
Essay Two

Parametric Truss Analysis using Graphic Statics

Laura Hannigan

14th June 2015

University of Cambridge

Interdisciplinary Design for the Built Environment

Word count: 2731

Contents

Abstract.....	3
Introduction	4
Literature review.....	6
Methodology.....	8
Results.....	11
Discussion and Further Research.....	15
Conclusions	16
References	17
Appendix	19
Developed script using Visual Basic in Grasshopper.....	19

Abstract

Graphic statics is a graphical technique of calculating a structure's internal forces developed in the early 1800s which saw its demise in the 1900s with the advent of the algebraic approach to structural calculations and computing power. Computing power has continued to grow exponentially since and 3D parametric modelling tools such as Grasshopper, a graphical algorithm editor for the 3D computer-aided design programme Rhino, have gained increasingly popularity amongst architects and engineers. The industry has reverted to a very graphical way of designing structures which has regenerated interest in graphical solutions to provide a greater understanding of the forces within a structure (Van Mele et al. 2012; Beghini et al. 2014; Fivet & Zastavni 2013). However, some of the tools developed which aim to capitalise on the benefits of graphic statics are limited in their applicability to capture user-defined problems ((BLOCK Research Group 2012; Greenwold 2003; Philippe 2005; Shearer 2010). This paper presents a computational tool which has the aim of increasing graphic statics applicability and appeal in today's world of 3D graphical designing. Visual Basic Scripting is used within Grasshopper, to script the basic steps of graphic statics and assess its applicability with the latest 3D modelling techniques to aid the understanding of structures. Despite some limitations associated with reading the results, this method presents the potential to offer a quicker, adaptable, workflow compared to the use of structural analysis software for concept truss design.

Introduction

'A new "graphical statics" has appeared, not in the sense of the automation and visual presentation of Culmann's graphical statics, but rather in the form of graphic displays and animated simulations of mechanical relationships and processes. This is a decisive step towards the evolution of constructions and to loadbearing structure synthesis, to a new type of structural doctrine.' (Kurrer 2008)

Graphic statics is a graphical method, developed in the early 1800s, of analysing a structure using equilibrium-based vector calculus, drawn to scale, to analyse axially-loaded structures such as trusses, cables, and arches. The method provides a visualisation of the forces within a structure and their dependencies.

The external loads on the structure are initially drawn to a desired scale to form a load line as shown in Figure 1. For truss analysis, each joint is then considered individually. Lines parallel to the truss members at the node considered are drawn at the relevant intersections on the load line to form the force polygon at that node. The force in the truss members are given by the length of the corresponding vector in the force polygon. Figure 2 illustrates the force polygon construction of a six-bay pratt truss. A thorough introduction to the technique and it's methodology can be found in *Forms and Forces* (Allen & Zalewski 2009).

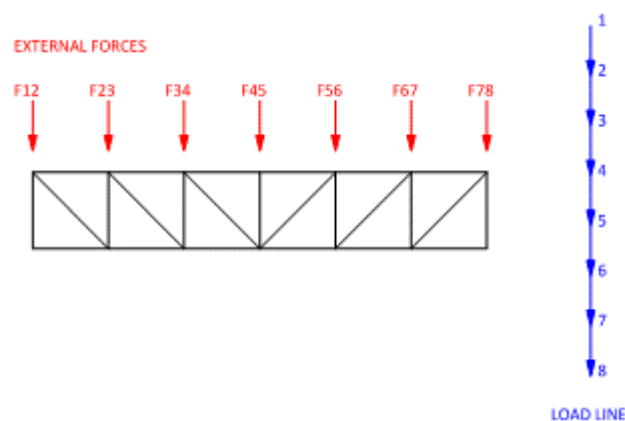


Figure 1 Load line

The technique saw its demise in the 1900s with the advent of the algebraic approach to structural calculations and computing power. Computing power has continued to grow exponentially since and the industry has reverted to a very graphical way of designing structures which has regenerated interest in graphical solutions to provide a greater understanding of the forces within a structure (Van Mele et al. 2012; Beghini et al. 2014; Fivet & Zastavni 2013). This paper investigates the place of graphic statics in today's world of 3D graphical design. A review of existing literature is presented, followed by a description of the script developed using graphic static methods to eliminate the manual construction process involved. This is followed by a discussion of the resulting tool, some suggestions for further research and conclusions.

