem dos autovalores dados pelas Eq. (2.26), (2.40) e (2.46) respectivamente, e dessa forma estes últimos devem ser resolvidos primeiro.

Se somarmos -Bi aos dois lados das Eq. (2.26), (2.40) e (2.46), obtemos equações igualadas a zero, cuja as raízes são iguais aos autovalores das equações originais:

$$\lambda_n\, tan\lambda_n - Bi = 0 \quad (3.1)$$

$$\lambda_n \frac{J_1(\lambda)}{J_0(\lambda)} - Bi = 0 \quad (3.2)$$

$$1 - \lambda_n cot\lambda_n - Bi = 0 \quad (3.3)$$

Esse procedimento é necessário, pois os métodos computacionais utilizados buscam os valores de raízes. No código fonte, essas equações estão designadas em %{1}. Os gráficos das novas equações são mostrados na Fig. (3.9) para Bi = 0.6.

Existem diversos métodos computacionais que podem estimar a raiz de uma equação, tais como Bissecção, Posição Falsa, Ponto Fixo, Newton e Secante. As suas eficiências estão relacionadas ao número de iterações que cada um deles executa para convergir uma estimativa inicial para o valor da raiz aproximado por um erro pré definido, que no código é identificado por erro %{2}.

Não há uma regra geral capaz de predizer qual método é o mais eficiente. Para uma determinada equação um método pode ser o mais eficiente, enquanto para outra, pode não ser tão eficaz. Também deve ser levado em conta o esforço matemático envolvido em cada iteração, já que alguns métodos recorrem a resolver várias vezes a equação identidade para diferentes valores, ou também ao cálculo de derivadas, como faz o método de Newton.

Outro fator fundamental é a estimativa inicial com que o método irá trabalhar. Uma estimativa distante da raiz obriga o método a efetuar mais iterações para a aproximação necessária. Quando não há como garantir a localização da raiz, o próprio procedimento de obtenção de estimativas também pode requerer processamento do sistema.

A tabela 3.1 mostra a quantidade de vezes que a equação identidade do problema foi resolvida por cada combinação de método de aproximação e procedimento de obtenção de raiz, com os mesmos parâmetros de Bi=0.6 e Fo=0.22, com uma série de 6 termos e uma aproximação de raiz com critério de parada para o erro não menor que 10^{-5}.

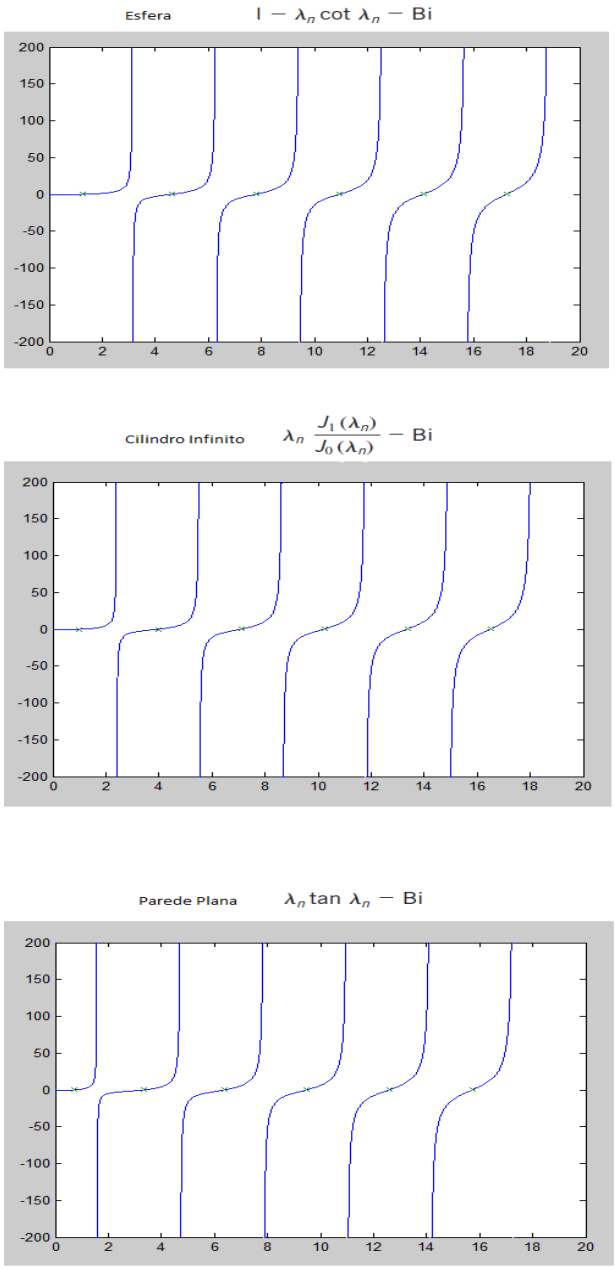


Figura 3.9 – Gráfico das funções dos autovalores para a esfera, o cilindro e a parede.

Tabela 3.1 – Quantidade de iterações de vários métodos de estimativas de raízes.

Método	Obtenção das Estimativas Iniciais	Esfera		Parede Plana		Cilindro Infinito	
		Quantidade de resoluções da equação identidade	Erro	Quantidade de resoluções da equação identidade	Erro	Quantidade de resoluções da equação identidade	Erro
Falsa Posição	varredura em derivada incremento de 0.01 até que |f'(x)|>1000	5154	2.7057*10^-4	5193	1.9324*10^-4	-	-
	varredura em derivada incremento de 0.1 até que |f'(x)|>1000	8578	2.6832*10^-4	7130	1.9568*10^-4	-	-
	varredura em derivada incremento de 0.01 até que |f'(x)|>100	2165	2.6126*10^-4	2058	1.7204*10^-4	5097	2.3315*10^-4
	varredura em derivada incremento de 0.1 até que |f'(x)|>100	1466	2.6124*10^-4	1713	1.9015*10^-4	5174	2.3322*10^-4
	varredura em sinal incremento de 0.1 até que f(x0)*f(x1)<0	416	1.9508*10^-5	406	4.7143*10^-5	424	1.7431*10^-4
	varredura em sinal incremento de 0.01 até que f(x0)*f(x1)<0	3484	1.5197*10^-6	3178	6.3896*10^-6	3361	4.5469*10^-7
	intervalo fixo com distância pi/4 do centro	345	2.3342*10^-4	365	8.4312*10^-5	1635	2.3387*10^-4
	intervalo fixo com distância pi/10 do centro	155	1.2266*10^-4	1055	1.7131*10^-4	170	1.9744*10^-4
Secante	varredura em derivada incremento de 0.01 até que |f'(x)|>1000	3460	2.7057*10^-4	3441	1.9324*10^-4	-	-
	varredura em derivada incremento de 0.1 até que |f'(x)|>1000	5188	2.6832*10^-4	4316	1.9568*10^-4	-	-
	varredura em derivada incremento de 0.01 até que |f'(x)|>100	1679	2.6126*10^-4	1572	1.7204*10^-4	5070	2.3315*10^-4
	varredura em derivada incremento de 0.1 até que |f'(x)|>100	922	2.6124*10^-4	879	1.9015*10^-4	4252	2.3322*10^-4
	varredura em sinal incremento de 0.1 até que f(x0)*f(x1)<0	388	1.9508*10^-5	370	4.7143*10^-5	410	1.7431*10^-4
	varredura em sinal incremento de 0.01 até que f(x0)*f(x1)<0	3464	1.5197*10^-6	3075	6.3896*10^-6	3260	4.5469*10^-7
	intervalo fixo com distância pi/4 do centro	207	2.3342*10^-4	561	8.4312*10^-5	1038	2.3387*10^-4
	intervalo fixo com distância pi/10 do centro	93	1.2266*10^-4	174	1.7131*10^-4	136	1.9744*10^-4
Newton-Raphson	derivada	45	4.6405*10^-5	69	5.7642*10^-9	-	-
	derivada aproximada	-	-	-	-	74	0.6337

Os métodos de aproximação são usuais na prática de programação, e o detalhamento de cada um deles não será abordado por fugir do escopo desse trabalho. Suas formulações podem ser encontradas na literatura, como em Ruggiero & Lopes (1996), enquanto que os procedimentos de obtenção de estimativas foram propostos pelos autores desse trabalho e são baseados na forma típica das equações identidades mostradas na Fig. (3.9).

Pela figura percebe-se que as funções são periódicas e não-contínuas. Cada período possui um comprimento adimensional de aproximadamente de π, e é composto de uma parte negativa $f(x) < 0$, da raiz $f(x) = 0$, de uma parte positiva $f(x) > 0$ e da descontinuidade $\lim f(x)_{x\to\pi} = \infty$. A curva inicia-se com uma inclinação acentuada, $df(x)/dx \gg 0$, depois atenua-se, $df(x) \geq 0$, e em seguida volta a ser íngreme $df(x)/dx \gg 0$.

O procedimento de varredura em derivada percorre o domínio da função com incrementos definidos, analisando a sua inclinação para esses pontos, e seleciona o primeiro dos pontos cuja derivada não passa no teste, qual seja ser menor do que um valor estipulado, apontando dois pontos por período, que limitam um intervalo dentro do qual a raiz deve estar contida. Foram utilizados incrementos de 0.1 e 0.01 e testados para valores de 1000 e 100.

Seria de se esperar que quanto menor o incremento, menos eficiente o processo, já que a cada incremento a função é resolvida para fazer o teste. No entanto, o intervalo gerado é menor, e esse atraso pode ser compensado na resolução do método, como aconteceu quando o valor de teste foi 1000, em que essa proporção não foi notada.

No problema do cilindro infinito a inclinação da função identidade não atingiu derivada superior a 1000 para esses parâmetros.

O procedimento de varredura em sinal percorre o domínio da função com dois pontos x_0 e x_1, incrementando-os igualmente. Quando o produto entre eles for menor do que zero ($x_0 * x_1 < 0$) necessariamente um ponto terá sinal positivo o outro negativo, limitando um intervalo em que deverá estar contida ou uma raiz, ou uma descontinuidade. Os intervalos são enumerados, e somente os ímpares são considerados, pois são os que contêm a raiz. Foram utilizados incrementos de 0.1 e 0.01.

O procedimento do intervalo fixo estima o intervalo a partir da raiz encontrada no período imediatamente anterior. Soma-se π a essa raiz e em seguida delimita-se um intervalo com centro no ponto encontrado. Para a primeira raiz delimita-se um intervalo centrado em π/2. Foram utilizadas distâncias de π/4 e π/10 em relação ao centro.

O método de Newton não utiliza intervalos para estimar a raiz, mas sim um único ponto. Os pontos foram estimados a partir da raiz do período imediatamente anterior, somando-se π a ela. Para o primeiro período a raiz é encontrada a partir da estimativa de π/2. Esse método utiliza-se da derivada da função que deseja-se encontrar a raiz. No caso do problema envolvendo o cilindro infinito, a função identidade requer cálculos elaborados para calcular a derivada, pois são compostas de funções de Bessel de primeira e segunda ordens. Foi proposta uma maneira alternativa de efetuar esse cálculo, que parte do conceito de aproximar a derivada por $df(x)/dx = (f(x) - f(x + dx))/dx$. Foi substituído *dx* por um valor mensurável, no caso 10^{-6}.

Para o cálculo da precisão, os resultados finais de cada combinação foram comparados com a solução apresentada para os respectivos problemas, resolvidos com os mesmos parâmetros, mas com aproximação de 30 termos e com critério de parada para o erro não menor que 10^{-10} para a obtenção de cada raiz.

A escolha do método de aproximação e do procedimento de obtenção de estimativas foi baseada principalmente na eficiência computacional. Assim foram escolhidos o método de Newton-Raphson para a solução do problema da parede plana e da esfera, e o método da Secante com o procedimento do intervalo fixo para o problema do cilindro infinito, que apesar de ser menos eficiente que o método de Newton com a derivada aproximada, é mais preciso %{3}.

Os procedimentos de obtenção de estimativas e aproximação para a raiz devem ser executados em cada período do qual deseja-se extrair a raíz. Quanto mais raízes forem solicitadas, mais preciso

será o resultado, porém o processamento computacional aumentará proporcionalmente, podendo eventualmente comprometer o desempenho do programa.

Dependendo dos parâmetros do sistema (propriedades do material e do meio), varia a quantidade de raízes necessárias para obtenção de um resultado com a precisão definida. Como exemplo, em problemas onde o número de Biot é alto, poucos termos compondo o somatório são suficientes para se alcançar uma dada precisão, devido ao decaimento exponencial $e^{-\lambda_n^2 \tau}$ que aparece na Eq. (2.37). Com outras condições, a quantidade de termos pode aumentar consideravelmente para se obter um resultado igualmente preciso.

Esse comportamento pode ser verificado pela Fig. (3.10) onde são exibidas soluções do problema para a esfera, o cilindro e a parede resolvidos para diferentes números de Biot e com a aproximação de diferentes números de termos na série, em um tempo próximo ao instante inicial. A figura deveria apresentar-se homogenia, visto que a temperatura inicial é uniforme para os corpos, mas aparecem faixas mais claras, indicando áreas que estariam a uma temperatura menor. Para um número de Bi fixo, essas imprecisões diminuem com o aumento de termos, e para um número de termos fixo, as imprecisões diminuem com a diminuição de Bi. Os resultados sugerem que as soluções apresentadas pela interface não apresentariam imprecisões se utilizado um número grande de raízes. Assim não fora feito, pois seria exigido um esforço computacional muito grande. Esse parâmetro é definido pelo programador no código identificado pela variável n %{4} e na seção seguinte serão mostrados os critérios para sua definição e qual o erro associado.

De qualquer forma, as *n* raízes usadas $\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_n$ preenchem os vetores identificados como $x_p, x_c \; e \; x_e$ respectivamente para os códigos fontes dos problemas da parede, do cilindro e da esfera %{5}. Cada posição do vetor é um autovalor das Eq. (2.30), (2.39), (2.46). Baseando-se nessas equações, e com os vetores recém criados, foram gerados os novos vetores c_p, c_c e c_e, em que cada posição *n* representa os termos A_n das equações %{6}. No passo seguinte todas as posições dos novos vetores são utilizadas nos somatórios de acordo com as Eq. (2.27), (2.38) e (2.45), e se obtém a temperatura final adimensionalizada %{7}. Em seguida o resultado é dimensionalizado utilizando-se da temperatura inicial do sólido e da temperatura no infinito %{8}.

A Tabela 3.2 apresenta, para os 6 primeiros termos das séries, os valores das respectivas raízes e dos termos A_n, que no código fonte correspondem respectivamente aos vetores ($x_p, x_c \; e \; x_e$) e (c_p, c_c e c_e), e ainda das contribuições adimensionais θ_n que cada termo adiciona a temperatura. Foram utilizados como parâmetros *Bi* = 0.6, *Fo* = 0.22 e χ = 1. É possível observar que para esses parâmetros, a contribuição do terceiro termo em diante é desprezível.

