Sandia National Laboratories

Operated for the U.S. Department of Energy by
Sandia Corporation
P.O. Box 5800
Albuquerque, New Mexico 87185-1189

date: September 24, 2014

from: Daniel Dolan

subject: The `ThermalDiffusion` class

The `ThermalDiffusion` class was created to simulate one-dimensional thermal diffusion across one or more material layers. Each layer is assumed to have constant conductivity K and diffusivity κ. Interface conductance between layers may be specified. Internal heating as a function of position and time is also supported. The `ThermalDiffusion` class is included in the SMASH package [1] as part of the PDE (Partial Differential Equation) subpackage.

Figure 1 illustrates how the `ThermalDiffusion` class represents a multi-layer diffusion problem. This particular problem has three layers, each represented by six nodes: layer 1 is nodes 1–6, layer 2 is nodes 7–12, and layer 3 is nodes 13–18. The nodes are uniformly spaced within each layer, but constant spacing between layers is not required. Contact between layers is represented by overlapping nodes: nodes 6–7 and nodes 12–13 are located at the same position but may have different temperature values.

Diffusion between nodes is simulated using the numerical method of lines (MOL). [2, 3] This approach represents the diffusion equation as a set of coupled, ordinary differential equations. At each time step, the time derivative of temperature T at interior position x_m is represented as:

$$\frac{dT(x_m,t))}{dt} \approx \frac{\kappa_m}{\Delta_m^2}\Big[T(x_{m+1},t) - 2T(x_m,t) + T(x_{m-1},t)\Big] + F(x_m,t) \tag{1}$$

where F is the local heating function (scaled by density and specific heat to have units of [temperature]/[time]). Time derivatives are calculated at all interior nodes to advance the calculation from the initial state to a specified stopping time. Boundary nodes are managed separately with non-symmetric derivatives that maintain the fourth-order accuracy of Equation 1.

- Exterior boundaries ($m = 1$ and $m = M$) are treated as perfectly insulating.

$$\frac{\partial T}{\partial x} = 0 \tag{2}$$

- Internal boundaries (material A on the left, material B on the right) are governed by two conditions.

$$K_A \frac{\partial T_A}{\partial x} = K_B \frac{\partial T_B}{\partial x} \tag{3}$$

$$T_B = T_A + \frac{K_A}{G}\frac{\partial T_A}{\partial x} \tag{4}$$

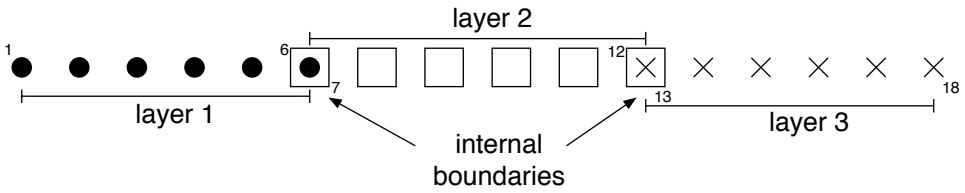

Figure 1: Conceptual setup of a three-layer diffusion problem

For ideal coupling, conductance approaches infinity and $T_A = T_B$, but real interfaces have finite conductance due to incomplete contact or interface mismatch. Incomplete surface contact effects depend significantly on the applied mechanical stress; [4, 5] characteristic values of G range from 10^3–10^5 W/m^2·K. Interface mismatch (also known as boundary conductance [6]) depends on the similarity of thermal carriers (electrons or phones) but typically allows much higher conductance (10^7–10^9 W/m^2·K at room temperature).

This article describes applications of the `ThermalDiffusion` class. First, the process of creating, configuring, and using a `ThermalDiffusion` object is described. Next, a series of example calculations are presented. In several cases, simulations are compared to analytical solutions from Reference 7.

Creating and using `ThermalDiffusion` objects

`ThermalDiffusion` objects are created by calling the class with a single input to define number of material layers.

```
>> object=ThermalDiffusion(2); % two-layer simulation
```

There is no limit to the number of layers in a simulation, but this number cannot be changed after an object is created. The number of layers L automatically defines the number of internal interface $(L - 1)$, and these are used to define internal storage arrays for layer/interface parameters.

