T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ

MÜHENDİSLİK FAKÜLTESİ

STAJ DEFTERİ

Öğrencinin;

Adı, Soyadı	**Mahmut Akkuş**
Numarası	**131024020**
Bölümü	**Elektronik Mühendisliği**
Staj Yaptığı Yer	
Staj Tarihleri	**22/01/2018 – 16/02/2018**

YAPILAN İŞİN	
TARİHİ: 22/01/2018	**KAPSAMI: Elektrik alan hacim integral denkleminin ayrıklaştırılması ve moment metodu ile çözümü.**

Stajımın ilk gününde, önce Doç. Dr. Fatih Dikmen ile birlikte staj boyunca hangi problemler üzerinde çalışabileceğimizi konuştuk. Çeşit çeşit problemler arasından ilk olarak elektrik alan hacim integral denkleminin moment metoduyla çözümüne başlama kararı aldık. Kendisinden konuyu ayrıntılı bir biçimde öğrendikten ve gerekli referans kaynakları edindikten sonra çalışmalara başladım.

1) Çalışmanın özeti:

Elektrik alan hacim integral denklemi şu şekilde ifade edilebilir:

$$\frac{\boldsymbol{D}(\boldsymbol{r})}{\epsilon_0\epsilon_r(\boldsymbol{r})} - jk_0\eta_0 \iiint_V j\omega\kappa(\boldsymbol{r}')\boldsymbol{D}(\boldsymbol{r}')G(\boldsymbol{r},\boldsymbol{r}')dV' + \frac{j\eta_0}{k_0}\nabla \iiint_V j\omega\kappa(\boldsymbol{r}')\boldsymbol{D}(\boldsymbol{r}')\cdot\nabla' G(\boldsymbol{r},\boldsymbol{r}')dV' = \boldsymbol{E}^{inc}(\boldsymbol{r})$$

Bu denklemde, elektrik deplasman alanı $\boldsymbol{D}(\boldsymbol{r})$ bilinmeyendir ve denklemi çözmek suretiyle elde edilmek istenen değerdir. Cisim üzerindeki elektrik deplasman alanı bulunduğunda, buna bağlı olarak cisimdeki akım yoğunluğu ve saçılan alan gibi diğer değerler de hesaplanabilir.

Denklemin çözümü, elbette analitik olarak mümkün değildir. Bu sebeple incelenen cismin hacmi, sonlu sayıda tetrahedrondan oluşan bir tetrahedral ağ yapısı şeklinde modellenir. Bu şekilde bir modelleme, bilgisayar destekli tasarım yazılımlarıyla yapılabilir.

Tetrahedral ağ yapısı şeklinde modellenmiş bir cisim için, $\boldsymbol{D}(\boldsymbol{r})$ şu şekilde ifade edilebilir:

$$j\omega\boldsymbol{D}(\boldsymbol{r}) = \sum_{n=1}^{N} a_n \boldsymbol{f}_n(\boldsymbol{r})$$

Buradaki $\boldsymbol{f}_n(\boldsymbol{r})$ fonksiyonları, cismin modelindeki her bir üçgensel yüz için var olan, ve bu yüze komşu olan tetrahedronlar içinde 0'dan farklı değer alan SWG fonksiyonlarıdır.

$$\boldsymbol{f}_n(\boldsymbol{r}) = \begin{cases} \dfrac{A_n}{3V_{n1}}(\boldsymbol{r}-\boldsymbol{r}_{n1}), & \boldsymbol{r}\in D_{n1} \\ \dfrac{A_n}{3V_{n2}}(\boldsymbol{r}_{n2}-\boldsymbol{r}), & \boldsymbol{r}\in D_{n2} \\ 0, & \boldsymbol{r}\notin D_{n1}\cup D_{n2} \end{cases}$$

a_n ise bilinmeyen katsayılardır. Böylece denklem ayrıklaştırılmış, sürekli bir vektör alanı olan bilinmeyen, sonlu sayıdaki N adet katsayıya indirgenmiştir.

Bu aşamada, N adet bilinmeyen katsayıyı bulmak için devreye moment metodu girer.

$$\iiint_V \{\boldsymbol{LHS}\}(\boldsymbol{r})\cdot\boldsymbol{f}_m(\boldsymbol{r})dV = \iiint_V \boldsymbol{E}^{inc}(\boldsymbol{r})\cdot\boldsymbol{f}_m(\boldsymbol{r})dV$$

$\{\boldsymbol{LHS}\}(\boldsymbol{r})$ burada denklemin sol yanıdır. Denklemin her bir SWG temel fonksiyonuyla bu şekilde iç çarpımının alınması suretiyle, N adet birbirinden farklı lineer denklem elde edilir.

$$\boldsymbol{Za} = \boldsymbol{w}$$

Ortaya çıkan bu N bilinmeyenli ve N denklemli lineer sistem çözülerek, a_n katsayılarının değerleri elde edilir. Böylece $\boldsymbol{D}(\boldsymbol{r})$ nümerik olarak hesaplanmış olur.

Elektrik alan hacim integral denkleminin nümerik olarak çözümü, yüzey integral formülasyonuna kıyasla çok daha fazla işlem gerektirir, ancak elektriksel iç yapısı homojen olmayan cisimlerin elektromanyetik analizini mümkün kılar.

Stajımı bu firmada yaptım. **Staj Yapanın İmzası**	**Staj Yeri Yetkilisinin** **Adı, Soyadı, İmzası, Firma Kaşesi**

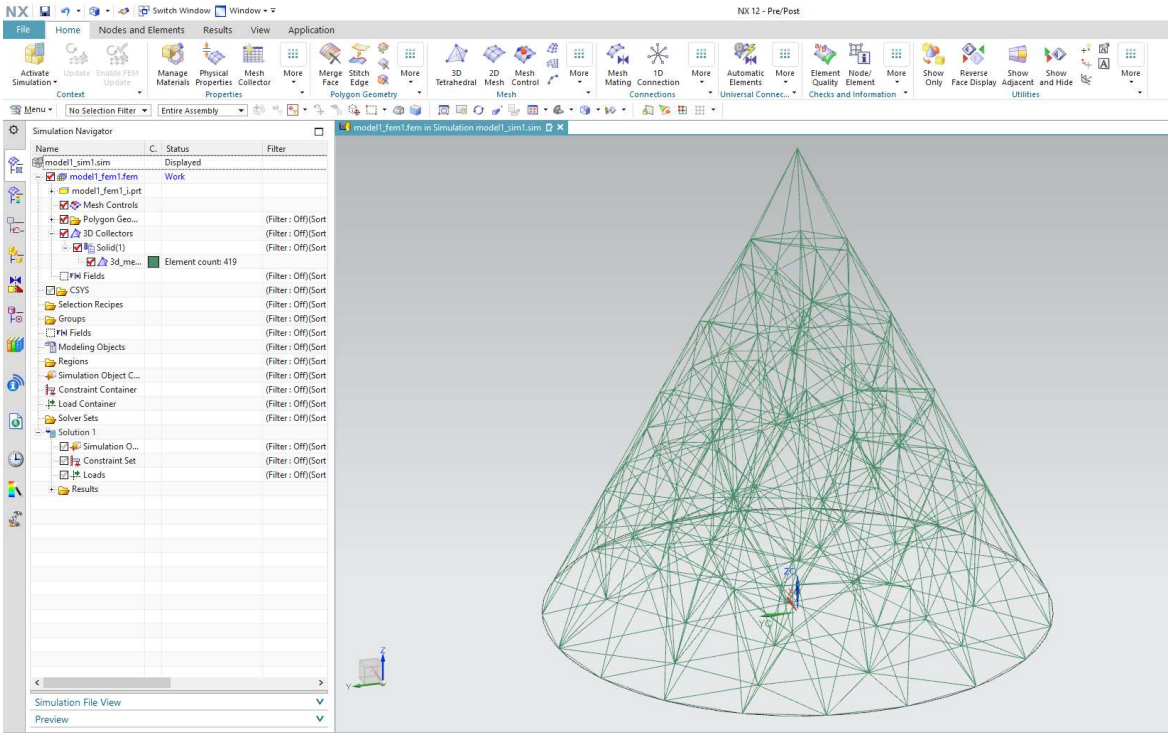
YAPILAN İŞİN	
TARİHİ: 23/01/2018	**KAPSAMI: Siemens NX ile modelleme.**

2) Siemens NX ile modelleme:

Staj boyunca incelenen cisimlerin tetrahedral elemanlarla ifade edilen modelleri, Siemens NX programı yardımıyla elde edilmiştir.

Siemens NX, yaygın olarak kullanılan ve oldukça kapsamlı olan bir bilgisayar destekli tasarım / modelleme / mühendislik programıdır.

Programın ucu bucağı olmayan işlevsellikleri arasında özellikle en önemlisi, programda yapılan 3 boyutlu modellerin düzgün bir şekilde belirli boyutlardaki tetrahedronlara bölünebilmesidir.



Siemens NX arayüzü ve 419 tetrahedral parçaya ayrılmış bir koni modeli.

Siemens NX ile üretilen tetrahedral yapı, .unv dosya formatında dışarı aktraılmakta ve programlar tarafından okunulabilmektedir.

.unv dosya formatında, yapıdaki her noktanın koordinatları, ve her bir tetrahedronu oluşturan 4 noktanın indisleri bulunmaktadır. Yapıdaki her bir nokta birden fazla tetrahedrona ait olduğu için, nokta sayısı tetrahedron sayısından çok daha azdır. Bu yüzden tetrahedronların her bir noktasının koordinatlarının bizzat kaydedilmesi yerine, noktaların ayrı kaydedilip tetrahedronların indislerle belirtilmesi dosya boyutunun gereksiz yere büyümesini engellemekte; aynı zamanda yapı hakkında hangi üçgensel yüzün hangi tetrahedrona ait olduğu gibi bir takım bilgilerin elde edilmesini kolaylaştırmaktadır.

