

CES320124

Civil 3D and Dynamo – Dynamic Culvert Design and Analysis

Andrew Milford
Autodesk

Learning Objectives

- Leverage the Civil 3D Corridor model to create 3D drainage culverts in Dynamo
- Use Dynamo to create complex 3D Drainage culvert models from user-defined parameters
- Explore Dynamo's Python node to undertake culvert analysis and sizing
- Create Civil 3D geometry from within the Dynamo workspace for documentation

Description

In this class you will learn how to use Autodesk Civil 3D and Dynamo to model and analyse roadway drainage culverts. Learn how to create dynamic 3D culvert models by leveraging information directly from your Civil 3D design corridor, using a combination of standard Dynamo nodes and custom Python scripts to analyse and optimise the desired culvert size. Finally, we will investigate how to take the Dynamo 3D model back into Civil 3D and Revit for documentation purposes. This session is intended for those wishing to begin their Civil 3D and Dynamo automation journey leveraging the power of the Dynamo interface.

Speaker(s)



Andrew Milford is a Senior Civil Infrastructure Implementation Consultant for Autodesk and is responsible for providing technical and business consulting, ensuring customers achieve successful adoption of Autodesk's Infrastructure Solutions across the Asia/Pacific region. Prior to joining Autodesk, Andrew gained over 25 years of design experience in the civil infrastructure industry, working as a geometric road designer for large consulting companies such as SKM/Jacobs, Arcadis Consulting, and AECOM. Andrew's experience extends from designing large highways,

tunnels, and interchanges down to smaller subdivision work using a variety of different design and drafting software packages. He is an AutoCAD Civil 3D Certified Professional and loves diving deep into AutoCAD Civil 3D software to develop and automate processes through scripts, AutoLISP, .NET API C# and VB, and Python.

Introduction

Civil 3D is a civil engineering design tool for producing high-quality 3D models for construction and documentation.

Dynamo is a visual programming and scripting tool that has been integrated into Civil 3D for the 2020 release, allowing you to work with native Civil 3D and AutoCAD objects. Dynamo also provides functionality to exchange data between Excel and allow for further extension with simple programming languages like Python.

In today's competitive environment, there is now a strong emphasis on leveraging automation to perform more and more design and documentation tasks, maintaining high-quality deliverables delivered within short timeframes. Typically, the development of these automation tools is reserved for the very few individuals within companies and require a mid-to-high degree of programming knowledge to create even the most basic of programs.

By combining native modelling tools within Civil 3D, the visual programming capabilities of Dynamo and the extended capabilities of Python integration, this class will demonstrate practical workflows in extracting and creating Civil 3D & AutoCAD design information using standard out-of-the-box (OOTB) Dynamo nodes and Python scripts.

Overview

This document has been split into 4 learning objectives, with each exercise building on the previous chapter's outcomes.

Each objective will also contain a sectioning outlining important nodes and processes undertaken to complete the related Dynamo graph.

Learning Objective 1 - Leverage the Civil 3D Corridor model to create 3D drainage culverts in Dynamo

The first objective is to understand how to setup a Civil 3D corridor for efficient use in Dynamo, including some tips and tricks for efficient culvert creation.

We will also look at the required parameters and design guidelines required to create a detailed culvert and headwall.

Finally, we will review some fundamental Dynamo principles, nodes and workflows that will be used throughout the class.

Learning Objective 2 - Use Dynamo to create complex 3D Drainage culvert models from user-defined parameters

Following from Objective 1, we will move into Dynamo to begin creating a single manual-designed culvert, leveraging data from the Civil 3D corridor including:

- Alignment and Profile
- Station
- Skew

Additionally, we will setup culvert parameters (i.e. pipe size, number of cells etc) to create Dynamo geometry which leverages Civil 3D corridor information.

Finally, we will investigate techniques to create and maintain tidy Dynamo graphs that are easy to read and amend. We will use all this information to create a single culvert based, in Civil 3D, with user-defined parameters

Exercise 3 - Explore Dynamo's Python node to undertake culvert analysis and sizing

In this exercise, we will optimise the graph created in Objective 2 using Dynamo's DesignScript coding language. Replacing Dynamo nodes with DesignScript code can drastically improve the readability of complex graphs.

Improving on the DesignScript optimisation, the third graph dive into the use of Python Nodes, which infinitely extends Dynamo's capabilities by adding more complex decision-making statements. We will use the power of Python \ Iron Python to automate the calculation of pipe sizes based on peak flow values.

Exercise 4 – Create Civil 3D geometry from within the Dynamo workspace for documentation

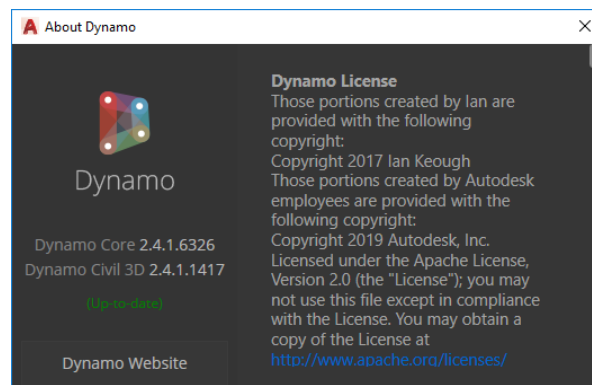
In our final exercise we will build on our learnings from the previous exercises and create a Dynamo graph which will attempt to build multiple culverts from within a single graph through reading data from an Excel table. This example will demonstrate the power of Dynamo and Excel to create a reusable graph which can adapt to virtually any design situation.

Finally, we will visualise the model in Civil 3D and plot the resulting geometry onto a Profile View and export the culvert information to an SAT file for import to Revit.

Software Requirements

For the exercises in this class, the following software was used:

- Civil 3D 2020.1
 - Dynamo 2.4.1



Learning Objective 1 - Leverage the Civil 3D Corridor model to create 3D drainage culverts in Dynamo

This section contains information on how to setup a Civil 3D project for use in Dynamo. Additionally, this section covers the geometric parameters required to create a detailed pipe culvert and headwall.

Civil 3D Project Setup

Before setting out a culvert using Dynamo, it is required to identify the appropriate location in Civil 3D. This can be achieved by turning on the necessary Civil 3D surfaces (Existing and Proposed) and visually identifying the contour flow. Additionally, the Water Drop can assist in determining the low flow path. Flow Path is located from the 'Analyze' tab on the ribbon (Analyze > Flow Paths > Water Drop)

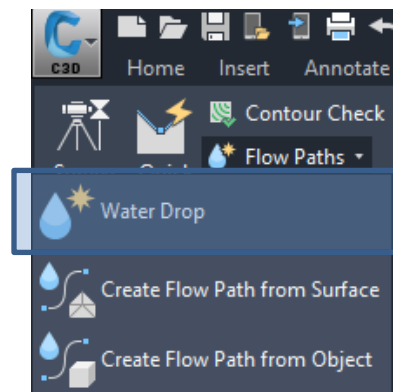


Figure 1: Culvert location using Water Drop

Civil 3D Corridor

It is important to setup the Civil 3D corridor model with the necessary Point Codes so Dynamo can extract the correct data from the model for use in the culvert analysis. Later in this document, where more automation is leveraged in the design calculations, various parameters (such as the size of the culvert, allowable headwater etc.) are driven by the vertical clearance. This vertical clearance is measured from the Hinge string (EH) to the Daylight (IA) string on the upstream side of the culvert.

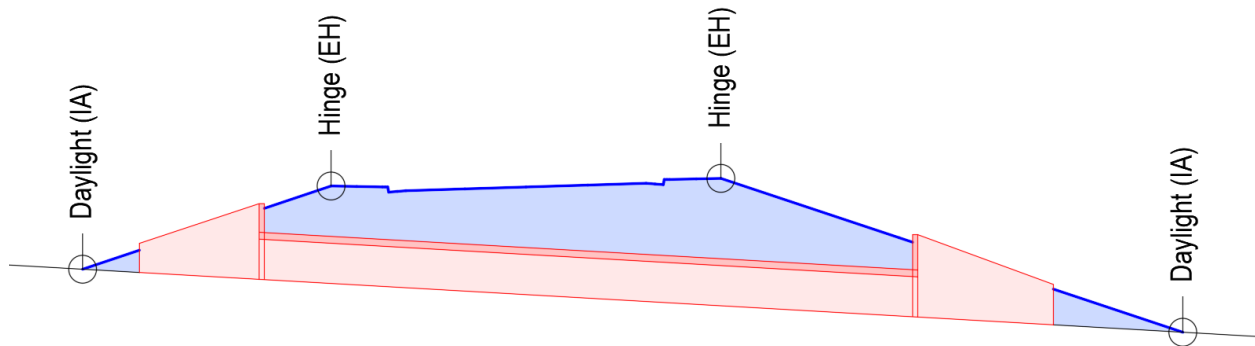


Figure 2: Corridor Codes used in the culvert design

Culvert Setout Parameters

When setting out the culvert, some of the common parameters used to define the location of the centreline include:

- Alignment String
- Station
- Skew

In Dynamo, these parameters will be setup as input parameters and affect the overall location and rotation of the culvert relative to the alignment.

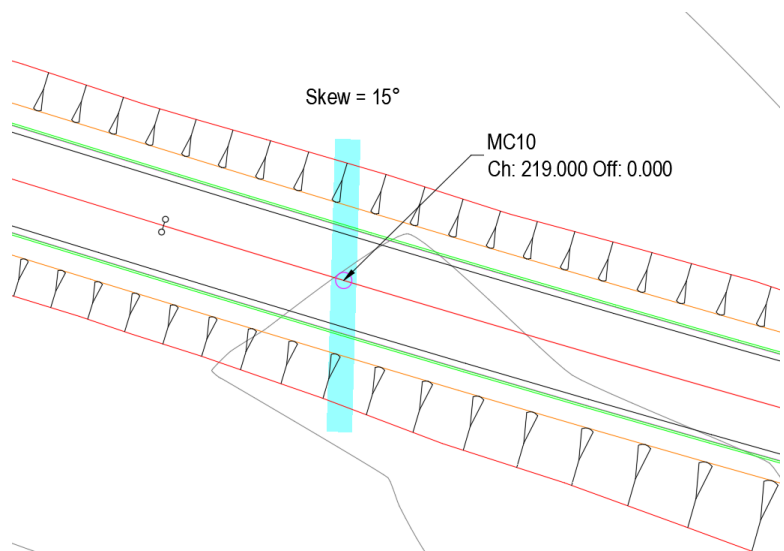


Figure 3: Culvert Setout Parameters

Culvert and Headwall Parameters

For the actual culvert itself, several parameters are required to generate more complex shapes that resemble real-world culvert dimensions, these include:

- Pipe Size
- Number of Cells
- Gap between Cells
- Culvert Height
- *Wingwall Length**
- *Wingwall Height**
- *Wingwall Angle**
- *Wingwall Thickness**

Note: These properties are duplicated 4x. Two wingwalls upstream, two wingwalls downstream

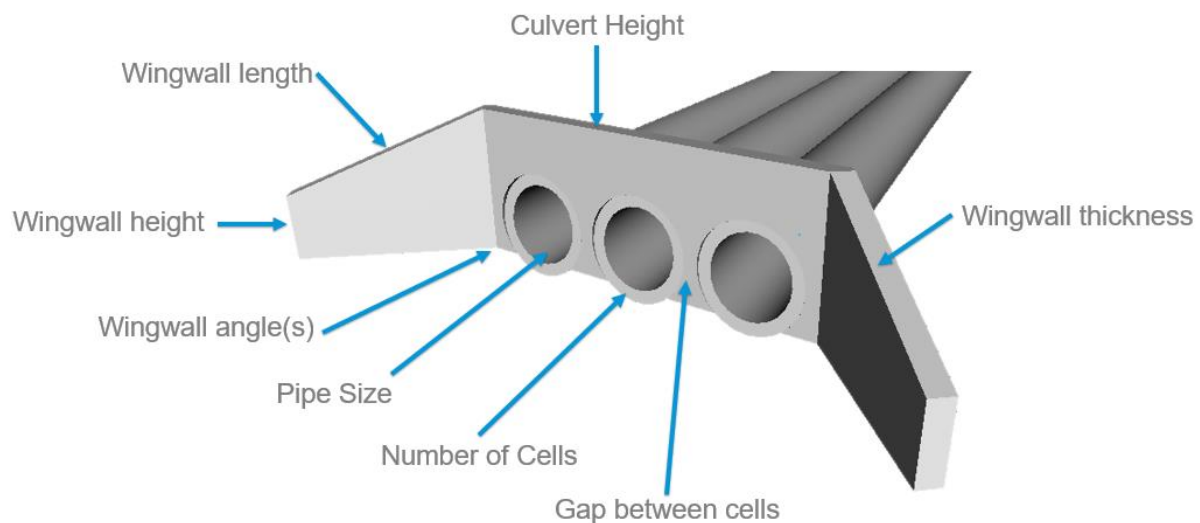


Figure 4: Culvert and Headwall parameters

Note that this exercise considers pipe culverts only. Similar methodologies and parameters can be applied to create box culverts and other culvert types.

Add design information

Dynamo Fundamentals

What is Dynamo?

Dynamo is an open-source visual programming tool that works with AutoCAD Civil 3D and Revit. Dynamo extends the power of Civil 3D by providing access to the Civil 3D API (Application Programming Interface) in a more accessible manner. Rather than typing code, with Dynamo you create programs by manipulating graphic elements called 'nodes' and joining them together with 'connectors' to build programming logic. Dynamo aids in the creation of new geometry in addition to improving workflow automation.

Add Dynamo information



Dynamo User Interface

The Dynamo User Interface is the main canvas where graphs are created. The UI consists several regions, as illustrated below:

Add UI information

Dynamo Nodes

Dynamo nodes are found in the library (located on the left side of the Dynamo canvas) and are grouped into logical categories.

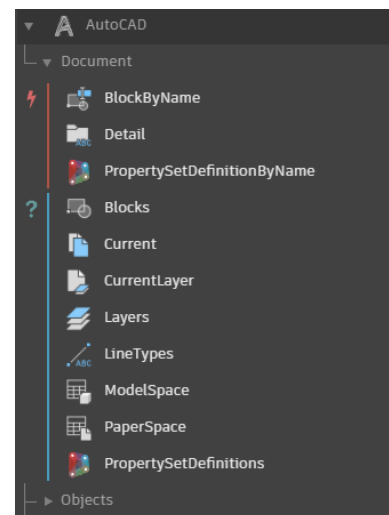
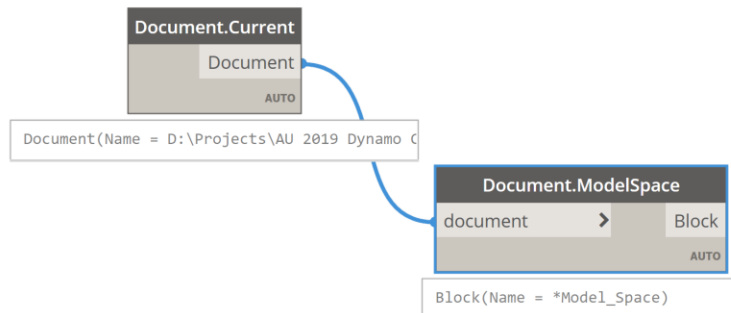
Expand the library category and sub-categories by left-clicking the mouse button. Select the desired node by simply left clicking to bring the node to the canvas.

AutoCAD

The AutoCAD library contains four Categories, **Document**, **Objects**, **PropertySets** and **Selection**

The **Document** library contains the nodes required to begin extraction of Civil 3D and AutoCAD data, specifically

- AutoCAD > Document > Current
- AutoCAD > Document > ModelSpace

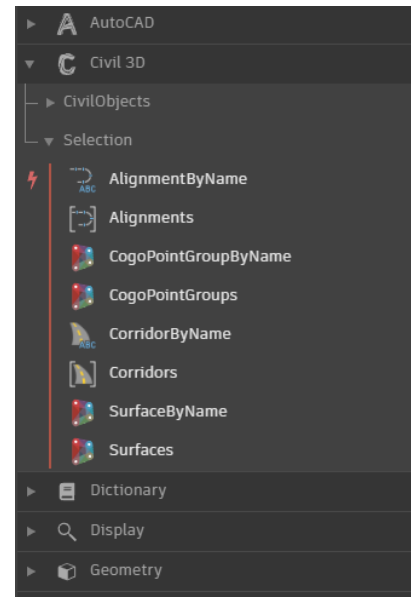
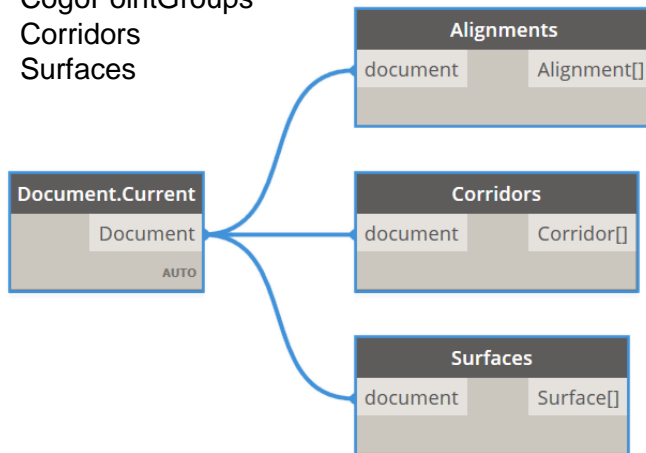


Civil 3D

The Civil 3D library contains only two Categories, **CivilObjects** and **Selection**

Select Civil 3D objects through the **Selection** category. Selectable objects include:

- Alignments
- CogoPointGroups
- Corridors
- Surfaces



Dynamo

The Dynamo installation contains literally hundreds of nodes, with the ability to download external packages to extend the capability through custom nodes. It is recommended to review the Dynamo Primer (<https://primer.dynamobim.org/>) to get an overview of what these nodes are capable of. The libraries are separated as follows:

- Dictionary
 - A dictionary is a special data type that uses a key-value pair to store data. These are used as an alternate to lists
- Display
 - A collection of nodes that control the display of geometry in Dynamo
- Geometry
 - Allows for the creation, editing and querying of all Dynamo geometry. Dynamo geometry includes elements such as Points, Curves, PolyCurves, Meshes, Surfaces, Solids. Additionally, abstract geometry such as Coordinate Systems, Planes and Vectors are also supported
- ImportExport
 - Allows interaction with external files, including Excel, CSV, JSON, as well as accessing File System objects, Images and Web Requests
- Input
 - Provides different methods in providing user-input into Dynamo, including Integers, Numbers, Strings and Dates
- List

- Provides nodes for all list manipulation, including creating, reversing, joining, sorting, grouping, counting and filtering to name a few.
Lists are one of the core concepts of Dynamo and a solid understanding of list structures is essential to understanding Dynamo.
- Math
 - These are basic commonly used math functions and logic operators used in the calculation of values
- Script
 - These nodes allow for enhanced flow control, and includes access to Code Blocks and Python nodes
- String
 - Allows for creating and manipulation of string (i.e. text) objects

Learning Objective 2 - Use Dynamo to create complex 3D drainage culvert models from user-defined parameters

This section contains information on how to create a culvert using the design parameters specified in Objective 1. This culvert will use read data from the Civil 3D model, including:

- Alignment and Profile
- Station
- Skew

Additionally, we will setup culvert parameters (i.e. pipe size, number of cells etc) to create Dynamo geometry which leverages Civil 3D corridor information.

Finally, we will investigate techniques to create and maintain tidy Dynamo graphs that are easy to read and amend. We will use all this information to create a single culvert based, in Civil 3D, with user-defined parameters

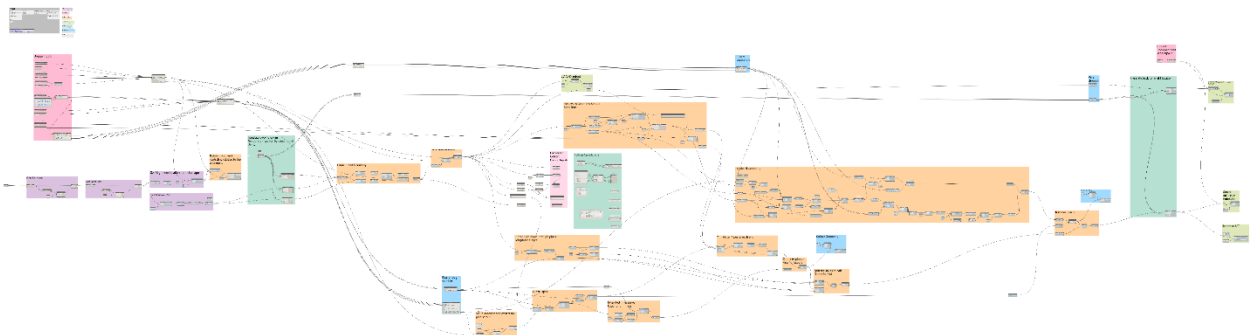


Figure 5: Manual Culvert Design Graph

Dynamo Graph Setup

Dynamo graphs should use a template as a starting point. The template is added to each graph to provide information to the intended user and can contain information such as:

- Author
- Version
- Instructions for use
- Known Issues
- External packages
- Group coding

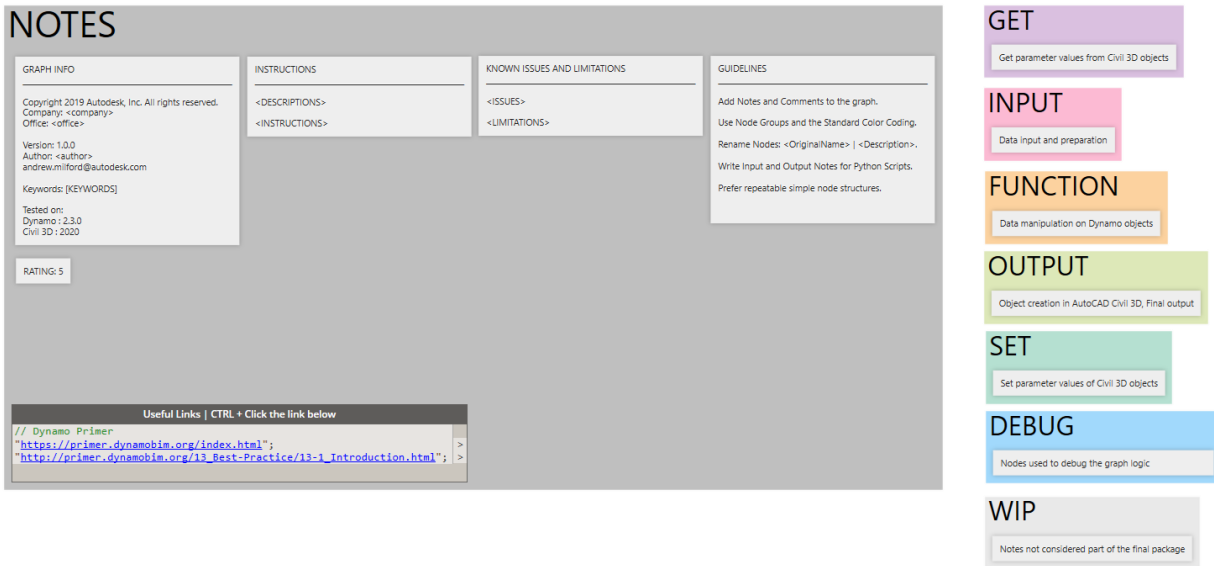


Figure 6: Dynamo Template

Add template information

Dynamo Graph Rules

Dynamo Graph Layout

Dynamo graphs should be set out to run in a left to right manner, with inputs collected on the left, functions and operations in the centre, and outputs towards the right.

Graph Warnings

When working with models of a Civil nature, coordinates tend to fall far from the origin (0,0). Dynamo cannot handle these large distances very well and will highlight a node with a warning when geometry is created too far from the origin, as shown below.

Add warning information

01_Culvert_2019 - Manual.dyn

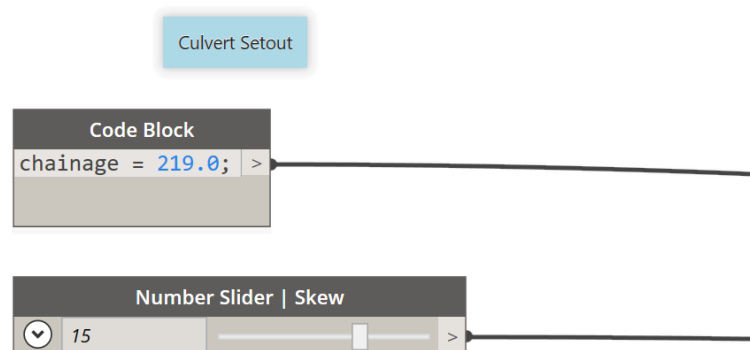
For the creation of the manual culvert, the following process was undertaken:

- Gather user-defined design inputs
- Get Civil 3D Corridor
- Get Corridor Baseline
- Get the Alignment location and skew
- Get the Corridor Featurelines (Hinge and Daylight)
- Create a setout line based on the Alignment station and skew
- Extract levels on the corridor along the culvert centreline
 - Slopes calculated from intersection of culvert and daylight strings
- Create culvert pipes (size, number and spacing)
- Create headwall setout and geometry in Dynamo
- Tidy headwall geometry
- Export Objects to AutoCAD / SAT

Important Dynamo Nodes

Some of the important steps in this process include:

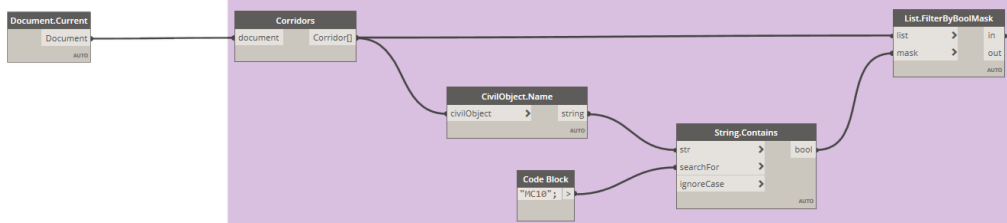
- **Number Slider**
 - A number slider allows users to enter values with a given range. The slider contains a start, end and step value.
- **Code Block**
 - Code blocks are probably the most useful of all Dynamo nodes, in that they allow any value to be entered. These are the equivalent of typing straight code into your Dynamo graph.



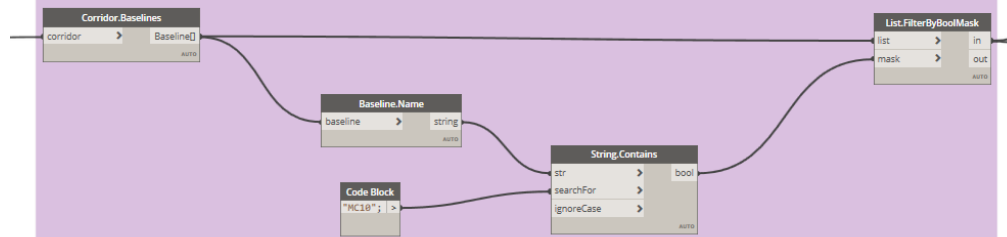
- **Corridors**
 - Extracts all corridors in the Civil 3D model

- **CivilObject.Name**
 - Gets the name of the Civil 3D object
- **String.Contains**
 - Check to see whether a specific string value exists within another string. This generates either a *true* or *false* value
- **List.FilterByBooleanMask**
 - Filters a list based on a corresponding list of true/false values. In the example, filters the corridor based on its name, the same methodology applies to extracting a baseline. For this node to work, an equal number of list and mask items is required.

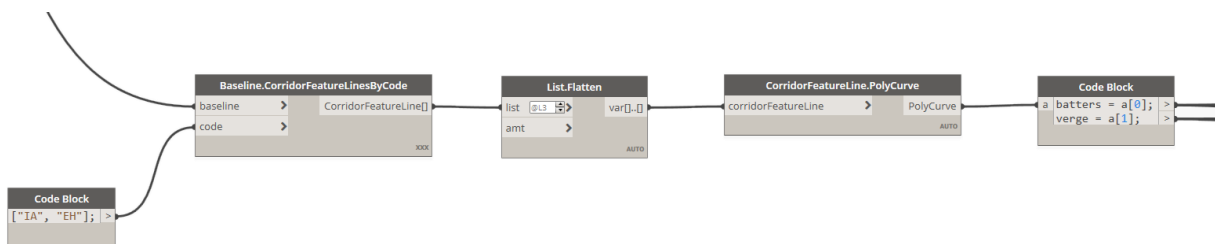
Get Corridor



Get Baseline

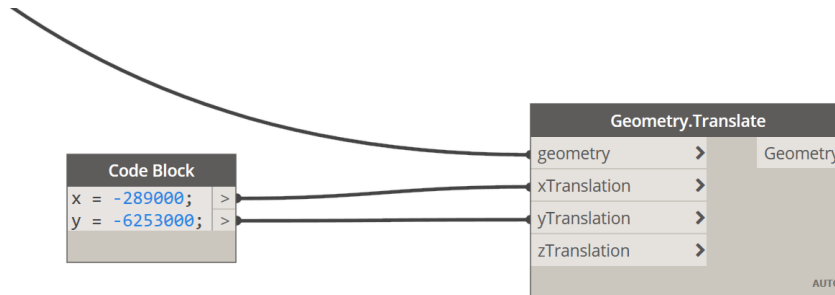


- **Baseline.GetFeatureLineByCode**
 - Gets a Civil 3D featureline object from the model, based on the passed-in code values
- **CorridorFeatureLine.PolyCurve**
 - Creates a Dynamo PolyCurve from a Civil 3D FeatureLine. The Dynamo geometry is then used downstream for geometry calculations.



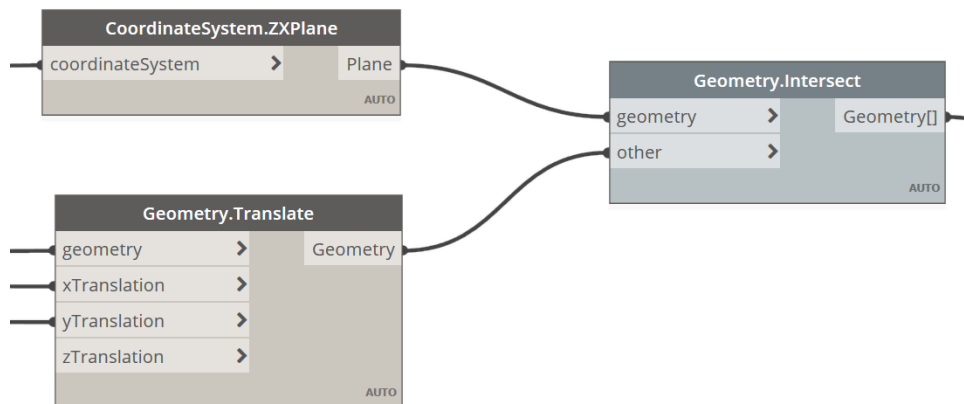
- **Geometry.Translate**

- Moves the geometry in any direction to a new location. In the example below the input geometry is moved -289000 in the 'X' direction and -6253000 in the 'Y' direction. (i.e. closer to the origin)



- **Geometry.Intersect**

- Takes two geometry inputs and will try to generate the intersection between the two passed-in objects. In the graph, the corridor featurelines are intersected with a Plane object to extract the 'cut' points.



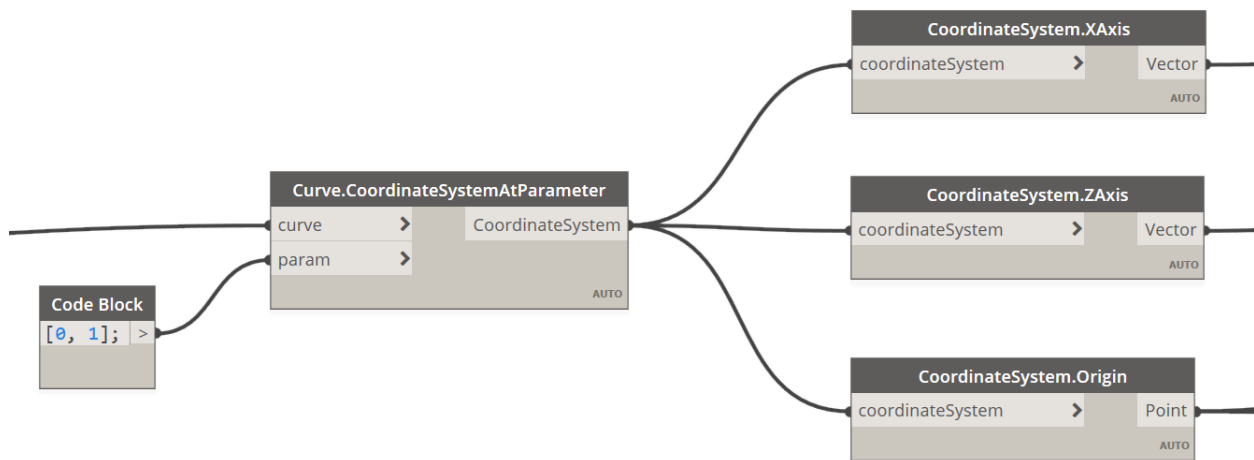
- **Curve.CoordinateSystemAtParameter**

- Creates a coordinate system along a curve at a specific parameter. The parameter value ranges between zero and one (0..1). A value of zero creates a coordinate system at the start of the curve, a value of one creates a coordinate system at the end of the curve. Additionally, a value of 0.5 places a coordinate system in the middle of the curve.

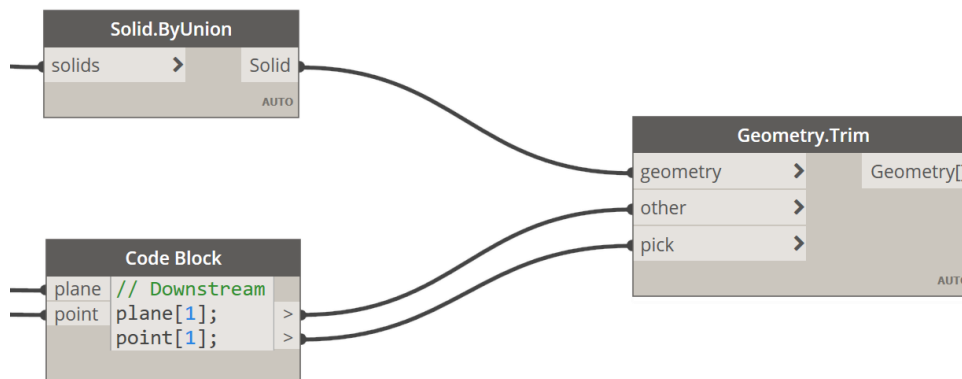
These are extremely useful, as the coordinate system is placed and oriented along the curve, with the 'X' axis projecting perpendicular to the curve, the 'Y'

axis projecting along the curve and the 'Z' axis projecting perpendicular upwards from the curve.

- **CoordinateSystem.Origin**
 - Creates a Dynamo point at the coordinate system origin
- **CoordinateSystem.XAxis**
 - Gets the X-Axis of the coordinate system. Similarly, you can extract the YAxis and ZAxis



- **Geometry.Trim**
 - Trim geometry to another geometry object. An point is also required to determine which side of the trimmed geometry to keep. For the culvert example, a pipe is trimmed back against a vertical plane lined up with the culvert headwall

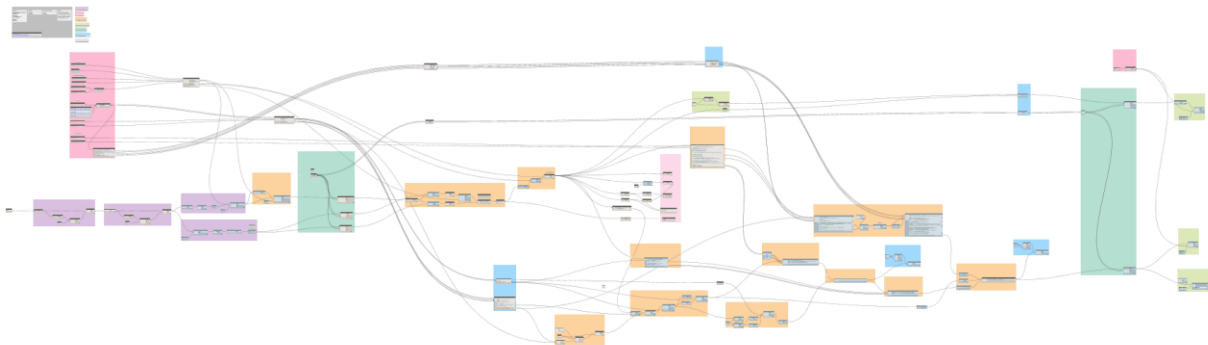


Learning Objective 3 - Explore Dynamo's Python node to undertake culvert analysis and sizing

We will now optimise the graph created in the previous chapter, using Dynamo's DesignScript coding language. Replacing Dynamo nodes with DesignScript code can drastically improve the readability of complex graphs.

Improving on the DesignScript optimisation, the third will graph dive into the use of Python Nodes, which infinitely extends Dynamo's capabilities by adding extended coding functionality. We will use the power of Python \ Iron Python to automate the calculation of pipe sizes based on peak flow values.

02_Culvert_2019 - DesignScript.dyn



DesignScript - Simplify your graphs

Add DesignScript Information

Node To Code

To better understand the DesignScript language, several resources are available online, including the Dynamo Primer

https://primer.dynamobim.org/07_Code-Block/7-2_DesignScript-syntax.html

However, to get a head start on understanding the basics of DesignScript, the easiest way is to select a group of nodes, right-click in an empty space on the workspace and select the 'Node to Code' option.

Instant code - it's that simple!

This function will convert all selected nodes to DesignScript and place this into a Code Block, assign some placeholder variables into the code blocks.

Please note that there is no 'Code to Node' function. A good way to achieve both (i.e. for reference), is to do the following:

- Copy the nodes to the clipboard (Ctrl-C)
- Node to Code
- Paste the original nodes back to the workspace (Ctrl-V)

There are a few important things worth noting about Node to Code:

- The code generate is not formatted in any way and can be difficult to read.
- Nominal variable names are used throughout and should be replaced with something more suitable.
- It is preferred to revisit the native Node to Code output and add comments to the code block
- Better still, once you have an understanding on how to code using DesignScript, begin by coding directly into a Code Block
- When previewing Code Blocks with the pushpin, only the last line is displayed. Watch nodes are required to view all other line outputs other than the last

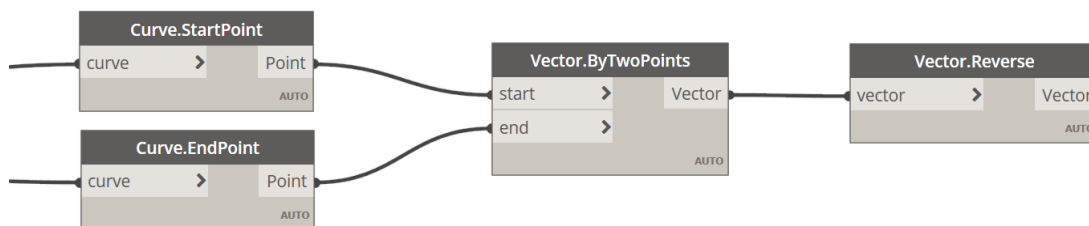
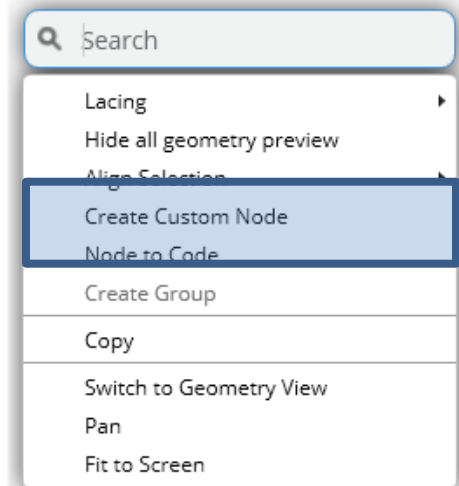


Figure 7: Original Dynamo Nodes

Code Block		
t1	point1 = Curve.StartPoint(t1);	>
	point2 = Curve.EndPoint(t1);	>
	vector1 = Vector.ByTwoPoints(point1, point2);	>
	vector2 = Vector.Reverse(vector1);	>

Figure 8: Original Node to Code output (no formatting)

Code Block		
crv	// Get the curve Start/End points	
	pt1 = Curve.StartPoint(crv);	>
	pt2 = Curve.EndPoint(crv);	>
	 // Get the forward/reverse vector	
	vec = Vector.ByTwoPoints(pt1, pt2);	>
	vecReverse = Vector.Reverse(vec);	>

Vector(X = 0.620, Y = 21.140, Z = 0.892, Length = 21.1

Figure 9: Formatted Node to Code Output

Code Blocks

The Code Block is probably the most versatile of all Dynamo Nodes, as it allows access to virtually any node (Out-of-the-box or custom nodes).

In addition to accessing node functions, Code Blocks also support more advanced programming methods, including basic math functions, looping and commenting.

Code Blocks are found in the **Script > Editor > Code Block** library. Alternatively, a code block can be placed in the workspace by double left-clicking in an empty place on the workspace. This method is preferred as it is much quicker to place than repeatedly navigating through the library

Code Block
Your code goes here

Add Code Block Information

Python – Unleash the power

The Python node allows Dynamo to access more extended functionality than what Dynamo provides OOTB.

The Python Script node uses a variant of the Python language called Iron Python (<https://ironpython.net/>). It is a C# open-source implementation of the Python language, which is tightly integrated with the .Net framework

Python Script node

The Python Script node is found in the **Script > Editor > Python Script** library

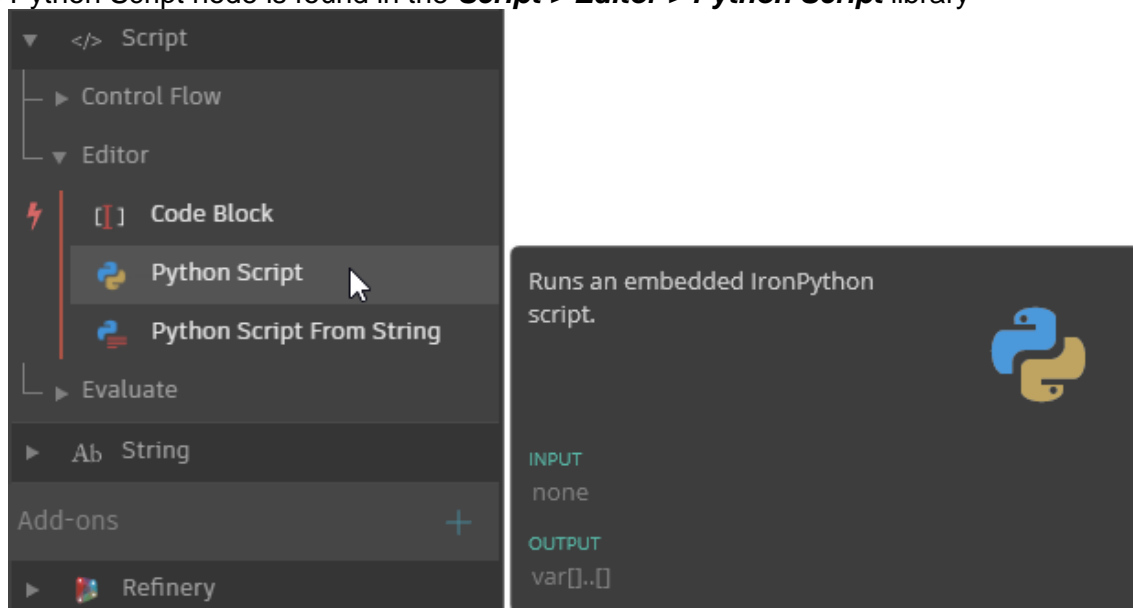


Figure 10: Python Node from the library

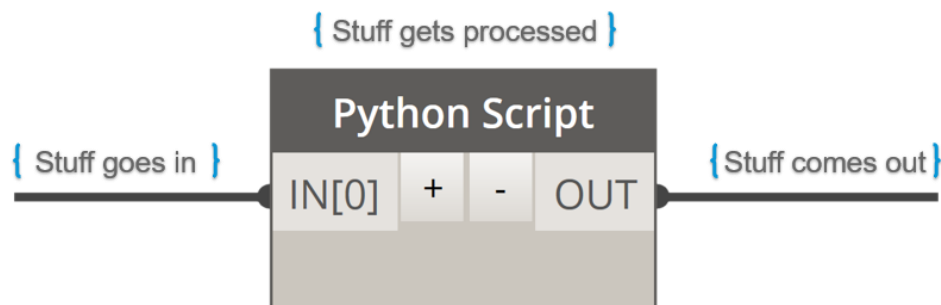


Figure 11: Python Script Node

The Python node can receive multiple inputs, indicated by the '**IN[0]**', '**IN[1]**' naming convention. The node can only contain one output port (labelled '**OUT**'). To export multiple values from a script, create a list of output values and return the list. It is recommended to comment on the incoming '**IN[]**' nodes, as these input names cannot be renamed in the node. In the image below, the input values are passed through a Code Block, with the description of each input clearly defined.

Python Boilerplate

A basic python script contains some 'boilerplate' code to get you started. A description of each line of code is illustrated below:

Add Civil 3D Template Boilerplate

03_Culvert_2019 - Python.dyn

This Dynamo graph is like the previous DesignScript example, with the exception that instead of the user inputting trial values for pipe size, sizing is automatically handled by numerous Python scripts inside the graph.

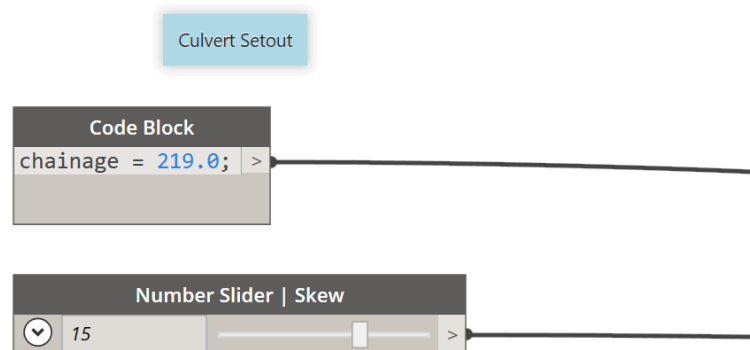
A toggle is now introduced to let the user determine whether to use the Python outputs, or the user-defined inputs to drive the Dynamo geometry.

Additionally, values calculated from the Python Scripts are exported to Excel for validation.

Another Excel sheet is generated for use in the following exercise, which creates multiple culverts from a single Excel worksheet.

Important Dynamo Nodes

- **Boolean**
 - A toggle that allows the user to select either a True or False value. This node is used to switch the resulting Dynamo /Civil 3D geometry from the 'user-defined' culvert *OR* the output geometry from the Python Script node.
- **Python Script**
 - Numerous Python script nodes are used to calculate not only the size and number of pipe cells, but also perform basic culvert analysis with regards to allowable headwater depths, tailwater, trial culvert selection, inlet/outlet control calculations, controlling headwater and outlet velocity
- **Code Block**
 - Code blocks are probably the most useful of all Dynamo nodes, in that they allow almost any expression to be entered. These are the equivalent of typing straight code into your Dynamo graph.



- **ExportExcel**
 - This node allows the export of data generated in the Dynamo graph out to an Excel file. The node required the File Path, Sheet name, starting column and row number and the data to write.
A Boolean 'Overwrite' toggle also allow data to continually overwrite the previous dataset.

Python Scripts used in the graph

Add all python script with descriptions

Learning Objective 4 - Create Civil 3D geometry from within the Dynamo workspace for documentation

In our final exercise we will build on our learnings from the previous exercises and create a Dynamo graph which will attempt to build multiple culverts from within a single graph through reading data from an Excel table. This example will demonstrate the power of Dynamo and Excel to create a reusable graph which can adapt to virtually any design situation.

Finally, we will visualise the model in Civil 3D and plot the resulting geometry onto a Profile View and export the culvert information to an SAT file for import to Revit

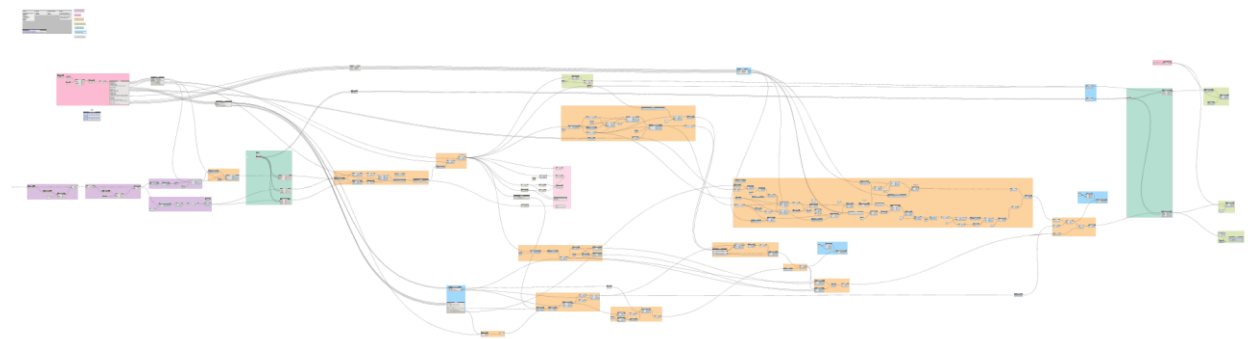


Figure 12: 04_Culvert_2019 - Excel.dyn

04_Culvert_2019 - Excel.dyn

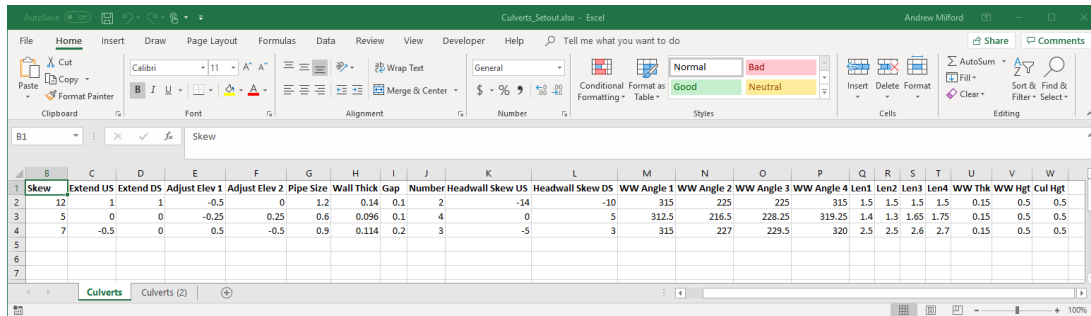
This Dynamo graph is similar to the first manual culvert graph, with the exception that the inputs are now handles by an Excel file, as opposed to a single set of parameters.

The Excel output from the previous Python exercise can also be copied across into a spreadsheet with multiple rows so that a central repository of culvert information is maintained

Excel Import

An Excel file is setup with headers to indicate the parameters that the user should complete before running

Each row in the graph, except the header, represents one culvert. The parameters used in the Excel file are identical to the parameters input into the first Dynamo graph (**Manual Culvert01_Culvert_2019 - Manual.dyn**)



1	Skew	Extend US	Extend DS	Adjust Elev 1	Adjust Elev 2	Pipe Size	Wall Thick	Gap	Number	Headwall Skew US	Headwall Skew DS	WW Angle 1	WW Angle 2	WW Angle 3	WW Angle 4	Len1	Len2	Len3	Len4	WW Thk	WW Hgt	Cul Hgt
2	12	1	1	-0.5	0	1.2	0.14	0.1	2	-14	-10	315	225	225	315	1.5	1.5	1.5	1.5	0.15	0.5	0.5
3	5	0	0	-0.25	0.25	0.6	0.096	0.1	4	0	5	312.5	216.5	228.25	319.25	1.4	1.3	1.65	1.75	0.15	0.5	0.5
4	7	-0.5	0	0.5	-0.5	0.9	0.114	0.2	3	-5	3	315	227	229.5	320	2.5	2.5	2.6	2.7	0.15	0.5	0.5
5																						
6																						
7																						

Figure 13: Import data from Excel to create multiple culverts

Managing Dynamo Lists

Output to AutoCAD

In addition to exporting solid objects to Civil 3D, two additional geometric elements are exported to assist in the setup of models in Revit. A simple circle and line element is drawn at the culvert setout point (i.e. upstream centreline).

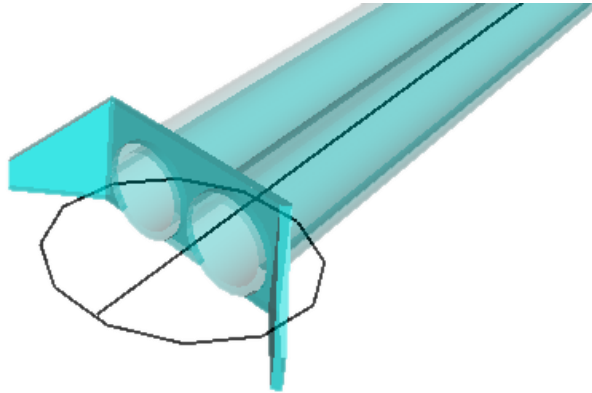


Figure 14: AutoCAD culvert setout geometry

In AutoCAD, WBlock out the two elements (circle and line) to a separate DWG file. This will be imported into Revit to acquire the coordinates and set the Project Base Point.

Civil 3D Profile and Section Views

The Solid outputs from Dynamo can be output to Civil 3D as AutoCAD Solids. These can be projected for use on Profile and Section Views, like any solid object.

- From the Home tab on the ribbon, Select **Home > Profile & Section Views > Project Objects to Profile View**

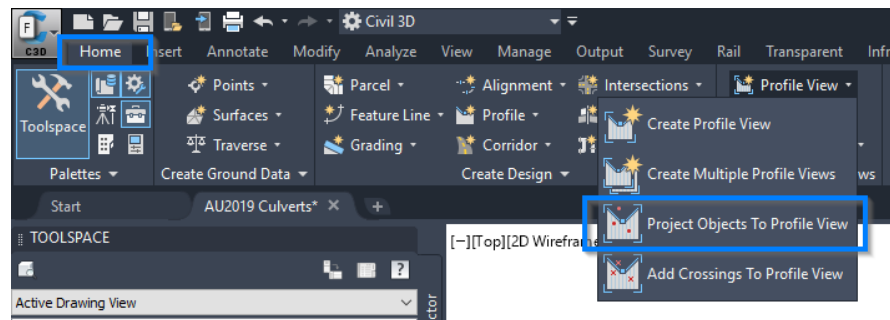


Figure 15: Project Objects to Profile View

- Select the culvert pipes.
- From the 'Project Objects to Profile View' dialog, change the style and label to suit your requirements
 - Click OK

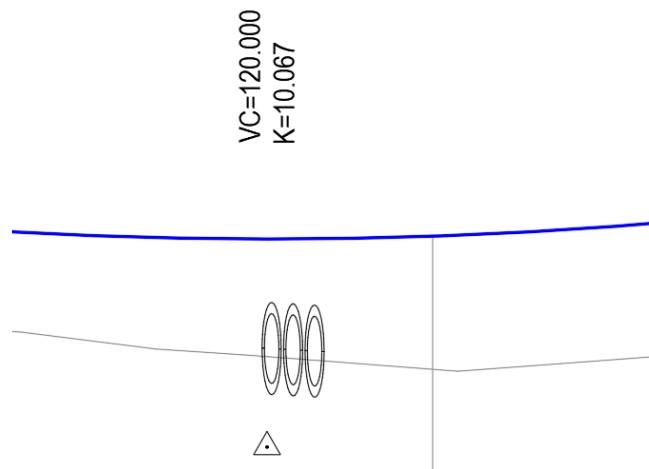


Figure 16: Solid pipes projected in Profile View

Output for Revit

Exporting of geometry for use in Revit is a bit limited, with the ExportToSAT node being the preferred option. This node takes the Dynamo geometry and a file name as its inputs and when run, will output the geometric SAT file to the desired location.

Import SAT file to Revit

Importing the SAT file requires some prior setup of the Revit model.

- Setup Revit project using any template. Ensure the units (UN) are set to metres (m)
- Move to a view which covers a larger extend of the project. 'Site' is usually good for this. In the 'Site' view, change the View Range to extend vertically

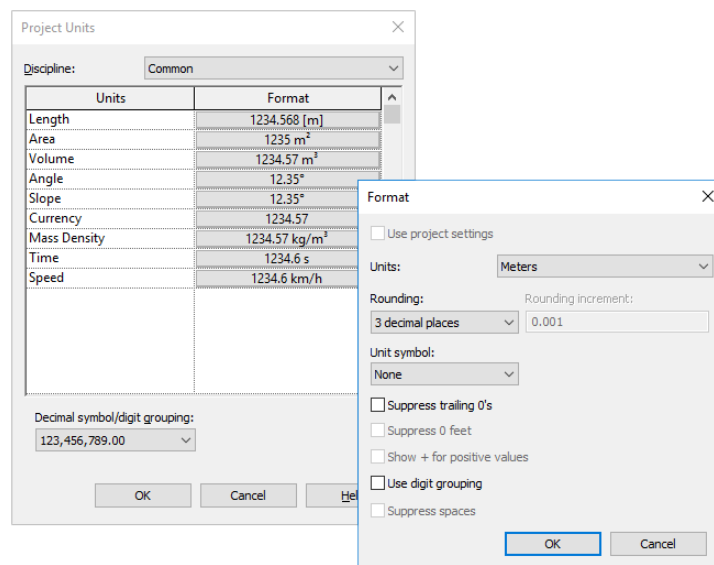


Figure 17: Setting Project units

Important Dynamo Nodes

Some of the important nodes used in this process include:

- **ImportExcel**
 - This node allows the import of data from an Excel file into a Dynamo list. The node requires a File Path and a File From Path node, a Sheet name. Optionally, data can be read in as strings and a Boolean toggle can determine whether to show Excel or not.
- **File Path**
 - Allows the selection of a file from the system.
- **File From Path**
 - Create a File object from the path. *Note:* this node **MUST** be used in conjunction with the File Path, otherwise the ImportExcel node will not read the file correctly.
- **Deconstruct**
 - Deconstruct will take a list and extract the first item in the list as a separate item, then place the remaining items into another list. This is handy when importing Excel data with a header.
- **Transpose**
 - Transpose will take a list of elements and 'flip' the list. For example, transposing a 2x3 list of elements
 - [1, 2, 3]
 - ["A", "B", "C"]
 - will become a 3x2 list
 - [1, "A"]
 - [2, "B"]
 - [3, "C"]
- **ByGeometry (AutoCAD > Objects > Object)**
 - Creates AutoCAD geometry from the passed-in Dynamo geometry. In these examples solids, lines and circles are created in Dynamo, and the exported to AutoCAD for documentation.
- **ExportToSAT**
 - Exports the solid culvert geometry to an external SAT file. This SAT file can then be read into Revit.
- **Code Block**
 - Code blocks are used extensively in this graph. The data coming from the Excel file is taken through a code block and separated into their respective columns. Variable names are assigned to maintain code readability and avoid confusion.

Conclusion

Using a combination of Civil 3D's model information and Dynamo's visual scripting capabilities, we have been able to create a set of re-useable graphs which allow us to create quick and dynamic drainage culverts, taking into consideration design principles that would traditionally take hours to compute and analyse.

The flexibility of Dynamo, and its ability to read and write Excel files, allows us to extract and store parameters for later reuse and wider team distribution.

This presentation has just been a small example of the power and flexibility of the Civil 3D& Dynamo interface, and hopefully will inspire you to create some amazing tools.

All the material available in the dataset is provided as open source so feel free to modify it and use it in your own applications. Feel free to contact me on LinkedIn to share workflows and ideas.

Good luck

Andrew.

Additional Resources

DynamoBIM

<https://dynamobim.org/>

Dynamo Primer

<http://dynamoprimer.com/en/>

GitHub/DynamoDS

<https://github.com/DynamoDS/Dynamo>

Blogs

<http://dynamobim.org/blog/>

Autodesk Knowledge Network

<https://knowledge.autodesk.com/support/civil-3d/learn-explore/caas/CloudHelp/cloudhelp/2020/ENU/Civil3D-UserGuide/files/GUID-E2122814-1957-4108-9BBF-0AD6AF1A63CB-htm.html>