Several object properties are used to define the simulation.

Nodes specifies the number of nodes used in each layer. The default value is 100 nodes per layer. This property can be overridden, but each layer must have at least 5 nodes.

Thickness* specifies the physical thickness of each layer.

InitialTemperature* specifies the initial temperature of each layer. Numerical values assign a uniform temperature to each layer. A function handle can also be used to calculate $T_0(x)$ at the beginning of the simulation.

Conductivity* specifies the thermal conductivity (K) of each layer.

Diffusivity* specifies the thermal diffusivity (κ) of each layer.

InterfaceConductance specifies the thermal conductance (G) of each interface (default is infinity).

InternalHeating specifies a function handle that evaluates the scaled heating function $F(x, t)$. If no function handle is provided, the simulation assumes $F(x, t) = 0$.

Output Display ('verbose') or suppress ('silent') solver output report (default is 'silent').

Properties marked with an asterisk have no default value and must be specified before a simulation can be performed. Layer/interface properties can be specified individually or set to a common value.

```
>> object.Nodes=[100 200]; % assign 100 nodes to layer 1, 200 nodes to layer 2
>> object.Nodes=[100]; % assign 100 nodes to every layer
```

Individual specifications must be consistent with the layers/interfaces in the simulations: a three-layer simulation can accept three Nodes, Thickness, Conductivity, or Diffusivity values and two InterfaceConductance values. The `ThermalDiffusion` class does not use any particular system of units for the diffusion calculations—users must maintain dimensional consistency!

Additional properties are provided for controlling the `ode15s` solver used by the simulation.

RelTol Relative error tolerance for the simulation (default is 10^{-4}).

AbsTol Absolute error tolerance for the simulation (default is 10^{-4}).

InitialStep Initial time step for the simulation (default is `[]`, causing the solver to determine initial time step).

MaxStep Maximum time step for the simulation (default is `[]`, causing the solver to increase time step as much as possible while maintaining tolerance requirements).

These properties should typically be left at their default values, but changes can be made as needed (as demonstrated in example E).

Once all required properties have been specified, the diffusion problem is initiated with the simulate method.

```
>> result=simulate(object,tmax);
```

Simulations begin at $t = 0$ and continue until the specified maximum time is reached. Results are stored as a `mesh1` object, which provides methods for displaying information at fixed time (snapshots) or fixed node value (histories).

```
>> snapshot(result,tplot); % plot snapshots at specified times
>> [x,T]=snapshot(result,tplot); % extract snapshot data
>> history(result,nodes); % plot histories at specified nodes
>> [t,T]=history(result,nodes); % extract history data
```

Both the `mesh1` and `ThermalDiffusion` class are part of the PDE subpackage. To learn more about either, access the online documentation by typing "doc SMASH.PDE".

Example A: perfect coupling between two solids

Diffusion between two layers (each with uniform initial temperature) is a convenient test problem for the `ThermalDiffusion` class. In these problems, temperature can be expressed in a dimensionless form using the minimum and maximum initial temperatures.

$$v_k = \frac{T_k - T_{min}}{T_{max} - T_{min}} \qquad k = 1..2 \tag{5}$$

The initial conditions can be now be specified as 0 (cold layer) and 1 (hot layer).

Consider a hot layer of tin ($K = 63$ W/m·K, $\kappa = 4 \times 10^{-5}$ m^2/s) in perfect contact ($G = \infty$) with a cold layer of lithium fluoride ($K = 10$ W/m·K, $\kappa = 2 \times 10^{-6}$ m^2/s). For all two-layer examples described here, the hot layer is located to the left of the cold layer. Figure 2 shows a diffusion simulation when both layers are 50 μm thick, each represented with 500 nodes. Such thickness allow the simulation to mimic the interface between two semi-infinite materials, a problem which has an analytic solution.

