Rectangular Meshes in $\mathbb{R}^d$ and Usage in WG

March 20, 2013

Let m be a rectangular mesh in $\mathbb{R}^d$, let its vertex of minimum coordinates be $(a_1, ..., a_d)$, the number of elements along the axes $(k_1, ..., k_d)$, and the dimensions of the individual mesh elements $(h_1, ..., h_d)$.

1 Interiors

In general an interior in the mesh is of the form

$$I(c_1, ..., c_d) = \prod_{i=1}^{d} (a_i + (c_i - 1)h_i,\ a_i + c_i h_i)$$

where the c_i are integers and $1 \leq c_i \leq k_i$.

We call the sequence $(c_i)_{i=1...,d}$ the mesh coordinates of the interior, and also of its closure rectangle which is a finite element of the mesh:

$$E(c_1, ..., c_d) = \prod_{i=1}^{d} [a_i + (c_i - 1)h_i,\ a_i + c_i h_i]$$

1.1 Enumeration of Interiors

The element interiors are enumerated with the lower mesh coordinate numbers sweeping their ranges before those of higher number, as follows:

Seq #	Interior
1	$I(1, \ldots, 1)$
2	$I(2, 1, \ldots, 1)$
	$\ldots$
k_1	$I(k_1, 1, \ldots, 1)$
$k_1 + 1$	$I(1, 2, \ldots, 1)$
$k_1 + 2$	$I(2, 2, \ldots, 1)$
	$\ldots$
$k_1 k_2 \cdots k_d$	$I(k_1, k_2, \ldots, k_d)$

1.2 Mesh Coordinates from Sequence Number

Let n be an interior number with interiors enumerated as above. We want to find the mesh coordinates for the given interior number. The reverse transformation will be described in the next section.

We start by observing that the d^{th}mesh coordinate starts at 1 and is incremented after every block of $k_1 k_2 \cdots k_{d-1}$ enumerated interiors. Thus it follows that the d^{th}mesh coordinate is given by

$$\pi_d(n) = (n-1) \setminus (k_1 k_2 \cdots k_{d-1}) + 1$$

where $\setminus$ represents integer division (e.g. $2 \setminus 3 = 0$, and $3 \setminus 3 = 1$).

Since the lower numbered coordinates are reset when the d^{th} coordinate changes, finding coordinate $\pi_{d-1}(n)$ for interior n amounts to applying the same reasoning as the above, this time on the block-relative interior number ((n-1) $\bmod (k_1 k_2 \cdots k_{d-1})$) +1 in the block of constant d coordinate interiors containing the n^{th} interior. Thus

$$\pi_{d-1}(n) = ((n-1)\, mod\, (k_1 k_2 \cdots k_{d-1})) \setminus (k_1 k_2 \cdots k_{d-2}) + 1$$

More generally, for any $r < d$, (n-1) $\bmod (k_1 k_2 \cdots k_r)$ is the relative index (0-based) of interior n within the block of constant coordinate $r+1$ in which it lies. Since the coordinates $1, \ldots, r$ were reset to ones at the beginning of this block, the r^{th} coordinate index (0-based form of the r^{th} coordinate) is therefore the number of times that the product of the lesser dimensions $k_1 k_2 \cdots k_{r-1}$ divides evenly into the block-relative index. Thus we have:

$$\pi_r(n) = ((n-1)\, mod\, (k_1 k_2 \cdots k_r)) \setminus (k_1 k_2 \cdots k_{r-1}) \quad + 1 \qquad (r = 1, ..., d)$$

Here we've used the fact that (n-1) $\bmod (k_1 k_2 \cdots k_d)$=(n-1) to justify including the original case $r = d$, and we've let an empty product signify 1 for the $r = 1$ case.

1.3 Interior Sequence Numbers from Mesh Coordinates

The interior number can also be recovered from the mesh coordinates via the equation:

$$i_\sharp(c_1, \ldots, c_d) = 1 + \sum_{i=1}^{d}(c_i - 1)\prod_{l=1}^{i-1} k_l$$

, the empty product again signifying 1.

This is easily proved by induction on the number of leading coordinates which we allow to be different from 1.