Stajımı bu firmada yaptım. **Staj Yapanın İmzası**	**Staj Yeri Yetkilisinin** **Adı, Soyadı, İmzası, Firma Kaşesi**

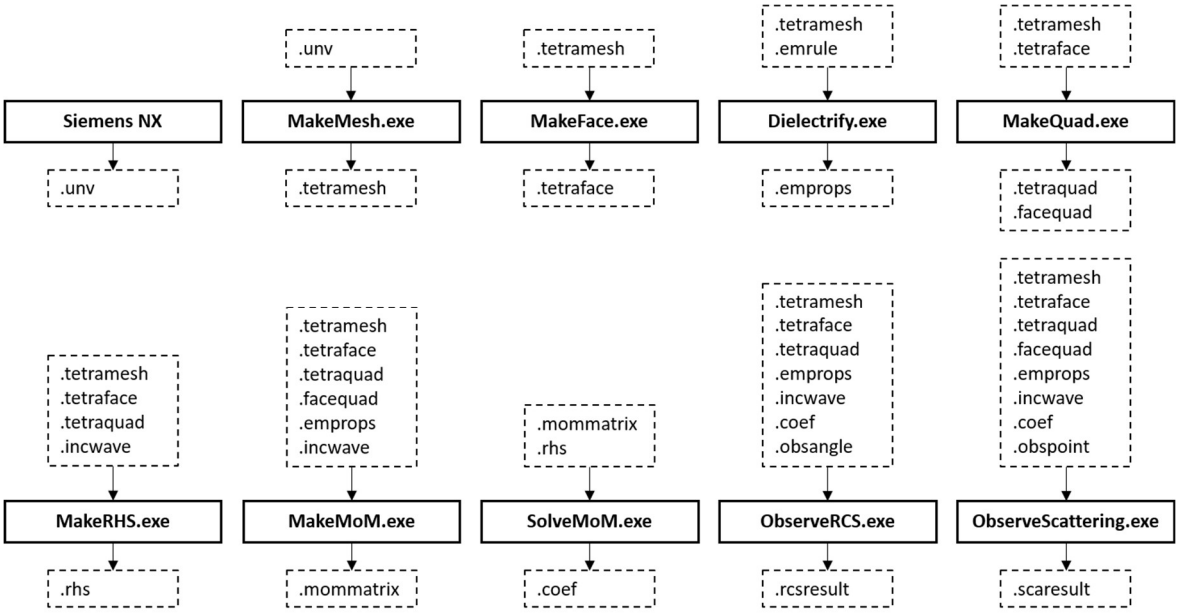
YAPILAN İŞİN	
TARİHİ: 24/01/2018	KAPSAMI: Çözüm için yazılacak olan programın akışı.

3) Çözüm için yazılacak olan programın akışı:

Çozüm için yazılacak olan programın, performans açısından kritik parçalarının C++ programcıkları olarak yazılması, programcıklar arasındaki veri akışının işe çeşitli dosya formatlarıyla ifade edilmesi kararlaştırılmıştır.

MATLAB ile bu programcıklar çalıştırılacak, aradaki veri akışı sağlanacak, ve elde edilen sonuçlar görüntülünecektir.

Programcıklar ve aradaki akış şeması şekildeki gibidir:



Programın tamamının MATLAB'da yazılmamasının sebebi, en basit hesaplar için bile aşırı miktarda hesap yapılmasının gerekmesidir. Tetrahedral yapıdaki her bir elemanın kenar uzunluğu, gelen dalganın dalga boyunun en fazla %10'u uzunluğunda olması gerekmektedir. Bu sepeble dalga boyu 2 kat küçükdüğünde, tetrahedral elemanlar 3 boyutlu olduklarından, yapıdaki tetrahedron miktarı yaklaşık 8 kat artmaktadır. Buna bağlı olarak üçgensel yüzey miktarı da yaklaşık 8 kat artmaktadır. Moment metodunda çözülen matrisin eleman sayısı, üçgensel yüzey miktarının karesi kadar olduğundan, 64 kat artmaktadır!

C++ ile yazılan programlar, direkt olarak makine koduna derlendikleri için çok daha hızlı çalışmaktadırlar. Özellikle döngüler ve bir takım veri yapıları MATLAB'da çok yavaş işlemektedirler. Daha önceki çalışmalarda, C++ programlarının MATLAB muadillerinden 1000 kata kadar daha hızlı çalıştığına bizzat şahit olunmuştur.

Yerine göre, C++'ın paralel hesaplama yapmada sunduğu olanaklar da MATLAB'dan daha fazladır. MATLAB'da GPU hızlandırması için yalnızca CUDA destekli Nvidia GPU'lar kullanılabilmektedir.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN	
TARİHİ: 25/01/2018	**KAPSAMI: Veri akışı için kullanılan dosya yapıları.**

4) Veri akışı için kullanılan dosya yapıları:

Dosya uzantısı: .unv
Veri saklama tipi: ASCII
Açıklama: Siemens NX çıktısı olan dosyadır. Tetrahedral yapının nokta koordinatlarını ve tetrahedron indislerini barındırır. İçindeki sayısal veriler ASCII karakterleriyle kaydedilmiştir. İlerleyen kısımlarda ayrıntılı olarak bahsedilecektir veri dizilimi hakkında.

Dosya uzantısı: .emrule
Veri saklama tipi: ASCII
Veri dizilimi: (her eleman satır)
[0]: ruleType
[1..N]: ruleData
Açıklama: Tetrahedronlara dielektrik sabiti atarken kullanılacak olan kural dosyasıdır. "ruleType" değişkeni, "Constant", "RadialLinear", "RadialLayered" olabilir. Kural tipine göre data miktarı değişir.

Dosya uzantısı: .incwave
Veri saklama tipi: ASCII
Veri dizilimi: (her eleman satır)
[0]: waveType
[1..N]: waveData
Açıklama: Cisme gelen dalganın tanımlandığı dosya tipidir. "waveType" değişkeni, "Plane", "Dipole" vs. olabilir. Dalga tipine göre data miktarı değişir.

Dosya uzantısı: .tetramesh
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: vertexCount (uint64)
[1]: tetraCount (uint64)
[2..A]: vertexData (3*[0]*double)
[B..C]: tetraVertexIndex (4*[1]*uint64)
Açıklama: Programcıklar arasında kullanılan tetrahedral model formatıdır.

Dosya uzantısı: .tetraface
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: faceCount (uint64)
[1]: tetraCount (uint64)
[2..A]: faceVertexIndex (3*[0]*uint64)
[B..C]: faceTetraIndex (2*[0]*uint64)
[D..E]: tetraFaceIndex (4*[1]*uint64)
Açıklama: Tetrahedral modeldeki üçgensel yüzlerin nokta indisleri ile yüzey-tetrahedron ilişkilerinin saklandığı dosyadır.

Dosya uzantısı: .emprop
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: tetracount (uint64)
[1..A]: emPropData (2*[0]*double)
Açıklama: Her bir tetrahedronun dielektrik sabitlerinin depolandığı dosya tipi.

Dosya uzantısı: .tetraquad
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: tetraCount (uint64)
[1]: quadCount (uint64)
[2..A]: quadData (4*[0]*[1]*double)
Açıklama: Tetrahedronlar üzerinden nümerik integral alınırken kullanılan quadrature noktaları ve ağırlıkları.

Dosya uzantısı: .facequad
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: faceCount (uint64)
[1]: quadCount (uint64)
[2..A]: quadData (4*[0]*[1]*double)
Açıklama: Üçgensel yüzler üzerinden nümerik integral alınırken kullanılan quadrature noktaları ve ağırlıkları.

Dosya uzantısı: .mommatrix
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: faceCount (uint64)
[1..A]: momData (2*[0]*[0]*double)
Açıklama: MoM matrisinin depolandığı dosya. Elemanlar column major olarak dizili.

Dosya uzantısı: .obsangle
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: obsCount (uint64)
[1..A]: obsData (2*[0]*double)
Açıklama: RCS analizinin yapılacağı gözlem yönlerini belirten küresel açılar.

Dosya uzantısı: .obspoint
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: obsCount (uint64)
[1..A]: obsData (3*[0]*double)
Açıklama: Saçılan elektromanyetik alanın ölçüleceği noktalar.

Dosya uzantısı: .rhs
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: faceCount (uint64)
[1..A]: rhsData (2*[0]*double)
Açıklama: MoM matris denkleminin sağ yanı.

Dosya uzantısı: .rcsresult
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: obsCount (uint64)
[1..A]: resultData ([0]*double)
Açıklama: .obsangle dosyasındaki gözlem yönlerinde ölçülen RCS değerleri.

Dosya uzantısı: .scaresult
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: obsCount (uint64)
[1..A]: resultData (6*[0]*double)
Açıklama: .obspoint dosyasındaki gözlem noktalarındaki saçılan elektrik alan değerleri.

Dosya uzantısı: .coef
Veri saklama tipi: Binary
Veri dizilimi: (her eleman 8 byte)
[0]: faceCount (uint64)
[1..A]: coefData (2*[0]*double)
Açıklama: MoM matris denkleminin çözülmesi sonucu elde edilen katsayılar.

