XDDM Expressions:
Parameter Scope and Syntax

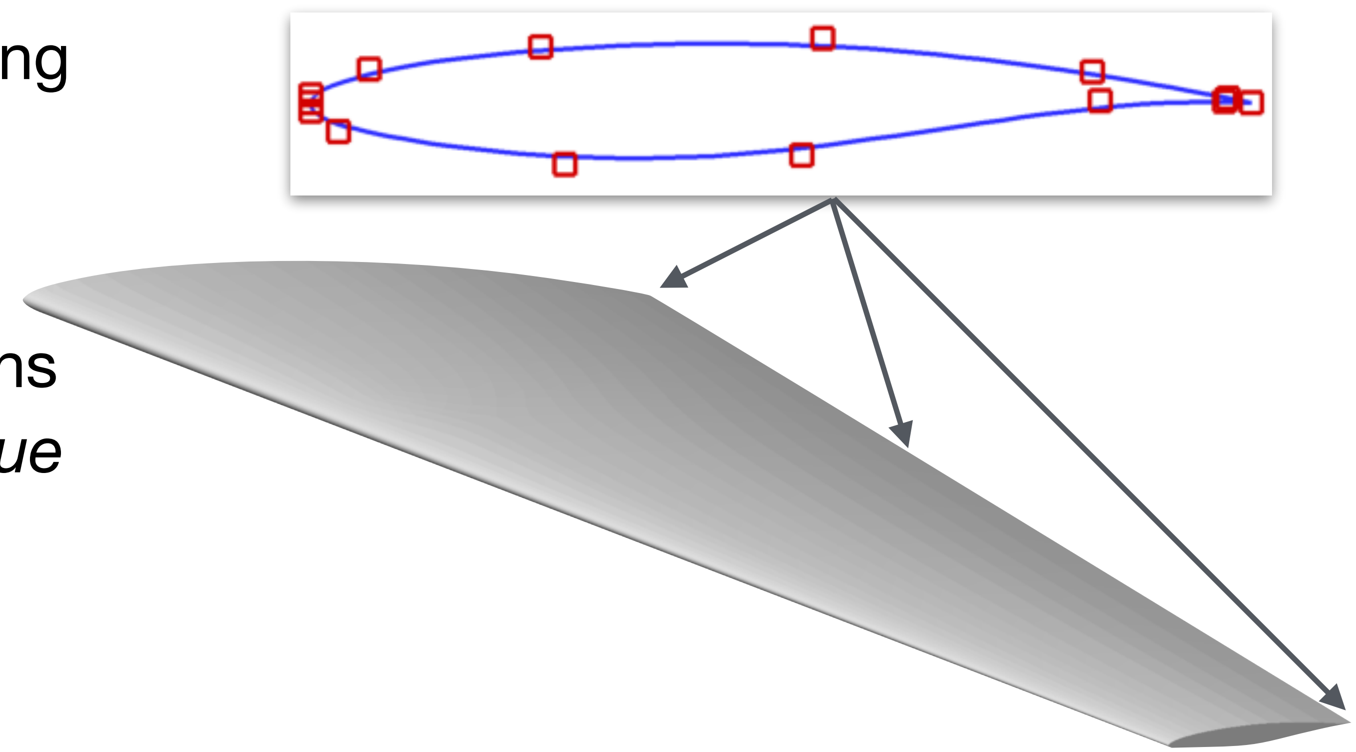
Supplement to XDDM documentation in doc/xddm/xddm.html

Version 1.0

Variables and Constants

- To setup a straightforward geometry parameterization file, flat list design variables by using unique *ID* attributes
  - Modeler parses and interprets the *ID's*
  - Snippet on right requires the modeler to parse "sec1_p1" and interpret that as something like "design variable controlling spline control point 1 of section 1 of the wing"
- Changing ***Variable*** to ***Constant*** simply turns off the design variable but still uses its *Value* attribute when building the model

```
<Model ID="wing">
    <Variable ID="sec1_p1" .../>
    <Constant ID="sec1_p2" .../>
    <Variable ID="sec2_p1" .../>
</Model>
```