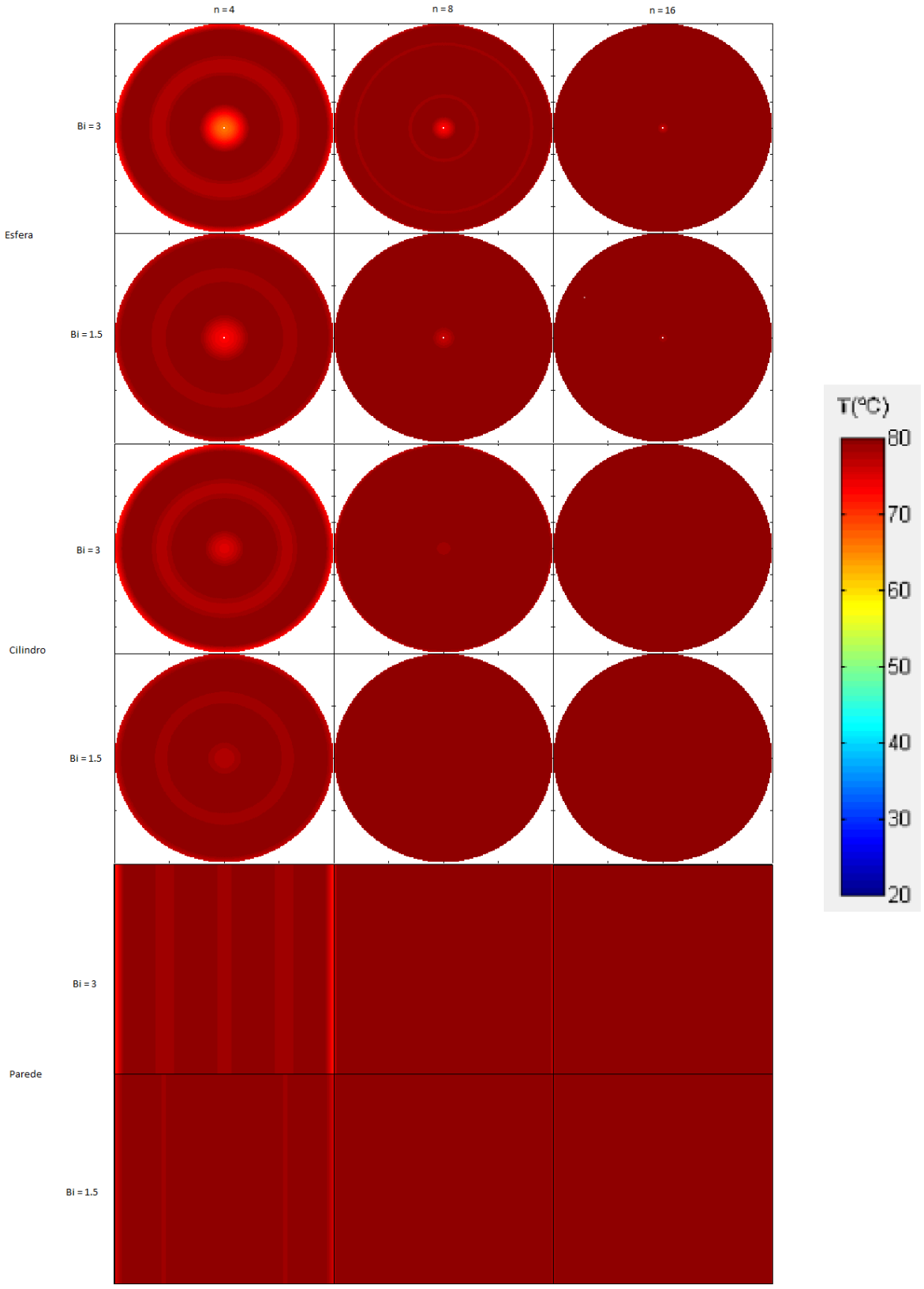


Figura 3.10 – Evolução dos perfis de temperatura para esfera e cilindro de raio $r_0 = 0.04$ m e parede de espessura $l = 0.04$ m com temperatura inicial constante $T_i = 80^0C$ na superfície e temperatura externa $T_\infty = 20^0C$.

A abordagem como foi feita até então, utilizou-se do número de Biot, número de Fourrier e o comprimento adimensional. Deve-se transformar esses parâmetros para que fiquem em função daqueles que são solicitados na interface, como foi feito em %{9}.

Dessa forma concluiu-se o desenvolvimento da função que retorna a temperatura final quando inseridos os parâmetros condutividade térmica (k), calor específico (c_p), comprimento característico (R ou L), massa específica (ρ), temperatura inicial (T_i), coeficiente de convecção (h), temperatura do meio externo (T_∞), localização do ponto de interesse (r) e tempo (t). Ou seja, $f(k, c_p, R, \rho, T_i, T_\infty, h, r, t) = T$, onde a função f foi denominado no código como condt_esfera_T, condt_parede_T e condt_cilindro_T, para os respectivos problemas. Essas funções serão chamadas para atualizar o campo de resultados e também resolvidas ponto a ponto no domínio tempo para plotar o gráfico de temperatura versus tempo, e no domínio comprimento r (ou l no caso da parede plana) para desenhar a imagem bidimensional do sólido com o perfil de temperatura ilustrado em cores, a cada vez que o usuário interagir com qualquer um dos campos que solicitam os parâmetros do problema, ou com a barra de rolagem. Para que essa função não retorne valores indefinidos no conjunto dos números racionais, e assim o programa não entre em processo de *loop* infinito, ela só é calculada quando o usuário já introduziu pelo menos os valores de k, c_p, R, ρ e h, que compõem o vetor v %{10}.

No caso do campo tipo de solução estar selecionado 'solução para a Temperatura', que retorna o tempo necessário para o sólido atingir a temperatura definida pelo usuário, será resolvida essa mesma função criada, mas de maneira recursiva, tal como a função identidade fora resolvida para se chegar a raíz, dessa vez utilizando-se do método da secante. Esse processo compõe uma nova função $f(k, c_p, R, \rho, T_i, T_\infty, h, r, t) = t$, a qual foi denominada condt_esfera_t, condt_parede_t e condt_cilindro_t, para os respectivos problemas %{11}. Mesmo quando se exige a obtenção poucas raízes, esse tipo de solução é mais demorado, pois a função de obtenção da temperatura deverá ser resolvida várias vezes, e a cada uma, a função identidade será também resolvida recursivamente. A animação portanto somente foi implementada para a solução do tipo ‘solução para o tempo’.

Foi necessário ainda definir o tempo máximo para o qual o problema será resolvido. Para tanto foi utilizado o procedimento de varredura em derivada tal como descrito anteriormente nesta mesma seção. Dessa vez é buscado o valor próximo a zero para a derivada da função da temperatura $f(k, c_p, R, \rho, T_i, T_\infty, h, r, t)/dt = T$, aqui chamadas de d_condt_esfera_T, d_condt_parede_T e d_condt_cilindro_T para os respectivos problemas, que é obtida a partir da aproximação de derivada em que dx é substituído por um valor mensurável em, $df(x)/dx = (f(x) - f(x + dx))/dx$, nesse caso 10^{-6} %{12}. O incremento inicial para os testes é de 5 unidades %{13}. Quando houver a primeira falha, o incremento passa a ser de uma unidade no sentido oposto %{14}, e quando houver falha novamente o incremento muda novamente de sentido e passa a ser de 0.5 unidade %{15}. Dependendo dos parâmetros do sistema, o tempo máximo pode ser de vários segundos, e nesse caso, um incremento pequeno comprometeria o desempenho do programa, por isso um incremento inicial de

cinco unidades. Em outras condições o tempo máximo pode ser de poucos segundos e um resultado impreciso ocasionaria um esforço desnecessário do programa já que o domínio do tempo é subdividido no mesmo número de partes independentemente dos parâmetros do sistema, e ainda corresponderia a plotagem de um gráfico bastante ocupado pela parte *df/dx~0*, que não é relevante para a análise. A função $f\ (k, c_p,\ R,\ T_i, T_\infty, h) = t_{MAX}$ que retorna o tempo máximo foi denominada de gettmax, e é chamada para a plotagem do gráfico e para a execução da animação %{16}.

Tabela 3.2 – Valores dos 6 primeiros termos da série da temperatura adimensional para os 3 sólidos.

Parede Plana			
n	x_n	A_n	θ_n
1	0,7051	1,0814	0,7382
2	3,3204	-0,1017	0,0089
3	6,3770	0,0290	3.7486*10^-6
4	9,4879	-0,0132	3.2966*10^-11
5	12,6139	0,0075	4.6845*10^18
6	15,7460	-0,0048	9.7906*10^-27
Esfera			
n	x_n	A_n	θ_n
1	1,2644	1,1713	0,6213
2	4,6261	-0,2633	5.1121*10^-4
3	7,8028	0,1546	3.0100*10^-8
4	10,9591	-0,1098	3.341*10^-14
5	14,1088	0,0852	5.7431*10^-22
6	17,2556	-0,0696	1.4221*10^-31
Cilindro Infinito			
n	x_n	A_n	θ_n
1	1,0184	1,1345	0,6836
2	3,9841	-0,1857	0,0022
3	7,1004	0,0790	3.5970*10^-7
4	10,2322	-0,0458	1.1300*10^-12
5	13,3686	0,0307	5.5988*10^-20
6	16,5070	-0,0224	4.0332*10^-29

Para desenhar a figura com o perfil de temperatura bidimensional é gerada uma matriz T %{17}, em que a função de temperatura é resolvida para cada posição da matriz, com as linhas e as colunas correspondendo às coordenadas cartesianas de cada ponto %{18}. No caso da esfera e do cilindro, é necessário transformar as coordenadas cartesianas bidimensionais na coordenada unidimensional esférica ou cilíndrica, através da fórmula $r = (x^2 + y^2)^{1/2}$ %{19}. Rejeita-se os pontos em que $r > R$ %{20}. No caso da parede plana, não é levada em consideração a ordenada cartesiana *y*, pois a temperatura é a mesma para todo *y* dado a abscissa *x*, assim $l = x$. A matriz *T* é o resultado de uma função $f\ (k, c_p,\ R,\ T_i, T_\infty, h,\ t) = T$, chamada de pcolor_esfera, pcolor_cilindro e

pcolor_parede para os respectivos problemas %{21}, e chamada sempre que for preciso atualizar o desenho, seja por iteração do usuário com os parâmetros e com a barra de rolagem ou no incremento de tempo na animação.

O desenvolvimento da interface para os problemas de sólidos semi-infinito e fatores de forma de radiação seguiram metodologias semelhantes ao problema da condução em sólidos finitos. Algumas particularidades porém mudaram a análise e a estrutura numérica do programa. Os sólidos semi-infinitos são regidos pelas Eq. (2.58) e (2.59).

$$\frac{T-T_s}{T_i-T_s} = \frac{2}{\sqrt{\pi}} \int_0^{\eta} e^{-u^2}\, du \tag{2.58}$$

$$q_s = \frac{\dot{k}(T_s-T_i)}{\sqrt{\pi\alpha t}} \tag{2.59}$$

Para o caso da radiação, a equação a ser resolvida para o cálculo dos fatores de forma é

$$F_{12} = \frac{1}{A_1} \int_{A_2} \int_{A_1} \frac{cos\theta_1\, cos\theta_2}{\pi r^2}\, dA_1 dA_2 \tag{2.60}$$

A Figura (3.11) mostra as configurações geométricas para as quais a Eq. (2.60) foi resolvida.

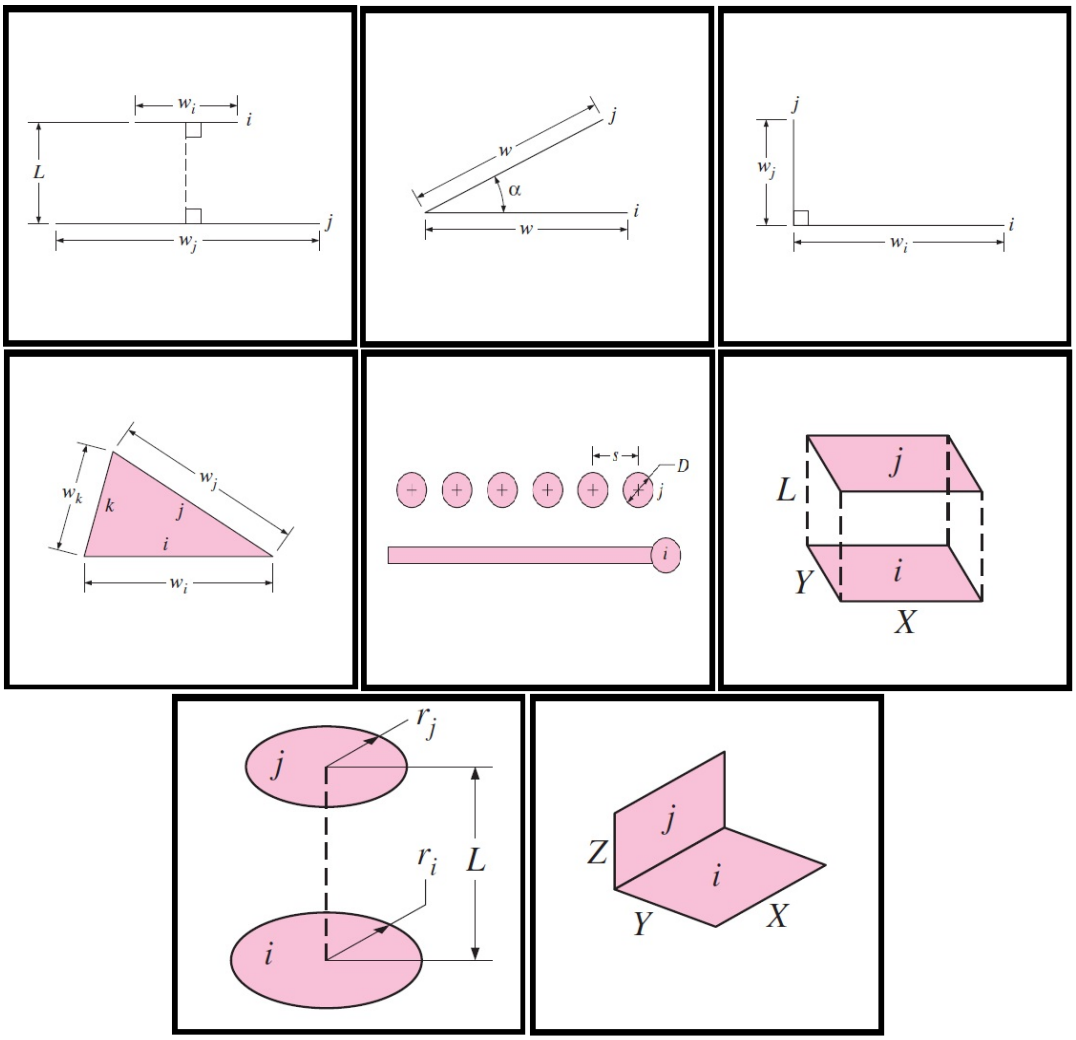


Figura 3.11 – Geometrias para cálculo dos fatores de forma

3. ANÁLISE DE DADOS

A Fig. (3.12) mostra 3 gráficos da temperatura adimensional pelo número de Fourrier para diversos números de *Bi* no centro da esfera. O gráfico superior é a solução de Arpaci (1966), o gráfico do meio foi gerado pelos métodos e funções utilizadas na plataforma, e finalmente o terceiro gráfico é composto pela sobreposição dos dois anteriores transparecidos, e serve para comparar a consistência dos resultados obtidos nesse projeto com os teóricos. Os gráficos de Arpaci são, na verdade, adaptações dos gráficos originais encontrados em Boelter (1965) e Grober (1961), citados nas referências bibliográficas.

As Fig. (3.13) e (3.14) mostram diretamente as soluções sobrepostas, ou seja, os dois gráficos (os de Arpaci e os obtidos pela programa) sobrepostos, para o problema da esfera com raios admensionais r_{ad} de 0, 0.2, 0.4, 0.6, 0.8 e 1. Percebe-se uma semelhança significativa nas retas que correspondem a 1/B1≠0, o que indica que o método utilizado é consistente para esse tipo de solução sob esta condição. Para 1/Bi=0, não puderam ser geradas as retas correspondentes pelo método de Newton, pois o número de Biot é um dos parâmetro de entrada da função, e nesse caso Bi=∞, o que leva a uma indefinição matemática. Análise análoga foi feita para o cilindro e a parede e estão apresentadas nas fig. (3.15), (3.16), (3.17) e (3.18).

Para avaliarmos o comportamento da solução perto do limite em que 1/Bi=0, foram gerados novamente os gráficos da temperatura adimensional pelo número de Fourrier para o problema da esfera com um raio adimensional de 0.2, acrescidos da curva correspondente a 1/Bi=10^{-3}, calculados para os números de termos de 10^1, 10^3 e 10^5, compondo o somatório (fig. 3.19).

É possível perceber que os intervalos 0<*Fo*<0.01 e 0<*Bi*<0.1 delimitam uma faixa de dados em que a convergência dos valores calculados para os valores fornecidos por Arpaci é mais difícil de acontecer, necessitando um número de termos muito grande, na ordem de 10^5, que caso fosse implementado, comprometeria a funcionalidade da aplicação.