Stajımı bu firmada yaptım. **Staj Yapanın İmzası**	**Staj Yeri Yetkilisinin** **Adı, Soyadı, İmzası, Firma Kaşesi**

YAPILAN İŞİN	
TARİHİ: 26/01/2018	**KAPSAMI: .unv dosyalarının okunması.**

5) .unv dosyalarının okunması:

Tetrahedral model için .unv dosyalarının veri dizilimi şu şekildedir:

```
### FILE START ###              -> ASCII file.
***
vertexStartStr                  -> Coordinates of vertices.
        //useless line
        {}v1x{}v1y{}v2z{}
        //useless line
        {}v2x{}v2y{}v2z{}       -> repeat for the number of vertices.
vertexEndStr
***
tetraStartStr                   -> Indices of tetrahedra vertices, starting from 1.
        //useless line
        {}1v1{}1v2{}1v3{}1v4{}
        //useless line
        {}2v1{}2v2{}2v3{}2v4{}  -> Repeat for the number of tetrahedra.
tetraEndStr
***
### FILE END ###

{}      -> denotes varying amounts of whitespace
***     -> denotes various amount of useless lines

string vertexStartStr = "  2411";
string vertexEndStr = "    -1";
string tetraStartStr = "  2412";
string tetraEndStr = "    -1";
```

.unv dosyaları, MakeMesh.exe programcığı tarafından aşağıdaki algoritma ile okunup .tetramesh formatına dönüştürülmektedirler:

```
vertexCount = 0;
tetraCount = 0;
Open .unv file;
Read line till you get “vertexStartStr”;
Read line till you get “vertexEndStr” and ++vertexCount;
vertexCount = ( vertexCount – 1 ) / 2;
Read line till you get “tetraStartStr”;
Read line till you get “tetraEndStr” and ++tetraCount;
tetraCount = ( tetraCount – 1 ) / 2;
vertexData = array[ 3 * vertexCount ];
tetraVertexIndex = array[ 4 * tetraCount ];
Set file pointer to the beginning of the file;
Read line till you get “vertexStartStr”;
For k = 0; k < vertexCount; ++k
    Read line and discard;
    lineStr = Read line, trim left and right, replace all whitesapce with a single space;
    vertexData[3*k...3*k+2] = Split lineStr at spaces, convert to double;
End
Read line till you get “tetraStartStr”;
For k = 0; k < tetraCount; ++k
    Read line and discard;
    lineStr = Read line, trim left and right, replace all whitesapce with a single space;
    tetraVertexIndex[3*k...3*k+3] = Split lineStr at spaces, convert to uint64, decrease by 1;
End
Close .unv file and create .tetramesh file;
```

Stajımı bu firmada yaptım. **Staj Yapanın İmzası**	**Staj Yeri Yetkilisinin** **Adı, Soyadı, İmzası, Firma Kaşesi**

YAPILAN İŞİN	
TARİHİ: 29/01/2018	KAPSAMI: Modeldeki üçgensel yüzlerin belirlenmesi ve tetrahedronlarla ilişkilendirilmeleri.

6) Modeldeki üçgensel yüzlerin belirlenmesi ve tetrahedronlarla ilişkilendirilmeleri:

Siemens NX ile oluşturulan .unv dosyalarında yalnızca tetrahedronlar belirtilmektedir. Ancak bizim hesaplarımızda ek olarak modeldeki her bir üçgensel yüz ve bu yüzlerin hangi tetrahedronlara ait olduğu bilgileri de gerekmektedir. Bu yüzden bu bilgileri oluşturan bir algoritmanın oluşturulması gerekmiştir.

Aşağıdaki algoritma, MakeFace.exe programcığı tarafından kullanılarak, .tetramesh dosyalarından .tetraface dosyaları üretilir:

```
Read .tetramesh file.
faceV1,V2,V3 = array[ 4 * tetraCount ];
faceT1,T2 = array[ 4 * tetraCount ];
tetraF1,F2,F3,F4 = array[ 4 * tetraCount ];
faceMap = map<string,uint>[];
faceCount=0;
For each tetra
    Sort tetra indices ascending;
    Face indices are (v0,v1,v2), (v0,v1,v3), (v0,v2,v3), (v1,v2,v3);
    Directly reinterpreting those indices as strings will create a unique hash for each face;
    For each face of the tetra
        hash = reinterpret face indices as 24 byte strings;
        if faceMap[ hash ] exists
            faceId = faceMap[ hash ];
            faceT2[ faceId ] = tetraId;
            tetraFn[ tetraId ] = faceId;
        else
            faceMap[ hash ] = faceCount;
            faceV1,V2,V3[ faceCount ] = indices;
            faceT1[ faceCount ] = tetraId;
            faceT2[ faceCount ] = -1;
            tetraFn[ tetraId ] = faceCount;
            ++faceCount;
        end
    End
End
Resize face related arrays to fit the faceCount;
Generate .tetraface file;
```

Cisim hacminin içindeki üçgensel yüzlerin ait olduğu tetrahedron sayısı 2'dir. Yüzeydeki üçgensel yüzler ise yalnızca 1 tetrahedrona aitlerdir. Bu yüzeydeki üçgensel yüzlerin 2. tetrahedronlarını belirten indis -1'dir. Yani, 2. tetrahedron indisinin -1 olup olmadığı kontrol edilerek, üçgensel yüzün cisim yüzeyinde olup olmadığı anlaşılabilir.

Tabi indisler uint64 tipinde olduğundan, aslında -1, 2^64 – 1'e denk gelmektedir. Bu yüzden programın kullanabileceği maksimum tetrahedron miktarı aslında 2^64 değil, 2^64 -1'dir. Elbette günümüz şartlarında bu derece büyük hesaplar yapılmaz, ancak yine de bu sınırı belirtmek gerekir.

Olur da değişkenler farklı bir implementasyonda 64 değil de 32 bit olarak tanımlanırlarsa, bu sınıra erişmek mümkün olabilir.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN	
TARİHİ: 30/01/2018	KAPSAMI: Modeldeki tetrahedronlara elektrik ve manyetik geçirgenlik değerlerinin atanması.

7) Modeldeki tetrahedronlara elektrik ve manyetik geçirgenlik değerlerinin atanması:

Aslında belki de Siemens NX programında bu iş için kullanılabilecek bir işlevsellik vardır, ancak yine de ben kendim bunu programlamak istedim. .emrule dosya formatına bakacak olursanız, ilk önce hangi kurala göre bu atama yapılacak onun adı giriliyor, ardından kurala ait değerler giriliyor her satıra. Örneğin, tüm cisim üzerinde değerler sabit olacaksa .emrule dosyası şu şekilde olmalıdır:

```
Constant
#epsValue#
#muValue#
```

Değerler radyal uzaklığa bağlı olarak lineer bir şekilde değişecekse şu şekilde olabilir:

```
RadialLinear
#epsStart# // orijindeki değer
#muStart#
#epsDelta# // metre başı değişim
#muDelta#
```

Dielectrify.exe programcığında, EmRule adında bir virtual base class vardır ve çeşit çeşit atama kuralı bu class'tan türetilebilir. Şimdilik yalnızca EmRule_Constant class'ı türetilmiştir, ancak ihtiyaç halinde her türlü kural kolayca türetilebilir.

EmRule_Factory adlı bir singleton factory class'da ise tüm türetilmiş kurallar kayıtlıdır ve aşağıdaki gibi bir metod sayesinde istenilen kuralın class'ının objesi, çalışma zamanında, kural adının string'i girilerek oluşturulabilir:

```
shared_ptr< EmRule< T > > GetRule( const string& name )
{
        auto it = ruleMap_.find( name );
        if( it == ruleMap_.end() )
        {
                // Kayıtlı olmayan bir kural istendi, hata verilmeli.
        }
        else
        {
                return it->second->Clone();
        }
}
```

İşleyiş oldukça modüler ve ileriye dönüktür, kullanımı ise çok basittir:

```
TetraMeshFile< FLOAT_T > meshFile;
meshFile.Load( meshFileName );

EmRuleFile< FLOAT_T > ruleFile;
ruleFile.Load( ruleFileName );

EmPropFile< FLOAT_T > propFile;
propFile.Initialize( meshFile.tetraCount_ );

EmRuleFactory< FLOAT_T >& ruleFactory = EmRuleFactory< FLOAT_T >::GetInstance();
shared_ptr< EmRule< FLOAT_T > > emRule = ruleFactory.GetRule( ruleFile.ruleType_ );
emRule->SetRuleData( ruleFile.ruleData_ );
emRule->GenerateProps(
        meshFile.tetraCount_,
        meshFile.vertexData_,
        meshFile.tetraVertexIndex_,
        propFile.emPropData_
);

propFile.Save( propFileName );
```

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN	
TARİHİ: 31/01/2018	**KAPSAMI: Gelen elektrik alanın programdaki ifadesi.**

8) Gelen elektrik alanın programdaki ifadesi:

Elektrik ve manyetik geçirgenlik atamada kullanılan yöntemlerin aynısı burada da kullanılmıştır.

.incwave dosyası içerisinde, ilk olarak gelen dalga tipinin adı, ardından dalgaboyu, polarizasyon gibi o dalga tipini tanımlayan değerler yer alır.

Birden fazla programcık içerisinde, IncWave virtual base class, türemiş class'lar, ve factory class vardır. Bu programcıkların kod tabanları da ortaktır. Değişik değişik tipteki gelen alan türleri, modülerlik sayesinde, daha sonradan programa kolayca eklenbilecektir. Şimdilik sadece düzlem dalga tipi tanımlanmıştır.