The interface temperature ($x = 50$ μm) in this problem determined by the effusivity ratio.

$$v_{\text{interface}} = \frac{\eta_2}{\eta_1 + \eta_2} \tag{6}$$

Effusivity ($\eta \equiv \sqrt{k\rho c_p}$) quantifies a material's ability to exchange energy though thermal contact; the "1" subscript refers to the cold material and the "2" subscript refers to the hot material. The effusivities for tin and lithium fluoride are 9.9×10^3 J/m^2·K·s$^{1/2}$ and 7.1×10^3 J/m^2·K·s$^{1/2}$, respectively. Despite these awkward dimensions, it is the effusivity ratio that really matters, and in this case the normalized interface temperature is 0.581. The simulation produces a similar result, though a few nanoseconds are required before the value stabilizes to this value. Smaller time steps and finer meshes seem to help, but several iterations are always needed to transition the problem from the infinite temperature gradient at $t = 0$ to the value specified by Equation 6.

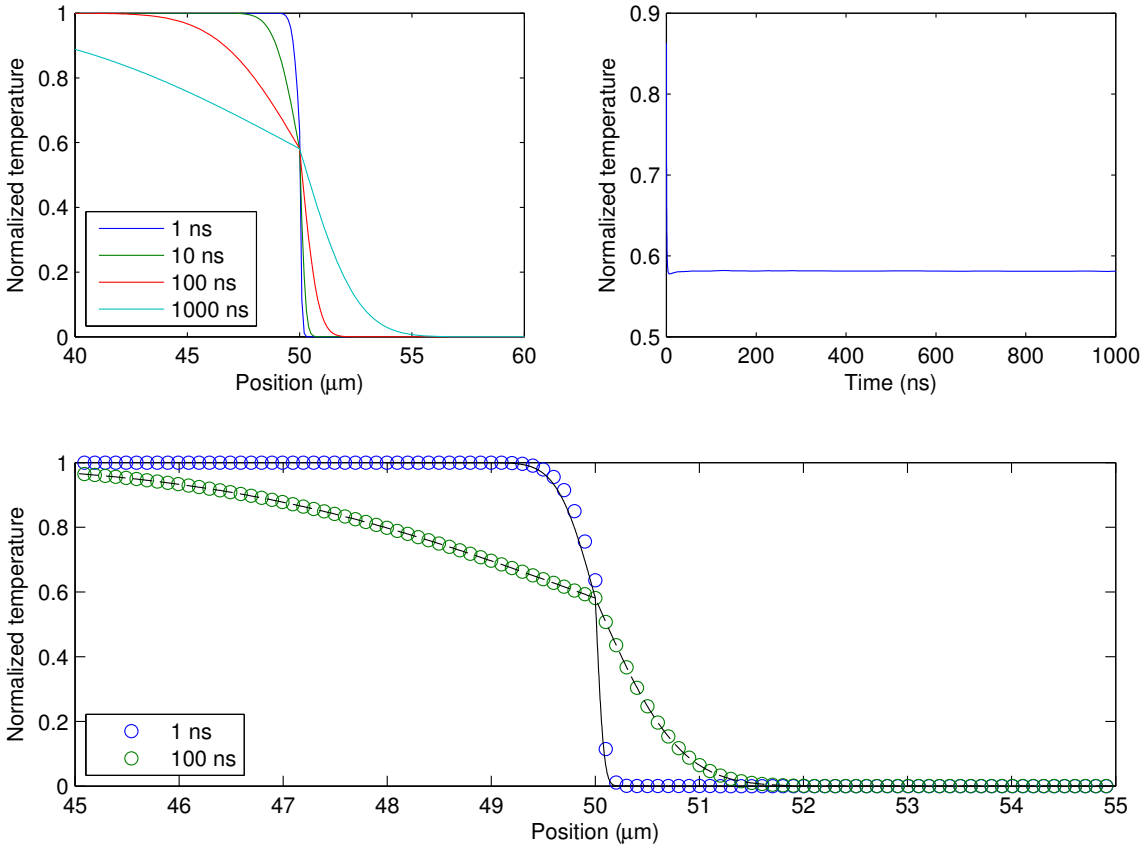

Figure 2: Simulation of hot tin in perfect contact with cold lithium fluoride. (top left) Temperature snapshots near the material interface. (top right) Temperature history on the tin side of the interface; the lithium fluoride side of this interface is identical to the tin side after a few nanoseconds. (bottom) Simulation snapshots are shown as open circles, analytic solutions as black lines.