2 Non-Boundary Sides

In this section "side" will mean a side of a single finite element that is not a subset of the outside boundary of the mesh. All of these sides are enumerated as a single sequence by the mesh in the general abstract mesh interface which all concrete mesh types must implement. This way users of the mesh modules will not have to distingutiple multiple types of sides when interacting with meshes, which concerns should be hidden inside the mesh implementations.

Even so, inside the rectangular mesh implementation our strategy will be to form sections in the overall enumeration for the sides, with sides grouped by their orientations with respect to the coordinate axes. Separating sides by orientation this way makes it easier to reason about the enumeration and resulting geometric positions of sides, and how they relate geometrically to mesh elements. This is because for rectangular meshes the set of all sides having a common orientation form a mesh very much like the overall mesh of finite elements.

In general a non-boundary side perpendicular to axis j is of the form:

$$\begin{aligned} S(c_1, ..., c_d) = {} & [a_1 + (c_1 - 1)h_1,\, a_1 + c_1 h_1] \times \\ & [a_2 + (c_2 - 1)h_2,\, a_2 + c_2 h_2)] \times \\ & ... \times \\ & \{\mathbf{a_j} + \mathbf{c_j h_j}\} \times \\ & ... \times \\ & [a_d + (c_d - 1)h_d,\, a_d + c_d h_d] \\ & \quad \text{where } 1 \le c_i \le k_i \text{ for } i \ne j \text{ and } 1 \le c_j \le k_j - 1 \end{aligned}$$

the j^{th} factor being the singleton set and all other factors being the closed intervals as in the definition of $E(c_1, ..., c_d)$.

As with interiors, we call the tuple $(c_1, ..., c_d)$ the mesh coordinates of the side in the mesh particular to the side's orientation. The mesh is particular to the side's orientation because of the short j^{th} dimension, resulting from the fact that side S(c_1,... ,$c_{j-1,}$,k_j, c_{j+1},... ,c_d) (extending our definition of S for a moment) is on the outside boudary of the mesh. This follows from the fact that it is easy to construct points arbitrarily close to such a side which are not within the mesh's overall rectangle, by approaching $a_j + k_j h_j$ from above in coordinate j.

We introduce the notation $k_{j,i}$ to represent the size of the i^{th} dimension of mesh coordinates in the mesh of sides perpendicular to axis j:

$$sdims_j = (k_{j,1}, ..., k_{j,d}), \text{ where } k_{j,i} = \begin{cases} k_i, & i \neq j \\ k_j - 1, & i = j \end{cases}$$

As an illustration of side orientation-dependent meshes, let c and r be the number of columns and rows of finite elements in a rectangular mesh in $\mathbb{R}^2$(see figure below). In the bottom left diagram, note how for the sides perpendicular to axis 1, which are the vertical interior sides, there are only $c - 1$ of these sides in a row, i.e. along axis 1, because the outside boundary sides are not included. We see that this lesser dimension in comparison to the finite element interiors mesh only occurs in the coordinate direction perpendicular to the sides: there are r rows of the vertical sides just as in the main (interiors) mesh. Likewise for the sides perpendicular to the second axis, there are only $r - 1$ of these sides along coordinate 2 when the other coordinates (just the column in this case) are fixed, but there are the full number of columns.

```
interiors
----------------------
|...             | rc |
        ...
----------------------
| c+1 | c+2 |...| 2c |
----------------------
|  1  |  2  |...|  c |
----------------------
(^also fe numbering)


sides perp. to axis 1           sides perp. to axis 2
--------------------------      --------------------------
|...           | r(c-1)|  |     |     |   ...   |         |
           ...                  --------------------------
--------------------------      |...              | (r-1)c^ |
|  c|  c+1|...| 2(c-1)|  |              ...
--------------------------      --------------------------
|  1|    2|...|  (c-1)|  |      | 1^  |   ...   |    c^   |
--------------------------      --------------------------
                                (^add #sides perp to axis 1)
```

2.1 Enumeration of Sides

We can now enumerate the sides perpendicular to any axis j in the same way that the interiors were enumerated, only using $k_{j,i}$ in place of k_i. Then by putting these separate enumerations together into a single enumeration of all non-boundary sides, we have the following division points marking the beginnings of the sections of sides having the same orientation:

$$\begin{aligned}
&s_1 = 1\\
&s_2 = 1 + (k_1 - 1)\, k_2 \cdots k_d\\
&s_3 = s_2 + k_1(k_2 - 1)\, k_3 \cdots k_d\\
&...\\
&s_j = s_{j-1} + \prod_{i=1}^{d} k_{j,i}\\
&...\\
&s_d = s_{d-1} + k_1 \cdots k_{d-1}(k_d - 1)
\end{aligned}$$

The total number of sides of all orientations is $\sum_{j=1}^{d} \prod_{i=1}^{d} k_{j,i}$.