Düzlem dalga için .incwave dosyası şu şekildedir:

```
Plane
#wavL# // Dalgaboyu.
#dirX# // Dalga ilerleme yönü.
#dirY#
#dirZ#
#polX# // İlerleme yönüne dik polarizasyon vektörü.
#polY# // Hem buna hem de ilerleme yönüne dik olan 2. polarizayon vektörü
#polZ# // otomatik olarak hesaplanacaktır.
#porX# // Faz orijini, bu noktada faz değeri başlangıç konumunda olacaktır.
#porY#
#porZ#
#pofU# // Girilen polarizasyon yönündeki başlangıç fazı.
#pofR# // Otomatik oluşturulan polarizasyon yönündeki başlangıç fazı.
#ampU# // Girilen polarizasyonun genliği.
#ampR# // Otomatik oluşturulan polarizasyonun genliği.
// bu değerlerle her türlü eliptik polarize düzlem dalga oluşturulabilir.
```

Düzlem dalgaya ait .incwave ile gelen dalga tanımlandığında, uzayın herhangi bir noktasındaki elektrik alan vektörü şu şekilde hesaplanmaktadır:

```
LUV::LuVector3c< T > FieldE_( const LUV::LuVector3< T >& obsPoint ) const override
{
        LUV::LuVector3< T > obsPointProj = LUV::ProjPlane( obsPoint, phaseOrigin_, direction_ );
        T dist = LUV::Dot( obsPoint - obsPointProj, direction_ );
        T lambdaDist = waveNumber_ * dist;

        LUV::LuVector3c< T > fieldEU = exp( j_ * ( phaseOffsetU_ - lambdaDist ) ) * ampPolEU_;
        LUV::LuVector3c< T > fieldER = exp( j_ * ( phaseOffsetR_ - lambdaDist ) ) * ampPolER_;
        LUV::LuVector3c< T > fieldE = fieldEU + fieldER;

        return fieldE;
}
```

Bir başka gelen alan olarak benzer şekilde hertzian dipole tanımlanabilir.

Aslında antin kuntin her türlü gelen alan tanımlanabilir, hatta bir başka cisimden saçılan alan hesaplandığında, o saçılan alanı veren bir gelen alan classı bile oluşturulabilir.

Stajımı bu firmada yaptım. **Staj Yapanın İmzası**	**Staj Yeri Yetkilisinin** **Adı, Soyadı, İmzası, Firma Kaşesi**

YAPILAN İŞİN	
TARİHİ: 01/02/2018	**KAPSAMI: Quadrature noktaları ve ağırlıkları.**

9) Quadrature noktaları ve ağırlıkları:

Hesapların çeşitli yerlerinde, tetrahedronlar ve yüzeyler üzerinden bir takım integrallerin alınması gerekmektedir. Bu integrallerin analitik formülasyonu olmadığından ötürü, quadrature yöntemleri ile nümerik olarak alınmak durumundadırlar.

$$\int f(\boldsymbol{r})dV \approx \sum w_n f(\boldsymbol{q}_n)$$

Bu sebepten ötürü, MakeQuad.exe tarafından, her tetrahedron için 64, her üçgensel yüz için 7 quadrature noktası belirlenmekte ve daha sonra kullanılmak üzere .tetraquad ve .facequad dosyalarına kaydedilmektedir.

Tetrahedron için quadrature noktalarını belirleyen algoritma olarak, MATLAB'ın sitesinde "TetraQuad.m" olarak paylaşılmış olan algoritma C++'a aktarılıp 64 nokta için optimize edilerek kullanıldı. Algoritmadan ziyade aslında bir takım matematiksel lineer cebir işlemlerinden ibaret sadece, en önemli kısmı şu şekildedir, olduğu gibi aktarıyorum:

```
af::array vertDiffX = af::matmul( diffMat_, vertMatAf );
af::array W = abs( af::det< FLOAT_T >( vertDiffX.rows( 1, 3 ) ) ) * Wr_;
af::array WXYZ = af::join( 1, W, af::matmul( af::join( 1, C1_, Qx_, Qy_, Qz_ ), vertDiffX ) );
transpose( WXYZ ).host( &( quadDataPtr[ idQuad ] ) );
```

Geri kalan kısmı önceden hesaplanmış bir takım değişik sabitlerdir. Üçgensel yüzlerin quadrature noktaları da buna benzer şekilde hesaplanmaktadırlar. Sadece, ArrayFire kullanılmadığı için kod miktarı daha fazla oldu:

```
LUV::LuVector< 3, T > v1( vertexDataPtr[ idv1x ], vertexDataPtr[ idv1y ], vertexDataPtr[ idv1z ] );
LUV::LuVector< 3, T > v2( vertexDataPtr[ idv2x ], vertexDataPtr[ idv2y ], vertexDataPtr[ idv2z ] );
LUV::LuVector< 3, T > v3( vertexDataPtr[ idv3x ], vertexDataPtr[ idv3y ], vertexDataPtr[ idv3z ] );
LUV::LuVector< 3, T > q[7];
q[0] = bFactor1_ * ( v1 + v2 + v3 );
q[1] = bFactor2_ * v1 + bFactor2_ * v2 + bFactor3_ * v3;
q[2] = bFactor2_ * v1 + bFactor3_ * v2 + bFactor2_ * v3;
q[3] = bFactor3_ * v1 + bFactor2_ * v2 + bFactor2_ * v3;
q[4] = bFactor4_ * v1 + bFactor4_ * v2 + bFactor5_ * v3;
q[5] = bFactor4_ * v1 + bFactor5_ * v2 + bFactor4_ * v3;
q[6] = bFactor5_ * v1 + bFactor4_ * v2 + bFactor4_ * v3;
T area = LUV::Length( LUV::Cross( v2 - v1, v3 - v1 ) ) / 2.0;
T w[7];
w[0] = wFactor1_ * area;
w[1] = wFactor2_ * area;
w[2] = wFactor2_ * area;
w[3] = wFactor2_ * area;
w[4] = wFactor3_ * area;
w[5] = wFactor3_ * area;
w[6] = wFactor3_ * area;
for( UINT_T idq = 0; idq < 7; ++idq )
{
        UINT_T idq1 = idQuad + 4 * idq;
        UINT_T idq2 = idq1 + 1;
        UINT_T idq3 = idq1 + 2;
        UINT_T idq4 = idq1 + 3;
        quadDataPtr[ idq1 ] = w[ idq ];
        quadDataPtr[ idq2 ] = q[ idq ][0];
        quadDataPtr[ idq3 ] = q[ idq ][1];
        quadDataPtr[ idq4 ] = q[ idq ][2];
}
```

Stajımı bu firmada yaptım. **Staj Yapanın İmzası**	**Staj Yeri Yetkilisinin** **Adı, Soyadı, İmzası, Firma Kaşesi**

YAPILAN İŞİN	
TARİHİ: 02/02/2018	KAPSAMI: Green fonksiyonu integrallerinden tekillik çıkarımı.

10) Green fonksiyonu integrallerinden tekillik çıkarımı:

Boş uzayın Green fonksiyonu şekildeki gibidir:

$$G(\boldsymbol{r},\boldsymbol{r}') = \frac{e^{jk|\boldsymbol{r}-\boldsymbol{r}'|}}{4\pi|\boldsymbol{r}-\boldsymbol{r}'|}$$

Ancak, fonksiyonun seri açılımındaki $\frac{1}{|\boldsymbol{r}-\boldsymbol{r}'|}$'lik bir terimden ötürü, $\boldsymbol{r}$ noktası, $\boldsymbol{r}'$ noktasına yaklaştıksa değeri sonsuza ıraksar ve bu bölgelerde alınan nümerik integraller düzgün sonuç vermezler.

Bu yüzden green fonksiyonu şu şekilde ifade edilerek, tekil kısmı ayrıştırılmış olur:

$$G(\boldsymbol{r},\boldsymbol{r}') = \frac{e^{jk|\boldsymbol{r}-\boldsymbol{r}'|}-1}{4\pi|\boldsymbol{r}-\boldsymbol{r}'|} + \frac{1}{4\pi|\boldsymbol{r}-\boldsymbol{r}'|}$$

Soldaki parça regülerdir ve $\boldsymbol{r}=\boldsymbol{r}'$ limit değeri için jk değerini alır. Sağdaki parçanın ise tetrahedron ve üçgen üzerinden integrali analitik olarak alınabilir.

Esasında, hesaplamalarımız içerisinde 3 farklı tekillik integrali vardır ve bunların analitik ifadeleri şu şekildedir:

$$\iint_S \frac{1}{4\pi|\boldsymbol{r}-\boldsymbol{r}'|}\,dS' = \frac{1}{4\pi}\sum_i \left[\left[\hat{P}_i^0\cdot\hat{u}_i\right]\left[P_i^0 A_i^3 - |d|\left[A_i^1 - A_i^2\right]\right]\right]$$

$$\iiint_V \frac{1}{4\pi|\boldsymbol{r}-\boldsymbol{r}'|}\,dV' = \frac{1}{8\pi}\sum_j \left[d_j \sum_i \left[\left[\hat{P}_{ij}^0\cdot\hat{u}_{ij}\right]\left[|d_j|\left[A_{ij}^1 - A_{ij}^2 - P_{ij}^0 A_{ij}^3\right]\right]\right]\right]$$

$$\iiint_V \frac{\boldsymbol{r}'-\boldsymbol{r}}{4\pi|\boldsymbol{r}-\boldsymbol{r}'|}\,dV' = \frac{1}{12\pi}\sum_j \left[\hat{n}_j \sum_i \left[\left[\hat{P}_{ij}^0\cdot\hat{u}_{ij}\right]\left[A_{ij}^3 A_{ij}^4 + A_{ij}^5 - |d_j|^3\left(A_{ij}^1 - A_{ij}^2\right)\right]\right]\right]$$

$$A_{ij}^1 = \operatorname{atan}\left(\frac{P_{ij}^0 l_{ij}^+}{\left(R_{ij}^0\right)^2 + |d_j| R_{ij}^+}\right)$$

$$A_{ij}^2 = \operatorname{atan}\left(\frac{P_{ij}^0 l_{ij}^-}{\left(R_{ij}^0\right)^2 + |d_j| R_{ij}^-}\right)$$

$$A_{ij}^3 = \ln\left(\frac{R_{ij}^+ + l_{ij}^+}{R_{ij}^- + l_{ij}^-}\right)$$

$$A_{ij}^4 = \frac{P_{ij}^0\left[\left(R_{ij}^0\right)^2 + 2d_j^2\right]}{2}$$

$$A_{ij}^5 = \frac{P_{ij}^0\left[l_{ij}^+ R_{ij}^+ - l_{ij}^- R_{ij}^-\right]}{2}$$