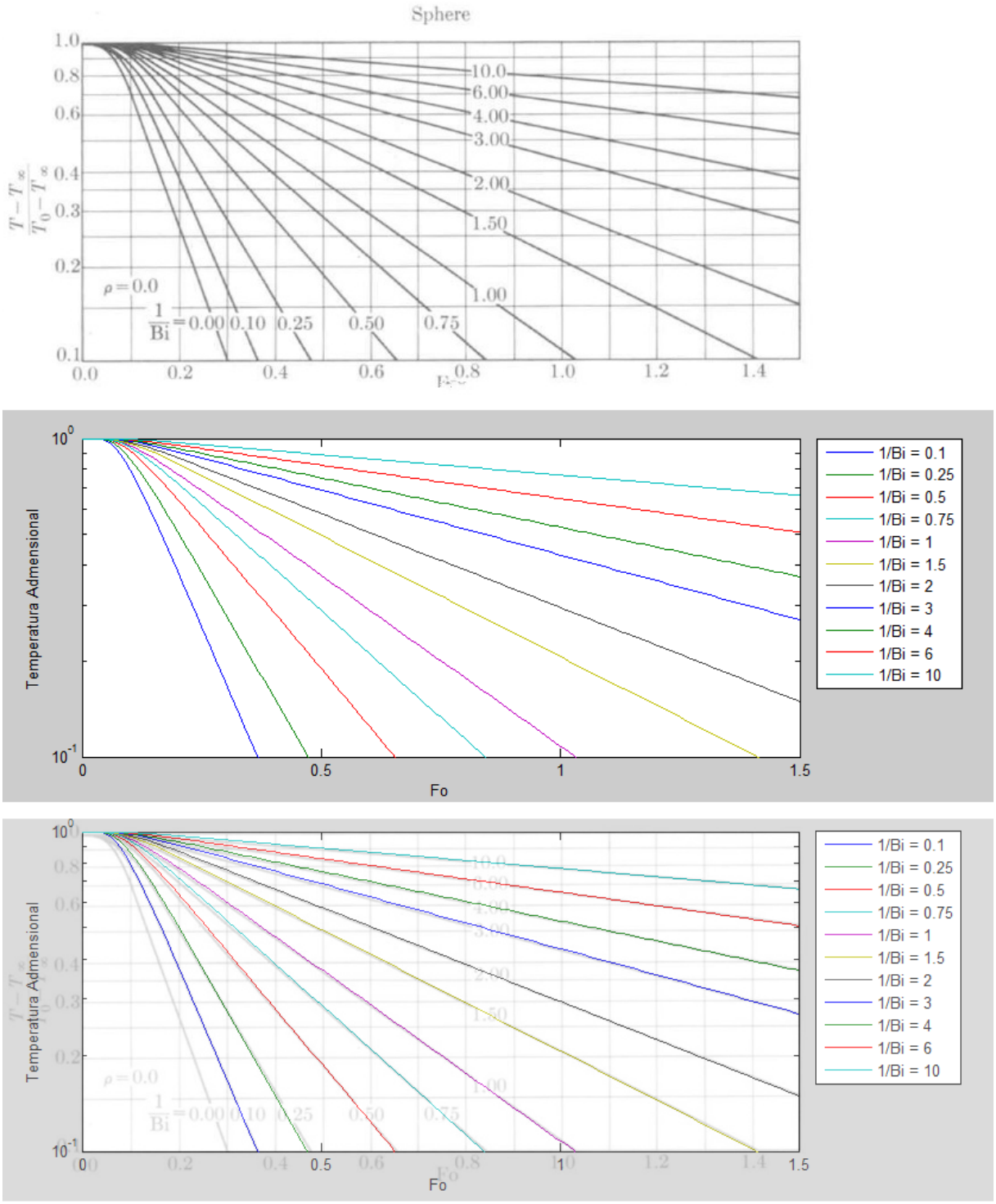


Figura 3.12 – Comparação dos gráficos obtidos da temperatura adimensional pelo numero de Fourier no centro da esfera com os de Arpaci.

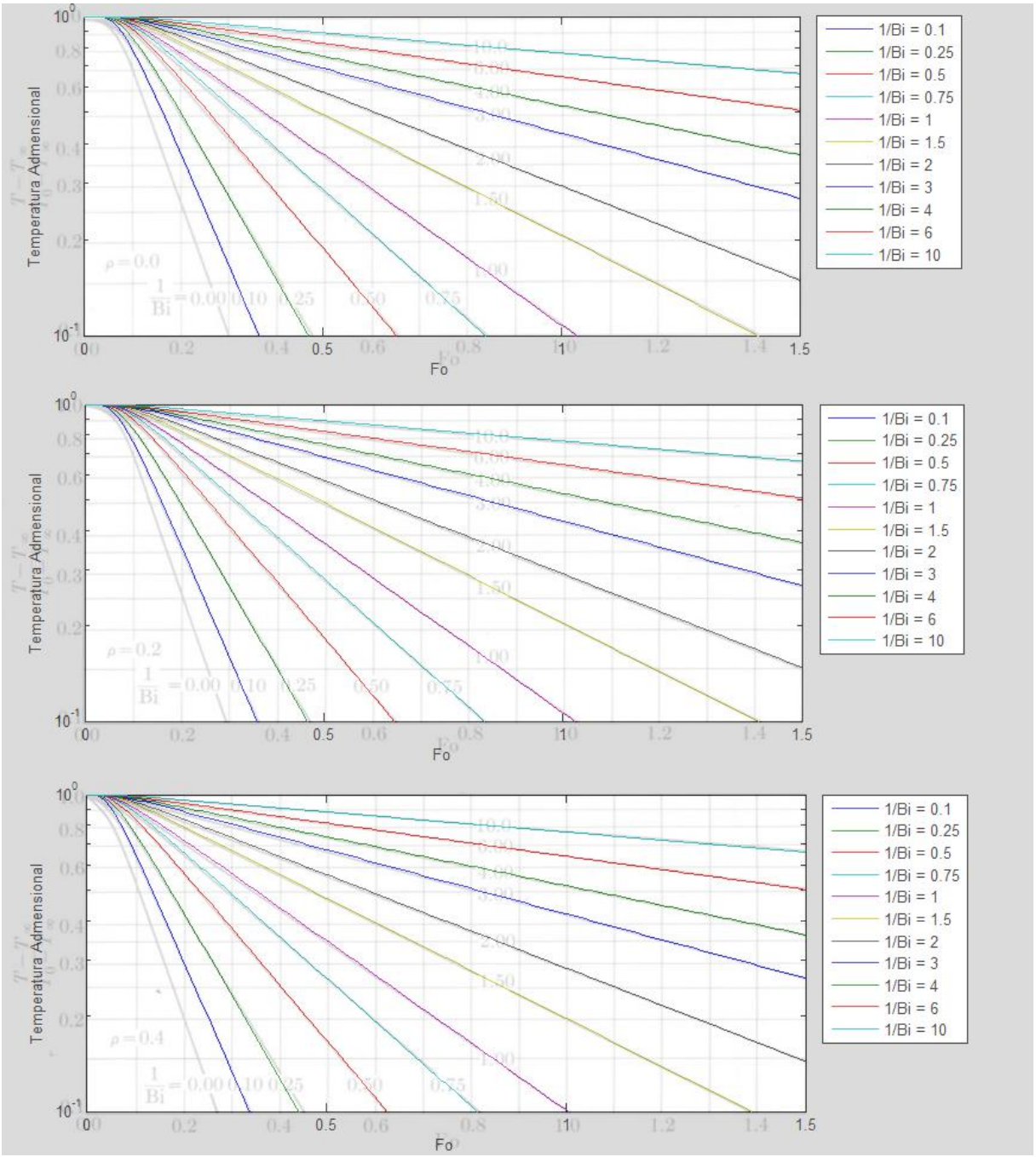


Figura 3.13 – Gráficos da temperatura adimensional pelo numero de Fourier obtidos sobrepostos com os de Arpaci para esferas de raios adminensionais $r_{ad} = 0, 0.2\ e\ 0.4$.

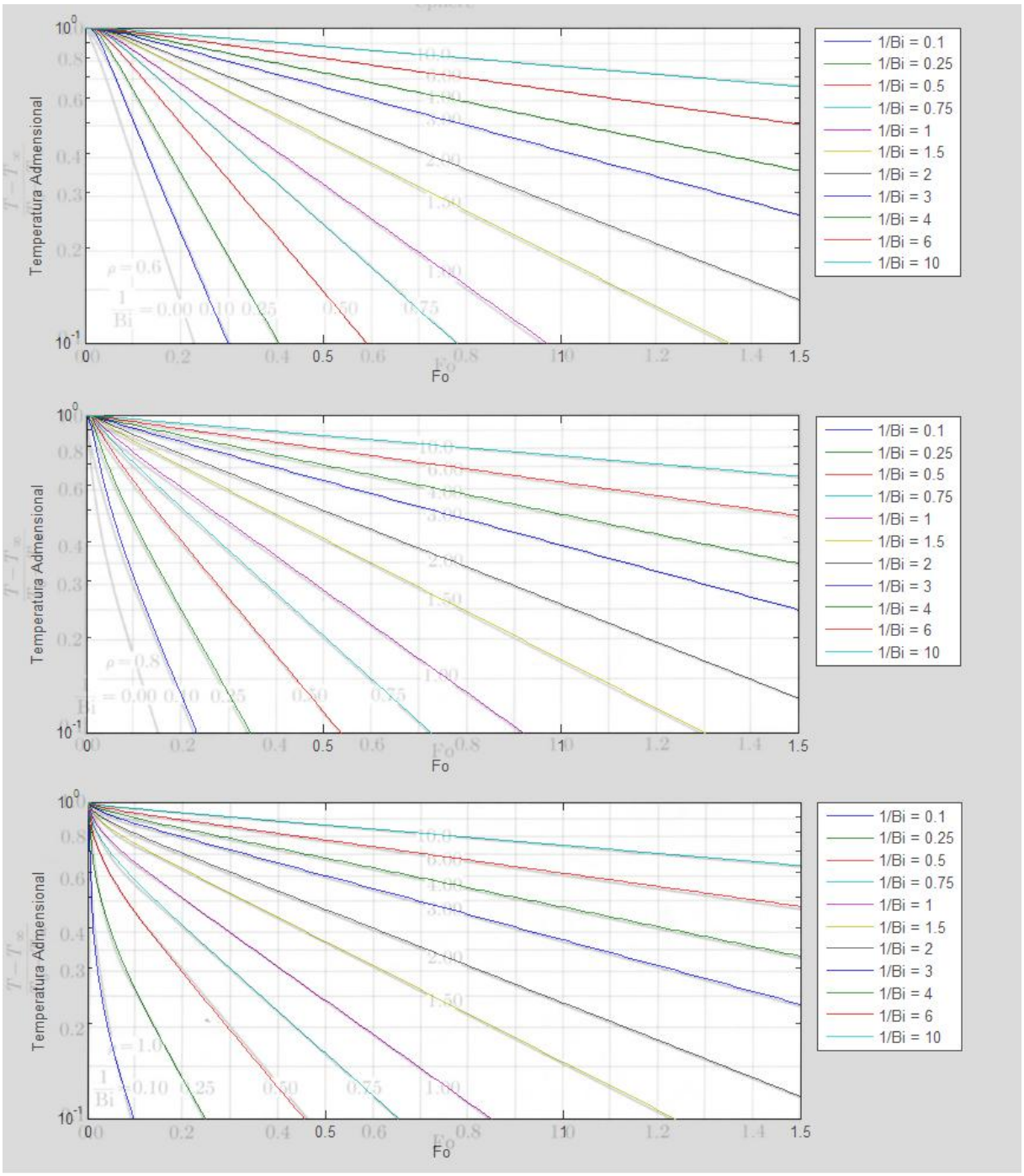


Figura 3.14 – Gráficos da temperatura adimensional pelo numero de Fourier obtidos sobrepostos com os de Arpaci para esferas de raios adminensionais $r_{ad} = 0.6, 0.8\ e\ 1$.

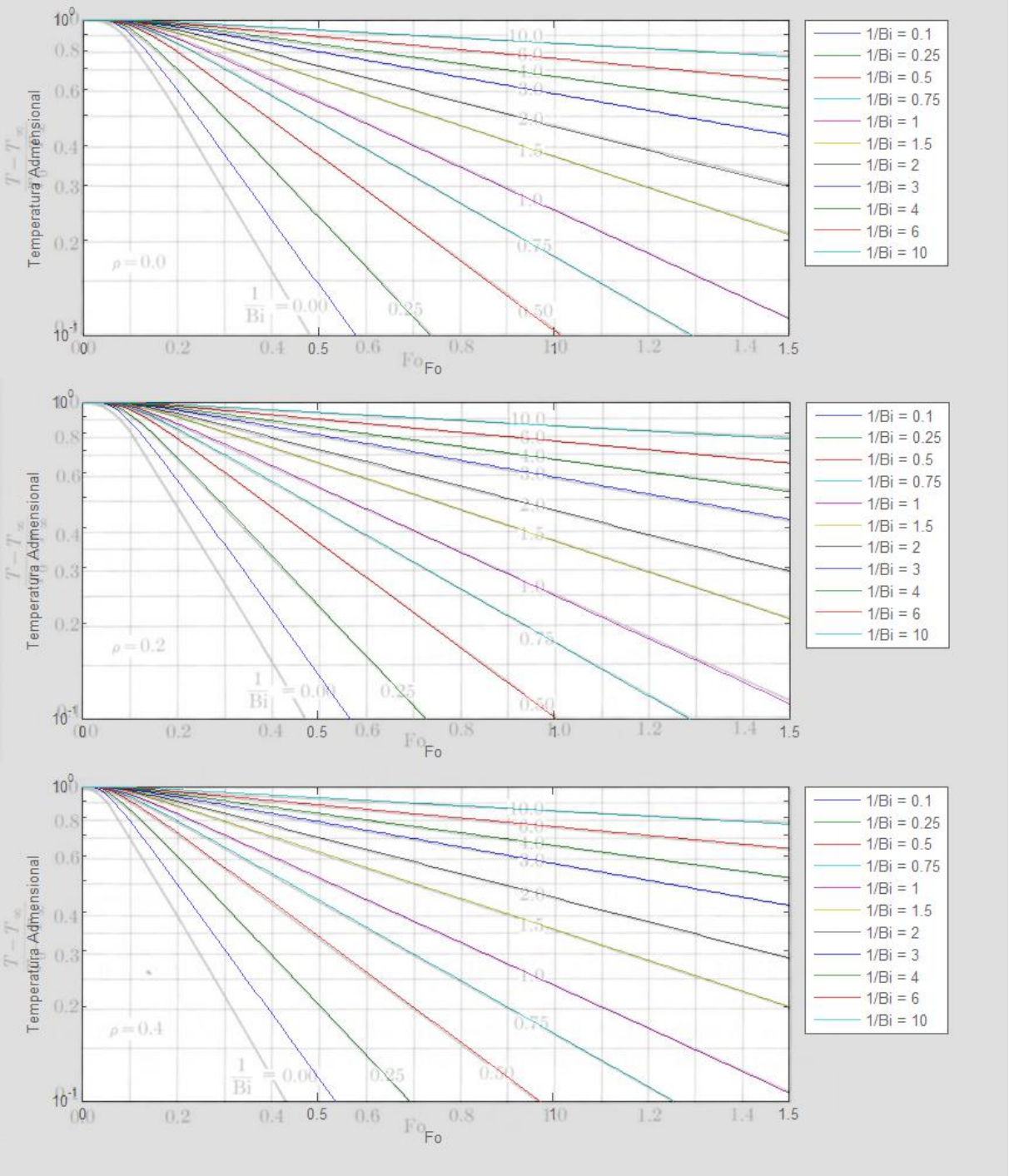


Figura 3.15 – Gráficos da temperatura adimensional pelo numero de Fourier obtidos sobrepostos com os de Arpaci para cilindros de raios adminensionais $r_{ad} = 0, 0.2\ e\ 0.4$.

