Datawave Marine Science

OFreq Dynamics

Motion Model

(Equation Of Motion)

Nicholas Barczak

April 4, 2013

Rev: 1.0

DMS1303-000-110-01

Revision History

Revision	Date	Changes	Approval
1.00	April 08, 2013	Initial Issue	Nicholas Barczak

Prepared By

Datawave Marine Science
3500 27th Pl West
Apt 423
Seattle, WA 98199

www.dmsonline.us

Table Of Contents

1 Introduction....4
2 Equation of Motion – Functionality....4
2.1 Force Coefficients....4
2.2 Force Results....5
2.3 Derivatives....6
2.4 Multiple Equations....7
2.5 Interacting Variables....8
2.6 Constants....9
3 Information Flow....10
3.1 Information Flow – Force Coefficient....10
3.2 Information Flow – Force Value....11
3.3 Parameters....11
4 Data Management....11
5 Six Degree of Freedom Motion Model....12
5.1 Equation in Scientific Format....13
5.2 Equation Rearranged for Coding....17
6 Conclusion....24
7 References....25

1 Introduction

A motion model, also known as equation of motion, is a series of equations that relates forces to motions for some pre-determined situation. This report describes motion models in a general sense, as applied in oFreq. The report covers the interaction between the motion models and calling force objects. It covers some of the data management aspects of motion model. It then provides the mathematics for the standard six degree of freedom (6DOF) motion model.

2 Equation Of Motion – Functionality

2.1 Force Coefficients

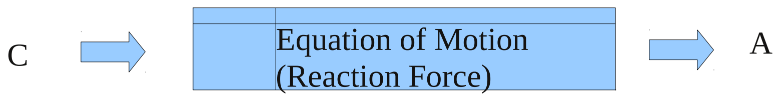
What is motion model? It takes many different forms, depending on which profession you ask. Engineers and physics present it as a series of mathematical equations with variables on the left side and constants on the right, such as this example.

$$A\,x = F \qquad \text{Equation 2.1}$$

Where:

A =	Some example constant
x =	The variable you want to solve for
F =	Some other constant that is known.

This is then solved analytically. From a programming perspective, a motion model is better thought of as a series of rules to convert an input coefficient for a specific variable over to a force coefficient.



Where:

C =	Some input coefficient
A =	Force coefficient shown in Equation 2.1

The motion model returns the value for the force coefficient, which can then be used to by a dynamics solver to actually solve the equation of motion.

The reason for defining an motion model in this manner is that it allows customization over how a coefficient affects an equation. Internally, the equation of motion can apply additional factors, such as the geometry for the body, unit conversions, or any other number of factors. Further, it provides some control over the validity of the inputs. For example, the motion model can be programmed to ignore input coefficients for certain force types by just putting a zero in front of the entry for that force type.

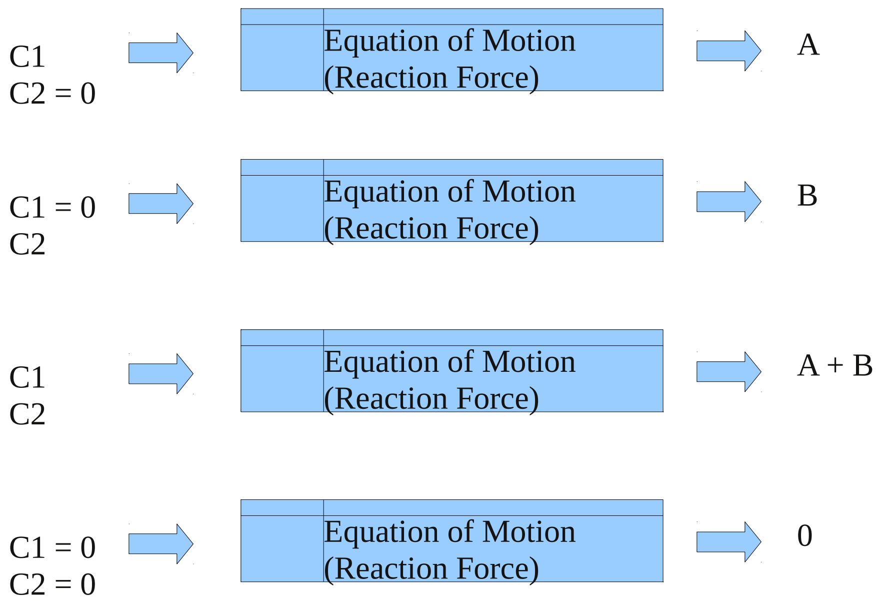
You can also enter any combination of variables. For example, if you had an equation that depended on variables *x,* and *y*, the equation would look like this.

$$A_{(C1)}x+B_{(C2)}y=F \qquad \text{Equation 2.2}$$

Where:

A =	1st example coefficient
B =	2nd example coefficient
C1 =	Input coefficient associated with A
C2 =	Input coefficient associated with B
x =	First variable to solve for
y =	Second variable to solve for

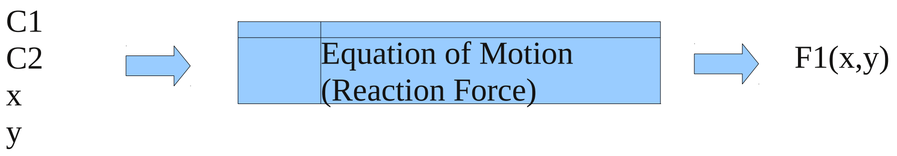
Then the motion model would still output a single force coefficient, and depending on the combination of inputs, this could be a force coefficient for A, for B, or for A and B in combination.


C1
C2 = 0
Equation of Motion (Reaction Force)
A
C1 = 0
C2
Equation of Motion (Reaction Force)
B
C1
C2
Equation of Motion (Reaction Force)
A + B
C1 = 0
C2 = 0
Equation of Motion (Reaction Force)
0

This generates four possibilities for outputs. The reason to allow combinations of variables as inputs is because then the motion model can be expanded to also provide actual values of returned force for a solved set of equations.

2.2 Force Results

At first, the motion model only returns a force coefficient. This needs to be used by a dynamics solver to solve for the actual motions. But once the solve finds the resulting motions, it may want to know what forces are actually generated by those motions. To accomplish that, it returns to the motion model. The inputs include the previous input coefficients, but also include the solve motion variables. With that, the equation returns the actual force value.



Where:
x = Solution for variable x.
y = Solution for variable y.
F1(x,y) = Resulting force from variables on the left side of the equation.