Toplam işlemlerindeki indislerden i kenar, j ise yüz belirtmektedir. İlk integral üçgen yüzeyinden, ikinci ve üçüncü ise tetrahedron hacimlerinden alınmaktadırlar. Integraller, bir takım matematiksel özdeşlikler kullanılarak, her yüzün her kenarının her köşesinden gelen bir takım büyüklüklerin toplamına indirgenmişlerdir.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN	
TARİHİ: 05/02/2018	**KAPSAMI: Tekillik integrallerindeki büyüklüklerin tanımı.**

11) Tekillik integrallerindeki büyüklüklerin tanımı:

Tekillik integrallerinin analitik ifadelerindeki büyüklüklerin tamamı, aslında yalnızca gözlem noktası ve şekillerin köşe noktalarından, bir takım geometrik hesaplarla türetilmişlerdir. Tam olarak ifadeleri şu şekildedir:

$\vec{r}$ = gözlem noktası

$\vec{v}_1, \vec{v}_2$ = j yüzünün i kenarına ait 2 köşe noktası

$\vec{v}_3$ = j yüzüne ait olup i kenarına ait olmayan nokta

$\vec{v}_4$ = j yüzüne ait olmayan tetrahedron köşesi

$\hat{n}_j$ = j yüzünün tetrahedron dışına bakan normali

$\hat{u}_{ij}$ = j yüzüne paralel, i kenarına dik olan olan ve üçgen dışına dönük olan yön

$\hat{l}_{ij} = \hat{n}_j \times \hat{u}_{ij}$

$\vec{s}_{ij} = \vec{v}_2 + \vec{v}_1$

$\vec{m}_{ij} = \vec{v}_2 - \vec{v}_1$

$\hat{m}_{ij}$ = $\vec{m}_{ij}$ vektörünün birim vektörü

$\vec{B}_{ij} = \left(\hat{m}_{ij} \cdot \hat{l}_{ij}\right)\vec{m}_{ij}$

$\vec{\rho}_{ij}^{-} = \frac{1}{2}\left(\vec{s}_{ij} - \vec{B}_{ij}\right)$

$\vec{\rho}_{ij}^{+} = \frac{1}{2}\left(\vec{s}_{ij} + \vec{B}_{ij}\right)$

$\vec{\rho}_j$ = gözlem noktasının j yüzüne iz düşümü.

$d_j = \left(\vec{r} - \vec{\rho}_j\right) \cdot \hat{n}_j$

$\vec{l}_{ij} = \vec{\rho}_{ij}^{+} - \vec{\rho}_{ij}^{-}$

$L_{ij} = \left|\vec{l}_{ij}\right|$

$\vec{P}_{ij}^{-} = \vec{\rho}_{ij}^{-} - \vec{\rho}_j$

$\vec{P}_{ij}^{+} = \vec{\rho}_{ij}^{+} - \vec{\rho}_j$

$l_{ij}^{-} = \vec{P}_{ij}^{-} \cdot \hat{l}_{ij}$

$l_{ij}^{+} = \vec{P}_{ij}^{+} \cdot \hat{l}_{ij}$

$P_{ij}^{0} = \left|\vec{P}_{ij}^{-} \cdot \hat{u}_{ij}\right|$

$P_{ij}^{-} = \left|\vec{P}_{ij}^{-}\right|$

$P_{ij}^{+} = \left|\vec{P}_{ij}^{+}\right|$

$\hat{P}_{ij}^{0} = \left(\vec{P}_{ij}^{-} - l_{ij}^{-}\hat{l}_{ij}\right) / P_{ij}^{0}$

R_{ij}^{0} = gözlem noktasının i kenarına uzaklığı

$R_{ij}^{-} = \left|\vec{r} - \vec{\rho}_{ij}^{-}\right|$

$R_{ij}^{+} = \left|\vec{r} - \vec{\rho}_{ij}^{+}\right|$

i ve j indisleri, toplamda 12 farklı değere denk gelmektedir. Bu yüzden bu hesaplar, tetrahedron köşe noktaları $\vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4$ değerlerinin farklı farklı kombinasyonlarıyla 12 kez hesaplanmak zorundadırlar. Üçgen yüzeyi üzerindeki integralde ise, her kenar için, toplam 3 kez yapılması gerekmektedir.

Stajımı bu firmada yaptım. **Staj Yapanın İmzası**	**Staj Yeri Yetkilisinin** **Adı, Soyadı, İmzası, Firma Kaşesi**

YAPILAN İŞİN	
TARİHİ: 06/02/2018	**KAPSAMI: Tekillik integrallerinin hesaplanması.**

12) Tekillik integrallerinin hesaplanması:

Tetrahedrondan alınan integral, ana fonksiyon, yüzey için fonksiyon ve kenar için fonksiyon şeklinde 3 farklı fonksiyonla aşağıdaki gibi hesaplanabilir. Üçgen integrali için de benzer yöntem kullanılabilir.

```
inline void GreenVolVolIntegralEdge(
        T& edgeGIVV, T& edgeGDVV,
        const LUV::LuVector3< T >& dirN, const LUV::LuVector3< T >& vecRho, const T& magD,
        const LUV::LuVector3< T >& vecObs,
        const LUV::LuVector3< T >& vecTetra, const LUV::LuVector3< T >& vecFace,
        const LUV::LuVector3< T >& vecV1, const LUV::LuVector3< T >& vecV2
){
        LUV::LuVector3< T > dirU = -LUV::LineNormalP( vecFace, vecV1, vecV2 );
        LUV::LuVector3< T > dirL = LUV::Cross( dirN, dirU );
        LUV::LuVector3< T > vecS = vecV2 + vecV1;
        LUV::LuVector3< T > vecM = vecV2 - vecV1;
        LUV::LuVector3< T > dirM = Unit( vecM );
        LUV::LuVector3< T > vecB = LUV::Dot( dirM, dirL ) * vecM;
        LUV::LuVector3< T > vecRhoM = ( vecS - vecB ) / 2;
        LUV::LuVector3< T > vecRhoP = ( vecS + vecB ) / 2;
        LUV::LuVector3< T > vecPM = vecRhoM - vecRho;
        LUV::LuVector3< T > vecPP = vecRhoP - vecRho;
        T magLM = LUV::Dot( vecPM, dirL );
        T magLP = LUV::Dot( vecPP, dirL );
        T magP0 = abs( LUV::Dot( vecPM, dirU ) );
        LUV::LuVector3< T > dirP0 = ( vecPM - magLM * dirL ) / magP0;
        T magR0 = LUV::Length( vecObs - LUV::ProjLine( vecObs, vecRhoM, vecRhoP ) );
        T magRM = LUV::Length( vecObs - vecRhoM );
        T magRP = LUV::Length( vecObs - vecRhoP );
        T magR0S = magR0 * magR0;
        T magAbsD = abs( magD );
        T magDS = magD * magD;
        T magAbsDC = magAbsD * magDS;
        T magA1 = atan( ( magP0 * magLP ) / ( magR0S + magAbsD * magRP ) );
        T magA2 = atan( ( magP0 * magLM ) / ( magR0S + magAbsD * magRM ) );
        T magA3 = log( ( magRP + magLP ) / ( magRM + magLM ) );
        T magA4 = ( magP0 * ( magR0S + 2.0 * magDS ) ) / 2.0;
        T magA5 = ( magP0 * ( magLP * magRP - magLM * magRM ) ) / 2.0;
        T magA6 = LUV::Dot( dirP0, dirU );
        T magA7 = magA1 - magA2;
        edgeGIVV += magA6 * ( magAbsD * magA7 - magP0 * magA3 );
        edgeGDVV += magA6 * ( magA3 * magA4 + magA5 - magAbsDC * magA7 );
}

inline void GreenVolVolIntegralFace(
        T& GIVV, LUV::LuVector3< T >& GDVV,
        const LUV::LuVector3< T >& vecObs, const LUV::LuVector3< T >& vecTetra,
        const LUV::LuVector3< T >& vecV1, const LUV::LuVector3< T >& vecV2, const LUV::LuVector3< T >& vecV3
){
        LUV::LuVector3< T > dirN = -LUV::PlaneNormalP( vecTetra, vecV1, vecV2, vecV3 );
        LUV::LuVector3< T > vecRho = LUV::ProjPlane( vecObs, vecV1, dirN );
        T magD = LUV::Dot( vecObs - vecRho, dirN );
        T magAbsD = abs( magD );
        T edgeGIVV = 0; T edgeGDVV = 0;
        GreenVolVolIntegralEdge( edgeGIVV, edgeGDVV, dirN, vecRho, magD, vecObs, vecTetra, vecV1, vecV2, vecV3 );
        GreenVolVolIntegralEdge( edgeGIVV, edgeGDVV, dirN, vecRho, magD, vecObs, vecTetra, vecV2, vecV3, vecV1 );
        GreenVolVolIntegralEdge( edgeGIVV, edgeGDVV, dirN, vecRho, magD, vecObs, vecTetra, vecV3, vecV1, vecV2 );
        GIVV += magD * edgeGIVV; GDVV += dirN * edgeGDVV;
}

inline void GreenVolVolIntegral(
        T& GIVV, LUV::LuVector3< T >& GDVV, const LUV::LuVector3< T >& vecObs,
        const LUV::LuVector3< T >& vecV1, const LUV::LuVector3< T >& vecV2,
        const LUV::LuVector3< T >& vecV3, const LUV::LuVector3< T >& vecV4
){
        GreenVolVolIntegralFace( GIVV, GDVV, vecObs, vecV1, vecV2, vecV3, vecV4 );
        GreenVolVolIntegralFace( GIVV, GDVV, vecObs, vecV2, vecV3, vecV4, vecV1 );
        GreenVolVolIntegralFace( GIVV, GDVV, vecObs, vecV3, vecV4, vecV1, vecV2 );
        GreenVolVolIntegralFace( GIVV, GDVV, vecObs, vecV4, vecV1, vecV2, vecV3 );
        GIVV *= 1.0 / ( 8.0 * pi_ ); GDVV *= 1.0 / ( 12.0 * pi_ );
}
```

Stajımı bu firmada yaptım. **Staj Yapanın İmzası**	**Staj Yeri Yetkilisinin** **Adı, Soyadı, İmzası, Firma Kaşesi**

YAPILAN İŞİN	
TARİHİ: 07/02/2018	KAPSAMI: MoM matrisinin oluşturulması.