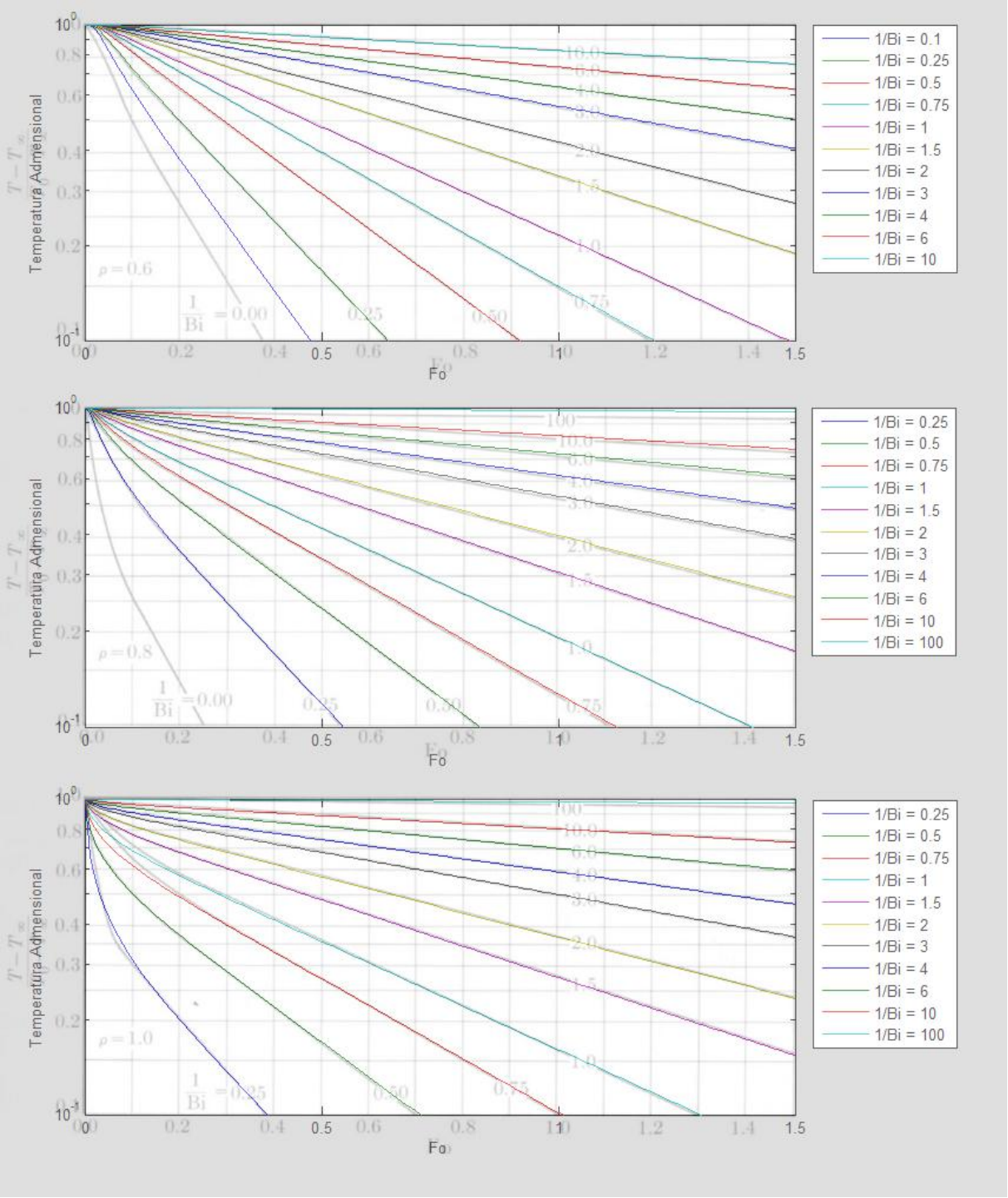


Figura 3.16 – Gráficos da temperatura adimensional pelo numero de Fourier obtidos sobrepostos com os de Arpaci para cilindros de raios adminensionais $r_{ad} = 0.6, 0.8\ e\ 1$.

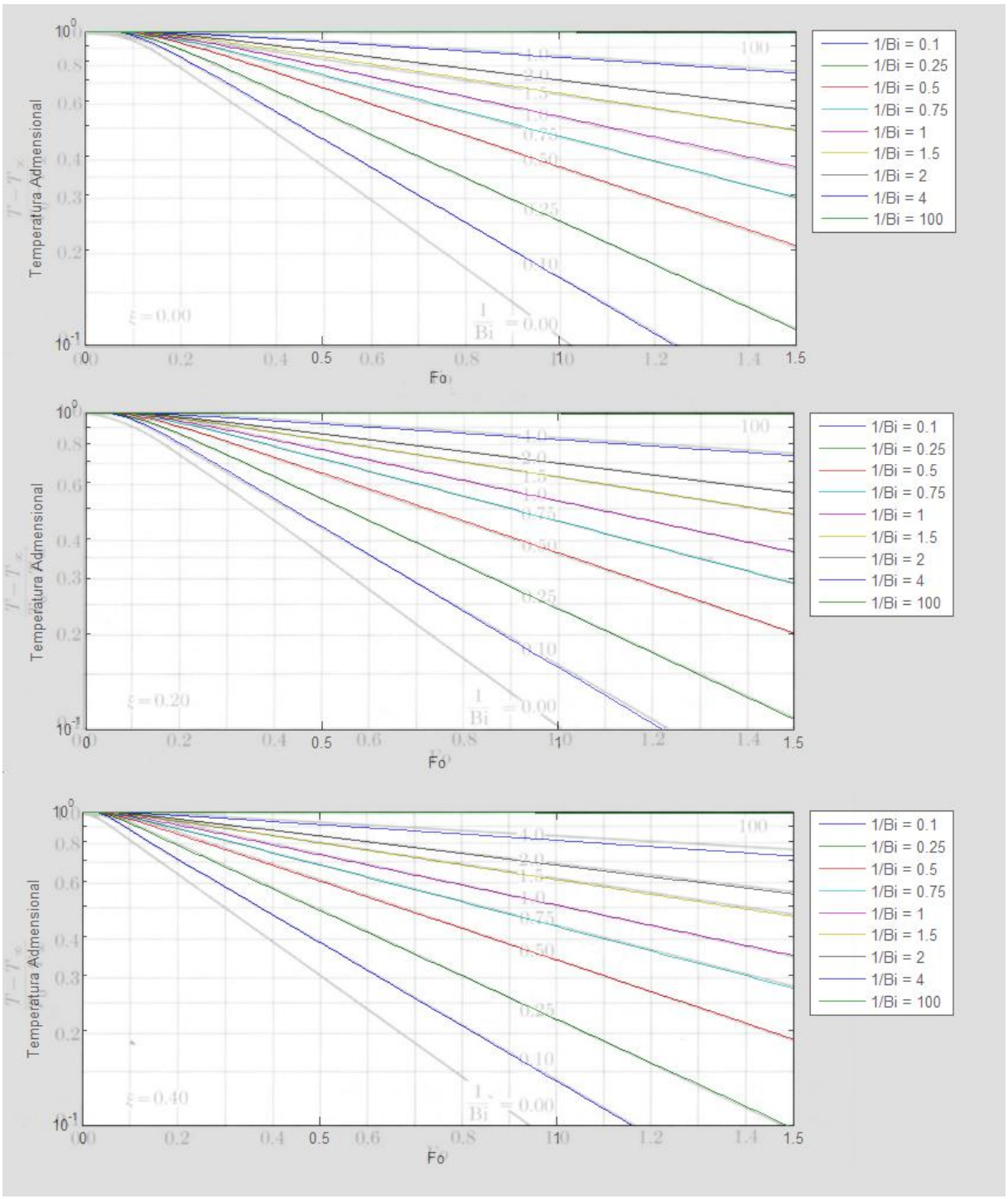


Figura 3.17 – Gráficos da temperatura adimensional pelo numero de Fourier obtidos sobrepostos com os de Arpaci para parede de comprimento adimensional $\chi = 0, 0.2\ e\ 0.4$.

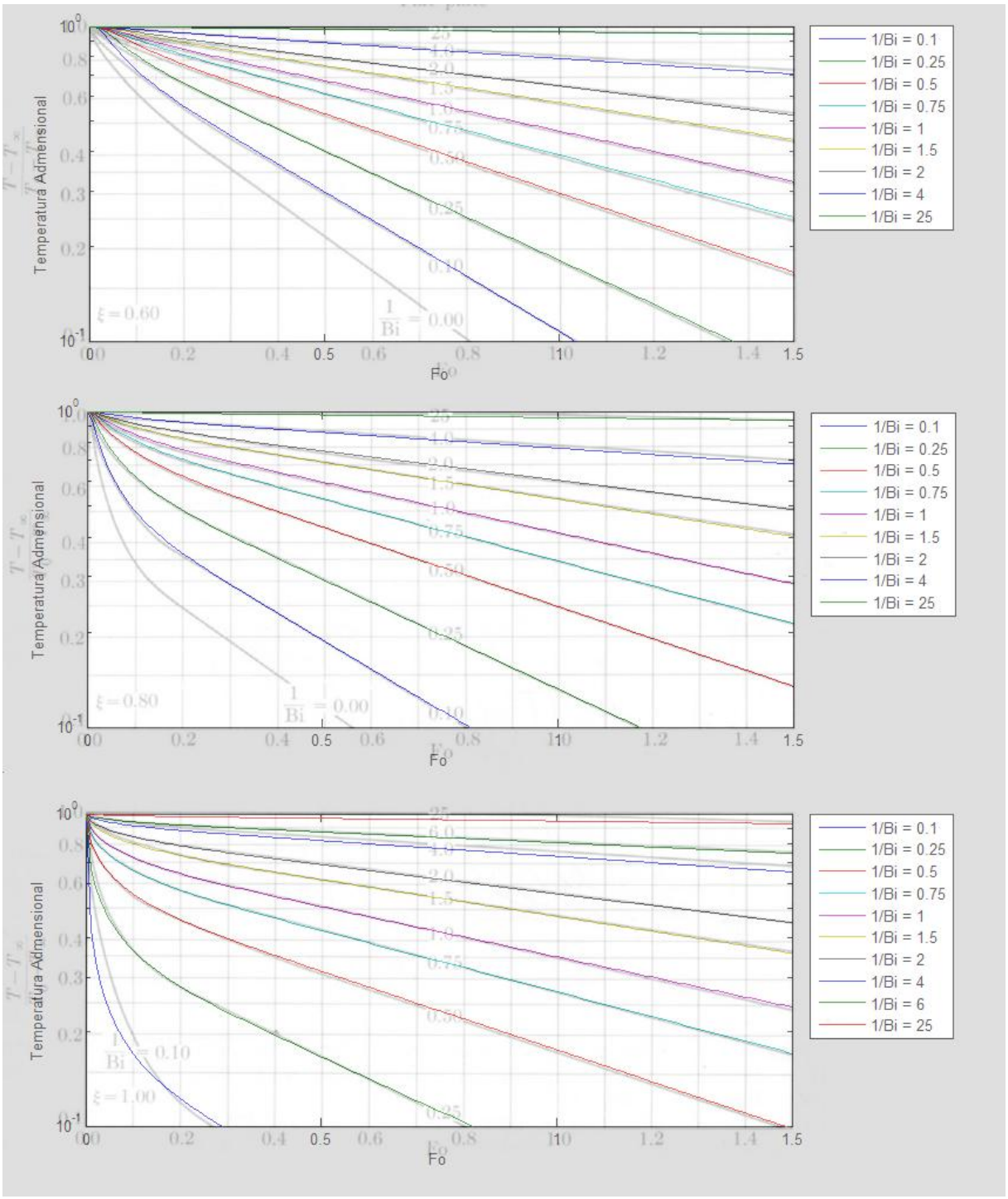


Figura 3.18 - Gráficos da temperatura adimensional pelo numero de Fourier obtidos sobrepostos com os de Arpaci para parede de comprimento adimensional $\chi = 0.6, 0.8$ *e* 1.

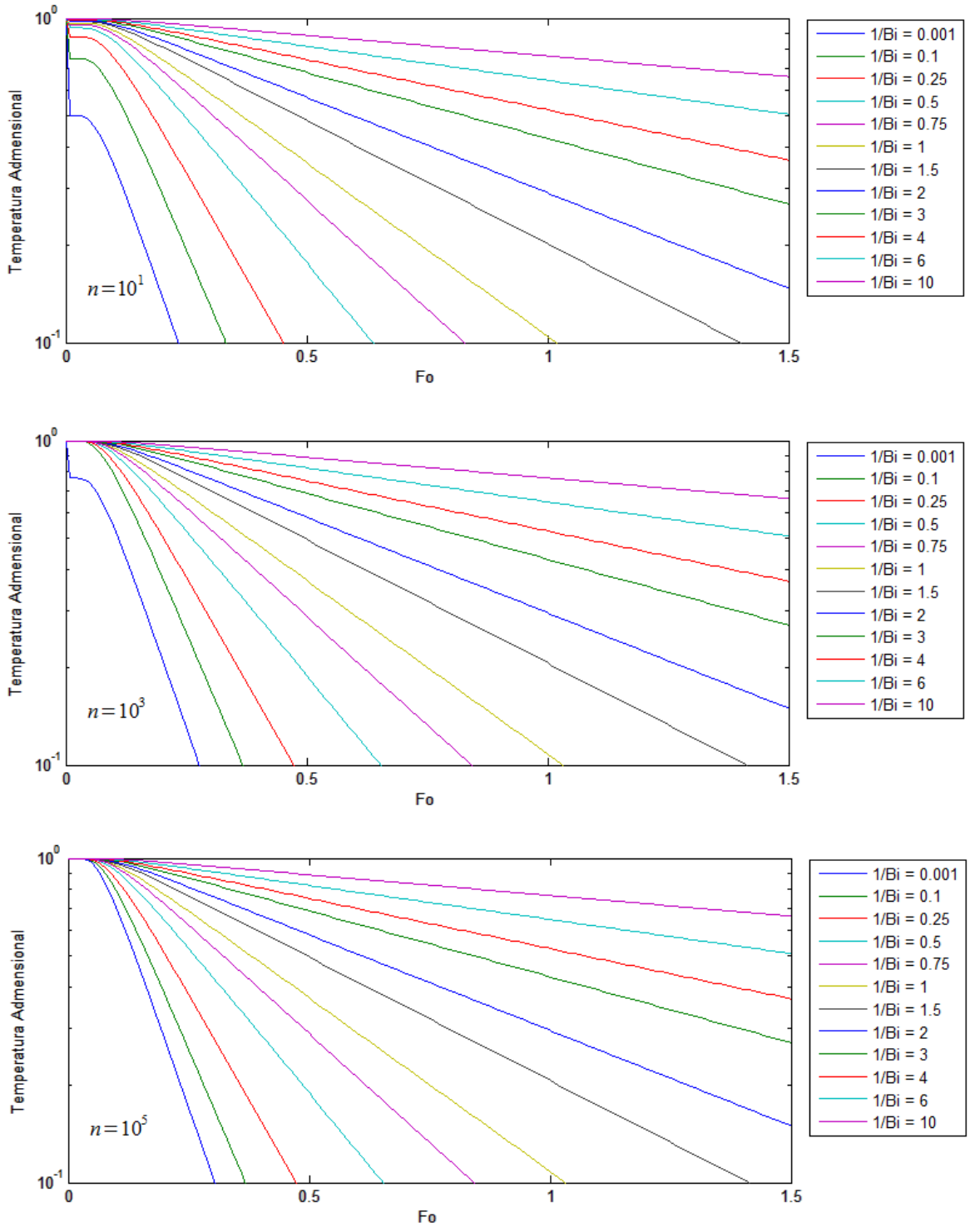


Figura 3.19 – Gráficos da temperatura adimensional pelo numero de Fourier para a esfera de raio adimensional $r_{ad} = 0.2$ para n= $10^1, 10^3 e\ 10^5$.

As Figuras (3.20), (3.21) e (3.22) mostram os erros associados com a obtenção da temperatura admensional da parede plana, da esfera e do cilindro infinito quando os valores se aproximam da referida faixa de dados comparados aos valores obtidos na solução dos respectivos problemaa utilizando 10^5 termos.