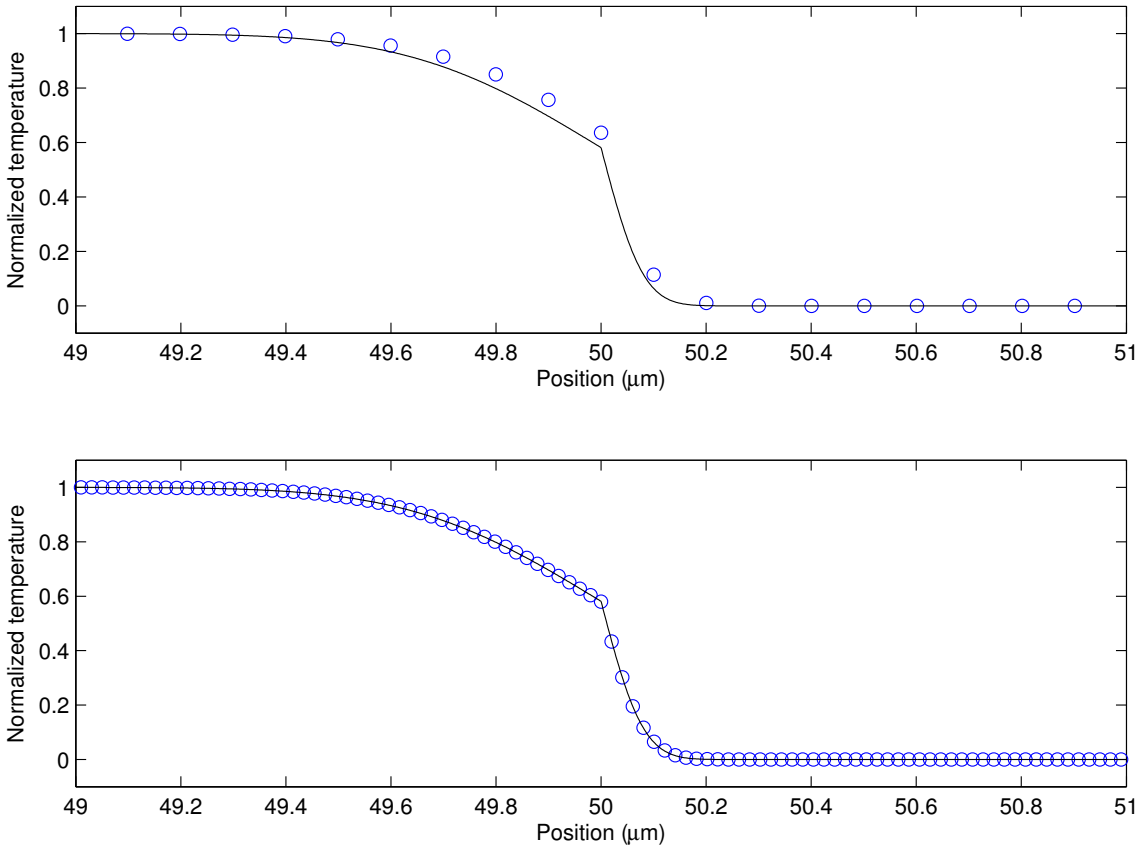

Figure 3: Interface details for the tin/lithium fluoride simulations ($t = 1$ ns). (top) Interface snapshot for a coarse mesh, two-layer simulation. (bottom) Refined simulation using fine meshing near the interface.

Example B: simulation refinements

Although the simulation describe in example A mimics the analytic solution, close inspection of the boundary region (Figure 3) reveals problems. Changing the tolerance time step options improves agreement with the analytic solution, but increasing the number of nodes is the most effective approach. Precisely how this is done greatly affects the time required for the simulation.

One way to increase node density is to use more nodes per layer. In example A, each layer was assigned 500 nodes, and this can easily be changed.

```
>> object.Nodes=5000; % use 5000 nodes per layer
```

The 500 node/layer simulation requires a few seconds, but it turns out that using 5000 nodes/layer takes substantially more time. The verbose output mode:

```
>> object.OutputMode='verbose';
```

allows users to quantify the amount of time and effort being used in each simulation. Experimenting with the number of nodes N (leaving all other settings fixed) reveals that the simulation time scales with N^3. As such, a simulation using 5000 nodes per layer takes 1000 times longer than a simulation with 500 nodes per layer!

