2.097J / 6.339J / 16.920J

Numerical Methods for Partial Differential Equations

Massachusetts Institute of Technology – Fall 2023

Project 2 – Design of a Thermal Fin Using the Finite Element Method

Version 1.0 *Due: Nov. 22, 2023*

Problem Statement

We consider the problem of designing a thermal fin to effectively remove heat from a surface. The two-dimensional fin, shown in Figure 1, consists of a vertical central "post" and four horizontal "subfins"; the fin conducts heat from a prescribed uniform flux "source" at the root, Γ_{root}, through the large-surface-area subfins to surrounding flowing air.

The fin is characterized by a five-component parameter vector, or "input," $\underline{\mu} = (\mu^1, \mu^2, \ldots, \mu^5)$, where $\mu^i = k^i,\ i = 1, \ldots, 4$, and $\mu^5 = \text{Bi}$; $\underline{\mu}$ may take on any value in a specified design set $\mathcal{D} \subset \mathbb{R}^5$. Here k^i is the thermal conductivity of the i^{th} subfin (normalized relative to the post conductivity $k^0 \equiv 1$); and Bi is the Biot number, a nondimensional heat transfer coefficient reflecting convective transport to the air at the fin surfaces (larger Bi means better heat transfer). For example, suppose we choose a thermal fin with $k^1 = 0.4,\ k^2 = 0.6,\ k^3 = 0.8,\ k^4 = 1.2,$ and $\text{Bi} = 0.1$; for this particular configuration $\underline{\mu} = \{0.4,\ 0.6,\ 0.8,\ 1.2,\ 0.1\}$, which corresponds to a single point in the set of all possible configurations $\mathcal{D}$ (the parameter or design set). The post is of width unity and height four; the subfins are of fixed thickness $t = 0.25$ and length $L = 2.5$.

We are interested in the design of this thermal fin, and we thus need to look at certain outputs or cost-functionals of the temperature as a function of $\underline{\mu}$. We choose for our output T_{root}, the average steady-state temperature of the fin root normalized by the prescribed heat flux into the fin root. The particular output chosen relates directly to the cooling efficiency of the fin — lower values of T_{root} imply better thermal performance.

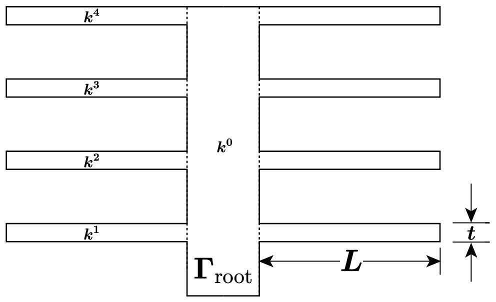

Figure 1: Thermal Fin

The steady–state temperature distribution within the fin, $u(\underline{\mu})$, is governed by the elliptic partial differential equation

$$-k^i\ \nabla^2 u^i = 0 \text{ in } \Omega^i,\ i = 0, \ldots, 4, \tag{1}$$

where ∇^2 is the Laplacian operator, and u^i refers to the restriction of u to Ω^i. Here Ω^i is the region of the fin with conductivity $k^i,\ i = 0, \ldots, 4$: Ω^0 is thus the central post, and $\Omega^i,\ i = 1, \ldots, 4$, corresponds to the four subfins. The entire fin domain is denoted Ω ($\bar{\Omega} = \cup_{i=0}^{4} \bar{\Omega}^i$); the boundary Ω is denoted Γ. We must also ensure continuity of temperature and heat flux at the conductivity–discontinuity interfaces $\Gamma^i_{\text{int}} \equiv \partial\Omega^0 \cap \partial\Omega^i,\ i = 1, \ldots, 4$, where $\partial\Omega^i$ denotes the boundary of Ω^i:

$$\left.\begin{array}{rcl} u^0 & = & u^i \\ -(\nabla u^0 \cdot \hat{\mathbf{n}}^i) & = & -k^i(\nabla u^i \cdot \hat{\mathbf{n}}^i) \end{array}\right\} \text{ on } \Gamma^i_{\text{int}},\ i = 1, \ldots, 4;$$

here $\hat{\mathbf{n}}^i$ is the outward normal on $\partial\Omega^i$. Finally, we introduce a Neumann flux boundary condition on the fin root