2.2 Side Mesh Coordinates from Sequence Number

To find the mesh coordinates for a side number n, we first have to find the axis $a(n)$ to which the side is perpendicular, given by:

$$a(n) = \begin{cases} 1, & s_1 \le n < s_2\\ ... & \\ j, & s_j \le n < s_{j+1}\\ ... & \\ d, & s_d \le n \le \sum_{j=1}^{d} \prod_{i=1}^{d} k_{j,i} \end{cases}$$

This function allows us to identify the mesh consisting of sides of the same orientation that we will use to locate the side geometrically. Recognizing that n-$s_{a(n)}$ is the 0-based index of the side relative to its particular mesh, like n-1 is in the discussion of the interiors mesh, we can now proceed just as in section 1.2 to determine the side's coordinates in the mesh of sides having the same orientation, obtaining:

$$\pi_r(n) = ((n - s_{a(n)})\, mod\, (\prod_{i=1}^{r} k_{a(n),i})) \setminus (\prod_{i=1}^{r-1} k_{a(n),i}) + 1 \qquad (r = 1, ..., d)$$

The coordinates obtained here are specific to the particular mesh for the orientation of side n. These coordinates will be related to the general finite element coordinates below in section 2.4.

2.3 Side Sequence Numbers from Side Mesh Coordinates

Given a side known to be perpendicular to axis j, we can find its sequence number in the overall enumeration of sides via:

$$s_{\sharp,j}(c_1,\ldots,c_d) = s_j + \sum_{i=1}^{d}(c_i - 1)\prod_{l=1}^{i-1} k_{j,l}$$

This result is easily proved by induction on the number of leading coordinates allowed to be different from 1.

Note that j must be known in order to recover the sequence number from the coordinates.

2.4 Main Mesh Elements Including a Side

We would like to relate a side's orientation-specific side mesh coordinates with those of the main mesh, by finding which finite elements the given side is between; that is, which elements include the side as a subset.

It is apparant from the representation for a general side shown in section 2 that there are always exactly two mesh element rectangles that include a given side as subsets. These have the same intervals as the side in factors other than j, and in factor j have the closed interval ending at factor j's singleton value for one, and the interval starting at factor j's singleton value for the other. For a side s=S(c_1,...,c_d) these including mesh rectangles are:

$$r_1 = [a_1 + (c_1 - 1)h_1,\, a_1 + c_1h_1] \times ... \times [\mathbf{a_j + (c_j - 1)h_j},\, \mathbf{a_j + c_jh_j})] \times ... \times [a_d + (c_d - 1)h_d,\, a_d + c_dh_d]$$
$$r_2 = [a_1 + (c_1 - 1)h_1,\, a_1 + c_1h_1] \times ... \times [\mathbf{a_j + c_jh_j},\, \mathbf{a_j + (c_j + 1)h_j}] \times ... \times [a_d + (c_d - 1)h_d,\, a_d + c_dh_d]$$

We can see that r_1 above is just the mesh element with coordinates (c_1,...,c_d) in the main finite elements/interiors mesh, in other words E(c_1,...,c_d). Likewise r_2 is just E(c_1,..., c_{j-1}, c_j+1, c_{j+1},. . . , c_d).

Summarizing, we have obtained:

For any axis j, $1 \le j \le d$, and any side $s = S(c_1, ..., c_d)$ perpendicular to axis j, there are exactly two finite elements which include side s as a subset, which are:

- $E(c_1, ..., c_d)$, and
- $E(c_1, ..., c_{j-1}, c_j + 1, c_{j+1}, \ldots, c_d)$.