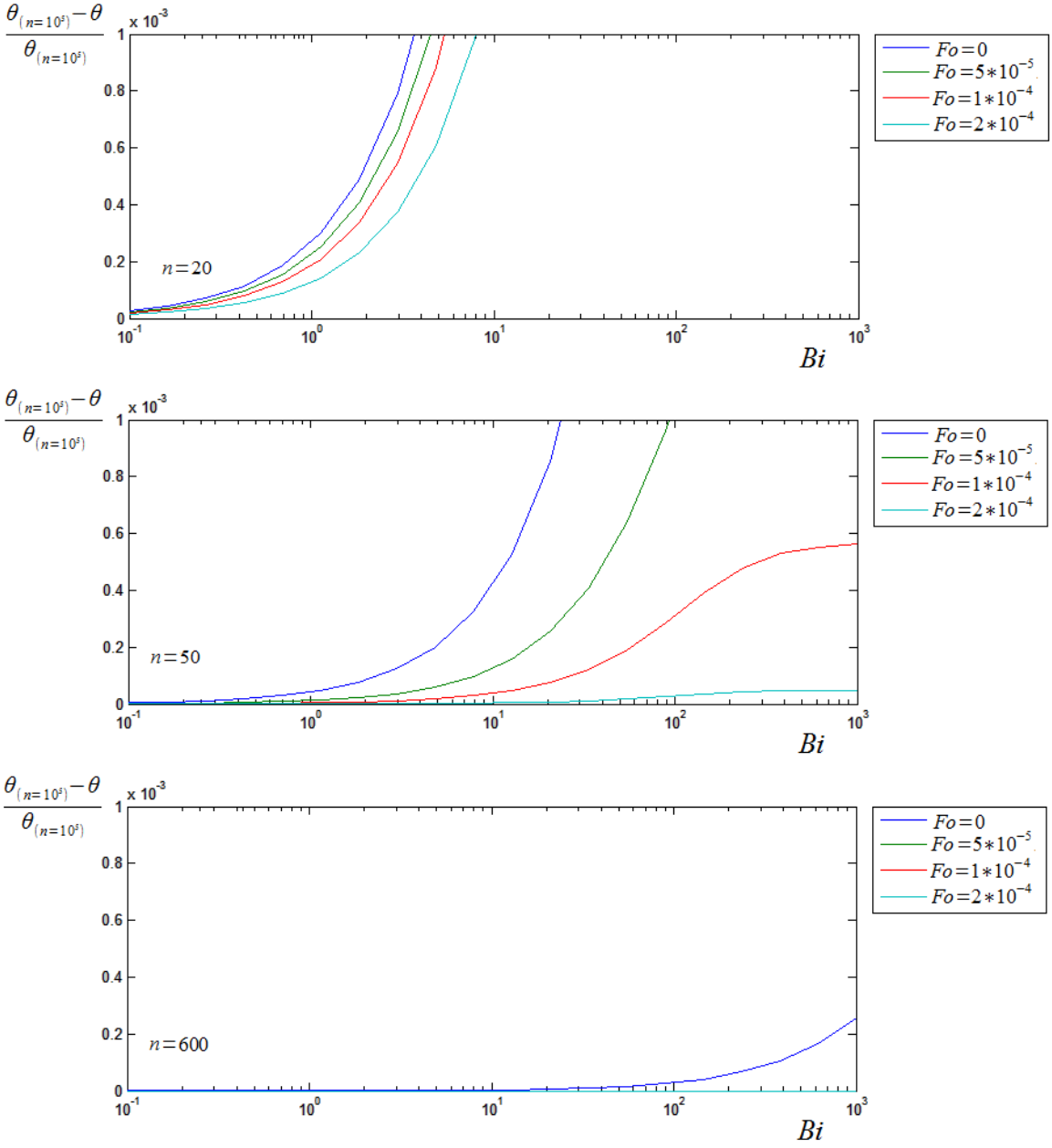


Figura 3.20 – Gráficos do erro associado a obtenção da temperatura adimensional pelo número de Biot da parede para n = 20, 50 e 600.

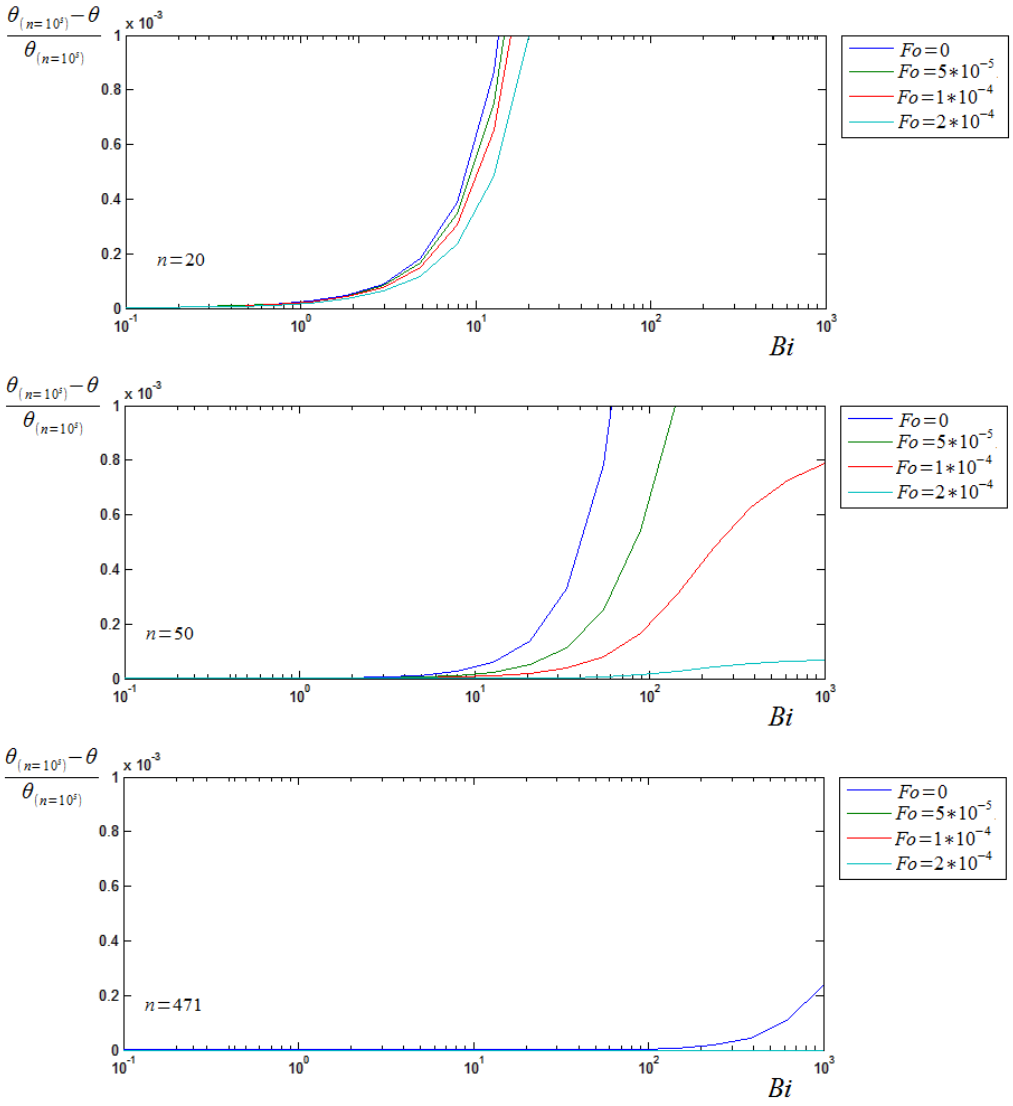


Figura 3.21 – Gráficos do erro associado a obtenção da temperatura adimensional pelo número de Biot da esfera para n = 20, 50 e 471.

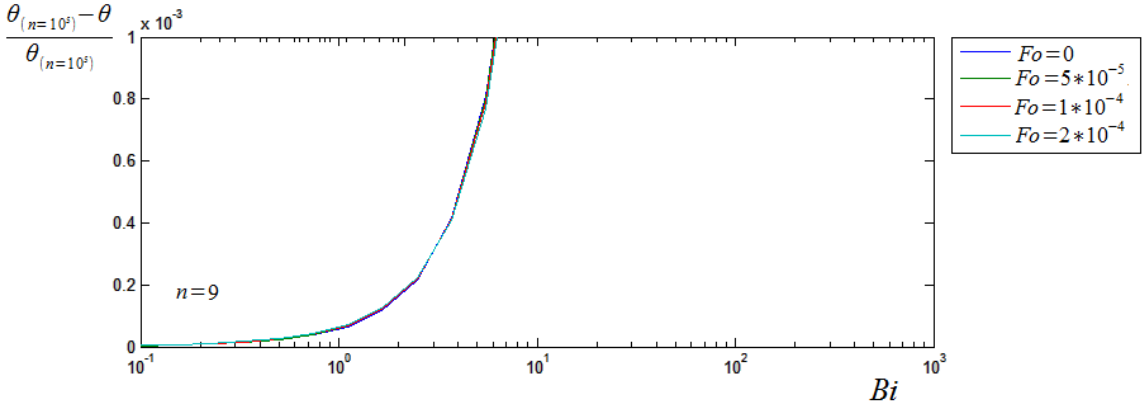


Figura 3.22 – Gráficos do erro associado a obtenção da temperatura adimensional pelo número de Biot do cilindro para n = 9.

Em vista de definir a faixa de parâmetros em que o programa aceitará trabalhar para que forneça apenas resultados confiáveis, primeiramente deve ser avaliado com quantos termos cada problema poderá ser resolvido sem comprometer o desempenho computacional e a funcionalidade para o usuário. Para isso, foram registrados os tempos médios que o programa leva para desenhar o perfil de temperatura do sólido, plotar o gráfico e apresentar os resultados de cinco pontos igualmente espaçados no domínio do tempo em escala logarítimica. A Figura (3.23) mostra o gráfico dos tempos médios pelos números de termos para os problemas da esfera, do cilindro infinito e da parede plana.

Esses tempos médios são foram calculados em um computador com capacidade de processamento de 3.4 Ghz (Intel Core i5), memória RAM de 2 GB e sistema operacional Windows 7. A maioria dos computadores pessoais disponíveis no mercado possuem performace semelhante ou superior, portanto, os resultados serão compatíveis quando comparados ao resultados do uso do programa em um notebook pessoal, por exemplo.

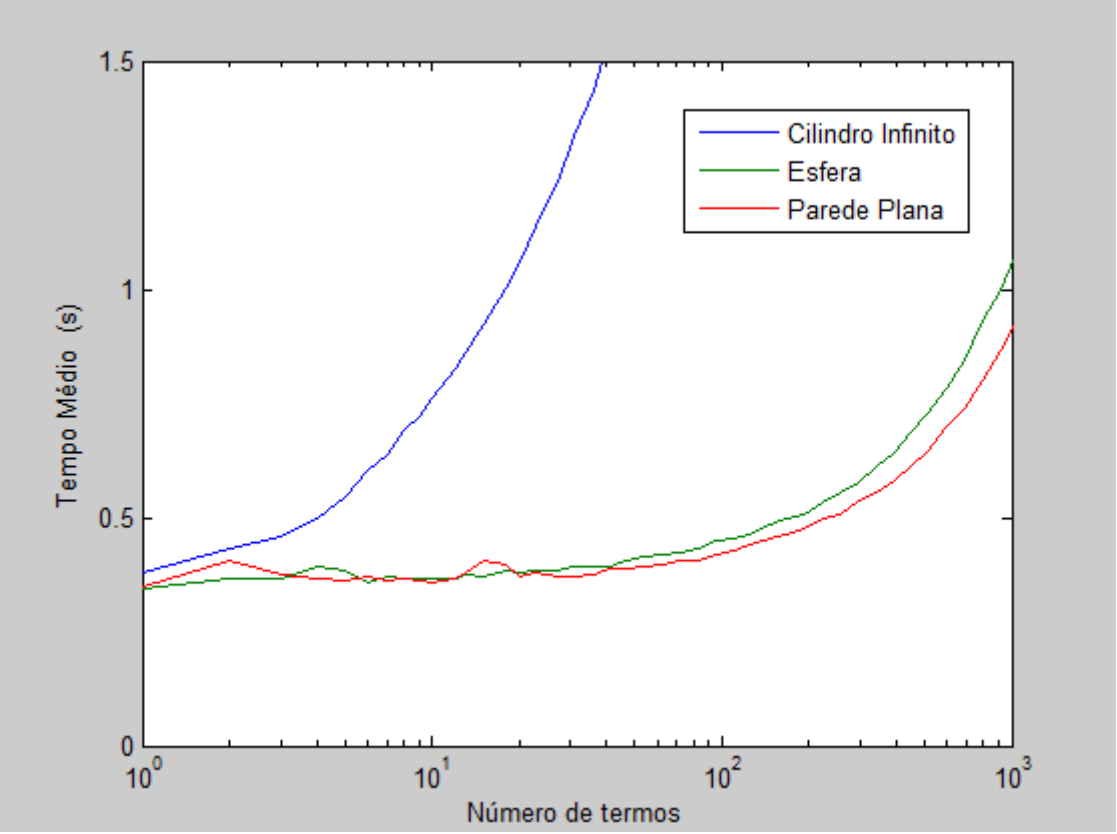


Figura 3.23 – Gráficos dos tempos médios de resolução dos problemas pelo número de termos para os casos da esfera, do cilindro e da parede.

Foi considerado que o tempo de 0,7 segundos corresponde ao esperado para a proposta de funcionalidade da plataforma. O número de termos n que faz com que os problemas sejam resolvidos no tempo médio desejado é 471 para a esfera, 600 para a parede plana e 9 para o cilindro infinito.

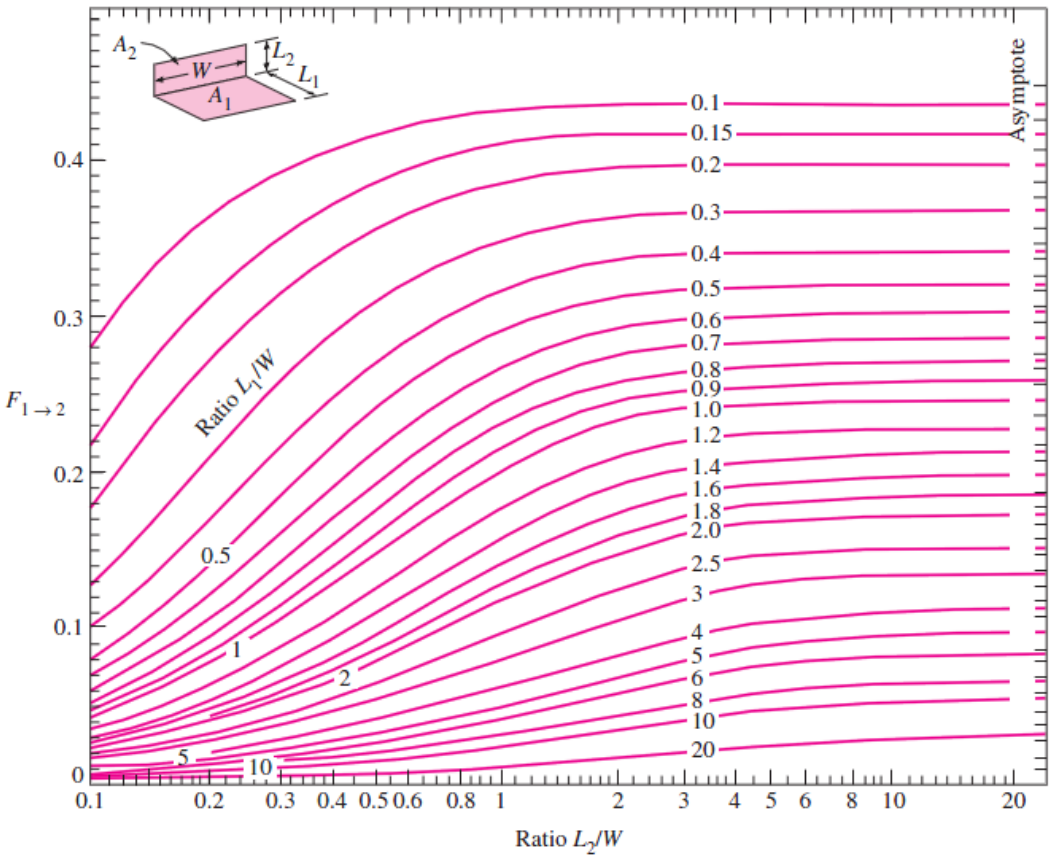
Voltando as Fig. (3.20), (3.21) e (3.22) é imediato notar que o erro é uma função do número de Biot, do número de Furrier e da quantidade de termos n, ou seja, $erro = f(Bi, Fo, n)$. Fixado n, tem-se que $erro = f(Bi, Fo)$, que decresce a uma taxa maior em *Fo* do que em *Bi*. É devido a *Bi* portanto a maior contribuição no erro.