The scaling of computation time with the number of nodes can be understood from the stability criteria. [2].

$$\frac{\kappa \Delta t}{(\Delta x)^2} \leq \frac{1}{2} \tag{7}$$

For uniform spacing over N nodes, the number of time steps that must be taken to complete the simulation scales with N^2. The number of calculations per step scales with N, so the total number of calculations is proportional to N^3 (consistent with empirical tests). Not only does uniformly increasing the node density become computationally slow, the effort is wasted wherever temperature varies gradually.

Strategic node placement is a far more effective strategy than uniformly increasing node density. For the two-layer example described in the previous section, sharp thermal gradients are confined to central interface. Rather than using two layers with a large number of nodes (> 1000), the problem can be recast as four layers: two composed of tin and two composed of lithium fluoride. The lower plot of Figure 3 was obtained by dividing the original 50 μm layers into an exterior 48 μm layer and an interior 2 μm layer; 100 nodes were assigned to each layer. The revised calculation slightly slower than the original simulation, but the results match the analytic result far more accurately.

Example C: imperfect coupling between two solids

Continuing with the four-layer approach described in example B, suppose that the interface between tin and lithium fluoride has finite conductance.[1] Figure 4 shows calculated snapshots and histories for $G = 10^8$ W/m^2·K. This value is characteristic of an insulator-metal interface at room temperature; extremely mismatched materials have surface conductances of 10^7 W/m^2·K.

Finite conductance causes a temperature discontinuity whenever heat is passing through an interface. This is particularly evident for snapshots prior to 100 ns, but the effect persists for a very long time. Even at $t = 1000$ ns, there is 2% temperature difference across the interface.

Example D: coupling between a solid and liquid

Consider the thermal transport between a hot metal (such as gold) and a cold liquid (such as water). This situation is extreme in several ways. Gold has a much larger conductivity (300 W/m·K) and diffusivity (10^4 m^2/s) than liquid water (0.6 W/m·K and 10^{-7} m^2/s, respectively). Furthermore, the conductance between these extremely different materials is of order 10^7 W/m^2·K, inhibiting thermal transfer at the interface.

Figure 5 shows a diffusion simulations for gold and water using the four-layer approach described in previous examples. The results illustrate important aspects of a highly mismatched diffusion problem.

- The temperature gradient is very low in the metal layer and very high in the liquid layer, consistent with Equation 3.
- Finite interface conductance permits temperature discontinuity, allowing the metal side of the interface to remain very near the initial bulk temperature ($v = 1$ in this case).
- The liquid side of the interface moves toward the temperature of the metal side, but full equilibration takes a very long time.

This and the previous example indicate that interface temperature is not a single, constant quantity as posited by Equation 6, at least not for some time. Interface temperature is controlled by multiple

[1] This simulation has three interfaces, but the first and third interfaces are artifacts of layering (coarse/fine/fine/coarse mesh) used to improve accuracy near the true material interface. The appropriate conductance setting is `[inf G inf]`.