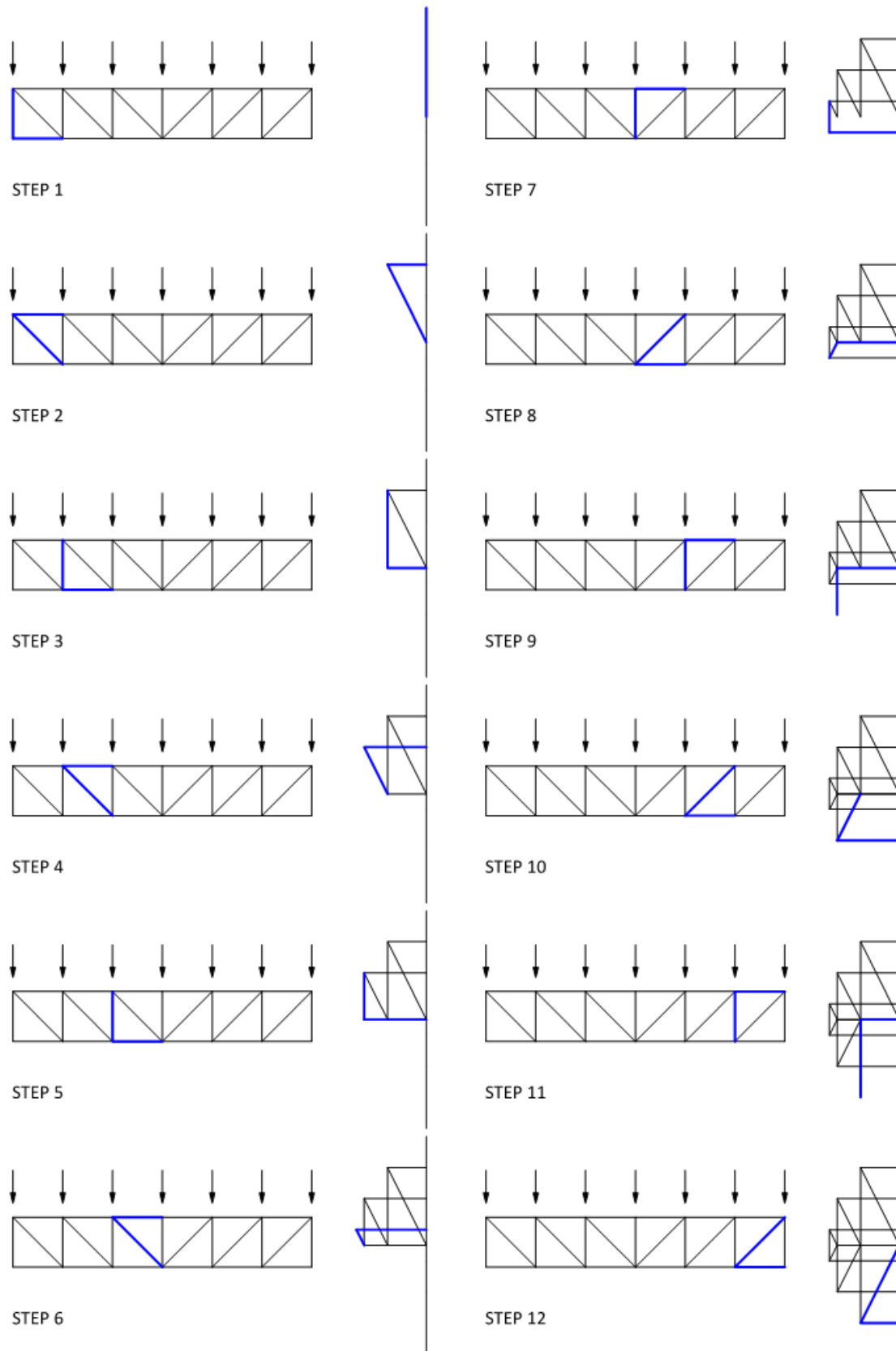


Figure 2 Force polygon construction for a six-bay pratt truss

Literature review

While the projective relationship of the form and force polygon was established by Varignon in 1725 (Figure 3), the first definitive, comprehensive collection of graphical techniques was Culmann's *Graphische Statik* (*Graphic Statics*) (Culmann 1866) which formed a milestone in the theory's definition. The method formed the basis of structural theory in the late 19th century, with the Eiffel Tower being a notable example of its use, co-designed by Maurice Koechlin, a student of Culmann (Kurrer 2008).

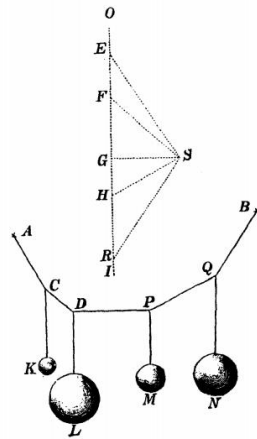


Figure 3 Funicular polygon and force polygon (Varignon 1725)

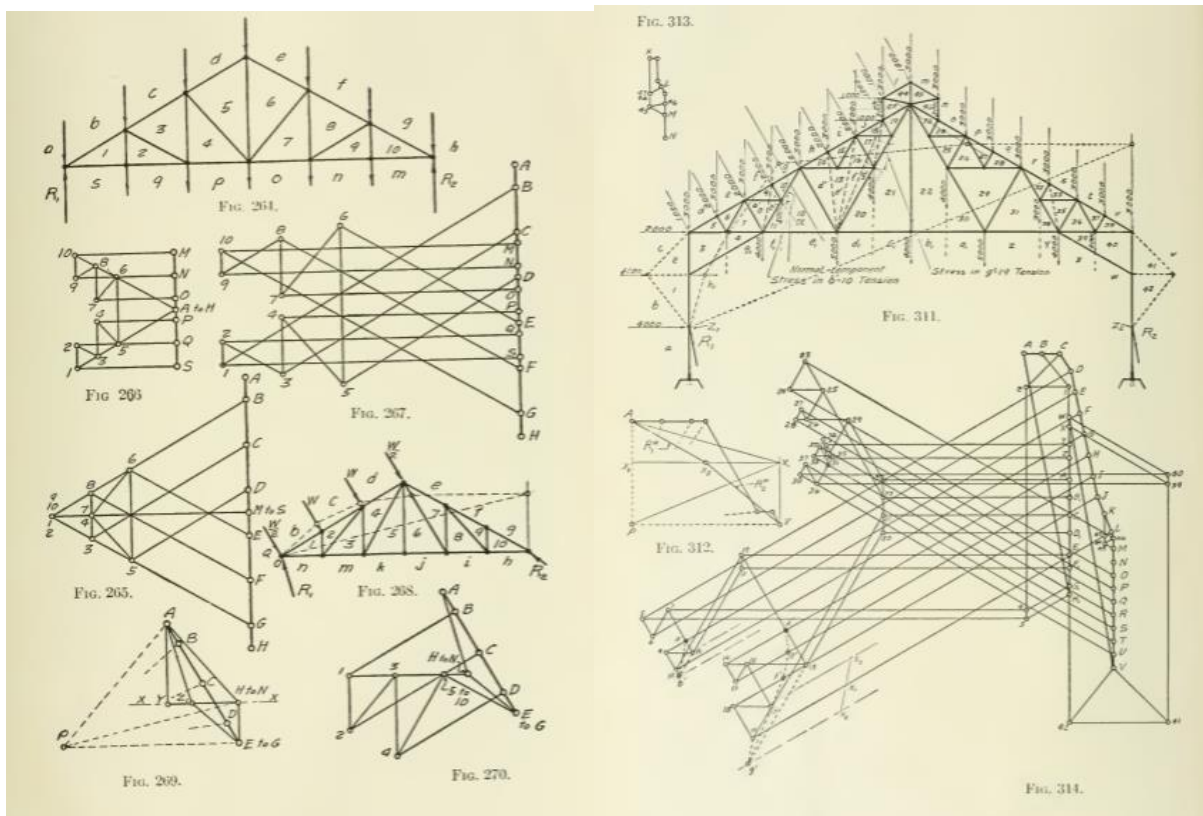


Figure 4 Graphic static solutions (Wolfe 1921)

However, the method's use is limited to specific statically determinate systems and has limited scope for expansion on a scientific basis. This, combined with the increasing popularity of linear algebra and analytical calculation methods, saw a decline in the use of graphic statics from 1900. Müller-Breslau's *Graphische Statik der Baukonstruktionen* (Graphical Statics of Structural Theory), which adopts the principle of virtual forces, includes graphic statics diagrams as mere accompaniment while by 1925 they are completely replaced by algebraic calculations in Martin Grüning's *Statik des ebenen Tragwerkes* (Structural Analysis of Plane Loadbearing Structures) (Müller-Breslau 1903; Grüning 1925; Kurrer 2008).

The 1930s saw the development of the first computer by the engineer Konrad Zuse, which was the precursor to rapid advances in engineering numerical methods and computing power has continued to grow exponentially to today. 3D parametric modelling tools such as Grasshopper, a graphical algorithm editor for the 3D computer-aided design programme Rhino, have gained increasingly popularity amongst architects and engineers. A number of other freely available 'plug-ins'¹ for Rhino, such as Karamba, Millipede and Kangaroo, perform preliminary structural analysis with the results easily and quickly visualised within the Rhino interface. However, the results of these tools are not always easily understood, or easy to interrogate and interpret, through the analytical formulae adopted which are embedded in the definition. A graphical solution provides a great benefit in this respect with the output readily understood, and if it's shortcoming of manual, sometimes tedious construction, could be addressed, it could play an important role in the development of understandable concept design solutions for 2D structures.

In light of this, there has been renewed interest in graphic statics due to its simplicity, graphical approach and because it provides a method through which interdependencies between geometry and force and geometry can be understood.

Active Statics, InteractiveThrust and eQUILIBRIUM are web-based applications illustrating graphic statics principles with a number of predefined structural forms (Greenwold 2003; Philippe 2005; BLOCK Research Group 2012). While eQUILIBRIUM provides a greater number of forms, greater flexibility and interactivity than Active Statics and InteractiveThrust, all three platforms are primarily educational tools and provide limited design applications. A review of current practice in conceptual structural design by Rolvink et al. (2014) noted that Active Statics and eQUILIBRIUM are not flexible enough to contribute to a user defined problem. Their formation was to provide a graphical method of understanding structures via computers, with the most recent application, eQUILIBRIUM, described as a 'learning platform for structural design' (BLOCK Research Group 2012). They do however highlight how computing offers 'new possibilities to an ancient field of research' (Block 2005). User-defined problems are permitted with scripts developed for use with Rhino (Shearer 2010), however the process is manual, time-consuming, and the end result is not parametric. This limits it's uptake in practice, past use as an educational tools, since rapid generation of design alternatives is key in conceptual design.

¹ A plug-in is a component that adds a specific feature to an existing software application

Methodology

A drawback in the use of graphics statics is the sometimes tedious and time consuming process of constructing the force diagram. Furthermore, the drawing process must be completed again for any change in the form, such as a change in depth. While some tools exist for implementing the method via computing, they are mainly designed as educational tools rather than user-defined design.

To improve the applicability and interactivity of the graphics static method, eliminate the manual construction process, and provide a broad platform for user-defined problems a script was developed for the graphical solution of a 2D truss using Visual Basic Scripting. The truss in this instance was chosen due to its common use in engineering and its suitability for simply demonstrating the key principles of the method. A parametric definition was also developed to avoid the redrawing process when investigating alternative design options. With parametric design, user-defined variable parameters are utilised to allow regeneration of a structure for various options, while maintaining the structural philosophy.

The script consists of three main parts:

- I. Construction of the truss geometry
- II. Plotting of the load line points
- III. Construction of the corresponding force polygon

The construction of the truss geometry is established with variable inputs which allow the user to change the truss parameters of length, depth and number of bays to see the corresponding force diagram update in real-time. Defining the truss geometry ensures the members are constructed in a certain sequence, as shown in Figure 5, which ensures compatibility with the force polygon construction script.

As previously discussed, the force polygon is constructed by methodically moving from joint to joint, plotting two member forces at a time. The procedure is completed in a specific joint order, moving to the joint with a maximum of two unsolved member forces each time. Due to the repetitive nature involved in constructing the force polygon, the technique lends itself to being scripted using a loop function. To script the construction, the construction points and lines of the force polygon were determined using coordinate geometry as illustrated in Figure 6 . At each joint, the intersection point of the two members under consideration was found using their slopes, which is known from the truss geometry definition. The corresponding force vectors are then drawn to this intersection point, from the relevant points on the load line, to form the joint's force polygon. This construction is then repeated for each node, with the previous node's intersection point used in the construction of the following node's force polygon.

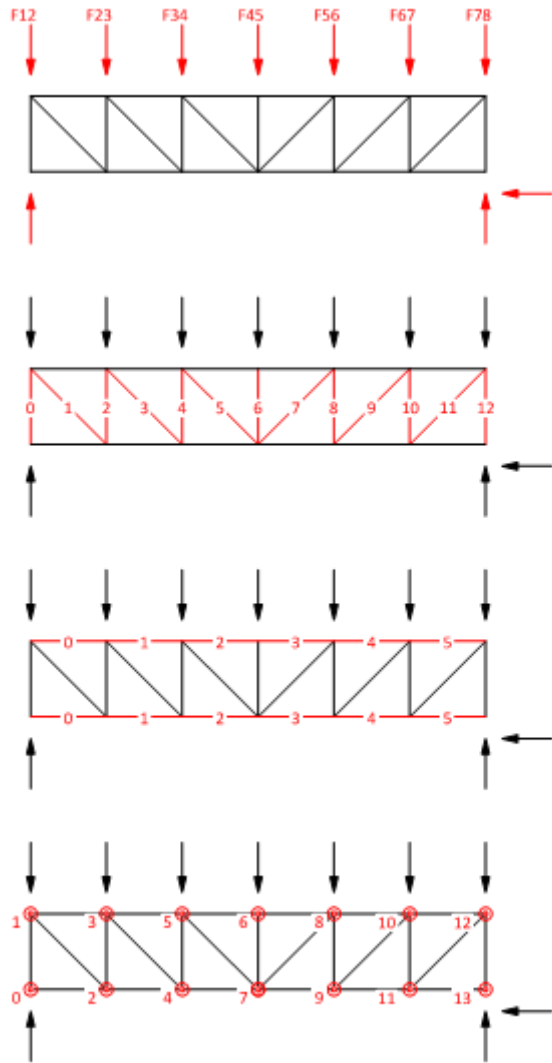


Figure 5 Order of forces, members, and joints required for use with the developed script

Intersection point occurs when:
 $y_1 = y_2$

$$\Rightarrow m_1x + c_1 = m_2x + c_2$$

$$(m_1 - m_2)x = c_2 - c_1$$

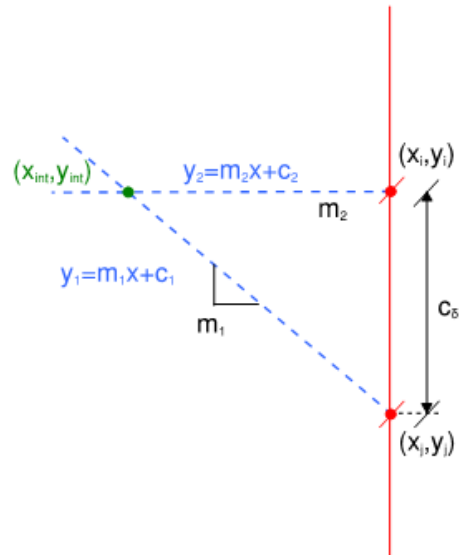
where:
 $c_2 - c_1 = c_0 = y_i - y_j$

$$x = c_0 / (m_1 - m_2)$$

$$y = m_1x + c_1 = y = m_2x + c_2$$

$$\Rightarrow x_{int} = x_1 + x = x_1 + x$$

$$\Rightarrow y_{int} = y_1 + y (= y_2 + y)$$



Intersection point occurs when:
 $y_1 = y_2$

$$\Rightarrow m_1x + c_1 = m_2x + c_2$$

$$(m_1 - m_2)x = c_2 - c_1$$

where:
 $c_2 - c_1 = y_2 - y_1$

$$x = (y_2 - y_1) / (m_1 - m_2)$$

$$y = m_1x + c_1 = y = m_2x + c_2$$

$$\Rightarrow x_{int2} = x_1 + x (= x_2 + x)$$

$$\Rightarrow y_{int2} = y_1 + y (= y_2 + y)$$

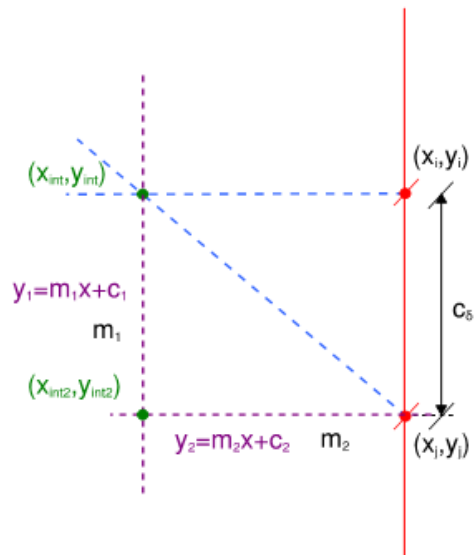


Figure 6 Coordinate geometry principles used in the construction of the force polygon

The script was implemented in Grasshopper with the output visualised in Rhino. Rhino was considered an ideal program to use as it is popular amongst both architects and engineers who are the target end users of this tool. The truss geometry can be exported from Rhino to a CAD format for import into a structural analysis software if further, more detailed analysis is required. Also, implementing the script in Grasshopper permits a parametric truss definition where the variables can be adjusted and the result viewed in real-time. The Grasshopper interface is shown in Figure 7 with the user-defined inputs highlighted in blue. The developed code, given in the Appendix, is contained within the components highlighted in green. While the output highlighted in pink is noted as the truss members and the corresponding force vectors, any element generated through the construction process can be output such as node points or force polygon points.

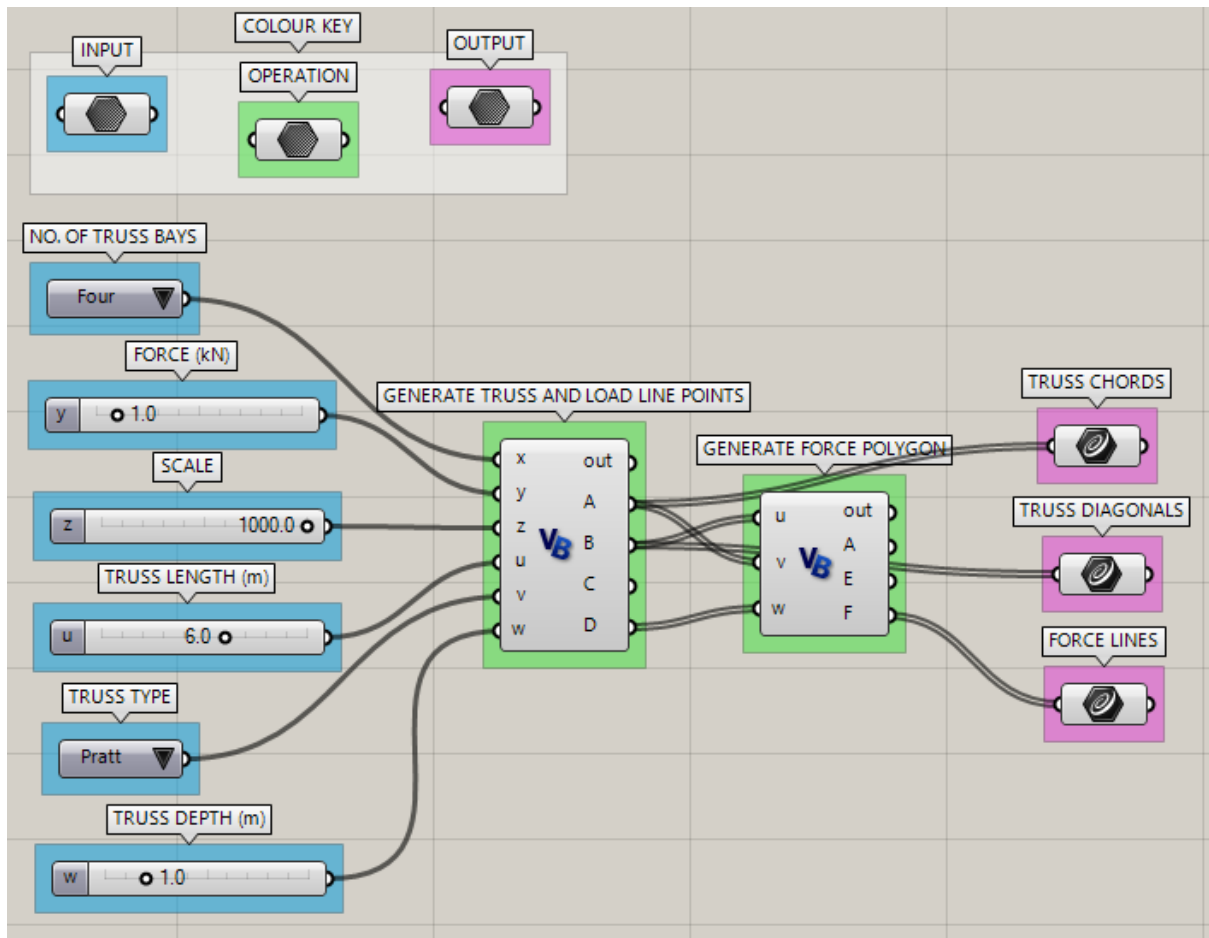


Figure 7 Grasshopper interface showing the user-defined input, operation components containing the developed script, and output

Results

The results of a number of different truss configurations are shown in Figure 8 to Figure 12. It can be seen that varying the defined user-inputs results in an updated force diagram eliminating any manual construction of the force polygon by the user.

The implementation process is straight-forward with the visualisation of the truss and force polygon in Rhino updating directly with a change in one of the parameters in Grasshopper.

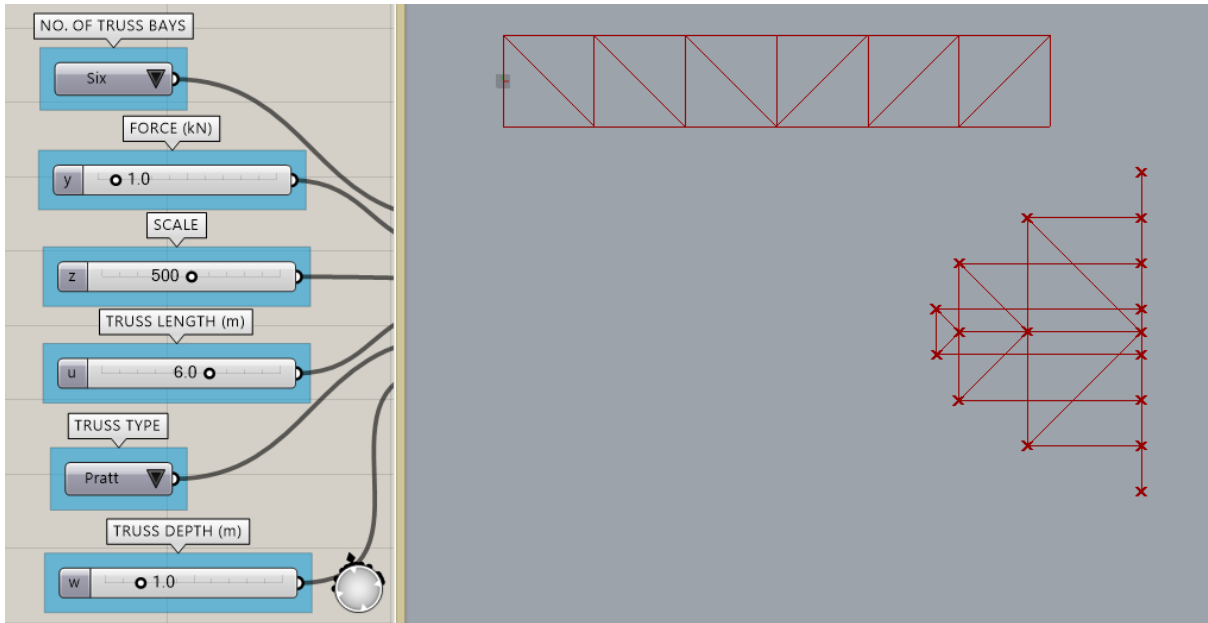


Figure 8 Six-bay pratt truss, 1m deep (Grasshopper and Rhino interface)

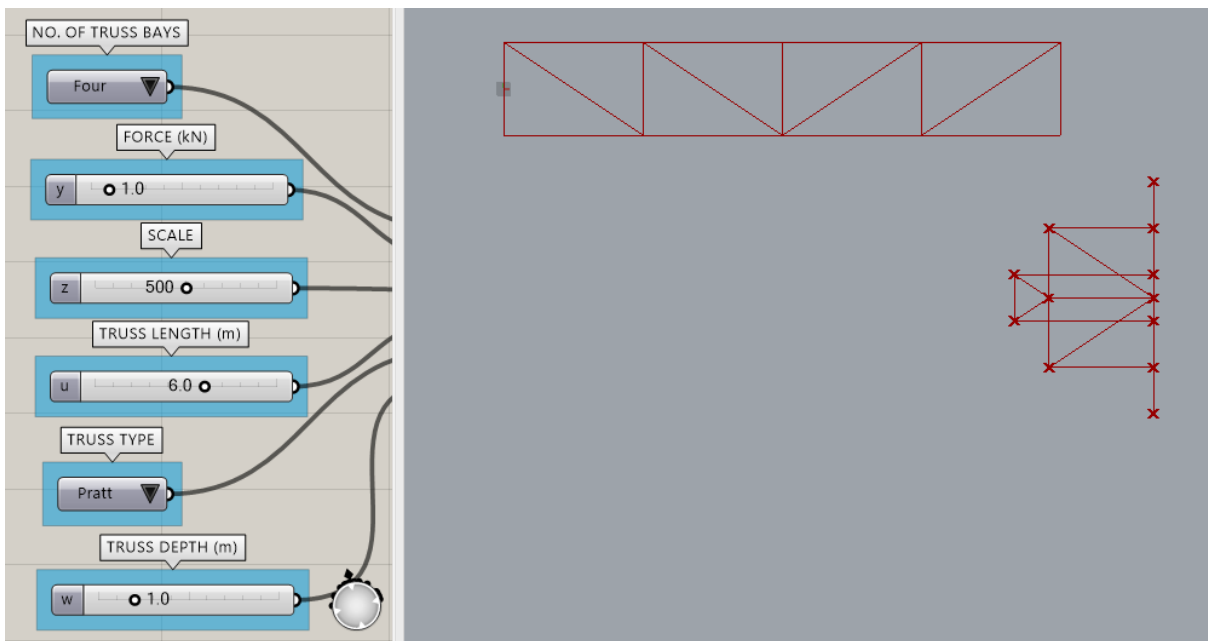


Figure 9 Four-bay pratt truss, 1m deep (Grasshopper and Rhino interface)

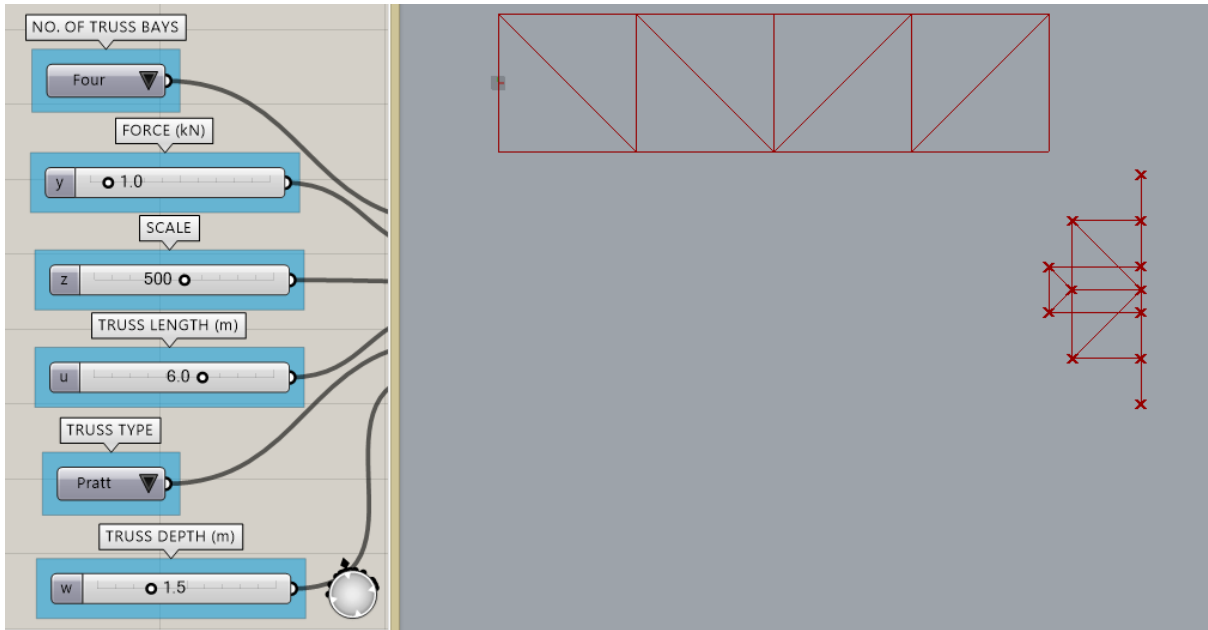


Figure 10 Four-bay pratt truss, 1.5m deep (Grasshopper and Rhino interface)

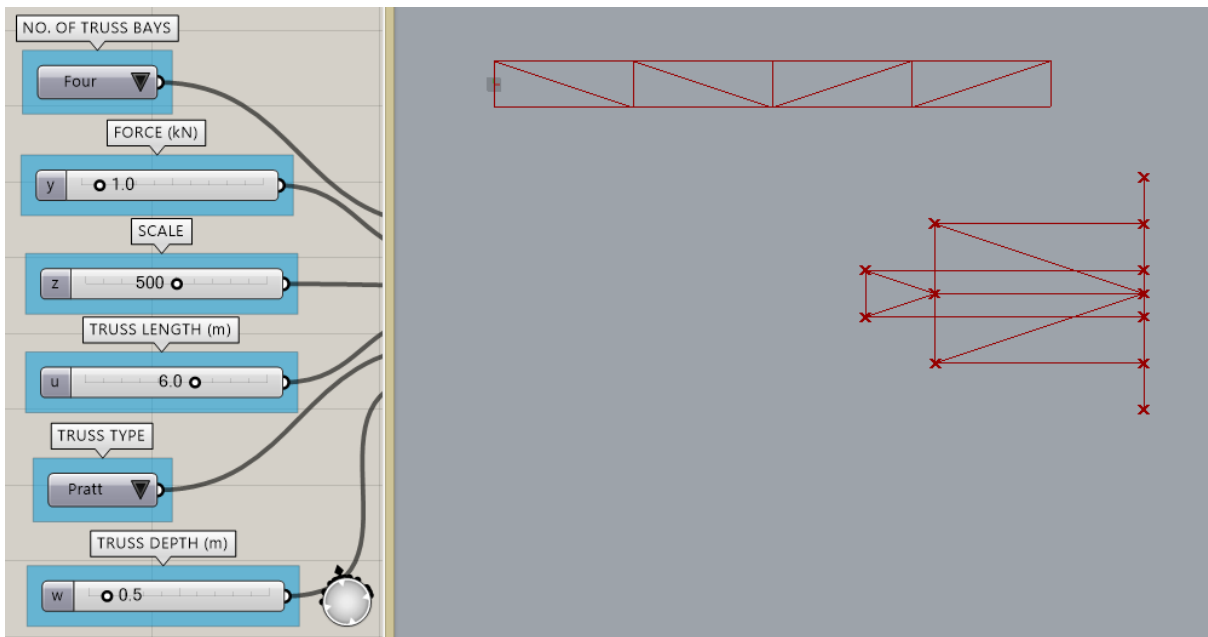


Figure 11 Four-bay pratt truss, 0.5m deep (Grasshopper and Rhino interface)

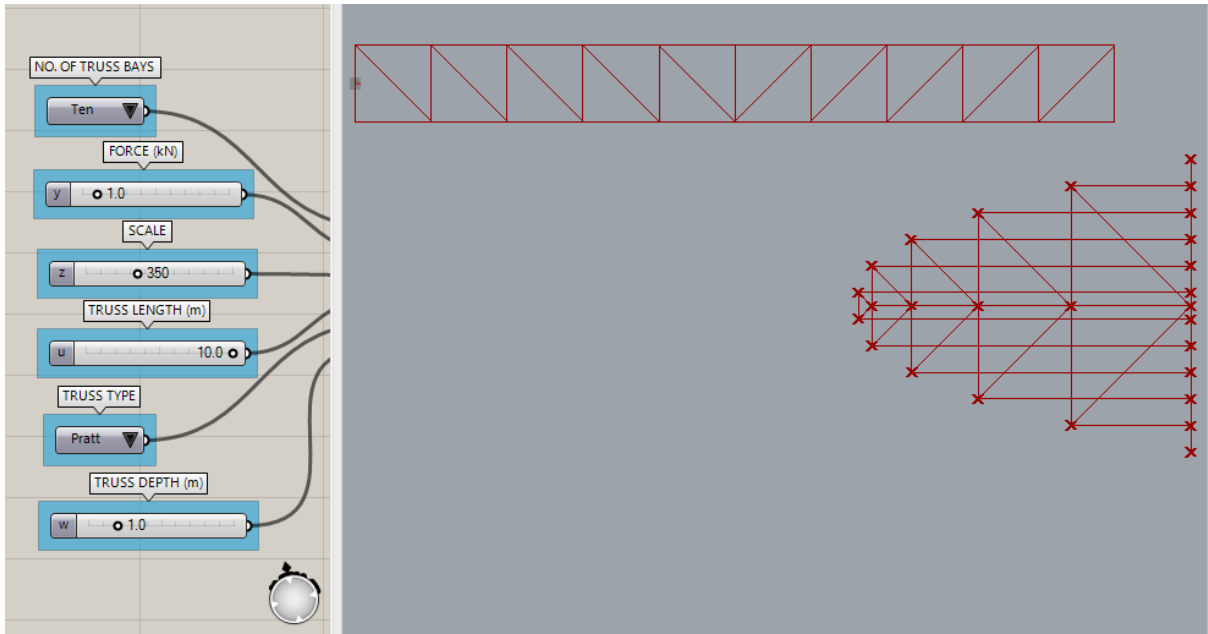


Figure 12 Ten-bay pratt truss, 1m deep (Grasshopper and Rhino interface)

Discussion and Further Research

Typically truss analysis is completed using structural analysis software to avoid time-consuming hand calculations using the method of joints or the method of sections, as described in *Statics and Mechanics of Structures*, (Krenk & Høgsberg 2013). In using structural analysis software, the truss geometry is defined, the structure is analysed and the truss geometry is then modified based on the analysis results as necessary. This process is repeated until the desirable solution is achieved. Using the developed script, the process of defining geometry and analysing the structure occur concurrently offering a more streamlined process (Figure 13).

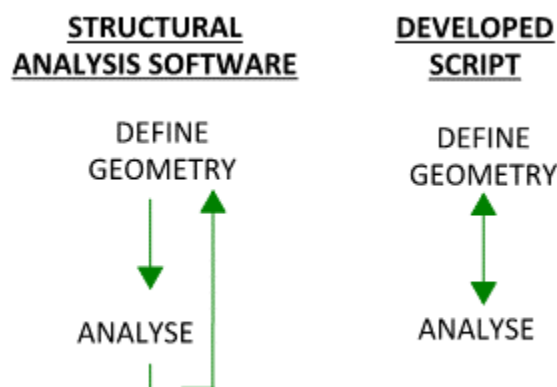


Figure 13 Workflow comparison; structural analysis software vs developed script implementing graphic static techniques

However, with the developed script a colour-coded system relating the truss member to their corresponding force vector, or a similar reference mechanism, is required. While graphical techniques typically employ Bow's Notation in their construction as referencing system (Shearer 2010; Allen & Zalewski 2009), this research aimed to eliminate its use in place of graphical visualisation, as the process of assigning the notation is a manual one. It would also be helpful to have the option to display the force magnitude value on the force polygon if desired. Despite the current limitations associated with reading the result, this method presents the potential to offer a quicker, adaptable, workflow compared to the use of structural analysis software for concept truss design.

A number of features that would enhance the tool include the ability to modify the force diagram with the corresponding form diagram updated in real time, and the inclusion of different truss types. The truss type definition influences the construction sequencing of the joints considered, as shown for the six-bay pratt truss in Figure 5. While adding additional truss types would be relatively simple and provide a wider application of the script, they was not considered necessary in demonstrating the application of the graphical statics method in a parametric tool. Further development of the tool could also include varying loads, curved and tapering truss profiles.

While the script successfully eliminates the manual construction process of the force polygon, the user will not gain the knowledge of its construction in using this tool. However, teaching the method was not the goal of this research with a number of educational resources available as previously discussed. The goal was to provide a platform through which the benefits of graphical construction could be capitalised upon whilst eliminating the time-consuming, manual process, in order to increase the method's appeal in practice.

As the force diagram is updated in real-time the tool could provide a powerful communication platform for discussions between engineers and the design team. Visualising the impact the truss depth, for example, can have on the member forces, provides instant and understandable structural feedback on design decisions.

Conclusions

The design industry has reverted to a very graphical way of designing structures which has regenerated interest in graphical solutions to provide a greater understanding of the forces within a structure (Van Mele et al. 2012; Beghini et al. 2014; Fivet & Zastavni 2013).

This paper presented a computational tool which has the aim of increasing graphic statics applicability and appeal in today's world of 3D graphical designing. Visual Basic Scripting is used within Grasshopper, to script the basic steps of graphic statics and assess its applicability with the latest 3D modelling techniques to aid the understanding of structures. Despite some limitations associated with reading the results, this method presents the potential to offer a quicker, adaptable, workflow compared to the use of structural analysis software for concept truss design. Some possible paths for further development in investigating the use of graphic statics methods in today's world of computing have been raised, with further development potential to enhance the tool proposed including the potential to include varying loads and curved forms.

The concept stage of a project has the largest amount of design freedom compared to other detailed design stages, and requires the ability to rapidly generate and compare design alternatives. The developed tool meets the requirement of providing a platform for rapid analysis while also providing a visualisation of the results for communication with the design team to aid their understanding of the structural dependencies. This adds to the growing body of research which aims to assess 'how computers and computation can offer new possibilities to an ancient field of research' (Block 2005).

References

- Allen, E. & Zalewski, W., 2009. *Form and Forces: Designing Efficient, Expressive Structures*, Hoboken, New Jersey: John Wiley & Sons.
- Beghini, L.L. et al., 2014. Structural optimization using graphic statics. *Structural and Multidisciplinary Optimization*, 49(3), pp.351–366.
- Block, P., 2005. *Equilibrium systems: Studies in Masonry Structure*. Massachusetts Institute of Technology.
- BLOCK Research Group, 2012. eEQUILIBRIUM. Available at: <http://block.arch.ethz.ch/equilibrium/> [Accessed May 10, 2015].
- Culmann, K., 1866. *Die Graphische Statik*, Zurich: Verlag von Meyer & Zeller.
- Fivet, C. & Zastavni, D., 2013. Constraint-based graphic statics: New paradigms of computer-aided structural equilibrium design. *Journal of the International Association for Shell and Spatial Structures*, 54(178), pp.271–280.
- Greenwold, S., 2003. Active Statics. Available at: <http://acg.media.mit.edu/people/simong/statics/Start.html> [Accessed May 10, 2015].
- Grüning, M., 1925. *Die Statik des ebenen Tragwerkes*, Berlin: Julius Springer.
- Krenk, S. & Høgsberg, J., 2013. Statics and Mechanics of Structures. In *Statics and Mechanics of Structures*. pp. 39–89. Available at: <http://link.springer.com/10.1007/978-94-007-6113-1>.
- Kurrer, K.-E., 2008. *The History of the Theory of Structures: From Arch Analysis to Computational Mechanics*, Berlin: Ernst & Sohn Verlag für Architektur und technische Wissenschaften GmbH & Co. KG.
- Van Mele, T. et al., 2012. Geometry-based understanding of structures. *Journal of the International Association for Shell and Spatial Structures*, 53(174), pp.285–295.
- Müller-Breslau, H., 1903. *Die Graphische Statik der Baukonstruktionen, Band II. Erste Abtheilung* 3rd, rev., Leipzig: Baumgärtner's Buchhandlung.
- Philippe, B., 2005. InteractiveThrust. Available at: <http://web.mit.edu/masonry/interactiveThrust/index.html> [Accessed May 10, 2015].
- Rolvink, A., Mueller, C. & Coenders, J., 2014. State on the Art of Computational Tools for Conceptual Structural Design. In *Proceedings of the IASS-SLTE 2014 Symposium "Shells, Membranes and Spatial Structures: Footprints."*
- Shearer, M.S., 2010. *Analyzing and Creating Forms : Rapid Generation of Graphic Statics Solutions through RhinoScript* by.
- Varignon, P., 1725. *Nouvelle mécanique ou statique*. 2 Vols., Paris: C. Jombert.

Wolfe, W.S., 1921. *Graphical Analysis: A textbook on Graphic Statics*, New York: McGraw & Hill Book Company.

Appendix

Developed script using Visual Basic in Grasshopper

```

'
'-----
'TRUSS DEFINITION

'Get node x coordinates
Dim totchordlength As Double
Dim x_coordlist As New list(Of Double)
Dim chordlength As Double
Dim x_coord As Double

totchordlength = u * 1000
chordlength = totchordlength / x

Dim i As Double
For i = 0 To x
    x_coord = chordlength * i
    x_coordlist.add(x_coord)
Next

'Truss depth
Dim depth1 As Double
Dim depth2 As Double
depth1 = (w * 1000) / 2
depth2 = (w * -1000) / 2

'Bottom nodes
Dim listbotnodes As New List(Of point3d)
Dim botnode As point3d

Dim u9 As Integer
For u9 = 0 To x_coordlist.count() - 1
    botnode.X = x_coordlist.item(u9)
    botnode.Y = depth2
    botnode.Z = 0
    listbotnodes.add(botnode)
Next

'Botchord
Dim listbotchord As New list(Of line)
Dim botpt1, botpt2 As point3d

Dim v3 As Integer
For v3 = 0 To listbotnodes.count() - 2

    botpt1 = listbotnodes.item(v3)
    botpt2 = listbotnodes.item(v3 + 1)

    Dim botchord As New Line(botpt1, botpt2)

    listbotchord.add(botchord)

Next

'Top nodes
Dim listtopnodes As New List(Of point3d)
Dim topnode As point3d

Dim v8 As Integer
For v8 = 0 To x_coordlist.count() - 1
    topnode.X = x_coordlist.item(v8)
    topnode.Y = depth1
    topnode.Z = 0

    listtopnodes.add(topnode)
Next

'Topchord
Dim listtopchord As New list(Of line)
Dim toppt1, toppt2 As point3d

Dim v2 As Integer
For v2 = 0 To listtopnodes.count() - 2

    toppt1 = listtopnodes.item(v2)
    toppt2 = listtopnodes.item(v2 + 1)

    Dim topchord As New Line(toppt1, toppt2)

    listtopchord.add(topchord)

Next

'weave top and bototm chord
Dim chordweave As New List(Of line)

Dim w2 As Integer
For w2 = 0 To listbotchord.count() - 1

    chordweave.add(listbotchord.item(w2))
    chordweave.add(listtopchord.item(w2))

Next

A = chordweave

```

```

'First half truss diagonals
Dim topnodes_half1 As New List(Of point3d)
Dim top As New point3d
Dim botnodes_half1 As New List(Of point3d)
Dim bot As New point3d
Dim weave1 As New list (Of point3d)
Dim diapt1 As Point3d
Dim diapt2 As point3d
Dim listdia1 As New List(Of line)

Dim w6 As Integer
For w6 = 0 To (listtopnodes.count() - 1) / 2
    top = listtopnodes.item(w6)
    topnodes_half1.add(top)
Next

Dim vv As Integer
For vv = 0 To (listbotnodes.count() - 1) / 2
    bot = listbotnodes.item(vv)
    botnodes_half1.add(bot)
Next

Dim kk As Integer
For kk = 0 To (topnodes_half1.count() - 1)
    weave1.add(botnodes_half1.item(kk))
    weave1.add(topnodes_half1.item(kk))
Next

Dim w9 As Integer
For w9 = 0 To weave1.count() - 2
    diapt1 = weave1.item(w9)
    diapt2 = weave1.item(w9 + 1)

    Dim dial As New Line(diapt1, diapt2)
    listdia1.add(dial)
Next

'Second half truss diagonals
Dim topnodes_half2 As New List(Of point3d)
Dim top2 As New point3d
Dim botnodes_half2 As New List(Of point3d)
Dim bot2 As New point3d
Dim weave2 As New list (Of point3d)
Dim diapt3 As Point3d
Dim diapt4 As point3d
Dim listdia2 As New List(Of line)

Dim uu As Integer
For uu = ((listtopnodes.count() - 1) / 2) + 1 To listtopnodes.count() - 1
    top2 = listtopnodes.item(uu)
    topnodes_half2.add(top2)
Next

Dim ww As Integer
For ww = ((listbotnodes.count() - 1) / 2) + 1 To listbotnodes.count() - 1
    bot2 = listbotnodes.item(ww)
    botnodes_half2.add(bot2)
Next

weave2.add(weave1.item(weave1.count() - 2))

Dim ss As Integer
For ss = 0 To (topnodes_half2.count() - 1)
    weave2.add(topnodes_half2.item(ss))
    weave2.add(botnodes_half2.item(ss))
Next
Dim t2 As Integer
For t2 = 0 To weave2.count() - 2

    diapt3 = weave2.item(t2)
    diapt4 = weave2.item(t2 + 1)

    Dim dia2 As New Line(diapt3, diapt4)

    listdia2.add(dia2)

Next

```

```

Dim t4 As Integer
For t4 = 0 To listdia2.count() - 1
    listdiagonals.add(listdia2.item(t4))
Next

B = listdiagonals

'-----
'GENERATE LOAD LINE POINTS

Dim force_mag As Double
Dim forces_no As Double
force_mag = x * z
forces_no = x + 2

Dim force_pointslist As New List(Of point3d)
Dim force_point As point3d

Dim u5 As Integer
For u5 = 0 To forces_no - 1
    force_point.X = u * 1000 + 1000
    force_point.Y = (u5 * -z) - 1000
    force_point.Z = 0

    force_pointslist.add(force_point)
Next

'-----
'DRAW LOAD LINES AND WEAVE POINTS

'draw load lines using input load line points
Dim listloadlines As New List(Of Line)
Dim point1, point2, point3, point4, midpoint As point3d

Dim r As Integer
For r = 0 To force_pointslist.count() - 2

    point1 = force_pointslist.item(r)
    point2 = force_pointslist.item(r + 1)
    Dim ln As New Line(point1, point2)

    listloadlines.add(ln)

Next

point3 = force_pointslist.item(0)
point4 = force_pointslist.item(force_pointslist.count() - 1)
Dim ln2 As New Line(point3, point4)

midpoint = ln2.PointAt(0.5)

Dim ln3 As New line(point4, midpoint)
Dim ln4 As New line(midpoint, point3)

listloadlines.add(ln3)
listloadlines.add(ln4)

C = listloadlines

'-----
'weave points
Dim listloadlinepoints As New List(Of Point3d)
Dim startpoints As Point3d
Dim midpoint2 As point3d
Dim listloadlinepoints2 As New List (Of Point3d)

Dim j As Integer
For j = 0 To (listloadlines.count() - 1)
    startpoints = listloadlines.item(j).PointAt(0.0)
    listloadlinepoints.add(startpoints)
Next

midpoint2 = listloadlinepoints.item(listloadlines.count() - 1)

listloadlinepoints.removeat(listloadlines.count() - 1)

```

```
midpoint2 = listloadlinepoints.item(listloadlines.count() - 1)

listloadlinepoints.removeat(listloadlines.count() - 1)

Dim k As Integer
For k = 0 To (listloadlinepoints.count() - 1)
    listloadlinepoints2.add(listloadlinepoints.item(k))
    listloadlinepoints2.add(midpoint2)
Next

If v = "0" Then
    listloadlinepoints2.removeat(listloadlinepoints.count() - 1)
    D = listloadlinepoints2
Else
    D = listloadlinepoints2
End If
```

```

'DRAW FORCE POLYGON

Dim listpoints As New List(Of Point3d)
Dim listlines As New List(Of Line)

Dim i As Integer
For i = 0 To (u.count - 2)

    Dim l_diagonal As Line
    Dim v_diagonal As vector3d
    Dim v_diagonalx, v_diagonaly, slope_diagonal

    Dim l_chord As Line
    Dim v_chord As vector3d
    Dim v_chordx, v_chordy, slope_chord

    Dim point1, point2 As point3D
    Dim line1
    Dim v1 As vector3d
    Dim c2
    Dim pointix, pointiy
    Dim xdelta, y, ydelta, y As Double
    Dim intpt As Point3d
    Dim forceline1, forceline2 As Line

    'Get unit vector of truss diagonal and x, y values
    l_diagonal = u.item(i)
    v_diagonal = l_diagonal.Direction
    v_diagonal.Unitize()
    v_diagonalx = v_diagonal.x
    v_diagonaly = v_diagonal.Y

    'Slope of the diagonal
    If v_diagonalx = 0 Then
        slope_diagonal = 0
    Else
        slope_diagonal = v_diagonaly / v_diagonalx
    End If

    'Get unit vector of truss chord and x, y values
    l_chord = v.item(i)
    v_chord = l_chord.Direction
    v_chord.Unitize()
    v_chordx = v_chord.X
    v_chordy = v_chord.Y

    'Slope of the chord
    If v_chordx = 0 Then
        slope_chord = 0
    Else
        slope_chord = v_chordy / v_chordx
    End If

    'Obtain point1
    If i = 0 Then
        point1 = w.item(i)
    Else
        point1 = A
    End If

    'Obtain c2
    point2 = w.item(i + 1)
    line1 = New Line(point1, point2)
    v1 = line1.Direction
    c2 = v1.Y

    'Get x and y coordinates of first point
    pointix = point1.X
    pointiy = point1.Y

    'intersection point
    If slope_diagonal = 0 Then
        xdelta = 0
    Else
        xdelta = (c2 / (slope_diagonal - slope_chord))
    End If
    x = pointix + xdelta
    ydelta = (x * slope_chord) + c2
    y = pointiy + ydelta

    intpt = New Point3d(x, y, 0)
    A = intpt

    'associated force lines
    forceline1 = New Line(point1, intpt)
    forceline2 = New Line(point2, intpt)

    'add all interations to lists
    listpoints.add(A)
    listlines.add(forceline1)
    listlines.add(forceline2)

    Next i

E = listpoints
F = listlines

```