Se limitarmos a faixa aceitável de *Fo*, a faixa aceitável de *Bi* seria ampliada consideravelmente. Por exemplo, se os parâmetros fossem aceitos apenas quando resultassem em $Fo > 2.10^{-4}$ no caso da parede plana, o erro envolvido seria de uma ordem menor do que 10^{-3} mesmo que o número de termos fosse 50, conforme a fig. (3.20). No entanto, o problema não seria resolvido para todo o domínio, inclusive para o instante inicial, já que *Fo* é proporcional ao tempo. Assim, foi decedido que os parâmetros seriam limitados somente a *Bi*, considerando apenas as curvas em que *Fo=0*.

Para que o erro não seja maior do que 0,02%, os parâmetros foram limitados para que resultem em *Bi*<754,1404 para a parede plana, *Bi*<895,5116 para a esfera, e *Bi*<2,3394 para o cilindro infinito.

Para comparar a consistência dos resultados obtidos com a interface e as soluções da literatura para os fatores de forma da radiação, plotou-se o gráfico do fator de forma entre duas placas perpendiculares com um vértice em comum abaixo do gráfico de Cengel (2008), Fig (3.24). Os gráficos possuem as mesmas escalas e valores de curvas, sendo estas omitidas no gráfico inferior para melhor visualização. Para outras configurações de superfícies os resultados obtidos foram igualmente consistentes.

Para uma análise efetiva dos resultados obtidos pela interface foi resolvido um problema de condução com aplicação na área industrial de duas maneiras . Primeiramente foi resolvido com o uso dos gráficos de Heisler e depois com o uso da interface. Em uma instalação de produção, grandes placas de bronze (k=110 W/moC, RO=8530 kg/m3, cp=380 J/kgoC e alfa=33,9x10-6m2/s) de 3 cm de espessura, inicialmente a uma temperatura uniforme de 700oC, que acabaram de sair de um formo, precisam ser resfriadas lentamente até a temperatura no centro atingir a temperatura de 50oC, quando então sofrerá um novo tratamento térmico. Deve-se determinar o tempo em que as placas precisam ficar esfriando ao tempo (26oC e h=80 W/m2oC). Assume-se que a placa pode ser considerada como uma placa infinita em virtude do comprimento ser muito maior que a espessura, há simetria térmica com o plano central da placa e as propriedades do material e o coeficiente de convecção são constantes e logo os gráficos de Heisler podem ser usados na solução. Usando o método gráfico, calcula-se o numero de Biot ($Bi = \frac{hL}{k} = \frac{(80W/m^2.℃)(0.015m)}{(110\,W/m.℃)} = 0.01090$) e a temperatura adimensional no centro da placa ($\theta_0 = \frac{T_0 - T_\infty}{T_i - T_\infty} = \frac{(50℃) - (26℃)}{(700℃) - (26℃)} = 0.03560$) . Com esses valores se procede a análise do gráfico de Heisler. Pelo gráfico encontrou-se um valor de aproximadamente 300 para o número de Fourrier. Assim o tempo encontrado vale 1191,1505 segundos ($t = \frac{\tau L^2}{\alpha} = \frac{(300)(0.015m)^2}{(33.9*10^6 m^2/s)} = 1191{,}1504s$).



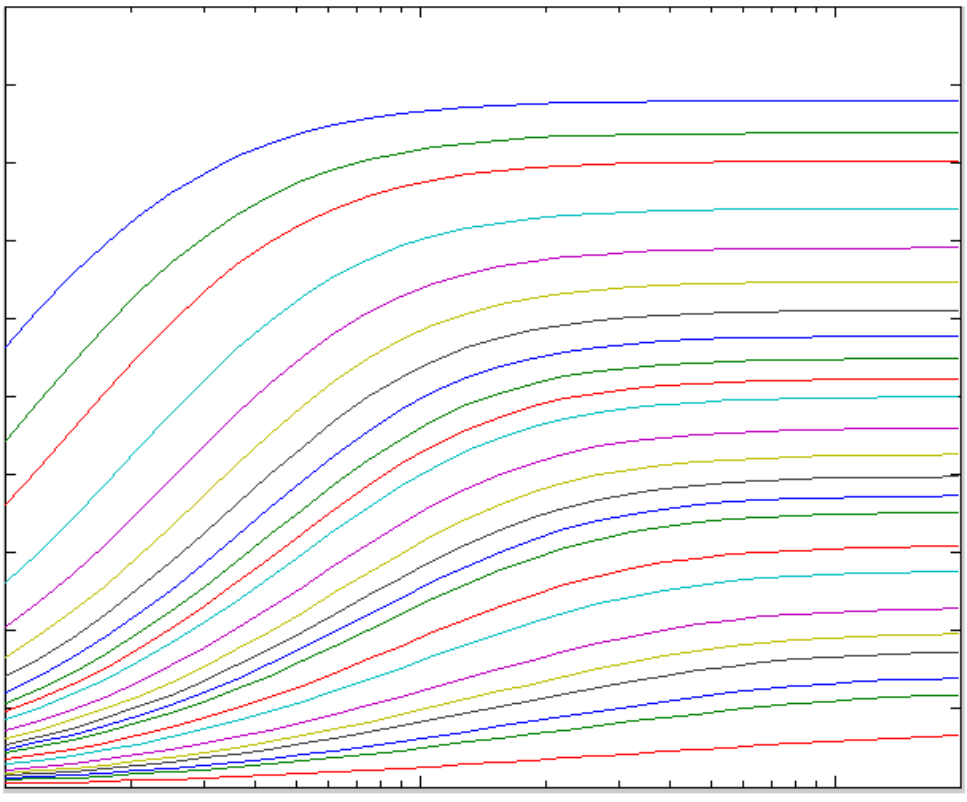


Figura 3.24 – Comparação do gráfico obtido com o de Cengel, para os fatores de forma de duas placas com um vértice em comum.

Resolvendo com a interface, simplesmente se preenche os campos com os dados e o resultado obtido é Fo=306,5259 e t=1098,3082s. Esse tempo encontrado é aproximadamente 9% menor. Logo, em uma indústria hipotética que ultizasse esse processo de produção, o tempo de resfriamento das placas seria 9% menor que o tempo calculado graficamente. De forma análoga, esses resultados podem ser expandidos para diversos processos que envolvam de transferência de calor e que possam aproximados modelados com algum dos modelos apresentados nesse projeto.

4 CONCLUSÃO

O objetivo principal desse trabalho foi a criação de uma interface gráfica que possibilite ao estudante dispensar o uso de gráficos ou tabelas na resolução de determinados problemas de transferência de calor, obtendo de forma prática resultados mais precisos. Se bem executado, o projeto pode resultar em uma ferramenta útil para a engenharia na produção de conhecimento. Além disso, a interface também pode ser usada na área industrial, onde resultados mais precisos podem diminuir o custo ou otimizar o tempo para certos processos. Um aluno de engenharia seria capaz de incorporar uma noção bem mais intuitiva do fenômeno de condução de calor em regime transiente do que pelos métodos tradicionais. Basta modificar um dos parâmetros do sistema para visualizar qual a sua influência no processo. Do outro lado, em uma indústria, a velocidade de uma linha de produção poderia ser significativamente aumentada, como, por exemplo, em um caso de resfriamento de determinada peça, onde o conhecimento desse tempo poderia otimizar o processo através do emprego de meios ou materiais. Daí a importância de se chegar a uma interface gráfica de aparência profissional, de boa concepção e amigável ao usuário. A esse propósito considera-se que os objetivos foram alcançados.

Alguns erros e desafios surgiram durante o desenvolvimento da interface, mostrados no relatório, sendo todos solucionados. O resultado foi uma interface criada em uma plataforma de iteração fácil e sugestiva, que apresenta de forma limpa e clara diversos resultados de cálculos complexos, sendo funcional e apresentando resultados satisfatórios para uma ampla faixa de condições e parâmetros.

A interface possui uma boa abrangência de fenômenos de transferência de calor, englobando condução transiente em paredes planas, cilindros infinitos e esferas, sólidos semi- infinitos e radiação térmica. Muitos problemas práticos podem ser resolvidos usando algum desses modelos.

REFERENCIAS BIBLIOGRAFICAS

Alves, M.B.M., Arruda, S.M. **COMO FAZER REFERÊNCIAS: bibliográficas, eletrônicas e demais formas de documentos.** Universidade Federal de Santa Catarina. Disponível em: < http://bu.ufsc.br/framerefer.html >. Acesso em: 26/05/2005.

Arpaci, V. S., **Conduction Heat Transfer**. University of Michigan. 1966.

Çengel, Y. A**., Transferência de Calor e Massa: Uma Abordagem Prática**. 3. ed. São Paulo. McGran-Hill, 2009.

Edwards, C. H. J.; Penney, D. E., **Introdução à Algebra Linear.** 5. ed. Rio de Janeiro. LTC. 2000.

Figueiredo, D. G., **Análise de Fourier e Equações Diferenciais Parciais.** 4. ed. Rio de Janeiro. Instituto Nacional de Matemática Pura e Aplicada. 2005.

Gregory, F. N.; Sanford, A. K., **Heat Transfer.** Cambridge University Press, 2009.

Incropera, F. P., et. al. **Fundamentos de Transferência de Calor e Massa**. 6. ed. Rio de Janeiro. LTC, 2008.

Ruggiero, M. A. G.; Lopes, V. L. R., **Cálculo Numérico: Aspectos Teóricos e Computacionais.** 2. ed. Rio de Janeiro. Makron Books, 1996.

Ribando, R. J. **Heat Transfer Tools.** 3. ed. McGraw-Hill. 2001

Caldeira, Armando. Condução: Fundamentos. 2011. 19slides. NotasdeAula. ApresentaçãoMSPowerPoint.

Boelter, L. M. K.; Cherry, V. H.; Johnson, H. A.; Martinelli, R. C. **Heat Transfer Notes.** New York. McGraw-Hill, 1965.

Grober, H.; Erk, S.; Grigull, U. **Fundamentals of Heat Transfer.** New York. McGraw-Hill, 1961.

UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO. Faculdade de Engenharia. **Curso de MATLAB 5.1.** 2. ed. Rio de Janeiro. 2009.

UNIVERSIDADE DO MINHO. Escola de Engenharia. **MATLAB: Uma Ferramenta para Engenharia.** Portugal. 1999.

UNIVERSIDADE DA BEIRA INTERIOR. Departamento de Engenharia Electromecânica. **Apontamentos de MATLAB**. Portugal. 2002.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023: Informação e Documentação** -Referências - Elaboração. Rio de Janeiro: ABNT, 2000