13) MoM matrisinin oluşturulması:

Elektrik alan hacim integral denkleminin, SWG fonksiyonları ile iç çarpımının, N adet N bilinmeyenli denklem oluşturacağını söylemiştik.

$$\iiint_V \{\boldsymbol{LHS}\}(\boldsymbol{r}) \cdot \boldsymbol{f}_m(\boldsymbol{r}) dV = \iiint_V \boldsymbol{E}^{inc}(\boldsymbol{r}) \cdot \boldsymbol{f}_m(\boldsymbol{r}) dV$$

N adet N bilinmeyenli denklem, şu şekilde bir matris denklemi oluşturur:

$$\boldsymbol{Za} = \boldsymbol{w}$$

$\boldsymbol{Z}$ burada MoM matrisidir. Satır indisi m, sütun indisi n'dir. Her bir elemanı, n'inci ve m'inci SWG fonksiyonları üzerinden yapılan bir takım hesaplarla bulunur. SWG fonksiyonları 2 adet tetrahedron üzerine tanımlı olduklarından, 2 SWG fonsiyonunun birbiriyle ilişkili hesapları, aslında 4 adet tetrahedron-tetrahedron ilişkili hesaplama getirecektir. Bu, programa şu şekilde gömüldüğünde;

```
inline void GenerateSingleElem( const UINT_T& idFaceM, const UINT_T& idFaceN )
{
	UINT_T* idTetraM = &faceTetraIndexPtr_[ 2 * idFaceM ];
	UINT_T* idTetraN = &faceTetraIndexPtr_[ 2 * idFaceN ];

	complex< T > result;

	result = TetrahedralIntegral( idFaceM, idFaceN, idTetraM[ 0 ], idTetraN[ 0 ] );

	if( idTetraN[ 1 ] != -1 )
	{
		result -= TetrahedralIntegral( idFaceM, idFaceN, idTetraM[ 0 ], idTetraN[ 1 ] );
	}

	if( idTetraM[ 1 ] != -1 )
	{
		result -= TetrahedralIntegral( idFaceM, idFaceN, idTetraM[ 1 ], idTetraN[ 0 ] );

		if( idTetraN[ 1 ] != -1 )
		{
			result += TetrahedralIntegral( idFaceM, idFaceN, idTetraM[ 1 ], idTetraN[ 1 ] );
		}
	}

	momDataPtr_[ idFaceN * faceCount_ + idFaceM ] = result; // column major
}
```

m'inci yüzeyin a'ıncı tetrahedronundan ve n'inci yüzeyin b'inci tetrahedronundan gelecek katkı, en basit haliyle şu şekilde ifade edilebilir (programdaki `TetrahedralIntegral` fonksiyonu):

$$\begin{aligned}\boldsymbol{Z}[m,n,a,b] = &\frac{C\gamma}{j\omega\epsilon_0\epsilon_{rb}}\left((\vec{v}_m \cdot \vec{v}_n)V_a + I_{17} + I_{18} + I_{19} - (\vec{v}_m + \vec{v}_n) \cdot \vec{I}_{14}\right) \\ &-j\omega\mu_0\kappa_b C\left((\vec{v}_m \cdot \vec{v}_n)I_1 + I_8 + I_9 + I_{10} - \vec{v}_n \cdot \vec{I}_2 - \vec{v}_m \cdot \vec{I}_5\right) \\ &-\frac{9C\kappa_b}{j\omega\epsilon_0}\left(I_1 - \frac{V_a}{A_m}I_{11} - \frac{V_b}{A_n}I_{12} + \frac{V_aV_b}{A_mA_n}I_{13}\right)\end{aligned}$$

Buradaki I değerleri, Green fonksiyonunun tetrahedronlar ve yüzeyler üzerinden bir takım integralleridir. Diğer değerlerle birlikte, bir sonraki sayfada tanımlanacaklardır.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN	
TARİHİ: 08/02/2018	**KAPSAMI: MoM matrisi formülündeki değerlerin tanımı.**

14) MoM matrisi formülündeki değerlerin tanımı:

$\vec{v}_m, \vec{v}_n$ = SWG noktaları, yani tetrahedronda olup da yüzeyde olmayan vertex

V_a, V_b = tetrahedron hacimleri

A_m, A_n = üçgensel yüz alanları

ϵ_0, μ_0 = boş uzayın elektrik ve manyetik geçirgenliği

ω = gelen alanın açısal frekansı

ϵ_{rb} = b'inci tetrahedronun dielektrik sabiti

γ = (a==b) durumunda (yani a ve b aynı tetrahedonlar ise) 1, değilse 0

κ_b = $\frac{1-\epsilon_{rb}}{\epsilon_{rb}}$, b'inci tetrahedronun kontrastı

C = $\frac{A_m A_n}{9 V_a V_b}$, formulde sıkça tekrarlanan bir çarpan

$\vec{I}_2$ = (I_2, I_3, I_4) şeklindeki vektör

$\vec{I}_5$ = (I_5, I_6, I_7) şeklindeki vektör

$\vec{I}_{14}$ = (I_{14}, I_{15}, I_{16}) şeklindeki vektör

$$I_1 = \iiint_{V_a} \iiint_{V_b} G(\boldsymbol{r}, \boldsymbol{r}') dV' \, dV$$

$$I_{\{2,3,4\}} = \iiint_{V_a} \{x, y, z\} \iiint_{V_b} G(\boldsymbol{r}, \boldsymbol{r}') dV' \, dV$$

$$I_{\{5,6,7\}} = \iiint_{V_a} \iiint_{V_b} \{x', y', z'\} G(\boldsymbol{r}, \boldsymbol{r}') dV' \, dV$$

$$I_8 = \iiint_{V_a} x \iiint_{V_b} x' G(\boldsymbol{r}, \boldsymbol{r}') dV' \, dV$$

$$I_9 = \iiint_{V_a} y \iiint_{V_b} y' G(\boldsymbol{r}, \boldsymbol{r}') dV' \, dV$$

$$I_{10} = \iiint_{V_a} z \iiint_{V_b} z' G(\boldsymbol{r}, \boldsymbol{r}') dV' \, dV$$

$$I_{11} = \iint_{S_m} \iiint_{V_b} G(\boldsymbol{r}, \boldsymbol{r}') dV' \, dS$$

$$I_{12} = \iiint_{V_a} \iint_{S_n} G(\boldsymbol{r}, \boldsymbol{r}') dS' \, dV$$

$$I_{13} = \iint_{S_m} \iint_{S_n} G(\boldsymbol{r}, \boldsymbol{r}') dS' \, dS$$

$$I_{\{14,15,16\}} = \iiint_{V_a} \{x, y, z\} dV$$

$$I_{\{17,18,19\}} = \iiint_{V_a} \{x^2, y^2, z^2\} dV$$

Stajımı bu firmada yaptım. **Staj Yapanın İmzası**	**Staj Yeri Yetkilisinin** **Adı, Soyadı, İmzası, Firma Kaşesi**

YAPILAN İŞİN	
TARİHİ: 09/02/2018	KAPSAMI: MoM matrisi formülündeki integrallerin çözümü.

15) MoM matrisi formülündeki integrallerin çözümü:

Bir önceki sayfadaki $I_{1..19}$ integrallerinin regüler kısımları direkt olarak quadrature noktalarıyla nümerik olarak hesaplanmıştır. Tekil kısımların integrali ise, 10. sayfadaki analitik çözümle hesaplanmıştır. Integraller, aslında toplamlarla ifade edilebilir ve bilgisayar tarafından direkt olarak çözülebilir haldedirler.

$\vec{r}_m^V, \vec{r}_n^V, \vec{r}_m^S, \vec{r}_n^S$ = tetrahedronlara ve üçgensel yüzeylere ait quadrature noktaları