3 Usage in the WG Method

So far we've only discussed the enumeration and geometry of the parts of the mesh itself, without regard to the formation of a basis of weak functions on these mesh elements. A basis for the WG approximation space in general assigns multiple polynomials or monomials to each side and interior in the mesh. The basis elements like the mesh elements must be enumerated, and we must be able to relate the enumeration to the supporting mesh elements on which the basis elements obtain their polynomial value. This is so that the program can answer fundamental questions such as whether two basis elements are supported on a common finite element, and if so also on which particular pieces (interior or a particular side) of the common finite element they are supported.

Thankfully it's possible to thoroughly separate the concerns of the mesh from those of the basis which uses the mesh, and this has been a major goal of the design of the program. This way the meshes themselves only have to deal with the minimal geometric concerns and are easier to implement and verify for correctness. Likewise there is a basis module, desribed below in 3.3, for handling the enumeration of basis elements and the querying of information about them, using any abstract mesh which implements a small set of standard generic functions. This leaves the concrete mesh implementations themselves only needing to implement this small set of generic geometry-related functions for their specific mesh types in order to automatically be usable by the other modules making up the WG method.

3.1 Abstract Mesh Interface

The functions which every mesh implementation must implement are defined for abstract meshes (subtypes M of AbstractMesh) in Mesh.jl, and they are:

- `num_fes(mesh::M)`

  Gives the number of finite elements (or element interiors) in the mesh.

- `num_nb_sides(mesh::M)`

  Returns "non-boundary" sides: the number of element sides which are not subsets of the outside boundary.

- `fe_inclusions_of_nb_side(i::SideNum, mesh::M)`

  Returns a structure containing information about which two finite elements include the passed non-boundary side, and which face (top, right, etc) the side represents within each of these two finite elements. The callers are not expected to recognize the possible values for the faces returned (since these are mesh specific), but may pass them subsequently into an integration function defined for the mesh, such as the integral_on_ref_fe_side_vs_outward_normal function below.

-
  ```
  integral_on_ref_fe_interior(mon::Monomial, mesh::M)
  ```

  Integrates the monomial on the interior of the reference finite elmenent.

-
  ```
  integral_on_ref_fe_side_vs_outward_normal(vm::VectorMonomial,
                                            face::Face,
                                            mesh::M)
  ```

  Integrates the vector monomial vs outward normal on the indicated face of the reference element.

-
  ```
  integral_on_fe_interior(f::Function, fe::FENum, mesh::M)
  ```

  Integrate an arbitrary function on a finite element interior.

3.2 Example Mesh Function Implementation

We'll provide a brief sketch of the implentation of a crucial mesh function listed above,

```
fe_inclusions_of_nb_side(i::SideNum, mesh::M)
```

to show how the results reached in previous sections can be used in practice. The job of this function is to return the finite elements (by interior number) that include a given side in a mesh, and also the particular part (top, right, etc) that the side occupies in each of the returned finite elements. Note that no such function is necessary for interiors, because interiors only ever intersect a single finite element which are numbered identically to the interior.

The first step is to employ section 2.2 to convert the side sequence number into side mesh coordinates, relative to the mesh specific to the side's orientation. We then can use section 2.4 to find the main mesh coordinates of the two finite elements which include the side. The callers know nothing of mesh coordinates returned by this function (nor necessarily that the mesh being employed is even rectangular), so we must convert the mesh coordinates to finite element/interior numbers, using section 1.3. Finally, we pair each of the returned finite element rectangle numbers with a tag indicating the particular part (top, right, etc) in the rectangle that the side occupies. The face is indicated by an axis number to which the face is perpendicular and whether it is the near or far face from the origin along the indicated axis. The caller is not expected know the meanings of these returned face codes, but can pass them back into the other mesh functions which do understand them, e.g. to a function to compute an integral on a face. This completes the example.

3.3 Basis Implementation

Since the mesh interfaces and implentatiions do not consider at all the overlay of polynomials on the mesh parts as basis elements, we will sketch how such a basis has been implemented for an arbitrary mesh. We will limit ourselves to using the mesh's abstract interface as described in 3.1, which makes no assumptions about the mesh other than the implicit assumptions that it consists of polygonal shapes (because they have faces) and that it is regular so that a reference element may be used for efficiency, though the WG method itself requires additional assumptions about the shapes employed. In particular the mesh need not be rectangular.