ANEXOS

```
function varargout = Heisler(varargin)
% HEISLER M-file for Heisler.fig
%      HEISLER, by itself, creates a new HEISLER or raises the existing
%      singleton*.
%
%      H = HEISLER returns the handle to a new HEISLER or the handle to
%      the existing singleton*.
%
%      HEISLER('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in HEISLER.M with the given input arguments.
%
%      HEISLER('Property','Value',...) creates a new HEISLER or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before Heisler_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Heisler_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Parede_Plana is made visible.
function Parede_Plana_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Parede_Plana (see VARARGIN)

% Choose default command line output for Parede_Plana
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

handles.data.x=0;
handles.data.L=0;
handles.data.T_inf=0;
handles.data.T_i=0;
handles.data.h=0;
```

```
handles.data.k=0;
handles.data.p=0;
handles.data.cp=0;
handles.data.T=NaN;
handles.data.t=0;
handles.data.erro=.001;
guidata(hObject,handles);

if isnan(k)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
x=handles.data.x;
L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
handles.data.k = k;
erro=handles.data.erro;

guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_parede_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_parede(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'L (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'L (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
```

```
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


L = str2double(get(hObject, 'String'));
if isnan(L)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
x=handles.data.x;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
handles.data.L = L;
erro=handles.data.erro;

guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_parede_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
```

```
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_parede(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'L (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'L (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


h = str2double(get(hObject, 'String'));
if isnan(h)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end

x=handles.data.x;
L=handles.data.L;
```

```
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
erro=handles.data.erro;

handles.data.h = h;
guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_parede_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_parede(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'L (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'L (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
```

```
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


handles.data.x = x;
guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
else
    t = condt_parede_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end

    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
```

```
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


if get(handles.radio_temperatura,'value')==1
    t=Tort;
    handles.data.t = Tort;
    guidata(hObject,handles);
    T=condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    T=Tort;
    handles.data.T = Tort;
    guidata(hObject,handles);
    t = condt_parede_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_parede(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'L (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'L (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
```

```
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


handles.data.cp = cp;
guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_parede_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_parede(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'L (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'L (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
```

```
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end

L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
cp=handles.data.cp;
T=handles.data.T; t=handles.data.t;
erro=handles.data.erro;
handles.data.p = p;
guidata(hObject,handles);

if get(handles.radio_temperatura,'value')==1
    T=condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_parede_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_parede(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
```

```
    shading interp
    xlabel(handles.ax,'L (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'L (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end

x=handles.data.x;
L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
tmax=handles.data.tmax;

if get(handles.radio_temperatura,'value')==1
    t=10^((log10(tmax)+3)*get(handles.slider1,'value')-3);
    handles.data.Tort=t;
    guidata(hObject,handles);
    T=condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.Tort,'String',t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    T=(T_i-T_inf)*get(handles.slider1,'Value')+T_inf;
    handles.data.T=T;
```

```
    guidata(hObject,handles);
    t=condt_parede_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.Tort,'String',T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end

if isfinite(T) && isfinite(t)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_parede(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
        xlabel('t(s)')
        ylabel('T(°C)')
        lbx=get(handles.axes6,'xlabel');
        set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)
0])
        set(lbx,'Backgroundcolor','White')
        lby=get(handles.axes6,'ylabel');
        set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
        set(lby,'rotation',1)
    else
        l1=semilogy(condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
        xlabel('T(°C)')
        ylabel('t(s)')
        lbx=get(handles.axes6,'xlabel');
        set(lbx,'position',get(lbx,'position')+[.4*(T_i-T_inf) .0006*tmax
0])
        set(lbx,'Backgroundcolor','White')
        lby=get(handles.axes6,'ylabel');
        set(lby,'position',get(lby,'position')+[.13*(T_i-T_inf) -.06*tmax
0])
        set(lby,'rotation',1)
    end
```

```
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
end


function mean = condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t)
%{9}
x_ad=x./L;
Bi=h*L/k;
Fo=k*t/p/cp/L^2;
%{10}
v=[L,h,p,cp,k];

%{4}
n=600;
%{2}
erro=0.00001;

x_=1.57;
xp=zeros(1,n);
cp=zeros(1,n);
if all(v)
    %{3}
    for i=1:n
        while tan_identity(x_,Bi)>=erro
            x_=x_-tan_identity(x_,Bi)/del_tan_identity(x_);
        end
        %{5}
        xp(1,i)=x_;
        x_=x_+pi;
    end

    for i = 1:n
        %{6}
        cp(i)=4*sin(xp(i))/(xp(i)*2+sin(xp(i)*2));
    end

    S=0;
    for i = 1:n
        %{7}
        S=S+cp(i)*exp(-xp(i)^2*Fo)*cos(xp(i)*x_ad);
    end
    %{8}
    T=T_inf+S*(T_i-T_inf);
    mean=T;
else
    mean=NaN;
end
%{1}
function tan_identity = tan_identity(x,Bi)
tan_identity=x.*tan(x)-Bi;
function dti = del_tan_identity(x)
dti=x*sec(x)^2+tan(x);

%{11}
function mean = condt_parede_t(x,L,T_inf,T_i,h,k,p,cp,T)
t0=0.000;
erro=0.0001;
```

```
t1=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
M=9;
A=0;
while M>erro
        t_=t1-(condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t1)-T)/...
            ((condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t1)-T)-
(condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t0)-T))*(t1-t0);
        M=condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t_)-T;
        if M<0
            t1=t_;
            M=-M;
        else
            t0=t_;
        end
end
mean=t_;

%{12}
function mean = d_condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)
mean=sqrt(((condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t)-
condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t+erro))/...
    erro)^2);

%{16}
function tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro)
t=15;
if d_condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
    while d_condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
        %{13}
        t=t-5;
    end
    while d_condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)>erro
        %{14}
        t=t+1;
    end
    while d_condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
        %{15}
        t=t-.05;
    end
else
    while d_condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)>erro
        %{13}
        t=t+5;
    end
    while d_condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
        %{14}
        t=t-1;
    end
    while d_condt_parede_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)>erro
        %{15}
        t=t+.05;
    end
end
tmax=t;

%{21}
function mean=pcolor_parede(L,T_inf,T_i,h,k,p,cp,t)
%{18}
i=-L:L/100:L;
i=meshgrid(i);
%{17}
T=condt_parede_T(i,L,T_inf,T_i,h,k,p,cp,t);
```

```
mean=T;

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Cilindro_Infinito_OpeningFcn, ...
                   'gui_OutputFcn',  @Cilindro_Infinito_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

k = str2double(get(hObject, 'String'));
if isnan(k)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
x=handles.data.x;
L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
handles.data.k = k;
erro=handles.data.erro;

guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_cilindro_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_cilindro(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
```

```
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end

guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_cilindro_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_cilindro(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
```

```
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


x=handles.data.x;
L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
erro=handles.data.erro;

handles.data.h = h;
guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
```

```
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_cilindro_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_cilindro(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end
```

```
T_inf = str2double(get(hObject, 'String'));
if isnan(T_inf)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
x=handles.data.x;
L=handles.data.L;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
erro=handles.data.erro;

handles.data.T_inf = T_inf;
guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_cilindro_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_cilindro(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
```

```
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


x = str2double(get(hObject, 'String'));
if isnan(x)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
erro=handles.data.erro;

handles.data.x = x;
guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
else
    t = condt_cilindro_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
```

```
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function T_i_Callback(hObject, eventdata, handles)

T_i = str2double(get(hObject, 'String'));
if isnan(T_i)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
x=handles.data.x;
L=handles.data.L;
T_inf=handles.data.T_inf;
h=handles.data.h;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
erro=handles.data.erro;

handles.data.T_i = T_i;
guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_cilindro_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
```

```
    Tp=pcolor_cilindro(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end

Tort = str2double(get(hObject, 'String'));
if isnan(Tort)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
x=handles.data.x;
L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
p=handles.data.p;
```

```
cp=handles.data.cp;
erro=handles.data.erro;

if get(handles.radio_temperatura,'value')==1
    t=Tort;
    handles.data.t = Tort;
    guidata(hObject,handles);
    T=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    T=Tort;
    handles.data.T = Tort;
    guidata(hObject,handles);
    t = condt_cilindro_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_cilindro(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
```

```
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


if get(handles.radio_temperatura,'value')==1
    T=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_cilindro_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_cilindro(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
```

```
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


if (hObject == handles.radio_temperatura)
    set(handles.Tort_text, 'String', 't :');
    set(handles.Tort, 'String', 0);
    set(handles.Tort_unit, 'String', 's');
    set(handles.torT_text, 'String', 'Temperatura');
    set(handles.torT_value, 'String', 'T = ');
else
    set(handles.Tort_text, 'String', 'T :');
    set(handles.Tort, 'String', 0);
    set(handles.Tort_unit, 'String', '°C');
    set(handles.torT_text, 'String', 'tempo');
    set(handles.torT_value, 'String', 't = ');
end
axes(handles.ax)
cla
axes(handles.axes6)
cla
set(handles.slider1,'Value',0)


if get(handles.radio_temperatura,'value')==1
    t=10^((log10(tmax)+3)*get(handles.slider1,'value')-3);
    handles.data.Tort=t;
    guidata(hObject,handles);
    T=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.Tort,'String',t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    T=(T_i-T_inf)*get(handles.slider1,'Value')+T_inf;
    handles.data.T=T;
    guidata(hObject,handles);
    t=condt_cilindro_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.Tort,'String',T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end

if isfinite(T) && isfinite(t)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_cilindro(L,T_inf,T_i,h,k,p,cp,t);
```

```
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
        xlabel('t(s)')
        ylabel('T(°C)')
        lbx=get(handles.axes6,'xlabel');
        set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)
0])
        set(lbx,'Backgroundcolor','White')
        lby=get(handles.axes6,'ylabel');
        set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
        set(lby,'rotation',1)
    else
        l1=semilogy(condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
        xlabel('T(°C)')
        ylabel('t(s)')
        lbx=get(handles.axes6,'xlabel');
        set(lbx,'position',get(lbx,'position')+[.4*(T_i-T_inf) .0006*tmax
0])
        set(lbx,'Backgroundcolor','White')
        lby=get(handles.axes6,'ylabel');
        set(lby,'position',get(lby,'position')+[.13*(T_i-T_inf) -.06*tmax
0])
        set(lby,'rotation',1)
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
end

    set(hObject,'BackgroundColor',[.9 .9 .9]);

T=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,L);
if isfinite(T)
    set(handles.radio_temperatura,'Value',1)
```

```
    set(handles.Animar,'String','Parar')
    set(handles.Tort,'Enable','off');
    set(handles.Tort_text, 'String', 't :');
    set(handles.Tort, 'String', 0);
    set(handles.Tort_unit, 'String', 's');
    set(handles.torT_text, 'String', 'Temperatura');
    set(handles.torT_value, 'String', 'T = ');
    tmax=handles.data.tmax;
    i=.001:tmax/1000:tmax;
    Tg=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,i);
    for t=10.^[-3:log10(tmax)/80:log10(tmax)]
        if get(hObject,'Value')
            axes(handles.ax)
            ii=-L:L/100:L;
            [ii,jj]=meshgrid(ii,ii);
            Tp=pcolor_cilindro(L,T_inf,T_i,h,k,p,cp,t);
            pcolor(ii,jj,Tp)
            C=[T_inf,T_i];
            caxis(C)
            colorbar
            lbc=colorbar;
            lbc=get(lbc,'Ylabel');
            set(lbc,'string','T(°C)')
            set(lbc,'rotation',1)
            set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
            shading interp
            xlabel(handles.ax,'R (m)');
            lbx=get(handles.ax,'xlabel');
            set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
            set(lbx,'Backgroundcolor','White')
            ylabel(handles.ax,'R (m)');
            lby=get(handles.ax,'ylabel');
            set(lby,'rotation',1)
            set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
            set(lby,'Backgroundcolor','White')

            axes(handles.axes6)
            cla
            T=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t);
            if get(handles.radio_temperatura,'value')==1
                l1=semilogx(i,Tg);
                l2=line([t,t],[T_inf,T]);
                l3=line([.001,t],[T,T]);
            else
                l1=semilogy(Tg,i);
                l2=line([T_inf,T],[t,t]);
                l3=line([T,T],[.001,t]);
            end
            set(l1,'LineWidth',2,'color','k')
            set(l2,'color','k')
            set(l3,'color','k')
            axis tight
            grid('on')
            set(handles.torT_value,'String',strcat('T =  ',num2str(T),'
°C'))
            set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
            set(handles.Tort,'String',t)
            xlabel('t(s)')
            lbx=get(handles.axes6,'xlabel');
            set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-
T_inf)  0])
            set(lbx,'Backgroundcolor','White')
```

```
            ylabel('T(°C)')
            lby=get(handles.axes6,'ylabel');
            set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf)
0])
            set(lby,'rotation',1)
        else
            break
        end
    end
    set(handles.Tort,'Enable','on');
    set(handles.Animar,'String','Animação');
    set(handles.Animar,'Value',0);
end


function mean = condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t)
%{9}
x_ad=x./L;
Bi=h*L/k;
Fo=k*t/p/cp/L^2;
%{10}
v=[L,h,p,cp,k];

%{4}
n=9;
%{2}
if all(v)
    erro=0.00001;

iBi=[.01,.02,.04,.06,.08,.1,.2,.3,.4,.5,.6,.7,.8,.9,1,2,3,4,5,6,7,8,9,10,20
,30,40,50,100,1000];

ixc=[.1412,.1995,.2814,.3438,.3936,.4417,.6170,.7465,.8516,.9408,1.0184,1.0
873,1.1490,1.2048,1.2558,1.5995,1.7887,1.9081,1.9898,2.049,2.0937,2.1286,2.
1566,2.1795,2.2880,2.3261,2.3455,2.3572,2.3809,2.4048];

idxc=[3.6931,3.6375,3.5607,3.5035,3.4565,3.4160,3.2665,3.1626,3.0828,3.0186
,2.9656,2.9212,2.8835,2.8513,2.8237,2.6915,2.6747,2.6938,2.7233,2.7543,2.78
34,2.8097,2.8331,2.8537,2.9688,3.0148,3.0391,3.0540,3.0843,3.1121];
    x_=interp1(iBi,ixc,Bi);
    dx=interp1(iBi,idxc,Bi);
    xc=zeros(1,n);
    cc=xc;
    %{3}
    for i=1:n
        x0=x_-pi/10000;
        x1=x_+pi/10000;
        M=9;
        while M>erro
            x_=x1-trc(x1,Bi)/(trc(x1,Bi)-trc(x0,Bi))*(x1-x0);
            M=trc(x_,Bi);
            if M>0
                x1=x_;
            else
                x0=x_;
                M=-M;
            end
        end
        %{5}
        xc(i)=x_;
        x_=x_+dx;
        if i~=1
```

```
            dx=xc(i)-xc(i-1);
        end
    end

    for i = [1:n]
        %{6}
cc(i)=2*besselj(1,xc(i))/(xc(i)*(besselj(0,xc(i))^2+besselj(1,xc(i))^2));
    end

    S=0;
    for i = [1:n]
        %{7}
        S=S+cc(i)*exp(-xc(i)^2*Fo)*besselj(0,xc(i)*x_ad);
    end
    %{8}
    T=T_inf+S*(T_i-T_inf);
    mean=T;
else
    mean=NaN;
end

%{1}
function trc = trc(x,Bi)
trc=x.*besselj(1,x)./besselj(0,x)-Bi;

%{11}
function mean = condt_cilindro_t(x,L,T_inf,T_i,h,k,p,cp,T)
t0=0.000;
erro=0.0001;
t1=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
M=9;
A=0;
while M>erro
        t_=t1-(condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t1)-T)/...
            ((condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t1)-T)-
(condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t0)-T))*(t1-t0);
        M=condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t_)-T;
        if M<0
            t1=t_;
            M=-M;
        else
            t0=t_;
        end
end
mean=t_;

%{12}
function mean = d_condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)
mean=sqrt(((condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t)-
condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t+erro))/...
    erro)^2);

%{16}
function tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro)
t=15;
if d_condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
    while d_condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
        %{13}
        t=t-5;
    end
```

```
    while d_condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)>erro
        %{14}
        t=t+1;
    end
    while d_condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
        %{15}
        t=t-.05;
    end
else
    while d_condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)>erro
        %{13}
        t=t+5;
    end
    while d_condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
        %{14}
        t=t-1;
    end
    while d_condt_cilindro_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)>erro
        %{15}
        t=t+.05;
    end
end
tmax=t;

%{21}
function mean=pcolor_cilindro(L,T_inf,T_i,h,k,p,cp,t)
%{18}
i=-L:L/100:L;
[i,j]=meshgrid(i,i);
%{17},{19}
T=condt_cilindro_T(sqrt(i.^2+j.^2),L,T_inf,T_i,h,k,p,cp,t);
for u=1:length(i)
    for v=1:length(j)
        if sqrt(i(1,u)^2+j(v,1)^2)>=L
            %{20}
            T(u,v)=NaN;
        end
    end
end
mean=T;


guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_esfera_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_esfera(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
```

```
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


L = str2double(get(hObject, 'String'));
if isnan(L)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
x=handles.data.x;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
```

```
handles.data.L = L;
erro=handles.data.erro;

guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_esfera_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_esfera(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
```

```
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


x=handles.data.x;
L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
erro=handles.data.erro;

handles.data.h = h;
guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_esfera_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_esfera(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
