Rácsos tartók megoldása erőmódszerrel, törzstartó nélkül

Szerző:
Kovács Krisztián
építészmérnök hallgató

Konzulens:
Dr. Sajtos István
egyetemi docens, tanszékvezető,
BME Építészmérnöki Kar, Szilárdságtani és Tartószerkezeti Tanszék

Budapesti Műszaki és Gazdaságtudományi Egyetem
Szilárdságtani és Tartószerkezeti Tanszék

TARTALOM

1. A DOLGOZAT CÉLJA

A dolgozat legfontosabb feladata, hogy a mostaninál szélesebb körben (és magyar nyelven) legyen ismert a Dale A. Hopkins, Gary R. Halford és Surya N. Patnaik által publikált[1] törzstartó nélküli erőmódszer, síkbeli rácsos tartókra alkalmazva.
Emellett még másodlagos cél volt az eljárás előnyeit két egyszerű programmal bemutatni. Korlátozott programozási ismereteim miatt természetesen egyáltalán nem tökéletesek, de remélem, hogy ötletet és lelkesedést adnak egy komolyabb szoftver létrehozására.

2. TÖRZSTARTÓ NÉLKÜLI ERŐMÓDSZER (INTEGRATED FORCE METHOD)

Jelen esetben nem a teljes módszert ismertetem, hanem annak csupán a rácsos tartókra vonatkozó formáját.

2.1. RÖVID ISMERTETÉS

Statikailag határozatlan szerkezeteknél (mint ahogy a nevük is mutatja) az egyensúlyi egyenletek direkt megoldása azért nem lehetséges, mert az ismeretlenek számánál kevesebb az egyensúlyi egyenletek száma. Az egyensúlyi egyenletek csak erő ismeretleneket tartalmaznak.
A törzstartó nélküli erőmódszer modellje esetén a szerkezet már nem merevtestekből épül fel, hanem lineárisan rugalmas (Hooke-törvény szerinti) rácsrudakból. Ennek előnye, hogy bővül a felírható egyenletek köre az erő-elmozdulás egyenletekkel és egyben nő az ismeretlenek száma is az elmozdulás ismeretlenekkel. Mivel ez az utóbbi egyenletrendszer pont olyan fokban „túlhatározott", mint amennyire az egyensúlyi egyenletrendszer „határozatlan", a két egyenletrendszer együtt már megoldhatóvá válik.
Ha csak a rúderőkre vagyunk kíváncsiak, akkor a második egyenletrendszerből kiküszöbölve az elmozdulás ismeretleneket, a szerkezet határozatlansági fokszámával megegyező egyenletrendszert fogunk kapni. Ezeket egyesítve az egyensúlyi egyenletrendszerrel meghatározhatók az erő ismeretlenek anélkül, hogy törzstartót kellett volna választani.
Természetesen ez fordítva is elvégezhető. Tehát ha az elmozdulásokat teikntjük elsődleges ismeretleneknek, akkor az erő ismeretlenek kiküszöbölése után jutunk az elmozdulás ismeretleneket megadó egyenletrendszerhez. (Ez a közismert elmozdulásmódszer.) A dolgozatban csak az erőmódszerrel foglalkozunk.

A törzstartó nélküli erőmódszer a következő lépéssorból áll:
- Hozzuk létre az egyensúlyi egyenletrendszert és mátrixát.
- Ebből hozzuk létre az alakváltozás - elmozdulás egyenletrendszert és mátrixát.
- Ebből eliminájuk (a geometriai peremfeltételek figyelembe vétele után) az elmozdulás ismeretleneket, ezáltal létrehozva az ún. peremfeltételi kompatibilitási egyenleteket.
- A Hooke-törvény segítségével az alakváltozásokat kifejezzük a rúderők függvényében. Ezt beírva a peremfeltételi kompatibilitási egyenletekbe megkapjuk az egyensúlyi egyenletrendszer megoldhatóságát biztosító többletegyenleteket.
- Ezzel kiegszítve az egyensúlyi egyenletrendszert meghatározzuk a rúderőket.

2.2. FELTEVÉSEK

A dolgozatban ismertetésre kerülő módszer esetében a következő feltevésekkel kell élnünk:
- A vizsgált szerkezet síkbeli rácsostartó.
- A szerkezet nem lehet statikailag túlhatározott. Várhatóan statikailag határozatlan.
- A szerkezetre csak a csomópontokban és csak a szerkezet síkjában hatnak terhek vagy

1 Dale A. Hopkins, Gary R. Halford és Surya N. Patnaik: Integrated Force Method Solution to Indeterminate Structural Mechanics Problems (2004)

támaszerők. Hajlított elem nem fordul elő.
- A rudaknak csupán „elméleti” keresztmetszetük van, azaz a modellben kétdimenziós elemként kezeljük őket.
- A rudak csak a csomópontokban találkoznak, látszólagos metszésük esetén „kitérő” helyzetben vannak.
- A csomópontokon belül a rudak tengelyei és a terhek hatásvonalai egy pontban metszik egymást. Nem keletkezik nyomaték.
- Érvényes a Bernoulli - Navier hipotézis.
- A rudak lineárisan rugalmasak. (A Hooke - törvény szerint.)
- A rudak egyformán viselkednek húzásra és nyomásra. Nem mennek tönkre semmilyen elsőrendű (például szakítószilárdság kimerülése) vagy magasabb rendű (például kihajlás) terhelésre, illetve ezek hatásai sem jelentkeznek.
- Csak kis elmozdulások jönnek létre.

2.3. AZ EGYENSÚLYI EGYENLETRENDSZER

Az egyensúlyi egyenleteket a rácsos tartó csomópontjaira felírt vetületi egyenletek. A gátolt elmozdulású, megtámasztott csomópontok esetén a gátolt elmozdulás irányában ezt nem teszszük meg, hiszen csak a támaszerőt adja meg.

A rendezett egyenletrendszer együtthatóiból hozzuk létre az egyensúlyi egyenletrendszer **B** mátrixát. (A továbbiakban **B** mátrixon ezt értem.)

2.4. ALAKVÁLTOZÁS - ELMOZDULÁS EGYENLETRENDSZER

Az alakváltozás - elmozdulás egyenleteket a rácsrudakra írjuk fel a végpontjaik, azaz a csomópontok elmozdulásának függvényében. Arra a rúdra nem írunk fel egyenletet, melynek mindkét vége mindkét irányban rögzített, hiszen ezáltal nem képes megnyúlni.
A rendezett egyenletrendszer együtthatóiból lehet létrehozni az alakváltozás – elmozdulás egyenletrendszer mátrixát. Szerencsére ezzel azonos eredményt kapunk, ha az egyensúlyi egyenletek mátrixának transzponáltját vesszük. (Ezzel a számítás folyamata is nagyban egyszerűsödik és rövidül.)
(A továbbiakban ezt $\mathbf{B}^T$ mátrixként jelölöm.)

2.5. A PEREMFELTÉTELI KOMPATIBILITÁSI EGYENLETEK

Az alakváltozás – elmozdulás egyenletrendszerből (az elmozdulásokra vonatkozó peremfeltételek figyelembevétele után) kiejtjük az elmozdulás ismeretleneket, amelyek a megmaradó elmozdulás ismeretlenekk kifejezhetők. Ezáltal egy kisebb egyenletrendszert kapunk, ami az alakváltozás ismeretlenek közötti kapcsolatot adja meg.
Ezeket (a csak alakváltozásokat tartalmazó) egyenleteket nevezzük peremfeltételi kompatibilitási egyenleteknek. (Továbbiakban **C** mátrix jelöli a peremfeltételi kompatibilitási egyenletek együttható mátrixát.)
Természetesen ezt bármilyen egyenletrendezési módszerrel el lehet érni, de a programban is alkalmazható algoritmus, hogy kifejezzük az egyik elmozdulást, behelyettesítjük a többi egyenletbe, majd ezt addig ismételjük, míg nem marad elmozdulás ismeretlen a megmaradó egyenletekben. (Felhasználható még esetleg a $\mathbf{C}^*\mathbf{B}^T=\mathbf{0}$ összefüggés[2] is.)

---

2 Dale A. Hopkins, Gary R. Halford és Surya N. Patnaik: Integrated Force Method Solution to Indeterminate Structural Mechanics Problems (2004) 13. oldalán

2.6. ERŐ - ALAKVÁLTOZÁS ÖSSZEFÜGGÉS

A Hooke-törvény szerint a rúderők és a rudak megnyúlása egyenesen arányos egymással.

$$\Delta l = \frac{l}{E \times A} \times F$$

Azaz a megnyúlás egyenlő a rúd eredeti hosszának és a rúderő szorzatának, valamint a rugalmassági modulus és a rúdkeresztmetszet szorzatának hányadosával.
Ezt felhasználva a peremfeltételi kompatibilitási egyenleteket át tudjuk alakítani oly módon, hogy csupán erő ismeretlenek legyenek bennük.

2.7. MEGOLDÁS

Az előbbi lépésben átalakított peremfeltételi kompatibilitási egyenleteket az egyensúlyi egyenletrendszerhez fűzve egy határozott egyenletrendszert kapunk eredményként. Ezt megoldva megkapható az összes rúderő. (Amely rúdnak mindkét vége mindkét irányban rögzítve volt, abban nem működik erő. Ez könnyen bizonyítható a Hooke-törvénnyel, ha Δl=0-ra keressük a hozzá tartozó erőt[3].)