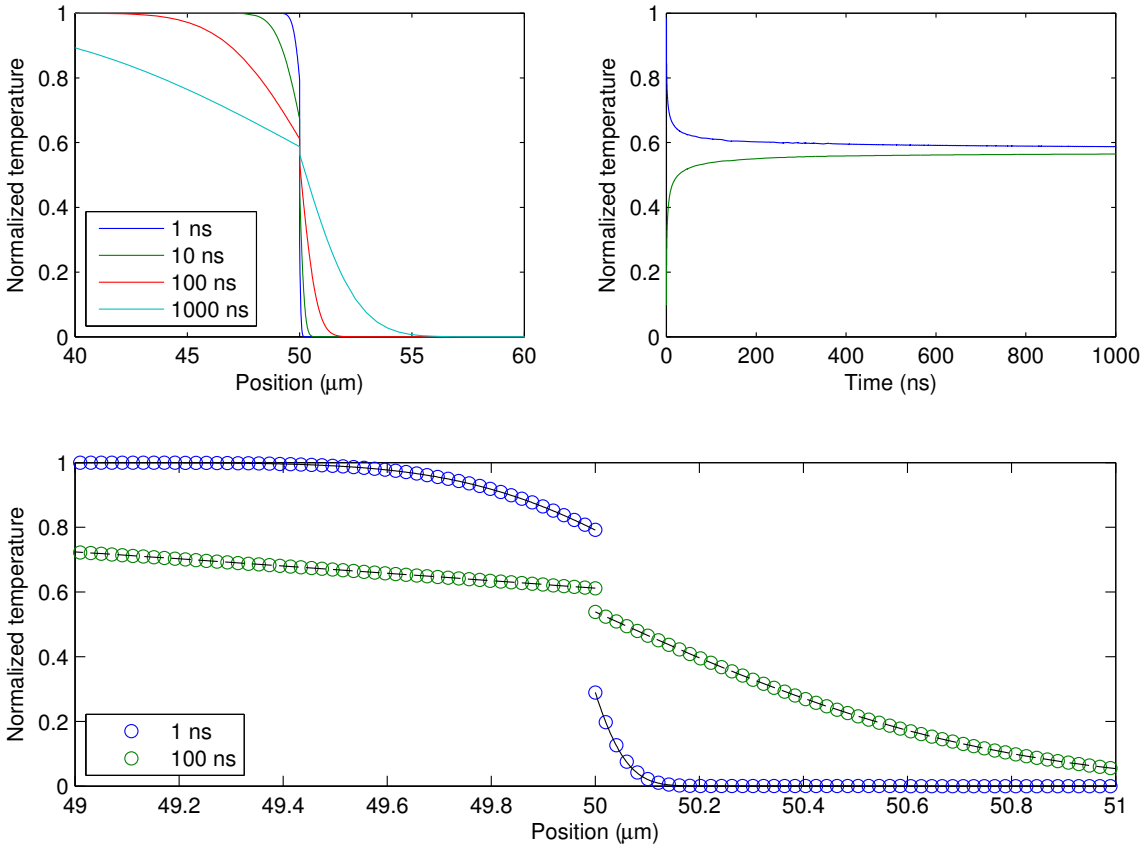

Figure 4: Simulation of a material interface (tin and lithium fluoride) with finite conductance. (top left) Temperature snapshots near the material interface. (top right) Temperature histories on the tin (blue) and lithium fluoride (green) side of the interface. (bottom) Simulation snapshots are shown as open circles, analytic solutions as black lines.