Our purpose is to provide, for a given mesh, polynomial degree, and domain dimension, an object representing a basis and its enumeration for the WG approximation space V_h^0, together with functions to interrogate the basis to determine the supporting finite element, face and monomial for any given basis element by its sequence number in the enumeration.

A basis element for V_h^0 in this implementation will be a weak function which is a monomial on one face of a single finite element, excluding sides on the outside boundary of the mesh, and which is 0 on all other faces. On interiors of the finite elements the maximum polynomial degree will be k for some arbitrary $k > 1$, and $k - 1$ on finite element sides.

3.3.1 Enumeration of Basis Elements

We'll first compute the number of basis elements needed, and then assign meaning to the sequence numbers by their locations in the sequence.

There are $\binom{d+l-1}{l}$ monomials of any given degree l and dimension d, which is the multiset coefficient of d and l. Thus we need $\nu_1 = \sum_{l=0}^{k} \binom{d+l-1}{l}$ monomial basis elements supported on each interior to span polynomials of degree k defined there, and likewise $\nu_2 = \sum_{l=0}^{k} \binom{d+l-2}{l-1}$ monomials supported on each side. Let m_{fe} be the count of mesh interiors or finite elements provided by the mesh via function `num_fes`, and let m_s be the count of mesh non-boundary sides as provided by mesh function `num_nb_sides`. Then the counts of interior-supported basis elements b_{int}, side-supported basis elements b_s, and total basis functions b_{tot} are

$$\begin{aligned}
b_{int} &= m_{fe}\,\nu_1 &&= m_{fe} \sum_{l=0}^{k} \binom{d+l-1}{l} \\
b_s &= m_s\,\nu_2 &&= m_s \sum_{l=0}^{k} \binom{d+l-2}{l-1} \\
b_{tot} &= b_{int} + b_s
\end{aligned}$$

We assign numbers to our basis elements as sugested by the two terms above, with all interior-supported basis elements enumerated first, as $1\ldots, b_{int}$, followed by those supported on sides, as $b_{int}+1,\ldots,b_{tot}$. Within each of these two groups the basis elements are assigned in blocks of ν_1 and ν_2 to interior numbers and side numbers respectively as defined by the mesh. Within a block of basis elements allocated to a particular interior or side, the monomials representing the basis element values on the face are arranged first in blocks by increasing degree, then within a degree block lexicographically by increasing exponent, so e.g. x^0 y^2 appears prior to x^1 y^1. Together with the mesh which determines the number and meaning of enumerated interiors and sides, these rules completely order our basis elements.

3.3.2 Basis Interface Functions

Using the above enumeration, we can now implement the functions that the basis exposes to the rest of the program, as functions of the basis element numbers.

Two basic basis related functions are `is_interior_supported` and `is_side_supported`. These are easiy implemented as simple range checks, with basis elements numbered at b_{int} and below being interior supported, and those of higher number being side supported.

After determining whether a basis element is an interior or side, we can now ask for the mesh interior or side on which the basis element is supported, with functions `support_interior_num` and `support_side_num`. The values of these functions can then be used in mesh interface functions, and so are the keys to access the mesh geometry for the supporting finite element faces of the basis elements.

To implement `support_interior_num` to find the interior number $i_{\#}$ for a given basis element number b, we only need to find how many times the number of interior monomials per interior, ν_1, divides evenly into the 0-based index of our basis element $b-1$, then adding 1 to obtain a 1-based sequence number:

$$i_{\#} = (b-1) \setminus \nu_1 + 1$$

The `support_side_num` is implemented in a similar way, only this time using the side-relative basis index $b-(b_{int}+1)$:

$$s_{\#} = (b-(b_{int}+1)) \setminus \nu_2 + 1$$

The final basis function to be implemented is `support_face_monomial`, which gives the monomial defined on the supporting face (interior or side) of the supporting finite element. The block of monomials for each interior or side are ordered in a standard way as described in 3.3.1, so we only need to find the basis number's offset relative to the beginning of the block to identify the monomial. Thus the monomial number is given by

$$m_{\#} = \begin{cases} (b-1)\, mod\, \nu_1 + 1, & b \le b_{int} \\ (b-(b_{int}+1))\, mod\, \nu_2 + 1, & b \ge b_{int}+1 \end{cases}$$

This completes the basis implementation. Thus we've shown how we can completely implement our basis, using only the minimal abstract interface for our mesh.