```

```
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


end
x=handles.data.x;
L=handles.data.L;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
erro=handles.data.erro;

handles.data.T_inf = T_inf;
guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_esfera_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_esfera(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
```

```
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end

x = str2double(get(hObject, 'String'));
if isnan(x)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
T=handles.data.T;
t=handles.data.t;
erro=handles.data.erro;

handles.data.x = x;
guidata(hObject,handles);
```

```
if get(handles.radio_temperatura,'value')==1
    T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
else
    t = condt_esfera_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end

T_i = str2double(get(hObject, 'String'));
if isnan(T_i)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
if get(handles.radio_temperatura,'value')==1
    T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_esfera_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
```

```
    Tp=pcolor_esfera(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end


if get(handles.radio_temperatura,'value')==1
    t=Tort;
    handles.data.t = Tort;
    guidata(hObject,handles);
    T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    T=Tort;
    handles.data.T = Tort;
    guidata(hObject,handles);
```

```
    t = condt_esfera_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_esfera(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end
```

```
handles.data.cp = cp;
guidata(hObject,handles);
if get(handles.radio_temperatura,'value')==1
    T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_esfera_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_esfera(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
```

```
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

p = str2double(get(hObject, 'String'));
if isnan(p)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
x=handles.data.x;
L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
cp=handles.data.cp;
T=handles.data.T; t=handles.data.t;
erro=handles.data.erro;
handles.data.p = p;
guidata(hObject,handles);

if get(handles.radio_temperatura,'value')==1
    T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    t = condt_esfera_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end
set(handles.Bi,'String', strcat('Bi = ',num2str(h*L/k)));

if isfinite(T)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_esfera(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')
```

```
    axes(handles.axes6)
    tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
    handles.data.tmax=tmax;
    guidata(hObject,handles)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
    else
        l1=semilogy(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
    xlabel('t(s)')
    lbx=get(handles.axes6,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)  0])
    set(lbx,'Backgroundcolor','White')
    ylabel('T(°C)')
    lby=get(handles.axes6,'ylabel');
    set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
    set(lby,'rotation',1)
end

if (hObject == handles.radio_temperatura)
    set(handles.Tort_text, 'String', 't :');
    set(handles.Tort, 'String', 0);
    set(handles.Tort_unit, 'String', 's');
    set(handles.torT_text, 'String', 'Temperatura');
    set(handles.torT_value, 'String', 'T = ');
else
    set(handles.Tort_text, 'String', 'T :');
    set(handles.Tort, 'String', 0);
    set(handles.Tort_unit, 'String', '°C');
    set(handles.torT_text, 'String', 'tempo');
    set(handles.torT_value, 'String', 't = ');
end
axes(handles.ax)
cla
axes(handles.axes6)
cla
set(handles.slider1,'Value',0)
if get(handles.radio_temperatura,'value')==0
end
x=handles.data.x;
L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
tmax=handles.data.tmax;

if get(handles.radio_temperatura,'value')==1
```

```
   t=10^((log10(tmax)+3)*get(handles.slider1,'value')-3);
    handles.data.Tort=t;
    guidata(hObject,handles);
    T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t);
    set(handles.Tort,'String',t);
    set(handles.torT_value,'String',strcat('T =  ',num2str(T),' °C'))
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
else
    T=(T_i-T_inf)*get(handles.slider1,'Value')+T_inf;
    handles.data.T=T;
    guidata(hObject,handles);
    t=condt_esfera_t(x,L,T_inf,T_i,h,k,p,cp,T);
    set(handles.Tort,'String',T);
    set(handles.torT_value,'String',strcat('t =  ',num2str(t),' s'));
    set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)));
end

if isfinite(T) && isfinite(t)
    axes(handles.ax)
    i=-L:L/100:L;
    [i,j]=meshgrid(i,i);
    Tp=pcolor_esfera(L,T_inf,T_i,h,k,p,cp,t);
    pcolor(i,j,Tp)
    C=[T_inf,T_i];
    caxis(C)
    lbc=colorbar;
    lbc=get(lbc,'Ylabel');
    set(lbc,'string','T(°C)')
    set(lbc,'rotation',1)
    set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
    shading interp
    xlabel(handles.ax,'R (m)');
    lbx=get(handles.ax,'xlabel');
    set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
    set(lbx,'Backgroundcolor','White')
    ylabel(handles.ax,'R (m)');
    lby=get(handles.ax,'ylabel');
    set(lby,'rotation',1)
    set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
    set(lby,'Backgroundcolor','White')

    axes(handles.axes6)
    i=.001:tmax/1000:tmax;
    cla
    if get(handles.radio_temperatura,'value')==1
        l1=semilogx(i,condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i));
        l2=line([t,t],[T_inf,T]);
        l3=line([.001,t],[T,T]);
        xlabel('t(s)')
        ylabel('T(°C)')
        lbx=get(handles.axes6,'xlabel');
        set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-T_inf)
0])
        set(lbx,'Backgroundcolor','White')
        lby=get(handles.axes6,'ylabel');
        set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf) 0])
        set(lby,'rotation',1)
    else
        l1=semilogy(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i),i);
        l2=line([T_inf,T],[t,t]);
        l3=line([T,T],[.001,t]);
        xlabel('T(°C)')
```

```
        ylabel('t(s)')
        lbx=get(handles.axes6,'xlabel');
        set(lbx,'position',get(lbx,'position')+[.4*(T_i-T_inf) .0006*tmax
0])
        set(lbx,'Backgroundcolor','White')
        lby=get(handles.axes6,'ylabel');
        set(lby,'position',get(lby,'position')+[.13*(T_i-T_inf) -.06*tmax
0])
        set(lby,'rotation',1)
    end
    set(l1,'LineWidth',2,'color','k')
    set(l2,'color','k')
    set(l3,'color','k')
    axis tight
    grid('on')
end
function Animar_Callback(hObject, eventdata, handles)
x=handles.data.x;
L=handles.data.L;
T_inf=handles.data.T_inf;
T_i=handles.data.T_i;
h=handles.data.h;
k=handles.data.k;
p=handles.data.p;
cp=handles.data.cp;
erro=handles.data.erro;

T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,L);
if isfinite(T)
    set(handles.radio_temperatura,'Value',1)
    set(handles.Animar,'String','Parar')
    set(handles.Tort,'Enable','off');
    set(handles.Tort_text, 'String', 't :');
    set(handles.Tort, 'String', 0);
    set(handles.Tort_unit, 'String', 's');
    set(handles.torT_text, 'String', 'Temperatura');
    set(handles.torT_value, 'String', 'T = ');
    tmax=handles.data.tmax;
    i=.001:tmax/1000:tmax;
    Tg=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,i);
    for t=10.^[-3:log10(tmax)/80:log10(tmax)]
        if get(hObject,'Value')
            axes(handles.ax)
            ii=-L:L/100:L;
            [ii,jj]=meshgrid(ii,ii);
            Tp=pcolor_esfera(L,T_inf,T_i,h,k,p,cp,t);
            pcolor(ii,jj,Tp)
            C=[T_inf,T_i];
            caxis(C)
            colorbar
            lbc=colorbar;
            lbc=get(lbc,'Ylabel');
            set(lbc,'string','T(°C)')
            set(lbc,'rotation',1)
            set(lbc,'position',get(lbc,'Position')+[-4 .6*(T_i-T_inf) 0])
            shading interp
            xlabel(handles.ax,'R (m)');
            lbx=get(handles.ax,'xlabel');
            set(lbx,'position',get(lbx,'position')+[.87*L .3*L 0])
            set(lbx,'Backgroundcolor','White')
            ylabel(handles.ax,'R (m)');
            lby=get(handles.ax,'ylabel');
```

```
            set(lby,'rotation',1)
            set(lby,'position',get(lby,'position')+[.3*L .87*L 0])
            set(lby,'Backgroundcolor','White')

            axes(handles.axes6)
            cla
            T=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t);
            if get(handles.radio_temperatura,'value')==1
                l1=semilogx(i,Tg);
                l2=line([t,t],[T_inf,T]);
                l3=line([.001,t],[T,T]);
            else
                l1=semilogy(Tg,i);
                l2=line([T_inf,T],[t,t]);
                l3=line([T,T],[.001,t]);
            end
            set(l1,'LineWidth',2,'color','k')
            set(l2,'color','k')
            set(l3,'color','k')
            axis tight
            grid('on')
            set(handles.torT_value,'String',strcat('T =  ',num2str(T),'
°C'))
            set(handles.Fo,'String', strcat('Fo = ',num2str(k*t/p/cp/L^2)))
            set(handles.Tort,'String',t)
            xlabel('t(s)')
            lbx=get(handles.axes6,'xlabel');
            set(lbx,'position',get(lbx,'position')+[.72*tmax .45*(T_i-
T_inf)  0])
            set(lbx,'Backgroundcolor','White')
            ylabel('T(°C)')
            lby=get(handles.axes6,'ylabel');
            set(lby,'position',get(lby,'position')+[.0012 .34*(T_i-T_inf)
0])
            set(lby,'rotation',1)
        else
            break
        end
    end
    set(handles.Tort,'Enable','on');
    set(handles.Animar,'String','Animação');
    set(handles.Animar,'Value',0);
end


function mean = condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t)
%{9}
x_ad=x./L;
Bi=h*L/k;
Fo=k*t/p/cp/L^2;
%{10}
v=[L,h,p,cp,k];

%{4}
n=471;
%{2}
erro=0.00001;
x_=3.1415;
xe=zeros(1,n);
ce=zeros(1,n);
if all(v)
```

```
    %{3}
    for i=1:n
        while sqrt(cot_identity(x_,Bi)^2)>erro
            x_=x_-cot_identity(x_,Bi)/del_cot_identity(x_);
        end
        %{5}
        xe(i)=x_;
        x_=x_+pi;
    end

    for i =1:n
        %{6}
        ce(i)=4*(sin(xe(i))-xe(i)*cos(xe(i)))/(xe(i)*2-sin(xe(i)*2));
    end

    S=0;
    for i = 1:n
        %{7}
        S=S+ce(i)*exp(-xe(i)^2*Fo)*sin(x_ad.*xe(i))/xe(i)./x_ad;
    end
    %{8}
    T=T_inf+S.*(T_i-T_inf);
    mean=T;
else
    mean=NaN;
end

%{1}
function ide = cot_identity(x,Bi)
ide=1-x.*cot(x)-Bi;
function dide = del_cot_identity(x)
dide=-cot(x)+x*csc(x)^2;

%{11}
function mean = condt_esfera_t(x,L,T_inf,T_i,h,k,p,cp,T)
t0=0.000;
erro=0.0001;
t1=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro);
M=9;
A=0;
while M>erro
        t_=t1-(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t1)-T)/...
            ((condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t1)-T)-
(condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t0)-T))*(t1-t0);
        M=condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t_)-T;
        if M<0
            t1=t_;
            M=-M;
        else
            t0=t_;
        end
end
mean=t_;

%{12}
function mean = d_condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)
mean=sqrt(((condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t)-
condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t+erro))/...
    erro)^2);

%{16}
```

```
function tmax=gettmax(x,L,T_inf,T_i,h,k,p,cp,erro)
t=15;
if d_condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
    while d_condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
        %{13}
        t=t-5;
    end
    while d_condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)>erro
        %{14}
        t=t+1;
    end
    while d_condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
        %{15}
        t=t-.05;
    end
else
    while d_condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)>erro
        %{13}
        t=t+5;
    end
    while d_condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)<erro
        %{14}
        t=t-1;
    end
    while d_condt_esfera_T(x,L,T_inf,T_i,h,k,p,cp,t,erro)>erro
        %{15}
        t=t+.05;
    end
end
tmax=t;

%{21}
function mean=pcolor_esfera(L,T_inf,T_i,h,k,p,cp,t)
%{18}
i=-L:L/100:L;
[i,j]=meshgrid(i,i);
%{17},{19}
T=condt_esfera_T(sqrt(i.^2+j.^2),L,T_inf,T_i,h,k,p,cp,t);
for u=1:length(i)
    for v=1:length(j)
        if sqrt(i(1,u)^2+j(v,1)^2)>=L
            %{20}
            T(u,v)=NaN;
        end
    end
end
mean=T;
```

```
    case 1
        if isfinite([a,b,c])
            Fij=ff1(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
```

```
        end
    case 2
        if isfinite(a)
            Fij=ff2(a);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 3
        if isfinite([a,b])
            Fij=ff3(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 4
        if isfinite([a,b,c])
            Fij=ff4(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 5
        if isfinite([a,b,c])
            Fij=ff5(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=2*pi*a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 6
        if isfinite([a,b,c,d])
            Fij=ff6(a,b,c,d);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=(a-b)*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 7
        if isfinite([a,b])
            Fij=ff7(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 8
```

```
        if isfinite([a,b,c])
            Fij=ff8(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
    case 9
        if isfinite([a,b,c])
            Fij=ff9(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=pi*a^2*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
    case 10
        if isfinite([a,b,c])
            Fij=ff10(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
end



switch choice
    case 1
        if isfinite([a,b,c])
            Fij=ff1(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 2
        if isfinite(a)
            Fij=ff2(a);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 3
        if isfinite([a,b])
            Fij=ff3(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
```

```
            end
        end
    case 4
        if isfinite([a,b,c])
            Fij=ff4(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 5
        if isfinite([a,b,c])
            Fij=ff5(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=2*pi*a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 6
        if isfinite([a,b,c,d])
            Fij=ff6(a,b,c,d);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=(a-b)*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 7
        if isfinite([a,b])
            Fij=ff7(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 8
       if isfinite([a,b,c])
            Fij=ff8(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
    case 9
        if isfinite([a,b,c])
            Fij=ff9(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=pi*a^2*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
```

```
    case 10
        if isfinite([a,b,c])
            Fij=ff10(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
end
switch choice
    case 1
        if isfinite([a,b,c])
            Fij=ff1(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 2
        if isfinite(a)
            Fij=ff2(a);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 3
        if isfinite([a,b])
            Fij=ff3(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 4
        if isfinite([a,b,c])
            Fij=ff4(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 5
        if isfinite([a,b,c])
            Fij=ff5(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=2*pi*a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
```

```
    case 6
        if isfinite([a,b,c,d])
            Fij=ff6(a,b,c,d);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=(a-b)*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 7
        if isfinite([a,b])
            Fij=ff7(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 8
       if isfinite([a,b,c])
            Fij=ff8(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
    case 9
        if isfinite([a,b,c])
            Fij=ff9(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=pi*a^2*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
    case 10
        if isfinite([a,b,c])
            Fij=ff10(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
end
switch choice
    case 1
        if isfinite([a,b,c])
            Fij=ff1(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
```

```
    case 2
        if isfinite(a)
            Fij=ff2(a);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 3
        if isfinite([a,b])
            Fij=ff3(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 4
        if isfinite([a,b,c])
            Fij=ff4(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 5
        if isfinite([a,b,c])
            Fij=ff5(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=2*pi*a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 6
        if isfinite([a,b,c,d])
            Fij=ff6(a,b,c,d);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=(a-b)*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 7
        if isfinite([a,b])
            Fij=ff7(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 8
       if isfinite([a,b,c])
```

```
            Fij=ff8(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
    case 9
        if isfinite([a,b,c])
            Fij=ff9(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=pi*a^2*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
    case 10
        if isfinite([a,b,c])
            Fij=ff10(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
end
switch choice
    case 1
        if isfinite([a,b,c])
            Fij=ff1(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 2
        if isfinite(a)
            Fij=ff2(a);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 3
        if isfinite([a,b])
            Fij=ff3(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 4
        if isfinite([a,b,c])
```

```
            Fij=ff4(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 5
        if isfinite([a,b,c])
            Fij=ff5(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=2*pi*a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 6
        if isfinite([a,b,c,d])
            Fij=ff6(a,b,c,d);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=(a-b)*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 7
        if isfinite([a,b])
            Fij=ff7(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 8
       if isfinite([a,b,c])
            Fij=ff8(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
    case 9
        if isfinite([a,b,c])
            Fij=ff9(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=pi*a^2*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
    case 10
        if isfinite([a,b,c])
            Fij=ff10(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
```

```
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
end
b = handles.data.b;
c = handles.data.c;
choice = handles.data.choice;
handles.data.d = d;
Ti=handles.data.Ti;
Tj=handles.data.Tj;
guidata(hObject, handles);
switch choice
    case 1
        if isfinite([a,b,c])
            Fij=ff1(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 2
        if isfinite(a)
            Fij=ff2(a);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 3
        if isfinite([a,b])
            Fij=ff3(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 4
        if isfinite([a,b,c])
            Fij=ff4(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 5
        if isfinite([a,b,c])
            Fij=ff5(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=2*pi*a*Fij*5.670e-8*(Ti^4-Tj^4);
```

```
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 6
        if isfinite([a,b,c,d])
            Fij=ff6(a,b,c,d);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=(a-b)*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m'))
            end
        end
    case 7
        if isfinite([a,b])
            Fij=ff7(a,b);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W/m^2'))
            end
        end
    case 8
       if isfinite([a,b,c])
            Fij=ff8(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
    case 9
        if isfinite([a,b,c])
            Fij=ff9(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=pi*a^2*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
    case 10
        if isfinite([a,b,c])
            Fij=ff10(a,b,c);
            set(handles.text18,'String',strcat('Fij =  ',num2str(Fij)))
            if isfinite([Ti,Tj])
                Qij=a*b*Fij*5.670e-8*(Ti^4-Tj^4);
                set(handles.text21,'String',strcat('Qij =  ',num2str(Qij),'
W'))
            end
        end
end
    set(hObject,'BackgroundColor','white');
end



function mean = ff1(wi,wj,L)
Wi=wi/L;
```

```
Wj=wj/L;
Fij=(sqrt((Wi+Wj)^2+4)-sqrt((Wj-Wi)^2+4))/2/Wi;
mean=Fij;

function mean = ff2(alfa)
Fij=1-sin(alfa/2);
mean=Fij;

function mean = ff3(wi,wj)
Fij=(1+(wj/wi)-sqrt(1+(wj/wi)^2))/2;
mean=Fij;

function mean = ff4(wi,wj,wk)
Fij=(wi+wj-wk)/2/wi;
mean=Fij;

function mean = ff5(ri,rj,s)
R=rj/ri;
S=s/ri;
C=1+R+S;
Fij=1/2/pi*(pi+sqrt(C^2-(R+1)^2)-sqrt(C^2-(R-1)^2)+(R-1)*acos((R/C)-(1/C))-
(R+1)*acos((R/C)+(1/C)));
mean=Fij;

function mean = ff6(s1,s2,r,L)
Fij=r/(s1-s2)*(atan(s1/L)-atan(s2/L));
mean=Fij;

function mean = ff7(s,D)
Fij=1-sqrt(1-(D/s)^2)+(D/s)*atan(sqrt((s^2-D^2)/D^2));
mean=Fij;

function mean = ff8(x,y,L)
X=x/L;
Y=y/L;
Fij=2/pi/X/Y*(log(sqrt((1+X^2)*(1+Y^2)/(1+X^2+Y^2)))+X*sqrt(1+Y^2)*atan(X/s
qrt(1+Y^2))+Y*sqrt(1+X^2)*atan(Y/sqrt(1+X^2))-X*atan(X)-Y*atan(Y));
mean=Fij;

function mean = ff9(ri,rj,L)
Ri=ri/L;
Rj=rj/L;
S=1+(1+Rj^2)/Ri^2;
Fij=1/2*(S-sqrt(S^2-4*(rj/ri)^2));
mean=Fij;

function mean = ff10(x,y,z)
h=z/x;
w=y/x;
Fij=1/pi/w*(w*atan(1/w)+h*atan(1/h)-
sqrt(h^2+w^2)*atan(1/sqrt(h^2+w^2))+1/4*log((1+w^2)*(1+h^2)/(1+w^2+h^2)*(w^
2*(1+w^2+h^2)/(1+w^2)/(w^2+h^2))^(w^2)*(h^2*(1+h^2+w^2)/(1+h^2)/(h^2+w^2))^
(h^2)));
mean=Fij;
```