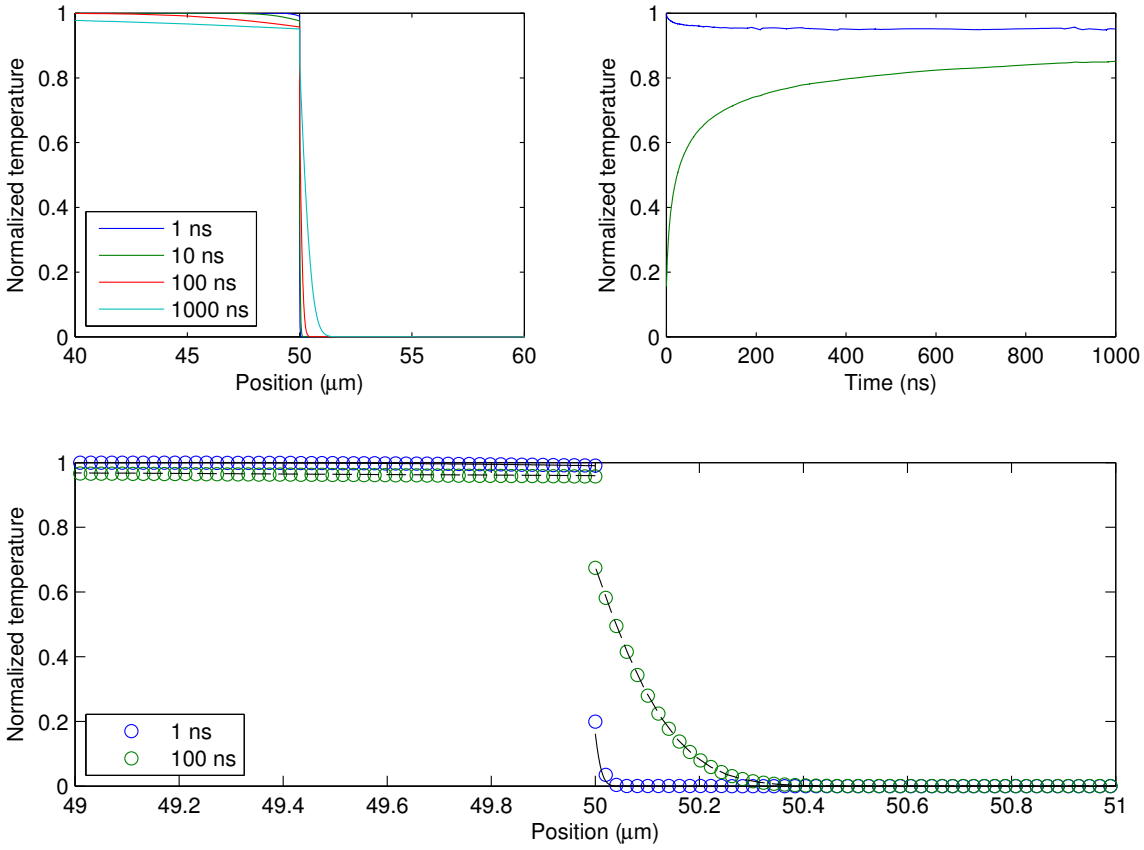

Figure 5: Simulation of metal-liquid (gold-water) interface with finite conductance. (top left) Temperature snapshots near the material interface. (top right) Temperature histories on the tin (blue) and lithium fluoride (green) side of the interface. (bottom) Simulation snapshots are shown as open circles, analytic solutions as black lines.

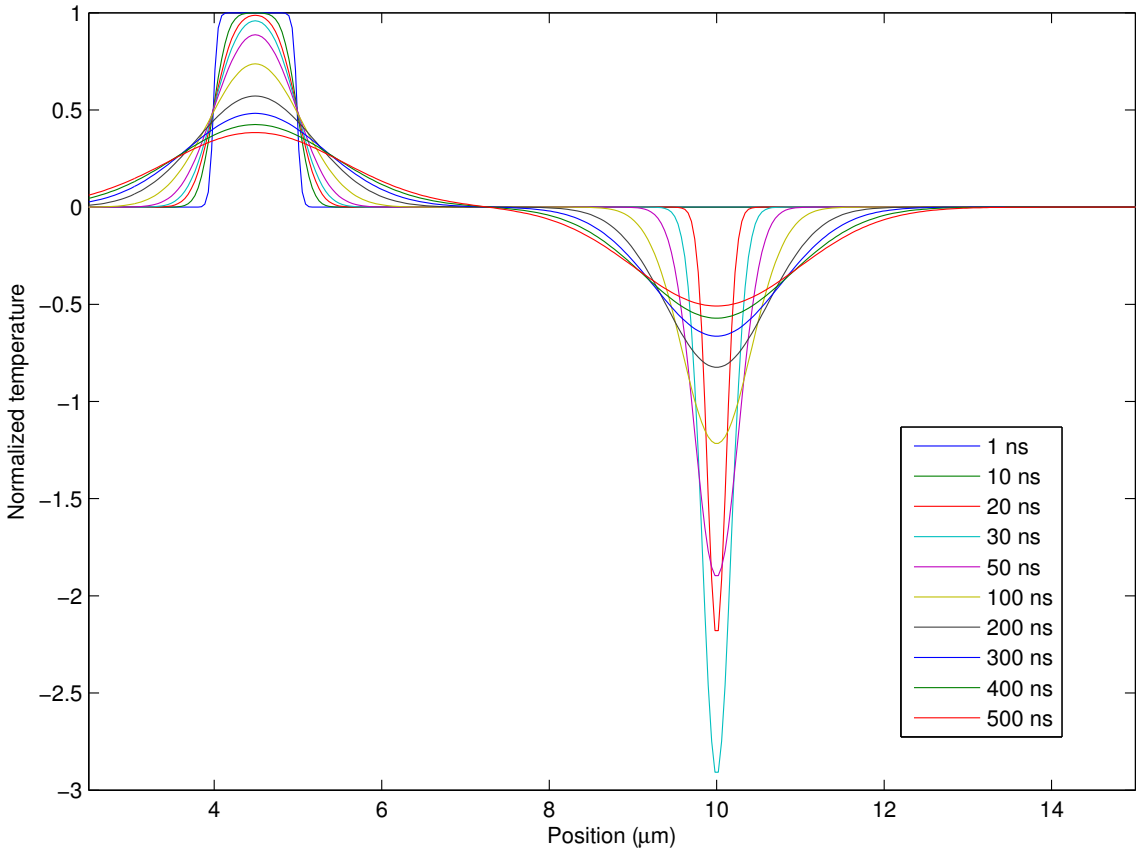

Figure 6: Temperature snapshots for lithium fluoride with an non-uniform initial temperature distribution (4–5 μm) and a transient heat sink (10 μm).

parameters, not effusivity ratio alone. Numerical simulations require five parameters (two conductivities, two diffusivities, and one conductance); the analytic solution (at the interface) reduces these five quantities to two distinct parameters.

Example E: internal heating

As a final example, consider a single layer of lithium fluoride that begins with a non-uniform temperature distribution and has a transient, localized heat sink.

$$T_0(x) = \begin{cases} 1 & 4\ \mu\text{m} < x < 5\ \mu\text{m} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$F(x,t) = Ae^{-(x-x_0)^2/2\sigma_x^2} e^{-(t-t_0)^2/2\sigma_t^2} \tag{9}$$

For this example, let $A = -10$ K/ns, $x_0 = 10$ μm, $\sigma_x = 0.1$ μm, $t_0 = 20$ ns, and $\sigma_t = 2$ ns.

Figure 6 shows temperature snapshots from a simulation of 20 μm of lithium fluoride using 500 nodes. The initial temperature distribution and heating functions are defined as subfunctions:

```
function T=initial(x)

T=zeros(size(x));
```

```
index=(x>4e-6) & (x<5e-6);
T(index)=1;

end

function F=sink(x,t)

A=-1/(1e-9); % K/s
x0=mean(x); % m
sigmax=0.1e-6; % m
t0=20e-9; % s
sigmat=2e-9; % s
F=A*exp(-(x-x0).^2/(2*sigmax^2))*exp(-(t-t0)^2/(2*sigmat^2));

end
```

and passed to the simulation using function handles.

```
>> object.InitialTemperature=@initial;
>> object.InternalHeating=@sink;
```

An additional customization is needed to prevent the simulation from skipping over the transient heat sink.

```
>> object.MaxStep=0.1e-9; % 0.1 ns maximum time steps
```

All other aspects of the calculation—parameter definition, simulation launch, and visualization—are similar to the multi-layer examples described above.

References

[1] D.H. Dolan and T. Ao. The Sandia Matlab AnalysiS Hierarchy (SMASH) package. Technical Report (in preparation), Sandia National Laboratories, (2014).

[2] W.E. Schiesser. *The Numerical Method of Lines*. Academic Press, San Diego, (1991).

[3] W.E. Schiesser and G.W. Griffiths. *A compendium of partial differential equation models: method of lines analysis with MATLAB*. Cambridge University Press, (2009).

[4] R.B. Jacobs and C. Starr. Thermal conductance of metallic contacts. *Review of Scientific Instruments* **10**, 140 (1939).

[5] H. Yüncü. Thermal contact conductance of nominally flat surfaces. *Heat and Mass Transfer* **43**, 1 (2006).

[6] P.E. Hopkins. Thermal transport across solid interfaces with nanoscale imperfections: effects of roughness, disorder, dislocations, and bonding on thermal boundary conductance. *ISRN Mechanical Engineering* **2013**, 682586 (2013).

[7] H.S. Carslaw and J.C. Jaeger. *Conduction of heat in solids*. Oxford University Press, Oxford, 2nd edition, (1959).

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000.