$w_m^V, w_n^V, w_m^S, w_n^S$ = tetrahedronlara ve üçgensel yüzeylere ait quadrature ağırlıkları

$$R_{nm}^{VV} = |\vec{r}_m^V - \vec{r}_n^V| \qquad R_{nm}^{SV} = |\vec{r}_m^S - \vec{r}_n^V| \qquad R_{nm}^{VS} = |\vec{r}_m^V - \vec{r}_n^S| \qquad R_{nm}^{SS} = |\vec{r}_m^S - \vec{r}_n^S|$$

$$G_{nm}^{NVV} = \frac{e^{jkR_{nm}^{VV}} - 1}{4\pi R_{nm}^{VV}} \qquad G_{nm}^{NSV} = \frac{e^{jkR_{nm}^{SV}} - 1}{4\pi R_{nm}^{SV}} \qquad G_{nm}^{NVS} = \frac{e^{jkR_{nm}^{VS}} - 1}{4\pi R_{nm}^{VS}} \qquad G_{nm}^{NSS} = \frac{e^{jkR_{nm}^{SS}} - 1}{4\pi R_{nm}^{SS}}$$

$$G_m^{IVV} = \iiint_{V_b} \frac{1}{4\pi|\vec{r}_m^V - \boldsymbol{r}'|} dV' \qquad G_m^{ISV} = \iiint_{V_b} \frac{1}{4\pi|\vec{r}_m^S - \boldsymbol{r}'|} dV' \qquad G_m^{IVS} = \iint_{S_n} \frac{1}{4\pi|\vec{r}_m^V - \boldsymbol{r}'|} dS' \qquad G_m^{ISS} = \iint_{S_n} \frac{1}{4\pi|\vec{r}_m^S - \boldsymbol{r}'|} dS'$$

$$\vec{G}_m^{DVV} = \iiint_{V_b} \frac{\boldsymbol{r}' - \vec{r}_m^V}{4\pi\left|\vec{r}_m^V - \boldsymbol{r}'\right|} dV' = \left(G_{mx}^{DVV}, G_{my}^{DVV}, G_{mz}^{DVV}\right)$$

$$I_1 = \sum_m w_m^V \left[\sum_n w_n^V G_{nm}^{NVV} + G_m^{IVV}\right]$$

$$I_{\{2,3,4\}} = \sum_m w_m^V \{x_m^V, y_m^V, z_m^V\} \left[\sum_n w_n^V G_{nm}^{NVV} + G_m^{IVV}\right]$$

$$I_{\{5,6,7\}} = \sum_m w_m^V \left[\sum_n w_n^V \{x_n^V, y_n^V, z_n^V\} G_{nm}^{NVV} + \left\{G_{mx}^{DVV}, G_{my}^{DVV}, G_{mz}^{DVV}\right\} + x_m^V G_m^{IVV}\right]$$

$$I_{\{8,9,10\}} = \sum_m w_m^V \{x_m^V, y_m^V, z_m^V\} \left[\sum_n w_n^V \{x_n^V, y_n^V, z_n^V\} G_{nm}^{NVV} + \left\{G_{mx}^{DVV}, G_{my}^{DVV}, G_{mz}^{DVV}\right\} + x_m^V G_m^{IVV}\right]$$

$$I_{11} = \sum_m w_m^S \left[\sum_n w_n^V G_{nm}^{NSV} + G_m^{ISV}\right]$$

$$I_{12} = \sum_m w_m^V \left[\sum_n w_n^S G_{nm}^{NVS} + G_m^{IVS}\right]$$

$$I_{13} = \sum_m w_m^S \left[\sum_n w_n^S G_{nm}^{NSS} + G_m^{ISS}\right]$$

$$I_{\{14,15,16\}} = \sum_m w_m^V \{x_m^V, y_m^V, z_m^V\}$$

$$I_{\{17,18,19\}} = \sum_m w_m^V \{x_m^V x_m^V, y_m^V y_m^V, z_m^V z_m^V\}$$

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN	
TARİHİ: 12/02/2018	**KAPSAMI: MoM matris denkleminin sağ yanının hesabı.**

16) MoM matris denkleminin sağ yanının hesabı:

Denklemin sağ yanını oluşturmak, matrisi oluşturmaya kıyasla çok daha az meşakkatlidir.

$$\iiint_V \boldsymbol{E}^{inc}(\boldsymbol{r}) \cdot \boldsymbol{f}_m(\boldsymbol{r}) dV$$

Gelen alanın, her bir SWG fonksiyonuyla iç çarpımı, MoM matris denkleminin sağ yanı olan N elemanlı $\boldsymbol{w}[m]$ vektörünü üretecektir.

Her bir SWG fonksiyonunda 2 adet tetrahedron olduğundan, aslında bu integral 2 farklı tetrahedral bölge üzerinden alınmaktadır. MoM matrisindeki gibi, bu durum programa şu şekilde gömülürse;

```
inline void GenerateSingleElem( const UINT_T& idFace ) const
{
        ...
        complex< T > result = TetrahedralIntegral( idFace, idTetra[ 0 ] );
        if( idTetra[ 1 ] != -1 )
        {
                result -= TetrahedralIntegral( idFace, idTetra[ 1 ] );
        }
        rhsDataPtr_[ idFace ] = result;
}
```

m'inci yüzeyin a'ıncı tetrahedronundan, yani `TetrahedralIntegral` fonksiyonundan gelen katkı,

$$\boldsymbol{w}[m,a] = \frac{A_m}{3V_a} \iiint_{V_a} \boldsymbol{E}^{inc}(\boldsymbol{r}) \cdot (\boldsymbol{r} - \vec{v}_m) dV$$

olmaktadır. Bu da quadrature noktaları ve ağırlıklarıyla basit bir nümerik integral şeklinde hesaplanabilir:

$$\boldsymbol{w}[m,a] = \frac{A_m}{3V_a} \sum_n w_n^V \boldsymbol{E}^{inc}(\vec{r}_n^V) \cdot (\vec{r}_n^V - \vec{v}_m)$$

Quadrature noktalarından integral alımı, kodlarda şu şekilde gözükmektedir:

```
UINT_T idTetra4 = 4 * idTetra;
UINT_T* idTetraVertex = &tetraVertexIndexPtr_[ idTetra4 ];
LUV::LuVector3< T >* tetraVertex[ 4 ] = {
    reinterpret_cast< LUV::LuVector3< T >* >( &vertexDataPtr_[ 3 * idTetraVertex[ 0 ] ] ),
    reinterpret_cast< LUV::LuVector3< T >* >( &vertexDataPtr_[ 3 * idTetraVertex[ 1 ] ] ),
    reinterpret_cast< LUV::LuVector3< T >* >( &vertexDataPtr_[ 3 * idTetraVertex[ 2 ] ] ),
    reinterpret_cast< LUV::LuVector3< T >* >( &vertexDataPtr_[ 3 * idTetraVertex[ 3 ] ] )
};
T tetraVolume = LUV::TetrahedronVolume( *tetraVertex[0], *tetraVertex[1], *tetraVertex[2], *tetraVertex[3] );
UINT_T idSwgVertex = GetSwgVert( &faceVertexIndexPtr_[ 3 * idFace ], &tetraVertexIndexPtr_[ 4 * idTetra ] );
LUV::LuVector3< T >* swgVertex = reinterpret_cast< LUV::LuVector3< T >* >( &vertexDataPtr_[ 3 * idSwgVertex ] );
complex< T > result( 0, 0 );
UINT_T addressStartQ = quadBlockSize_ * idTetra;
for( UINT_T idq = 0; idq < quadCount_; ++idq )
{
    UINT_T addressQ = addressStartQ + 4 * idq;
    T* magW = &tetraQuadDataPtr_[ addressQ ];
    LUV::LuVector3< T >* vecQ = reinterpret_cast< LUV::LuVector3< T >* >( &tetraQuadDataPtr_[ addressQ + 1 ] );
    result += LUV::Dot( incWave_->FieldE( *vecQ ), *vecQ - *swgVertex ) * *magW;
}
```

Stajımı bu firmada yaptım. **Staj Yapanın İmzası**	**Staj Yeri Yetkilisinin** **Adı, Soyadı, İmzası, Firma Kaşesi**

YAPILAN İŞİN	
TARİHİ: 13/02/2018	KAPSAMI: MoM matris denkleminin çözümü.

17) MoM matris denkleminin çözümü:

MoM matrisi $\boldsymbol{Z}$ ve sağ yan vektörü $\boldsymbol{w}$ oluşturulduktan sonra, bilinmeyen katsayılarımız $\boldsymbol{a}$'yı elde edebilmek için geriye yalnızca matris denlemini çözmek kalıyor.

$$\boldsymbol{Za} = \boldsymbol{w}$$

Kullandığım ArrayFire kütüphanesinde bunu yapan bir fonksiyon hazı bulunduğu için, bu aşama hiç zorlayıcı olmadı.

SolveMoM.exe programcığı'nın main fonksiyonundaki önemli kısım şu şekildedir:

```
MomMatrixFile< FLOAT_T > momFile;
momFile.Load( momFileName );

RhsFile< FLOAT_T > rhsFile;
rhsFile.Load( rhsFileName );

CoefFile< FLOAT_T > coefFile;
coefFile.Initialize( momFile.faceCount_ );

MomSolver< FLOAT_T > momSolver(
        momFile.faceCount_,
        momFile.momData_,
        rhsFile.rhsData_,
        coefFile.coefData_
);

momSolver.Solve();

coefFile.Save( coefFileName );
```

Denklem çözücü MomSolver class'ındaki Solve() metodu ise, şu şekilde yalnızca 6 satırdan oluşur:

```
void Solve()
{
        af::array afMomMatrix = af::array( faceCount_, faceCount_, AF_COMPLEX_T );
        afMomMatrix.write( reinterpret_cast< T* >( momDataPtr_ ), SIZEOF_CT * faceCount_ * faceCount_, afHost );

        af::array afRhsVector = af::array( faceCount_, 1, AF_COMPLEX_T );
        afRhsVector.write( reinterpret_cast< T* >( rhsDataPtr_ ), SIZEOF_CT * faceCount_, afHost );

        af::array afCoefVector = af::solve( afMomMatrix, afRhsVector );
        afCoefVector.host( coefDataPtr_ );
}
```

Gayet hızlı da çalışmaktadır. Stajda kullanmakta olduğum, 3.3 Ghz hızında çalışan 6 çekirdekli bir işlemciye sahip olan bilgisayarın 24 GB RAM'inin 20 GB'ını dolduran bir matris denklemi, 400 saniye gibi bir süre zarfında çözülmektedir.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN	
TARİHİ: 14/02/2018	KAPSAMI: Radar kesit alanı hesaplama.

18) Radar kesit alanı hesaplama:

MoM matris denklemi çözülüp a_n katsayıları elde edilince, cisim üzerindeki deplasman alanı da nümerik olarak ifade edilebilmiş olur.

$$j\omega \boldsymbol{D}(\boldsymbol{r}) = \sum_{n=1}^{N} a_n \boldsymbol{f}_n(\boldsymbol{r})$$

Buna bağlı olarak, cisim üzerindeki akım yoğunluğu da bulunmuş olur.