2.8. ÖSSZEFOGLALÁS

Amint látható, az egész módszer igen könnyen algoritmizálható, ami a számítógépes számításoknak kedvez. További előnye, hogy nem kell törzstartót kiválasztani, ami különösen a magas határozatlansági fokszámú szerkezeteknél előnyös. Ezzel egyszerre ejtünk ki egy komoly hibalehetőséget (így nem tudunk rossz törzstartót választani) és könnyítjük meg a számítógépes programozást (nem kell megtanítani a szoftvernek, hogy mitől jó egy törzstartó).
Ugyanakkor komoly hátránya, hogy igen nagyméretű, nem szimmetrikus egyenletrendszerrel kell dolgozni, amiben könnyű számítási hibát véteni. Ezt érdemes számítógépes mátrix-kezelő programmal áthidalni.

---

3 Ez nem igaz abban az esetben, ha hőterhelésnek tesszük ki a szerkezetet, de ettől most tekintsünk el.

3. PROGRAMOK

A törzstartó nélküli erőmódszer demonstrálására Matlab programnyelven írtam két programot. Az egyszerűbb lényegében a 2.1. pontban ismertetett eljáráson alapul, annak viszonylag alacsony számítási kapacitását és könnyen követhető algoritmusát kihasználva. Célja a rúderők minél gyorsabb kiszámítása.
A bonyolultabb program ezzel szemben az eredeti módszernek egy módosított változatát alkalmazza, de hasonló logikán alapul. A számítási idő itt másodlagos volt, fő célja a minél általánosabb, minél több - a szerkezetet vagy a viselkedését jellemző - adat kezelése és összekapcsolása.
Mindkét esetben a mellékletben található maga a kódsor, valamint a használatot illusztráló egy-egy példa.

3.1. A GYORSABB PROGRAM

Ez a program a rövid számítási időre van „kihegyezve”, így viszonylag egyszerűbb is.
A könnyebb kezelhetőség érdekében több futási szakaszra van bontva:
- topológia megadása,
- geometria, támaszok, terhek megadása és
- megoldás kiszámítása.

3.1.1. TOPOLÓGIA MEGADÁSA

Itt lényegben egyetlen 2 oszlopos mátrixot kell megadni bemenő adatként. Minden eleme egy-egy csomópont sorszáma, minden sora egy-egy rúd (amit meghatároz a két végén lévő csomópontok sorszáma).

3.1.2. GEOMETRIA, TÁMASZOK, TERHEK MEGADÁSA

Ha megtartjuk, hogy 1-től indulva számozunk, meghatározható a csomópontok száma (legnagyobb sorszám) és a rudak száma (sorok száma). Ez alapján összeállítható két (az adott szerkezetre vonatkozó) táblázat, mely egyszerre könnyíti meg a további adatok bevitelét és az adatok tárolását. (Egyet a csomópontok adatainak és egyet a rudak adatainak.)
(A kezelhetőség kedvéért érdemes egy fejlécet is beilleszteni, hogy ne kavarodjunk bele az oszlopokba.)
A terheket x és y vetületre bontva lehet megadni, míg a támaszoknál értelemszerűen gátolni kell az adott csomópont mozgását a kívánt irányban.
Természetesen az adatokat külső táblázatból is lehet importálni.

3.1.3. MEGOLDÁS

Ez a program lényegi, egyúttal a leghosszabb futási idővel bíró része. A következő főbb lépésekkel jellemezhető:
- egyensúlyi mátrix létrehozása,
- peremfeltételi kompatibilitási mátrix létrehozása,
- alakváltozások kifejezése a rúderők függvényében (Hooke-törvény),
- terhek oszlopvektorrá alakítása és végül
- az egyenletrendszer megoldása.

3.1.3.1. Egyensúlyi mátrix

Viszonylag egyszerű algoritmussal előállítható. Lényegében minden csomópontra (x és y irányban) fel kell írni a vetületi egyenleteket. (Kivéve a gátolt elmozdulások irányában.) Mivel nem

szükséges a teljes egyenletrendszert felírni, csupán az egyenletrendszer együttható mátrixát, így mindössze a csomópontokhoz kapcsolódó rudak vetületi iránnyal bezárt szögének koszinuszait kell kiszámolni, ami a vetületi és a valós rúdhossz hányadosa. (A csomópontra nem ható rúderők együtthatója nulla.)

3.1.3.2. Peremfeltételi kompatibilitási mátrix

A számítási idő lerövidítése érdekben a peremfeltételi kompatibilitási mátrixot az egyensúlyi mátrixból számítjuk ki. Ehhez a $\mathbf{C}^*\mathbf{B}^T=\mathbf{0}$ összefüggésből (is) adódó $\mathbf{B}^*\mathbf{C}^T=\mathbf{0}$ összefüggést érdemes alkalmazni.
Egyes programozási környezetekben (például az általam is alkalmazott Matlab szoftvernél) ehhez pár előkészítő lépést kell tenni. Először ki kell számolni a határozatlanság fokszámát ($\mathbf{C}$ mátrix sorainak száma). Mivel annyi egyenlet kell, ahány rúd(erő) van, így egyszerűen kivonjuk a rudak számából az egyensúlyi mátrix sorainak számát. Ennek segítségével definiálni tudjuk $\mathbf{C}^T$ mátrixot. Végül pedig fel kell töltenünk a megnyúlás nélküli - és még egy - rudak helyét egyesekkel[4].
Ekkor még mindig fennáll, hogy a fennmaradó értékek a $\mathbf{C}$ mátrixban bármilyen *k* konstanssal megszorozva teljesen megfelelnek. ($k^*\mathbf{B}^*\mathbf{C}^T=\mathbf{0}$ bármely *k*-ra igaz, ha $\mathbf{B}^*\mathbf{C}^T=\mathbf{0}$.) Ám amennyiben az egyik rúderőhöz tartozó értékeket manuálisan (például 1-nek) megadjuk, ezzel az összes többi értéket kényszerítjük, hogy igazodjon hozzá. Ezáltal már csak egyetlen megoldás létezik, amit a Matlab már egyértelműen képes meghatározni.

Amint az esetleg szükséges előkészületekkel végeztünk, oldassuk meg a programmal a $\mathbf{B}^*\mathbf{C}^T=\mathbf{0}$ összefüggést. (Ezen a ponton lehet tovább rövidíteni a futási időt, ha egy erre a problémára optimalizált megoldó szoftvert vagy kódsort alkalmazunk.) Végül vegyük a $\mathbf{C}^T$ mátrix transzponáltját.

3.1.3.3. A Hooke-törvény

Mivel a kompatibilitási mátrix együtthatói még a megnyúlásokra vonatkoznak, így mindet meg kell szorozni a hozzá tartozó rúd hosszával és el kell osztani a keresztmetszetével és a rugalmassági modulusával.

$$k \times \Delta l = k \times \frac{l}{E \times A} \times F$$

Azaz ahhoz, hogy az alakváltozáshoz tartozó *k* együttható helyett a rúderőkhöz tartozó együtthatókat kapjunk, *k* értékét meg kell szoroznunk a vonatkozó rúd hosszával és el kell osztanunk a rúd rugalmassági modulusával és keresztmetszetvel, azaz a húzási merevségével.
Erre megfelelő eljárás, hogy egy alkalmas méretű (rudak száma x rudak száma) kvadratikus zérus mátrix főátlóját feltöltjük a rudakhoz tartozó hossz és a rúd húzási merevségének hányadosával. Ezzel a $\mathbf{C}$ mátrixot megszorozva a megoldáshoz szükséges plusz egyenletekhez jutunk.

3.1.3.4. Terhek oszlopvektorrá alakítása

Ennél a lépésnél a következőkre kell figyelni:
- A gátolt elmozdulású irányokban a „terheket” a támaszerők jelentik, emiatt ez a vetületi egyenlet eleve bele sem került az egyensúlyi egyenletrendszerbe. Tehát a hozzá tartozó terhet

4 Ennek oka, hogy lehetnek olyan rudak, melyeknek mindkét vége rögzített mindkét irányban, így nem tudnak megnyúlni, ezáltal nem képesek erőt felvenni. Továbbá az egyensúlyi mátrixban a hozzájuk tartozó rúderők együtthatója mindig nulla, emiatt a $\mathbf{B}^*\mathbf{C}^T=\mathbf{0}$ műveletben a Matlab nem tudja megfelelően kezelni őket. Mivel tudjuk, hogy végül a rúderő értéke úgyis zérus lesz, így nyugodt szívvel helyezhetünk a kompatibilitási egyenletekben nem nulla tényezőt eléjük. (És a végeredményként ezt a zérus értéket meg is kapjuk.)

is ki kell hagyni.
- A terheket abban a sorrendben kell elhelyezni a vektorban, amilyenben a hozzájuk tartozó vetületi egyenletek vannak.
- A peremfeltételi kompatibilitási egyenletekhez nem tartoznak terhek, tehát a tehervektor utolsó sorait (amennyi sora **C** mátrixnak van) nullával kell feltölteni.

(Az első két pont miatt érdemes az egyensúlyi egyenlet előállításával párhuzamosan elkészíteni a tehervektort.)

3.1.3.5. Megoldás

Az egyensúlyi egyenletrendszerből és a Hooke-törvény segítségével átalakított peremfeltételi kompatibilitási egyenletekből (pontosabban ezek mátrixaiból és a tehervektorból) hozzuk létre a végleges egyenletrendszert, majd ezt megoldva megkapjuk a keresett rúderőket.

3.2. AZ ÁLTALÁNOSABB PROGRAM

Az előző fejezetben ismertetett módszer kifejezetten a rúderők meghatározására van optimalizálva. Ám hasonló logika mentén eljuthatunk egy olyan eljárásig, ami sokkal több, a szerkezetet és erőjátékát jellemző paraméter kezelésére alkalmas.
Lényege, hogy nem küszöbölünk ki ismeretlent. Tehát az összes csomóponra felírjuk a vetületi egyenleteket (így a reakcióerők is az ismeretlenek közé kerülnek), valamint nem hozzuk létre a peremfeltételi kompatibilitási egyenletrendszert, hanem az erő-elmozdulás egyenletrendszerrel helyettesítjük. Emellett a támaszok típusa (görgős vagy csuklós) és szöge állítható, valamint a terheket sem vetületükkel, hanem nagyságukkal és szögükkel lehet megadni.
Így egy olyan összefüggés-rendszert kapunk, amely kifejezi a szerkezet geometriája, megtámasztása, alakváltozása, terhelése, valamint a szerkezetet alkotó rudak belső ereje, keresztmetszete és anyaga (rugalmassági modulusa) közötti kapcsolatot.
Ha mindezt paraméteresen vezetjük le, akkor egy adott szerkezet-típusra alkalmazható általános megoldó-egyenletrendszert kapunk. Ezt egyes, számunkra fontos ismeretlenekre rendezve annak értékét a többi ismeretlen zárt alakú függvényként meghatározza. Ezáltal alkalmassá válik optimalizálásra és hibavizsgálatra is. (Ez utóbbi alatt az építési pontatlanságokból származó geometriai hibákat értem.)
Hátránya ugyanakkor, hogy a több ismeretlen és egyenlet miatt sokkal nagyobb méretű mátrixokkal kell dolgoznunk, így számításigényesebb és lassabb, mint az előző program.

A könnyebb kezelhetőség érdekében itt is több futási szakaszra van bontva a program:
- topológia megadása,
- geometria, támaszok, terhek megadása és
- megoldás kiszámítása.

3.2.1. TOPOLÓGIA MEGADÁSA

Itt az előző programban leírt 2 oszlopos mátrix mellett egy (oszlop)vektort is meg kell adni, melyben felsoroljuk azon csomópontokat, melyeket meg akarunk támasztani.

3.2.2. GEOMETRIA, TÁMASZOK, TERHEK MEGADÁSA

A már említett módon lehet a csomópontok és a rudak táblázatát előállítani. Emellett itt az összes felsorolt támasznak is kell egy külön táblázat (illetve abban egy sor), amelyben többek között azt is be lehet állítani, hogy görgős vagy fix csuklós megtámasztást kívánunk alkalmazni. Fontos különbség az eddigiekkel ellentétben, hogy nem kell a teljes táblázatot kitölteni, az

alapértelmezetten paraméterekkel van feltöltve, melyek akár benn is hagyhatók. Természetesen az adatokat itt is lehet külső táblázatból importálni.

3.2.3. MEGOLDÁS

Ennél a programnál lényegesen hosszabb futási idővel kell számolni, mint az előzőnél. (A konkrét értéket nagyban befolyásolja, hogy hány paramétert hagyunk a rendszerben.) A következő főbb lépésekkel jellemezhető:
- egyensúlyi mátrix létrehozása,
- alakváltozás - elmozdulás mátrix létrehozása,
- alakváltozások kifejezése a rúderők függvényében (Hooke-törvény),
- terhek oszlopvektorrá alakítása és végül
- az egyenletrendszer megoldása.

3.2.3.1. Egyensúlyi mátrix

Hasonlóan állítható elő, mint a korábbi esetben. Lényegében minden csomópontra (x és y irányban) fel kell írni a (kapcsolódó) rúderők vetületi egyenletét. (A megtámasztott irányokban is.) Jelen esetben nem csupán a rúderők, hanem a támaszerők együtthatói is szükségesek. Tehát a kapcsolódó rudak és támaszok vetületi iránnyal bezárt szögeinek koszinuszait kell kiszámolni, ami a rudak esetében a vetületi és a valós rúdhossz hányadosa. (A csomópontra nem ható rúd- és támaszerők együtthatója nulla.)

3.2.3.2. Alakváltozás - elmozdulás mátrix

Jelen esetben is fel lehetne írni az alakváltozás - elmozdulás egyenleteket: a támaszokat rugalmas rudakként kezelve, minden rúd megnyúlását a rúdvégek elmozdulásának függvényében kifejezzük (x és y irányban külön-külön). Szerencsére ennél sokkal egyszerűbb az előbb létrehozott egyensúlyi mátrixot transzponálni.

Azért, hogy a későbbiekben könnyebben tudjuk majd kezelni az alakváltozás - elmozdulás egyenletrendszert, érdemesebb nullára redukálni. Ezáltal az egyenletrendszer mátrixa némileg módosul, az elejét az egyensúlyi mátrix transzponáltja, míg a végét az elmozdulások együtthatóiból kialakuló diagonál - hipermátrix (nevezzük **D** mátrixnak) képezi.

$$[\ \mathbf{B}^T \mid \mathbf{D}\ ]$$

3.2.3.3. Hooke-törvény

A deformációk együtthatóit az előző pontban leírt **D** diagonálmátrix tartalmazza, ennek tagjait kell megszorozni a vonatkozó rúd hosszával, valamint osztani a rúd húzási merevségével. (Nevezzük $\mathbf{D}_H$-nak az így előállított mátrixot.) Támaszok esetén a helyettesítő rugalmas rúd hosszát és keresztmetszetét javasolt egységnyire választani, míg rugalmassági modulusát igen nagyra.
(Megkönnyíti a számítógépes kezelést, ha **D** mátrixot már az előző pontban leválasztjuk és különálló mátrixként kezeljük.)

3.2.3.4. Terhek oszlopvektorrá alakítása

Jelen esetben csupán arra kell figyelni, hogy a terhek megfelelő vetületét alkalmazzuk (a teher szögének megfelelő szögfüggvényével szorozzuk meg a teher nagyságát). Természetesen csak az egyensúlyi egyenletekhez tartozik teherérték, így a tehervektor alsó sorai ($\mathbf{B}^T$ mátrix sorainak számával egyező) ismét zérus vektort fognak alkotni.

Fontos megjegyezni azt is, hogy az ismeretleneket tartalmazó oszlopvektor elején a rúderőket és a támaszerőket kell felsorolni, míg az alsó sorokban ($\mathbf{B}^T$ mátrix sorainak számával egyező) az elmozdulás ismeretlenek fognak szerepelni.

3.2.3.5. Az egyenletrendszer megoldása

Az egyensúlyi egyenletrendszert (illetve annak az elmozdulásokra bővített, azaz egy nullmátrixszal megtoldott változatát) kiegészítve az erő - elmozdulás egyenletrendszerrel (az átalakított deformáció - elmozdulás egyenletrendszer, a Hooke-törvényt is alkalmazva) megkapjuk a végső egyenletrendszer kvadratikus mátrixát.

$$\left[\begin{array}{c|c} \mathbf{0} & \mathbf{B} \\ \hline \mathbf{B}^T & \mathbf{D}_H \end{array}\right]$$

Bár elképzelhető más elrendezés is, javasolt az itt közöltet alkalmazni, mivel így szimmetrikus mátrixot kapunk eredményként.
Teljesen paraméteres egyenletrendszer esetében a rúderők, támaszerők és elmozdulások keresésekor érdemes a kapott mátrix inverzét venni, majd ezt megszorozni a tehervektorral.
Természetesen meg lehet oldani más módon is, de minél kevesebb a skalár tényező, egyes automatikus megoldó alprogramok (például a Matlab solve parancsa) annál több erőforrást és időt fognak igényelni.

4. ÖSSZEHASONLÍTÁS MÁS MÓDSZEREKKEL

Ebben a fejezetben a rácsostartókra alkalmazott törzstartó nélküli erőmódszerhez képest próbálom felsorolni más (határozatlan szerkezetekhez alkalmas) megoldási módszerek előnyeit és hátrányait. Ennek alapját Dr. Szabó János és Dr. Roller Béla könyve[5] adja.

4.1. ELMOZDULÁSMÓDSZER

Jelentős előnye, hogy merev csomópontok esetén is alkalmazható. Ezáltal a vizsgálhatóak a keretek és a befogott támaszú szerkezetek is, valamint a nyomatékokat is képes kezelni.
A megoldási elv alapjaiban nagyon hasonló a törzstartó nélküli erőmódszer elvéhez, hiszen a csomópontok elmozdulásából adódó szerkezeti rudak válaszreakcióját vizsgálja, ezekből a teljes szerkezet viselkedését leíró mátrixot állít fel. E merevségi mátrix egy kvadratikus, szimmetrikus hipermátrix, melynek dimenzióit a szabadon elmozduló csomópontok elmozdulási szabadságfoka adja meg. Általában megfelelő csomópontszámozással szalagmátrixszá alakítható, így tárolása kisebb kapacitást igényel.
Ugyanakkor jóval időigényesebb és összetettebb a teljes számítás elvégzése. (Főleg amiatt, hogy elsődlegesen az elmozdulásokat kapjuk meg eredményként, amiből vissza kell számítani a rúderőket.)

4.2. ÁLTALÁNOSÍTOTT ELMOZDULÁSMÓDSZER

Az előző pontban felsoroltak mind érvényesek erre az eljárásra is. Ezen felül további nagy előnye, hogy változó keresztmetszet, görbe rúd, megoszló teher vagy rúdközben ható teher esetén is legalább közelítő eredmény megadására alkalmas. Ugyanakkor megsokszorozódik a csomópontok száma, így még több tárolási és számítási kapacitást igényel. (Bár előbbi hátrányát kissé tompítja, hogy megfelelő csomópontszámozás esetén ún. kvázi-hiperdiagonálmátrixszá alakítható, ami még a szalagmátrixnál is könnyebben tárolható.) További előnye, hogy kinematikai vizsgálatra is alkalmas.
Ugyanakkor ismét kiemelném, hogy elsődlegesen az elmozdulásokat kapjuk meg eredményként, így a több csomópont azért is gond, mert a rúderők kiszámítása is még hosszadalmasabb műveletté válik.

4.3. ERŐMÓDSZER

Ahogyan a neve is mutatja, egyik nagy előnye, hogy (a törzstartó nélküli erőmódszerhez hasonlóan) eredményként magukat a rúderőket kapjuk meg. Igen hasznos módszer abban az esetben, ha csak néhány rúderőre vagyunk kíváncsiak. Szintén előnyös olyan esetekben, ha a vizsgált szerkezet határozatlansági fokszáma alacsony.
Viszont a számítások munka- és időigénye közel egyenes arányban nő a határozatlansági fokszámmal, ami komoly hátrány az egyesített erődmódszerhez képest.

4.4. STATIKAILAG HATÁROZATLAN TÖRZSTARTÓ MÓDSZERE

Ez az eljárás főleg több hasonló, összekapcsolt határozatlan szerkezet estén a leghatékonyabb. Lényege, hogy ott alkalmazzuk az erőmódszert és az elmozdulásmódszert, ahol azok előnyei jobban érvényesülnek. (Például szelemenekkel összekapcsolt befogott állások esetén a szelemenekben ható belső erőket erőmódszerrel számítjuk ki, míg az állásokat elmozdulásmódszerrel. Mivel az alszerkezeti csoportok többsége feltehetően azonos kialakítású és terhelésű, minden esetet csak egyszer kell megoldani.)
Legnagyobb hátránya, hogy nehezen algoritmizálható, hiszen az egész eljárás alapja az, hogy jó érzékkel különítjük el az erőmódszerrel és az elmozdulásmódszerrel megoldandó szerkezeti részeket.

5 Dr. Szabó János, Dr. Roller Béla: Rúdszerkezetek elmélete és számítása (Műszaki Könyvkiadó, Budapest, 1971)

5. FELHASZNÁLT IRODALOM ÉS PROGRAMNYELV

Bízom benne, hogy sikerült érthetően és szabatosan ismertetnem a törzstartó nélküli erőmódszer rácsos tartókra vonatkozó részét. Amennyiben esetleg sikerült felkeltenem az érdeklődést a téma iránt, bátran ajánlom a dolgozathoz felhasznált irodalmakat.

5.1. Felhasznált irodalom

- Dale A. Hopkins, Gary R. Halford és Surya N. Patnaik: Integrated Force Method Solution to Indeterminate Structural Mechanics Problems (2004) - http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20040045162.pdf (utoljára megtekintve: 2014. október)

- Ugyanerről a témáról egy bővebb kiadvány a szerzőktől:
  Surya Patnaik,Dale Hopkins: Strength of Materials - A New Unified Theory for the 21st Century (Butterworth-Heinemann, 2003)

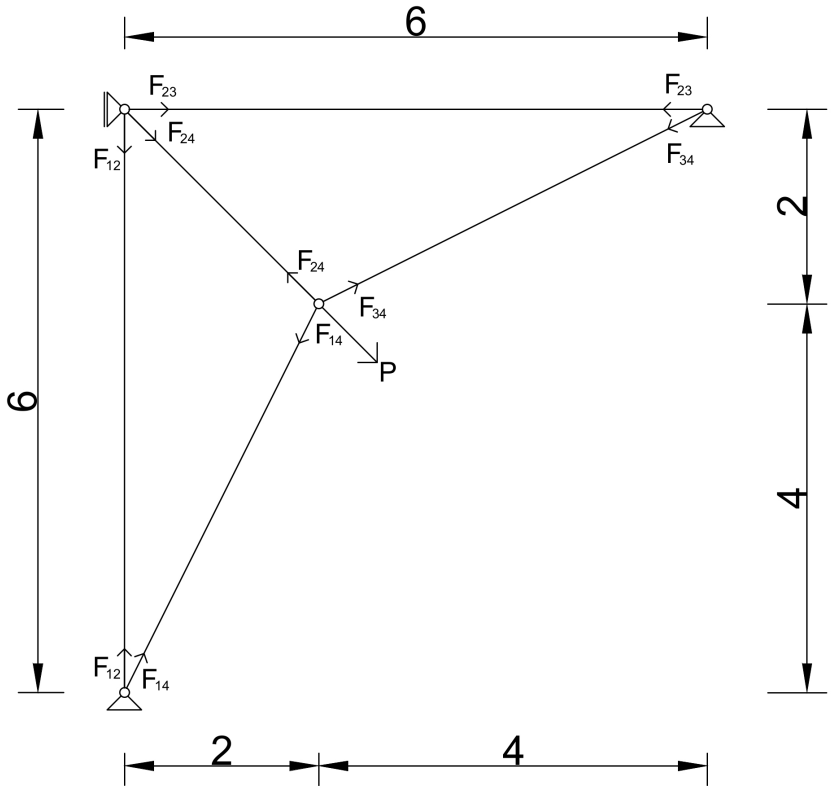
- Dr. Szabó János, Dr. Roller Béla: Rúdszerkezetek elmélete és számítása (Műszaki Könyvkiadó, Budapest, 1971)

5.2. Alkalmazott programnyelv

A mellékletben található kódokat **Matlab programnyelven** írtam, azokat megfelelő szoftverkörnyezetbe másolva a programok teljes értékűen működnek.

MELLÉKLET

6. PÉLDA EGY SZERKEZET MEGOLDÁSÁRA EGYESÍTETT ERŐMÓD-SZERREL



EA=áll.

$$P_x = P_y = \frac{P}{\sqrt{2}}$$

$$l_{14} = l_{34} = \sqrt{20}$$
$$l_{12} = l_{23} = 6$$
$$l_{24} = 2\sqrt{2}$$

A nem rögzített csomóponti irányokra felírt egyensúlyi egyenletek:

$$-F_{12} - \frac{1}{\sqrt{2}}F_{24} = 0$$
$$-\frac{1}{\sqrt{2}}F_{24} - \frac{1}{\sqrt{5}}F_{14} + \frac{2}{\sqrt{5}}F_{34} = -\frac{1}{\sqrt{2}}P$$
$$\frac{1}{\sqrt{2}}F_{24} - \frac{2}{\sqrt{5}}F_{14} + \frac{1}{\sqrt{5}}F_{34} = \frac{1}{\sqrt{2}}P$$

Az egyensúlyi egyenletrendszer mátrix alakja, valamint a transzponáltjából kialakított deformáció - elmozdulás egyenlet mátrix alakja:

$$\begin{matrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ 0 & \frac{-1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} \\ \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{matrix} \times \begin{matrix} v_2 \\ u_4 \\ v_4 \end{matrix} = \begin{matrix} \Delta l_{12} \\ \Delta l_{23} \\ \Delta l_{34} \\ \Delta l_{14} \\ \Delta l_{24} \end{matrix}$$

$$\begin{matrix} -1 & 0 & 0 & 0 & \frac{-1}{\sqrt{2}} \\ 0 & 0 & \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & \frac{-1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} & \frac{1}{\sqrt{2}} \end{matrix} \times \begin{matrix} F_{12} \\ F_{23} \\ F_{34} \\ F_{14} \\ F_{24} \end{matrix} = \begin{matrix} 0 \\ \frac{-F}{\sqrt{2}} \\ \frac{F}{\sqrt{2}} \end{matrix}$$

A deformáció - elmozdulás egyenletrendszer:

$$\Delta l_{12} = -v_2$$
$$\Delta l_{23} = 0$$
$$\Delta l_{34} = \frac{2}{\sqrt{5}}u_4 + \frac{1}{\sqrt{5}}v_4$$
$$\Delta l_{14} = -\frac{1}{\sqrt{5}}u_4 - \frac{2}{\sqrt{5}}v_4$$
$$\Delta l_{24} = -\frac{1}{\sqrt{2}}v_2 - \frac{1}{\sqrt{2}}u_4 + \frac{1}{\sqrt{2}}v_4$$

Ebből eliminálva az elmozdulásokat megkapjuk a kompatibilitási egyenleteket:

$$\Delta l_{23} = 0$$
$$\sqrt{2}\Delta l_{24} = \Delta l_{12} - \sqrt{5}(\Delta l_{34} + \Delta l_{14})$$

A Hooke-törvényt alkalmazva a következő kiegészítő egyenleteket kapjuk:

$$F_{23} = 0$$
$$0 = 6F_{12} - 10F_{14} - 10F_{34} - 4F_{24}$$

Az így adódó, immár határozott egyenletrendszer:

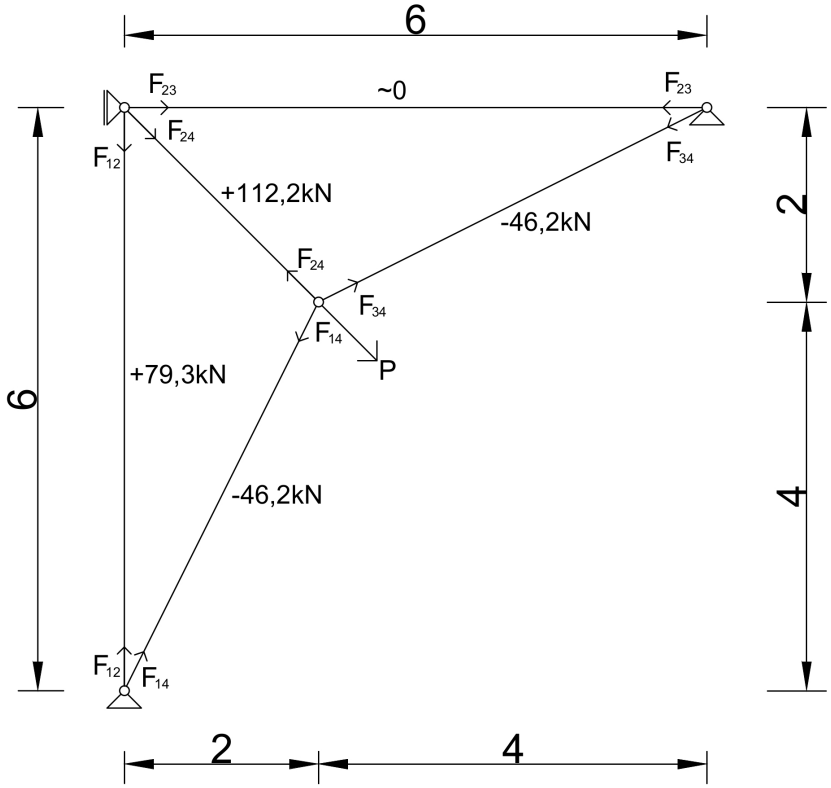
$$-F_{12} - \frac{1}{\sqrt{2}}F_{24} = 0$$
$$-\frac{1}{\sqrt{2}}F_{24} - \frac{1}{\sqrt{5}}F_{14} + \frac{2}{\sqrt{5}}F_{34} = -\frac{1}{\sqrt{2}}P$$
$$\frac{1}{\sqrt{2}}F_{24} - \frac{2}{\sqrt{5}}F_{14} + \frac{1}{\sqrt{5}}F_{34} = \frac{1}{\sqrt{2}}P$$
$$F_{23} = 0$$
$$0 = 6F_{12} - 10F_{14} - 10F_{34} - 4F_{24}$$

Ezt rendezve:

$$F_{12} = -\frac{1}{\sqrt{2}}F_{24}$$
$$F_{14} = F_{34}$$
$$F_{24} = P + \frac{\sqrt{2}}{\sqrt{5}}F_{14}$$
$$F_{14} = \frac{-(3\sqrt{2}+4)P}{\frac{4\sqrt{2}+6}{\sqrt{5}} + 20} \approx -46{,}2kN$$

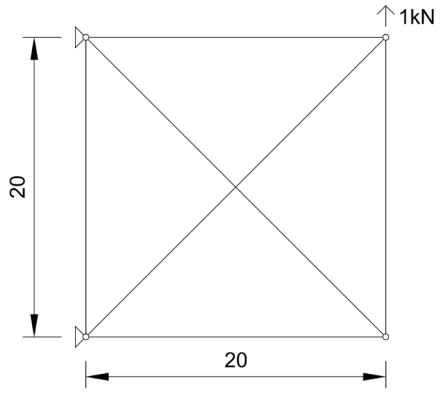
Tehát a keresett rúderők a következők:

$$F_{12} \approx 79{,}3kN$$
$$F_{23} = 0$$
$$F_{34} \approx -46{,}2kN$$
$$F_{14} \approx -46{,}2kN$$
$$F_{24} \approx 112{,}2kN$$



7. A GYORSABB PROGRAM KÓDJA ÉS EGY PÉLDA MEGOLDÁSA

A vizsgálandó szerkezet:



=====IFMgyors_Start.m==========

```
clear all
clc
‚Ide jön az a szöveg, hogy mit kell tudni a programról, valamint hogy hogyan kell kitölteni a mátrixokat.'
‚Például:'
kapcsolatok=[1,2;2,3;3,4;4,1;1,3;2,4]
```

=====IFMgyors_Start.m==========

```
kapcsolatok=[1,2;2,3;3,4;4,1;1,3;2,4]
```

=====IFMgyors_Tablazat.m==========

```
clc

kapcsolatok

rudak_szama=length(kapcsolatok);
csomopontok_szama=max(max(kapcsolatok));

%Csomópontok táblázatának elkészítése.

syms csomopont_szama x_koordinata y_koordinata gatolt_mozgas_x_iranyban gatolt_mozgas_y_iranyban teher-
_x_iranyban teher_y_iranyban;

‚Gátolt mozgás: 1, szabad mozgás: 0.'

CSPcim=[csomopont_szama, x_koordinata, y_koordinata, gatolt_mozgas_x_iranyban, gatolt_mozgas_y_irany-
ban, teher_x_iranyban, teher_y_iranyban];
CSPszam=[[1:csomopontok_szama]', zeros(csomopontok_szama,6)];
CSP=[CSPcim;CSPszam]

%Rudak táblázatának elkészítése.

syms rud_egyik_vege rud_masik_vege rugalmassagi_modulus keresztmetszet;
RUDcim=[rud_egyik_vege,rud_masik_vege, rugalmassagi_modulus, keresztmetszet];
RUDszam=[kapcsolatok, ones(rudak_szama,2)];
RUD=[RUDcim;RUDszam]

‚Adja meg az értékeket CSPszam és RUDszam mátrixokat megnyitva. Az >IFMgyors_Frissit.m< script indításával
frissíthet.'
```

=====IFMgyors_Tablazat.m==========

CSPszam módosítása:

1	20	20	0	0	-1	0
2	20	0	0	0	0	0
3	0	20	1	1	0	0
4	0	0	1	1	0	0

RUDszam módosítása:

```
1       2       210     300
2       3       210     300
3       4       210     300
4       1       210     300
1       3       210     300
2       4       210     300
```

=====IFMgyors_Frissit.m==========

```
clc
kapcsolatok
‚Gátolt mozgás: 1, szabad mozgás: 0.’
CSP=[CSPcim;CSPszam]
RUD=[RUDcim;RUDszam]
‚Adja meg az értékeket CSPszam és RUDszam mátrixokat megnyitva. A >IFMgyors_Frissit.m< script indításával frissíthet.’
```

=====IFMgyors_Frissit.m==========
=====IFMgyors_Megoldo.m==========

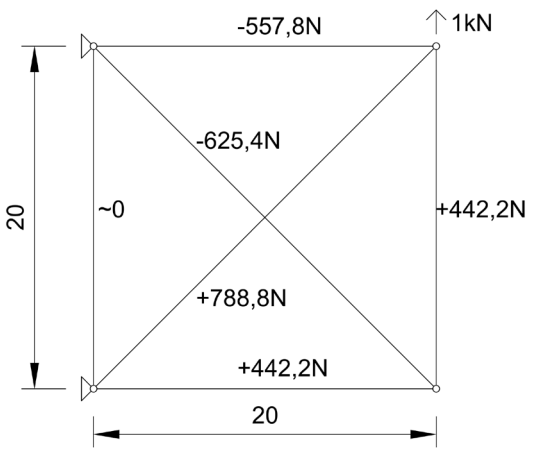
```
clc

CSP=[CSPcim;CSPszam]
RUD=[RUDcim;RUDszam]

%Egyensúlyi egyenletek mátrixának előállítása

syms Bx By

Bx=zeros(double(csomopontok_szama-sum(CSP(2:csomopontok_szama+1,4))),double(rudak_szama));
By=zeros(double(csomopontok_szama-sum(CSP(2:csomopontok_szama+1,5))),double(rudak_szama));
px=zeros(double(csomopontok_szama-sum(CSP(2:csomopontok_szama+1,4))),1);
py=zeros(double(csomopontok_szama-sum(CSP(2:csomopontok_szama+1,5))),1);


k=0;
for i=1:csomopontok_szama

  if CSP(i+1,4)==0
    k=k+1 ;
    px(k,1)=CSP(i+1,6);
     for j=1:rudak_szama

        if RUD((j+1),1)==i
         xegyik=CSP(i+1,2);
         xmasik=CSP(RUD(j+1,2)+1,2);
         yegyik=CSP(i+1,3);
         ymasik=CSP(RUD(j+1,2)+1,3);
         Bx(k,j)=(xegyik-xmasik)/sqrt(((xegyik-xmasik)^2)+((yegyik-ymasik)^2));


        elseif RUD((j+1),2)==i
         xegyik=CSP(i+1,2);
         xmasik=CSP(RUD(j+1,1)+1,2);
         yegyik=CSP(i+1,3);
         ymasik=CSP(RUD(j+1,1)+1,3);
         Bx(k,j)=(xegyik-xmasik)/sqrt(((xegyik-xmasik)^2)+((yegyik-ymasik)^2));

        end

     end

  elseif CSP(i+1,4)==1
```

```
  else
      ,Hibás megadás! Gátolt mozgás: 1, szabad mozgás: 0.'

  end

end
k=0;
for i=1:csomopontok_szama

  if CSP(i+1,5)==0
        k=k+1;
        py(k,1)=CSP(i+1,7);
     for j=1:rudak_szama

        if RUD((j+1),1)==i
         xegyik=CSP(i+1,2);
         xmasik=CSP(RUD(j+1,2)+1,2);
         yegyik=CSP(i+1,3);
         ymasik=CSP(RUD(j+1,2)+1,3);
         By(k,j)=(yegyik-ymasik)/sqrt(((xegyik-xmasik)^2)+((yegyik-ymasik)^2));


        elseif RUD((j+1),2)==i
         xegyik=CSP(i+1,2);
         xmasik=CSP(RUD(j+1,1)+1,2);
         yegyik=CSP(i+1,3);
         ymasik=CSP(RUD(j+1,1)+1,3);
         By(k,j)=(yegyik-ymasik)/sqrt(((xegyik-xmasik)^2)+((yegyik-ymasik)^2));

        end
     end

  elseif CSP(i+1,5)==1

  else
      ,Hibás megadás! Gátolt mozgás: 1, szabad mozgás: 0.'

  end

end

B=[Bx;By];
pc=zeros(double(rudak_szama-min(size(B))),1);
p=[px;py;pc];
fok=(rudak_szama-min(size(B)))
Ct=sym(,Ct',[rudak_szama fok]);

F=sym(,F', [rudak_szama 1]);
Fcim=F;

%Fölös Ct-k és mozdulatlan rudak kiejtése.
k=0;
for i=1:rudak_szama
   if B(:,i)==0
     Ct(i,:)=1;
     F(i)=0;
   else
      if k==0
         Ct(i,:)=1;
         k=1;
      end
   end
end
sol=solve(B*Ct==0);
```

```
for i=1:rudak_szama
   for j=1:fok

      if Ct(i,j)==1

      else
      Ct(i,j)=eval([,sol.', char(sym(Ct(i,j)))]);
      end
   end
end
%Hooke-törvény
M=zeros(double(rudak_szama));
for i=1:rudak_szama
         xegyik=CSP(RUD(i+1,1)+1,2);
         xmasik=CSP(RUD(i+1,2)+1,2);
         yegyik=CSP(RUD(i+1,1)+1,3);
         ymasik=CSP(RUD(i+1,2)+1,3);

   M(i,i)=sqrt(((xegyik-xmasik)^2)+((yegyik-ymasik)^2))/(RUD((i+1),3)*RUD((i+1),4));
end

C=Ct.';

CM=C*M;

BCM=[B;CM];

sol=solve(BCM*F==p);

for i=1:rudak_szama
   if F(i)==0

   else
      F(i)=eval([,sol.F', num2str(i)]);
   end
end

Fcim==F

syms F2 F3;
for i=1:(max(size(F)))
F2(i,1)=char(vpa(double(F(i)),4));
end

Fcim==F2
```

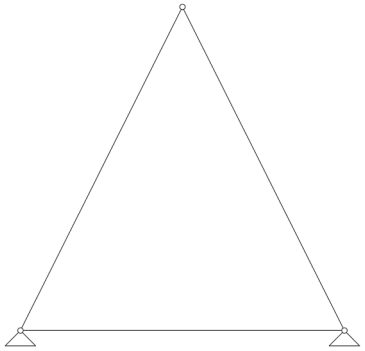
=====IFMgyors_Megoldo.m==========

A kapott rúderők segítségével:



8. AZ ÁLTALÁNOSABB PROGRAM KÓDJA ÉS EGY PÉLDA MEGOLDÁSA

A vizsgálandó szerkezet struktúrája:



=====IFMaltalanos_Start.m==========

```
clear all
clc
‚Ide jön az a szöveg, hogy mit kell tudni a programról, valamint hogy hogyan kell kitölteni a mátrixokat.’
‚Például:’
kapcsolatok=[1,2;2,3;3,1]
tamaszok=[1;2]
```

=====IFMaltalanos_Start.m==========

```
kapcsolatok=[1,2;2,3;3,1]
tamaszok=[1;2]
```

=====IFMaltalanos_Tablazat.m==========

```
clc

kapcsolatok
tamaszok

rudak_szama=length(kapcsolatok);
csomopontok_szama=max(max(kapcsolatok));
tamaszok_szama=length(tamaszok);

%Csomópontok táblázatának elkészítése.
syms x y p alpha dx dy;
syms x_koordinata y_koordinata teher teher_szoge x_elmozdulasa y_elmozdulasa;
x=x_koordinata;
y=y_koordinata;
p=teher;
alpha=teher_szoge;
dx=x_elmozdulasa;
dy=y_elmozdulasa;
for i=1:csomopontok_szama

    %x-ek
    eval([,syms x’,num2str(i)]);
    x=[x, eval([,x’,num2str(i)])];

    %y-ok
    eval([,syms y’,num2str(i)]);
    y=[y, eval([,y’,num2str(i)])];

    %terhek nagysága
    eval([,syms p’,num2str(i)]);
    p=[p, eval([,p’,num2str(i)])];

    %terhek szöge
    eval([,syms alpha’,num2str(i)]);
    alpha=[alpha, eval([,alpha’,num2str(i)])];
```

```
    %x irányú elmozdulás
    eval([,syms dx',num2str(i)]);
    dx=[dx, eval([,dx',num2str(i)])];

    %y irányú elmozdulás
    eval([,syms dy',num2str(i)]);
    dy=[dy, eval([,dy',num2str(i)])];

end

CSP=[x;y;p;alpha;dx;dy].'

%Rudak táblázatának elkészítése.
syms E A F;
syms rud_egyik_vege rud_masik_vege rugalmassagi_modulus keresztmetszet rudero;
kapcsolatok_rud=[rud_egyik_vege,rud_masik_vege;kapcsolatok];
E=rugalmassagi_modulus;
A=keresztmetszet;
F=rudero;

for i=1:rudak_szama

    %E-k
    eval([,syms E',num2str(i)]);
    E=[E, eval([,E',num2str(i)])];

    %A-k
    eval([,syms A',num2str(i)]);
    A=[A, eval([,A',num2str(i)])];

    %F-ek
    eval([,syms F',num2str(i)]);
    F=[F, eval([,F',num2str(i)])];

end

RUD=[E;A;F].';
RUD=[kapcsolatok_rud,RUD]

%Támaszok táblázatának elkészítése.
syms beta Rp Rm;
syms kapcsolodo_csomopont tamasz_tipusa tamasz_szoge szog_iranyu_reakcio szogre_meroleges_reakcio;
tamaszok_tamasz=[tamaszok,ones(tamaszok_szama,1)];
tamaszok_tamasz=[kapcsolodo_csomopont,tamasz_tipusa;tamaszok_tamasz].';
beta=tamasz_szoge;
Rp=szog_iranyu_reakcio;
Rm=szogre_meroleges_reakcio;

for i=1:tamaszok_szama

    %Támasz-szögek
    eval([,syms beta',num2str(i)]);
    beta=[beta, eval([,beta',num2str(i)])];

    %Párhuzamos reakciók
    eval([,syms Rp',num2str(i)]);
    Rp=[Rp, eval([,Rp',num2str(i)])];

    %Merőleges reakciók
    eval([,syms Rm',num2str(i)]);
    Rm=[Rm, eval([,Rm',num2str(i)])];

end
```

```
 TAMASZ=[tamaszok_tamasz;beta;Rp;Rm].'

syms Efix;
Efix=Inf

,Támaszok típusa: 0-görgős, 1-fix'
,Adja meg az értékeket (pl. x1=2), majd az >eval< parancs segítségével megnézheti a kitöltött táblázatot: pl.
CSP=eval(CSP).'
```

=====IFMaltalanos_Tablazat.m==========

Mivel jelen esetben egy általános megoldást szeretnénk (például több ilyen struktúrájú szerkezetünk is van), ezért most mindent paraméteresen hagyunk.

=====IFMaltalanos_Megoldo.m==========

```
clc
eval(CSP);
CSP
eval(RUD);
RUD
eval(TAMASZ);
TAMASZ

fix_tamaszok_szama=TAMASZ(:,2);
fix_tamaszok_szama=sum(fix_tamaszok_szama(2:(max(size(fix_tamaszok_szama)))));
fix_tamaszok_szama=double(fix_tamaszok_szama);
%Egyensúlyi egyenletek mátrixának előállítása
Bx=sym(,Bx',[csomopontok_szama (rudak_szama+tamaszok_szama+fix_tamaszok_szama)]);
By=sym(,By',[csomopontok_szama (rudak_szama+tamaszok_szama+fix_tamaszok_szama)]);
for i=1:csomopontok_szama

   for j=1:rudak_szama
      if RUD((j+1),1)==i
         xegyik=eval([,x',num2str(i)]);
         xmasik=eval([,x',char(RUD((j+1),2))]);
         yegyik=eval([,y',num2str(i)])    ;
         ymasik=eval([,y',char(RUD((j+1),2))]);
         Bx(i,j)=(xegyik-xmasik)/sqrt(((xegyik-xmasik)^2)+((yegyik-ymasik)^2));

         By(i,j)=(yegyik-ymasik)/sqrt(((xegyik-xmasik)^2)+((yegyik-ymasik)^2));

      elseif RUD((j+1),2)==i
         xegyik=eval([,x',num2str(i)])  ;
         xmasik=eval([,x',char(RUD((j+1),1))]);
         yegyik=eval([,y',num2str(i)])         ;
         ymasik=eval([,y',char(RUD((j+1),1))]);
         Bx(i,j)=(xegyik-xmasik)/sqrt(((xegyik-xmasik)^2)+((yegyik-ymasik)^2));

         By(i,j)=(yegyik-ymasik)/sqrt(((xegyik-xmasik)^2)+((yegyik-ymasik)^2));

      else
         Bx(i,j)=0;
         By(i,j)=0;
      end

   end
   %támaszrudak
   for j=(rudak_szama+1):(rudak_szama+tamaszok_szama)

      if TAMASZ((j-rudak_szama+1),1)==i
         Bx(i,j)=cos(TAMASZ((j-rudak_szama+1),3));
         By(i,j)=sin(TAMASZ((j-rudak_szama+1),3));
      else
         Bx(i,j)=0;
```

```
            By(i,j)=0;
        end
    end

    for j=(rudak_szama+tamaszok_szama+1):(rudak_szama+2*tamaszok_szama)
        if TAMASZ((j-rudak_szama-tamaszok_szama+1),2)==1

        if TAMASZ((j-rudak_szama-tamaszok_szama+1),1)==i
            Bx(i,j)=cos(TAMASZ((j-rudak_szama-tamaszok_szama+1),3)+pi/2);
            By(i,j)=sin(TAMASZ((j-rudak_szama-tamaszok_szama+1),3)+pi/2);
        else
            Bx(i,j)=0;
            By(i,j)=0;
        end

        elseif TAMASZ((j-rudak_szama-tamaszok_szama+1),2)==0
            j=j+1;
        else
            ,Hibás támasz típus! 0-görgős 1-fix'
        end
    end
end

B=[Bx;By];

%Elmozulások mátrixa
syms Bt;
Bt=B.';

%Merevségi/rugalmassági mátrix
%D=sym(,D',[(rudak_szama+2*tamaszok_szama) (rudak_szama+2*tamaszok_szama)]);
D=zeros((rudak_szama+tamaszok_szama+fix_tamaszok_szama));
D=sym(D);
for i=1:rudak_szama
            xegyik=eval([,x',char(RUD((i+1),1))]);
            xmasik=eval([,x',char(RUD((i+1),2))]);
            yegyik=eval([,y',char(RUD((i+1),1))])    ;
            ymasik=eval([,y',char(RUD((i+1),2))]);

    D(i,i)=sqrt(((xegyik-xmasik)^2)+((yegyik-ymasik)^2))/(RUD((i+1),3)*RUD((i+1),4));
end
for i=(rudak_szama+1):(rudak_szama+tamaszok_szama)
    D(i,i)=1/Efix;
end
for i=(rudak_szama+tamaszok_szama+1):(rudak_szama+2*tamaszok_szama)
    if TAMASZ((i-rudak_szama-tamaszok_szama+1),2)==1
        D(i,i)=1/Efix;
    elseif TAMASZ((i-rudak_szama-tamaszok_szama+1),2)==0
        i=i+1;
    else
        ,Hibás támasz típus! 0-görgős 1-fix'
    end
end
D=-1*D;

%Nagy mátrix
N=[sym(zeros(2*csomopontok_szama)).B;Bt,D]

%Erő és elmozdulás vektor
f=zeros((rudak_szama+tamaszok_szama+fix_tamaszok_szama+2*csomopontok_szama),1);
f=sym(f);
for i=1:rudak_szama
    f(i,1)=RUD((i+1),5);
end
```

```
for i=(rudak_szama+1):(rudak_szama+tamaszok_szama)
   f(i,1)=TAMASZ((i-rudak_szama+1),4);
end
for i=(rudak_szama+tamaszok_szama+1):(rudak_szama+2*tamaszok_szama)
   if TAMASZ((i-rudak_szama-tamaszok_szama+1),2)==1
   f(i,1)=TAMASZ((i-rudak_szama-tamaszok_szama+1),5);
   elseif TAMASZ((i-rudak_szama-tamaszok_szama+1),2)==0
      i=i+1;
   else
      ,Hibás támasz típus! 0-görgős 1-fix'
   end
end
for i=(rudak_szama+tamaszok_szama+fix_tamaszok_szama+1):(rudak_szama+tamaszok_szama+fix_tamaszok_
szama+csomopontok_szama)
   f(i,1)=CSP((i-rudak_szama-tamaszok_szama-fix_tamaszok_szama+1),5);
end
for i=(rudak_szama+tamaszok_szama+fix_tamaszok_szama+csomopontok_szama+1):(rudak_szama+tamaszok_
szama+fix_tamaszok_szama+2*csomopontok_szama)
   f(i,1)=CSP((i-rudak_szama-tamaszok_szama-fix_tamaszok_szama-csomopontok_szama+1),6);
end

%Terhek vektor
e=zeros((rudak_szama+tamaszok_szama+fix_tamaszok_szama+2*csomopontok_szama),1);
e=sym(e);
for i=1:csomopontok_szama
   e(i,1)=CSP((i+1),3)*cos(CSP((i+1),4));
end
for i=(csomopontok_szama+1):(2*csomopontok_szama)
   e(i,1)=CSP((i-csomopontok_szama+1),3)*sin(CSP((i-csomopontok_szama+1),4));
end

f==inv(N)*e
```

=====IFMaltalanos_Megoldo.m==========

Ezek alapján a következő megoldást kapjuk:

F1 == 0

F2 == (p3*cos(alpha3)*(y1 - y3)*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(1/2))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (p3*sin(alpha3)*(x1 - x3)*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(1/2))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2)
F3 == (p3*sin(alpha3)*(x2 - x3)*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(1/2))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2) - (p3*cos(alpha3)*(y2 - y3)*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(1/2))/(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2)

Rp1 == (p1*cos(alpha1)*sin(pi/2 + beta1))/(cos(beta1)*sin(pi/2 + beta1) - cos(pi/2 + beta1)*sin(beta1)) - (p1*sin(alpha1)*cos(pi/2 + beta1))/(cos(beta1)*sin(pi/2 + beta1) - cos(pi/2 + beta1)*sin(beta1)) - (p3*cos(alpha3)*(y3^2*cos(pi/2 + beta1) + y1*y2*cos(pi/2 + beta1) - y1*y3*cos(pi/2 + beta1) - y2*y3*cos(pi/2 + beta1) - x1*y2*sin(pi/2 + beta1) + x1*y3*sin(pi/2 + beta1) + x3*y2*sin(pi/2 + beta1) - x3*y3*sin(pi/2 + beta1)))/(x1*y2*cos(beta1)*sin(pi/2 + beta1) - x1*y2*cos(pi/2 + beta1)*sin(beta1) - x2*y1*cos(beta1)*sin(pi/2 + beta1) + x2*y1*cos(pi/2 + beta1)*sin(beta1) - x1*y3*cos(beta1)*sin(pi/2 + beta1) + x1*y3*cos(pi/2 + beta1)*sin(beta1) + x3*y1*cos(beta1)*sin(pi/2 + beta1) - x3*y1*cos(pi/2 + beta1)*sin(beta1) + x2*y3*cos(beta1)*sin(pi/2 + beta1) - x2*y3*cos(pi/2 + beta1)*sin(beta1) - x3*y2*cos(beta1)*sin(pi/2 + beta1) + x3*y2*cos(pi/2 + beta1)*sin(beta1)) - (p3*sin(alpha3)*(x3^2*sin(pi/2 + beta1) - x2*y1*cos(pi/2 + beta1) + x3*y1*cos(pi/2 + beta1) + x2*y3*cos(pi/2 + beta1) - x3*y3*cos(pi/2 + beta1) + x1*x2*sin(pi/2 + beta1) - x1*x3*sin(pi/2 + beta1) - x2*x3*sin(pi/2 + beta1)))/(x1*y2*cos(beta1)*sin(pi/2 + beta1) - x1*y2*cos(pi/2 + beta1)*sin(beta1) - x2*y1*cos(beta1)*sin(pi/2 + beta1) + x2*y1*cos(pi/2 + beta1)*sin(beta1) - x1*y3*cos(beta1)*sin(pi/2 + beta1) + x1*y3*cos(pi/2 + beta1)*sin(beta1) + x3*y1*cos(beta1)*sin(pi/2 + beta1) - x3*y1*cos(pi/2 + beta1)*sin(beta1) + x2*y3*cos(beta1)*sin(pi/2 + beta1) - x2*y3*cos(pi/2 + beta1)*sin(beta1) - x3*y2*cos(beta1)*sin(pi/2 + beta1) + x3*y2*cos(pi/2 + beta1)*sin(beta1))

Rp2 == (p2*cos(alpha2)*sin(pi/2 + beta2))/(cos(beta2)*sin(pi/2 + beta2) - cos(pi/2 + beta2)*sin(beta2)) - (p2*sin(alpha2)*cos(pi/2 + beta2))/(cos(beta2)*sin(pi/2 + beta2) - cos(pi/2 + beta2)*sin(beta2)) + (p3*cos(alpha3)*(y3^2*cos(pi/2 + beta2) + y1*y2*cos(pi/2 + beta2) - y1*y3*cos(pi/2 + beta2) - y2*y3*cos(pi/2 + beta2) - x2*y1*sin(pi/2 + beta2) + x3*y1*sin(pi/2 + beta2) + x2*y3*sin(pi/2 + beta2) - x3*y3*sin(pi/2 + beta2)))/((cos(beta2)*sin(pi/2 + beta2) - cos(pi/2 + beta2)*sin(beta2))*(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2)) + (p3*sin(alpha3)*(x3^2*sin(pi/2 + beta2) - x1*y2*cos(pi/2 + beta2) + x1*y3*cos(pi/2 + beta2) + x3*y2*cos(pi/2 + beta2) - x3*y3*cos(pi/2 + beta2) + x1*x2*sin(pi/2 + beta2) - x1*x3*sin(pi/2 + beta2) - x2*x3*sin(pi/2 + beta2)))/((cos(beta2)*sin(pi/2 + beta2) - cos(pi/2 + beta2)*sin(beta2))*(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2))

Rm1 == (p1*cos(beta1)*sin(alpha1))/(cos(beta1)*sin(pi/2 + beta1) - cos(pi/2 + beta1)*sin(beta1)) - (p1*cos(alpha1)*sin(beta1))/(cos(beta1)*sin(pi/2 + beta1) - cos(pi/2 + beta1)*sin(beta1)) + (p3*cos(alpha3)*(y3^2*cos(beta1) + y1*y2*cos(beta1) - y1*y3*cos(beta1) - y2*y3*cos(beta1) - x1*y2*sin(beta1) + x1*y3*sin(beta1) + x3*y2*sin(beta1) - x3*y3*sin(beta1)))/(x1*y2*cos(beta1)*sin(pi/2 + beta1) - x1*y2*cos(pi/2 + beta1)*sin(beta1) - x2*y1*cos(beta1)*sin(pi/2 + beta1) + x2*y1*cos(pi/2 + beta1)*sin(beta1) - x1*y3*cos(beta1)*sin(pi/2 + beta1) + x1*y3*cos(pi/2 + beta1)*sin(beta1) + x3*y1*cos(beta1)*sin(pi/2 + beta1) - x3*y1*cos(pi/2 + beta1)*sin(beta1) + x2*y3*cos(beta1)*sin(pi/2 + beta1) - x2*y3*cos(pi/2 + beta1)*sin(beta1) - x3*y2*cos(beta1)*sin(pi/2 + beta1) + x3*y2*cos(pi/2 + beta1)*sin(beta1)) + (p3*sin(alpha3)*(x3^2*sin(beta1) - x2*y1*cos(beta1) + x3*y1*cos(beta1) + x2*y3*cos(beta1) - x3*y3*cos(beta1) + x1*x2*sin(beta1) - x1*x3*sin(beta1) - x2*x3*sin(beta1)))/


```
```

(x1*y2*cos(beta1)*sin(pi/2 + beta1) - x1*y2*cos(pi/2 + beta1)*sin(beta1) - x2*y1*cos(beta1)*sin(pi/2 + beta1) + x2*y1*cos(pi/2 + beta1)*sin(beta1) - x1*y3*cos(beta1)*sin(pi/2 + beta1) + x1*y3*cos(pi/2 + beta1)*sin(beta1) + x3*y1*cos(beta1)*sin(pi/2 + beta1) - x3*y1*cos(pi/2 + beta1)*sin(beta1) + x2*y3*cos(beta1)*sin(pi/2 + beta1) - x2*y3*cos(pi/2 + beta1)*sin(beta1) - x3*y2*cos(beta1)*sin(pi/2 + beta1) + x3*y2*cos(pi/2 + beta1)*sin(beta1))

Rm2 == (p2*cos(beta2)*sin(alpha2))/(cos(beta2)*sin(pi/2 + beta2) - cos(pi/2 + beta2)*sin(beta2)) - (p2*cos(alpha2)*sin(beta2))/(cos(beta2)*sin(pi/2 + beta2) - cos(pi/2 + beta2)*sin(beta2)) - (p3*cos(alpha3)*(y3^2*cos(beta2) + y1*y2*cos(beta2) - y1*y3*cos(beta2) - y2*y3*cos(beta2) - x2*y1*sin(beta2) + x3*y1*sin(beta2) + x2*y3*sin(beta2) - x3*y3*sin(beta2)))/((cos(beta2)*sin(pi/2 + beta2) - cos(pi/2 + beta2)*sin(beta2))*(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2)) - (p3*sin(alpha3)*(x3^2*sin(beta2) - x1*y2*cos(beta2) + x1*y3*cos(beta2) + x3*y2*cos(beta2) - x3*y3*cos(beta2) + x1*x2*sin(beta2) - x1*x3*sin(beta2) - x2*x3*sin(beta2)))/((cos(beta2)*sin(pi/2 + beta2) - cos(pi/2 + beta2)*sin(beta2))*(x1*y2 - x2*y1 - x1*y3 + x3*y1 + x2*y3 - x3*y2))

dx1 == 0

dx2 == 0

dx3 == (p3*cos(alpha3)*(A2*E2*y2^2*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) + A2*E2*y3^2*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) + A3*E3*y1^2*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2) + A3*E3*y3^2*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2) - 2*A2*E2*y2*y3*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) - 2*A3*E3*y1*y3*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2)))/(A2*A3*E2*E3*x1^2*y2^2 - 2*A2*A3*E2*E3*x1^2*y2*y3 + A2*A3*E2*E3*x1^2*y3^2 - 2*A2*A3*E2*E3*x1*x2*y1*y2 + 2*A2*A3*E2*E3*x1*x2*y1*y3 + 2*A2*A3*E2*E3*x1*x2*y2*y3 - 2*A2*A3*E2*E3*x1*x2*y3^2 + 2*A2*A3*E2*E3*x1*x3*y1*y2 - 2*A2*A3*E2*E3*x1*x3*y1*y3 - 2*A2*A3*E2*E3*x1*x3*y2^2 + 2*A2*A3*E2*E3*x1*x3*y2*y3 + A2*A3*E2*E3*x2^2*y1^2 - 2*A2*A3*E2*E3*x2^2*y1*y3 + A2*A3*E2*E3*x2^2*y3^2 - 2*A2*A3*E2*E3*x2*x3*y1^2 + 2*A2*A3*E2*E3*x2*x3*y1*y2 + 2*A2*A3*E2*E3*x2*x3*y1*y3 - 2*A2*A3*E2*E3*x2*x3*y2*y3 + A2*A3*E2*E3*x3^2*y1^2 - 2*A2*A3*E2*E3*x3^2*y1*y2 + A2*A3*E2*E3*x3^2*y2^2) - (p3*sin(alpha3)*(A2*E2*x2*y2*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) - A2*E2*x2*y3*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) - A2*E2*x3*y2*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) + A2*E2*x3*y3*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) + A3*E3*x1*y1*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2) - A3*E3*x1*y3*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2) - A3*E3*x3*y1*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2) + A3*E3*x3*y3*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2)))/(A2*A3*E2*E3*x1^2*y2^2 - 2*A2*A3*E2*E3*x1^2*y2*y3 + A2*A3*E2*E3*x1^2*y3^2 - 2*A2*A3*E2*E3*x1*x2*y1*y2 + 2*A2*A3*E2*E3*x1*x2*y1*y3 + 2*A2*A3*E2*E3*x1*x2*y2*y3 - 2*A2*A3*E2*E3*x1*x2*y3^2 + 2*A2*A3*E2*E3*x1*x3*y1*y2 - 2*A2*A3*E2*E3*x1*x3*y1*y3 - 2*A2*A3*E2*E3*x1*x3*y2^2 + 2*A2*A3*E2*E3*x1*x3*y2*y3 + A2*A3*E2*E3*x2^2*y1^2 - 2*A2*A3*E2*E3*x2^2*y1*y3 + A2*A3*E2*E3*x2^2*y3^2 - 2*A2*A3*E2*E3*x2*x3*y1^2 + 2*A2*A3*E2*E3*x2*x3*y1*y2 + 2*A2*A3*E2*E3*x2*x3*y1*y3 - 2*A2*A3*E2*E3*x2*x3*y2*y3 + A2*A3*E2*E3*x3^2*y1^2 - 2*A2*A3*E2*E3*x3^2*y1*y2 + A2*A3*E2*E3*x3^2*y2^2)

dy1 == 0

dy2 == 0

dy3 == (p3*sin(alpha3)*(A2*E2*x2^2*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) + A2*E2*x3^2*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) + A3*E3*x1^2*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2) + A3*E3*x3^2*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2) - 2*A2*E2*x2*x3*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) - 2*A3*E3*x1*x3*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2)))/(A2*A3*E2*E3*x1^2*y2^2 - 2*A2*A3*E2*E3*x1^2*y2*y3 + A2*A3*E2*E3*x1^2*y3^2 - 2*A2*A3*E2*E3*x1*x2*y1*y2 + 2*A2*A3*E2*E3*x1*x2*y1*y3 + 2*A2*A3*E2*E3*x1*x2*y2*y3 - 2*A2*A3*E2*E3*x1*x2*y3^2 + 2*A2*A3*E2*E3*x1*x3*y1*y2 - 2*A2*A3*E2*E3*x1*x3*y1*y3 - 2*A2*A3*E2*E3*x1*x3*y2^2 + 2*A2*A3*E2*E3*x1*x3*y2*y3 + A2*A3*E2*E3*x2^2*y1^2 - 2*A2*A3*E2*E3*x2^2*y1*y3 + A2*A3*E2*E3*x2^2*y3^2 - 2*A2*A3*E2*E3*x2*x3*y1^2 + 2*A2*A3*E2*E3*x2*x3*y1*y2 + 2*A2*A3*E2*E3*x2*x3*y1*y3 - 2*A2*A3*E2*E3*x2*x3*y2*y3 + A2*A3*E2*E3*x3^2*y1^2 - 2*A2*A3*E2*E3*x3^2*y1*y2 + A2*A3*E2*E3*x3^2*y2^2) - (p3*cos(alpha3)*(A2*E2*x2*y2*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) - A2*E2*x2*y3*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) - A2*E2*x3*y2*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) + A2*E2*x3*y3*(x1^2 - 2*x1*x3 + x3^2 + y1^2 - 2*y1*y3 + y3^2)^(3/2) + A3*E3*x1*y1*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2) - A3*E3*x1*y3*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2) - A3*E3*x3*y1*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2) + A3*E3*x3*y3*(x2^2 - 2*x2*x3 + x3^2 + y2^2 - 2*y2*y3 + y3^2)^(3/2)))/(A2*A3*E2*E3*x1^2*y2^2 - 2*A2*A3*E2*E3*x1^2*y2*y3 + A2*A3*E2*E3*x1^2*y3^2 - 2*A2*A3*E2*E3*x1*x2*y1*y2 + 2*A2*A3*E2*E3*x1*x2*y1*y3 + 2*A2*A3*E2*E3*x1*x2*y2*y3 - 2*A2*A3*E2*E3*x1*x2*y3^2 + 2*A2*A3*E2*E3*x1*x3*y1*y2 - 2*A2*A3*E2*E3*x1*x3*y1*y3 - 2*A2*A3*E2*E3*x1*x3*y2^2 + 2*A2*A3*E2*E3*x1*x3*y2*y3 + A2*A3*E2*E3*x2^2*y1^2 - 2*A2*A3*E2*E3*x2^2*y1*y3 + A2*A3*E2*E3*x2^2*y3^2 - 2*A2*A3*E2*E3*x2*x3*y1^2 + 2*A2*A3*E2*E3*x2*x3*y1*y2 + 2*A2*A3*E2*E3*x2*x3*y1*y3 - 2*A2*A3*E2*E3*x2*x3*y2*y3 + A2*A3*E2*E3*x3^2*y1^2 - 2*A2*A3*E2*E3*x3^2*y1*y2 + A2*A3*E2*E3*x3^2*y2^2)

Ebbe egy tetszőleges paraméterekkel rendelkező tartót behelyettesítve megkapjuk a rúderőket, a támaszerőket és az elmozdulásokat: