TUDOMÁNYOS DIÁKKÖRI KONFERENCIA

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM

ÉPÍTÉSZMÉRNÖKI KAR

TARTÓSZERKEZETEK SZEKCIÓ

KÉSZÜLT: 2015 / 2016 - I. SZEMESZTER

KÉSZÍTETTE: FÜZES BÁLINT PÉTER

KONZULENS: DR. HEGYI DEZSŐ

TANSZÉK: SZILÁRDSÁGTANI ÉS TARTÓSZERKEZETI TANSZÉK

NEPTUN KÓD: AK3RKI

MEMBRÁNSZERKEZETEK PARAMETRIKUS MODELLGYŰJTEMÉNYE

PARAMETRIC DETAILS OF MEMBRANE CONSTRUCTIONS

WWW.MEMBRANEDETAIL.COM

MEMBRANE DETAIL
PARAMETRICS

TARTALOMJEGYZÉK

01 BEVEZETÉS

Ez a dolgozat folytatása az ugyanitt íródott, *Feszített membránszerkezetek részletmegoldásai - virtuális csomópontgyűjtemény (2012)* és *Membránszerkezetek megtámasztási megoldásai - sarokcsomópontok (2011)* című TDK munkáimnak. Mindez elérhető a *www.membranedetail.com* honlapon. Jelen kutatásom tárgya a feszített membránszerkezetek csomópontképzésének parametrikus modellezhetősége.

A kutatás célja elsőként egy nyílt, internetes tudástár létrehozása volt, ahol membráncsomópontok tekinthetőek meg kiviteli és erőtani szempontok szerint kategorizálva. Ez a honlap formájában megvalósult, a mostani fázisban gyakorlati hasznosságot kölcsönzök a rendszernek.

A parametrikus modellezés lényege, hogy adott geometria belső összefüggésrendszere kerül megalkotásra algoritmikus módon. Nem egyszeri forma, hanem egy lépéssor (kód, algoritmus) készül, mely a formát leírja. A paraméterek utólagos változtatásával a forma újragenerálódik - tetszés és igény szerint módosítható. Ilyen kódok használata többek közt időigényesen szerkeszthető, bonyolult geometriák esetén indokolt, ide sorolhatóak a feszített membránszerkezetek (sátrak, ponyvák) szerkezeti csomópontjai. Membránoknál rengeteg felületet, görbét, kapcsolóelemet, egyebet kell megmodellezni. Az elemek sokasága és erős térbelisége folytán a kiviteli tervek kidolgozásnál szinte fokozhatatlanul megnő a térbeli szerkesztési munka mennyisége.

Egy olyan parametrikus algoritmuscsokrot alkottam meg, mely rendkívül felhasználóbarát működésű. A kódoláshoz nem értő felhasználó példaképp megrajzol három, egymást metsző görbét, és a kód kigenerál egy komplex sarokcsomópontot. Átpozicionálja a metszéspontot, és a sarok újragenerálódik. A valós gyártmányokat reprezentáló elemek az új szitációhoz igazodnak, emellett egyetlen kattintással cserélhetőek, léptékezhetőek (pl. a kötélfej típusa, csavarszám, stb). A kódok a Rhinoceros® (McNeel - Rhino 5) Grasshopper® plug-injével készültek. A felhasználónak nincs teendője kódolás terén.

Célom jelentős mértékű manuális szerkesztési munkát és időt spórolni vállalati szinten. A projekt jelenlegi célkitűzése a kódok használhatóságának prezentálása és levalidáltatása valós, ipari felhasználókkal. Amennyiben ez a néhány kód pozitív visszajelzést kap, érdemes lehet a továbbfejlesztésén dolgozni.

A dolgozat a membránszerkezetek rövid bemutatásával, valamint a projekt előzményeinek és régi-új célkitűzésének ismertetésével kezdődik. Ezután a parametrikus algoritmusok alapvető működését és a felvetődött problémaköröket taglalom, majd az azt követő fejezetekben és a Mellékletben az általam írt algoritmusokat mutatom be írásos és vizuális formában.

02 MEMBRÁNSZERKEZETEK RÖVID ISMERTETÉSE

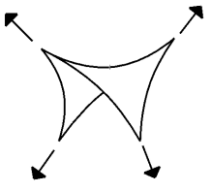
A membránok (közismertebb nevükön sátor- vagy ponyvaszerkezetek) a tartószerkezetek egy aránylag ritkán előforduló csoportját alkotják. Elsősorban árnyékolóként és előtetőként, köztéri lefedésként, stadion- és átriumtetőként, illetve újabban homlokzatszerkezetekként fordulnak elő. A szerkezet a kifeszített ponyvaanyagból, illetve a kapcsolt feszítőelemekből áll.

A ponyvaszerkezetek két legfőbb tartószerkezeti sajátossága a következő:

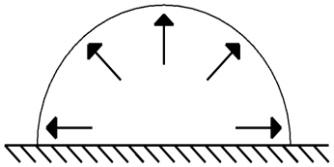
- a membránfelületekben kizárólag húzófeszültségek működnek. Ezen erők által alakul ki a felület formája, geometriája, mely a szerkezet stabilitását biztosítja a külső erőkkel szemben.

- a felület önsúlya elhanyagolható, átlagosan 1 kg/m^2 . A tartószerkezetek között jelenleg a ponyváknál a legkisebb az önsúly az áthidalt fesztávolsághoz képest.

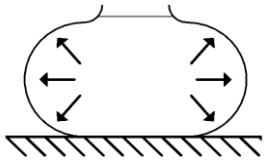
Három fő működési kategóriát különböztethető meg aszerint, hogy a felületben terjedő húzófeszültségi erőrendszer hogyan hozható létre.



1. ábra



2. ábra



3. ábra

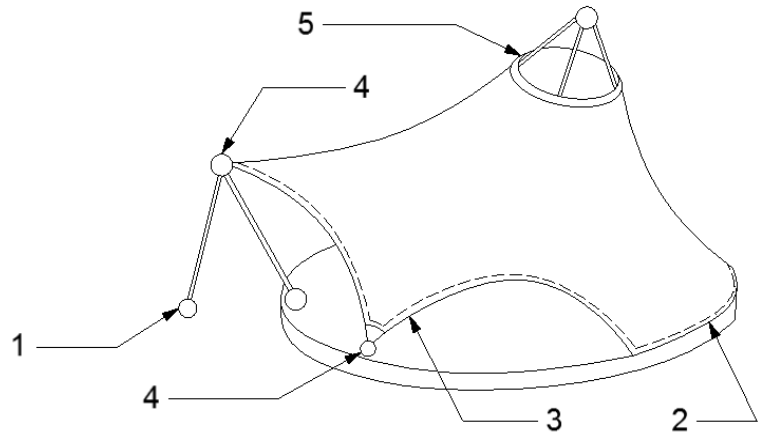
Feszített membrán
Architectural Association, London, UK

Légtartós membrán
Mecenatpolis Shopping Mall, US

Biogáz tároló dupla membrán
DL

Az *1. sémaábra* a feszített membránszerkezetek kategóriájára utal. A teljes projekt folyamán kizárólag ennek a szerkezettípusnak a csomópontjaival foglalkozom. A felületben terjedő húzás a peremszerkezetek megfeszítése által jön létre, így állítható be a lefedés kívánt geometriája.

A *2. ábrán* a légnyomásos szerkezetek működésének sémája látható. A felületet a membránnal lehatárolt térben létrehozott mesterséges légnyomás feszíti meg. Sok működési változatuk ismert (egyrétegű sátrak, légpárnák, légtömlők, tensairity szerkezetek). A *3. sémaábrán* a tartályok kategóriája látható. Itt a felületet megfeszítő belső nyomást tárolt anyagok, folyadékok, gázok hozzák létre.



4. ábra

A *4. ábrán* egy általános feszített membránszerkezet rajza látható. Jellegzetes szerkezeti helyzetei a következők:

1 - Alapozási kapcsolatok, lehorgonyzások

2 - Külső, vonal menti megtámasztások – merev peremek

3 - Külső, vonal menti megtámasztások – lágy (kötél)peremek

4 - Külső, pontszerű megtámasztások – sarokcsomópontok

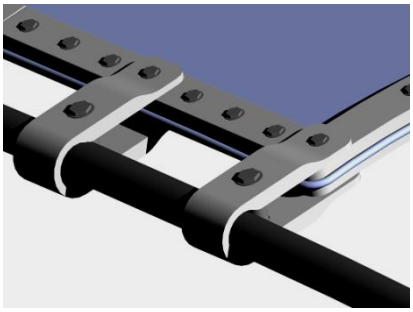
5 - Pontszerű megtámasztások – árboccsomópontok

A membránok geometriai sokfélesége miatt a csomópontok ennél a tiszta kategorizálásnál jobban összemosódnak: előfordulnak belső peremek (membrántoldások kiegészítő kötélszerkezet mentén), szélső árboccsomópontok, belső sarkok, stb.
A következő fejezetek során ezekhez az alapvető kategóriákhoz nyúlok vissza, ezért egy-egy példával illusztrálom ezeket a megoldásokat (a 2. példa kivételével a képek a honlapomról származnak).

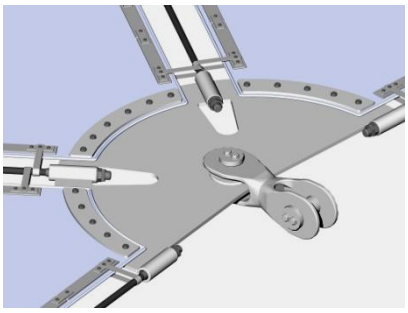
(Megjegyzés: a dolgozat folyamán a net szó a képaláírások alatt akkor fordul elő, ha az adott kép nem az általam készített ábraanyag része, hanem az Internet egyéb szakmai oldalairól származik.)

1 - Csuklós tömbalapkapcsolat - kötéllehorgonyzás

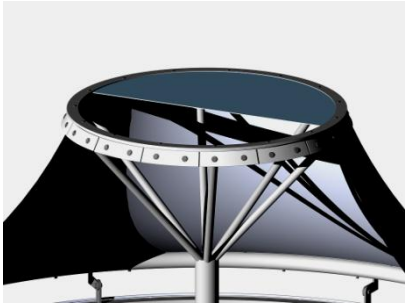
2 - Merev peremkapcsolat szorítólapokkal (net)



3 - Lágy kötélperem - szorítólapos-füles kapcsolat



4 - Sarokkapcsolat rugalmas kötélperemekkel



5 - Merev árbockapcsolat bevilágítóval

03 ELŐZMÉNYEK ÉS CÉLKITŰZÉSEK

A *Membránszerkezetek megtámasztási megoldásai - sarokcsomópontok (2011)* és *Feszített membránszerkezetek részletmegoldásai - virtuális csomópontgyűjtemény (2012)* című TDK munkáim, illetve jelen dolgozatom egy kutatás egymást követő fázisai.

A kollekció elsődleges, kezdeti célja egy ismertető jellegű mérnöki csomópontgyűjtemény létrehozása volt. A kiindulópont a feszített membránszerkezetek szakirodalmának azon hiányossága volt, hogy a fellelhető tudásanyag leginkább a membránfelületekkel foglalkozik, kevés célzott információ áll rendelkezésre a csomópontképzéssel kapcsolatban. A projekt ennek kiegészítését célozta meg. Minél több csomópont rajza és modellje készült el, annál hasznosabb lett a gyűjtemény.

Az első dolgozat során 2D-s műszakirajz-gyűjtemény formájában kb. 40 db sarokcsomópontot dolgoztam fel erőtani működés szerinti bontásban. A második dolgozatban (a merev peremektől eltekintve) az összes szituációhoz számos példa (kb. 90 db) CAD-modelljét készítettem el. Az ezekből a modellekből kinyert vizuális anyag (több száz kép) kompletten elérhető a *www.membranedetail.com* honlapon, amely ezáltal az Interneten egyedülállóan az első ilyen témájú nyílt tudástár lett. Használható oktatási célokra, illetve valós tervezés folyamán a különböző helyzeteknél fellelhető konstrukciókkal kapcsolatos adat- és ötletgyűjtésre.

A projekt jelen fázisában algoritmusokkal dolgozom, és gyakorlati használhatóságot kölcsönzök a rendszernek. Jelen célkitűzésem az, hogy bonyolult térbeli szerkezetek 3D-s modellezését gyorsítsam fel és egyszerűsítsem le. Ezzel vállalati szinten takarítható meg jelentős mennyiségű manuális tervezési és modellezési munka. Különösen a membránszerkezetek kiviteli terveinél fordul elő rengeteg, erősen térbeli (és nem síkbeli) forma, tehát "nem-ortogonális" tervezési helyzet (pl.: Sanghai Expo Axis Membrane, *5. ábra*)

5. ábra (net)

Az algoritmusokat magam írtam a Rhino McNeel® 3D-s tervezőprogram Grasshopper® nevű parametrikus plug-injében. Az algoritmusokba szervesen épülnek be azok a CAD-modellek, melyek az előző dolgozatomhoz készültek.

Célom prezentálni ezeket a példaképpen megírt kódokat, illetve ipari felhasználókkal levalidáltatni a működési elvüket. Akkor érdemes ezeket az algoritmusokat továbbfejleszteni, ha valós tervezői (vagy oktatási) igény merül fel.

A továbbfejlesztés lényege jelen elképzeléseim szerint az lehet, hogy egy kiterjedt algoritmusgyűjteményt hozok létre. Ez lefedné az összes jelentős szerkezeti helyzetet, és tökéletesen alkalmas lenne a leggyakoribb membránformák gyors felfejlesztéséhez kiviteli tervek részletezettségi fokára. Használható lenne ezáltal látvány- és koncepcionális tervezéshez is: néhány görbe megrajzolásával komplett szerkezetek jönnének létre. A leggyakoribb árboc-, kötél-, horgony- és peremkapcsolatok kódjai foglaltanának benne újrafelhasználható módon.

Ez kiegészülne egyéni, sajátos csomópontok hozzárendelhetőségével, és a jelenlegivel azonos módon fogadna valós gyártmánymodelleket. Kiegészülne egy automatikus Excel-táblával is, amely kimutatást végez az összes lényeges hosszról, elemszámról, stb. Egy ilyen fejlesztés kizárólag ipari igény megléte esetén tud értelmet nyerni.

A széleskörű prezentációhoz összeállítok egy, az ezeket az algoritmusokat bemutató videót, melyet szintén a közeljövőben megújuló honlapon fogok megosztani. A honlapra emellett egy kód is fel fog kerülni, mely szabadon letölthető és kipróbálható lesz.

04 A PARAMETRIKUS ALGORITMUSOK ALAPVETŐ MŰKÖDÉSE

A parametrikus tervezés és modellezés lényege, hogy egy adott geometria belső összefüggésrendszere *algoritmus (kód)* formájában fejeződik ki. (A parametrikus modellezés nem csak geometriai-matematikai problémamegoldásra alkalmas: használható koncepcionális és művészeti tervezésre is, de ez jelen dolgozatnak nem tárgya.)

A parametrikus algoritmus (összefüggésrendszer) tehát egy olyan lépéssor, amely *logikai-matematikai-geometriai* utasításokon keresztül fejezi ki egy forma tetszőleges számú variációját. Ugyanis egy adott *paraméter* (a kód egy számszerű eleme) a forma egyik változtatható geometriai tényezője: egy ismeretlen egy "sokismeretlenes egyenletben".

Az algoritmusok hatékonysága összhasonlíthatatlanul jobb a manuális formaleírás hatékonyságával szemben. Ez a hasznosság és ésszerűsíthetőség természetesen csak olyan esetekben tud érvényesülni, amikor a modellezési szituáció pl. valamelyik eleme erősen repetitív, vagy erős valamely függőségi viszony, vagy nagy pontosságot, esetleg rengeteg időt igényel, azaz *ésszerűen automatizálható*.

Egy ilyen ésszerűsíthető eset a membránszerkezeti csomópontmodellezés. Ilyenkor rengeteg (főként acél) elem kerül megmodellezésre (rengeteg kötélfej, csavar, lemez, horgony, szorítólap, saru, stb *(*La Defense Membrane - Párizs, *6. ábra)*. Valós tervezéskor az idő és a manuális munka erőforrási kérdés. Kiviteli terveknél majdhogynem fokozhatatlanná válik ez a szerkezetmodellezési feladat. Ezért ha ilyenkor változik egy távolság (pl. odébb kerül egy oszlop akár 10 cm-t), időveszteséget jelent az újramodellezése. Parametrikus esetben a forma egyszerűen újragenerálódik és az új helyzethez igazodik.

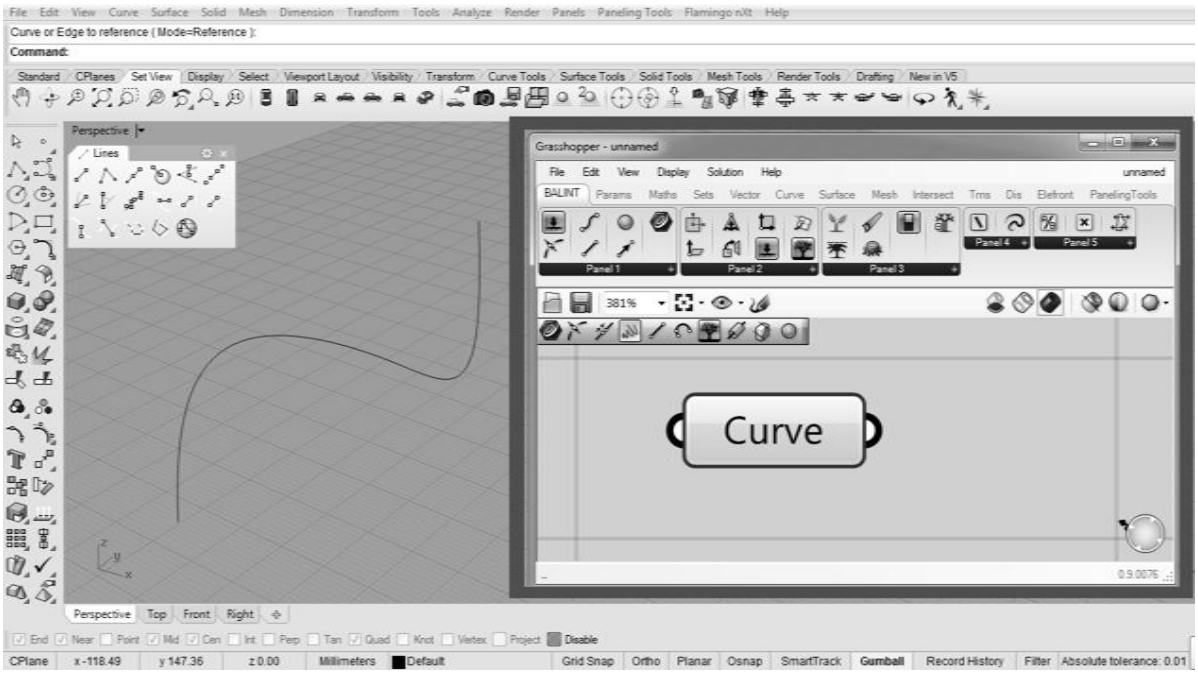
6. ábra (net)

Leegyszerűsítve: ha egyszer megírásra kerül egy kötél modelljének algoritmusa, sosem kell többé kötelet modellezni, pusztán aktiválni és alkalmazni kell az algoritmust.

A fentiekre a megoldás olyan szoftverek használata, amelyek beépítetten kínálják a standardeknek megfelelő acélszerkezetek csomópontjait. Ilyen szoftverek pl. az *Easy (http://technet-gmbh.de/)*, a *formfinder (http://www.formfinder.at/),* és az *IX-Cube (http://www.ixray-ltd.com/).* Ezeknek előnye, hogy tőkeerős vállalatok állnak a hátterükben. Hátrányuk a magas ár, illetve az, hogy ezeknek a szoftvereknek a lényege a membránstatikai modellezés, tehát az erőtani viselkedés és az alakkeresés tervezése ("formfinding") - a csomópontok csak újonnan kezdenek belekerülni ezekbe. Pontosan emiatt nem helyettesítik a hagyományos CAD-munkát, ugyanis egy tervezési-modellezési feladatnak rengeteg egyéb eleme is van a membrán alakképzése mellett: akár a membránnal kapcsolatban, akár a környező épület(szerkezetek)et illetően. Végeredményben tehát leginkább Autodesk AutoCAD® és Rhino® programokban folyik a modellezés.

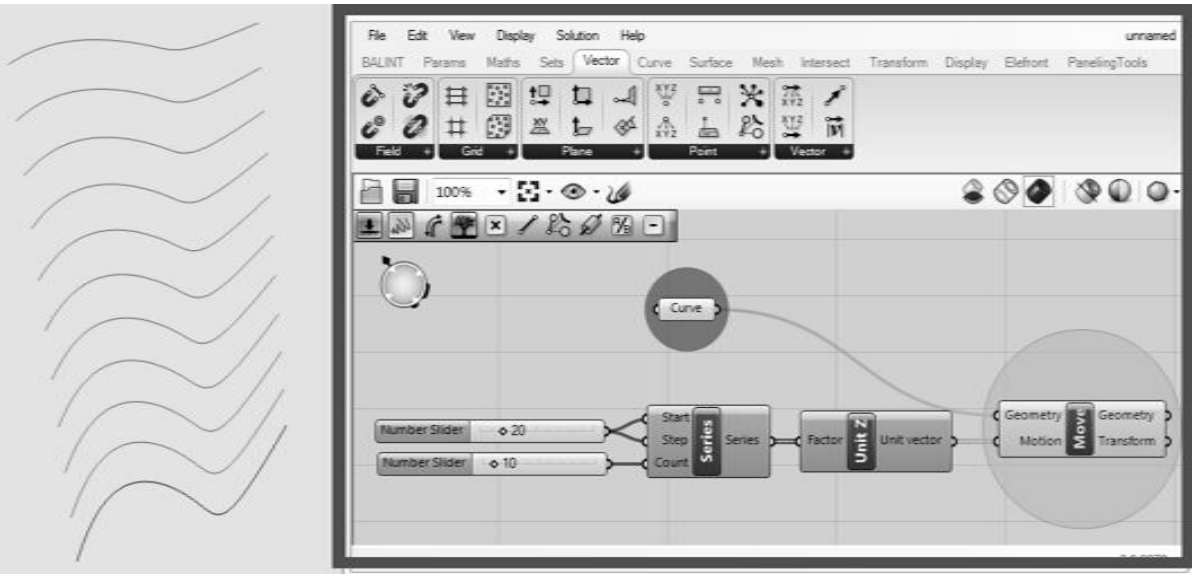
A legszéleskörűbben használt fájlkiterjesztés ezáltal a *.dwg*, illetve egyre inkább (pont a parametrikus lehetőségek miatt) a *.3dm* a Rhino McNeel® -től. Ezek között egyenes az átjárhatóság, tehát a konvertálhatóság export-import módon. Az általam írt algoritmusok a Rhino McNeel® Grasshopper® (innentől GHP) nevű parametrikus rendszerében készültek. Az alábbiakban röviden összefoglalom a beépített parametrikus GHP plug-in működését.

A Rhino alapvető tervezőfelületével (*modelltér*) párhuzamosan lehet működtetni a GHP plugint (*parametrikus tábla*), melyet a 7. ábrán kereteztem. A baloldalt látható modelltéri görbe megfelel a táblán lévő Curve (görbe) *utasításnak.* Ha pl. a modelltéri görbét odébbtolom vagy megnövelem a hagyományos CAD modelltéri parancsokkal, a táblán lévő görbe *utasítás* továbbra is érti, hogy erről a görbéről van szó.



7. ábra

Ha viszont táblatéri parametrikus utasításokat rendelek a Curve utasításhoz, akkor a modelltérben végbemennek a változások (*8.ábra*). Itt több dolog is látható.



8. ábra

Látható egy parametrikus GHP algoritmus kinézete és működési elve. Utasítások vannak összekötve egymással: egy adott utasítás vagy számszerű *paraméter* egy következő utasítás *inputjává* válik. Ilyen módon haladnak előre az összekötő görbék mentén az adatok, és sosem köthető önmagába vissza egy utasítássor. A legfontosabb, hogy a paraméterek folyamatosan változtathatóak, és a forma (itt a görbesor) a számoknak megfelelően újragenerálódik.

Látható, ahogy a görbe sokszorosítva eltolódott a koordinátarendszer z-tengelye mentén. Az eltolás a *Move utasítás* volt, annak iránya és darabszáma a *Motion inputfülön* beérkező *adathalmaz*. Ez az adathalmaz a *Z-egységvektor* megsokszorozása a *Series utasítással*, illetve annak bemenő *paramétereivel*. Ez utóbbi kifejezi, hogy az egységvektort 20 mm-enként 20 mm eltolásról indulva 10-szer kell megismételni. Ezek a számok tetszés szerint átírhatóak, a görbe újrgenerálódik.

A görbékkel összekötött utasítások hátterében mindig egy-egy *adathalmaz, ún. lista* van. Ez többnyire szám, koordináta, szöveg, igaz-hamis tulajdonság vagy közvetlen utalás egy modelltéri geometriai elemre (pl. egyenesek halmaza, vágatlan felületdarab, stb.)

Utolsóként fontos: ahhoz, hogy ez az új görbesor *rögzüljön* a modelltérben, meg kell *sütni* - ez az elnevezés a parancs nevéből fakad (Bake). Amíg nem sül meg a forma, addig változtatható a kigondolt szempontrendszer szerint.

Ez volt az alapelve a GHP algoritmusoknak egy egyszerű példán megmutatva: egy vizuális utasításrendszer felhasználóbarát módon megalkotva. Innentől a saját algoritmusaim (és az azok elkészítése során felvetődött problémák) bemutatása következik.

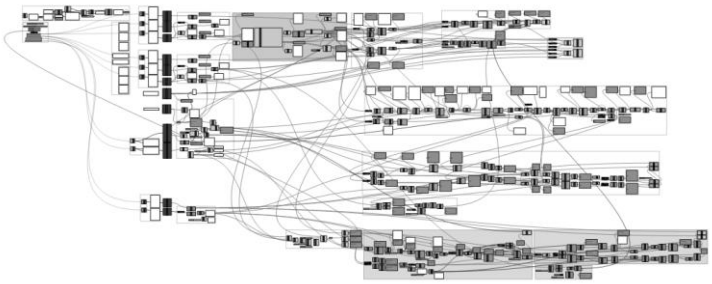
05 AZ ALGORITMUSGYŰJTEMÉNY RENDSZERE

Az általam létrehozott algoritmusok elsődleges célja az, hogy a kódoláshoz egyáltalán nem értő felhasználók is alkalmazni tudják. Ez a gyakorlatban a következőt jelenti:

A *felhasználó* ***kijelöl*** *egy alapelemet (pl. egy egyszerű kontúrt), az algoritmus pedig elkészíti a membráncsomópontot*. ***Változtat*** *a kontúrelemen, és az algoritmus újragenerálja a membráncsomópontot*. *Ezután a felhasználó a neki felajánlott számszerű paraméterek közül* ***válogat*** *ízlés szerint, mellyel a membráncsomópontot felruházza a neki tetsző tulajdonságokkal. Amikor megfelelő tulajdonságúvá változott a csomópont, akkor egy kattintással* ***rögzíti*** *a formát.*

Pontosan ehhez a tovább nem egyszerűsíthető működéshez kell igen bonyolult algoritmusokat írni, hogy eszerint az elv szerint használható legyen. A kutatás és a programnyelv elsajátítása során arra jutottam, hogy két fájl párhuzamos működtetése a *legstabilabb* és *legkönnyebben terjeszthető* megoldása a célkitűzésnek.

Az egyik a *háttérfájl,* egy egyszerű CAD-fájl, amely a feszítőelemek különböző (akár gyártmánytervi szintű) modelljeit tartalmazza. A másik maga a *GHP algoritmus fájlja*.

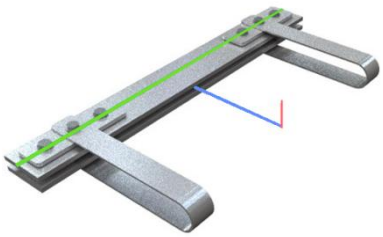


HÁTTÉRFÁJL (.3dm) ALGORITMUS (.ghp)

MD_MODELS

MD_CORNER

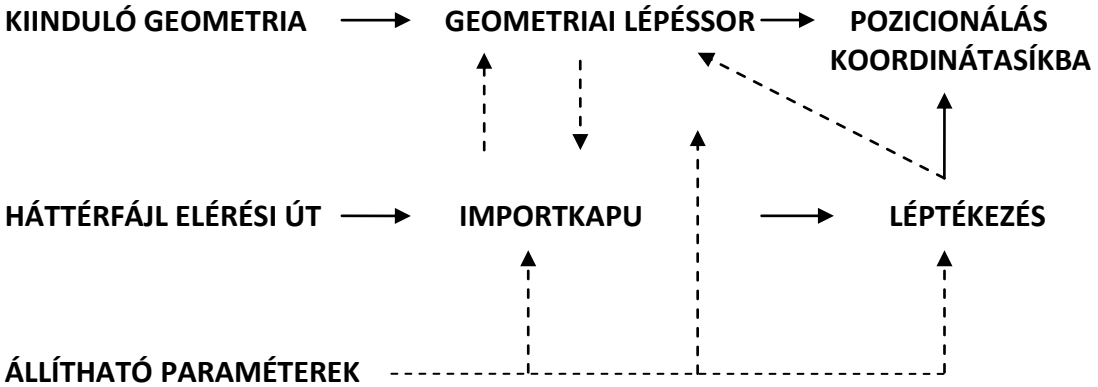
Ez az a megoldás, amely által megoldható, hogy valós elemek legyenek rendelhetőek a különböző algoritmusokhoz. A háttérfájlban könnyedén elhelyezhetőek olyan modellek, amelyek konkrét gyártmányokat reprezentálnak.



9. ábra

A háttérfájlban elhelyezett modellek rendszer szerint vannak pozicionálva, hogy az algoritmus a célzott működés szerint emelje be és orientálja őket. Koordinátatengelyek és különböző hosszok lettek berajzolva külön elemekként, külön fóliákon elhelyezve, rendszer szerinti elnevezéssel. (Egyetelen rossz elnevezés, például elírt hivatkozás a háttérfájl fóliájának nevére tönkreheti az algoritmust.) Ez a rendszer biztosítja, hogy bonyolult elemek lényeges tulajdonságaira (elemhossz, kapcsolodó kötél átmérője, csavarlyuk átmérője) felfigyeljen az algoritmus, és felhasználja ezeket az adatokat (*9. ábra*).

A következő ábra összefoglalja az algoritmusaim *generálműködését*. Minden egyes felirat egy kapcsolási rajzra hasonlító *összefüggésrendszert* jelent a háttérben, a kódolás szintjén. A rendszer a következő:



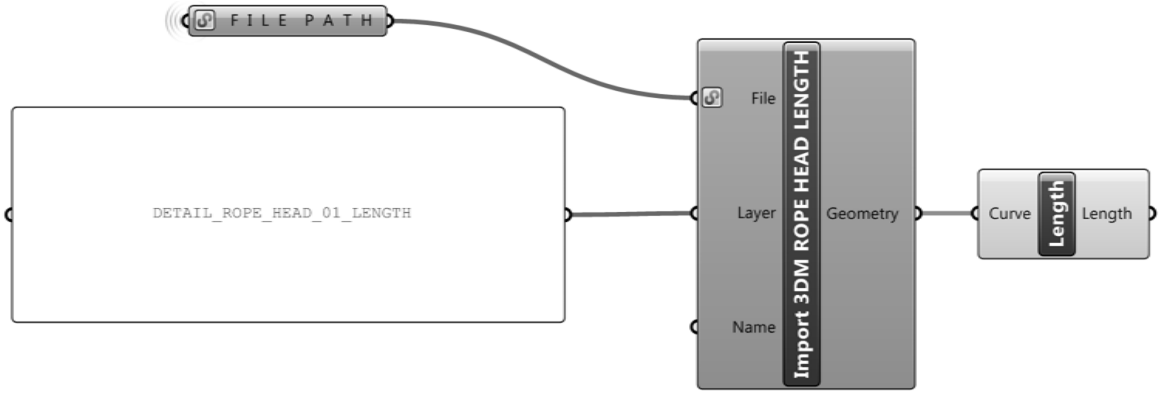
Adott egy ***kiinduló geometria***, amely az adott csomópontnál a *legkézenfekvőbb* a felhasználó számára. A csomópontonként eltérő kiinduló geometriákat bemutatom a Mellékletben algoritmusonként. Fontos, hogy ez a geometria legyen az alapvető "tengelye" az adott szerkesztési szituációnak (pl. két befutó kötél tengelygörbéje, vagy egy árbocszerkezetnél a tölcsérszerű membrán két kontúrköre, stb.). Ezt a felhasználó az általam meghatározott és jelzett nevű *fóliára teszi*, és az algoritmus *aktiválódik*.

Életbe lép egy parancssor, amely *felismeri a* ***háttérfájl helyét*** a számítógépen és meghatározza az elérési utat hozzá (pl.: C:\...\...\...\Desktop\MD_MODELS.3dm).

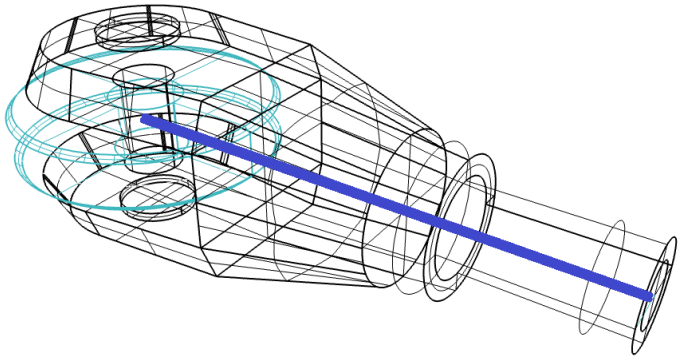
Az elérési út alapján az általam ***Importkapunak*** nevezett kódsor kikeresi a *háttérfájlban* a különböző a szerkezeti elemek modelljeit, kiegészítve az elemek orientációs- és méretvonalaival. Ezzel az elemek (feszítőelemek, kötélfejek, csavarok, stb.) beimportálódnak a felhasználó saját modelljébe a méretadataikkal együtt.

(Az *10. ábrán* látható egy *egyszerűsített importutasítás* kódja: megadjuk a háttérfájlban a keresett elem *fóliájának nevét* (Detail Head Rope 01 Length), melynek a háttérfájlban egy egyenes felel meg: ez a háttérfájlban úgy van a kötélfej modelljéhez rajzolva, hogy hossza pontosan egyenlő a kötélfej hosszával (*11.ábra*). A *File Path* utasítás egyértelműen tartalmazza a fájl elérési útját, és annak tartalmával állandó szinkronban van (ha az alapfájlban változik és lementődik valami, azonnal érzékeli). Az IMPORT 3DM

ROPE HEAD LENGTH utasítás *Geometry outputja* már magát a beimportált görbét jelenti a modelltérben. Ennek pontos hosszát pedig leolvassa a jobb szélső *Length* utasítás. Az Importkapu ezen eleme tehát beimportálta a kötélfej hosszát, mely tetszőlegesen használható tovább, mint számadat.)

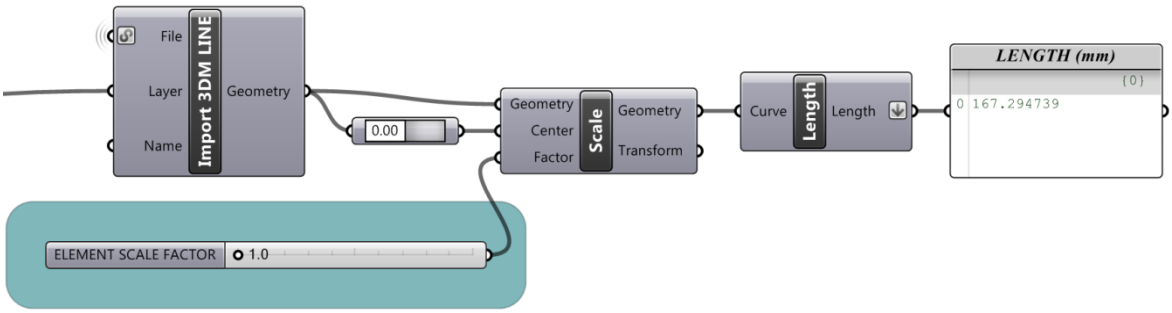


10. ábra



11. ábra

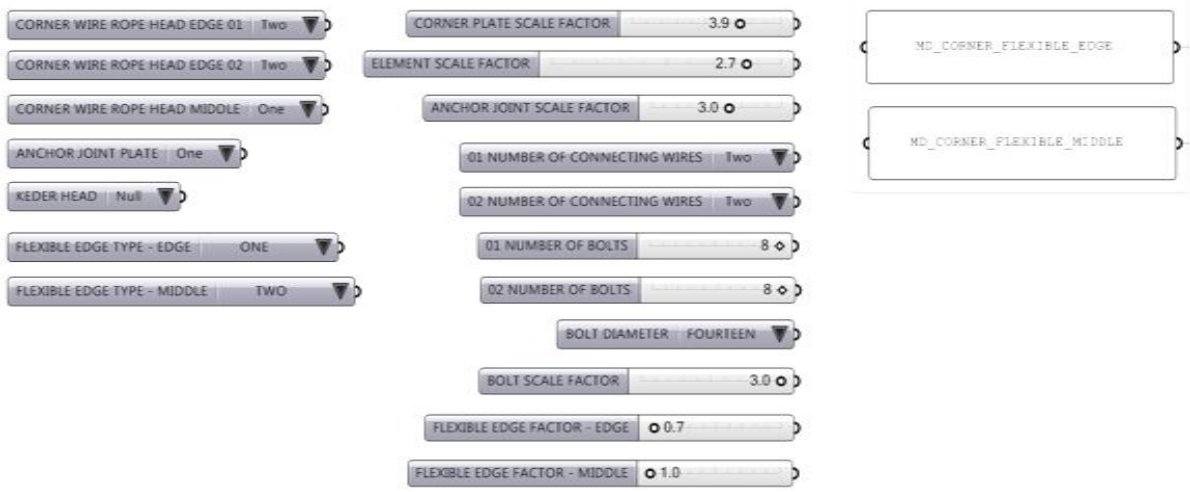
Az ***Importkapuhoz*** kötve rögtön a ***léptékezés*** megy végbe (*12. ábra*): tekintve, hogy minél szélesebb körben alkalmazhatóvá szeretném tenni a rendszert, nincs értelme lekorlátozni az általam eddig ismert elemek méreteire. Ha a felhasználó az általam megadott kötélfejet vagy csavarlyukat épp a 4,2-szeresére szeretné növelni, értelmes és arányos keretek közt semmi akadálya: a csomópont hozzáigazodik. (Értelmet arányú szorzóknál a kód összeomlik.) Valós gyártmányok esetén a léptékezés csak valós méreteknek lesz megfelelő. A léptékszorzók a felhasználónak fel lesznek kínálva az ***állítható paraméterek*** között.



12. ábra

Közben lezajlik az algoritmus legkomplexebb része, a ***geometriai lépéssor,*** amely a felhasználó által megadott egyszerű elemek alapján bonyolult térbeli modellezést hajt végre. Itt többek között metszősíkok, metszőgörbék, irányegyenesek, vektorok, sugaruk segítségével távolságot meghatározó gömbök, vetítések működnek. Minden geometriai lépéssor az adott geometriai helyzettől függ. Már ezeknél a lépéseknél szerves összefüggés van az Importkapuval: az egyes távolságok függnek a *beimportált gyártmányok léptékezett méreteitől* (pl. egy sarokcsomópontban lévő acéllemez vastagsága együtt változik a választott a kötélfej villamagasságával, melytől függ a szorítólap távolsága: minél nagyobb egy kötélfej, annál nagyobb erőket közvetít és ezért annál vastagabb a kapcsolt szerkezet).

A *geometriai lépéssorhoz* szintén hozzákapcsolódnak **az *állítható paraméterek.*** Itt szerepelnek olyan adatok, amelyek a legnagyobb hasznosságot jelentik a felhasználónak a tervezés folyamán, amikor minden képlékeny: a gyártó, az átmérők, az elemszámok,stb. (Itt lehet váltogatni, hogy milyen befutó kötélhez milyen kötélfejgyártmány tartozzon, hányszor legyen toldva egy kötél, és milyen arányban, és így tovább algoritmus szerint.) Itt van például feltüntetve, hogy pl. melyik kötélfejtípust használjuk - ezáltal az állítható paraméterek az importkapuhoz kapcsolódnak egyenesen. A *13. ábrán* látható, hogy mi az, amit a felhasználó az egész algoritmusból *használni fog*: csakis ezeket az *állítható paramétereket*. A jobb oldali két név a fólia neve, amire a felhasználónak a két görbét helyeznie kell, a többi paraméter pedig az a lehetőséghalmaz, amik változtathatóak. A későbbi fejezetben illusztrálva lesz, melyik mit csinál ezek közül algoritmusonként.



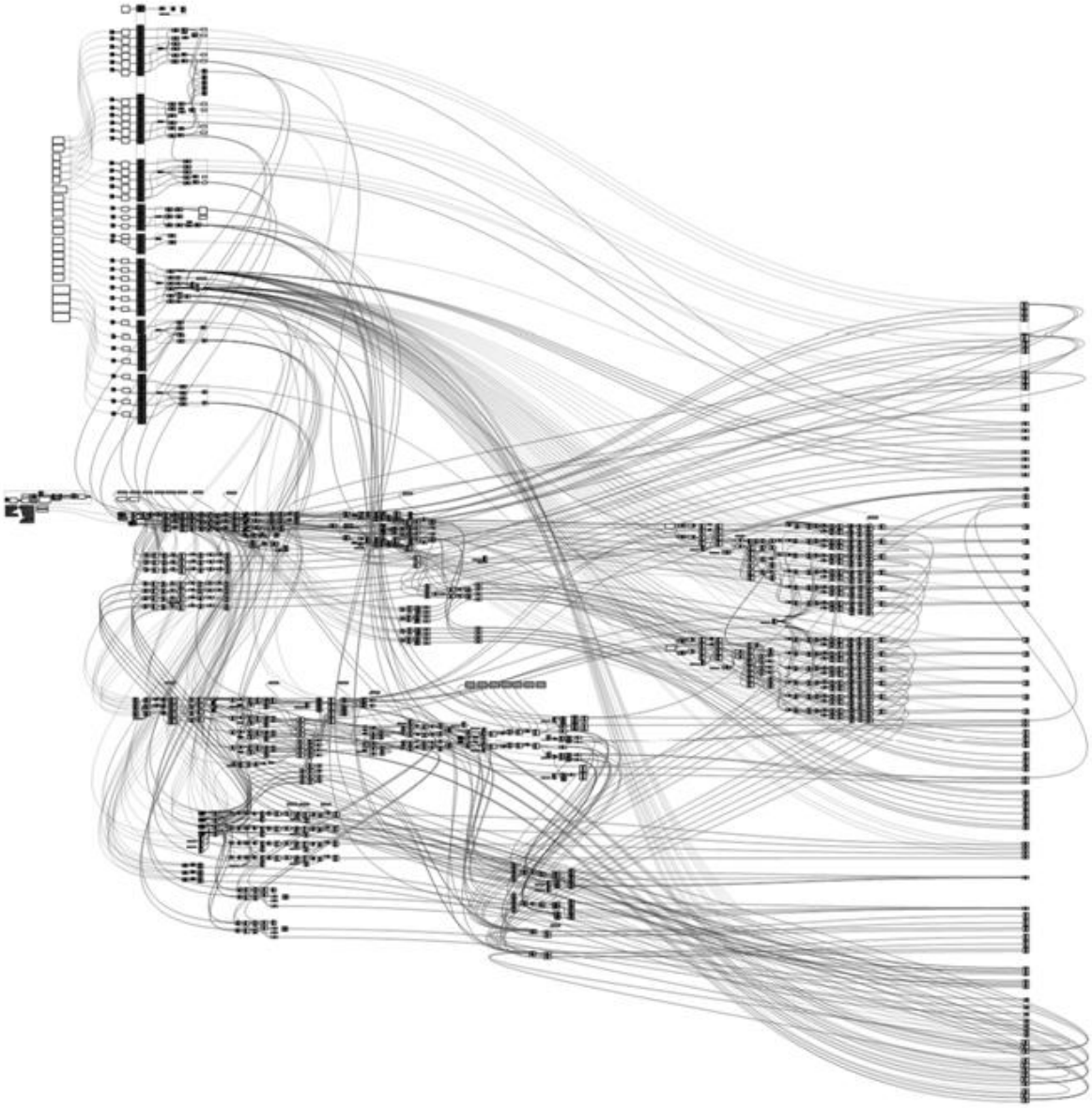
13. ábra

CORNER WIRE ROPE HEAD EDGE 01 - egyes számú szélső sarokcsomóponti kötélfej típusa
CORNER WIRE ROPE HEAD EDGE 02 - kettes számú szélső sarokcsomóponti kötélfej típusa
CORNER WIRE ROPE HEAD MIDDLE - középső sarokcsomóponti kötélfej típusa
ANCHOR JOINT PLATE - lehorgonyzó feszítőlapok száma
KEDER HEAD - kéderkötél kötélfejének típusa
FLEXIBLE EDGE TPYE - EDGE 01 - lágyperem-feszítőelem típusa egyes számú kötélen
FLEXIBLE EDGE TPYE - EDGE 02 - lágyperem-feszítőelem típusa kettes számú kötélen
CORNER PLATE SCALE FACTOR - saroklemez generálléptékszorzója
CORNER SCALE FACTOR - sarokszerkezet kötelelemeinek léptékszorzója
ANCHOR JOINT SCALE FACTOR - horgonylemez léptékszorzója
NUMBER OF CONNECTING WIRES 01 - egyes számú feszítőlap-összekötő kapcsolóelem száma

NUMBER OF CONNECTING WIRES 02 - kettes számú feszítőlap-összekötő kapcsolóelem száma
NUMBER OF BOLTS 01 - egyes számú feszítőlap csavarszáma
NUMBER OF BOLTS 02 - kettes számú feszítőlap csavarszám
BOLT DIAMETER - csavar átmérője
BOLT SCALE FACTOR - csavar léptékszorzója
FLEXIBLE EDGE FACTOR - EDGE - lágyperem-feszítőelem léptékszorzója - szélső kötelek
FLEXIBLE EDGE FACTOR - MIDDLE - lágyperem-feszítőelem léptékszorzója - középső kötél
A *16. ábrán* látható az ezekhez az adatokhoz tartozó kész csomópont.

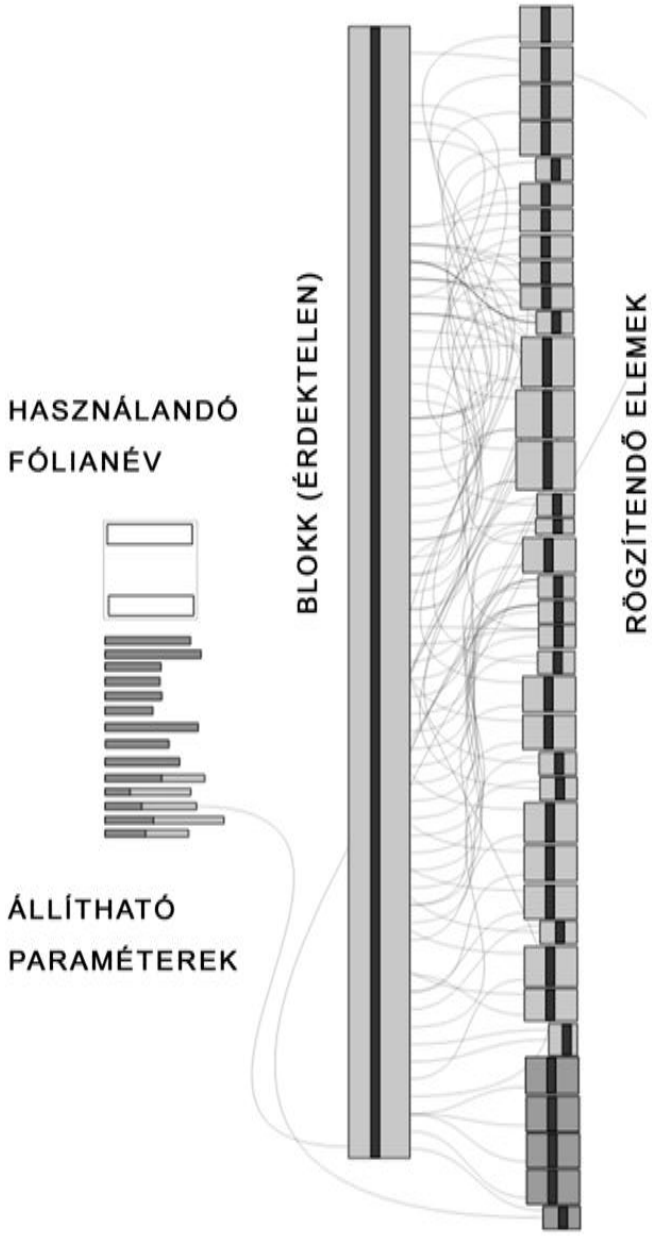
Az importálással és léptékezéssel szorosan összefüggő *geometriai utasítások végeredménye* minden esetben egy sor ***koordinátasík***, amelyekbe a beimportált elemek *belefekszenek* a saját belső koordinátarendszerük szerint. Ez az az orientálás, amely a végén a helyére illeszti az elemeket.

Fontos kiemelni, hogy a felhasználó a lenti bonyolult összefüggésrendszert, a kódot (*14. ábra*) tulajdonképp *nem is látja*. Az ún. ***Cluster*** *utasítással* tetszőleges mennyiségű utasítást egyetlen *blokká* (tehát egyetlen utasítássá) össze lehet húzni, és csak a megmutatni kívánt elemek látszanak (pl. *az állítható paraméterek*). A blokkot kóddal le lehet védeni. A *15. ábrán* látható, hogy összegészében mi látszik az algoritmusból. (Ez majdnem teljesen egyezik a *13. ábrával*, pusztán jobb oldalt az egész algoritmus Clustere van, amellyel tehát nincs teendő, tőle jobbra pedig az egyes elemeket tartalmazó utasítások, melyeken keresztül rögzíthetőek egyenként (vagy egyszerre) a modellek.)



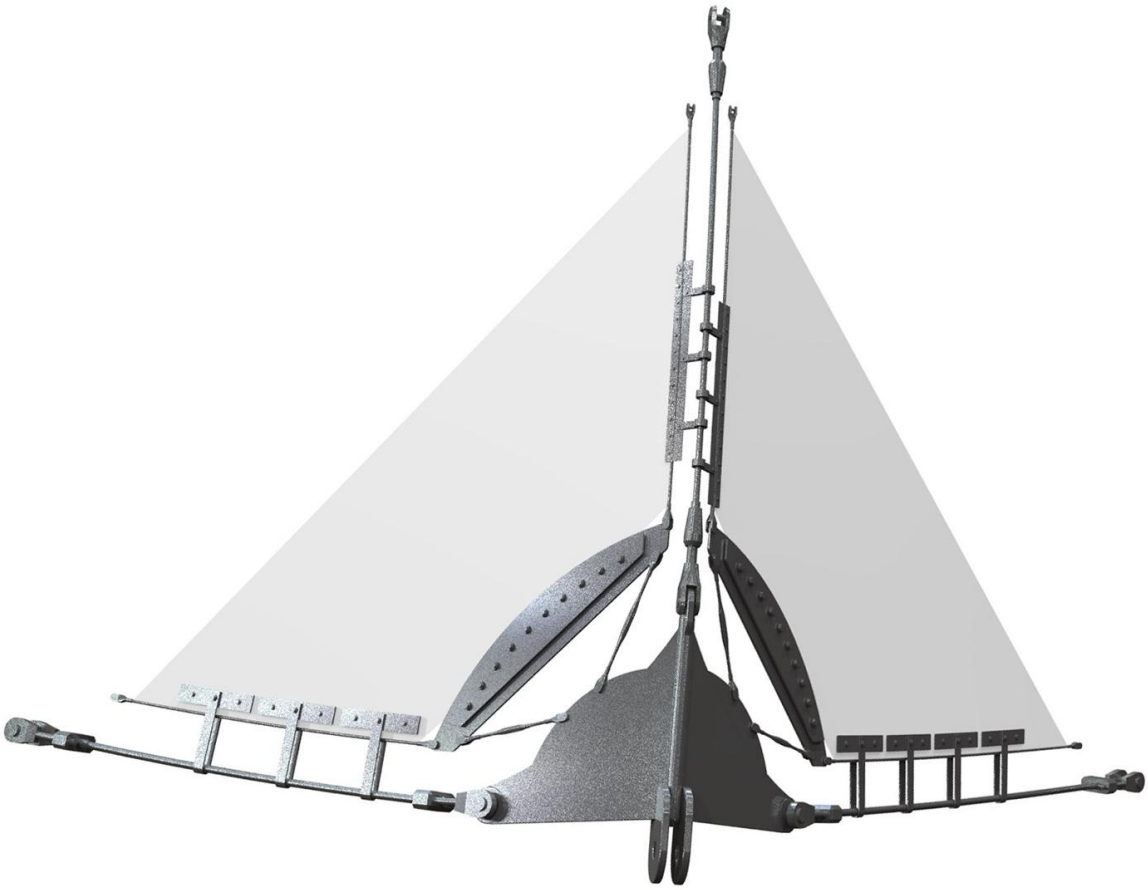
14. ábra

14. ábra - itt látható az MD_CORNER_FLEXIBLE.ghp nevű kész algoritmus a fenti importkapuval, állítható paraméterekkel, geometriai lépéssorral. A jobb oldali oszlopba sorolt *utasítások* egyenlőek magukkal a pozicionált szerkezeti elemekkel a modelltérben: jobb gombbal rájuk kattintva kiválaszható a "sütés" parancs, ezáltal az egyes elemek külön-külön (vagy egyszerre) rögzíthetőek a modelltérben. Magyarázó feliratok, további vizuális csoportosítások természetesen kialakíthatóak, ha ipari érdek esetén az algoritmust nem kell teljesen blokkosítani, tehát ha láttatni kívánjuk egy részét.



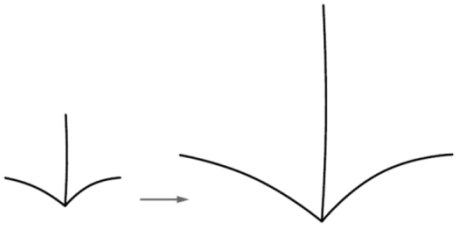
15. ábra

A *15. ábrán* látható minden összefoglalva, amit egy felhasználó lát és használ: válogat az állítható paraméterek közt, miközben a modelltérben elkészül a csomópont. Az "érdektelen" Clusterrel semmi teendője, a jobb oldali oszlopban pedig egymás alatt találhatóak az egyes pozicionált elemeket reprezentáló kódok (egyenként a csavarok, lemezek, stb.) - rájuk kattintva egyenként (vagy többet kijelölve) rögzíthetőek a modellek a modelltérben.



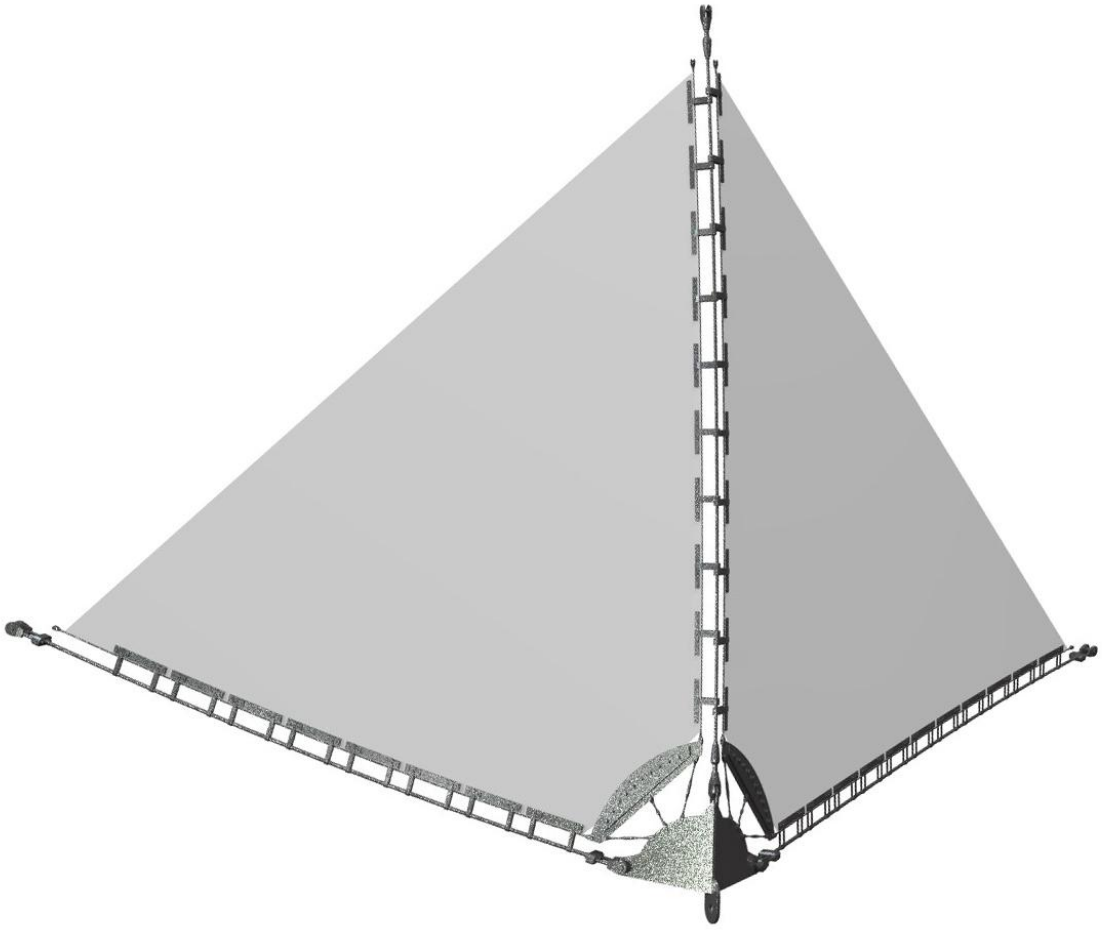
16. ábra

A *16. ábrán* látható a végeredmény: egy három kötélre megkomponált algoritmus modelltérben rögzített formája. Az egyes elemek (a füles peremszorítólapok, a kötélfejek, a horgonylemez száma, a lemezeket összekötő ellenmenetes csavarok száma, stb.) mind variálható (sütés előtt) egyéb gyártmányokra.



17. ábra

A kiindulási alap itt három görbe volt a térben (*17. ábra*). Példaképpen: növelem az alapgörbék léptékét, megváltoztatom a szoírótperemek gyártmánytípusát, az ellenmenetes csavarok számát 2-ről 3-ra, és kialakul az új csomópont (*18. ábra*).



18. ábra

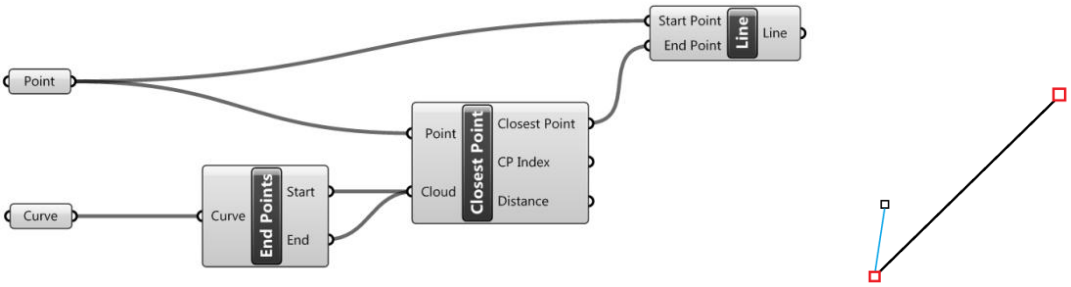
06 PROBLÉMAKÖRÖK

Ebben a fejezetben összefoglalok néhány kérdéskört, amelyek problematikusnak bizonyultak. Ezek általános parametrikus modellezéssel kapcsolatos "informatikai" problémák. Azok számára bírhatnak jelentőséggel, akik ilyen kódok megírásával foglalkoznak.

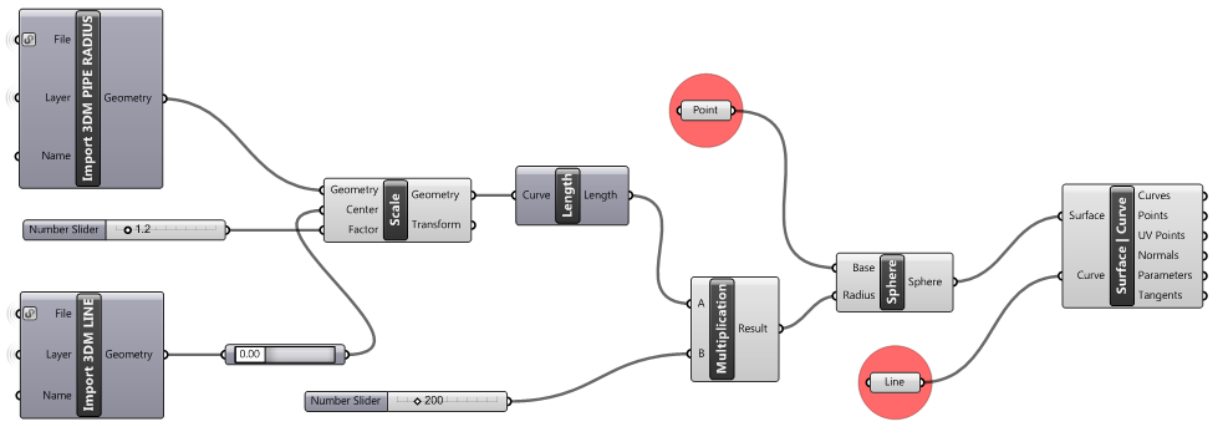
Megjegyezendő, hogy az algoritmusok *geometriai összefüggésrendszerének* részletesebb kifejtése majdhogynem egyenértékű lenne a megírásukkal, ezáltal ennek a prezentációja oktatási helyzetben nyerne értelmet kizárólag. Lépésről lépésre geometriai problémák és programozási kérdések kapcsolódtak össze. A mellékelt tablókon bemutatásra kerül, hogy az egyes algoritmusoknál milyen kiinduló geometriából milyen csomópont jön létre, és milyen választható tulajdonságokkal bír.

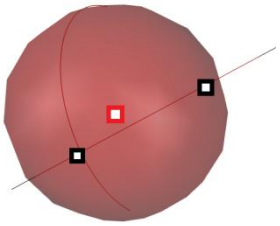
A problémakörök előtt álljon itt egy kevés kifejezetten egyszerű kód ábrája, amelyeken keresztül bepillantást engedek abba, hogy milyen lépéssorok mentén lehet metszéseket, pontmegkereséseket, és egyéb hasonló műveleteket elvégezni.

- Adott pontból kiindulva egy egyenest húzható egy másik egyenes két végpontja közül a közelebbihez:

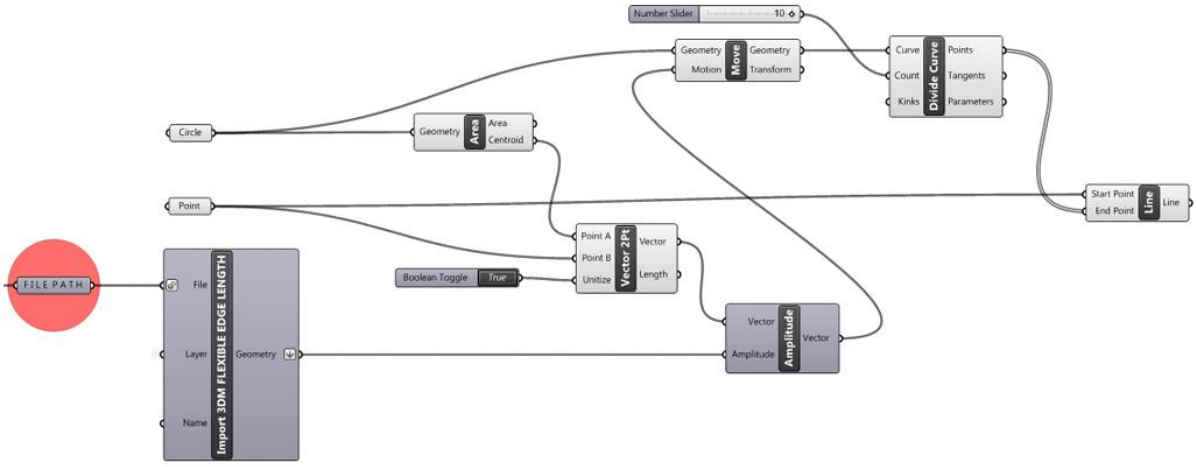


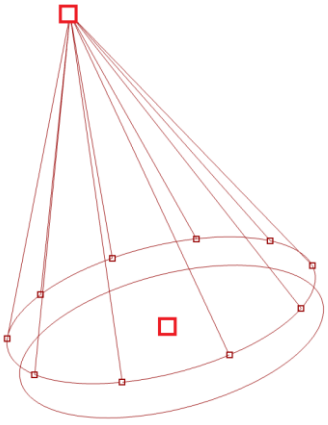
- Beimportáltódik a háttérfájlból két egyenes: egy gyártmányhossz (Import 3dm Line), és egy gyártmányrádiusz (Import 3dm Pipe Radius). A gyártmányhossz nullapontját kiindulópontnak tekintve megnövelhető 1.2-szeresére a gyártmányrádiusz. Ennek a megnövelt vonalnak a hossza további 200-szorosára szorozódik fel. Ez lesz a sugara annak a gömbnek, amely egy tetszőleges pont köré íródik (Point, piros). Az így létrejött gömbnek két metszéspontja van a pirossal jelzett modelltéri egyenessel:





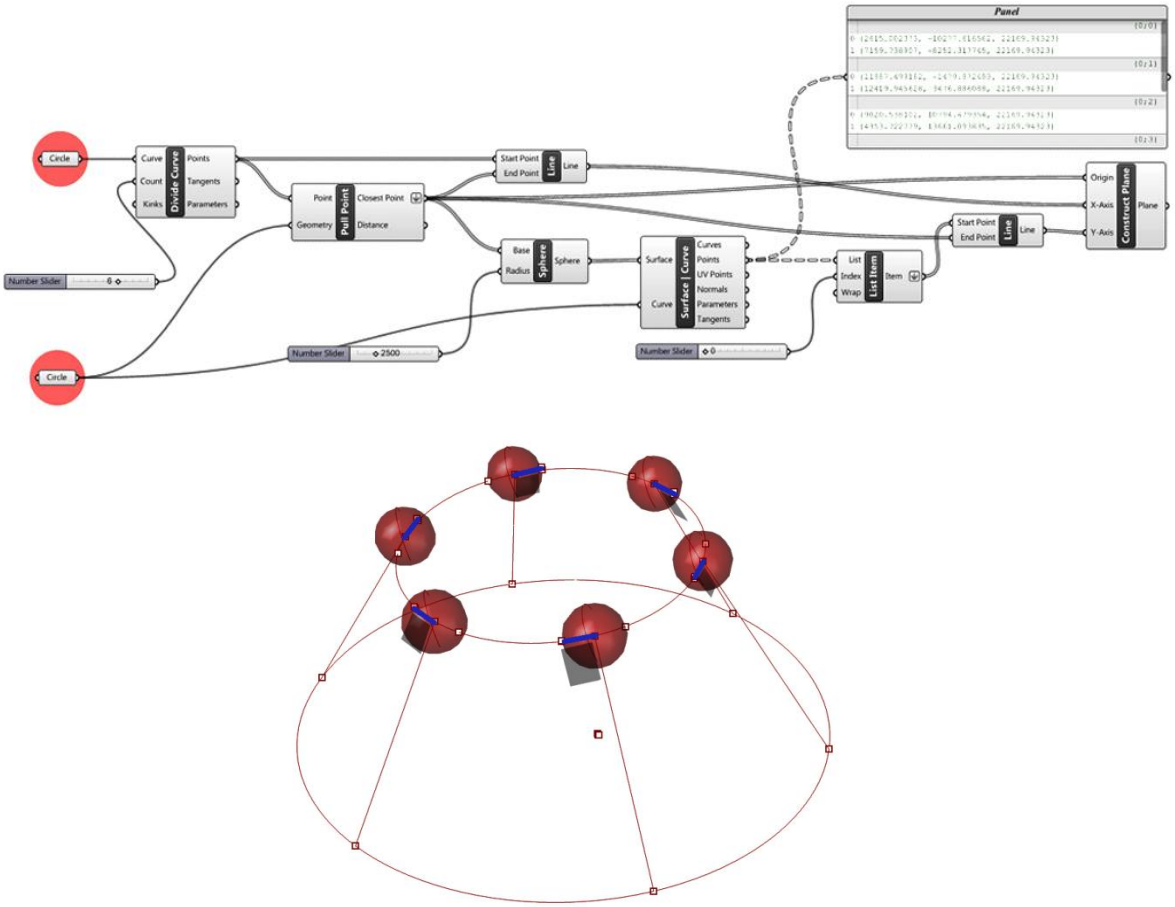
- Adott egy pont és egy kör a modelltérben. Az alapkör eltolódik a pont irányába egy vektorral. A *Vector 2 Pt utasítással* megadható az alapkör középpontjából (*Centroid*) a pontba mutató irány, és ez egységvektorrá konvertálódik (*Unitize*). Ennek az egységvektornak a hossza (=1) felszorzódik egy bizonyos beimportált hosszértékkel (Import 3dm Flexible Edge Length), ez lesz az egységvektor amplitúdója. Ez a vektor lesz az eltolás irány és hossza (Move utasítás Motion inputfül). Az eltolt kört 10 részre oszlik, és ezek et az osztópontok összekötődnek az eredetileg megadott ponttal:





- Adott két kör, az egyik 6 részre bomlik. Mindegyik osztópont "behúz" egy-egy pontot a másik körről (megtalál egy-egy aktuálisan legközelebbi pontot, *Pull point*), majd ezek egy-egy vonallal összekötődnek. Ezeken a megtalált pontokon létrehozódik egy-egy 2500 mm sugarú gömb. Ezek két-két pontot metszenek ki ebből a felső körből:

minden gömbhöz ezáltal két pont tartozik. Ebből a List Item utasítás segítségével kiválasztódik egy-egy. Erről a parancsról még lesz szó, haszna az, hogy egy *Index* segítségével kiválasztódik egy elem egy adatlistából. A *List Item* hatására tehát adott egy-egy pont 2500 mm távolságra minden "behúzott" ponttól, ezek egyenként összekötődnek. Így létrejött egy egyeneshalmaz, amelyek a kör adott pontbeli tangenseitől ugyan kis mértékben eltérnek, de hosszuk meghatározott, paraméteresen állítható (és a körre mutatnak vissza). A két kör pontjait összekötő egyenesek és ezek az utóbbi kísérőegyenesek lesznek annak a koordinátasíksornak a tengelypárosai, amelyeket aztán pozicionálásra használhatóak (szürke lapok):

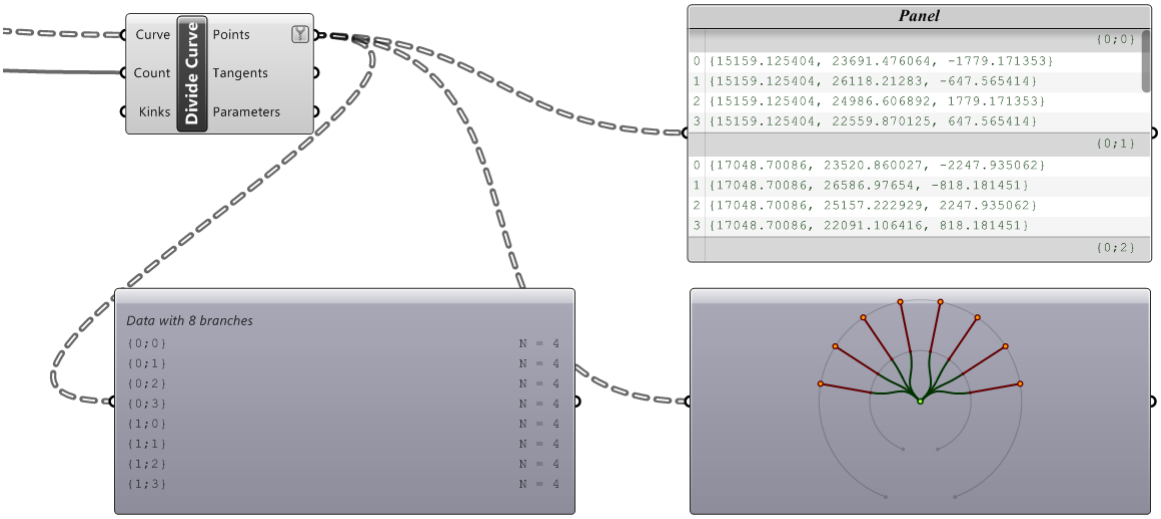


Az adatáramlás problémája

Mint látni lehetett, egy utasítás *outputja* egy másik utasítás *inputja* lehet: az adatok végighaladnak az utasítások között. Elengedhetetlen fontosságú ezáltal az, hogy folyamatosan figyelemmel kísérje a programozó az adatok csoportba, azaz *adatfákba való rendezettségét,* azaz a *listákat*. A rendezettség az egyik legfontosabb tényezője, hogy egy algoritmus használható, vagy használhatatlan eredményt ad.

A *19. ábrán* látható egy utasítás, melyhez hozzákötődik három további: ezek megmutatják, hogy mit tartalmaz az adott utasítás outputja, nem változtatnak rajta. A felső panel konkrétan mutatja, hogy milyen adatokról van szó: koordinátákról. Minden csoportban található négy koordináta. Az alul lévő két panelen ugyanez látszik, csak az ágstruktúra rendszerét kifejezve: a középső ábrán számokkal (Data with 8 branches, stb.:

azaz 8 ág található a rendszer végén, melyek mindegyike négy darab adatot tartalmaz. A 0;0, 0;1 .... 1;2 és 1;3 elnevezések a csoportok nevei. A kezdő 0 és 1 mutatja, hogy két fő ág van, azon belül négy-négy csoport, és azon belül négy-négy adat (koordináta). Pontosan ugyanezt fejezi ki a másik alsó panel vizuálisan: itt ténylegesen látszik az ágstruktúra.



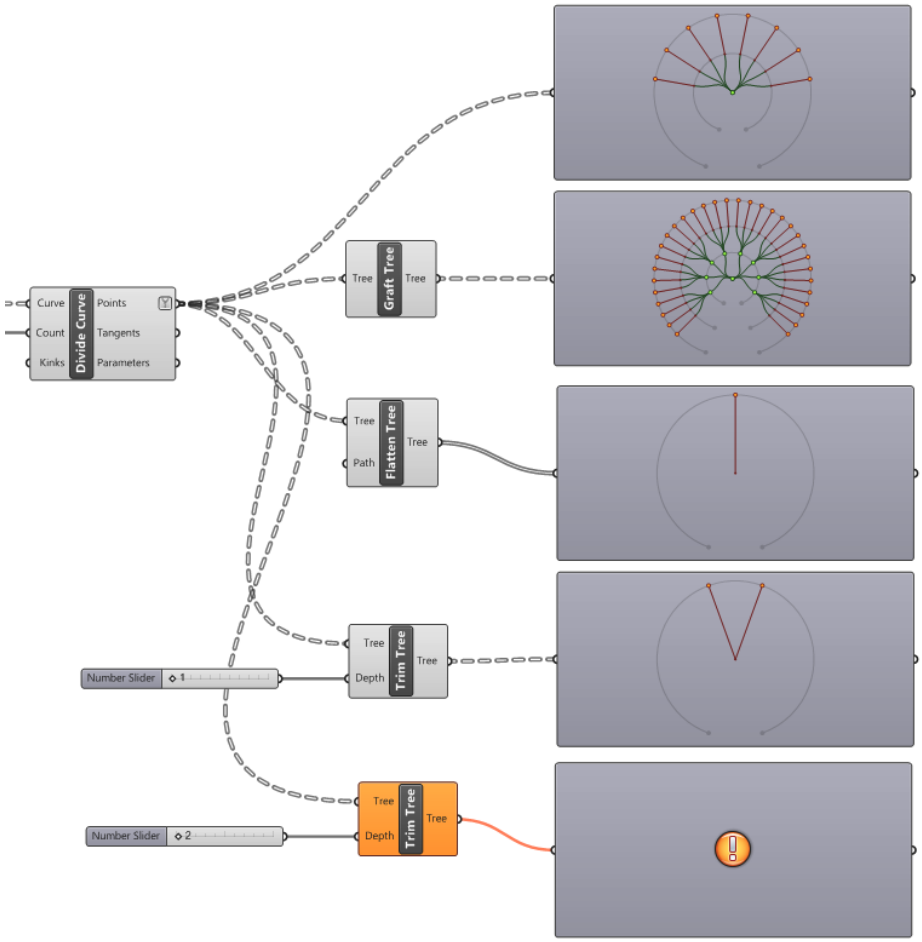
19. ábra

A feszültséget lépésenként az okozza, hogy egyeznie kell egy adott utasítással szembeni elvárásoknak az utasítás belső működésével. Ennek feloldására több *adatrendező utasítás* is rendelkezésre áll, mellyel a struktúrát változtatni lehet. Ez látható a *20. ábrán*, ahol egyetlen utasítás mögé több ilyen adatrendező utasítás is be van kötve, és ezek aztán egy-egy táblán vizualizálva is vannak: látható, hogy mennyire eltérő eredményt lehet létrehozni.

A *Graft Tree* utasítás minden elemet külön ágra rendez, ezzel megnöveli a törzsek (lépcsők) számát, ezzel minden elem egy-egy individdumnak számít külön ágon. Ez például remekül jön, amikor az algoritmus legvégén létrejönnek a koordinátasíkok, és a modellek ezekbe beleorientálódnak: ilyenkor minden egyes koordinátasík külön-külön kell, hogy egy elemet fogadjon (nem cél, hogy egy kötélfej darabjaira szóródjon x db koordinátasík között) .

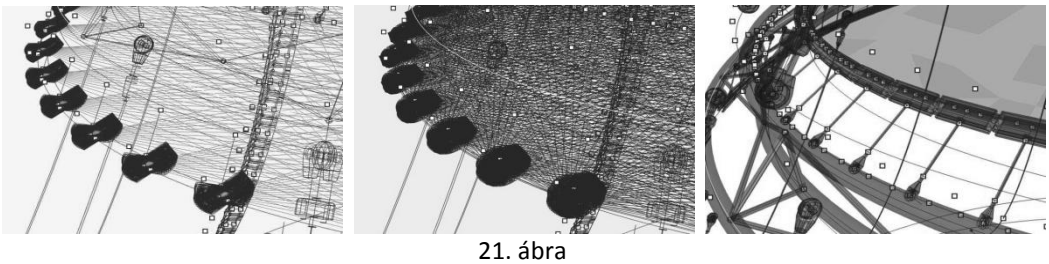
A *Flatten Tree* kompletten eliminálja az ágstruktúrát: minden adat egyetlen törzsre kerül. Nagyon hasznosnak bizonyult minél többször használni ezt, amíg nem áll elő olyan helyzet, ahol már fontos a rendezettség. Az utasítások rendszerint egyenként emelik a lépcsők számát, ezért ha lépésről lépésre eliminálják az ágstruktúrát, nagyon sokáig fenntartható az áttekinthetőség.

A *Trim Tree* egy lépcsőt eltüntet (összevonja az adatokat egy alacsonyabb ágra): marad az eddig belül lévő két fő ág. A *Trim Tree* tetszőleges számú lépcsőt eltüntethet: jelen adathalmaznál 2 lépcső már túl sok, értelmetlen, ekkor már üres az eredményhalmaz. (Ez utóbbi problémára visszatérek a "Trim Tree-váltó"kérdéskörnél.)



20. ábra

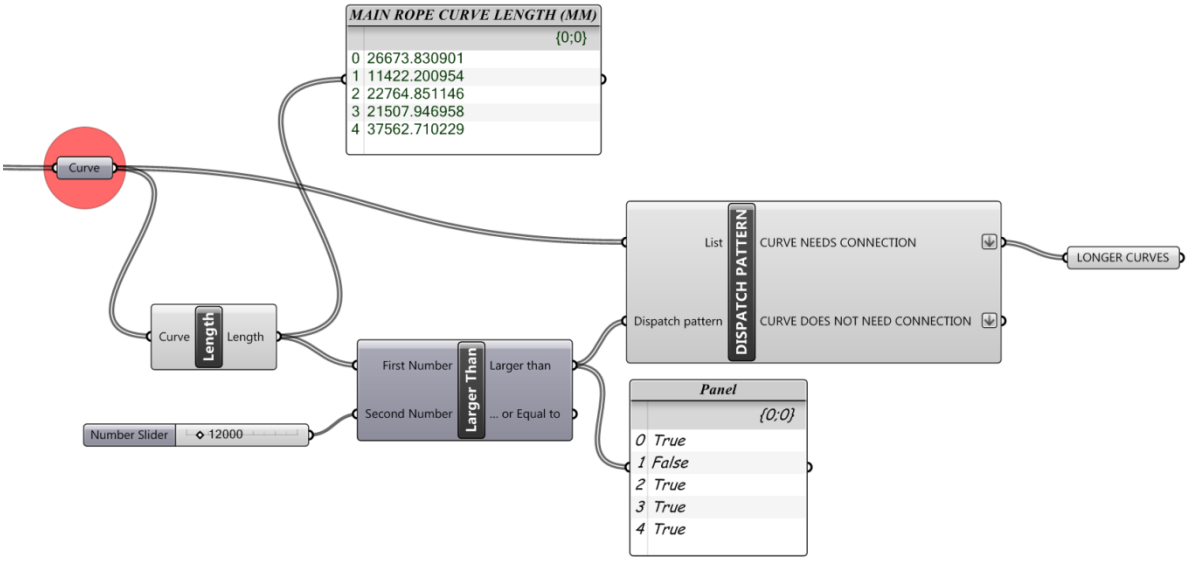
Az *21. ábrán* látható példaképpen egyazon utasítás három kimenetele. Különböző ágstruktúrák miatt a rendszer eltérően értelmezi az összekötendő pontokat, így az ezek alapján a pontok alapján pozicionálni kívánt feszítőelemek használhatatlan eredményt adnak. Jobb oldalt látszik a jó eredmény: elemek sora feszíti a membránt egy horganyzott acélcsőhöz. Ezeket a variációkat az algoritmuson belül pontosan egy-egy adatrendezési utasítás választja el. Mindennek folyamatos figyelemmel kísérése elengedhetetlen.



21. ábra

A "ha" kérdéskör

A GHP rendszerében sajnos nincsen egy egyszerűen használható HA parancs. Amit ennek megkerülésére használtam, az egy *Igaz-Hamis utasítás* és egy *Elválasztó utasítás* kombinációja. A kód az *22. ábrán* látható.



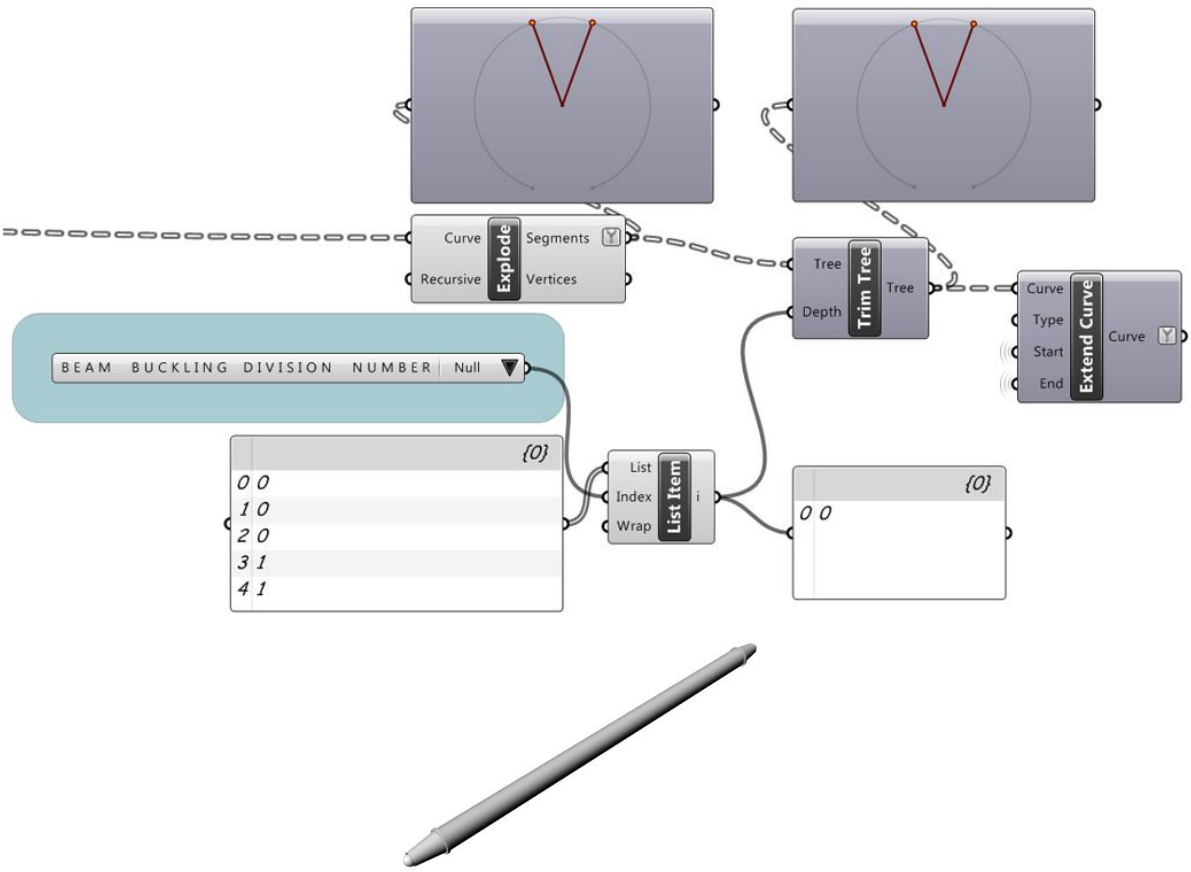
22. ábra

Adott 5 darab tetszőleges görbe a modelltérben, ezekre utal rá a bal felső *Curve* utasítás. A *Length* utasítás leolvassa a hosszukat, ez látható a mellette lévő panelen mm-ben kifejezve. A *Larger Than* utasítás minden egyes hosszról eldönti, hogy hosszabb-e, mint 12 000 mm (ez tetszés szerinti *állítható paraméter*). A *Larger Than* utasítás outputja (*Larger than* fül) tartalmazza görbénként, hogy igaz, vagy hamis az adott görbére nézve, hogy hosszabb, vagy rövidebb az ívhossz 12 000 mm-nél. Ezt az igaz-hamis halmazt követi a *Dispatch pattern*, azaz az *Elosztó utasítás*. Ide a *List* (lista) fülön fut be az az adathalmaz, amely a modelltéri görbékre utal. Így ez az elosztó felbontja a görbelistát olyan görbékre, amelyre nézve igaz, és amelykre nézve nem igaz, hogy hosszabbak 12 000 mm-nél. Az utasítás outputjai már ez a két görbecsoport. Ezt a összefüggést célszerű adaptálni minden ilyen, és ehhez hasonló szituációra, mely szintén nagyon gyakori.

A Mellékletben található algoritmusaim közül a köteleknél jön ez elő a leglátványosabban (mind az ívelt, mind az egyenes tengelyű köteleknél). Ezek a görbék úgy vannak beállítva, hogy a felhasználó *eldöntheti*, a menüből *kiválaszthatja*: hány mm-es ívhosszig toldás nélküli a kötél, hány és hány mm között egyszer toldott a kötél (a felhasználó által megadott arányban), és hány mm felett a felhasználó által megadott számban egyenletesen toldott a kötél. Itt egy, a fentihez hasonló, de kétlépcsős kód dönti el, hogy melyik görbével mi történjen az ív hossza alapján.

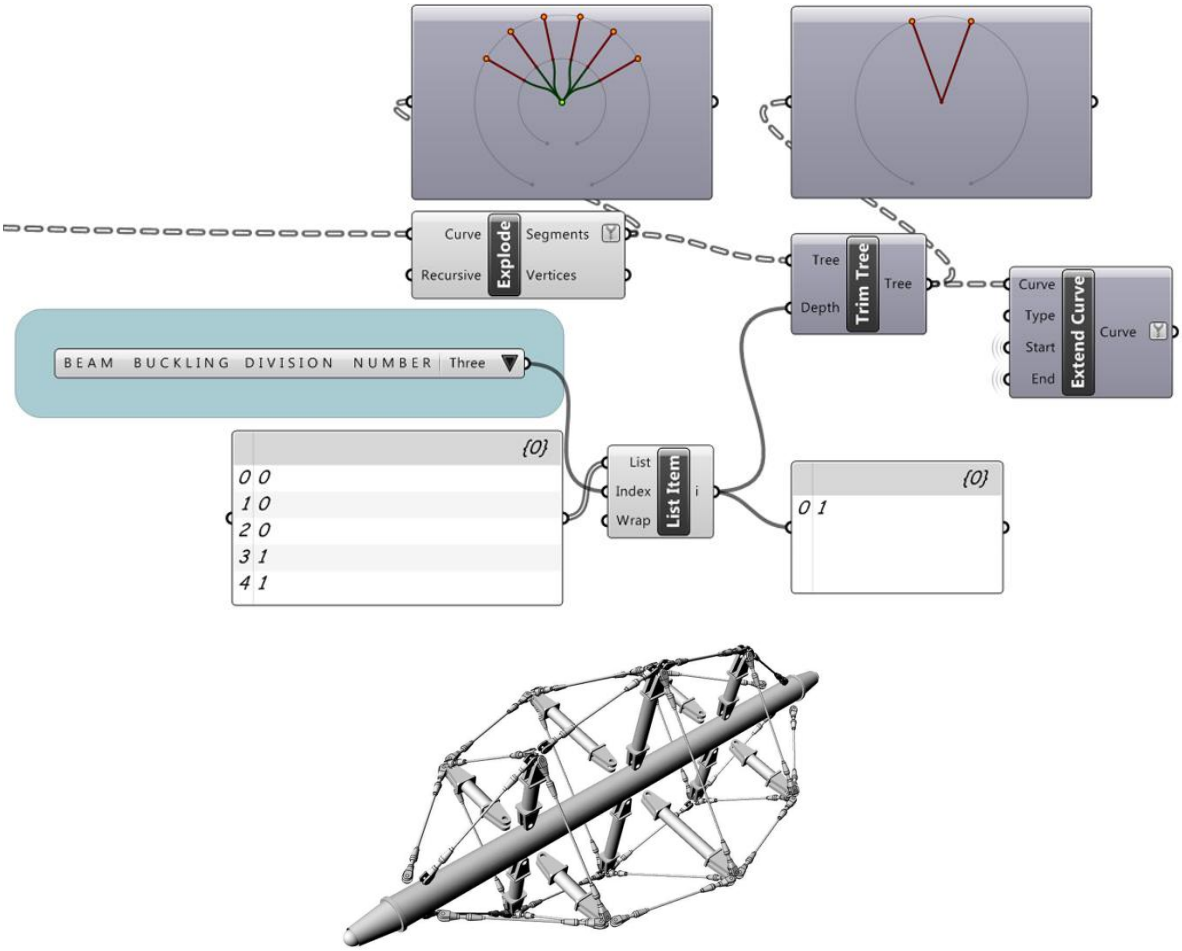
A Trim Tree-váltó kérdéskör

Mint feljebb láttuk, a Trim Tree *általunk meghatározott számú lépcsőt eltörtöl* az ágstruktúrából, a külső ágon lévő adatok "visszazuhannak" az alsóbb lépcsőre, elkülönülésük megszűnik, a különböző ágak külön tartalmai egy csoporttá, törzzsé állnak össze. Gyakran előfordul azonban, hogy az *állítható paraméterekkel* nem együtt változik az eliminálandó lépcsők száma! Álljon itt egy erősen leegyszerűsített példa.



23. ábra

23. ábra: Ha egy oszlop kihajlás ellen egy helyen sem támasztódik meg (BEAM BUCKLING DIVISION NUMBER = NULL, akkor az ehhez kapcsolt tartalmak ésszerű ágstruktúrája a *két törzs* (nem lényeges, hogy miért, ennek az algoritmus későbbi részén van jelentősége). Látható, hogy a felső Explode parancshoz kötött panel mutatja, hogy a beérkező adathalmaz eleve két ágra bomlik, és cél, hogy ez így maradjon. Eltörtölni tehát 0 lépcsőt kell ebből az ágstruktúrából. A Trim Tree utasítás *Depth inputfülén* megérkező szám a 0 kell, hogy legyen, hiszen nem cél eltörölni egy lépcsőt sem. Ide az alatta lévő *List item utasításból* fut be a 0 érték, ez leolvasható a jobb alsó panelen is. Látható a jobb felső panelen, hogy a Trim Tree utasítás outputja pontosan ugyanaz az ágstruktúra, mint előtte volt.

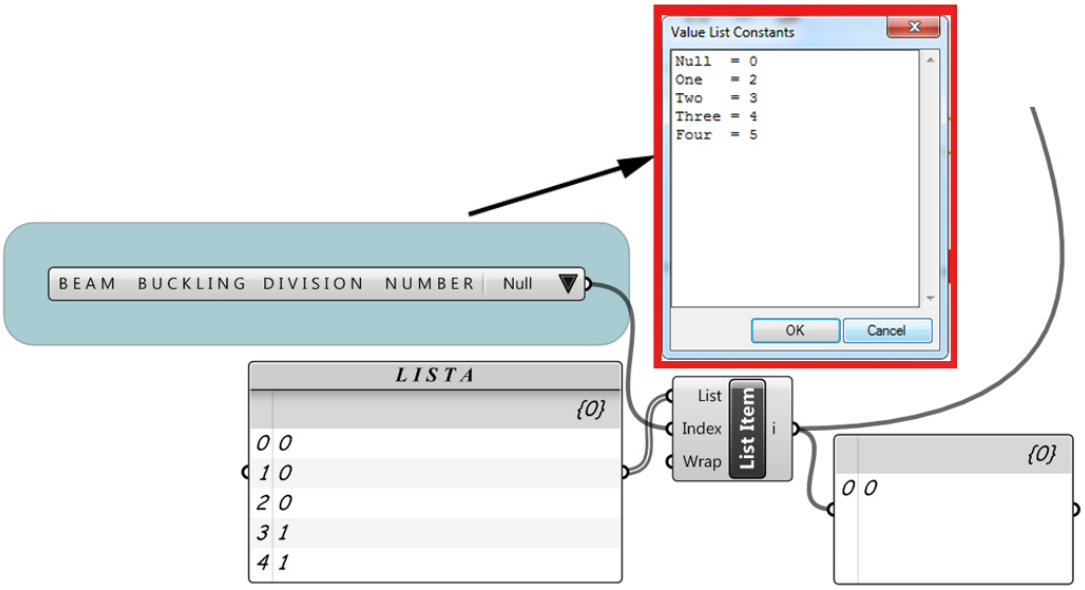


24. ábra

24. ábra: Ha viszont három helyen támasztódik meg az oszlop, akkor szintén a két törzs az algoritmus számára a jó ágstruktúra (egy továbbra is érdektelen indok miatt). Látható viszont, hogy itt a fent beérkező ágstruktúra kiterjedtebb, mint később szükséges. A Trim Tree-vel most tehát egy lépcsőt kell eltörölni, így a *Depth inputfülön* most az 1 érték kell, hogy befusson. Látható a jobb alsó panelen, hogy a *List Item utasítás* most ezt az értéket közvetíti.

Ilyen belátható számú esetben (a projektnél ilyenek fordultak elő) tehát ez a megoldás! *25. ábra*: Szükségünk van egy listára, amelyet egy szimpla üres panelbe be kell gépelni. A lista sorszáma (a panel bal oszlopa) 0-4-ig terjed, az értékek mellette 0,0,0,1,1 egymás alatt. A 2. sorhoz pl. a 0 érték kapcsolódik. A List Item utasítás ebből a Listából választ egy elemet a *sorszáma alapján,* ez a List item *Index nevű inputfülén fut be.* Ezáltal a kékkel keretezett BEAM BUCKLING DIVISION NUMBER érték, ami egy *állítható paraméter a felhasználó számára* (0-tól 4-ig választhatja ki, hogy hány helyen támasztaná meg az oszlopot), egyben egy *Indexként* is működik a List Item számára! Az, hogy mit tartalmaz a BBDNumber, a pirossal keretezett mezőben látszik, ez a tartalma. Ha a felhasználó három helyen támasztaná meg az oszlopot, akkor kiválasztja a Three feliratot, de ez a *négyes számot jelenti a kód számára*. A négyes számhoz a LISTÁBAN az 1 érték tartozik, a List Itemen keresztül ez halad tovább a korábbi *Trim Tree*

utasításhoz. (Az ábrán épp a Nulla megtámasztás van kiválasztva, ehhez a nulla érték tartozik.)



25. ábra

Ez a saját kezűleg megírt Lista tehát egy váltóként működik, és leköveti a felhasználó igényeit a program nyelvén. Nagy számú elemsoroknál gyakran felléphet az igény a használatára.

07 ÖSSZEFOGLALÁS

A projekt alapvetése, hogy gyakorlati használhatóságot kölcsönözzek meglévő membráncsomópontjaimnak, a bemutatott rendszer szerint megvalósíthatónak bizonyult a Grasshopper parametrikus modellezési rendszerében. Létrejött egy olyan algoritmuscsokor, amelyen keresztül bemutatható a rendszer működése, és amely stabilan és gyorsan alkalmazható, beépíthető valós tervezési szituációba.

Továbbfejlesztés esetén előre meghatározandó, hogy milyen jellegű csomópontokra van a leginkább szükség (melyek a legnehézkesebb modellezési helyzetek), és ezen csomópontoknál melyek a *legkézenfekvőbb kiindulási elemek* és *leginkább használt állítható paraméterek*. Beépítendőek a rendszerbe további ipari termékek modelljei a tervezési pontosság fokozása érdekében. Nem szükséges, de a hasznosságot tovább tudja növelni, ha a membrán *generálalakja* (leendő főbb kontúrjait vagy alakját meghatározó egyéb tényezőket, pl. belső megtámasztásokat) előre meghatározott, és azt figyelembe véve készítül el az algoritmus.

A rendszer adaptálható membránszerkezetektől eltérő konstrukciók modellezési helyzeteire is (pl. pontmegfogásos homlokzati üvegszerkezetek, kötélhálók, stb.). Tulajdonképp a membránfelület (mint a rögzített modell egy eleme) *nehezíti* az algoritmus alapvető működésének kialakítását. (Például két görbéből egyértelműen megalkotható egy komplex sarokszerkezet, de a két görbe között kifeszített membránnak marad egy nyitott harmadik oldala, melynek lezárása (pl. egy egyenessel) kis valószínűséggel egyezik csak meg az adott tervezési helyzettel. Ettől persze továbbra is jól használható az algoritmus, de ha a felület modelljét is minél inkább közelíteni cél az igényekhez (mint ezt feljebb is írom), érdemes minél inkább komplexen meghatározni előre, milyen "kontúrú" membránhoz fog a csomópont kötődni.)

A dolgozat Mellékletében megtalálható az algoritmusokból kinyert vizuális ábraanyag a részletes bemutatás céljából. A *www.membranedetail.com* honlapon hamarosan elérhető lesz egy kipróbálható algoritmus, illetve a korábbi ábraanyag is modernebb formában lesz megtekinthető.

Köszönettel tartozom Dr. Hegyi Dezsőnek a rendszer kigondolásában és a dolgozat elkészítésében nyújtott segítségéért.

08 FORRÁSJEGYZÉK

- Zubin Khabazi: *Generative algorithms (using Grasshopper)*© 2012 (www.morphogenesism.com)

- Andrew Payne & Rajaa Issa: *The Grasshopper Primer, Second Edition - for version 0.6.0007*© 2009

- Füzes Bálint Péter: *Feszített membránszerkezetek részletmegoldásai - virtuális csomópontgyűjtemény - www.membranedetail.com* (2012 TDK, BME ÉPK, Szilárdságtani és Tartószerkezetei Tanszék, konz.: dr. Hegyi Dezső)

- Füzes Bálint Péter: *Membránszerkezetek megtámasztási megoldásai - sarokcsomópontok (2011, TDK BME* ÉPK, Szilárdságtani és Tartószerkezetei Tanszék, konz.: dr. Hegyi Dezső)*)*

- http://www.membranedetail.com

- http://www.grasshopper3d.com/

-http://www.grasshopper3d.com/forum

- http://www.tensinet.com

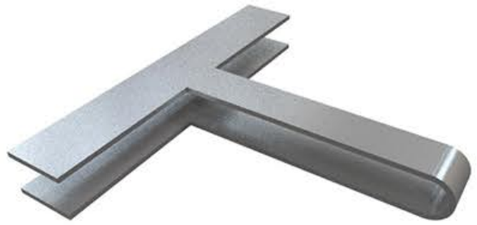
- http://www.pfeifer.de/

MEMBRANE DETAIL
PARAMETRICS

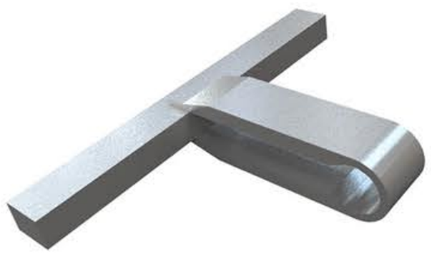
Tudományos Diákköri Konferencia 2015/2016 - I. szemeszter
Budapesti Műszaki és Gazdaságtudományi Egyetem
Építészmérnöki Kar - Szilárdságtani és Tartószerkezeti Tanszék
Füzes Bálint Péter - Konzulens: dr. Hegyi Dezső

Membránszerkezetek parametrikus modellgyűjteménye

AZ ALGORITMUSGYŰJTEMÉNYBEN FELHASZNÁLT 3D-S MODELLEK

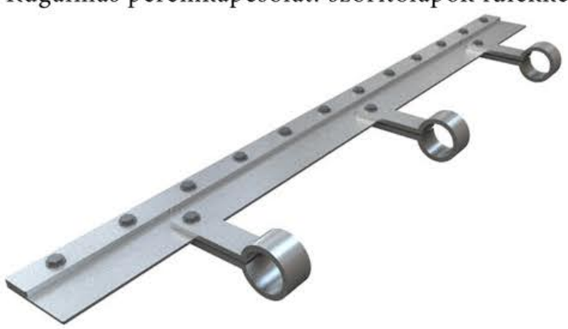


Rugalmas peremkapcsolat: szorítólapok fülekkel - 01 - egyszerűsített modell 01

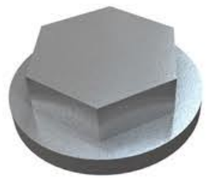


Rugalmas peremkapcsolat: szorítólapok fülekkel - 02 - egyszerűsített modell 01

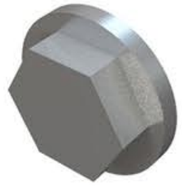
Rugalmas peremkapcsolat: szorítólapok fülekkel - 03 - egyszerűsített modell 03



Rugalmas peremkapcsolat: szorítólapok fülekkel - 04 - egyszerűsített modell 04



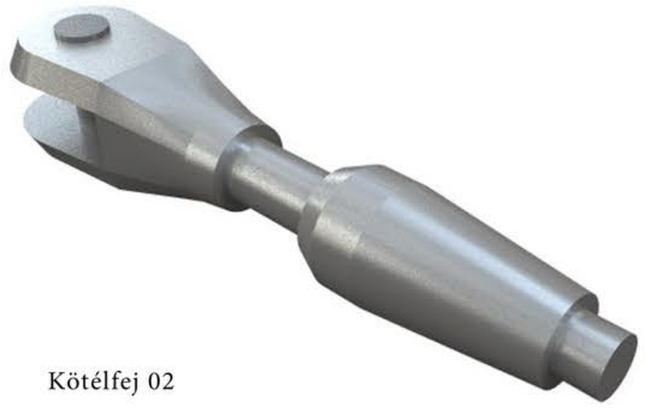
Csavarfej - vízszintes



Csavarfej - függőleges

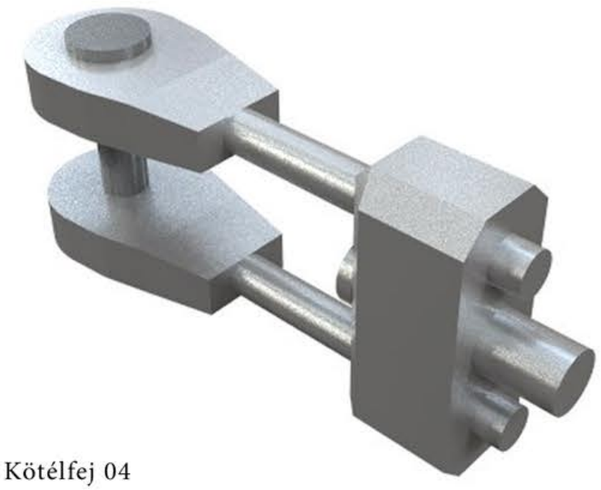
Kapcsolólemez

Kötélfej 01



Kötélfej 02

Kötélfej 03

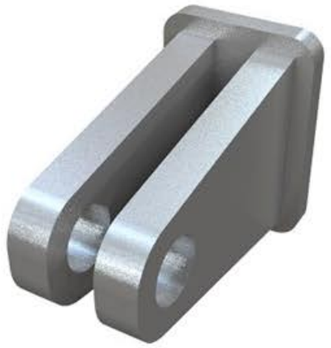


Kötélfej 04

MEMBRANE DETAIL
PARAMETRICS

AZ ALGORITMUSGYŰJTEMÉNYBEN FELHASZNÁLT 3D-S MODELLEK

Csuklós támaszelem egy lemezzel



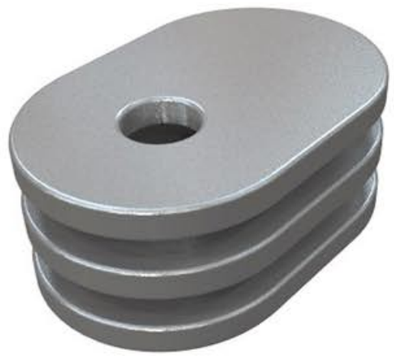
Csuklós támaszelem két lemezzel

Gömbcsuklós támaszelem

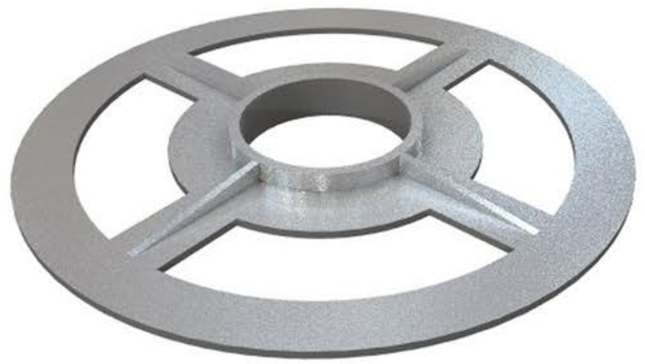
Kötélfej fém kéderhez

Egyszeres fém kapcsolólemez

Kétszeres fém kapcsolólemez

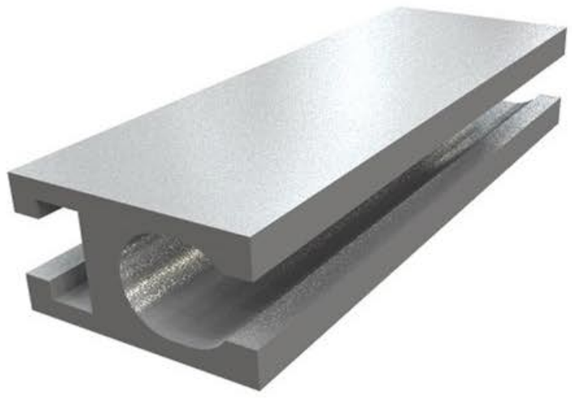


Háromszoros fém kapcsolólemez

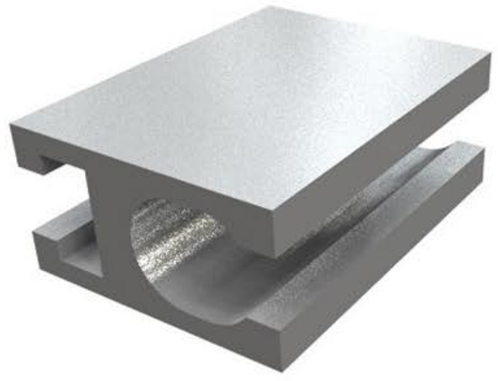


Fém rögzítőlemez

Kapcsolóelem árboccsúcson



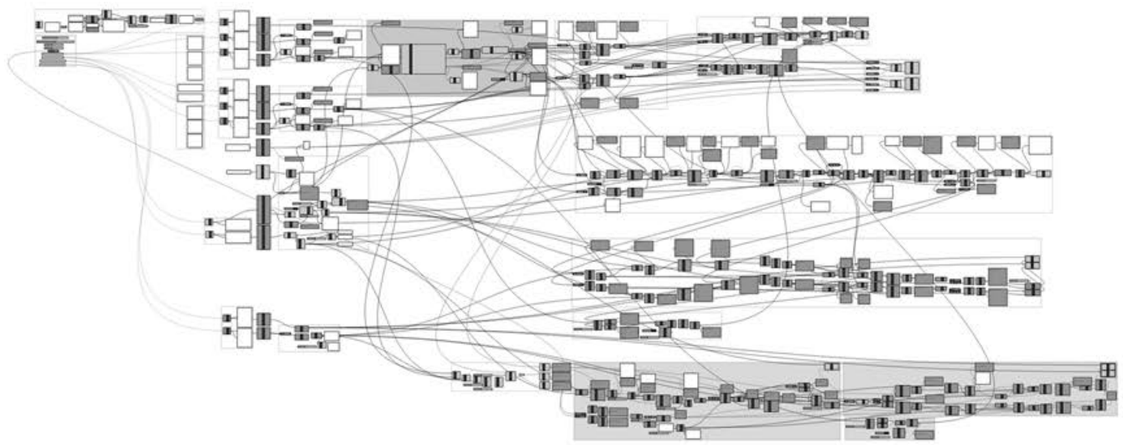
Kéderfeszítő profil



Kéderfeszítő profil

MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 01 - ÍVELT KÖTÉLSZERKEZET



Az MD_WIRE_CURVED.ghp ívelt kötélszerkezetet működtető algoritmus tetszőleges számú *síkgörbét* alakít át kötélekké. A kötelek végére kerülő kötélfejek belefekszenek a kötél síkjába, ezzel megfelelnek a valós elfordulási lehetőségnek. A felhasználó által a menüben meghatározható két számszerű paraméter két mm értéket jelent. Az első számnál rövidebb görbék toldás nélküli kötelekké válnak. A két szám közötti hosszúságú görbék egyszer toldott kötelekké válnak, a toldás arányát a menüben látható *slider*rel lehet tetszőlegesen változtatni. A második számnál hosszabb görbék többször toldott kötelekké válnak, ekkor a toldás száma a menüben meghatározható, eloszlása egyenletes.

MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 01 - ÍVELT KÖTÉLSZERKEZET

általános csomóponti léptékszorzó

toldott kötél szegmenseinek száma

első határszám (mm) - rövidebb kötél toldás nélkül

második határszám (mm) - eddig egyszer toldás

egyszeri toldás arányslidere

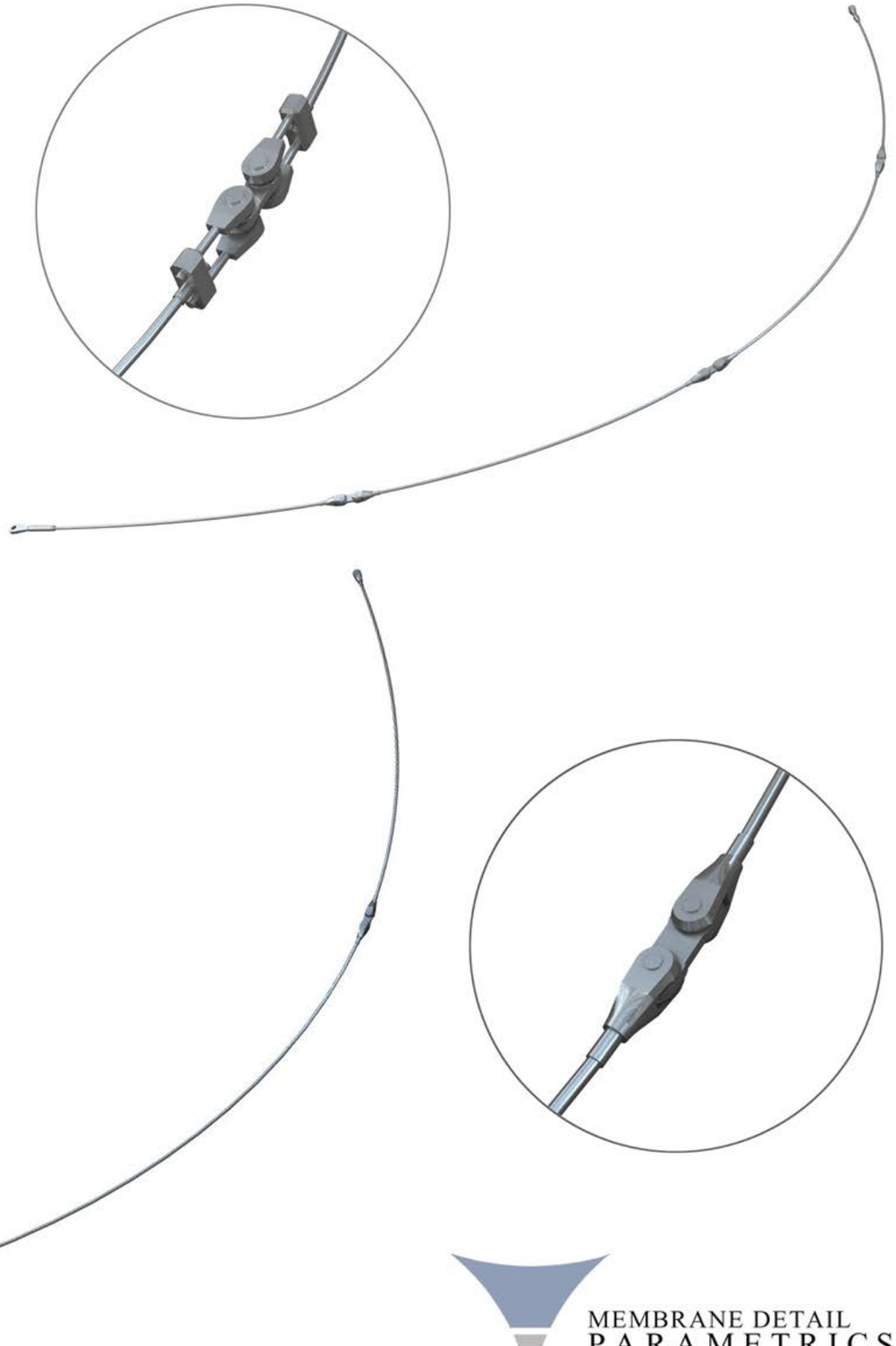
egyik oldali kötélfej típusa

második oldali kötélfej típusa

kapcsolólemez típusa

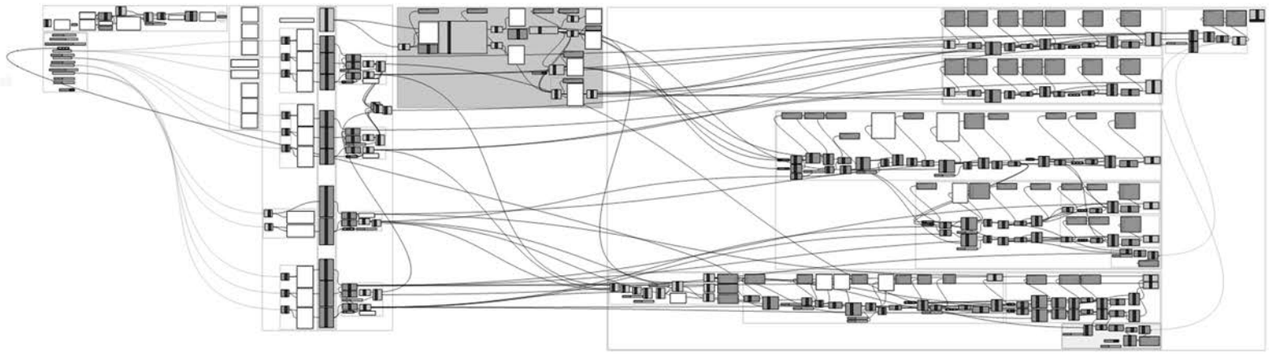
toldásnál lévő kötélfej típusa

a kiinduló görbesor erre a fóliára helyezendő

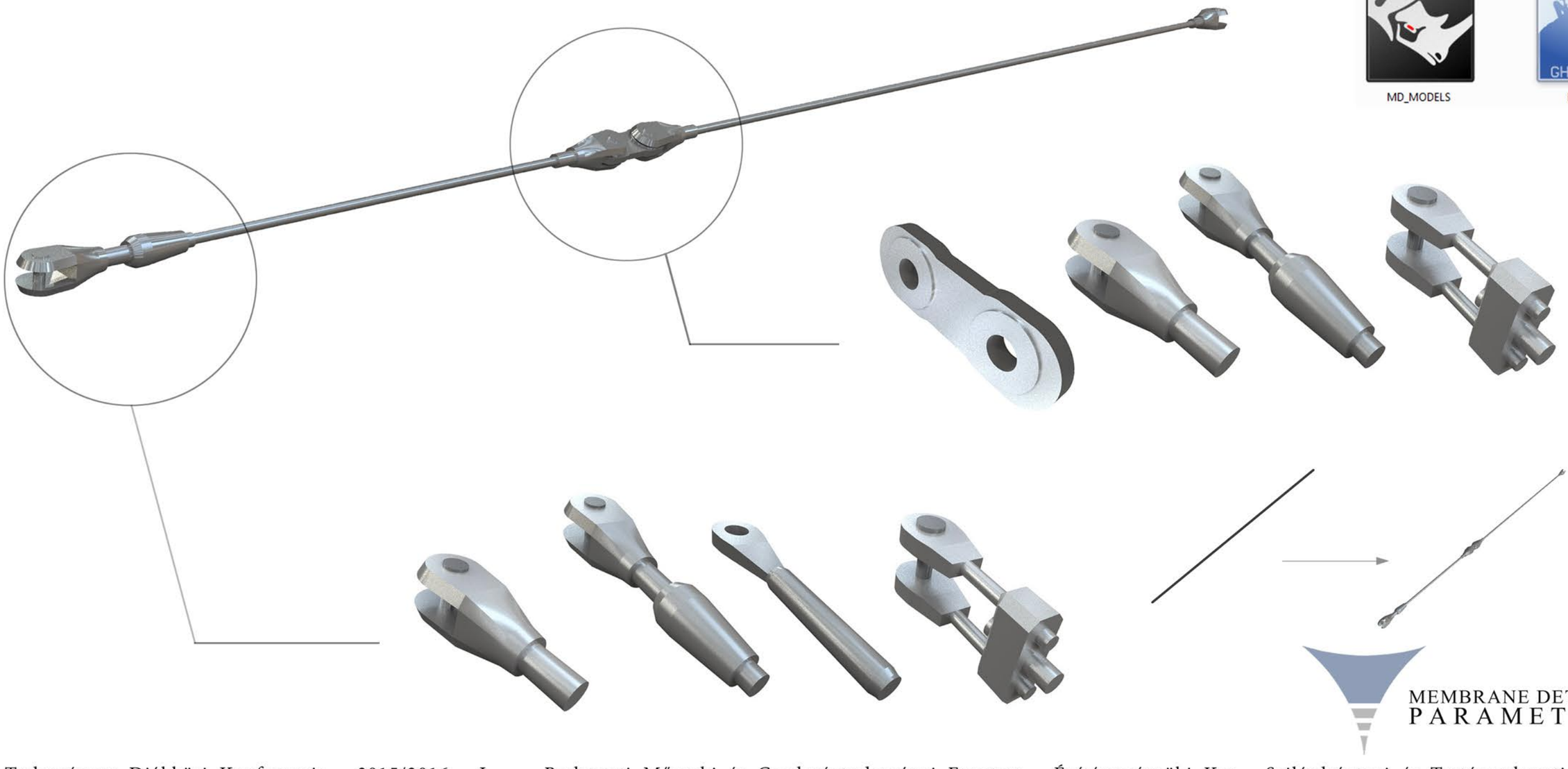


MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 02 - EGYENES KÖTÉLSZERKEZET



Az MD_WIRE_CURVED.ghp egyenes, húzott kötélszerkezetet működtető algoritmus tetszőleges számú *egyenest* alakít át kötélekké. A kötelek végére kerülő kötélfejek a menüben felajánlott *sliderrel* forgathatóak a kötél tengelye körül külön-külön, ezáltal nem korlátozódik le a használat. A felhasználó által a menüben meghatározható két számszerű paraméter két mm értéket jelent. Az első számnál rövidebb görbék toldás nélküli kötelekké válnak. A két szám közötti hosszúságú görbék egyszer toldott kötelekké válnak, a toldás arányát a menüben látható *slider*rel lehet tetszőlegesen változtatni. A második számnál hosszabb görbék többször toldott kötelekké válnak, ekkor a toldás száma a menüben meghatározható, eloszlása egyenletes. Külön egységként forgatható a toldási szerkezet, mint elemegyüttes.



MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 02 - EGYENES KÖTÉLSZERKEZET

általános csomóponti léptékszorzó

többször toldott kötél szegmenseinek száma

egyszeri toldás arányslidere

egyik oldali kötélfej típusa

egyik oldali kötélfej forgatóslidere

második oldali kötélfej típusa

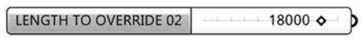
másik oldali kötélfej forgatóslidere

kapcsolólemez típusa

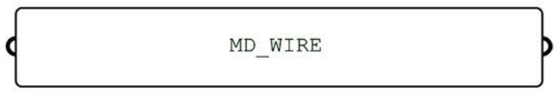
kapcsolólemez forgatóslidere

toldásnál lévő kötélfej típusa

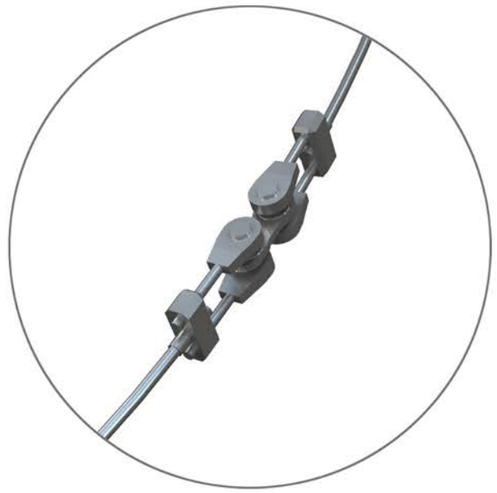
első határszám (mm) - rövidebb kötél toldás nélkül

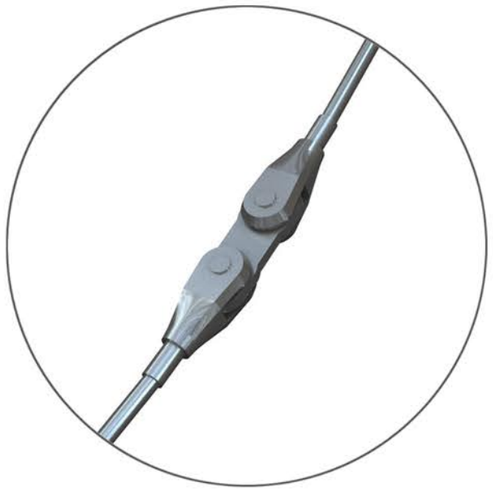


második határszám (mm) - eddig egyszer toldás



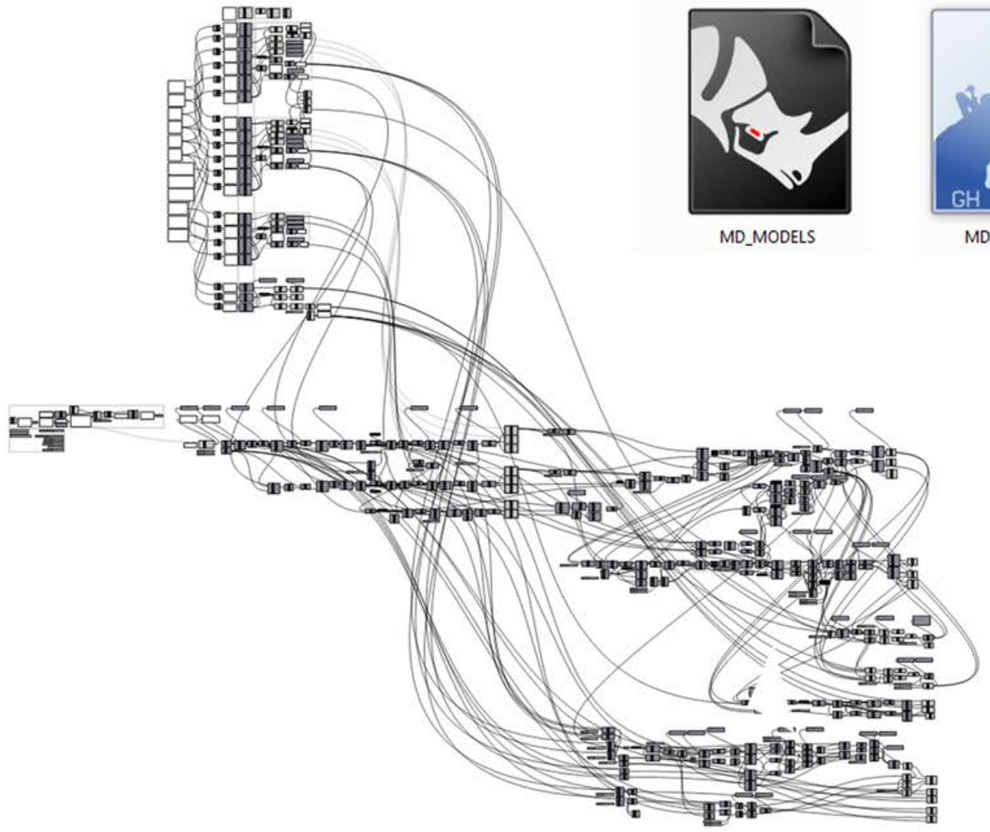
a kiinduló görbesor erre a fóliára helyezendő



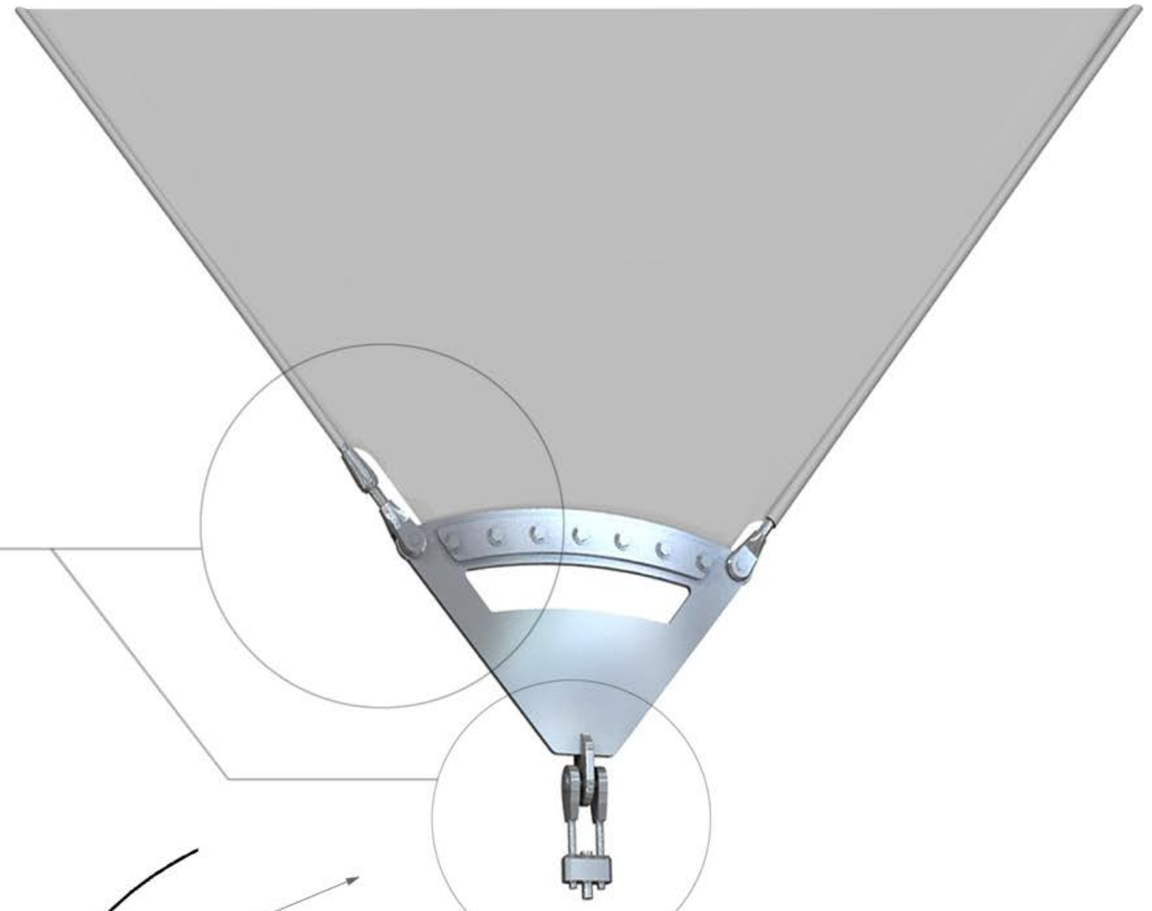


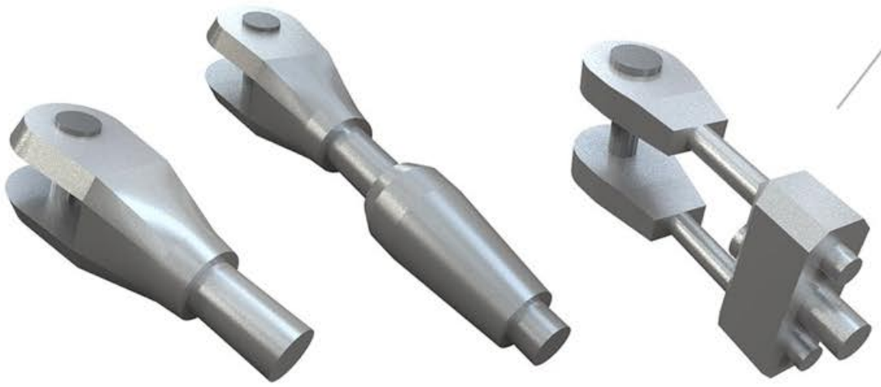
MEMBRANE DETAIL
PARAMETRICS

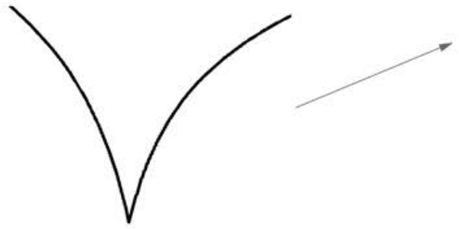
ALGORITMUS 03 - EGYSZERŰ SAROKCSOMÓPONT



Az MD_CORNER.ghp két görbe között hoz létre egy egyszerű sarokszerkezetet. A görbéknél nem követelmény, hogy síkgörbék legyenek (ezáltal kisebb szerkesztési pontosság van megkövetelve a felhasználótól, valamint a sarokszerkezetnél a lemezsík független tud maradni a kötél saroktól távoli alakjától). A sarokszerkezet egy fémlemez, mely két irányból fogad köteleket, és egy harmadik irányba lehorgonyzásként továbbítja az eredő húzóerőt. A két kötél közé kifeszített ponyvaanyag a lemezre van futtatva, és egy csavarokkal rögzített szorítólappal a lemezhez van rögzítve. Az itt amúgy használt feszítőkédertől ennél a modellnél el lett tekintve. A membránnak ennél a csomópontnál állítási lehetősége nincs (merev kapcsolat). Elfordulási lehetőség a köteleken kívül a lehorgonyzásnál kialakított (saroklemez síkjára merőleges) fémlemezcsuklónál van. A csomópont tovább kapcsolható egyéb szerkezetekhez.

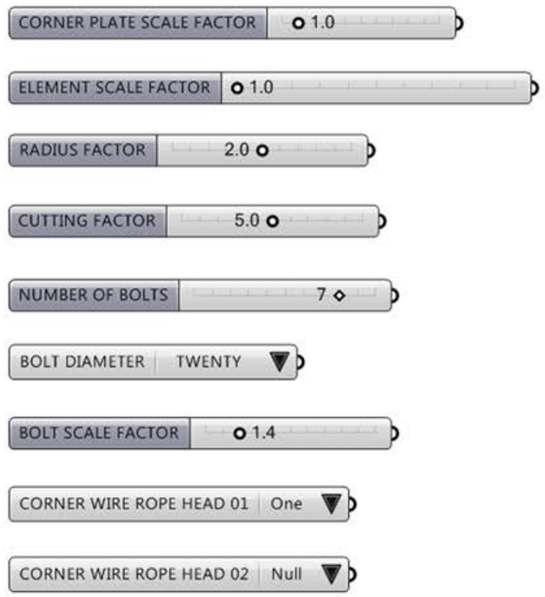






MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 03 - EGYSZERŰ SAROKCSOMÓPONT



általános sarokcsomóponti léptékszorzó

kötélelemek másodlagos léptékszorzója

fémlemez lekerekítéseinak tényezője

lemezkivágás tényezője

csavarszám

csavar átmérője

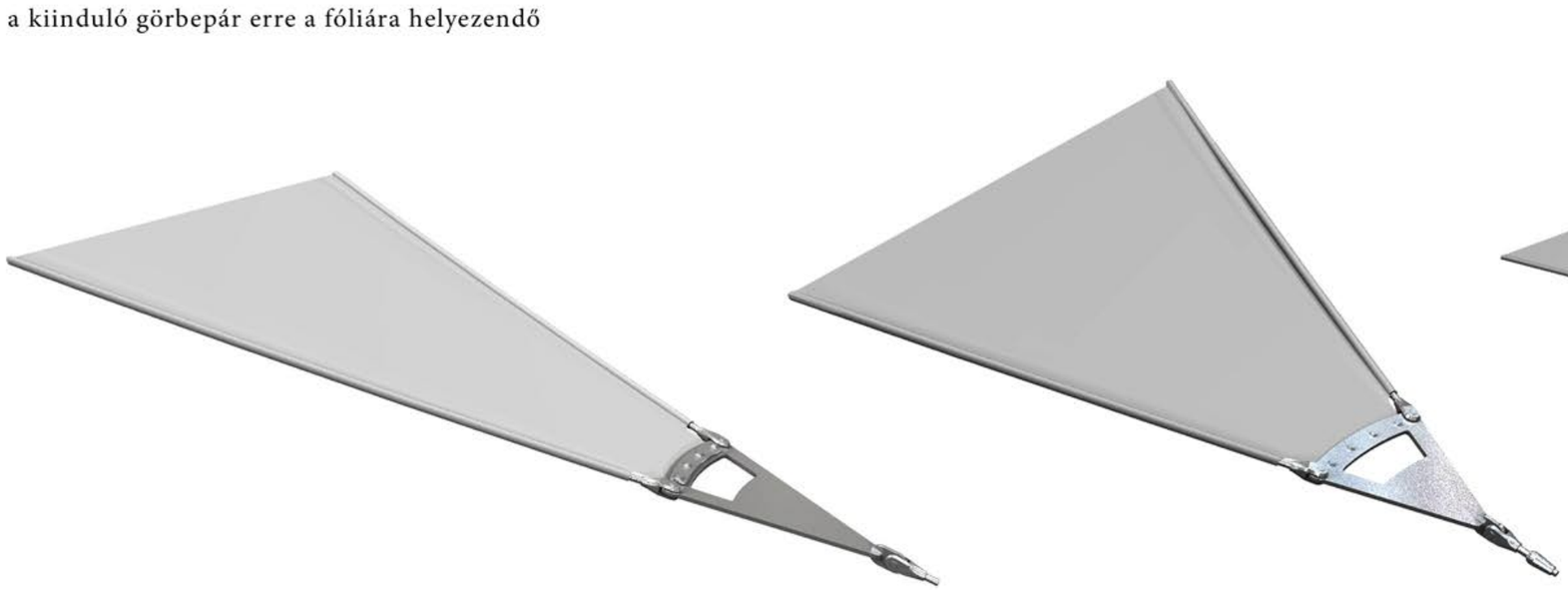
csavar másodlagos léptékszorzója

egyik kötél fejtípusa

másik kötél fejtípusa

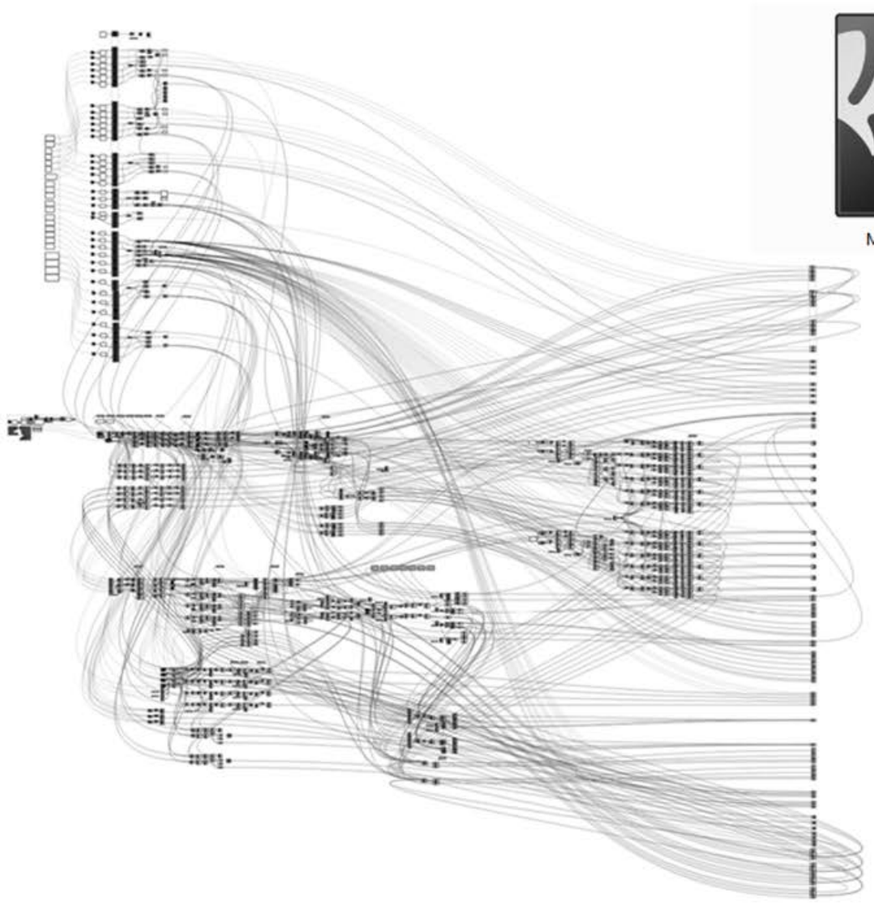
horgonykötél fejtípusa

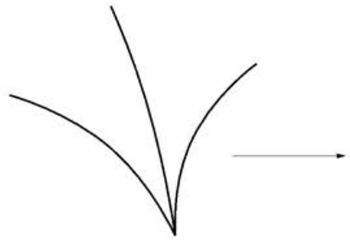
a kiinduló görbepár erre a fóliára helyezendő

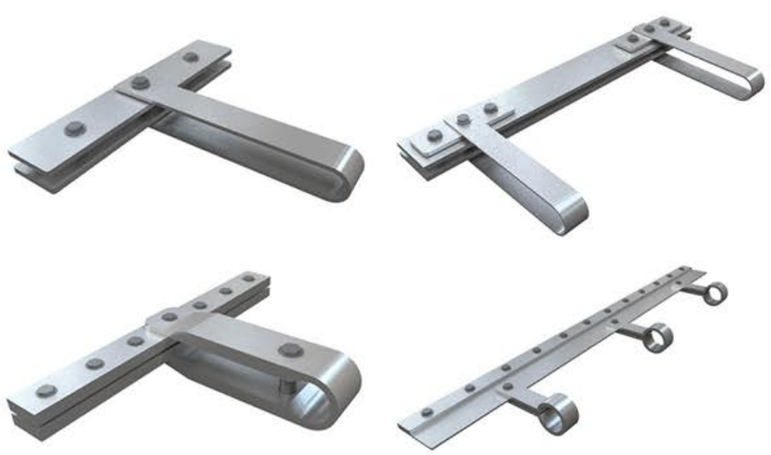


MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 04 - KOMPLEX RUGALMAS SAROKCSOMÓPONT







Az MD_CORNER_FLEXIBLE.ghp algoritmus egy három kötélgörbére megkomponált lágy peremű komplex sarokcsomópont kódja. A sarokba három kötél fut be, melyek indirekten fogadják a membránfelületben terjedő húzóerőt. Az erőátvitel úgy zajlik, hogy elsőként a membránt egy fém anyagú kéderen (másodlagos kötelen) borítjuk át, amely átborítás által a membrán be tud feszülni egy sor füles szorítólemez közé. Ezek a fülek közvetítik a húzóerőt a kötelekre. (A nagyfokú eltérő nyúlások, a közvetített erők nagymértékű eltérése és a súrlódásból fakadó problémák elkerülése végett gyakori ez a lépcsőzetes erőátvitel nagyobb konstrukcióknál.) A sarok csuklósan el tud fordulni az eredő erőt közvetítő horgonylemezek körül. A csomópontnál állítható paraméterként megjelenik a menüben szinte az összes olyan elem modellje, amely a korábbi tervlapokon fel van sorolva.

MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 04 - KOMPLEX RUGALMAS SAROKCSOMÓPONT

CORNER WIRE ROPE HEAD EDGE 01 | Two — egyes számú szélső sarokcsomóponti kötélfej típusa

CORNER WIRE ROPE HEAD EDGE 02 | Two — kettes számú szélső sarokcsomóponti kötélfej típusa

CORNER WIRE ROPE HEAD MIDDLE | One — középső sarokcsomóponti kötélfej típusa

ANCHOR JOINT PLATE | One — lehorgonyzó feszítőlapok száma

KEDER HEAD | Null — kéderkötél kötélfejének típusa

FLEXIBLE EDGE TYPE - EDGE | ONE — lágyperem-feszítőelem típusa egyes számú kötélen

FLEXIBLE EDGE TYPE - MIDDLE | TWO — lágyperem-feszítőelem típusa kettes számú kötélen

CORNER PLATE SCALE FACTOR | 3.9 — saroklemez generálléptékszorzója

ELEMENT SCALE FACTOR | 2.7 — saroklemez kötélelemeinek léptékszorzója

ANCHOR JOINT SCALE FACTOR | 3.0 — horgonylemez léptékszorzója

01 NUMBER OF CONNECTING WIRES | Two — egyes számú feszítőlap-összekötő ellenmenetes csavar száma

02 NUMBER OF CONNECTING WIRES | Two — kettes számú feszítőlap-összekötő ellenmenetes csavar száma

01 NUMBER OF BOLTS | 8 — egyes számú feszítőlap csavarszáma

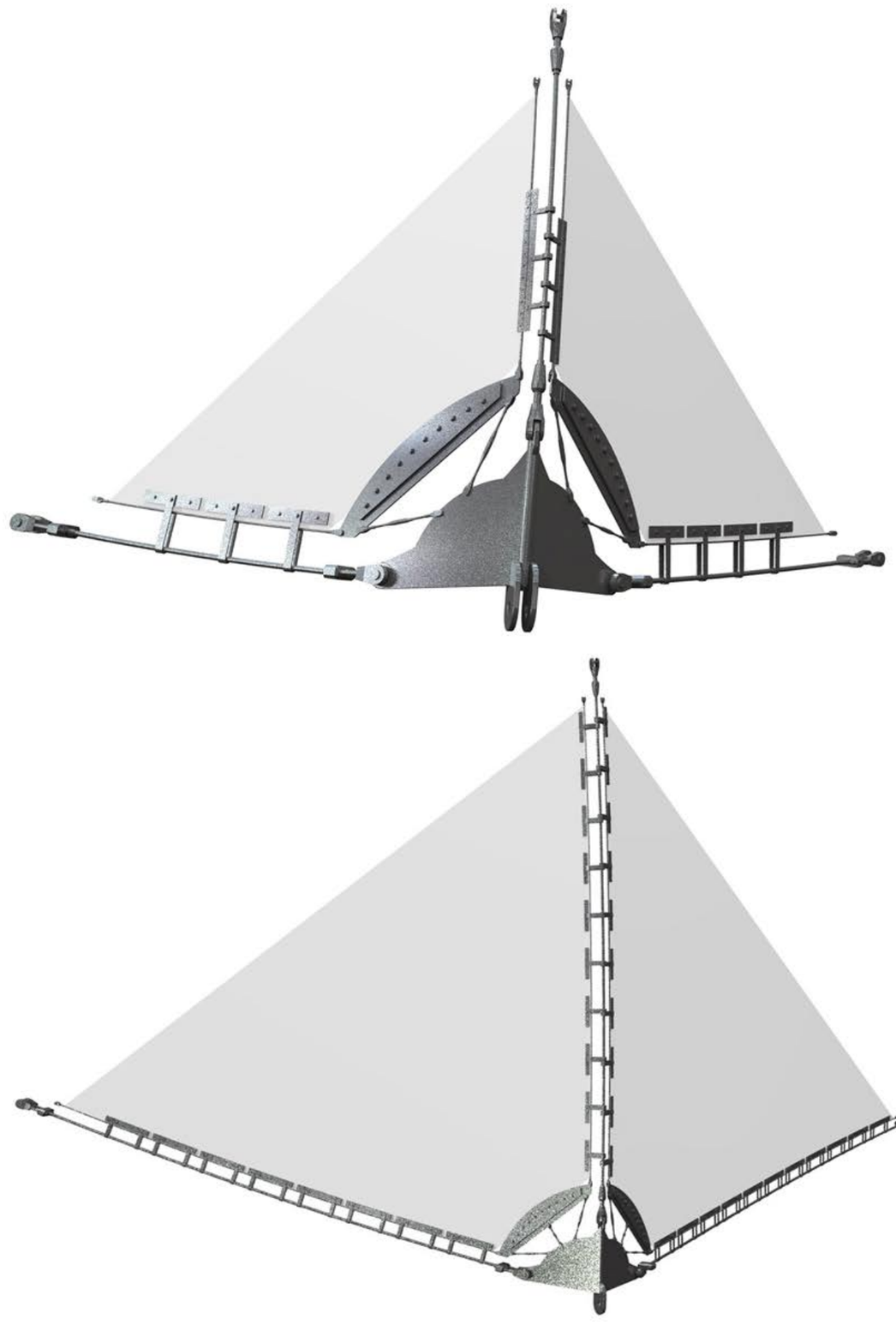
02 NUMBER OF BOLTS | 8 — kettes számú feszítőlap csavarszám

BOLT DIAMETER | FOURTEEN — csavarok átmérője

BOLT SCALE FACTOR | 3.0 — csavarok másodlagos léptékszorzója

FLEXIBLE EDGE FACTOR - EDGE | 0.7 — peremelem léptékszorzója - szélső kötelek

FLEXIBLE EDGE FACTOR - MIDDLE | 1.0 — peremelem léptékszorzója - középső kötél



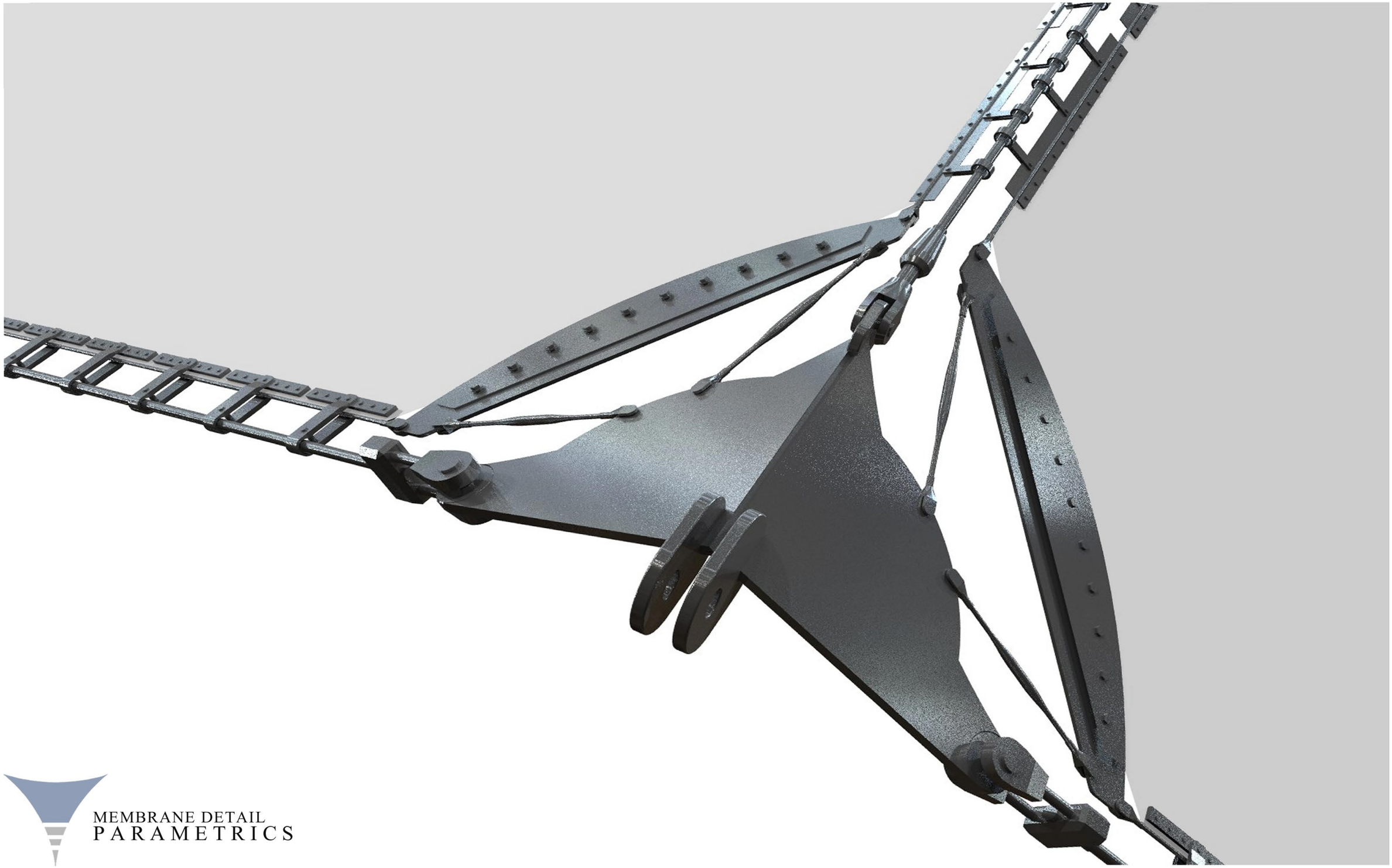
MD_CORNER_FLEXIBLE_EDGE

MD_CORNER_FLEXIBLE_MIDDLE

MEMBRANE DETAIL
PARAMETRICS

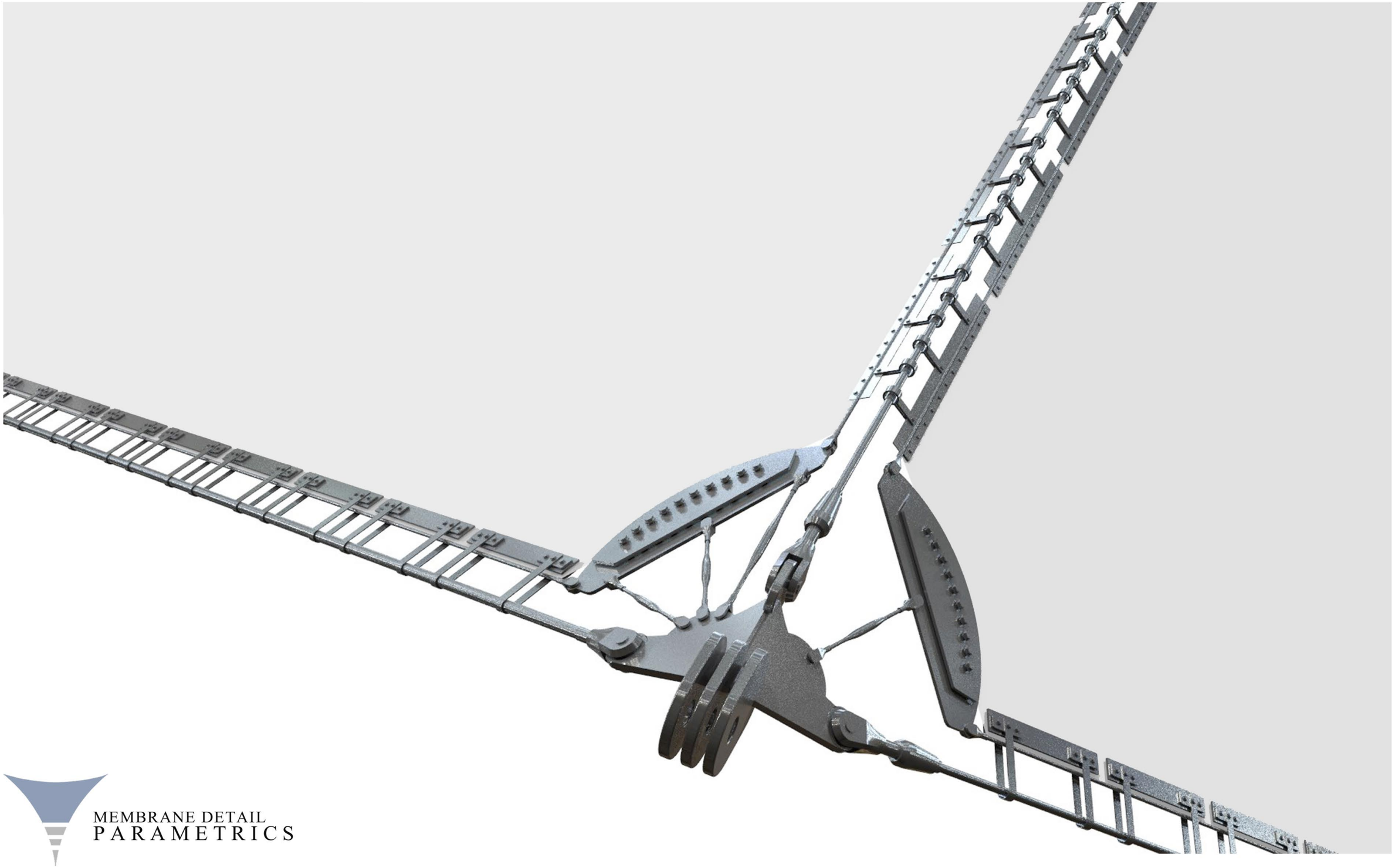
a kiinduló két szélső görbe és a köztes belső görbe ezekre a fóliákra helyezendő

ALGORITMUS 04 - KOMPLEX RUGALMAS SAROKCSOMÓPONT



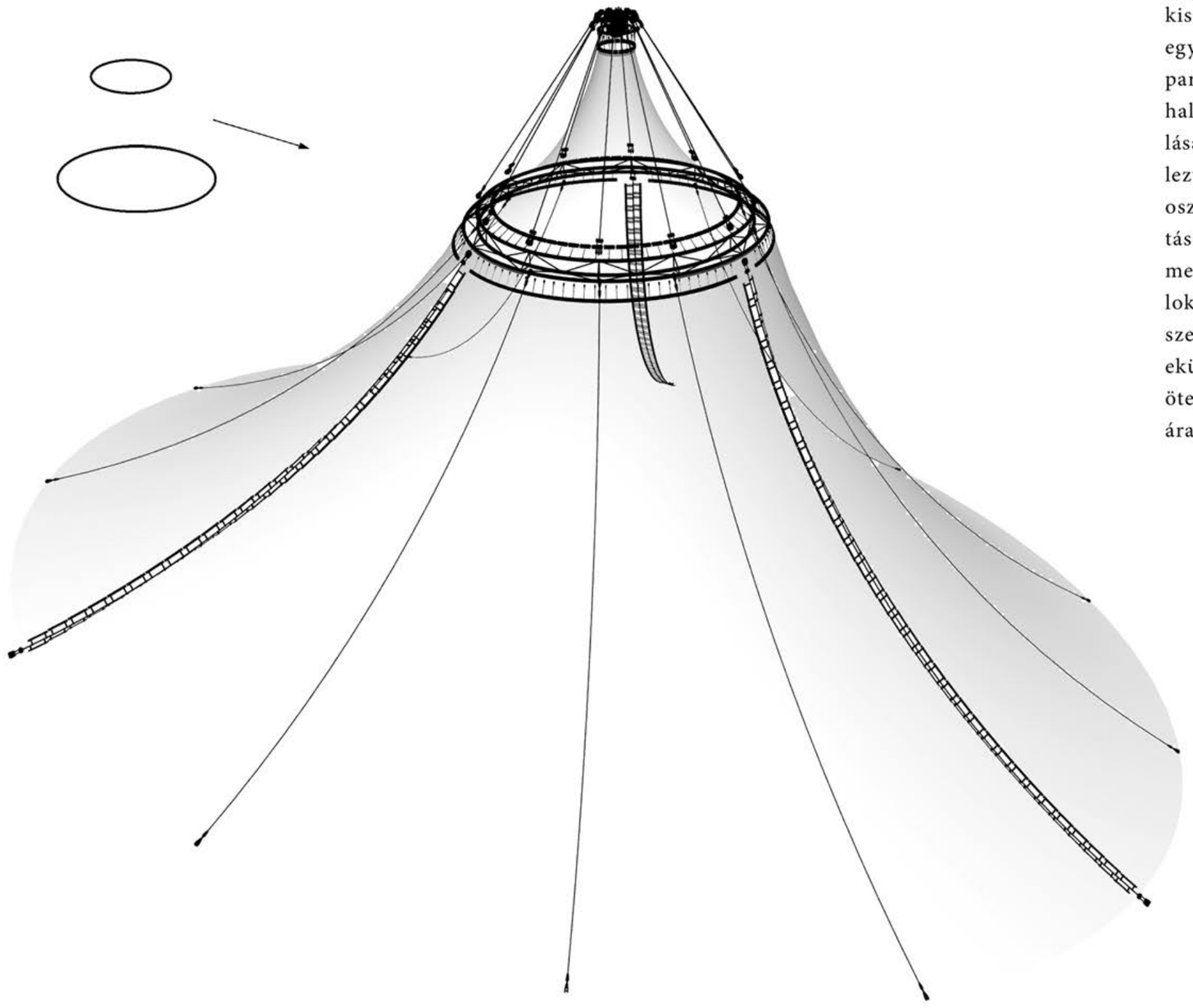
MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 04 - KOMPLEX RUGALMAS SAROKCSOMÓPONT

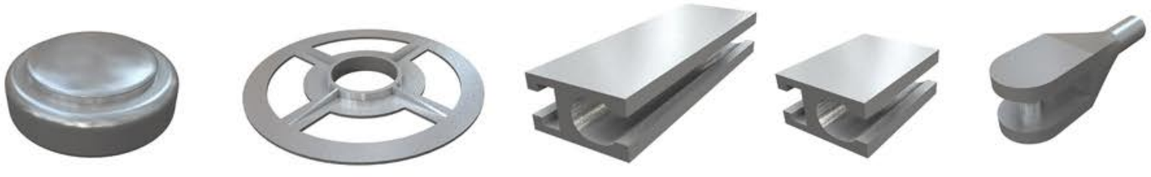


MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 05 - KOMPLEX ÁRBOCSZERKEZET



Az MD_MAST.ghp algoritmus két tetszőleges térbeli kör közé feszít ki egy membránt, valamint a felső görbe fölé a szerkezet lezárásaként egy kisebb ponyvát generál. A görbék közti összeszűkülés mértéke (rengeteg egyéb tényezővel együtt) állítható. Ez az algoritmus kísérleti jelleggel parametrizál egy komplett sátrat (nem elemként működik, de az itt felhalmozott tapasztalatok tovább vihetőek ilyen jellegű membránok kódolásához). A kifeszített membrán alá egy katasztrófakötél-sor van modellezve a sátorponyva tönkremenetelének esetére. Maga a membrán is felosztható (az osztás száma ennél az algoritmusnál 0-5-ig terjed), az osztásoknál rugalmas peremként belső kötélerősítések találhatóak. A nagy membrán egy rácsos tartóra rugalmasan van felfüggesztve (kéderprofilok feszítik fel a membránt), mely rácsos tartót a felső ponyva fölött összemetsződő kötélsor függeszt tovább az arboccsúcsra. Az algoritmus reekül kombinálható a 02-es algoritmussal, amellyel egy kihajlás ellen k ötelekkel biztosított oszlop modellezhető az árboccsúcs megtámasztására parametrikusan.

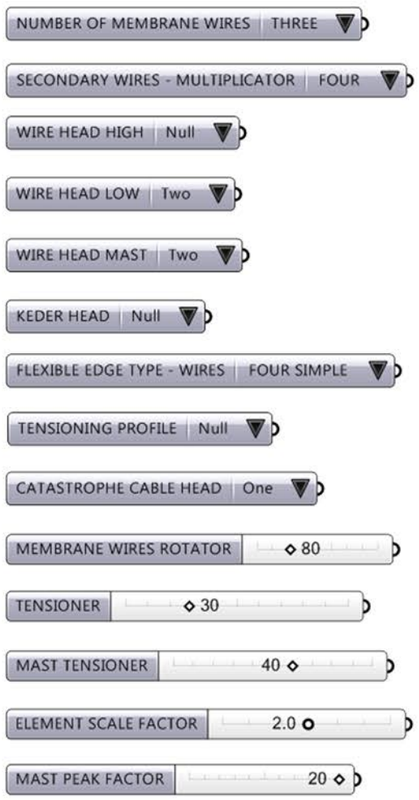


MD_MODELS

MD_MAST

MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 05 - KOMPLEX ÁRBOCSZERKEZET



belső kötélerősítések (membránszegmensek) száma

felső kötélsor mennyiségi szorzója a membránosztásokhoz

felső ponyva körüli kötelek felső fejtípusa

felső ponyva körüli kötelek alsó fejtípusa

alsó nagy membrán kötélerősítéseinek fejtípusa

kéderkötél kötélfejének típusa

lágyperem-feszítőelem típusa belső kötélerősítéseken

membrán függesztéséhez használt feszítőkéderprofil típusa

katasztrófakötelek fejtípusa

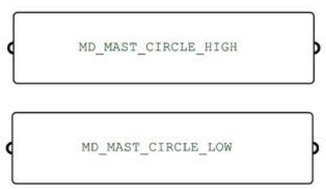
szögtényező a kötélszerkezetek forgatására a membrán főtengelye körül

nagy membránszerkezet feszítési (szűkítési) tényezője

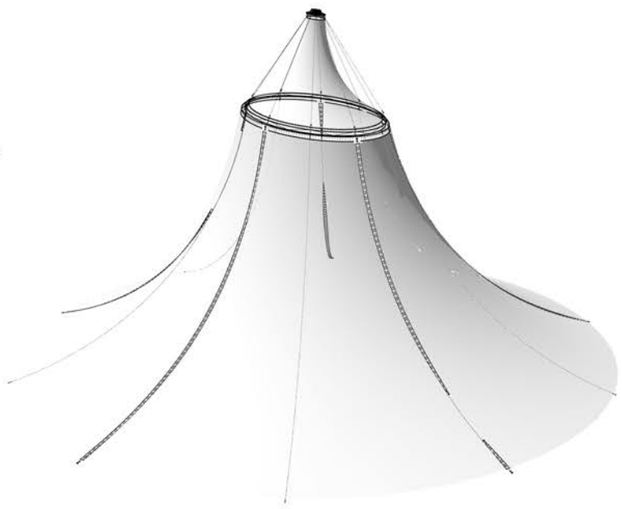
felső membránponyva feszítési (szűkítési) tényezője

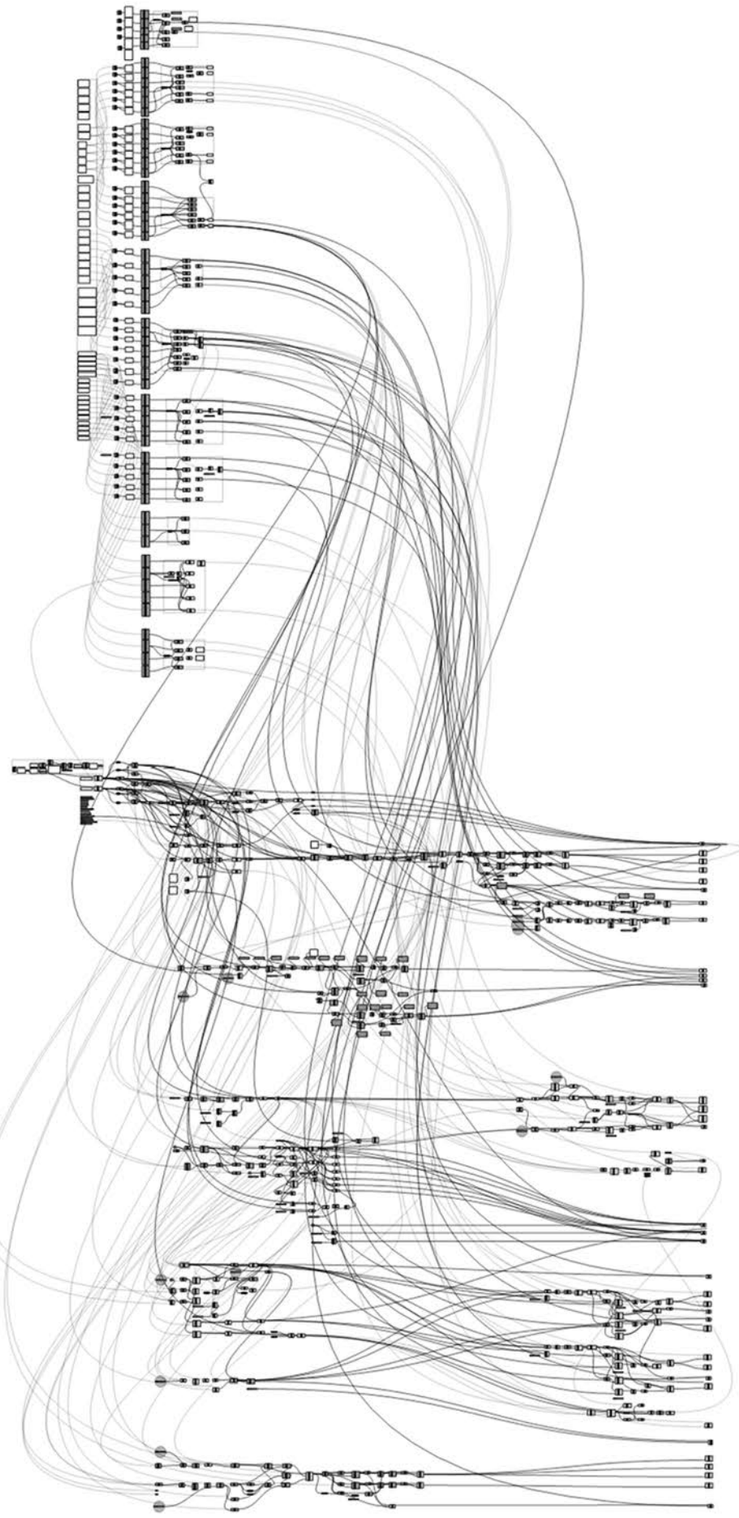
feszítőelemek generálléptékszorzója

felső ponyva kiemelkedésének magassági tényezője



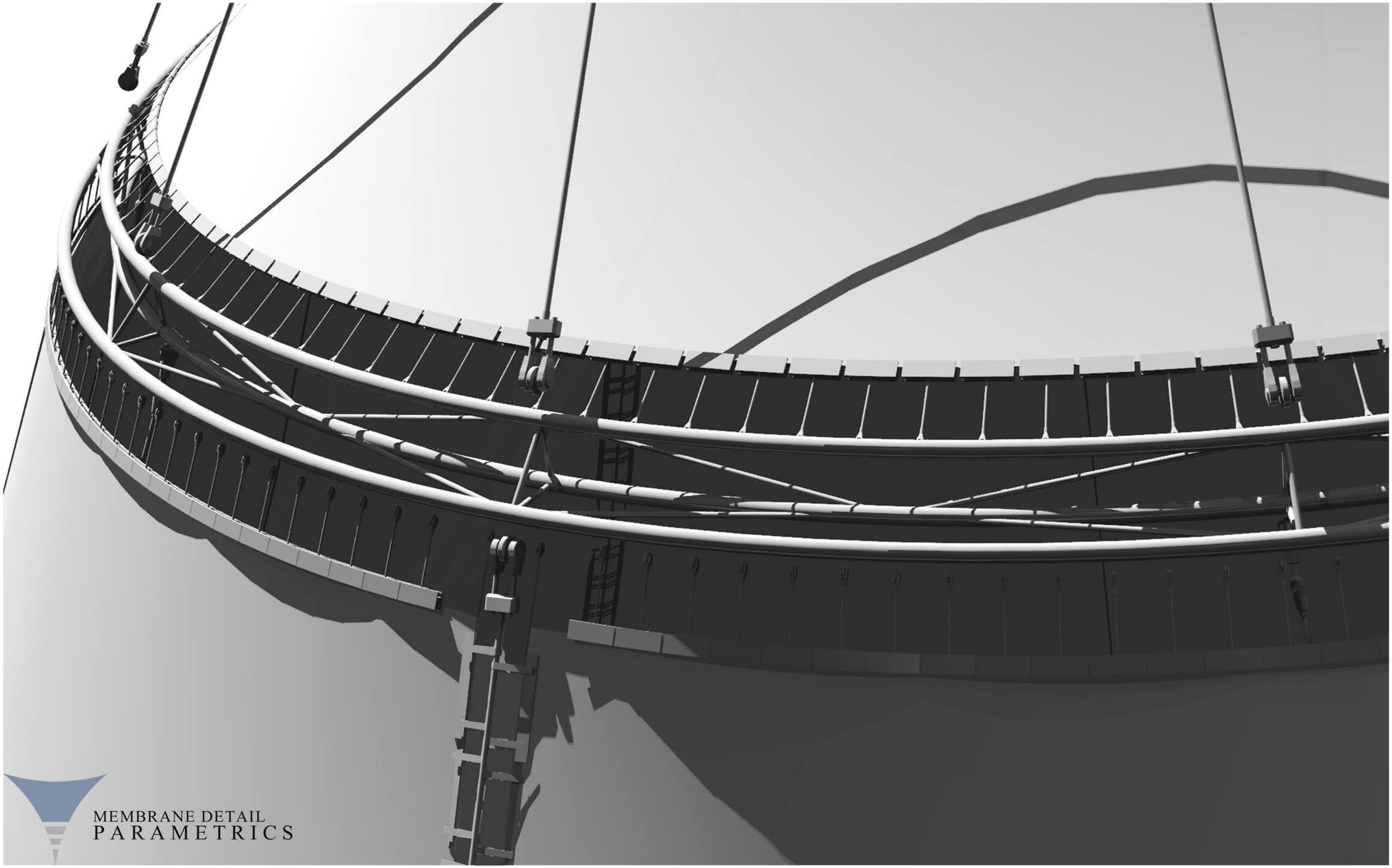
a kiinduló felső és alsó görbe
ezekre a fóliákra helyezendő





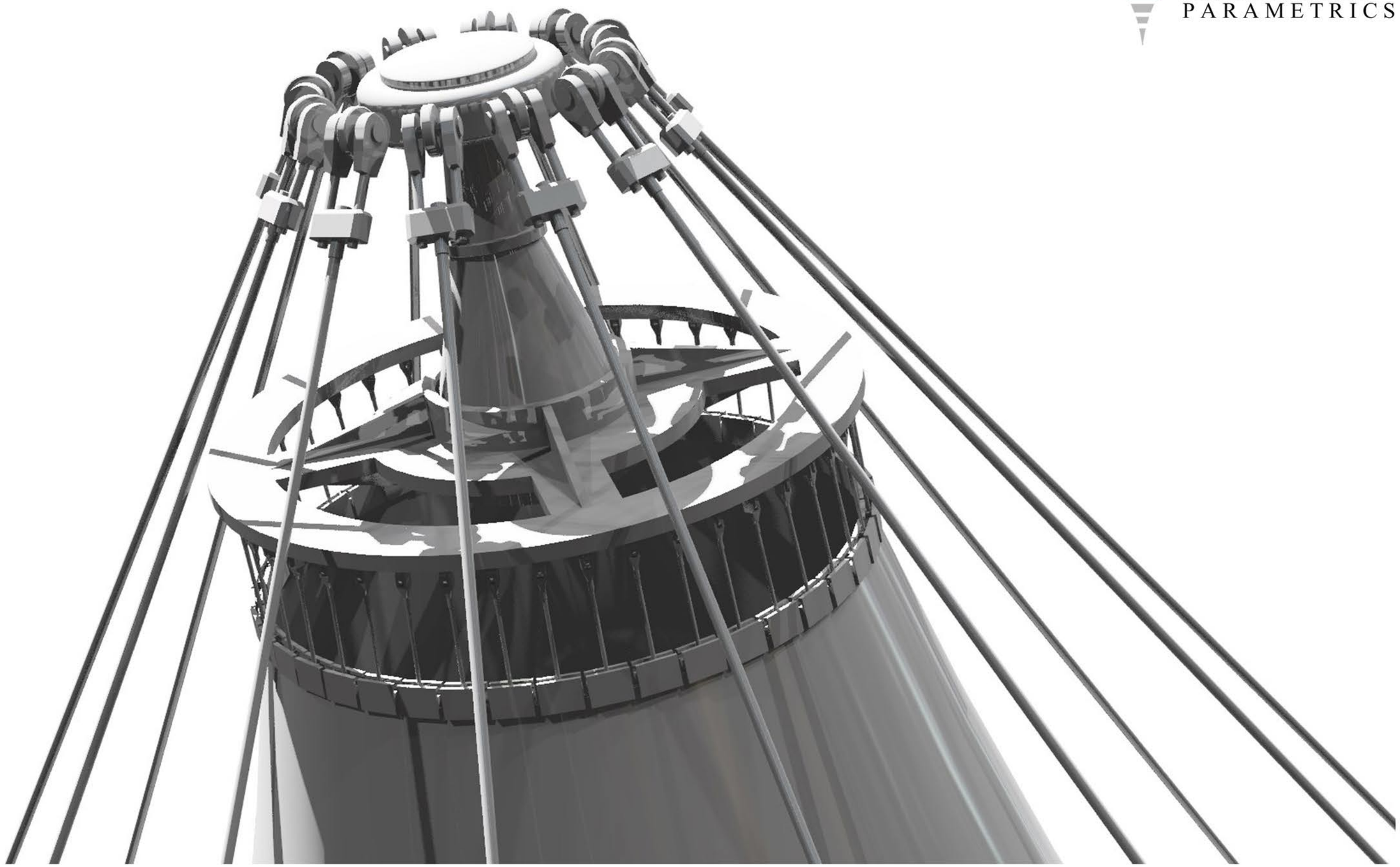
MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 05 - KOMPLEX ÁRBOCSZERKEZET



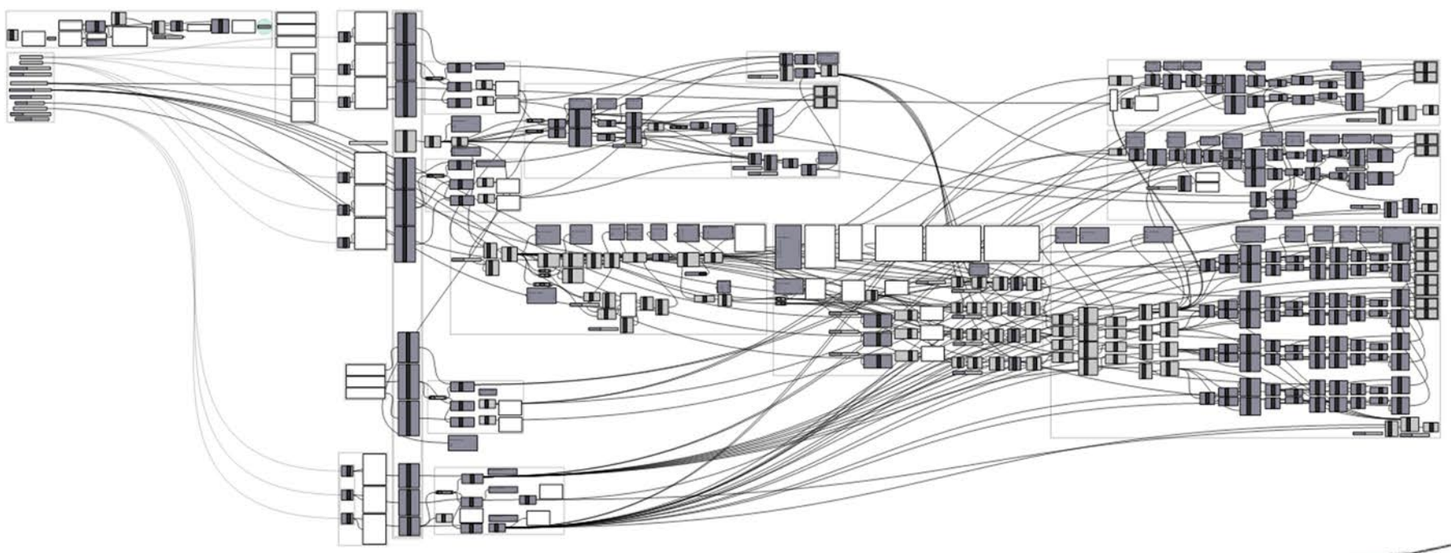
ALGORITMUS 05 - KOMPLEX ÁRBOCSZERKEZET

MEMBRANE DETAIL
PARAMETRICS



ALGORITMUS 06 - NYOMOTT RÚDSZERKEZET

Az MD_BEAM.ghp egyenes, nyomott rúdszerkezetet működtető algoritmus tetszőleges számú *egyenest* alakít át kihajlás ellen külső kötélszerkezetekkel megtámasztott rúdszerkezetté. A rudak végére kerülő támaszelemek a menüben felajánlott *sliderrel* forgathatóak a rúd tengelye körül külön-külön. Szintén a rúd főtengelye körül forgatható a rúd köré generáltatható kötélszerkezet is. Ennek a támaszrendszernek a különböző számszerű értékei és szerkezeti elemei mind állítható paraméterek a felkínált menüben (kötelek száma, nyomott támaszok száma, főtengelytől való távolság mértéke, stb.). Ha az egyszeres kötélszerkezetet választja a felhasználó, akkor alulfeszített gerendaként is használható az algoritmus.



MEMBRANE DETAIL
PARAMETRICS

ALGORITMUS 06 - NYOMOTT RÚDSZERKEZET

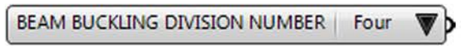
egyik rúdvégi támasztípus

másik rúdvégi támasztípus

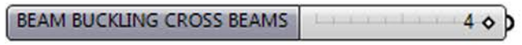
rúd vastagságának léptékszorzója

egyik rúdvégi támasz forgatótényezője

másik rúdvégi támasz forgatótényezője



kihajlás elleni keresztrudak száma



kihajlás elleni hosszkötelek száma

kötélszerkezet forgatótényezője

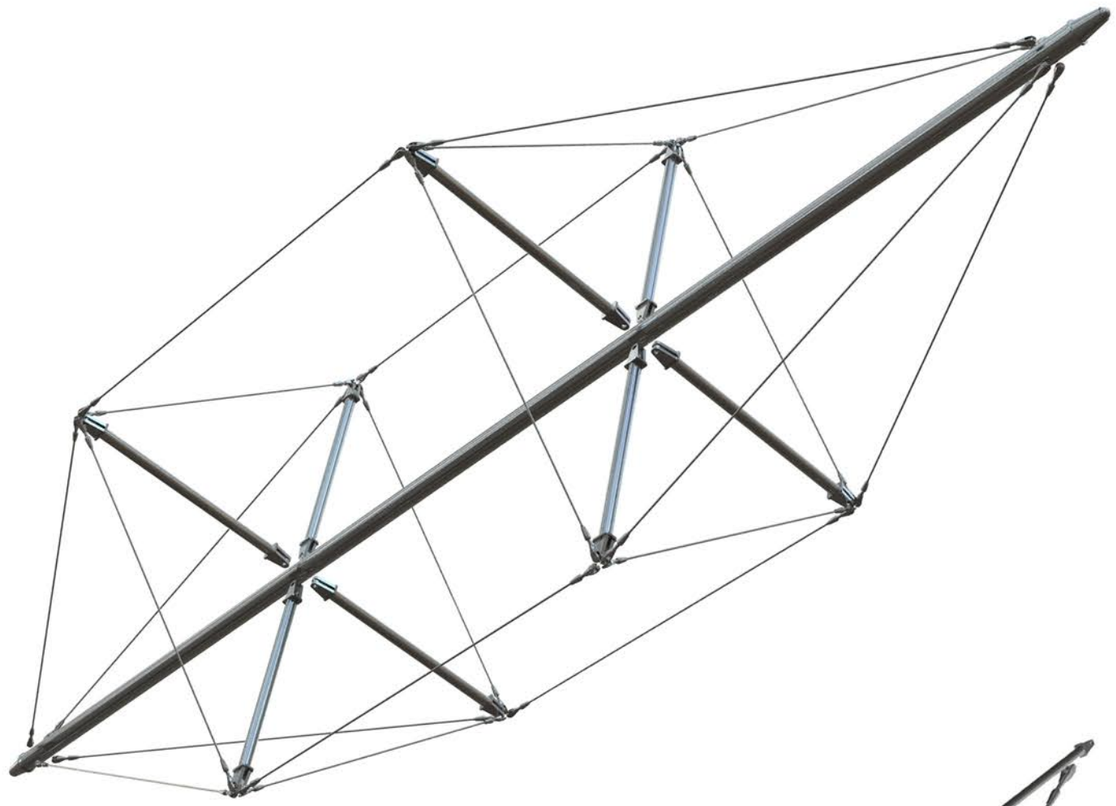
kötélcsomópontok főtengelytől való távolsága

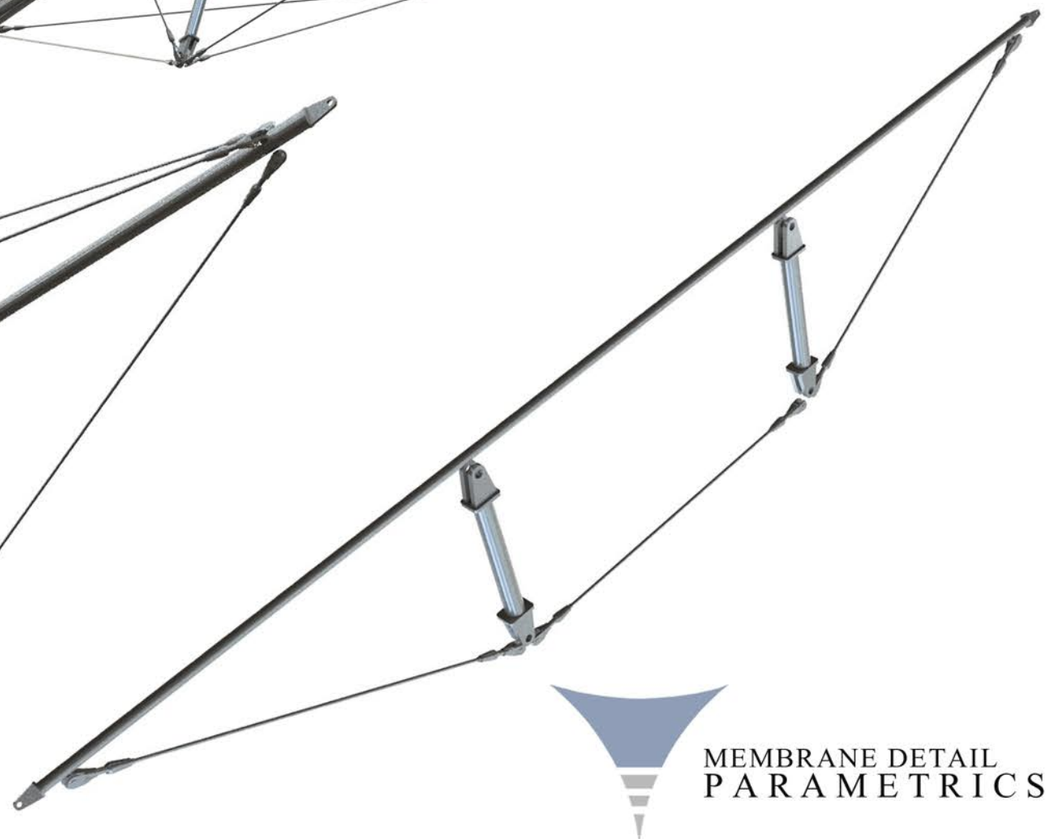
kötélszerkezetek fejtípusa

kötélfejek léptékszorzója

kötélfejek csomóponttól való távolsági tényezője

a kiinduló görbesor erre a fóliára helyezendő





MEMBRANE DETAIL
PARAMETRICS