Notes on a 2D Thermal Model: an Example for MATLAB's `Parallel Computing Toolbox`

https://people.sc.fsu.edu/∼jburkardt/presentations/...
...fem2d_heat_steady_spmd.pdf

J.V. Burkardt (burkardt@vt.edu) E.M. Cliff (ecliff@vt.edu)
Interdisciplinary Center for Applied Mathematics
Virginia Tech
Blacksburg, VA 24061

February 3, 2024

1 Background

We develop a finite-element model for steady heat-conduction in two space dimensions. This is the basis for a MATLAB code illustrating the use of `spmd`-mode to assemble the needed matrices and the use of `codistributed`-arrays to solve the linear system.

2 A Mathematical Model

We consider steady heat conduction in a plane. The governing partial differential equation is

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) + F(x,y) \ = \ 0 \, , \quad (x,y) \in \Omega \, , \tag{1}$$

where:

- $\Omega = \{(x,y) \quad | \quad 0 \le x \le L, \quad 0 \le y \le w\} \subset \mathbb{R}^2$,
- $F(x,y)$ is a specified source term,

- $k_x > 0$ ($k_y > 0$) is the conductivity in the x direction (the y-direction).

Boundary conditions for our problem are:

$$\frac{\partial T(x,0)}{\partial y} = \frac{\partial T(x,w)}{\partial y} = 0 \ , \tag{2}$$

$$k_x \frac{\partial T(L,y)}{\partial x} = f(y) \ , \tag{3}$$

$$k_x \frac{\partial T(0,y)}{\partial x} = \alpha(y) \ \left(T(0,y) - \beta(y)\right) \ . \tag{4}$$

In the final (Robin) boundary condition we require that $\alpha(y) > 0$.

3 Numerical Approximation

3.1 Spatial discretization

Our numerical solution of (1) is based on a weak formulation. We multiply by a test function $\Psi(x,y)$ and integrate over the spatial domain Ω:

$$\int_\Omega \left[\frac{\partial}{\partial x}\left(k_x \frac{\partial}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial}{\partial y}\right)\right] T \ \Psi \ \mathrm{d}\omega + \int_\Omega F(x,y) \ \Psi \ \mathrm{d}\omega = 0 \ . \tag{5}$$

The 1st term in (5) is integrated by parts

$$\int_\Omega \left[\frac{\partial}{\partial x}\left(k_x \frac{\partial}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial}{\partial y}\right)\right] T \ \Psi \ \mathrm{d}\omega \\ = -\int_\Omega \left(k\nabla T \cdot \nabla \Psi\right) \ \mathrm{d}\omega + \int_{\partial\Omega} \left(k\nabla T \cdot \hat{n}\right) \ \Psi \ \mathrm{d}\sigma \ , \tag{6}$$

where in the boundary integral, $\hat{n}$ is an outward normal to the surface, and the integration is in an anti-clockwise sense around the region Ω. Imposing the specified boundary conditions (2 - 4), the boundary term in (6) evaluates to

$$\int_{\partial\Omega} \left(k\nabla T \cdot \hat{n}\right) \ \Psi \ \mathrm{d}\sigma = \int_0^w f(y)\Psi(L,y) \ \mathrm{d}y \\ - \int_w^0 \alpha(y) \left[T(0,y) - \beta(y)\right] \Psi(0,y) \ \mathrm{d}y \ . \tag{7}$$

3.2 Galerkin Finite Element

We seek an approximate solution of the form

$$T_N(x,y) = \sum_{\jmath=1}^{N} z_\jmath \; \Phi_\jmath(x,y) \; . \tag{8}$$

Substitute the approximation (8) into the weak-form and use for test functions $\Psi = \Phi_\imath$ leads to:

$$\begin{aligned}
&\sum_\jmath z_\jmath \int_\Omega (k\nabla\Phi_\jmath \cdot \nabla\Phi_\imath) \;\mathrm{d}\omega - \int_\Omega F(x,y)\;\Phi_\imath \;\mathrm{d}\omega \\
&-\left[\int_0^w f(y)\Phi_\imath(L,y)\;\mathrm{d}y - \int_w^0 \alpha(y)\left(\sum_\jmath z_\jmath\Phi_\jmath(0,y) - \beta(y)\right)\Phi_\imath(0,y)\;\mathrm{d}y\right] \;=\; 0 \\
&\hspace{20em} \text{for } \imath = 1,2,...,N \; .
\end{aligned} \tag{9}$$

Gathering terms leads to

$$\begin{aligned}
&\sum_\jmath \left[\int_\Omega \left(k\nabla\Phi_\jmath \cdot \nabla\Phi_\imath \;\mathrm{d}\omega + \int_w^0 \alpha(y)\;\Phi_\jmath(0,y)\;\Phi_\imath(0,y)\;\mathrm{d}y\right)\right] z_\jmath \\
&\qquad - \left[\int_\Omega F(x,y)\Phi_\imath \;\mathrm{d}\omega\right] \\
&\qquad - \left[\int_0^w f(y)\Phi_\imath(L,y)\;\mathrm{d}y + \int_w^0 \alpha(y)\beta(y)\Phi_\imath(0,y)\;\mathrm{d}y\right] = 0 \\
&\hspace{20em} \text{for } \imath = 1,2,...,N \; .
\end{aligned} \tag{10}$$

In matrix terminology

$$\mathbf{M}\,z \;-\; \mathbf{F} \;-\; \mathbf{b} = 0 \; . \tag{11}$$

3.2.1 Quadratic Functions on Triangular Elements

We define a set of x coordinates $X = \{0 = x_1 < x_2 < ... < x_{2\ell+1} = L\}$, a set of y coordinates $Y = \{0 = y_1 < y_2 < ... < y_{2m+1} = w\}$, and impose a regular $((2\ell+1)\times(2m+1))$ grid on Ω $(\ell, m \geq 1)$. Using odd-labeled abscissa values from X and the odd-labeled ordinate values from Y generate $\ell\; m$ rectangles; diagonals divide these into $2\;\ell\;m$ global triangles. Figure 1 shows the case $\ell = 10$, $m = 6$ $(n_x = 21,\; n_y = 13)$.

A local computational triangle is shown in Figure 2. Note that the (local) vertex points are numbered 1 - 3 in order as one traverses the edges of the

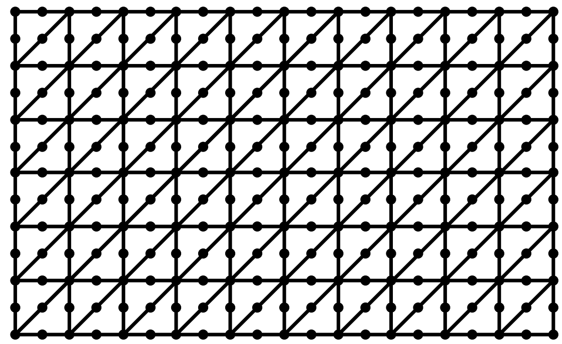


Figure 1: 21×13 Grid

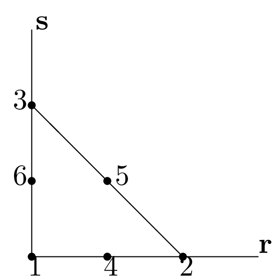


Figure 2: Computational Triangle

triangle in counter-clockwise fashion. The edge-center points are similarly numbered 4 - 6.

We construct six quadratic functions: three of these interpolate values at vertex points (H_1, H_2, H_3), and three interpolate values at the segment center points (H_4, H_5, H_6).

$$\begin{aligned}
H_1(r,s) &= 1 - 3r + 2r^2 - 3s + 4rs + 2s^2 \\
H_2(r,s) &= -r + 2r^2 \\
H_3(r,s) &= -s + 2s^2 \\
H_4(r,s) &= 4r - 4r^2 - 4rs \\
H_5(r,s) &= 4rs \\
H_6(r,s) &= 4s - 4rs - 4s^2
\end{aligned}$$

Figure 3 displays the shape of these local interpolating functions for the vertex points (left) and the segment center points (right) [1, from p 139].

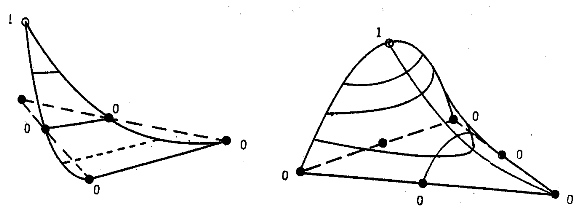


Figure 3: Basic Quadratic Functions

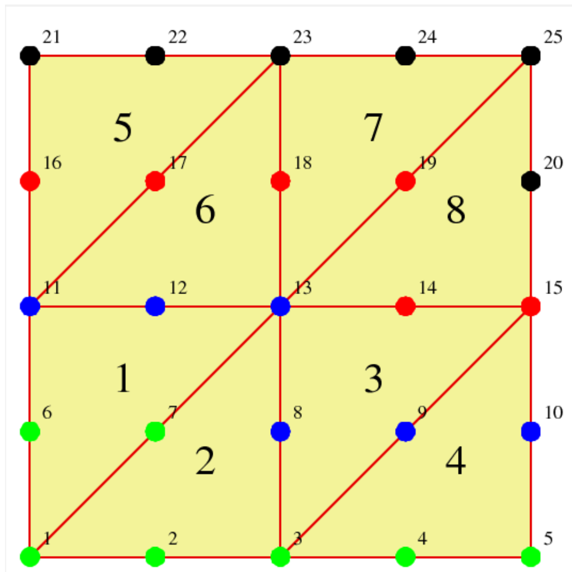


Figure 4: 5×5 grid with labelled points & elements

4 Assembling the arrays

As a first step in assembling the arrays $\mathbf{M}, \mathbf{F}, \mathbf{b}$ (see equ'n 11), it's necessary to label the nodes (so we know which z component corresponds to which node). Furthermore, it's helpful to label the (triangular) elements.