2.3 Derivatives

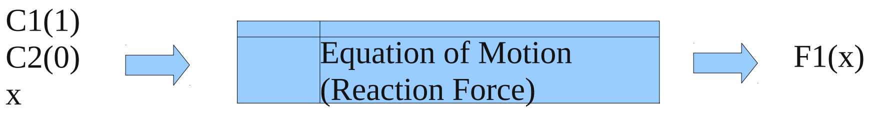
The last issue is that equations of motion often include derivatives. True, derivatives can become a complicated issue within dynamics. But the motion solver of oFreq makes this relatively easy. To return to previous example, think of it like every variable can become an array, and the order of the derivative accesses the index in the array.

$$A_{(C1)}x+B_{(C2)}y=F \qquad \text{Equation 2.3}$$

becomes

$$A_{(C1)}x_{(1)}+B_{(C2)}x_{(0)}=F \qquad \text{Equation 2.4}$$

The motion model returns a different value depending on what order of derivative is provided with the variable. Essentially, providing a different order is just accessing a different variable in the equation of motion.



Where:
C1(1) = Input coefficient, and integer to specify it is associated with 1st order derivative.
C2(0) = Input coefficient, and integer to specify it is associated with 0th order derivative.
x = Numerical value of the solved variable x.
F1(x) = Resulting force from variables on the left side of the equation.

Notice that to get the force, we don't need to supply values for all derivatives of *x*. This is because, within the class of equations used by oFreq, the derivative is fairly easy to work out internally by the equation of motion.

2.4 Multiple Equations

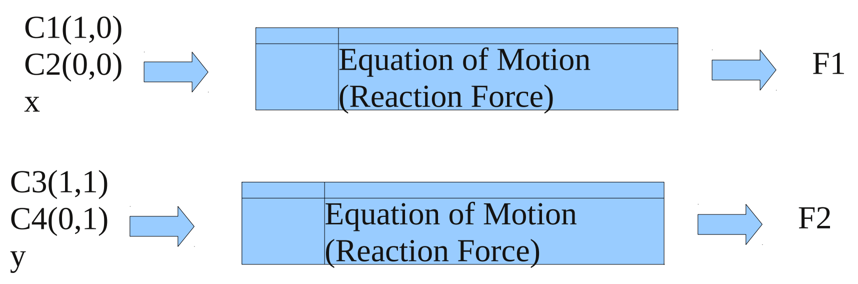
To solve real-world problems, the motion model usually consists of multiple simultaneous equations. Simultaneous only means that they are coupled and must be solved as a single unit. Thankfully, the motion model only needs to handle definition of these equations, not the solution. The equations can still be defined one at a time, one piece at a time. This does add another layer to the equation of motion. There are now multiple equations, each with derivatives. Now, when the force object supplies a user input coefficient, it must specify the derivative, and the equation that it goes go.

Take the following example equation set.

$$A_{(C1)} x_{(1)} + B_{(C2)} x_{(0)} = F$$

$$A_{(C3)} y_{(1)} + B_{(C4)} y_{(0)} = G \qquad \text{Equation 2.5}$$

This now modifies the input format for the equation coefficients to the following.



Where:

C1(1,0) = Input coefficient, and integer to specify it is associated with 1st order derivative, assigned to the 1st equation (0th index in an array).

C2(0,0) = Input coefficient, and integer to specify it is associated with 0th order derivative, assigned to the 1st equation (0th index in an array).

C3(1,1) = Input coefficient, and integer to specify it is associated with 1st order derivative, assigned to the 2nd equation (1st index in an array).

C4(0,1) = Input coefficient, and integer to specify it is associated with 0th order derivative, assigned to the 2nd equation (1st index in an array).

x = Numerical value of the solved variable x.

y = Numerical value of the solved variable y.

F1 = Resulting force from variables on the left side of the 1st equation (0th index in an array).

F2 = Resulting force from variables on the left side of the 2nd equation (1st index in an array).

Notice that it requires two uses of the motion model to return all the forces. The model only returns values for a single equation at a time. Part of the input selects which equation that will be and based on that selection, decided how to process the user input coefficients.

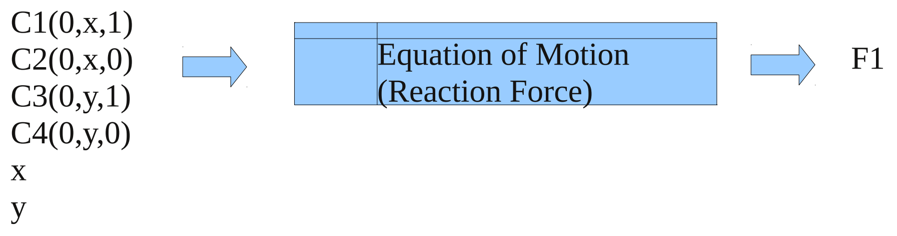
2.5 Interacting Variables

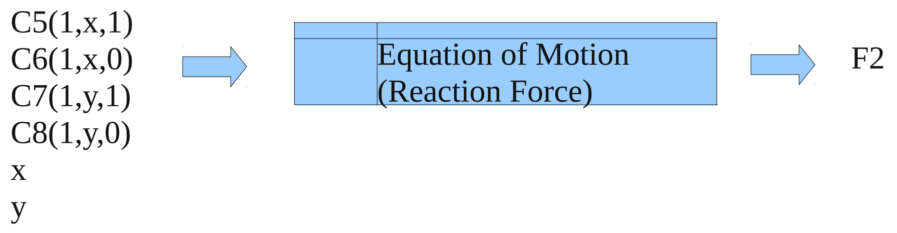
The last complexity to an equation of motion model is interacting variables. A correct model will always have the same number of equations as variables. So a model with two equations has two variables of motion to solve for. However, often these variables interact between the equations. And the derivatives may interact as well. This grows the equation of motion to looking like this.

$$A_{(C1)}x_{(1)}+B_{(C2)}x_{(0)}+C_{(C3)}y_{(1)}+D(C4)y_{(0)}=F$$

$$A_{(C5)}x_{(1)}+B_{(C6)}x_{(0)}+C_{(C7)}y_{(1)}+D_{(C8)}y_{(0)}=G$$ Equation 2.6

This turns the inputs for the equation of motion into the following two function returns. Notice now that each coefficient has an equation number (first argument), variable (second argument), and derivative (third argument) associated with it.





Where:

C1(1,x,0) = Input coefficient, and integer to specify it is associated with 1st order derivative of the variable x, assigned to the 1st equation (0th index in an array).

C2(0,x,0) = Input coefficient, and integer to specify it is associated with 0th order derivative of the variable x, assigned to the 1st equation (0th index in an array).

C3(1,y,0) = Input coefficient, and integer to specify it is associated with 1st order derivative of the variable y, assigned to the 1st equation (0th index in an array).

C4(0,y,0) = Input coefficient, and integer to specify it is associated with 0th order derivative of the variable y, assigned to the 1st equation (0th index in an array).

C5(1,x,0) = Input coefficient, and integer to specify it is associated with 1st order derivative of the variable x, assigned to the 2nd equation (1st index in an array).

C6(0,x,0) = Input coefficient, and integer to specify it is associated with 0th order derivative of the variable x, assigned to the 2nd equation (1st index in an array).

C7(1,y,0) = Input coefficient, and integer to specify it is associated with 1st order derivative of the variable y, assigned to the 2nd equation (1st index in an array).

C8(0,y,0) = Input coefficient, and integer to specify it is associated with 0th order derivative of the variable y, assigned to the 2nd equation (1st index in an array).

x = Numerical value of the solved variable x.

y = Numerical value of the solved variable y.

F1 = Resulting force from variables on the left side of the 1st equation (0th index in an array).

F2 = Resulting force from variables on the left side of the 2nd equation (1st index in an array).

2.6 Constants

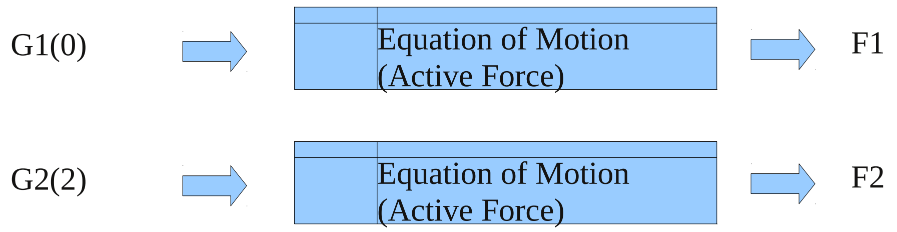
The last part to address for an equation of motion is the active force. Most of the equations have some constant term that represents the active force and does not depend on any of the variables or variable derivatives.

$$A_{(C1)}x+B_{(C2)}y=F1_{(G1)}$$
$$C_{(C3)}x+D_{(C4)}y=F2_{(G2)}$$

Equation 2.7

Notice that now the active forces *F1* and *F2* also depend on input coefficients *G1* and *G2*.

This active force works just like the reactive force. Only no variables or derivatives are associated with it.


G1(0)
Equation of Motion (Active Force)
F1
G2(2)
Equation of Motion (Active Force)
F2

Once again, we need to submit a single request for each equation that we want to obtain a force from. And notice that this time we are supplying coefficients for the active force instead of the reactive force. And the equation of motion becomes aware of this.

3 Information Flow

Information flow can become slightly confusing with the equation of motion, as defined in oFreq. The forces all exist external to the equation of motion. The equation of motion is simply a filter to allow custom definitions for converting from input coefficients to force coefficients.

3.1 Information Flow – Force Coefficient

The flowchart below shows the information flow for returning a force coefficient from an equation of motion. The figure only shows an example of a hydrodynamic cross-body force, but any of the seven force types could supply force information and return a value from the equation of motion. The equation is aware of each force type, which allows the equation definition to internally handle each force type differently, depending on the equation of motion.

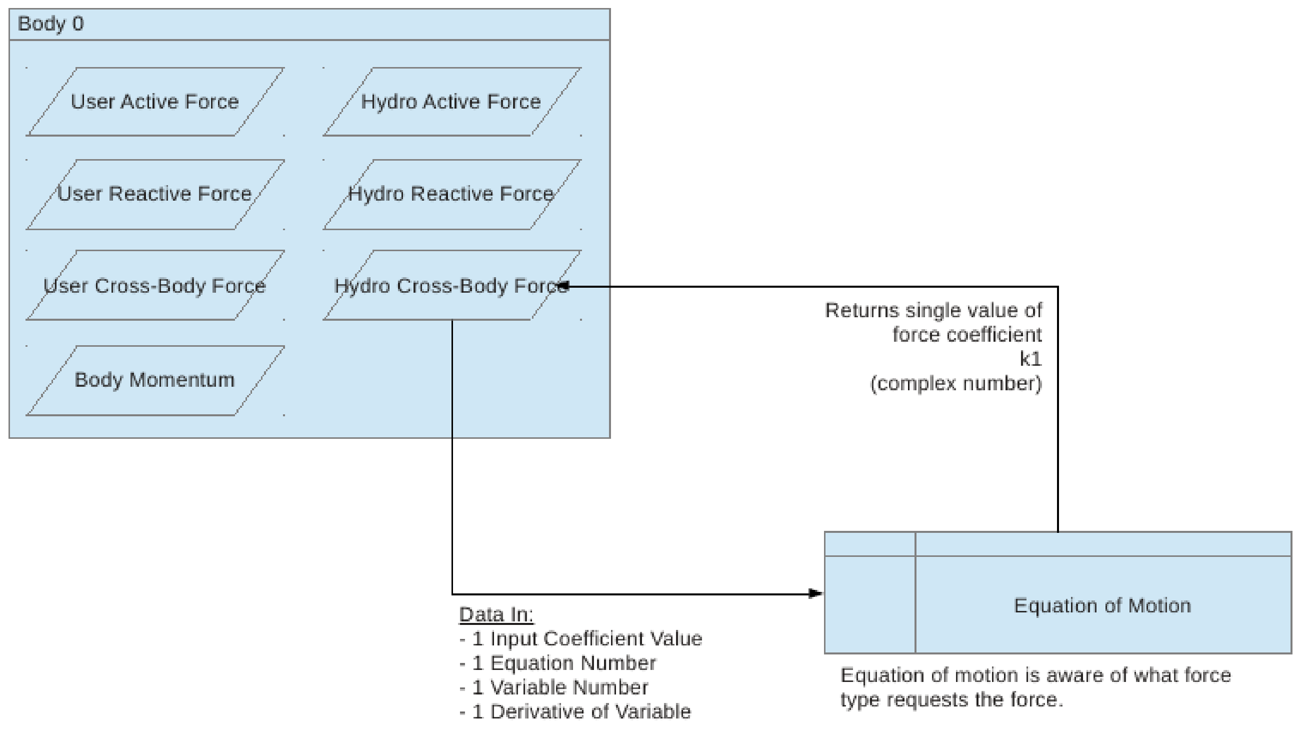


Figure 3.1: Force Coefficient Information Flow

3.2 Information Flow – Force Value

The flowchart below shows the information flow for returning a force coefficient from an equation of motion. Notice that it is the same procedure, just that the motion variable values are also supplied.

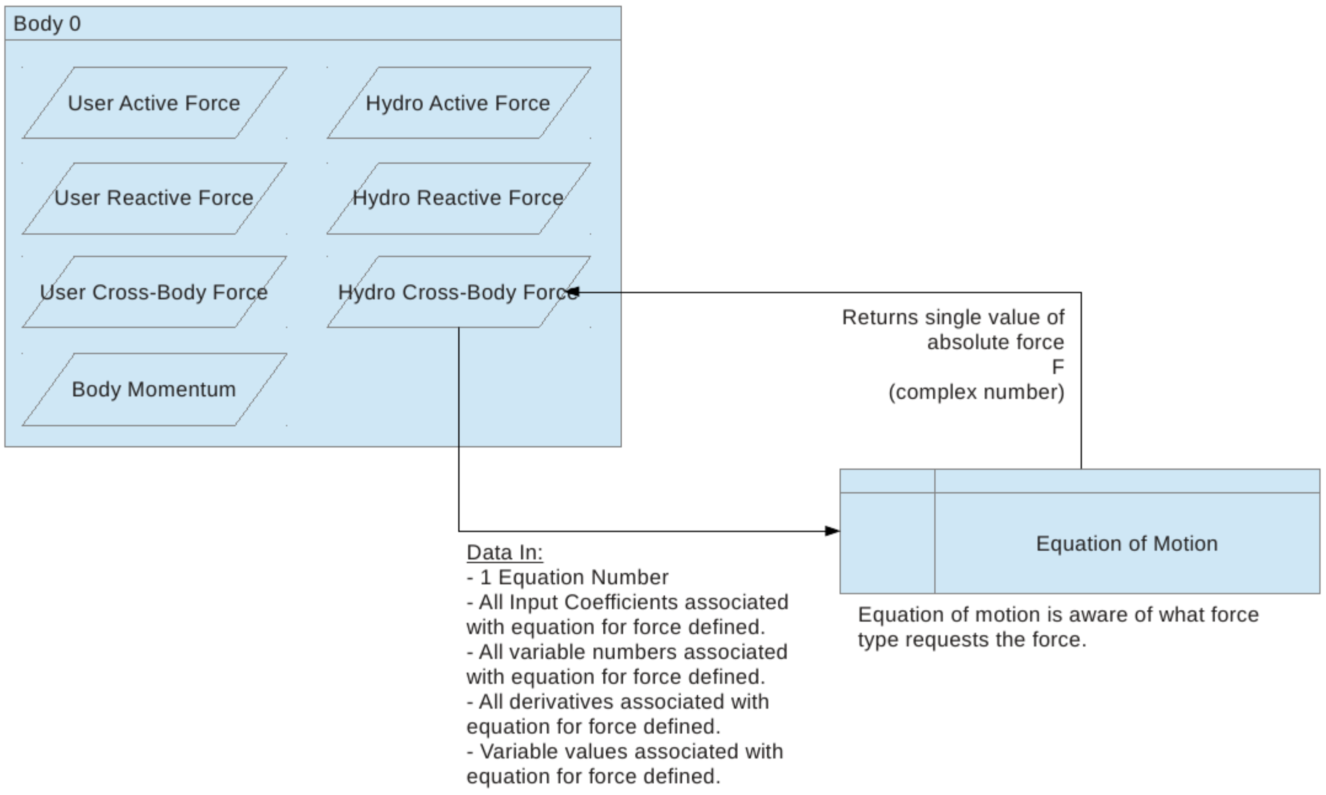

Body 0
User Active Force
Hydro Active Force
User Reactive Force
Hydro Reactive Force
User Cross-Body Force
Hydro Cross-Body Force
Body Momentum
Returns single value of absolute force F (complex number)
Equation of Motion
Data In:
- 1 Equation Number
- All Input Coefficients associated with equation for force defined.
- All variable numbers associated with equation for force defined.
- All derivatives associated with equation for force defined.
- Variable values associated with equation for force defined.
Equation of motion is aware of what force type requests the force.

Figure 3.2: Force Value Information Flow

3.3 Parameters

In addition to the force coefficients and returned values, the equation of motion also needs an interface for generic parameters. This might be things like the geometry features. Since the full list of possible parameters for all situations in the world is too general to anticipate, this list should be as open as possible. Probably just an integer to state the number of inputs, and an array containing each input.

4 Data Management

The equation of motion needs to return several different values depending on what the inputs are. This can all be accomplished through a single return function. The equation of motion just needs to assume appropriate default values for any inputs not supplied. With the correct selection of inputs, you can return any combination of outputs from any equation of motion. Assume the following default values.

1. Any force coefficients not set by input are zero valued by default.
2. Any motion variables not set by input are (1.0 + i0.0) by default. (Remember that motion variables are a complex number).

All the equation has to do is insert these default values, plus the few user specified inputs, and evaluate the entire equation.

For example, if you want a single force coefficient returned for an input coefficient C1, then any input coefficients not C1 are set to zero and their terms don't add into the equation. All the motion variables are set to (1.0 + i0.0), which is just 1. Anything times 1.0 is still 1.0, so the variable values don't affect the result.

Now if you wanted to get an actual force out, set several coefficients for an equation, C1 through C4. (C1-C4 is just an example. An actual equation of motion may have any number of input coefficients.) And set the actual values for the motion variables. Now, all of the force coefficients for that equation are real values, and will not evaluate to zero. And all the motion variables are set to something other than one, and will interact with the forces appropriately.

Treating the equation of motion in this manner allows programming a single method to evaluate an equation and allows for numerous combinations of usage.

5 Six Degree Of Freedom Motion Model

Thus far, the document has discussed motion models in general terms. It covered the purpose of a motion model and methods of interaction with it. This section now provides the specific mathematics for a single motion model: the six degree of freedom model (6DOF). This is the most common model for dynamics solvers. The equations are first presented in their typical scientific format. And then they are rearranged and expanded into a format for coding, which actually shows each force type.

The 6DOF model has six equations and six variables. The variables are:

1. TX = x0: Translation in the X-axis (m).
2. TY = x1: Translation in the Y-axis (m).
3. TZ = x2: Translation in the Z-axis (m).
4. RX = x3: Rotation about the X-axis (rad).
5. RY = x4: Rotation about the Y-axis (rad).
6. RZ = x5: Rotation about the Z-axis (rad).

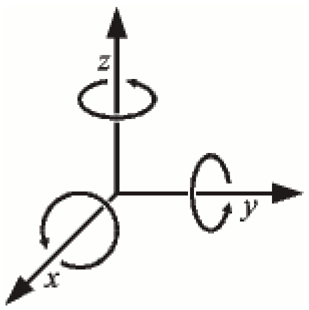


Figure 5.1: Six Degree of Freedom Coordinate System

5.1 Equation In Scientific Format

5.1.1 Translation X – Equation 0 (x0)

$$
\begin{aligned}
& M\frac{d^2\underline{x0}}{dt^2} - M\,y_g\frac{d^2\underline{x4}}{dt^2} + M\,x_g\frac{d^2\underline{x5}}{dt^2} + \\
& \sum_{u=0}^{5}\left(\sum_{v=0}^{5} C(u)_{HydroReact(0,v)}\frac{d^u\underline{x_v}}{dt^u}\right) + \\
& \sum_{u=0}^{N}\left(\sum_{v=0}^{5} C(u)_{UserReact(0,v)}\frac{d^u\underline{x_v}}{dt^u}\right) + \\
& \sum_{b=0}^{M}\delta_{0b}\left[\sum_{u=0}^{5}\left(\sum_{v=0}^{5} C(u,b)_{HydroCross(0,v)}\,d^u\frac{\underline{x_v}}{dt^u}\right)\right] + \\
& \sum_{b=0}^{M}\delta_{0b}\left[\sum_{u=0}^{N}\left(\sum_{v=0}^{5} C(u,b)_{UserCross(0,v)}\,d^u\frac{\underline{x_v}}{dt^u}\right)\right] + \\
& = \\
& \underline{F0_{HydroActive}} + \\
& \underline{F0_{UserActive}}
\end{aligned}
\qquad \text{Equation 5.1}
$$

Subscripts:
v = Variable index of x0 to x5 (0 to 5).
u = Order of derivative (0 to 5 for hydrodynamic forces, 0 to N for user forces).
b = Index of the body for a cross body force (0 to M).
N = Highest order of derivative for user forces.
M = Total number of bodies

Variables:

M =	Mass of body.
x_g =	X-axis distance from center of gravity to point of rotation (parameter 2, taken as zero in our case).
y_g =	Y-axis distance from center of gravity to point of rotation (parameter 2, taken as zero in our case).
$C(u)_{HydroReact(0,v)}$ =	Hydrodynamic input coefficient for reaction forces for variable *u* of derivative *v*.
$C(u)_{UserReact(0,v)}$ =	User input coefficient for reaction forces for variable *u* of derivative *v*.
$C(u,b)_{HydroCross(0,v)}$ =	Hydrodynamic input coefficient for reaction forces for variable *u* of derivative *v,* linking to body *b*.
$C(u,b)_{UserCross(0,v)}$ =	User input coefficient for reaction forces for variable *u* of derivative *v,* linking to body *b*.
δ_{0b} =	Kronecker Delta. Just a fancy math term that means when *b* = 0, the value is zero, and that part of the summation does not count.
$F0_{HydroActive}$ =	Hydrodynamic active force for body.
$F0_{UserActive}$ =	User defined active force for the body.

The underline notation means a complex number.

Pay careful attention to the subscript notations. These six equations will start to look very similar, and the only major difference between them is the subscript notation.

5.1.2 Translation Y – Equation 1 (x1)

$$M\frac{d^2\underline{x1}}{dt^2}+M\,y_g\frac{d^2\underline{x1}}{dt^2}-M\,z_g\frac{d^2\underline{x5}}{dt^2}+$$
$$\sum_{u=0}^{5}\left(\sum_{v=0}^{5}C(u)_{HydroReact(1,v)}\frac{d^u\underline{x_v}}{dt^u}\right)+$$
$$\sum_{u=0}^{N}\left(\sum_{v=0}^{5}C(u)_{UserReact(1,v)}\frac{d^u\underline{x_v}}{dt^u}\right)+$$
$$\sum_{b=0}^{M}\delta_{0b}\left[\sum_{u=0}^{5}\left(\sum_{v=0}^{5}C(u,b)_{HydroCross(1,v)}d^u\frac{\underline{x_v}}{dt^u}\right)\right]+$$
$$\sum_{b=0}^{M}\delta_{0b}\left[\sum_{u=0}^{N}\left(\sum_{v=0}^{5}C(u,b)_{UserCross(1,v)}d^u\frac{\underline{x_v}}{dt^u}\right)\right]+$$
$$=$$
$$\underline{F1}_{HydroActive}+$$
$$\underline{F1}_{UserActive}$$

Equation 5.2

5.1.3 Translation Z – Equation 2 (x2)

$$M\frac{d^2\underline{x2}}{dt^2}-M\,x_g\frac{d^2\underline{x3}}{dt^2}+M\,z_g\frac{d^2\underline{x4}}{dt^2}+$$
$$\sum_{u=0}^{5}\left(\sum_{v=0}^{5}C(u)_{HydroReact(2,v)}\frac{d^u\underline{x_v}}{dt^u}\right)+$$
$$\sum_{u=0}^{N}\left(\sum_{v=0}^{5}C(u)_{UserReact(2,v)}\frac{d^u\underline{x_v}}{dt^u}\right)+$$
$$\sum_{b=0}^{M}\delta_{0b}\left[\sum_{u=0}^{5}\left(\sum_{v=0}^{5}C(u,b)_{HydroCross(2,v)}d^u\frac{\underline{x_v}}{dt^u}\right)\right]+$$
$$\sum_{b=0}^{M}\delta_{0b}\left[\sum_{u=0}^{N}\left(\sum_{v=0}^{5}C(u,b)_{UserCross(2,v)}d^u\frac{\underline{x_v}}{dt^u}\right)\right]+$$
$$=$$
$$\underline{F2}_{HydroActive}+$$
$$\underline{F2}_{UserActive}$$

Equation 5.3

5.1.4 Rotation X – Equation 3 (x3)

$$-M\,y_g\frac{d^2\,\underline{x1}}{dt^2}+M\,x_g\frac{d^2\,\underline{x2}}{dt^2}+I_{xx}\frac{d^2\,\underline{x3}}{dt^2}+I_{xy}\frac{d^2\,\underline{x4}}{dt^2}+I_{xz}\frac{d^2\,\underline{x5}}{dt^2}$$

$$\sum_{u=0}^{5}\left(\sum_{v=0}^{5}C(u)_{HydroReact(3,v)}\frac{d^u\,\underline{x_v}}{dt^u}\right)+$$

$$\sum_{u=0}^{N}\left(\sum_{v=0}^{5}C(u)_{UserReact(3,v)}\frac{d^u\,\underline{x_v}}{dt^u}\right)+$$

$$\sum_{b=0}^{M}\delta_{0b}\left[\sum_{u=0}^{5}\left(\sum_{v=0}^{5}C(u,b)_{HydroCross(3,v)}d^u\frac{\underline{x_v}}{dt^u}\right)\right]+ \qquad \text{Equation 5.4}$$

$$\sum_{b=0}^{M}\delta_{0b}\left[\sum_{u=0}^{N}\left(\sum_{v=0}^{5}C(u,b)_{UserCross(3,v)}d^u\frac{\underline{x_v}}{dt^u}\right)\right]+$$

$$=$$

$$\underline{F3_{HydroActive}}+$$
$$\underline{F3_{UserActive}}$$

5.1.5 Rotation Y – Equation 4 (x4)

$$M\,y_g\frac{d^2\,\underline{x0}}{dt^2}-M\,z_g\frac{d^2\,\underline{x3}}{dt^2}+I_{xy}\frac{d^2\,\underline{x3}}{dt^2}+I_{yy}\frac{d^2\,\underline{x4}}{dt^2}+I_{yz}\frac{d^2\,\underline{x5}}{dt^2}$$

$$\sum_{u=0}^{5}\left(\sum_{v=0}^{5}C(u)_{HydroReact(4,v)}\frac{d^u\,\underline{x_v}}{dt^u}\right)+$$

$$\sum_{u=0}^{N}\left(\sum_{v=0}^{5}C(u)_{UserReact(4,v)}\frac{d^u\,\underline{x_v}}{dt^u}\right)+$$

$$\sum_{b=0}^{M}\delta_{0b}\left[\sum_{u=0}^{5}\left(\sum_{v=0}^{5}C(u,b)_{HydroCross(4,v)}d^u\frac{\underline{x_v}}{dt^u}\right)\right]+ \qquad \text{Equation 5.5}$$

$$\sum_{b=0}^{M}\delta_{0b}\left[\sum_{u=0}^{N}\left(\sum_{v=0}^{5}C(u,b)_{UserCross(4,v)}d^u\frac{\underline{x_v}}{dt^u}\right)\right]+$$

$$=$$

$$\underline{F4_{HydroActive}}+$$
$$\underline{F4_{UserActive}}$$

5.1.6 Rotation Z – Equation 5 (x5)

$$-M x_g \frac{d^2 \underline{x0}}{dt^2} + M z_g \frac{d^2 \underline{x3}}{dt^2} + I_{xz} \frac{d^2 \underline{x3}}{dt^2} + I_{yz} \frac{d^2 \underline{x4}}{dt^2} + I_{zz} \frac{d^2 \underline{x5}}{dt^2}$$

$$\sum_{u=0}^{5} \left(\sum_{v=0}^{5} C(u)_{HydroReact(5,v)} \frac{d^u \underline{x_v}}{dt^u} \right) +$$

$$\sum_{u=0}^{N} \left(\sum_{v=0}^{5} C(u)_{UserReact(5,v)} \frac{d^u \underline{x_v}}{dt^u} \right) +$$

$$\sum_{b=0}^{M} \delta_{0b} \left[\sum_{u=0}^{5} \left(\sum_{v=0}^{5} C(u,b)_{HydroCross(5,v)} d^u \frac{\underline{x_v}}{dt^u} \right) \right] +$$

$$\sum_{b=0}^{M} \delta_{0b} \left[\sum_{u=0}^{N} \left(\sum_{v=0}^{5} C(u,b)_{UserCross(5,v)} d^u \frac{\underline{x_v}}{dt^u} \right) \right] +$$

$$=$$

$$\underline{F5_{HydroActive}} +$$
$$\underline{F5_{UserActive}}$$

Equation 5.6

5.2 Equation Rearranged For Coding

5.2.1 Supporting Functions

To code to the equations, there are a few supporting functions that first must be defined. These supporting functions are very useful for any motion model and should be coded to be available for any user defined equation of motion.

5.2.1.1 Parameters Function

The parameters function is the array of user defined parameters for the equation of motion. For the 6DOF equation, the parameters have no meaning and are not used. They are merely included as an additional feature for the user at a later date.

5.2.1.2 Kronecker Delta Function

The Kronecker delta sounds far more complicated than it really is. The function can only have one of two values: 0 or 1.

kronecker(i,j)
Inputs:
i = Integer index 1.
j = Integer index 2.
Output:
kronecker = 0 or 1, real number, double variable.

It is defined as follows.

$$kronecker = \begin{pmatrix} 0 & i=j \\ 1 & i\neq j \end{pmatrix}$$ Equation 5.7

This is just used when iterating through two indices to calculate some relationship between two objects and you want to stop an object from referencing itself.

5.2.1.3 Differentiation Function

The time differentiation function is fairly simply defined.

ddt(variable $\underline{x}$, order *v*)

Inputs:
$\underline{x}$ = Variable to provide a differential for. Complex variable
v = Order of differential. Integer value.
Output:
ddt = Complex value that is the differential of x

The function is defined as follows.

$$ddt = \begin{pmatrix} 1 & x=1 \\ \underline{x}(i)^{v}\omega^{v} & x\neq 1 \end{pmatrix}$$ Equation 5.8

Where:
ω = Wave frequency (rad/s).

First the function checks if x = 1. If so, it just returns the value of 1, as a real number. Otherwise, the function follows the second formula and multiplies the value of x by an imaginary number (1,i) and the wave frequency, both raised to the power of the derivative order.

5.2.1.4 Summation Function

The summation function adds all the elements of a supplied variable within a given set of limits. The function can be supplied with lower and upper limits of summation. Or the user can supply a keyword to denote no limit of summation, in which case the function just sums over all available indices in the array. The function also accepts the input variable as a single column of values, or the user can enter a set of keywords to specify one of the built in variables. Finally, the function requires the user to specify a keyword to determine which part of the array to sum over. Adding all these elements together, the summation function can take one of eight forms.

sum(from<integer>, to<integer>, variable<string>, index<string>)

sum(from<integer>, to<integer>, variable<array>, index<string>)

sum(from<integer>, to<string>, variable<string>, index<string>)

sum(from<integer>, to<string>, variable<array>, index<string>)

sum(from<string>, to<integer>, variable<string>, index<string>)

sum(from<string>, to<integer>, variable<array>, index<string>)

sum(from<string>, to<string>, variable<string>, index<string>)

sum(from<string>, to<string>, variable<array>, index<string>)

Inputs:
from = If <integer> value, the starting integer for the summation. If a <string> value, (normally the keyword 'N'), then the lower limit of the array.
to = If <integer> value, the ending integer for the summation. If a <string< value, (normally the keyword 'N'), then the upper limit of the array.
variable = If <array> value, the direct array of values that the function should sum over. If a <string> value, then one of the seven main force types built into the motion model. A list of keywords is provided later.
index = The summation index. Most of these variables are multi-dimensional. The function definition supplies the one variable that should be summed over, using a keyword. Then the function finds the appropriate index for that variable, based on the keyword supplied. A list of keywords is provided later.

Output:
sum = A single value.
sum = A single column array of values. Even if the summation of a multidimensional array would generate a row of value, this must be rotated to always output as a column of values.

The keywords referencing the built in forces can be any of the following. If the function receives a string for the input data set that matches one of these inputs, it will search to find the corresponding array within the object's private variables. Otherwise, it will sum the column array that was provided.

- **HydroReact**: Hydrodynamic reaction forces for each equation (2 dimensional array depending on variable and order of derivative).
- **HydroActive**: Hydrodynamic active forces for equation (single complex value).
- **HydroCrossBody**: Hydrodynamic cross-body forces (3 dimensional array depending on linking body, variable, and order of derivative).
- **UserReact**: User reaction forces for each equation (2 dimensional array depending on variable and order of derivative).
- **UserActive**: User active forces for equation (single complex value).

- **UserCrossBody**: User cross-body forces (3 dimensional array depending on linking body, variable, and order of derivative).
- **BodyMass**: Mass properties of body for each equation. (single dimensional array, depending on variable).

Note that in this representation, each equation has this set of arrays assigned for each force type. Whichever equation calls the reference to these forces determines which array of forces gets returned.

The notation for representing a summation to the limits of the array will simply be the letter “N” instead of an integer. And actually, the function only needs to know that a string was supplied as a variable type instead of an integer, but the user usually prefers to think that some specific keyword will trigger the action.

And finally, the indices must be specified. Normally, a program code would set hard limits for array indices and then take the one index for variation and assign it to a variable. This would require the user to know the exact sequence of the array indices and their appropriate limits. A far easier option for the user is to instead specify a keyword that lets the program know which type of data is summed across. Depending on the data type supplied, the program handles the indices for each force type differently. The following keywords are provided for summation index specification.

- b = sum across entries for linking bodies (only applies to cross-body forces).
- d = sum across all derivatives for force.
- v = sum across all variables defined for a force.

This covers all aspects of the summation function. Frequently, equations arise that require nested summations and this function is not sufficient. For that, the user has no recourse but to write their own nested for loops.

5.2.1.5 For Loop Summation

For-loop summation does not specifically require a function definition. This only shows an example of how math summation notation translates to C++ for-loop code. For example, take the following math summation notation.

$$\sum_{u=0}^{N}\left(\sum_{v=0}^{5} C(u,v)\right)$$ Equation 5.9

This translates to the following C++ code, where maxsize() is the maximum entry of the array.

```
out = 0;
for (u = 0 ; u > C.maxsize() ; u++) {
 for (v = 0 ; v > 5 ; v++) {
  out = out + C[u][v];
 }
}
```

5.2.2 Translation X – Equation 0 (x0)

This provides the equation of translation along the X-axis, converted over closer to program code. Most of the math notation was replaced with function or variable definitions. Math summation notation must be replaced with custom for-loop summation.

$$\begin{aligned}
&Equation\,0 = \\
&BodyMass(0)\,ddt(\underline{x(0)},2) + BodyMass(4)\,ddt(\underline{x(4)},2) + BodyMass(5)\,ddt(\underline{x(5)},2) + \\
&\sum_{u=0}^{5}\left(\sum_{v=0}^{5} HydroReact(u,v)\,ddt(\underline{x(v)},u)\right) + \\
&\sum_{u=0}^{N}\left(\sum_{v=0}^{5} UserReact(u,v)\,ddt(\underline{x(v)},u)\right) + \\
&\sum_{b=0}^{M} kronecker(0,b)\left[\sum_{u=0}^{5}\left(\sum_{v=0}^{5} HydroCross(b,u,v)\,ddt(\underline{x(v)},u)\right)\right] + \\
&\sum_{b=0}^{M} kronecker(0,b)\left[\sum_{u=0}^{N}\left(\sum_{v=0}^{5} UserCross(b,u,v)\,ddt(\underline{x(v)},u)\right)\right] - \\
&\underline{HydroActive} - \\
&\underline{UserActive}
\end{aligned}
\qquad \text{Equation 5.10}$$

Where:

x(v) = Complex number. Variable xv, where v = 0,1,2, … 5. For example, if v = 0, then x(0) = x0 = translation in X-axis.
ddt() = Differential function as defined in Section 5.2.1.3
kronecker() = Kronecker delta function as defined in 5.2.1.2
u = Represents order of derivative reference.
v = Represents variables used in equation.
b = Represents bodies listed for the run, including own body that is calling the equation model.
HydroReact(u,v) = Hydrodynamic reaction forces for equation (2 dimensional array depending on variable, *v*, and order of derivative, *u*).
HydroActive: Hydrodynamic active forces for equation (single complex value).
HydroCrossBody(b,u,v) = Hydrodynamic cross-body forces (3 dimensional array depending on body, *b*, variable, *v*, and order of derivative, *u*).
UserReact(u,v) = User reaction forces for each equation (2 dimensional array depending on variable, *v*, and order of derivative, *u*).

UserActive: User active forces for equation (single complex value).
UserCrossBody(b,u,v): User cross-body forces (3 dimensional array depending on body, *b*, variable, *v*, and order of derivative, *u*).
BodyMass(v): Mass properties of body for each equation. (single dimensional array, depending on variable).

5.2.3 Translation Y – Equation 1 (x1)

$$Equation1=$$

$$\begin{aligned}&BodyMass(1)ddt(\underline{x(1)},2)+BodyMass(3)ddt(\underline{x(3)},2)+BodyMass(5)ddt(\underline{x(5)},2)+\\&\sum_{u=0}^{5}\left(\sum_{v=0}^{5}HydroReact(u,v)ddt(\underline{x(v)},u)\right)+\\&\sum_{u=0}^{N}\left(\sum_{v=0}^{5}UserReact(u,v)ddt(\underline{x(v)},u)\right)+\\&\sum_{b=0}^{M}kronecker(0,b)\left[\sum_{u=0}^{5}\left(\sum_{v=0}^{5}HydroCross(b,u,v)ddt(\underline{x(v)},u)\right)\right]+\\&\sum_{b=0}^{M}kronecker(0,b)\left[\sum_{u=0}^{N}\left(\sum_{v=0}^{5}UserCross(b,u,v)ddt(\underline{x(v)},u)\right)\right]-\\&\underline{HydroActive}-\\&\underline{UserActive}\end{aligned}$$

Equation 5.11

5.2.4 Translation Z – Equation 2 (x2)

$$Equation2=$$

$$\begin{aligned}&BodyMass(2)ddt(\underline{x(2)},2)+BodyMass(3)ddt(\underline{x(3)},2)+BodyMass(4)ddt(\underline{x(4)},2)+\\&\sum_{u=0}^{5}\left(\sum_{v=0}^{5}HydroReact(u,v)ddt(\underline{x(v)},u)\right)+\\&\sum_{u=0}^{N}\left(\sum_{v=0}^{5}UserReact(u,v)ddt(\underline{x(v)},u)\right)+\\&\sum_{b=0}^{M}kronecker(0,b)\left[\sum_{u=0}^{5}\left(\sum_{v=0}^{5}HydroCross(b,u,v)ddt(\underline{x(v)},u)\right)\right]+\\&\sum_{b=0}^{M}kronecker(0,b)\left[\sum_{u=0}^{N}\left(\sum_{v=0}^{5}UserCross(b,u,v)ddt(\underline{x(v)},u)\right)\right]-\\&\underline{HydroActive}-\\&\underline{UserActive}\end{aligned}$$

Equation 5.12

5.2.5 Rotation X – Equation 3 (x3)

$$
\begin{aligned}
&Equation3 = \\
&BodyMass(1)\,ddt(\underline{x(1)},2)+BodyMass(2)\,ddt(\underline{x(2)},2)+\\
&BodyMass(3)\,ddt(\underline{x(3)},2)+BodyMass(4)\,ddt(\underline{x(4)},2)+\\
&BodyMass(5)\,ddt(\underline{x(5)},2)+\\
&\sum_{u=0}^{5}\left(\sum_{v=0}^{5} HydroReact(u,v)\,ddt(\underline{x(v)},u)\right)+\\
&\sum_{u=0}^{N}\left(\sum_{v=0}^{5} UserReact(u,v)\,ddt(\underline{x(v)},u)\right)+\\
&\sum_{b=0}^{M} kronecker(0,b)\left[\sum_{u=0}^{5}\left(\sum_{v=0}^{5} HydroCross(b,u,v)\,ddt(\underline{x(v)},u)\right)\right]+\\
&\sum_{b=0}^{M} kronecker(0,b)\left[\sum_{u=0}^{N}\left(\sum_{v=0}^{5} UserCross(b,u,v)\,ddt(\underline{x(v)},u)\right)\right]-\\
&\underline{HydroActive}-\\
&\underline{UserActive}
\end{aligned}
$$

Equation 5.13

5.2.6 Rotation Y – Equation 4 (x4)

$$
\begin{aligned}
&Equation4 = \\
&BodyMass(0)\,ddt(\underline{x(0)},2)+BodyMass(2)\,ddt(\underline{x(2)},2)+\\
&BodyMass(3)\,ddt(\underline{x(3)},2)+BodyMass(4)\,ddt(\underline{x(4)},2)+\\
&BodyMass(5)\,ddt(\underline{x(5)},2)+\\
&\sum_{u=0}^{5}\left(\sum_{v=0}^{5} HydroReact(u,v)\,ddt(\underline{x(v)},u)\right)+\\
&\sum_{u=0}^{N}\left(\sum_{v=0}^{5} UserReact(u,v)\,ddt(\underline{x(v)},u)\right)+\\
&\sum_{b=0}^{M} kronecker(4,b)\left[\sum_{u=0}^{5}\left(\sum_{v=0}^{5} HydroCross(b,u,v)\,ddt(\underline{x(v)},u)\right)\right]+\\
&\sum_{b=0}^{M} kronecker(4,b)\left[\sum_{u=0}^{N}\left(\sum_{v=0}^{5} UserCross(b,u,v)\,ddt(\underline{x(v)},u)\right)\right]-\\
&\underline{HydroActive}-\\
&\underline{UserActive}
\end{aligned}
$$

Equation 5.14

5.2.7 Rotation Z – Equation 5 (x5)

$$\begin{aligned}
&Equation5=\\
&BodyMass(0)\,ddt(\underline{x(0)},2)+BodyMass(1)\,ddt(\underline{x(1)},2)+\\
&BodyMass(3)\,ddt(\underline{x(3)},2)+BodyMass(4)\,ddt(\underline{x(4)},2)+\\
&BodyMass(5)\,ddt(\underline{x(5)},2)+\\
&\sum_{u=0}^{5}\left(\sum_{v=0}^{5} HydroReact(u,v)\,ddt(\underline{x(v)},u)\right)+\\
&\sum_{u=0}^{N}\left(\sum_{v=0}^{5} UserReact(u,v)\,ddt(\underline{x(v)},u)\right)+\\
&\sum_{b=0}^{M} kronecker(0,b)\left[\sum_{u=0}^{5}\left(\sum_{v=0}^{5} HydroCross(b,u,v)\,ddt(\underline{x(v)},u)\right)\right]+\\
&\sum_{b=0}^{M} kronecker(0,b)\left[\sum_{u=0}^{N}\left(\sum_{v=0}^{5} UserCross(b,u,v)\,ddt(\underline{x(v)},u)\right)\right]-\\
&\underline{HydroActive}-\\
&\underline{UserActive}
\end{aligned}$$

Equation 5.15

5.2.8 Coding Suggestions

The equations have a large number of summation loops in them. In many scenarios, many of these loops will iterate through arrays that are all blank (filled with zero). To create more efficient execution, have some way to limit the for-loops for scenarios where none of the coefficients within a loop are defined. (The scenario where all items in an array are zero.) This may be some internal property that the object turns on or off when an array gets non-zero values added.

6 Conclusion

This completes the major concepts for the motion models. In summary, a motion model is really just a large versatile method of converting input coefficients into force coefficients or forces directly. It gets defined in such a way to allow any set of motions or equations. And the equations for a six degree of freedom model were presented, both in their pure mathematical form and in a form better suited to program coding.

7 References