$$-(\nabla u^0 \cdot \hat{\mathbf{n}}^0) = -1 \text{ on } \Gamma_{\text{root}}, \tag{2}$$

which models the heat source; and a Robin boundary condition

$$-k^i(\nabla u^i \cdot \hat{\mathbf{n}}^i) = \text{Bi } u^i \text{ on } \Gamma^i_{\text{ext}},\ i = 0, \ldots, 4, \tag{3}$$

which models the convective heat losses. Here Γ^i_{ext} is that part of the boundary of Ω^i exposed to the flowing fluid; note that $\cup_{i=0}^4 \Gamma^i_{\text{ext}} = \Gamma \backslash \Gamma_{\text{root}}$.

The average temperature at the root, $T_{\text{root}}(\underline{\mu})$, can then be expressed as $\ell^O(u(\underline{\mu}))$, where

$$\ell^O(v) = \int_{\Gamma_{\text{root}}} v$$

(recall Γ_{root} is of length unity). Note that $\ell(v) = \ell^O(v)$ for this problem.

Part 1 - Finite Element Approximation

α) Show that $u(\underline{\mu}) \in X \equiv H^1(\Omega)$ satisfies the weak form

$$a(u(\underline{\mu}), v; \underline{\mu}) = \ell(v),\ \forall v \in X, \tag{4}$$

with

$$\begin{aligned} a(w, v; \underline{\mu}) &= \sum_{i=0}^{4} k^i \int_{\Omega^i} \nabla w \cdot \nabla v \, dA + \text{Bi} \int_{\Gamma\backslash\Gamma_{\text{root}}} wv \, dS, \\ \ell(v) &= \int_{\Gamma_{\text{root}}} v \, dS. \end{aligned}$$

β) Show that $u(\underline{\mu})$ is the argument that minimizes

$$J(w) = \frac{1}{2} \sum_{i=0}^{4} k^i \int_{\Omega^i} \nabla w \cdot \nabla w \, dA + \frac{\text{Bi}}{2} \int_{\Gamma\backslash\Gamma_{\text{root}}} w^2 \, dS - \int_{\Gamma_{\text{root}}} w \, dS \tag{5}$$

over all functions w in X.

γ) We now consider the linear finite element space

$$X_h = \{v \in H^1(\Omega) |\ v|_{T_h} \in \mathbb{P}^1(T_h),\ \forall T_h \in \mathcal{T}_h\},$$

and look for $u_h(\underline{\mu}) \in X_h$ such that

$$a(u_h(\underline{\mu}), v; \underline{\mu}) = \ell(v),\ \forall v \in X_h; \tag{6}$$

our output of interest is then given by

$$T_{\text{root}\,h}(\underline{\mu}) = \ell^O(u_h(\underline{\mu})). \tag{7}$$

Applying our usual nodal basis, we arrive at the matrix equations

$$\begin{aligned} \underline{A}_h \, \underline{u}_h(\underline{\mu}) &= \underline{F}_h, \\ T_{\text{root}\,h}(\underline{\mu}) &= (\underline{L}_h)^T \underline{u}_h(\underline{\mu}), \end{aligned}$$

where $\underline{A}_h \in \mathbb{R}^{n\times n}$, $\underline{u}_h \in \mathbb{R}^n$, $\underline{F}_h \in \mathbb{R}^n$, and $\underline{L}_h \in \mathbb{R}^n$; here n is the dimension of the finite element space, which (given our natural boundary conditions) is equal to the number of nodes in $\mathcal{T}_h$.

Derive the elemental matrices $\underline{A}_h^k \in \mathbb{R}^{3\times 3}$, load vectors $\underline{F}_h^k \in \mathbb{R}^3$, and output vectors, $\underline{L}_h^k \in \mathbb{R}^3$, with particular attention to those elements on Γ; and describe the procedure for creating $\underline{A}_h$, $\underline{F}_h$, and $\underline{L}_h$ from these elemental quantities.

δ) Write a finite-element code that takes a configuration $\underline{\mu}$ and a triangulation $\mathcal{T}_h$ (see Appendix 1) and returns $u_h(\underline{\mu})$ and $T_{\text{root}\,h}(\underline{\mu})$. For the particular configuration $\underline{\mu}_0 = \{0.4,\ 0.6,\ 0.8,\ 1.2,\ 0.1\}$ (i) plot the solution $u_h(\underline{\mu}_0)$ (see Appendix 1), and (ii) evaluate the output $T_{\text{root}\,h}(\underline{\mu}_0)$; use $\mathcal{T}_{h_{\text{fine}}}$ for this calculation.

ϵ) Show that

$$T_{\text{root}}(\underline{\mu}) - T_{\text{root}\,h}(\underline{\mu}) = a(e(\underline{\mu}), e(\underline{\mu})), \tag{8}$$

where $e(\underline{\mu}) = u(\underline{\mu}) - u_h(\underline{\mu})$ is the error in the finite element solution. If $u \in H^2(\Omega)$, how would you expect $T_{\text{root}}(\underline{\mu}) - T_{\text{root}\,h}(\underline{\mu})$ to converge as a function of h? In practice, what do you observe? To answer the latter question, take the finest of the three triangulations given ($\mathcal{T}_{h_{\text{fine}}}$) as the "truth," and suppose that

$$\begin{aligned}(T_{\text{root}})_{h_{\text{fine}}} - (T_{\text{root}})_{2h_{\text{fine}}=h_{\text{medium}}} &= C(2h_{\text{fine}})^b\\ (T_{\text{root}})_{h_{\text{fine}}} - (T_{\text{root}})_{4h_{\text{fine}}=h_{\text{coarse}}} &= C(4h_{\text{fine}})^b;\end{aligned}$$

then find b in the obvious fashion.

Part 2 - Reduced-Basis Approximation

In general, the dimension of the finite element space, $\dim X_h = n$, will be quite large (in particular if we were to treat the more realistic three-dimensional fin problem), and thus the solution of $\underline{A}_h \underline{u}_h(\underline{\mu}) = \underline{F}_h$ can be quite expensive. We investigate here an alternative, *reduced-basis* methods, that allow us to accurately and very rapidly predict $T_{\text{root}}(\underline{\mu})$ in the limit of many evaluations — that is, at many different values of $\underline{\mu}$ — which is precisely the "limit of interest" in *design* and *optimization* studies. To derive the reduced-basis approximation we shall exploit the energy principle,

$$u(\underline{\mu}) = \arg\min_{w\in X} J(w),$$

where $J(w)$ is given by (5).

To begin, we introduce a sample in parameter space,

$$S_N = \{\underline{\mu}^1, \underline{\mu}^2, \ldots, \underline{\mu}^N\}$$

with $N \ll n$. Each $\underline{\mu}^i$, $i = 1, \ldots, N$, belongs in the parameter set $\mathcal{D}$. For our parameter set we choose $\mathcal{D} = [0.1, 10.0]^4 \times [0.01, 1.0]$, that is, $0.1 \le k^i \le 10.0$, $i = 1, \ldots, 4$, for the conductivities, and $0.01 \le \text{Bi} \le 1.0$ for the Biot number.

We then introduce the reduced-basis space as

$$W_N = \text{span}\{u_h(\underline{\mu}^1), u_h(\underline{\mu}^2), \ldots, u_h(\underline{\mu}^N)\} \tag{9}$$

where $u_h(\underline{\mu}^i)$ is the finite-element solution for $\underline{\mu} = \underline{\mu}^i$. To simplify the notation, we define $\zeta^i \in X$ as

$$\zeta^i = u_h(\underline{\mu}^i), \quad i = 1, \ldots, N;$$

we can then write $W_N = \text{span}\{\zeta^i,\ i = 1,\ldots,N\}$. Recall that $W_N = \text{span}\{\zeta^i,\ i = 1,\ldots,N\}$ means that W_N consists of all functions in X that can be expressed as a linear combination of the ζ^i; that is, any member v_N of W_N can be represented as

$$v_N = \sum_{j=1}^{N} \beta^j \zeta^j, \tag{10}$$

for some unique choice of $\beta^j \in \mathbb{R},\ j = 1,\ldots,N$. (We implicitly assume that the $\zeta^i,\ i = 1,\ldots,N$, are linearly independent; it follows that W_N is an N-dimensional subspace of X.)

In the reduced-basis approach we look for an approximation $u_N(\underline{\mu})$ to $u_h(\underline{\mu})$ (which for our purposes here we presume is *arbitrarily close* to $u(\underline{\mu})$) in W_N; in particular, we express $u_N(\mu)$ as

$$u_N(\underline{\mu}) = \sum_{j=1}^{N} u_N^j\, \zeta^j; \tag{11}$$

we denote by $\underline{u}_N(\underline{\mu}) \in \mathbb{R}^N$ the coefficient vector $(u_N^1,\ldots,u_N^N)^T$. The premise — or hope — is that we should be able to accurately represent the solution at some new point in parameter space, $\underline{\mu}$, as an appropriate linear combination of *solutions previously computed* at a small number of points in parameter space (the $\underline{\mu}^i,\ i = 1,\ldots,N$). But how do we find this appropriate linear combination? And how good is it? And how do we compute our approximation efficiently?

The energy principle is crucial here (though more generally the weak form would suffice). To wit, we apply the classical Rayleigh-Ritz procedure to define

$$u_N(\underline{\mu}) = \arg \min_{w_N \in W_N} J(w_N); \tag{12}$$

alternatively we can apply Galerkin projection to obtain the equivalent statement

$$a(u_N(\underline{\mu}), v; \underline{\mu}) = \ell(v),\ \forall v \in W_N. \tag{13}$$

The output can then be calculated from

$$T_{\text{root}\,N}(\underline{\mu}) = \ell^O(u_N(\underline{\mu})). \tag{14}$$

We now study this approximation in more detail.

α) Prove that, in the energy norm $|\!|\!| \cdot |\!|\!| \equiv (a(\cdot,\cdot))^{1/2}$,

$$|\!|\!| u(\underline{\mu}) - u_N(\underline{\mu}) |\!|\!| \le |\!|\!| u(\underline{\mu}) - w_N |\!|\!|,\ \forall w_N \in W_N. \tag{15}$$

This inequality indicates that out of all the possible choices of w_N in the space W_N, the reduced-basis method defined above will choose the "best one" (in the energy norm). Equivalently, we can say that even if we knew the solution $u(\underline{\mu})$, we would not be able to find a better approximation to $u(\underline{\mu})$ in W_N — *in the energy norm* — than $u_N(\underline{\mu})$.

β) Show that $\underline{u}_N(\underline{\mu})$ as defined in (11–13) satisfies a set of $N \times N$ linear equations,

$$\underline{A}_N(\underline{\mu})\underline{u}_N(\underline{\mu}) = \underline{F}_N;$$

and that

$$T_{\text{root}\,N}(\underline{\mu}) = \underline{L}_N^T\, \underline{u}_N(\underline{\mu}).$$

Give expressions for $\underline{A}_N(\underline{\mu}) \in \mathbb{R}^{N\times N}$ in terms of $\underline{A}_h(\underline{\mu})$ and $\underline{Z}$, $\underline{F}_N \in \mathbb{R}^N$ in terms of $\underline{F}_h$ and $\underline{Z}$, and $\underline{L}_N \in \mathbb{R}^N$ in terms of $\underline{L}_h$ and $\underline{Z}$; here $\underline{Z}$ is an $n \times N$ matrix, the j^{th} column of which is $\underline{u}_h(\underline{\mu}^j)$ (the nodal values of $u_h(\underline{\mu}^j)$).

γ) Show that the bilinear form $a(w, v; \underline{\mu})$ can be decomposed as

$$a(w, v; \underline{\mu}) = \sum_{q=1}^{Q} \sigma^q(\underline{\mu}) a^q(w, v), \ \forall w, v \in X, \ \forall \underline{\mu} \in \mathcal{D}, \tag{16}$$

for $Q = 6$; and give expressions for the $\sigma^q(\underline{\mu})$ and the $a^q(w, v)$. Notice that the $a^q(w, v)$ are not dependent on $\underline{\mu}$; the parameter dependence enters only through the functions $\sigma^q(\underline{\mu}), \ q = 1, \ldots, Q$. Further show that

$$\underline{A}_h(\underline{\mu}) = \sum_{q=1}^{Q} \sigma^q(\underline{\mu}) \underline{A}_h^q,$$

and

$$\underline{A}_N(\underline{\mu}) = \sum_{q=1}^{Q} \sigma^q(\underline{\mu}) \underline{A}_N^q. \tag{17}$$

Give an expression for the $\underline{A}_h^q$ in terms of the nodal basis functions; and develop a formula for the $\underline{A}_N^q$ in terms of the $\underline{A}_h^q$ and $\underline{Z}$.

δ) Implement an off-line/ on-line version of the reduced-basis approximation following the computational decomposition indicated below.

- *Off-line*

  1. Choose N.

  2. Choose the sample S_N.

  3. Construct $\underline{Z}$.

  4. Construct $\underline{A}_N^q, \ q = 1, \ldots, Q$; $\underline{F}_N$; and $\underline{L}_N$.

- *On-line*

  1. Form $\underline{A}_N(\underline{\mu})$ from (17).

  2. Solve $A_N(\underline{\mu}) \underline{u}_N(\underline{\mu}) = \underline{F}_N$.

  3. Evaluate the output $T_{\text{root}\, N}(\underline{\mu})$ from (14).

The idea is that the off-line stage is done only once, generating a small datafile with the $\underline{A}_N^q, \ q = 1, \ldots, Q$, $\underline{F}_N$, and $\underline{L}_N$; the on-line stage then accesses this datafile to provide real-time response to new $\underline{\mu}$ queries. For the required off-line finite element calculations in this and the following questions, you should use the medium triangulation $\mathcal{T}_{h_{\text{medium}}}$.

For $N = 10$, and the sample set S_N given in the datafile `sn.dat` (available on the course website), verify that for $\underline{\mu}_0 = \{0.4, \ 0.6, \ 0.8, \ 1.2, \ 0.1\}$ the value of the output is $T_{\text{root}\, N}(\underline{\mu}_0) = 1.7291$. For $\underline{\mu}_1 = \{1.8, \ 4.2, \ 5.7, \ 2.9, \ 0.3\}$ what is the value that you obtain for the output $T_{\text{root}\, N}(\underline{\mu}_1)$?

ϵ) Consider a thermal fin with specified $\{k_1, k_2, k_3, k_4\} = \{0.4, 0.6, 0.8, 1.2\}$ and the Biot number varying from $[0.1, 10]$. The Biot number is directly related to the cooling method; higher cooling rates (higher Bi) imply lower (better) T_{root} but also higher (worse) initial and operational costs. We can thus define (say) a total cost function as

$$C(\text{Bi}) = 0.2\,\text{Bi} + T_{\text{root}}(\text{Bi}), \tag{18}$$

minimization of which yields an optimal solution. Apply your (on-line) reduced-basis approximation for $T_{\text{root}\,N}$ (that is, replace $T_{\text{root}}(\text{Bi})$ in (18) with $T_{\text{root}\,N}(\text{Bi})$) to find the optimal Bi. Any (simple) optimization procedure suffices for the minimization.

Appendix 1 - Finite Element Method Implementation

For the implementation of the finite element method, three possible triangulations of the fin problem are provided. To obtain the triangulation data, download from the course web site, the file `grids.mat`. To download and save this file to your hard drive right-click on the link of this file and choose *Save as...*. To load then the triangulation data in the MATLAB workspace:

```
>> load grids
```

This creates three variables named `coarse, medium,` and `fine`. Each of these variables is a different triangulation $\mathcal{T}_h$ for the fin problem. More specifically

- `coarse` defines $\mathcal{T}_{h_{\text{coarse}}}$, with 1333 nodes, and 2095 elements.
- `medium` defines $\mathcal{T}_{h_{\text{medium}}}$, with 4760 nodes, and 8380 elements, and
- `fine` defines $\mathcal{T}_{h_{\text{fine}}}$, with 17889 nodes, and 33520 elements.

Each of these variables is of type struct, with four different fields.

```
>> coarse
coarse =
      nodes:  1333
      coor:  [1333x2 double]
      elements:  2095
      theta:  1x7 cell
```

Description of the fields: (assume that we are using the coarse triangulation)

- `nodes:` The number of nodes in the triangulation.
- `coor:` Two-dimensional matrix with size (nodes×2), where each row i has the x and y coordinates for node i. For example, the location of node 49 can be determined by two coordinates. The coordinate in the x-direction would be `coarse.coor(49,1)` and in the y-direction `coarse.coor(49,2)`.
- `elements:` The number of elements in the triangulation.
- `theta:` The adjacency matrix $\theta(k,\alpha)$ which defines the local-to-global mapping required in the elemental assembly procedure. Since we have regions with different physical properties, for each region a seperate adjacency matrix is provided. The regions considered are
  - Region 1: Domain Ω^1, $\theta^1(k,\alpha) =$`coarse.theta{1}`,
  - Region 2: Domain Ω^2, $\theta^2(k,\alpha) =$`coarse.theta{2}`,
  - Region 3: Domain Ω^3, $\theta^3(k,\alpha) =$`coarse.theta{3}`,

- Region 4: Domain Ω^4, $\theta^4(k, \alpha) =$`coarse.theta{4}`,
- Region 5: Domain Ω^0, $\theta^5(k, \alpha) =$`coarse.theta{5}`.

For each of these regions i, the index k varies in the range $k \in \{1, \dots, n_i\}$, where n_i are the number of elements in region i. For example element 12 in region 3 is has the global nodes ν_1=`coarse.theta{3}(12,1)`, ν_2=`coarse.theta{3}(12,2)`, and ν_3=`coarse.theta{3}(12,3)`.

In addition, for the treatment of the boundary conditions, the boundary is divided into two sections. The first is $\Gamma \backslash \Gamma_{\text{root}}$, where Robin boundary conditions are applied; the second is Γ_{root}, where the incoming heat flux is applied. For each segment in these sections, the associated global nodes are provided.

- Section 1: $\Gamma \backslash \Gamma_{\text{root}}$, $\kappa^1(m, \alpha) =$`coarse.theta{6}`,
- Section 2: Γ_{root}, $\kappa^2(m, \alpha) =$`coarse.theta{7}`.

For each of the sections i, the index m varies in the range $m \in \{1, \dots, s_i\}$, where the s_i are the number of segments in section i. As an example, to find the nodes ν_1, and ν_2 for segment 5 in the first section, we would use $\nu_1 =$`coarse.theta{6}(5,1)`, and $\nu_2 =$`coarse.theta{6}(5,2)`.

To plot the temperature distribution, `plotsolution.m` can be used (available on the course web site). If $z \equiv \underline{u}_h$ is the vector with the computed temperature values for each of the nodes, then a contour plot of the temperature distribution can be obtained by

```
>> plotsolution(coarse, z)
```

The first argument is the mesh used in the calculation of z, and the second is the solution vector z.

For the storage of the finite element matrices, use MATLAB's sparse matrix data structure. Also, for the solution of the resulting linear systems, use the default solution methods provided in MATLAB.

Appendix 2 - Python Tips

A Python skeleton code is also provided for this project. Here are a few tips that you should read before getting started.

- This code package uses the Python library `PyVista` (https://docs.pyvista.org/index.html) for plotting. You will need to install the module before producing plots - `pip install pyvista`.
- A script `driver.py` has been provided. This function has examples of what calls to the FEM solver and reduced basis solver look like.
- Functions to abstract away the plotting and data loading procedures are provided in the `utils.py` file. See `driver.py` for how to use them.
- Meshes are stored in a dictionary data structure and contain the same information as their MATLAB counterparts. Use `dict_instance.keys()` to check which fields exist in a dictionary.
- Extracting a submatrix from a larger matrix (for example, the 3x3 matrix corresponding to the elemental support extracted from the matrix A) is slightly different with `numpy` than with MATLAB. You will need to use "advanced indexing" for this, which you can read about here: https://numpy.org/doc/stable/reference/arrays.indexing.html. Indexing in `numpy` is very powerful!

- Sparse matrices and vectors (A and F) are dealt with using the `scipy.sparse` package. Make sure you have `scipy` installed (`pip install scipy`). With this package, matrices are created and edited with an array type called `lil_matrix`, but matrix operations are performed on an array of type `csc_matrix` - this is a design feature of `scipy`. Thus, the array type must be converted from `lil_matrix` to `csc_matrix` before doing matrix math, which can be done using `csc_matrix_instance = lil_matrix_instance.asformat('csc')`

- WARNING: Always use the `scipy` sparse linear algebra functions for performing matrix operations with sparse matrices. Using regular `np.dot` for matrix multiplication, for example, will lead to unexpected results because `numpy` does not know how to handle the sparse matrix data type. Thus, `scipy.sparse` has its own linear algebra library, accessed through `scipy.sparse.linalg`. These functions return dense arrays.

- NOTE: These sparse linear algebra functions need at least one input array to be sparse. For example, you can perform matrix multiplication with a sparse and a dense matrix, or two sparse matrices. However they will encounter issues with both matrices dense, for example.