For the nodes begin by noting that we have a structured (in fact, Cartesian) grid (see Figure 4). Begin with `node 1` at the lower left. `Node` 2 is immediately to the right and so on to `node` 5 at the far right at the bottom. `Node` 6 is back at the left, just above `node` 1, and so on to `node` 25 at the upper right.

For the triangles, note that we have ℓ rectangles across Ω (see Figure 4) and that the diagonals divide these into $2\,\ell$ triangles. Beginning at the lower left, we label the first 'row' of triangles as elements $1, 2, ..., 2\,\ell$. The 2^{nd} row is similarly labelled, and so on to the last $(2\,\ell\, m)$.

Evaluating the matrix $\mathbf{M}$ and the vectors $\mathbf{F}$ and $\mathbf{b}$ requires integration of the basis functions (and their gradients) along with the source function F and boundary functions, over the region Ω and along its boundaries at $x = 0$ and $x = L$. This is accomplished by decomposing Ω into triangular regions, evaluating the appropriate integrals over the triangular regions and summing the results.

4.1 Serial Implementation

We begin by describing the steps in a serial implementation (single-processor).

- Geometry Module:
  - define `x` and `y` grids and assemble the `node` array (`n_nodes` × `2`)
  - define element connectivity (which nodes are in each triangle) `e_conn(n_elements, 6)`
  - construct list(s) of elements with points on the left (right) boundary
- Integration Module
  (`do for each element`)
  - identify nodes in the element and their global coordinates
  - compute Gauss points and weights, evaluate the (six) shape functions and their spatial derivatives at the Gauss points.
  - evaluate the (`6` × `6`) matrix of integrals of the products of the shape function (derivatives) over this element (**M_loc**)
  - evaluate the distributed source term at the Gauss points and integrate the weighted integral of the shape functions over this element (**F_loc**)
  - if the element contains node points on the left boundary, evaluate the functions $\alpha(\cdot)$ and $\beta(\cdot)$ at the Gauss points on the boundary, and evaluate the (`6` × `6`) matrix of integrals of the products of the shape function and α along the $x = 0$ boundary of this element (**M_loc**). Then evaluate the (`6` × `1`) matrix of integrals of the products of the shape function and the $\alpha\ \beta$ product along the $x = 0$ boundary for this element (**b_loc**)

– if the element contains node points on the right boundary, evaluate the function $f(\cdot)$ at the Gauss points on the boundary, and evaluate the (6×1) matrix of integrals of the products of the shape function and the f along the $x = L$ boundary for this element (**b_loc**)

– map the contributions from the local arrays ((6×6) and (6×1)) to the global arrays $\mathbf{M}, \mathbf{F}$, and $\mathbf{b}$ ((`n_nodes` $\times$ `n_nodes`) and (`n_nodes` $\times$ `1`)).

4.2 Parallel considerations

In the parallel implementation the matrix $\mathbf{M}$ and the vectors $\mathbf{F}$ and $\mathbf{b}$ are `codistributed`. For the matrix $\mathbf{M}$ the columns are distributed so that each `lab` has (about) `n_nodes/n_labs` columns (and `n_nodes` rows). The matrix `M_lab` consists of the columns of $\mathbf{M}$ corresponding to nodes on this `lab`. The vectors `F_lab` (`b_lab`) consists of the rows of $\mathbf{F}$ ($\mathbf{b}$) corresponding to nodes on this `lab`.

In the loop over the set of triangular elements, if the intersection of the set of the nodes associated with the current element and the set of nodes on the particular `lab` is empty, then simply *fall through the loop*. Figure 4 provides a depiction of the situation with 4 `labs`: each lab is responsible for the grid points of a given color. `lab_1` (green) has nodes `1 - 7`. These nodes appear in triangles 1 - 4, but not in triangles 5 - 8. Note, however, that in this case the blue nodes appear in all eight triangles, so that `lab_2` must evaluate integrals over all of the elements. Clearly, parallelism is not useful for this (small) case.

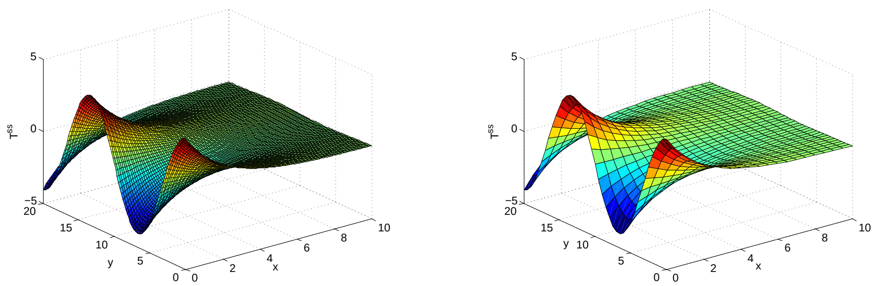
5 Example Results

Example 1

We first consider a case with $\Omega = [0,\ 10] \times [0,\ 20]$ with $k_x = k_y = 1$, and $F \equiv 0$. On the right boundary we take $f = 0$, while on the left boundary we take $\alpha = \hat{\alpha}$ (a constant), $\beta(y) = \hat{\beta} \cos \frac{p\pi y}{w}$. In this case a standard separation of variables analysis leads to a solution:

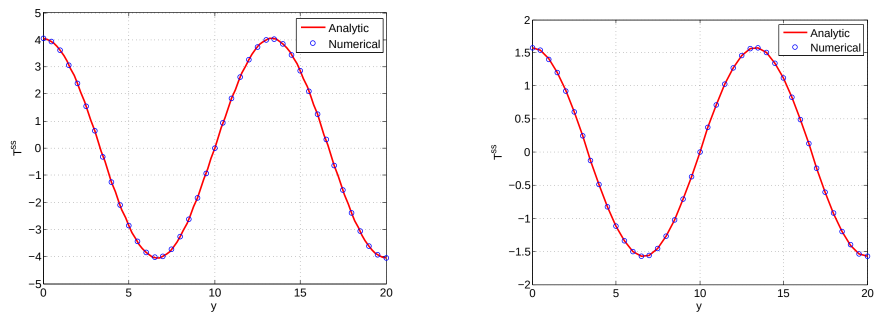
$$T^{\mathrm{ss}}(x, y) = \frac{\hat{\alpha}\hat{\beta}\ \cos \frac{p\pi y}{w}\ \cosh \frac{p\pi(L-x)}{w}}{\hat{\alpha} \cosh \frac{p\pi L}{w} + \frac{p\pi}{w} \sinh \frac{p\pi L}{w}}\ . \tag{12}$$

Figure 5 compares surface plots of the analytic solution (5a) and the numerical approximation on a 21×41 grid (5b). Figure 6 compares line



(a) Analytic Solution (b) 21×41 grid

Figure 5: Surface Plot Solutions



(a) $x = 0$ (b) $x = 2$

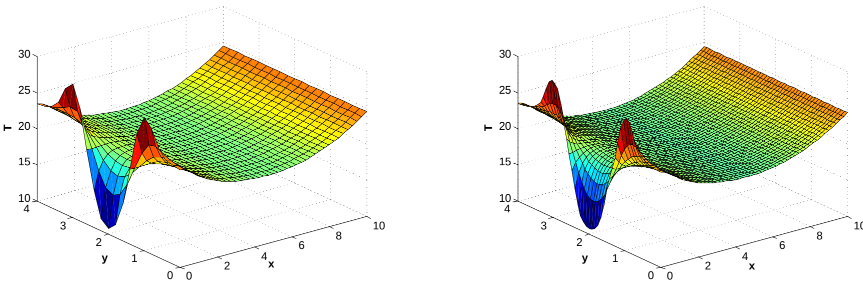
Figure 6: Solutions at Two Values of x

plots of the analytic solution and the same numerical approximation along lines at $x = 0$ (6a) at $x = 2$ (6b). It appears that the approximation for the steady-state solution (at least) is quite good.

Example 2

For our second example we change $\Omega = [0,\ 10] \times [0,\ 4]$ and introduce several 'zones' along the $x = 0$ boundary with the parameters α and β varying in step fashion (see 4). Specifically, we have:

$$\alpha(y) = \begin{cases} 4 & \text{if } 0.8 \le y \le 1.2 \\ 2 & \text{if } 1.6 \le y \le 2.4 \\ 4 & \text{if } 2.8 \le y \le 3.2 \\ 0 & \text{otherwise, and} \end{cases}$$



(a) 41×21 (b) 51×51

Figure 7: Example 2 - Solution

$$\beta(y) = \begin{cases} 35 & \text{if } 0.8 \le y \le 1.2 \\ 35 & \text{if } 2.8 \le y \le 3.2 \\ 0 & \text{otherwise.} \end{cases}$$

On the right boundary we have:

$$k \frac{\partial T}{\partial x}|_{x=10} = 2 \ ,$$

whereas along the upper and lower boundaries we use (2). Figure 7 compares the numerical results on a 41×21 grid and a 51×51 grid

References

[1] J.E. Akin, *Finite Elements for Analysis and Design*, Academic Press, 1994