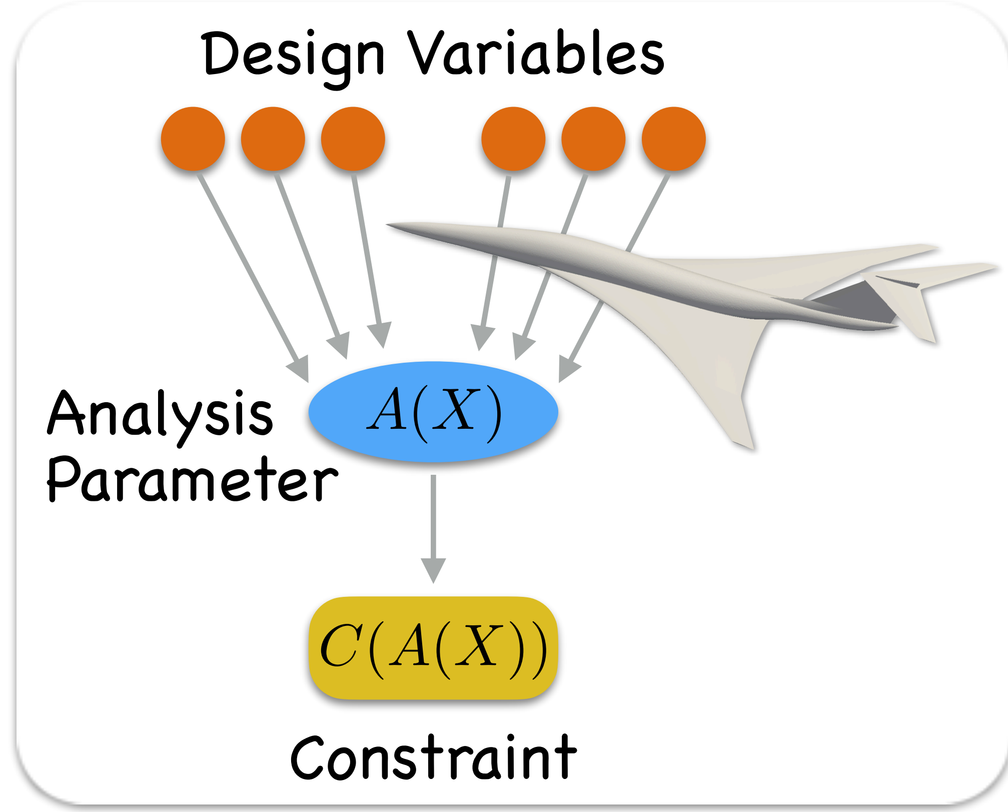
Signatures

- Faster problem setup if the XML structure reflects the parameterization hierarchy
  - Example: parameterizations of wing sections reuse the same airfoil parameterization
    - Convenient to use the same design variable *ID's* but associate them with different wing sections, as shown to the right
- Design variable *ID* attributes no longer unique
- XDDM constructs unique names (signatures) by concatenating element names with *ID's* with a double underscore
- Snippet on right defines three design variables
  - Model__wing__Section__Root__Variable__1
  - Model__wing__Section__Tip__Variable__1
  - Model__wing__Variable__1

```
<Model ID="wing">
    <Section ID="Root">
        <Variable ID="1" .../>
    </Section>
    <Section ID="Tip">
        <Variable ID="1" .../>
    </Section>
    <Variable ID="1" .../>
</Model>
```

Analysis Parameters

- ***Analysis*** elements are the driven outputs, such as lift, drag, airfoil area and wing volume
  - They depend on design variables
  - They become objectives and constraints
- To use ***Analysis*** values in optimizations, reference their *ID's* in the *Expr* attribute of ***Function, Constraint*** and ***Objective*** elements
  - More generally, the *ID* of any XDDM element can be used in the *Expr* attribute

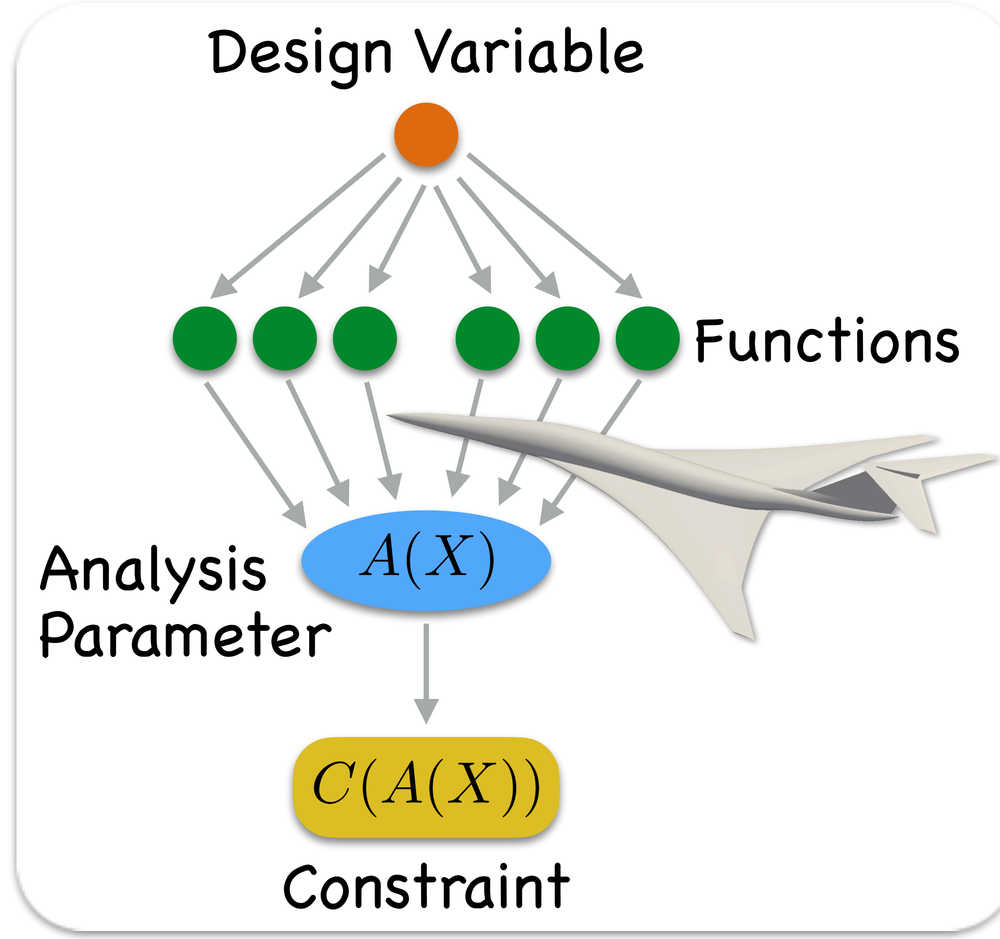



```
<Model ID="wing">
    <Variable ID="sec1_p1" …/>
    <Variable ID="AR" …/>
    <Analysis ID="Vol" …/>
    <Constraint ID="C" Expr="Vol/AR" …/>
</Model>
```

Functions

- Frequently, model parametrization is good but not perfect for a particular optimization problem
  - Example: one may wish to link model parameters together and control them with a new design variable
- Use the ***Function*** element to link variables with other XDDM elements
  - Allows augmenting the model parameterization to better fit a particular optimization without changing the model
  - More broadly, allows propagation of functionals throughout the framework
  - Requires modeler to parse ***Function*** elements

"Functions driven by Functions from above"




```
<Model ID="wing">
    <Variable ID="t" …/>
    <Function ID="r" Expr="cos(t)-sin(t)"/>
    <ControlPoint ID="1">
        <Function ID="x" Expr="r+1"/>
    </ControlPoint>
    <ControlPoint ID="2">
        <Function ID="x" Expr="r+2"/>
    </ControlPoint> …
```

Rules and Regulations

- What can be included in the ***Expr*** attribute of ***Functions***?
  - *ID* tags of ***Constants***, ***Variables***, ***Analysis*** elements and other ***Functions***
  - Scope of ***Variable***, ***Constant*** and ***Analysis*** elements is only within the *parent* element
- ***Function** ID's must be globally unique when referenced in expressions*
- Expressions support the constant *PI* and *EULER*
- Supported operators listed on right (number of operands and their function in parentheses)
- Expression parameters must begin with a letter and may involve letters, numbers and underscores
  - To use other characters, enclose the parameter in braces

Expr = "{1.1} + {2.1}"

Parameters with ID's "1.1" and "2.1"

+	=> sum (2)
-	=> difference (2)
*	=> product (2)
/	=> division (2)
log	=> logarithm (2: base, function)
^	=> exponentiation (2: base, exponent)
neg	=> unary minus (1)
sin	=> sine (1)
cos	=> cosine (1)
tan	=> tangent (1)
cot	=> cotangent (1)
asin	=> arc sine (1)
acos	=> arc cosine (1)
atan	=> arc tangent (1)
atan2	=> arc tangent of y/x (2: y, x)
acot	=> arc cotangent (1)
sinh	=> hyperbolic sine (1)
cosh	=> hyperbolic cosine (1)
asinh	=> hyperbolic area sine (1)
acosh	=> hyperbolic area cosine (1)

Example A

Local scope of Constant, Variable and Analysis Parameters

```
<Example>
   <DesignPoint ID="1">          ← Parent element of F1
      <Variable ID="alpha"/>
      <Analysis ID="CL"/>
      <Analysis ID="CM"/>
      <Function ID="F1" Expr="CM/CL"/>
   </DesignPoint>

   <DesignPoint ID="2">          ← Parent element of F2
      <Analysis ID="CD"/>
      <Function ID="F2" Expr="(CD-0.001)^2"/>
   </DesignPoint>

   <Function ID="F3" Expr="F1 + F2"/>
</Example>
```

Globally unique ID's

Parameters of F1 are local to DesignPoint "1"

Parameter of F2 is local to DesignPoint "2"

Example B

Function *ID's* must be Globally Unique when Referenced in Expressions

```
<Model ID="wing">               Parent of Function A
    <Section ID="1">
        <Variable ID="1"…/>
    </Section>
    <Section ID="2">
        <Variable ID="1"…/>
    </Section>
    <Function ID="A" Expr="{1}*{1}"/>
</Model>
```

Use braces to express parameters that don't satisfy the naming rules

```
<Model ID="wing">               Parent of function C
    <Section ID="1">
        <Variable ID="1" …/>
        <Function ID="C" Expr="{1}"/>
    </Section>
    <Section ID="2">            Parent of function B
        <Variable ID="1" …/>
        <Function ID="B" Expr="{1}"/>
    </Section>
    <Function ID="A" Expr="B*C"/>
</Model>
```

Wrong, variables {1} in Function A are not unique

OK, B and C are unique so Function A can be evaluated

*Note: The typing of "extra" functions around **Analysis** and **Variable** parameters may seem redundant. In practice, however, **Analysis** and **Variable** ID's are frequently restricted to certain keywords while **Functions** are under user control. This makes the propagation of functional values and sensitivities uniform and less error prone.*

Example C

Local References to Variables — Global References to Functions

```
<Variable ID="x" .../>
<Function ID="F1" Expr="x*x"/>
<A>
  <Function ID="F2" Expr="F1*F3*F4"/>
  <B>
    <Variable ID="x" .../>
    <Function ID="F3" Expr="x*x"/>
  </B>
  <C>
    <Variable ID="x" .../>
    <Analysis ID="y" .../>
    <Function ID="F4" Expr="x*y*F3"/>
  </C>
</A>
```

- Order of evaluation is determined automatically
- ***Functions*** F1 and F3 depend only on their local ***Variables***
- Since function *ID's* are unique, the *Expr* tags can reference any other function
  - F4 depends not only on its own ***Variable*** and ***Analysis*** parameter, but also on F3 and hence the design variable in element B
  - F2 is a function of all three design variables via its dependence on F1, F3 & F4

Example D

Using Functions as Subroutines

```
<Root>
    <Variable ID=“x” .../>
    <Function ID=“F1”  Expr=“x*t”/>
    <A>
        <B>
            <Constant ID=“t”  Value=“1”/>
            <Function ID=“F2” Expr=“F1+2”/>
        </B>
```

“x*t+2”

```
        <C>
            <Constant ID=“t”  Value=“2”/>
            <Function ID=“F3” Expr=“F1*F2”/>
        </C>
```

“(x*t)(x*t+2)”

```
    </A>
</Root>
```

- Constant “t” may be different in each element (B, C, etc.), such as in a parametric spline vector
- F1 cannot be explicitly evaluated since there is no definition of “t” at the Root level
- F2 and F3 can be evaluated. While “t” does not explicitly appear in their Expr’s, its value from elements B and C is used to evaluate F1 to in turn evaluate F2 and F3
- Not tested for case when “t” is a Variable

Objectives and Constraints

- Keywords ***Objective*** and ***Constraint*** are similar to ***Function***, except they are parsed by the optimizer interface
  - Typical gradient-based optimization problem consists of one objective and several constraints

```
<Optimize>
 <Objective ID="CDmin" Expr="CD1 + CD2"/>
   <DesignPoint ID="1">
      <Analysis ID="CL"/>
      <Analysis ID="CD"/>
      <Constraint ID="CL1" Expr="CL" Min="0.9"/>
      <Function ID="CD1" Expr="CD"/>
   </DesignPoint>
   <DesignPoint ID="2">
      <Analysis ID="CL"/>
      <Analysis ID="CD"/>
      <Constraint ID="CL2" Expr="CL" Min="0.4"/>
      <Function ID="CD2" Expr="CD"/>
   </DesignPoint>
```

Minimize drag subject to lift constraints

Assign unique ID

Assign unique ID

External Parameters

- Consider an optimization that requires importing a parameter from a part file into the objective function defined in design.xml
  - For example, the part file may contain the weight of an airplane as an analysis parameter and we wish to multiply by L/D in design.xml

design.xml

part.xml

design.xml

```
<Optimize>
    <DesignPoint ID="1">
        <Analysis ID="CL".../>
        <Analysis ID="CD".../>
        <Objective ID="Range" Expr="-CL/CD*log(2,File{wing.xml}__weight)"/>
    </DesignPoint>
</Optimize>
```

Function "weight" is in wing.xml

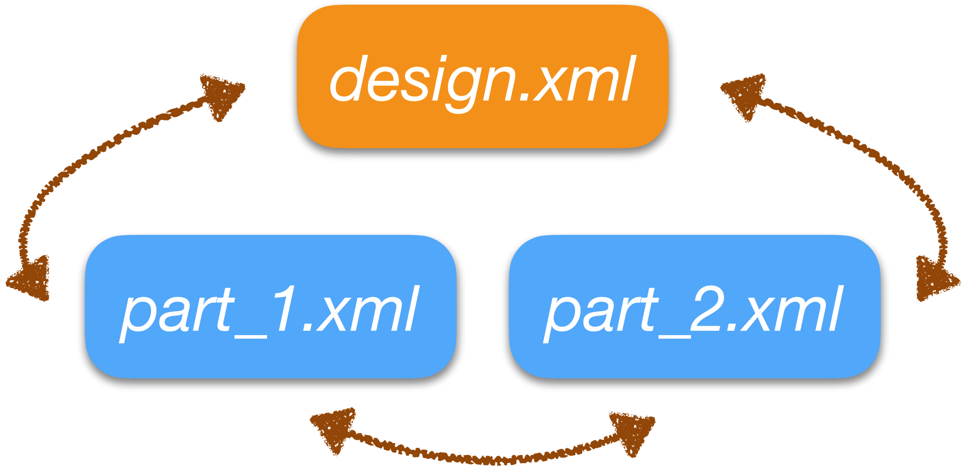
wing.xml

```
<Model ID="wing">
    <Variable ID="X".../>
    <Analysis ID="w".../>
    <Function ID="weight" Expr="w"/>
</Model>
```

External Parameters

Rules and Regulations

- Expression syntax: File{filename.xml}__ID where ID references the *ID* attribute of a ***Function*** element located in *filename.xml* and a double underscore is the separator
- Only ***Functions*** can be exchanged between files



External Parameters

Global Design Variables

design.xml

```
<Optimize>
 <Intersect Parts=“wing.xml,body.xml”…/>
   <Variable ID=“theta”…/>
   <Function ID=“theta_r” Expr=“theta*PI/180”/>
   …
</Optimize>
```

wing.xml

```
<Model ID=“wing”>
   <Function ID=“g_theta” Expr=“File{design.xml}__theta_r”/>
   <Variable ID=“X”/>
   …
</Model>
```

Pull global variable from design.xml

body.xml

```
<Model ID=“body”>
   <Function ID=“g_theta” Expr=“File{design.xml}__theta_r”/>
   <Variable ID=“Y”/>
   …
</Model>
```

Pull global variable from design.xml