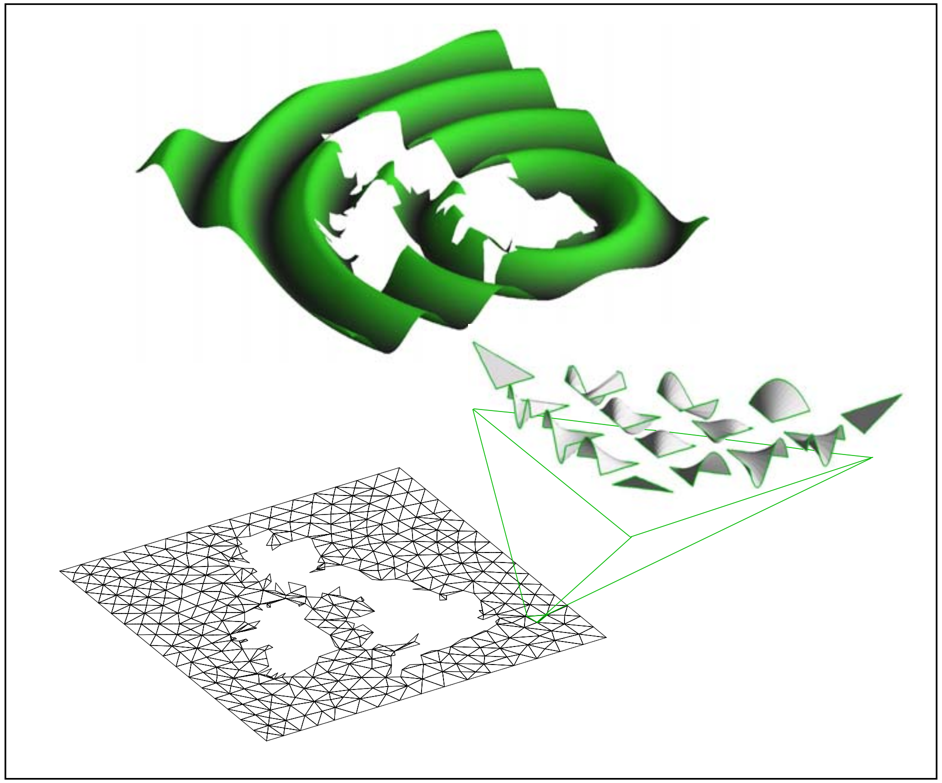
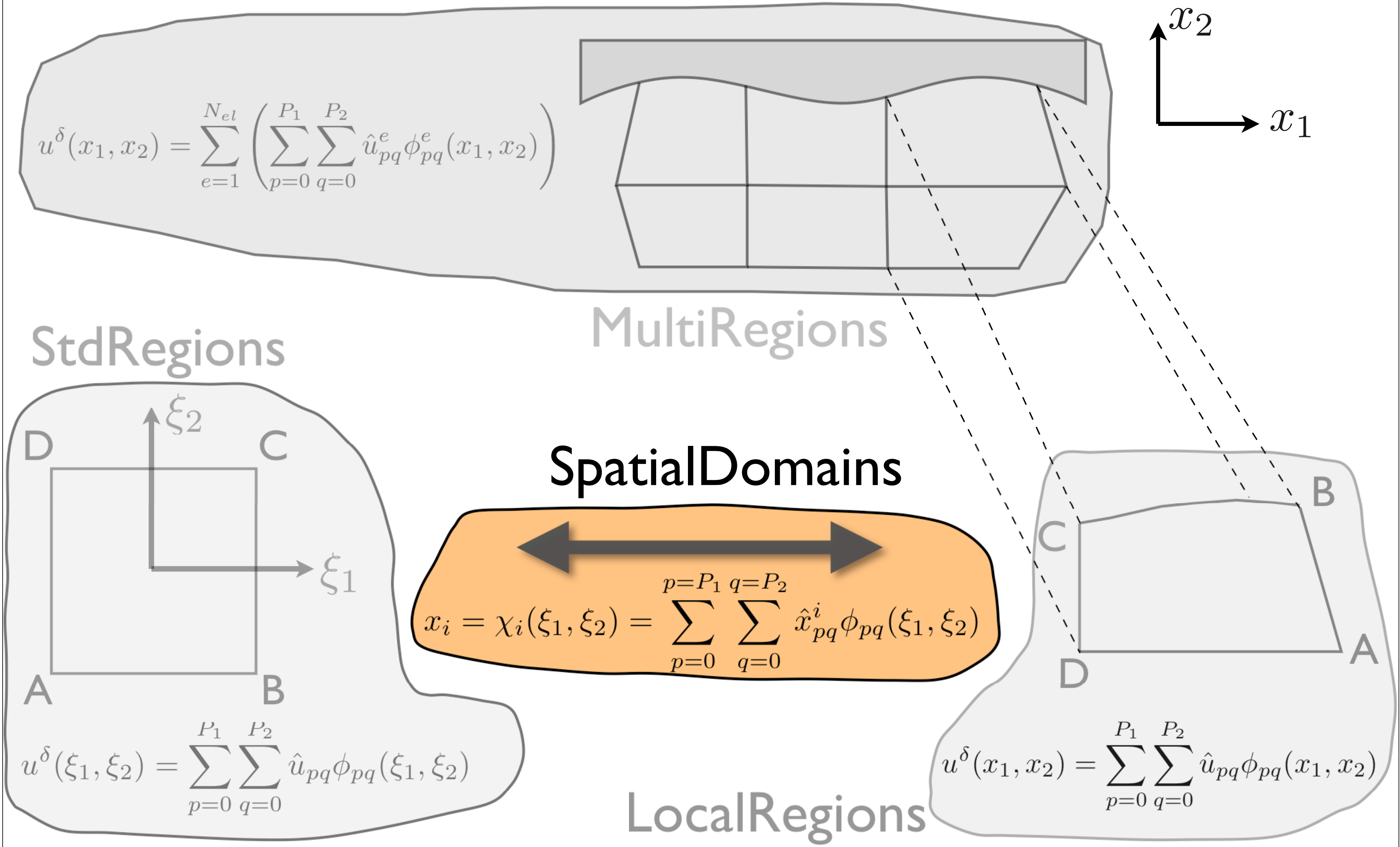
Spatial Construction of Elements



The big picture



Nektar++

Outline

- I/O Issues
- Geometric Construction of Elements
    - Segment Geometry (SpatialDomains::SegGeom)
    - Quadrilateral Geometry (SpatialDomains::QuadGeom)
    - Triangle Geometry (SpatialDomains::TriGeom)
- Metric Construction

Approximating the Geometry

$$\text{Find } u \text{ such that } \mathcal{L}(u) = f \text{ on } \Omega$$

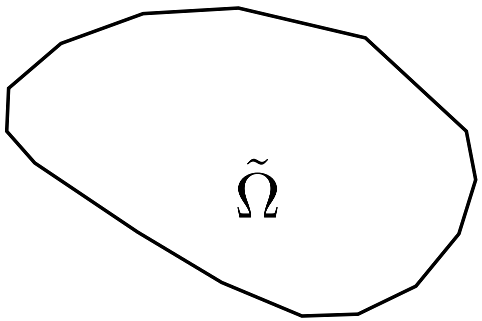
subject to boundary and initial conditions

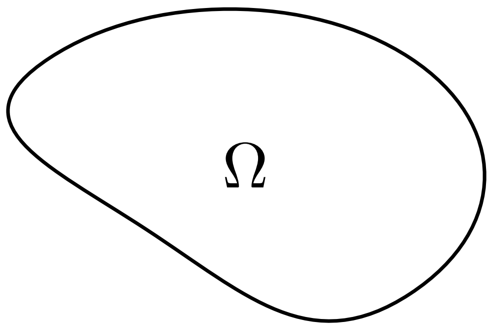
$$\text{Form } \tilde{\Omega} = \mathcal{T}(\Omega)$$

Approximation of Domain

Tessellation of Domain

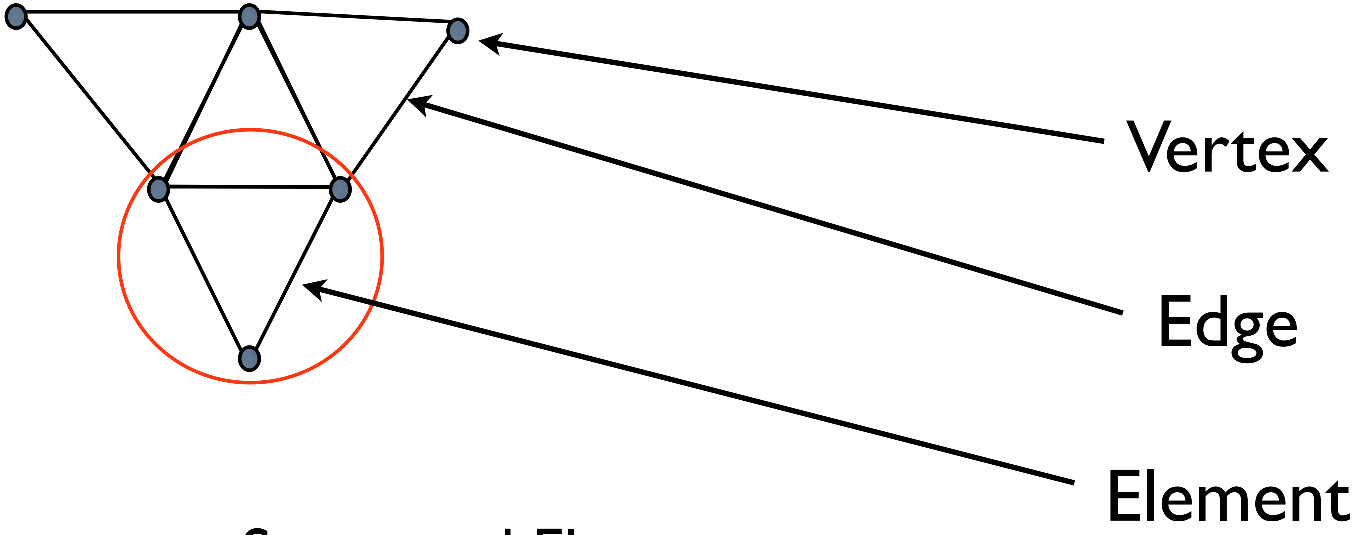
True Domain





Nomenclature

Mesh - a collection of vertices, edges and faces which form elements



Supported Elements:
1D: Segments
2D: Triangles and Quadrilaterals
3D: Tetrahedra, Hexahedra, Prisms and Pyramids

I/O: Header Information

```
<?xml version="1.0" encoding="utf-8"?>

<NEKTAR>
<!-- Embed a 1-dimensional object in a 2-dimensional space -->
<!-- DIM <= SPACE -->
<!-- This provides a method of optimizing code for a 1-D curve embedded in 3-space.
-->
<GEOMETRY DIM="2" SPACE="2">
```

Example: A one-parameter curve in three dimensions

$$x(t) = f_1(t)$$
$$y(t) = f_2(t)$$
$$z(t) = f_3(t)$$

I/O: Definitions

```
<!-- Definitions that can be used below in this file. -->
<DEF>
  <D> A = 1.0 </D>
  <D> B = 2.0 </D>
  <D> C = 3.0 </D>
</DEF>
```

Example: Setting Boundary Conditions

$$u(x) = Asin(x) + Bcos(x) + Cx$$

I/O: Vertices

```xml
<VERTEX>
  <!-- Always must have four values per entry. -->
  <V ID="0">  -1.000000000000000  3.500000000000000   0.0 </V>
  <V ID="1">  -1.000000000000000  0.500000000000000   0.0 </V>
  <V ID="2">  -1.000000000000000  2.500000000000000   0.0 </V>
  <V ID="3">  -1.000000000000000  1.500000000000000   0.0 </V>
  <V ID="4">   3.800000000000000  4.500000000000000   0.0 </V>
  <V ID="5">   0.200000000000000  4.500000000000000   0.0 </V>
  <V ID="6">   2.900000000000000  4.500000000000000   0.0 </V>
  <V ID="7">   2.000000000000000  4.500000000000000   0.0 </V>
  <V ID="8">   1.100000000000000  4.500000000000000   0.0 </V>
  <V ID="9">  5.000000000000000  0.500000000000000   0.0 </V>
  <V ID="10">  5.000000000000000  3.500000000000000   0.0 </V>
  <V ID="11">  5.000000000000000  1.500000000000000   0.0 </V>
  <V ID="12">  5.000000000000000  2.500000000000000   0.0 </V>
  <V ID="13">  0.200000000000000 -0.500000000000000   0.0 </V>
  <V ID="14">  3.800000000000000 -0.500000000000000   0.0 </V>
  <V ID="15">  1.100000000000000 -0.500000000000000   0.0 </V>
  <V ID="16">  2.000000000000000 -0.500000000000000   0.0 </V>
  <V ID="17">  2.900000000000000 -0.500000000000000   0.0 </V>
  <V ID="18"> -0.400000000000000  4.000000000000000   0.0 </V>
  <V ID="19">  4.400000000000000  4.000000000000000   0.0 </V>
```

I/O: Edges

```
<EDGE NUMBER="96">
  <E ID="0"> 1 21 </E>
  <E ID="1"> 13 21 </E>
  <E ID="2"> 13 15 </E>
  <E ID="3"> 15 16 </E>
  <E ID="4"> 16 17 </E>
  <E ID="5"> 14 17 </E>
  <E ID="6"> 1 3 </E>
  <E ID="7"> 1 29 </E>
  <E ID="8"> 21 29 </E>
  <E ID="9"> 21 30 </E>
  <E ID="10"> 13 30 </E>
  <E ID="11"> 15 30 </E>
  <E ID="12"> 15 27 </E>
  <E ID="13"> 16 27 </E>
  <E ID="14"> 16 25 </E>
  <E ID="15"> 17 25 </E>
  <E ID="16"> 3 29 </E>
  <E ID="17"> 29 30 </E>
  <E ID="18"> 27 30 </E>
  <E ID="19"> 25 27 </E>
  <E ID="20"> 2 3 </E>
```

Edge 0 consists of (Vertex=0,Vertex=21)

Edge 10 consists of (Vertex=13,Vertex=30)

I/O: Elements

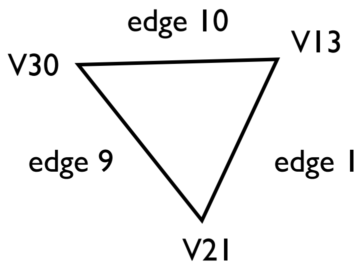
```
<!-- Q - quads, T - triangles, S - segments, E - tet, P - pyramid, R - prism, H - hex -->
<!-- Only certain element types are appropriate for the given dimension (dim on mesh) -->
<!-- Can also use faces to define 3-D elements.  Specify with F[1] for face 1, for example. -->
<ELEMENT>
  <T ID="0"> 6 7 16 </T>
  <T ID="1"> 0 8 7 </T>
  <T ID="2"> 9 17 8 </T>
  <T ID="3"> 1 10 9 </T>
  <T ID="4"> 2 11 10 </T>
  <T ID="5"> 11 12 18 </T>
  <T ID="6"> 3 13 12 </T>
  <T ID="7"> 14 19 13 </T>
  <T ID="8"> 4 15 14 </T>
  <T ID="9"> 5 39 15 </T>
```

Element 3 is a Triangle consists of

Edge = 1 (vertex 13 and 21)

Edge = 10 (vertex 13 and 30)

Edge = 9 (vertex 21 and 30)



I/O: Composite and Domain

Composite: A collection of objects

```
<COMPOSITE>
  <C ID="0"> T[0-21] </C>
  <C ID="1"> Q[0-25] </C>
  <C ID="2"> E[0-1] </C>
  <C ID="3"> E[2-5] </C>
```

Collection of Quads 0-25

Collection of Edges 2-5

```
  <DOMAIN> C[0-1] </DOMAIN>
</GEOMETRY>
<EXPANSIONS>
 <E COMPOSITE="C[0]" NUMMODES="7" TYPE="MODIFIED" />
 <E COMPOSITE="C[1]" NUMMODES="7" TYPE="MODIFIED" />
</EXPANSIONS>
```

Domain is Composites 0 and 1

I/O: Parameters and Boundaries

Parameters

```
<CONDITIONS>

    <!-- Removed redundancy since we can specify any level of granularity
     in the ExpansionTypes section below.-->
    <PARAMETERS>
      <P> Lambda    = 1 </P>
    </PARAMETERS>
```

Solution Variable

```
    <!--One of these for each dimension. These are the vector
    components, say, s = (u,v); comprised of two components in
    this example for a 2D dimension.-->
    <VARIABLES>
      <V ID="0"> u </V>
    </VARIABLES>
```

Boundary Regions (where boundary conditions are to be applied)

```
    <!--These composites must be defined in the geometry file.-->
    <BOUNDARYREGIONS>
      <B ID="0"> C[2] </B>
      <B ID="1"> C[3] </B>
      <B ID="2"> C[4] </B>
      <B ID="3"> C[5] </B>
      <B ID="4"> C[6] </B>
```

World Space versus Reference Space

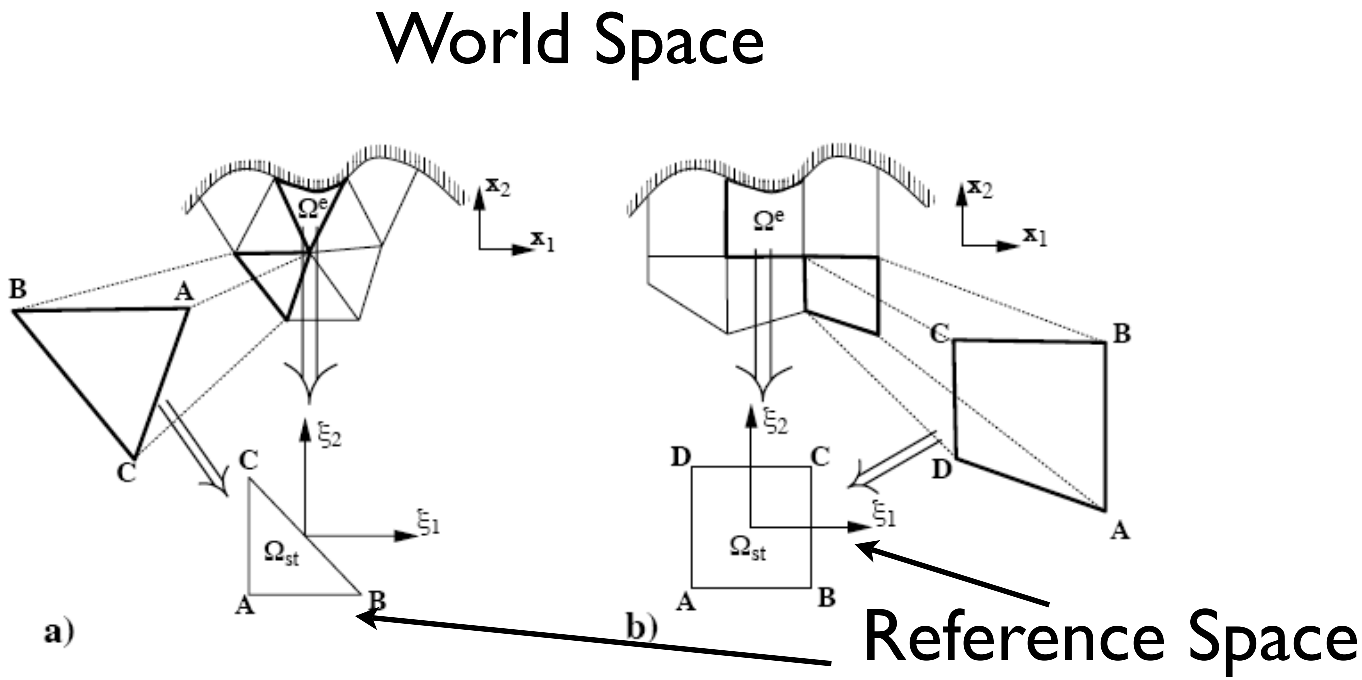
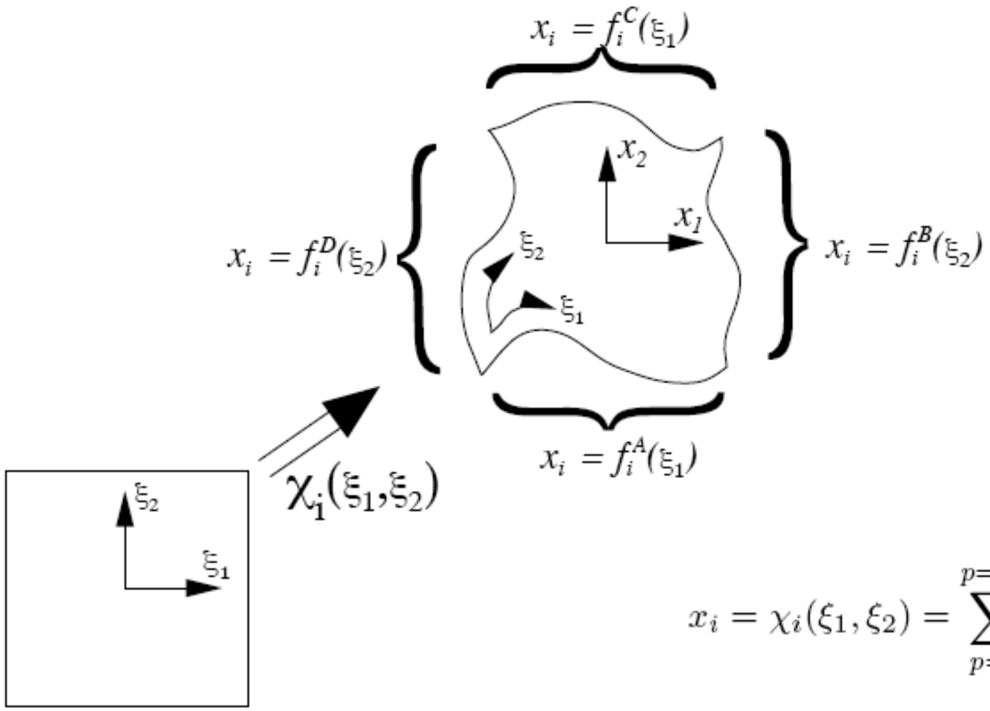


Figure 3.2 To construct a C^0 expansion from multiple elements of specified shapes (for example, triangles or rectangles), each elemental region Ω^e is mapped to a standard region Ω_{st} in which all local operations are evaluated.

Course Notes: Section 3.1.3.1

Mapping from reference space to world space



$$x_i = \chi_i(\xi_1, \xi_2) = \sum_{p=0}^{p=P_1} \sum_{q=0}^{q=P_2} \hat{x}^i_{pq} \phi_{pq}(\xi_1, \xi_2)$$

Figure 3.4 A general curved element can be described in terms of a series of parametric functions $f^A(\xi_1), f^B(\xi_2), f^C(\xi_1)$, and $f^D(\xi_2)$. Representing these functions as a discrete expansion we can construct an iso-parametric mapping $\chi_i(\xi_1, \xi_2)$ relating the standard region (ξ_1, ξ_2) to the deformed region (x_1, x_2).

Course Notes: Section 3.1.3.2

Metric Terms

Recall from Calculus:

$$\int_a^b f(x)dx \longleftrightarrow \int_{-1}^{1} f(\psi^{-1}(\xi)) \left| \frac{(b-a)}{2} \right| d\xi$$

$$x = \psi(\xi) = \frac{(b-a)}{2}(\xi + 1) + a$$

Jacobian
of the mapping

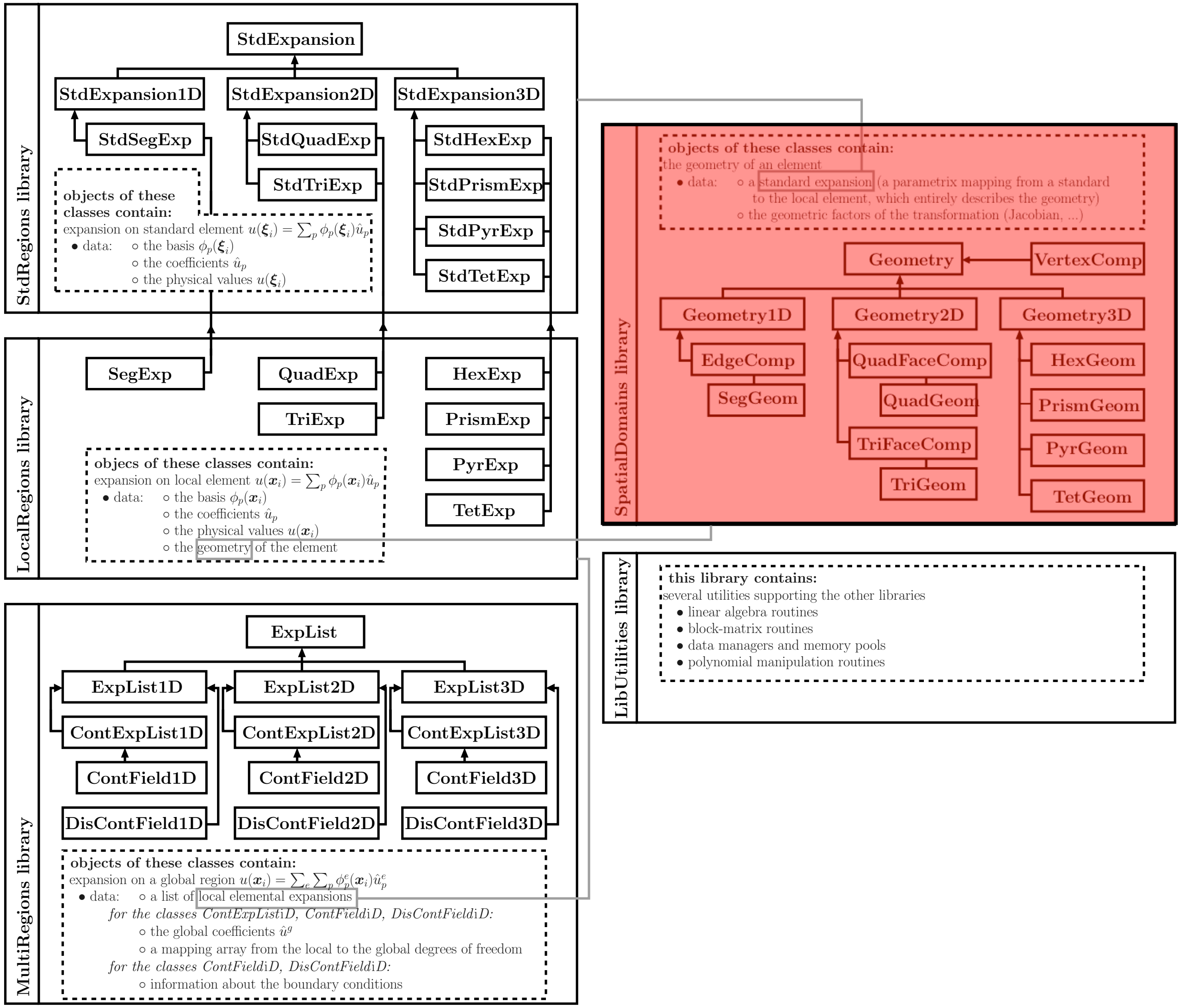
Chain Rule:

$$\begin{aligned} \text{Given } f(x) &= H(\psi^{-1}(x)) \\ \frac{d}{dx} f(x) &= \frac{d}{dx} H(\psi^{-1}(x)) \\ &= \left[\frac{d}{dx} \psi^{-1}(x) \right] \frac{dH(\xi)}{d\xi} \end{aligned}$$

Impact of the
mapping

Course Notes: Section 3.1

Nektar++ code

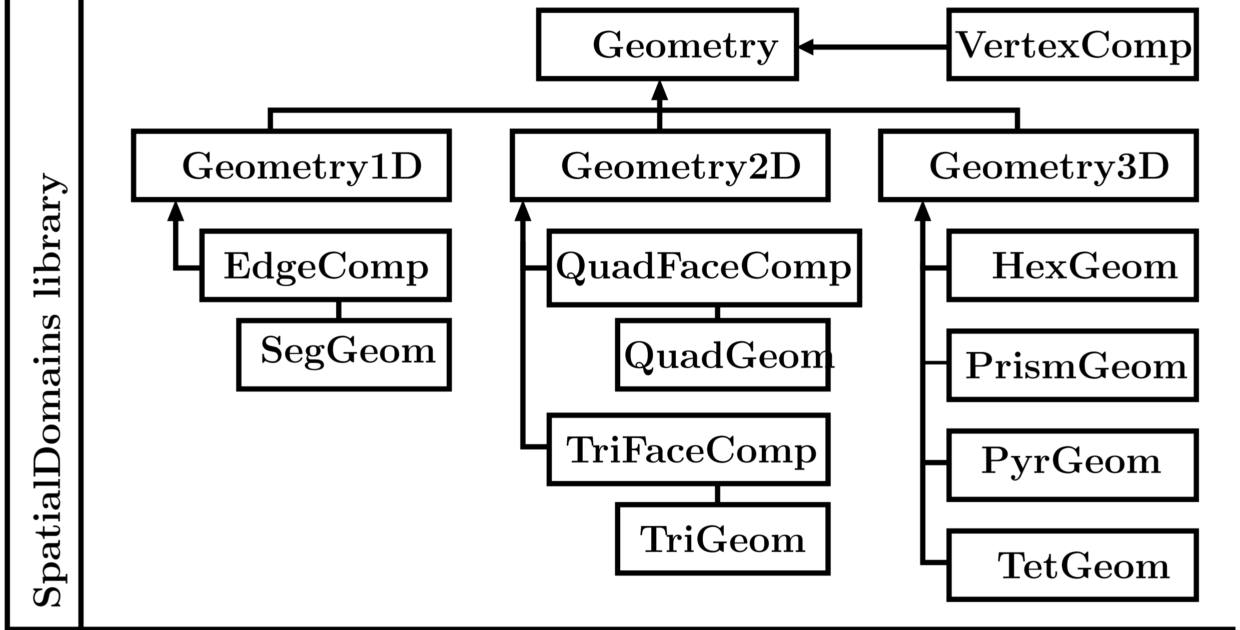


StdRegions library
StdExpansion
StdExpansion1D
StdExpansion2D
StdExpansion3D
StdSegExp
StdQuadExp
StdTriExp
StdHexExp
StdPrismExp
StdPyrExp
StdTetExp
objects of these classes contain:
expansion on standard element $u(\boldsymbol{\xi}_i) = \sum_p \phi_p(\boldsymbol{\xi}_i)\hat{u}_p$
• data: ∘ the basis $\phi_p(\boldsymbol{\xi}_i)$
∘ the coefficients $\hat{u}_p$
∘ the physical values $u(\boldsymbol{\xi}_i)$
LocalRegions library
SegExp
QuadExp
TriExp
HexExp
PrismExp
PyrExp
TetExp
objecs of these classes contain:
expansion on local element $u(\boldsymbol{x}_i) = \sum_p \phi_p(\boldsymbol{x}_i)\hat{u}_p$
• data: ∘ the basis $\phi_p(\boldsymbol{x}_i)$
∘ the coefficients $\hat{u}_p$
∘ the physical values $u(\boldsymbol{x}_i)$
∘ the geometry of the element
MultiRegions library
ExpList
ExpList1D
ExpList2D
ExpList3D
ContExpList1D
ContExpList2D
ContExpList3D
ContField1D
ContField2D
ContField3D
DisContField1D
DisContField2D
DisContField3D
objects of these classes contain:
expansion on a global region $u(\boldsymbol{x}_i) = \sum_e \sum_p \phi_p^e(\boldsymbol{x}_i)\hat{u}_p^e$
• data: ∘ a list of local elemental expansions
for the classes ContExpListiD, ContFieldiD, DisContFieldiD:
∘ the global coefficients $\hat{u}^g$
∘ a mapping array from the local to the global degrees of freedom
for the classes ContFieldiD, DisContFieldiD:
∘ information about the boundary conditions
SpatialDomains library
objects of these classes contain:
the geometry of an element
• data: ∘ a standard expansion (a parametrix mapping from a standard to the local element, which entirely describes the geometry)
∘ the geometric factors of the transformation (Jacobian, ...)
Geometry
VertexComp
Geometry1D
Geometry2D
Geometry3D
EdgeComp
SegGeom
QuadFaceComp
QuadGeom
TriFaceComp
TriGeom
HexGeom
PrismGeom
PyrGeom
TetGeom
LibUtilities library
this library contains:
several utilities supporting the other libraries
• linear algebra routines
• block-matrix routines
• data managers and memory pools
• polynomial manipulation routines

Nektar++ code

objects of these classes contain:
the geometry of an element

- data:
  - a standard expansion (a parametrix mapping from a standard to the local element, which entirely describes the geometry)
  - the geometric factors of the transformation (Jacobian, ...)



brary

this library contains:
several utilities supporting the other libraries