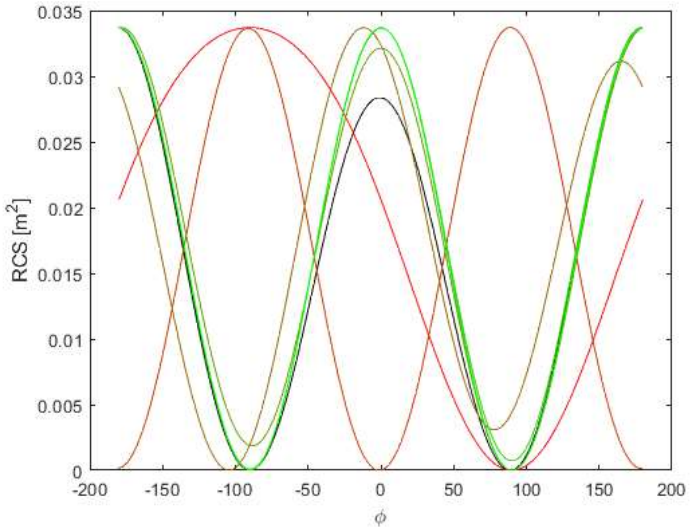
$$\boldsymbol{J}(\boldsymbol{r}) = \sum_{n=1}^{N} a_n \boldsymbol{f}_n(\boldsymbol{r})\, \kappa(\boldsymbol{r})$$

$\kappa(\boldsymbol{r})$, her tetrahedronda sabit olan kontrast oranıdır.

Belirli bir gözlem yönünde ölçülecek radar kesit alanı ise şu şekildedir:

$$\sigma(\hat{\boldsymbol{r}}) = 4\pi \left| \iiint_V \boldsymbol{J}(\boldsymbol{r}') e^{-jk(\boldsymbol{r}'\cdot\hat{\boldsymbol{r}})} dV' \right|^2$$

ObserveRCS.exe programcığında, her bir gözlem yönü için bu integral nümerik olarak hesaplanır ve .rcsresult dosyasına kaydedilir.



Grafikte, kırmızıdan yeşile sırasıyla 6, 24, 105, 278, 591 ve 1110 tetrahedronla modellenmiş bir kürenin, bistatik radar kesit alanı görüntülenmektedir. Kürenin yarıçapı 1 metre, dielektrik sabiti 2, gelen alanın dalga boyu 12 metre, ilerleme yönü -z, polarizasyon yönü +x'dir.

Siyah olan eğri, Feko adlı program kullanılarak ölçülmüştür. Tüm ölçümler, siyah olan eğrinin boyutlarına normalize edilmiştir, aslında çok daha farklı ölçülerde çıkmaktadırlar. Bir yerlerde bir takım yazılımsal hatalar kaldı, ancak stajda kalan son günlerimde bu hatayı bulmayı başaramadım.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN	
TARİHİ: 15/02/2018	KAPSAMI: MATLAB üzerinden C++ programcıklarının kullanımı.

19) MATLAB üzerinden C++ programcıklarının kullanımı:

MATLAB scriptlerinde, system() fonksyionu ile işletim sistemine komutlar gönderilebilmektedir.

C++ programcıkları, bu sayede MATLAB üzerinden çalıştırılabilirler. Giriş-çıkış dosyalarının adları da komut satırı argümanı olarak verilir.

Örneğin aşağıdaki fonksiyon, .unv, .emrule, .incwave, ve .obspoint dosya adlarını alarak, sırasıyla programcıkları çalıştırıp RCS ölçümü yapmış oluyor.

```
function ObserveRCS( geomN, ruleN, waveN, obspN )
    dataN = "Data/";
    ruleF = " " + dataN + ruleN + ".emrule";
    waveF = " " + dataN + waveN + ".incwave";
    obspF = " " + dataN + obspN + ".obsangle";
    unvmF = " " + dataN + geomN + ".unv";
    meshF = " " + dataN + geomN + ".tetramesh";
    faceF = " " + dataN + geomN + ".tetraface";
    propF = " " + dataN + geomN + ".emprops";
    tetqF = " " + dataN + geomN + ".tetraquad";
    triqF = " " + dataN + geomN + ".triquad";
    rhsvF = " " + dataN + geomN + ".rhs";
    mommF = " " + dataN + geomN + ".mommatrix";
    coefF = " " + dataN + geomN + ".coef";
    rsltF = " " + dataN + geomN + ".rcsresult";
    system( "MakeMesh.exe" + unvmF + meshF );
    system( "MakeFace.exe" + meshF + faceF );
    system( "Electrify.exe" + meshF + ruleF + propF );
    system( "MakeQuad.exe" + meshF + faceF + tetqF + triqF );
    system( "MakeRHS.exe" + meshF + faceF + tetqF + waveF + rhsvF );
    system( "MakeMoM.exe" + meshF + faceF + tetqF + triqF + propF + waveF + mommF );
    system( "SolveMoM.exe" + mommF + rhsvF + coefF );
    system( "ObserveRCS.exe" + meshF + faceF + tetqF + propF + waveF + coefF + obspF + rsltF );
end
```

Bu script ise bir önceki sayfadaki grafiği oluşturmakta kullanıldı:

```
N = 6;
geom(1) = "sphere1r1p6t";
geom(2) = "sphere1r2p24t";
geom(3) = "sphere1r3p105t";
geom(4) = "sphere1r4p278t";
geom(5) = "sphere1r5p591t";
geom(6) = "sphere1r6p1110t";
ruleN = "eps2";
waveN = "PlaneZinXpol";
obspN = "xycircular181";

rcs = zeros( 181, N );

for k = 1:N
    ObserveRCS( geom( k ), ruleN, waveN, obspN );
    rcs( :, k ) = ReadRcsResult( geom( k ) );
end

T = readtable( "RCSs.dat" );
rcs0 = table2array( T(:,2) );
figure(  );
plot( -180:2:180, rcs0, "black" )
hold on;
for k = 1:N
    rcsN = rcs( :, k ) - min( rcs( :, k ) );
    rcsN = rcsN ./ max( rcsN );
    rcsN = rcsN .* ( max( rcs0 ) - min( rcs0 ) );
    rcsN = rcsN + min( rcs0 );
    plot( -180 : 2 : 180, rcsN, "Color", [ 1-((k-1)/(N-1)), ((k-1)/(N-1)), 0 ] );
end
hold off;
```

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN	
TARİHİ: 16/02/2018	KAPSAMI: Hızlandırma ve hata giderimine yönelik çalışmalar.

20) Hızlandırma ve hata giderimine yönelik çalışmalar:

Yazılan programlar her ne kadar prototip de olsa, hızlı çalışmaları, hata giderimini ve test prosedürlerini hızlandıracaktır. Programları MATLAB yerine C++ üzerinden yazmış olmak, en başta yapılan hızlandırma çalışmasıdır.

C++ üzerinde kullandığım ArrayFire kütüphanesinin hızlandırma yapmasını umuyordum en başta, ancak sonradan fark ettim ki, onu fütursuzca kullanmak, her adımda gereksiz yere veri kopyalamaya sebebiyet verdiğinden, ciddi bir darboğaz oluşturuyor. Çok fazla veri kopyalayarak çalıştığı yerlerden kaldırınca ArrayFire kullanımını, 200 tetrahedronlu bir kürenin MoM matrix doldurma süresi 1500 saniyelerden 40 saniye civarlarına indi.

Ek olarak, MoM matrisini doldururken hesaplanan integrallerin bazılarının, aslında birden fazla matris elemanında kullanıldığı için, bunları önceden hesaplayıp gerektiğinde hesaplanmış integral sonuçlarını kullanmak, 200 tetrahedronlu kürenin matris doldurma süresini 40 saniyeden 1.5 saniye indirdi!

İşlemlerin çoğu multi-threaded çalışmaktadır. Çekirdek sayısı otomatik olarak algılanıp, o kadar sayıda thread oluşturulmakta ve bu threadler, matris doldurma gibi işlemleri, birbirlerinden bağımsız bir şekilde paralel olarak gerçekleştirirler.

Bir diğer hızlandırma ise, kullanılan matematiksel vektör class'ının N boyutlu template ve loop unrolled oluşudur. Vektör elemanları üzerinde işlemler yapılırken, i'nin 0'dan N'e kadar gittiği bir for döngüsü kullanılmak yerine, template recursion ile for döngüsünün iç kısmı N kere arka arkaya yazılmaktadır program derlenirken. İ'yi sürekli bir arttırma ve N'den küçüklüğünü kontrol etme işi ortadan kalktığı için, 2-3 kata varan bir hızlanma sağlayabilmektedir bu yöntem!

Hata giderme çalışmalarında, ilk olarak 13. sayfadaki matris elemanları formülünde γ çarpanının eklenmemiş olduğu fark edildi. Meğerse o terimin, hem baz hem de testing tetrahedronu aynı tetrahedron olduğunda gelmesi gerekiyormuş. Çarpan bunu düzeltti.

Daha sonradan MoM matrisine tekillik integrallerinden NaN geldiği, fakat ArrayFire matris denklem çözücüsünün hiç gıkını çıkarmadan, NaN'ları 0 gibi kabul ederek hesaplamalara devam ettiği fark edildi. Sorun elbette düzeltildi, bir yerlerde bir takım değerler 0/0 olmaktaymış.

Hacim integral formülasyonunu en baştan tekrar oluşturup, acaba temel matematikte mi bir eksiklik oldu diye kontrol ediyor iken, staj süresi sonlandı.

Nihayetinde, henüz hataları tam çözülememiş de olsa, bir elektrik alan hacim integral denklemi çözücünün temellerini atmış oldum. Nümerik elektromanyetik ile alakalı bir takım şeyleri çok daha iyi kavradım. Programlama yeteneklerimi bilemiş, bazı konularda 1 adım öteye gitmiş oldum.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi