BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM

ELEKTRONIKUS ESZKÖZÖK TANÉSZKE

Témalaboratórium - BMEVIEEAL02 beszámoló

IT20. Végeselemes megoldó részletének kidolgozása C++14-ben (MS)

Trembeczki Bálint

EA795D

Trembeczki.balint@simonyi.bme.hu

Konzulensek:

Dr. Czirkos Zoltán (BME-EET),

Dr. Nasztanovics Ferenc (Morgan Stanley)

2017. 12. 05.

Tartalomjegyzék

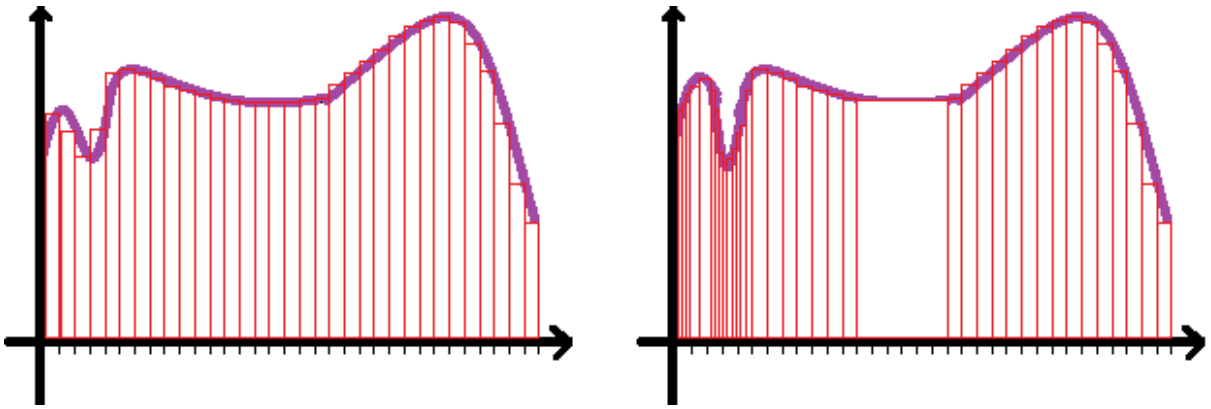
FEM – VEM

Bevezetés

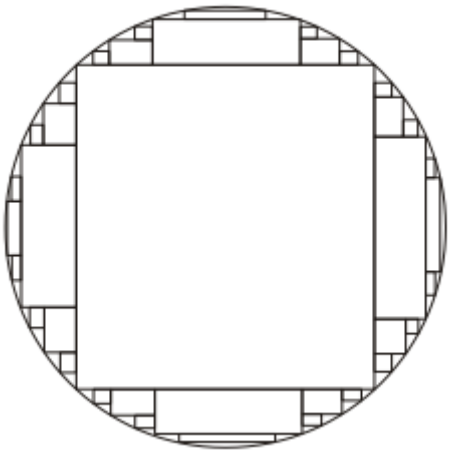
A végeselem-módszer egy olyan matematikai numerikus megoldó, amely parciális differenciálegyenletek közelítő megoldását adja meg. Eredeti célja egzakt megoldással nem rendelkező parciális differenciál egyenletrendszerek viszonylag pontos megoldása 'numerikusan'. Ennek köszönhetően ezek az egyébként nem megoldható problémák, nemcsak hogy megoldhatóvá, de könnyen programozható megoldókkal megadhatóak lehetnek. A mérnöki tudományokon kívül, ahol szinte mindent, a tér- és időfüggő problémákat parciális differenciálegyenletek írnak le (illetve az oktatás során elég sokszor egyszerűsített modellekkel találkozhatunk, amelyek azonban komolyabb felhasználási területeken nem adnak elég pontos közelítést), a pénzügyi tudományokban is használt matematikai formuláról van szó (derivatívaárazás). Talán a legjobb példa a mechanika területe: mind a statikai , dinamikai és szilárdságtani feladatokat is parciális differenciálegyenletrendszerek írnak le, azonban ez a helyzet hővezetéses, áramlástani, illetve elektromágneses problémák esetén, illetve csatolt feladatoknál, amik olyan esetek, ahol akár a fenti, akár eddig nem említett folyamatok együttes jelenlétével kell számolnunk. De ugyancsak parciális differenciálegyenletekkel találkozhatunk a saját szakom meghatározó területein: a fentebb említett elektromágnesesség és hullámtanban, a nemlineáris áramkörök működésének leírásában, irányítástechnikában a vezérlők leírásában, vagy akár a modern számítástechnika alappillérében, a félvezetőtechnonológia alapját képező diffúziós egyenletekben. Ezekre a végeselem-módszer az egyik legkönnyebben használható módszer, noha semmiképpen sem mondhatjuk, hogy az egyetlen. (Peremelemmódszer, véges differenciálmódszer, stb) A végeselemű módszer legnagyobb előnye a többivel szemben, hogy könnyen leprogramozható általánoscélú számítógépeken, így megfelelően univerzális a hozzáférhetősége.

A végeselem-módszer egyszerű alapelve

A végeselem-módszer arról kapta a nevét, hogy a vizsgált végtelenhalmazt véges sok nagyjából szabályos részre daraboljuk fel, és a számítógép ezekre a diszkrét elemekkel végzi el a számítást. Néhány egyszerű, de látványos példán át könnyen megérthető az emögött húzódó elv: középiskolában az integrálás bevezetésekor a függvény alatti területet végtelen sok téglalappal fedjük le, és ezzel pontos eredményre jutunk. Az alábbi ábrán láthatjuk, azonban, véges sok téglalap segítségével is egészen pontos közelítést tudunk adni, és mint a második ábrán láthatjuk, a felbontás okos megvalósításával akár jobb közelítést is elérhetünk effektívebb erőforráshasználattal.



Hasonlóan szemléletes példa a kör területének meghatározása a π ismerete nélkül, avagy a π közelítése ezen módszer alapján. Mint az ábrán láthatjuk, a kör területét 'egyszerű' alakzatokkal, téglalapokkal töltjük ki, melyek területét könnyen tudjuk számítani.

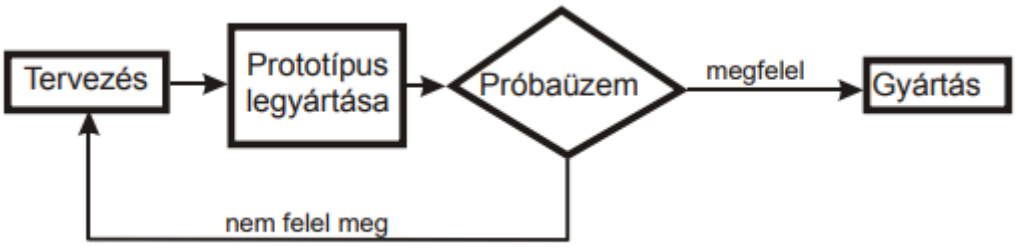


Kör területének közelítése téglalapokkal

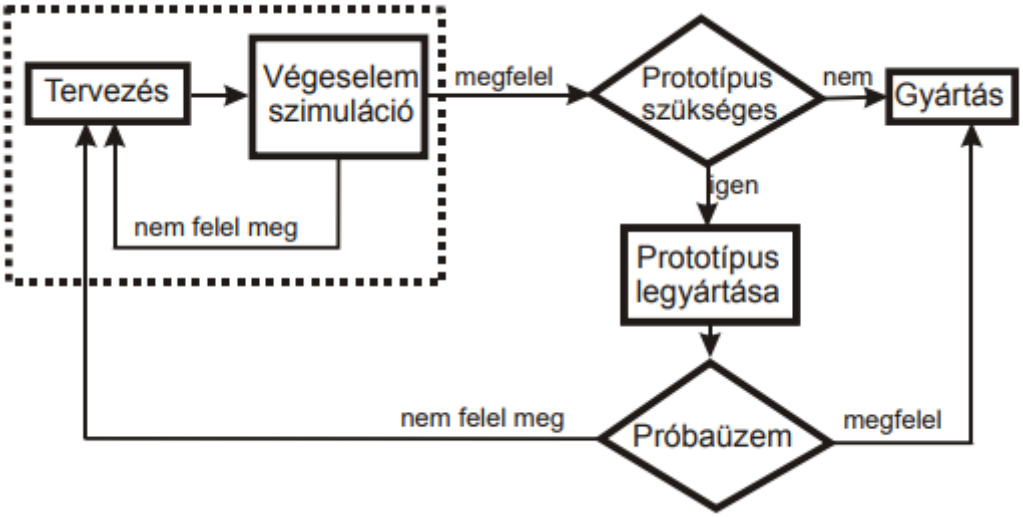
Hogy értsük, mennyire jó közelítést érhetünk el ezzel a módszerrel: Időszámításunk szerint kb. 480-ban egy kínai mérnök Tszu Csang Csik (feltételezhetően) a fenti téglalapokra bontás módszerével meghatározta, hogy a π 3,1415926 és 3,1415927 közé esik. Ha az ember belegondol, hogy milyen infrastruktával érte ezt el, nehezen juthat más megállapításra, hogy elképesztő teljesítményről beszélünk.

A VEM ma

A végeselem-módszer ilyen szintű elterjedése számottevő változást hozott a klasszikus gyártási folyamatokba. Míg a korábbi módszertan során a fejlesztési költségek nagyon jelentős részét az újabb és újabb prototípusok, és azok tesztelése tette ki, amely költségeket igen jól lefaraghatja a nagypontosságú, könnyen használható szimuláció, hiszen jóval kevesebb prototípus legyártására van szükség, így ennek közvetett és közvetlen költségei is visszaszorulunak.



A klasszikus gyártási modell folyamatábrája



A végeselmes szimulációval segített gyártási modell

Szimulációs szoftverek azonban nem csak a prototípusgyártás szükségességét tudják csökkenteni, hanem magának a gyártási technológia kialakításában is segíthetnek: fröccsöntési, kovácsolási, mélyhúzási, litográfiai, maratási, stb folyamatokat szimulálhatunk, és vehetünk észre olyan technológiai buktatókat amik egy elkészült gyártási sor esetén hatalmas újra tervezési és újraépítési költségeket jelentenének. A fentiek fényében nem meglepő, hogy mára a végeselem-módszer megkerülhetetlen alapeszköze számtalan mérnök munkájának. Sokszor olyan szinten van integrálva bizonyos szoftvercsomagokba, hogy intuitíven használható, nem szükséges a módszer mélyebb ismerete, értése. Ennek a projektnek a célja ugyanakkor a módszer átfogó megismerése, és megértése, illetve első lépésként mintegy „bevezetés a végeselem-módszer”-be tanulási folyamat elindítása.

A végeselemmódszerről

Dióhéjban

Nagy vonalakban a végeselem-módszer bemutatását a következő formában szokás megadni: vegyünk példának egy u függvényt, amely egy parciális differenciálegyenletrendszer egy eleme. Ezt a függvényt közelítjük u_h-val, amely alapfüggvények lineáris kombinációjaként adódik.

$$u \approx u_h$$

$$u_h = \sum_i u_i \psi_i$$

Ahol ψ_i az alapfüggvényeket, míg u_i az u függvényt közelítő u_h megfelelő együtthatója. A végeselem-módszer legnagyobb előnyei, hogy felhasználás függően igen nagy szabadságot ad mind az alapfüggvények meghatározásában, mind a tér diszkretizálásában. Ez azt jelenti, hogy akár két azonos fizikai folyamat leírására is különböző alapokat használhatunk, ha ezek jobban megfelelnek például a problémák geometriájának, valamint nagyfokú erőforrásoptimalizációra ad lehetőséget, amely bármilyen -ipari környezetbe helyezett-program(csomag) esetén kiemelkedően fontos szempont.

Hálózás

Bár a végeselem-módszer rendkívüli szerteágazó, és mindeközben egyszerre speciális és univerzális eszköz, mintegy mínusz első lépéseként szükséges a vizsgált tér megfelelő hálózása, az ún.: geometriai finitizálás.

A féléves munkám során a megfelelő háló elkészítésének módszereit, lehetőségeit vizsgáltam, és egy nagyobb projekt első lépéseként egy egyszerű hálózó C++-os implementációját készítettem el. Ennek a programnak nem volt célja, hogy innovatív, vagy tökéletes legyen. Későbbi munkám során alkalmas lehet az összetettebb, adaptív hálózókkal való összevetés során szemléltetni a hálózó minőségének a hatását a numerikus módszer kimenetére.

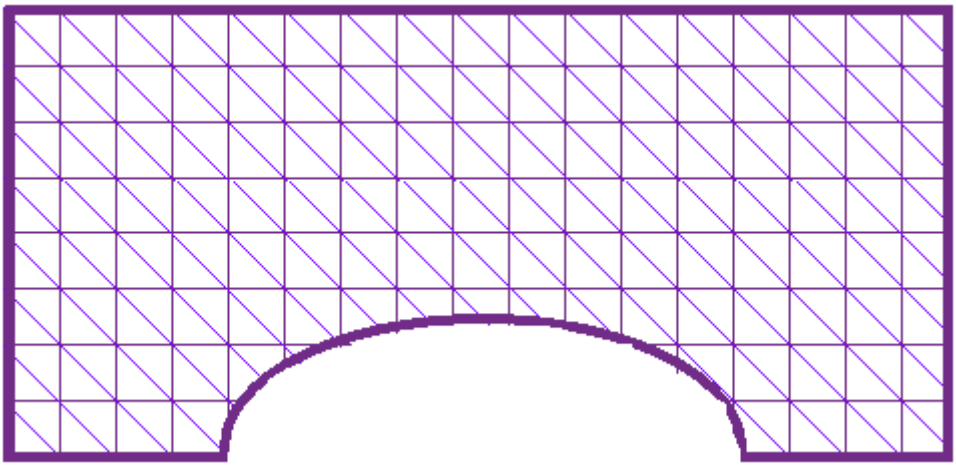
A megfelelő háló

Egy jó hálót a tudományokból jól ismert módszerrel lehet meghatározni: több, egymásnak ellentmondó szempont közelítése, az arany középút megtalálása a cél. Jelen esetben az egyik ilyen szempont, hogy egy háromszögháló alapegységei lehetőleg minél inkább hasonlítsanak egy egyenlő oldalú háromszögre (illetve ennek analógiája nem háromszög alapú síkhálóra, illetve háromdimenziós alkalmazásokra), méreteiben legyenek egymáshoz hasonlóak, és adjon minél pontosabb képet, ami igen sűrű ponthálózatot jelentene. Ezzel ellentmondó igény, hogy minél jobb legyen az erőforrásfelhasználása, mely éppen ritkább ponthálót jelentene, és természetesen a közelítés pontosságának a romlását.

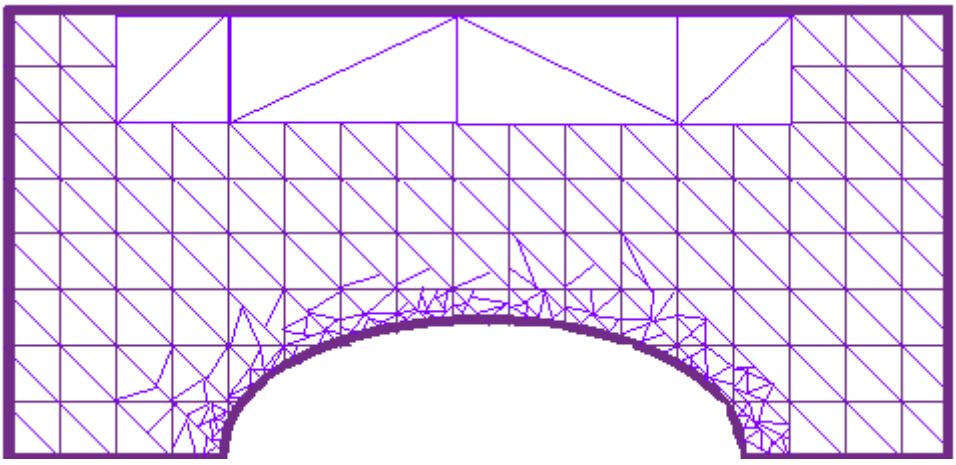
A fenti elvek példája

A fenti gondolatok szemléltetésére készültek az alábbi ábrák, melyek egy kétdimenziós híd háromszög alapú hálózásának két verziójára mutat rá. A hídon egy autó (tömegpont) halad végig, a cél a híd pontjaira ható erők és torzító tényezők vizsgálata volna.

Egyértelműen egyszerű megoldás egy normál háromszöghálót használni, melynek elemei egymáshoz igen hasonlóak (bár nem egyenlő oldalú, csak egyenlő szárú háromszögek).



A problémát ennél a hálónál az jelenti, hogy a híd minden egyes szeletéről pontosan ugyanannyi információt fog hordozni a későbbi analízisünk, és igen nagy redundanciával fogunk adatokat kapni, mivel a hídnak bizonyos részen nagyon hasonló erőket tapasztalhatunk, míg -például a görbület közelében- igen kis távolságra is jelentős különbség van a ható erőkben. Ez nem effektív erőforráshasználat, ugyanis akár ugyanennyi háromszögel is jóval pontosabb eredményt érhetünk el, tehát nem az a megoldás, hogy sűrűbb mintavételezést használunk az egész területen (sűrűbb pontháló). A megoldás az lehet, ha a probléma ismertében különböző pontsűrűséget használunk a híd különböző részeinél. Ugyanezen gondolatmenet vezetett a bevezetőben látható függvény alatti terület második ábrájához is.



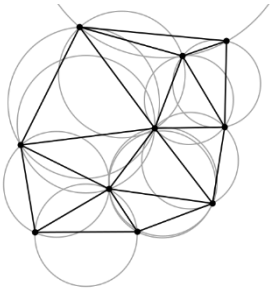
Természetesen ez az ábra főként szemléltetési célú, az átmenetek nem folyamatosak, és így feltehetően nem is ideálisak, ám céljának megfelelő. Egy ilyen hálóval sokkal több információt nyerhetünk azokon a területeken, ahol erre szükségünk van, ugyanakkor az erőforrásainkat is kevésbé terheljük ott, ahol erre nincsen szükség.

Az ideális megoldás ugyanakkor még egy lépéssel túlmegy a fenti gondolatmeneten: nem feltétlenül szükséges egybefüggő háló készítése, nagyon jól optimalizálható volna ez a feladat, ha mintegy lokális doménekre határoznánk meg ponthalmazokat, és ezekre egymástól függetlenül végeznénk el a későbbiekben ismertetett háromszögelést.

Delaunay-háromszögelés

Bár több igen hatékony hálózó algoritmus létezik, melyek különböző összetettségűek lehetnek (lásd mikro vagy makro szintű háló készítése), azonban munkám során én a Delaunay-háromszögeléssel ismerkedtem meg.

A geometriában és a számítástudományban egy adott P ponthalmaz Delaunay-háromszögelése egy olyan egyenes szakaszokból álló vonalhálózat, aminek sokszögtartományai köré írt gömbjei csak határukon tartalmazzák a P ponthalmaz pontjait.



A fenti definíció, és szemléltető ábra a magyar wikipédia 'Delaunay-háromszögelés' szócikkéből származik

A Delaunay-háromszögelés nagy előnye, hogy ötvözi a fentebb kifejtett igényeinket, és nagyjából az elvárt aranyközéputat valósítja meg. Ehhez hozzá kell tenni, hogy a módszer egy előre definiált ponthalmazon hozza létre a háromszöghálózatát, tehát adott esetben az első ábrához hasonló mintát eredményezne, azonban a módszer nem szabályos pontmintán is relatíve hasonló háromszögeket generál, maximalizálja a háromszögek minimális szögét, ezzel kerüli a túlzottan „vékony" háromszögeket.

A fentebb említett gyakorlati alkalmazás során azonban érdemes lokálisan használni ezt az eszközt, és a megfelelő helyeken pedig eltekinteni a használtatától, és más irányban elindulni. (Például a határvonalak felbontása és ezekre egyenlő oldalú háromszögek illesztése).

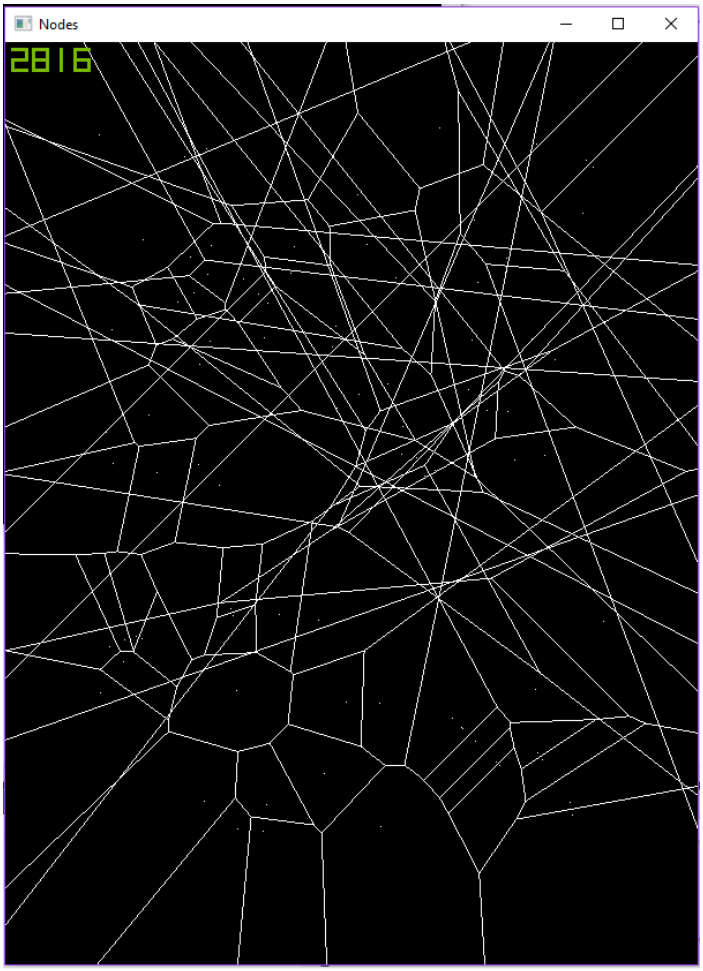
Fontos érdekesség, hogy a Delaunay-háromszöghálózat az adott ponthalmaz Voronoi-diagrammjának a duálisa, ugyanis munkám során először ennek megfelelően indultam el.

Delaunay-háromszögelés Voronoi-diagrammon keresztül

A félév elején egy olyan hálózó elkészítését tűztem ki célul, amely egy adott ponthalmaznak elkészíti a Voronoi-diagrammját, majd ennek képezve a duálisát.

A Voronoi-ábra elkészítésére Steven Fortune „sweeping line" algoritmusát szemeltem ki, mivel egyrészt igen látványos lehet, másrészt az ábra elkészítésének idejét is nagyban lecsökkentheti más algoritmusokhoz képest.

Az interneten egy ukrán-cseh programozó C++-os implementációja szabadon felhasználható volt, így a projektembe beemelve azt nekiálltam az adatok megjelenítésének, és a működés tesztelésének. Ugyan kis számú pont esetén kielégítően működött a program, voltak bizonyos esetek amelyeknél a program nem megfelelő kimenetet adott, és nagy elemszámú pontlista esetén szinte kivétel nélkül előjöttek ezek a hibák. Az alábbi ábrán láthatjuk, ahogyan egy jellegre helyes Voronoi-diagrammot oda nem illő egyenesek metszik, vagy éppen teljesen értelmezhetetlen metszési régiókat is felfedezhetünk.



Természetesen megpróbáltam a hiba előfordulását, és ezzel a probléma forrásának meghatározását, amely végül az egyenesek definíciójában volt. A programozó a klasszikus $y(x) = m * x + c$ módon definiálta az egyeneseket meghatározó függvényeket, azonban nem különböztette meg azokat az eseteket, ha y vagy x konstans volt. Ekkor két lehetőségem volt, vagy megpróbálom kijavítani ezt a figyelmetlenséget, vagy nem használom a kódot. Mivel túlzottan beágyazódott a fenti függvény használata a felhasznált osztályokba és függvényekbe, így némi próbálkozás után lemondtam róla, hogy ki tudnám javítani, és más irányba indulva újrakezdtem a projektemet.

Közvetlen Delaunay-háromszögelés

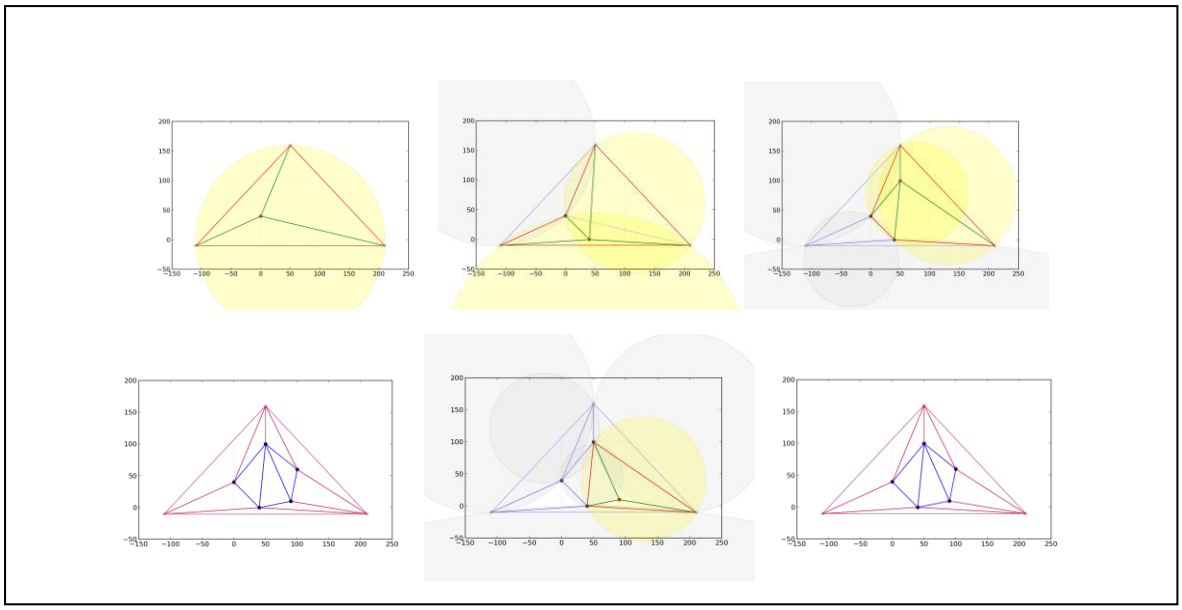
A korábbi kudarc miatt újragondoltam a hálózó működését, és a pontlistából való közvetlen Delaunay-háromszögelés mellett döntöttem. Ennek megvalósítására a Bowyer-Watson algoritmust választottam.

A Bowyer-Watson algoritmus

```
function BowyerWatson (pointList)
      // pointList is a set of coordinates defining the points to be
triangulated
      triangulation := empty triangle mesh data structure
      add super-triangle to triangulation // must be large enough to
completely contain all the points in pointList
      for each point in pointList do // add all the points one at a time to
the triangulation
         badTriangles := empty set
         for each triangle in triangulation do // first find all the triangles
that are no longer valid due to the insertion
            if point is inside circumcircle of triangle
               add triangle to badTriangles
         polygon := empty set
         for each triangle in badTriangles do // find the boundary of the
polygonal hole
            for each edge in triangle do
               if edge is not shared by any other triangles in badTriangles
                  add edge to polygon
         for each triangle in badTriangles do // remove them from the data
structure
            remove triangle from triangulation
         for each edge in polygon do // re-triangulate the polygonal hole
            newTri := form a triangle from edge to point
            add newTri to triangulation
      for each triangle in triangulation // done inserting points, now clean
up
         if triangle contains a vertex from original super-triangle
            remove triangle from triangulation
      return triangulation
```

A Bowyer-Watson algoritmus pszeudó-kódja, forrás: wikipedia

Láthatjuk, hogy az algoritmus alapelve, hogy pontonként újra és újra elkészíti a háromszöghálózatot: a nem megfelelő háromszögeket eltávolítja, és megfelelőekkel tölti ki azok területét.



A Bowyer-Watson algoritmus szemléltetése (forrás: wikipedia)

MeshMerise

A programom második verziója a Bowyer-Watson algoritmus C++-os implementációját tartalmazza, amely githubon elérhető, és szabadon felhasználható.

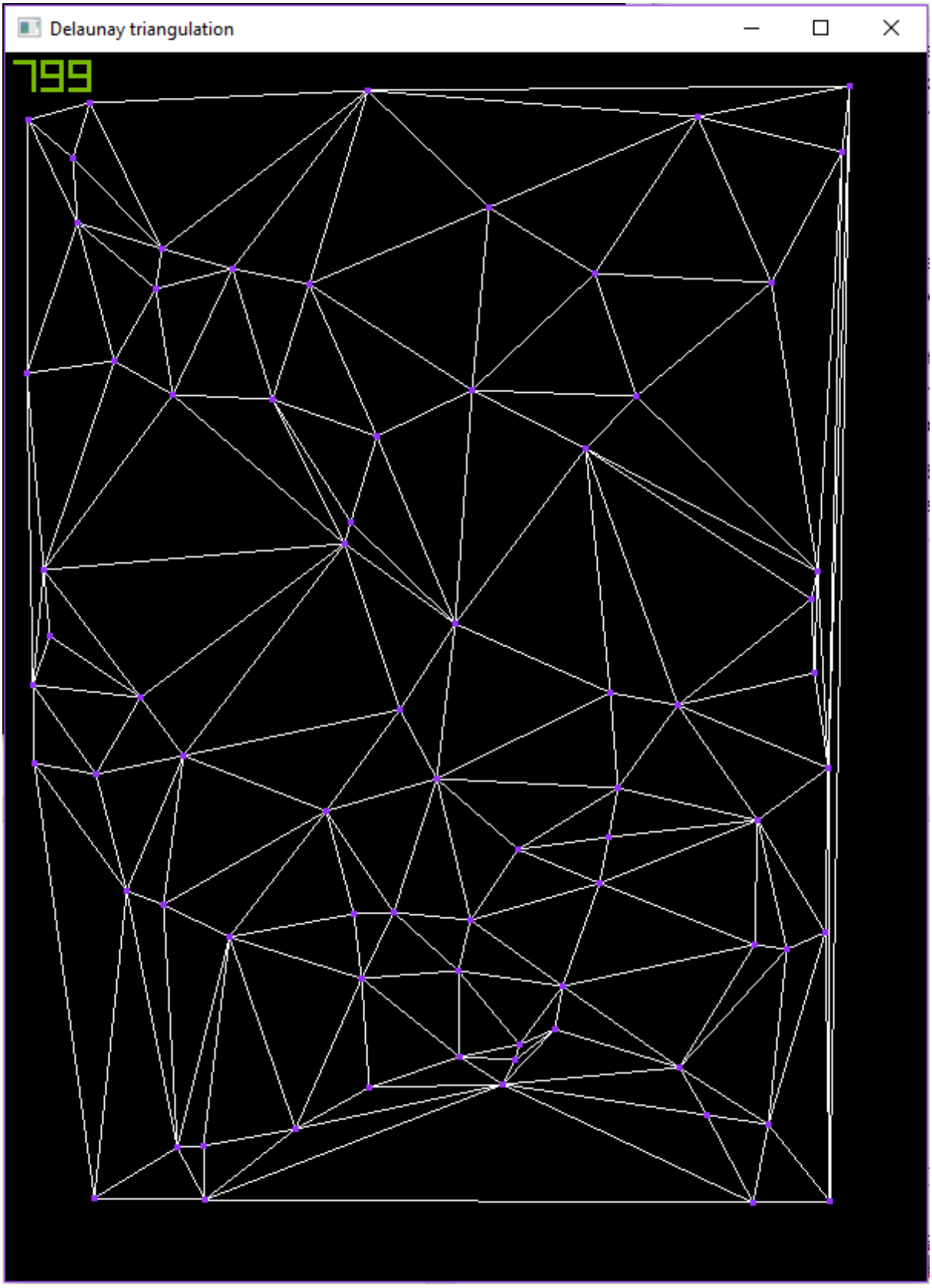
A MeshMerise nevű projektem jelen állapotában a bemenő pontlistát:

1) le tudja generálni
   - véletlenszerű ponthalmazként
   - szabályos ponthalmazként
   - szabályos ponthalmaz torzításaként
2) be tudja emelni .txt file-ból, amennyiben annak megfelelő a formátuma:
   (x koordináta) (szóköz) (y koordináta) (sorvége)

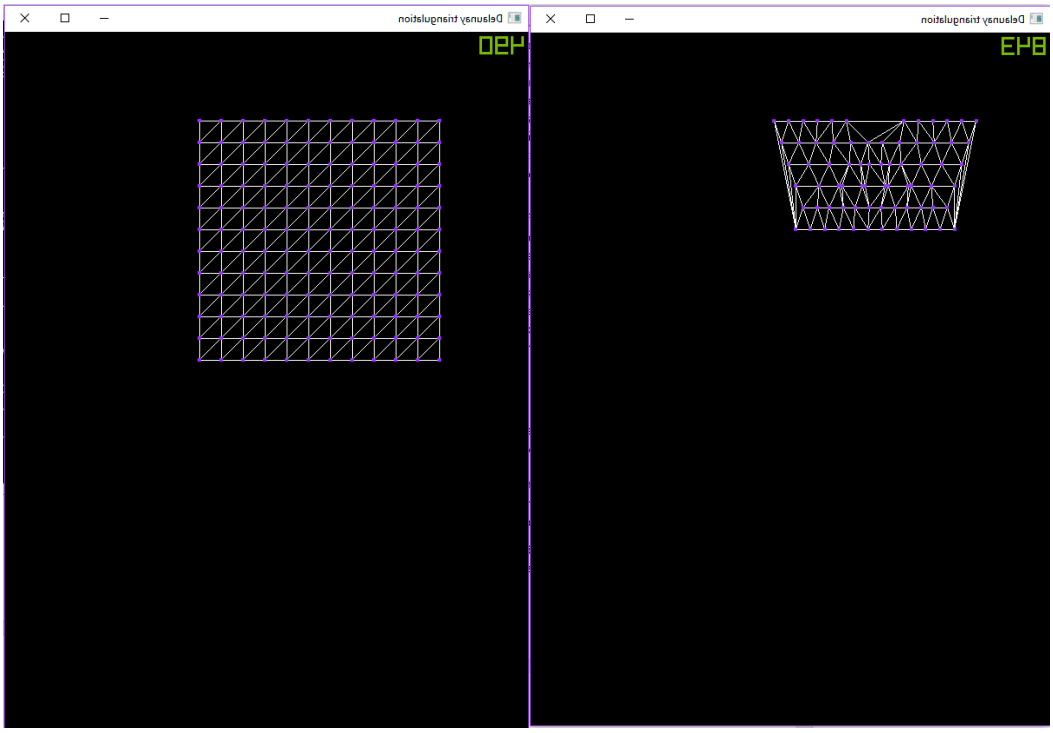
A bemenő ponthalmaztól származtatásától függetlenül elvégzi a Delaunay-háromszögelést, és ennek eredményét egyrészt egy output .txt file-ba írja (további lehetséges felhasználásra), illetve az SFML grafikus library segítségével meg is jeleníti az elkészült hálózatot.

A projekt az Bl4ckb0ne implementációjában található **Vector2**, **Edge**, **Triangle** és **Delaunay** osztályokat használja, a bemenő pontlistát **[x koordináta] [szóköz] [y koordináta] [sorvége]** formátumban várja (illetve generálja), az elkészült háromszöglista elemeit pedig a három csúcs fenti formátumú megjelenítése utáni **[sorvége]**-vel elválasztva adja meg.

Képernyőképek a programból



Véletlenszerűen generált ponthalmaz



Szabályos és torzított ponthálózat

Jövőbeli fejlesztési irányok

A projekt továbbfejlesztésére igen sok lehetséges irányt látok: egyrészt a korábban említett lehetőség, komoly, adaptív hálózó elkészítése, és ezek működésének összevetése, illetve célszerű lenne a kimeneti háromszögháló felhasználásával a végeselem-módszer numerikus megoldásának a vizsgálatára, akár ezek leprogramozásával, akár meglévő függvénykönyvtárak működésének a vizsgálatával.

Felhasznált források

Könyvek

- The Finite Element Method: its Basis and Fundamentals, O.C. Zienkiewicz, R.L. Taylor and J.Z. Zhu
- Végeselem-módszer (Egyetemi tananyag), Moharos István, Oldal István és Szekrényes András, szerk.: Kovács Ádám

Internet

- www.comsol.com
- www.youtube.com
- www.wikipedia.com
- www.cppreference.com

- www.stackoverflow.com
- www.reddit.com
- www.github.com
  https://github.com/Bl4ckb0ne/delaunay-triangulation
- http://blog.mmacklin.com
- http://blog.ivank.net/