

Hızlandırıcı Fiziği

MADX I

(Methodical Accelerator Design)

Yöntemli Hızlandırıcı Tasarımı Programı

Öznur METE

CERN, Hızlandırıcı Fizikçisi

e-posta: oznur.mete@cern.ch www: www.cern.ch/omete

Teşekkürler

Bu ders 2009'da düzenlenen CERN Hızlandırıcı Okulu'nda verilmiş olan MADX dersi temel alınarak hazırlanmıştır. Desteği için Dr. Werner Herr'e en derin teşekkürlerimi sunarım.

Acknowledgments

This lecture was prepared based on the MADX lecture in the CERN Accelerator School organized in 2009. I express my deepest gratitude to Dr. Werner Herr for his support.

- ▶ Temel kavramların tanımlanması ve MADX dili ile tanışma.
- ▶ Örgü tanımlama.
- ▶ Şimdiye kadar öğrendiğimiz demet optiği fonksiyonlarının verilen bir örgü için hesaplatılması.
- ▶ Hesaplanan parametrelerin benzetim çıktısı olarak alınması,
 - ▶ demet ölçüleri,
 - ▶ ayar, renksellik,
 - ▶ ...

Genel amaçlı hızlandırıcı örgü tasarım programları hakkında...

- ▶ Doğrusal ve dairesel hızlandırıcılar için kullanılabilirler.
- ▶ Özellikleri tanımlanmış bir hızlandırıcı için optik parametreleri hesaplarlar.
- ▶ İstenilen nicelikleri hesaplarlar (denkleştirme / matching).
- ▶ Hızlandırıcı üzerinde oluşabilecek kusurları benzetimini yapabilirler ve bu kusurları düzeltebilirler.
- ▶ Demet dinamiği benzetimi yaparlar.
- ▶ Bu kursta kullanılacak olan genel amaçlı hızlandırıcı örgü tasarımı programı ise MADX olacak.

MADX

- ▶ Şu anda kullanacağımız son hali aslında yıllar sürmüş uzun bir gelişim zincirinin son halkasıdır.
- ▶ 20 yıldan fazladır, CERN'deki çeşitli hızlandırıcıların tasarımında ve benzetim çalışmalarında kullanıldı (PS, SPS, LEP, LHC, gelecek nesil hızlandırıcılar, demet hatları).
- ▶ Şu anda gündemde olan sürümleri: MAD8, MAD9, **MADX (PTC, polymorphic tracking code -- çok biçimli izleme kodu)**.
- ▶ Temel olarak büyük projeler (LEP, LHC, CLIC...) için tasarlanmış bir programdır.

Bu okulda neden MADX kullanmayı seçtik?

- ▶ MADX, büyük tasarım çalışmalarında kullanılabileceği gibi bizim yapacağımız gibi küçük uygulamalar içinde kolayca kullanılabilir.
- ▶ Çok amaçlıdır. Tasarımımızın en erken evrelerinden en gelişmiş değerlendirmelere kadar açılmamıza olanak verir.
- ▶ Tüm işletim sistemleri üzerinde çalıştırabiliriz.
- ▶ Açık kaynak bir uygulamadır ve genişletilmesi kolaydır.
- ▶ Program girdilerinin anlaşılmasının kolay olduğunu hep birlikte göreceğiz.

Herhangi bir optik tasarım programı için ne tür veriler gerekir?

- ▶ Hızlandırıcının tanımı:
 - ➔ Herbir hızlandırıcı bileşeninin tanımı.
 - ➔ Bileşenlerin özellikleri.
 - ➔ Bileşenlerin konumları.
- ▶ Demetin (ya da demetlerin) tanımı.
- ▶ Programın ne yapmasının istenildiğine dair yönergeler.

Peki bu tür verileri MADX'e nasıl gireceğiz?

- ▶ MADX yorumlamalı bir dildir.
 - ➔ Önergeleri kabul eder ve çalıştırır.
 - ➔ Bu önermeler, atamalar, program ifadeleri (expressions) ya da karmaşık eylemlerin başlatılması şeklinde olabilir.
 - ➔ Etkileşimli ya da toplu iş dosyası yardımıyla (batch mode) kullanılabilir.
 - Emirleri bir girdi akışından ya da dosyadan, "grafik kullanıcı arayüzü kullanmadan" okuyabilir.
- ▶ Bir programlama dilinin sahip olduğu pekçok özelliğe sahiptir. (Döngüler, şartlı ifadeler, makrolar, altyardamlar...)

MADX için Girdi Dili

- ▶ C dili ile oldukça fazla benzerlik gösterir.
- ▶ Tüm ifadeler **;** ile sonlandırılır.
- ▶ Yorum satırları **** ya da **!** ile başlar.
- ▶ Temel fonksiyonlar da (**exp, log, sin, cosh, ...**) içinde olmak üzere aritmetik fonksiyonlar kullanılabilir.
- ▶ İçinde farklı dağılımlar için çeşitli gelişigüzel sayı üreticileri bulundurur.
- ▶ Geciktirilmiş ifadeler (**=** yerine **:=**) kullanılabilir.
- ▶ Önceden-tanımlanmış sabitler (**clight, e, pi, mp, me, ...**) kullanılabilir.

MADX için Geleneksel Kullanımlar

- ▶ Büyük harf - küçük harf duyarlılığı yoktur.
- ▶ Hızlandırıcı bileşenler ideal yörünge boyunca yerleştirilir (**s** değişkeni).
- ▶ Yatay eksen **x** (varsayılan eğme düzlemi), dikey eksen **y**'dir.
- ▶ s boyunca ilerleyen bölgesel bir koordinat sistemi kullanır.
 - ❖ Yani $x=y=0$ curvilinear sistemi izler.
- ▶ MADX değişkenleri çifte duyarlılıkta, kayan noktalı sayılardır.
- ▶ = ve := işlemcileri oldukça farklı anlama sahiptir (Gelişigüzel sayı üretici olabilir!).
 - ▶ $DX = GAUSS()*1.5E-3;$
Değer **sadece bir kere** hesaplanır ve DX'te saklanır.
 - ▶ $DX := GAUSS()*1.5E-3;$
Değer DX'in kullanıldığı **her seferinde** yeniden hesaplanır.

Deneyelim...

```
X: ==> angle = 2*pi/1232;
```

```
X: ==> value, angle;
```

```
X: ==> value, asin(1,0)*2;
```

```
X: ==> dx = gauss()*2.0;
```

```
X: ==> value, dx;
```

```
X: ==> value, dx;
```

```
X: ==> dx := gauss()*2.0;
```

```
X: ==> value, dx;
```

```
X: ==> value, dx;
```

Deneyelim...

Bir önceki yazdıklarımızı bir dosya içine yazalım.

```
> madx
```

```
X: == > call, file=dosyam.dosya;
```

```
> madx < dosyam.dosya (linux)
```

MAD girdileri olarak kullanılan ifadeler

- ▶ Genellikle yapılan değer atamalar:
 - ❖ Hızlandırıcı bileşenlerinin özellikleri.
 - ❖ Hızlandırıcı örgüsünün kurulması.
 - ❖ Demet özelliklerinin tanımlanması (parçacık türü, enerjisi, yayınımlı...).
 - ❖ Hızlandırıcı kusurlar ve hatalar.
- ▶ Genellikle gerçekleştirilen eylemler:
 - ❖ Örgü fonksiyonlarının hesaplanması.
 - ❖ Hızlandırıcı kusurlarının düzeltilmesi.

Hızlandırıcı bileşenlerinin tanımlanması

- ▶ Hızlandırıcının tüm bileşenleri MADX'e tanıtılmalıdır.
- ▶ Bileşenler teker teker tanımlanabildiği gibi,
- ▶ aileler (**sınıf, "class"**) halinde de tanımlanabilirler.
- ▶ Bir ailenin içinde olan tüm bileşenlerin özellikleri aynı olacaktır.
- ▶ Tüm bileşenlerin kendine özgü isimleri olabilir (koşul değildir).
- ▶ Bu tanımlamalar, birbirini izleyen emirler ve ifadelerde kullanılabilir.

Hızlandırıcı bileşenlerinin tanımlanması

- ▶ Bir hızlandırıcı bileşeninin türünü tanımlamak için bazı MADX anahtar kelimeleri kullanırız (madx keywords).
- ▶ Tek bir bileşen ya da ortak bir isim vererek bir bileşenler sınıfı tanımlayabiliriz.
- ▶ Genel tanımlama biçimi:
 - ❖ **isim:** **anahtar_kelime**, özellikler;
 - ❖ Şimdi bazı örnekler görelim.

Örnekler: Hızlandırıcı bileşenlerinin tanımlanması

- ▶ İki-kutuplu (eğici) magnet:

MBL: SBEND, L=10.0, ANGLE = 0.0145444;

- ▶ Dört-kutuplu (odaklayıcı) magnet:

MQ: QUADRUPOLE, L=3.3, K1 = 1.23E-02;

- ▶ Alto-kutuplu magnet:

ksf = 0.00156;

MSF: QUADRUPOLE, K2 := ksf, L=1.0;

Magnetik kuvvetlerin tanımlanması

- ▶ İki-kutuplu (eğici) magnet:

$$k_0 = \frac{1}{p/c} B_y [T] \quad \left(= \frac{1}{\rho} = \frac{angle}{\ell} [rad/m] \right)$$

DIP01: SBEND, L=10.0, ANGLE = angle, K0=k0;

DIP02: MBL; ! MBL ailesine ait

DIP03: MBL; ! MBL sınıfının bir "durumu"

- ▶ Dört-kutuplu (odaklayıcı) magnet:

$$k_1 = \frac{1}{p/c} \frac{\partial B_y}{\partial x} [T/m] \quad \left(= \frac{1}{\ell \cdot f} \right)$$

MQA: QUADRUPOLE, L=3.3, K1 =k1;

Magnetik kuvvetlerin tanımlanması

► Altı-kutuplu magnet:

$$k_2 = \frac{1}{p/c} \frac{\partial^2 B_y}{\partial x^2} [T/m^2]$$

KLSF = k2;

MSXF: SEXTUPOLE, L=1.1, K2 = KLSF;

► Sekiz-kutuplu magnet:

$$k_3 = \frac{1}{p/c} \frac{\partial^3 B_y}{\partial x^3} [T/m^3]$$

KLOF = k3;

MOF: OCTUPOLE, L=1.1, K3 = KLOF;

Magnetik kuvvetlerin tanımlanması

► LHC'nin iki-kutuplu magneti:

uzunluk = 14.3;

B = 8.33;

PTOP = 7.0E12

ANGLHC = $B \cdot c_{light} \cdot uzunluk / P_{TOP}$;

MBLHC: **SBEND**, L=uzunluk, ANGLE = anglhc;

ANGLHC = $2 \cdot \pi / 1232$;

MBLHC: **SBEND**, L=uzunluk, ANGLE = anglhc;

Deneyelim...

```
> madx
X: ==> uzunluk = 14.3;
X: ==> B = 8.33;
X: ==> PTOP = 7.0E12;
X: ==> ANGLHC = B*clight*uzunluk/PTOP;
X: ==> MBLHC: SBEND, L = UZUNLUK, ANGLE = ANGLHC;
X: ==> value, mblhc->angle;
```

Kalın ve İnce Bileşenler

- ▶ **Kalın bileşenler:** Şimdiye kadar gördüğümüz tüm örnekler kalın bileşenlerle (ya da mercekler) idi.
- ▶ Bileşen uzunluğu ve kuvveti ayrı ayrı tanımlanır (dipole dışında):
 - + Daha hassas bir yaklaşım, yol farkları ve kenar alanları doğru.
 - Parçacık izleme yaparken symplectic değil (bir sayısal integral metodu)

Kalın ve İnce Bileşenler

- ▶ **İnce bileşenler:** Uzunlukları "sıfır" olan bileşenler olarak tanımlanırlar.
- ▶ Bunların etkin alan integralleri tanımlanır ($k_0.L$, $k_1.L$, $k_2.L$, ...):
 - + Kullanımı kolaydır.
 - + Tekmeleri (kicks) kullanır (genlik-bağımlıdır), her zaman "symplectic" tir.
 - + Bu yüzden izleme için bu yaklaşım kullanılır.
 - Yol uzunlukları tam olarak doğru değildir.
 - Kenar alan etkileri tam olarak doğru değildir.
 - Küçük hızlandırıcılar için bazı sorunlara sebep olabilir.

Kalın ve İnce Bileşenler

MADX'e Özel Bileşenler: Çok-kutuplular (multipoles)

- **Çok-kutuplu (multipole):** Uzunluğu sıfır olan genel bir bileşendir. Herhangi bir dereceden bir ya da daha fazla çok-kutuplu bileşeniyle kullanılabilir:

cok_kutuplu: multipole, knl := {kn0L, kn1L, kn2L, kn3L, ...};

---> knl = kn . L (n. dereceden normal alan bileşenleri)

- **Kullanması çok kolay:**

mul1: multipole, knl := {0, k1L, 0, 0, ...};

(Bir odaklayıcı magnetin tanımlamasına eşdeğerdir.)

mul0: multipole, knl := {angle, 0, 0, ...};

(Bir eğici magnetin tanımlamasına eşdeğerdir.)

Kalın ve İnce Bileşenler

- ▶ Dersimizdeki örnekler için “**ince mercek**” yaklaşımı yapacağız (aksi belirtilmediği durumlarda).
- ▶ Kullanımı kolay ve analitik hesaplarımızla tutarlı olacak.
- ▶ Magnetlerimizi aşağıdaki gibi tanımlayabiliriz:

dipolum: multipole, $k_{nl} := \{\text{angle}, 0, 0, \dots\};$

odaklayıcı: multipole, $k_{nl} := \{0, k_{1L}, 0, \dots\};$

Bir "dizi" nin (sequence) tanımlanması

- ▶ Hızlandırıcımızı tanımlarken tüm bileşenlerinin konumlarını da tanımlamalıyız.
- ▶ Bileşenlerin konumları bir "dizi" (sequence) dosyasında, isimleri ile birlikte verilir.
- ▶ Herhangi bir bileşenin konumunu bileşenin **merkezinde**, **çıkışında** ya da **girişinde** tanımlayabiliriz.
- ▶ Konumlar mutlak ya da görece tanımlanabilir.

```
hpfbu_sps: SEQUENCE, REFER=CENTRE, L=6912;
```

```
...
```

Çeşitli bileşenlerin konumları burada belirtilir.

```
...
```

```
ENDSEQUENCE;
```

Bir "dizi" nin (sequence) tanımlanması

```
SPS : SEQUENCE,                L = 6911.5038;

BEGI.10010      : STARTSPS      , AT = 0;
QF.10010        : QF            , AT = 1.5425      , SLOT_ID = 2361953;
MBA.10030       : MBA           , AT = 6.575      , SLOT_ID = 2361954;
MBA.10050       : MBA           , AT = 13.235     , SLOT_ID = 2361955;
MBB.10070       : MBB           , AT = 19.885     , SLOT_ID = 2361956;
MBB.10090       : MBB           , AT = 26.525     , SLOT_ID = 2361957;
VVSA.10101      : VVSA          , AT = 29.9385    , SLOT_ID = 2361958;

...

...

MBA.63570       : MBA           , AT = 6899.3611  , SLOT_ID = 2363841;
MBA.63590       : MBA           , AT = 6906.0211  , SLOT_ID = 2363842;
LOE.63602       : LOE           , AT = 6909.8401  , SLOT_ID = 2363843;
LSF.63605       : LSF           , AT = 6910.5088  , SLOT_ID = 2363844;
MDH.63607       : MDH           , AT = 6910.9838  , SLOT_ID = 2363845;
BPH.63608       : BPH           , AT = 6911.2713  , SLOT_ID = 2363846;
END.10010       : ENDOFSPS      , AT = 6911.5038;
ENDSEQUENCE;
```

SPS Dizisinden Bir Kesit

Bir "dizi" nin (sequence) tanımlanması

```
circum = 6912;  
  
// bending magnets as thin lenses  
mbsps: multipole,knl={0.007272205};  
  
// quadrupoles and sextupoles  
qf: quadrupole,l=3.085,k1 = 0.0146315;  
qd: quadrupole,l=3.085,k1 = -0.0146434;  
lsf: sextupole,l=1.0, k2 = 1.9518486E-02;  
lsd: sextupole,l=1.0, k2 = -3.7618842E-02;  
  
// monitors and orbit correctors  
bpm: monitor,l=0.1;  
ch: hkicker,l=0.1;  
cv: vkicker,l=0.1;
```

SPS Dizisi - Kalın Bileşenler

sps_all.seq

Bir "dizi" nin (sequence) tanımlanması

```
cassps: sequence, l = circum;  
start_machine: marker, at = 0;  
qf, at = 1.5425;  
lsf, at = 3.6425;  
ch, at = 4.2425;  
bpm, at = 4.3425;  
mbsps, at = 5.0425;  
mbsps, at = 11.4425;  
mbsps, at = 23.6425;  
mbsps, at = 30.0425;  
qd, at = 33.5425;  
lsd, at = 35.6425;  
cv, at = 36.2425;  
bpm, at = 36.3425;  
...  
...  
mbsps, at = 6903.6425;  
mbsps, at = 6910.0425;  
end_machine: marker, at = 6912;  
endsequence;
```

SPS Dizisi - Kalın Bileşenler

sps_all.seq

Bir "dizi" nin (sequence) tanımlanması

```
circum=6912.0; // define the total length
ncell = 108; // define number of cells
lcell = circum/ncell;
// all magnets as multipoles
mbsps: multipole, knl={2.0*pi/(2*ncell)};
qfsps: multipole, knl={0.0, 4.36588E-02};
qdsps: multipole, knl={0.0, -4.36952E-02};
// sequence declaration;
cassps: sequence, refer=centre, l=circum;
  n = 1;
  while (n <= ncell) {
    qfsps: qfsps, at=(n-1)*lcell;
    mbsps: mbsps, at=(n-1)*lcell+16.0;
    qdsps: qdsps, at=(n-1)*lcell+32.00;
    mbsps: mbsps, at=(n-1)*lcell+48.00;
    n = n + 1;
  }
endsequence;
```

SPS Dizisi - Döngü içinde Bileşenler

s1.seq

MADX'in Kullanımı

► Etkileşimli kullanım:

- Terminale **madx** komutunu girelim ve açılacak olan madx oturumunda komutları sıralayalım. (Büyük hızlandırıcılar için çok kullanışlı bir yöntem sayılmaz!)
- Komut satırına **madx** yazalım ve önceden hazırlanmış bir girdi dosyasını çağıralım.

call, file=sps.mad;

► Toplu iş dosyası ile kullanım için komut satırına aşağıdaki komutlar yazılır:

→ **madx < sps.mad**

► Şimdi komutların ve hızlandırıcı tanımlamalarının farklı dosyalarda oluşturulduğu bir örnek görelim:

- **sps.madx**: MADX komutları
- **sps.seq**: Hızlandırıcı ile ilgili tanımlar.

Basit MADX yönergeleri

- ▶ Öncelikle girdi dosyasını ya da dosyalarını tanımlayacağız.
- ▶ Sonra kullanılacak demetin özelliklerini tanımlayacağız.
- ▶ Çeşitli hesaplamaları başlatacağız. (Twiss hesapları, hataların atanması, yörünge düzeltmeleri v.b. ...)
- ▶ Hangi parametrelerin sonuçlarını istediğimizi ve sonuçların çıktı biçimini belirleyeceğiz.
- ▶ Gerekli parametreleri denkleştireceğiz.

Kullanacağımız MADX komutları dosyası

```
TITLE, 'MAD-X deneme';

// Hızlandırıcı tanımlamasını içeren girdi dosyasını oku.
call file="sps.seq";
option,-echo,-thin_foc;

// Hızlandırıcı için kullanılacak demeti tanımla.
Beam, particle = proton, sequence=hpfbu_sps, energy = 450.0;

// hpfbu_sps isimli diziyi kullan.
use, sequence=hpfbu_sps;

// Sonucun içermesini istediğin parametreleri belirle.
select,flag=twiss,column=name,s,betx,bety;

// Twiss parametrelerini hesaplamak için TWISS komutunu çalıştır.
// Hesabi bileşenlerin merkezi için yap ve sonucu twiss.out dosyasına
// yazdır.
twiss,save,centre,file=twiss.out;

// Bir dağıtıcı magnetin 10. ve 16. kez belirlediği aralıkta
// yatay ve dikey beta fonksiyonunu çizdir.

plot, haxis=s, vaxis=betx, bety,colour=100,range=qd[10]/qd[16];
plot, haxis=s, vaxis=dx, colour=100,range=qd[10]/qd[16];

survey,file=survey.out;

stop;
```


Kullanacağımız MADX komutları dosyası

```
TITLE, 'MAD-X deneme';
```

```
// Hızlandırıcı tanımlamasını içeren girdi dosyasını oku.  
call file="sps.seq";  
option,-echo,-thin_foc;
```

```
// Hızlandırıcı için kullanılacak demeti tanımla.  
Beam, particle = proton, sequence=hpfbu_sps, energy = 450.0;
```

```
// hpfbu_sps isimli diziyi kullan.  
use, sequence=hpfbu_sps;
```

```
// Sonucun  
select, flag
```

```
// Twiss pa  
// Hesabi b  
// yazdır.  
twiss,save,
```

```
// Bir dagit  
// yatay ve
```

```
plot, haxis=  
plot, haxis=
```

```
survey,file=survey.out;
```

```
stop;
```

► Girdi dosyasını çağıracağız:

- ❖ **call, file="sps.seq";**
- ❖ Burada hızlandırıcının tanımlandığı bir dosya seçiyoruz.
- ❖ Bu birkaç dosyaya da paylaştırılmış olabilir.

Kullanacağımız MADX komutları dosyası

```
TITLE, 'MAD-X deneme';
```

```
// Hızlandırıcı tanımlamasını içeren girdi dosyasını oku.
```

```
call file="sps.seq";
```

```
option,-echo,-thin_foc;
```

```
// Hızlandırıcı için kullanılacak demeti tanımla.
```

```
Beam, particle = proton, sequence=hpfbu_sps, energy = 450.0;
```

```
// hpfbu_sps isimli diziyi kullan.
```

```
use, sequence=hpfbu_sps;
```

```
// Sonucun
```

```
select, flag
```

```
// Twiss par
```

```
// Hesabi b
```

```
// yazdır.
```

```
twiss,save,c
```

```
// Bir dağıt
```

```
// yatay ve
```

```
plot, haxis=
```

```
plot, haxis=
```

```
survey,file=survey.out;
```

```
stop;
```

► Demetin türü ve özelliklerini tanımlamalıyız:

❖ Parçacık türü

❖ Enerjisi

❖ Yayınımı, içindeki parçacık sayısı, ...

Kullanacağımız MADX komutları dosyası

```
TITLE, 'MAD-X deneme';

// Hızlandırıcı tanımlamasını içeren girdi dosyasını oku.
call file="sps.seq";
option,-echo,-thin_foc;

// Hızlandırıcı için kullanılacak demeti tanımla.
Beam, particle = proton, sequence=hpfbu_sps, energy = 450.0;
```

```
// hpfbu_sps isimli diziyi kullan.
use, sequence=hpfbu_sps;
```

```
// Sonucun içermesini istediğin parametreleri belirle.
select,flag=twiss,column=name,s,betx,bety;
```

```
// Twiss parametreleri
// Hesabi birim
// yazdır.
twiss,save,c
```

```
// Bir digit
// yatay ve
```

```
plot, haxis=
plot, haxis=
```

```
survey,file=survey.out;
```

```
stop;
```

▶ Hızlandırıcıyı etkinleştirmeliyiz:

- ❖ USE, sequence = hpfbu_sps;
- ❖ "sps.seq" içinde başka diziler de bulunabilir.
- ❖ USE komutu ile belirlenen dizi kullanılır.

Kullanacağımız MADX komutları dosyası

```
TITLE, 'MAD-X deneme';

// Hızlandırıcı tanımlamasını içeren girdi dosyasını oku.
call file="sps.seq";
option,-echo,-thin_foc;

// Hızlandırıcı için kullanılacak demeti tanımla.
Beam, particle = proton, sequence=hpfbu_sps, energy = 450.0;

// hpfbu_sps isimli diziyi kullan.
use, sequence=hpfbu_sps;
```

```
// Sonucun içermesini istediğin parametreleri belirle.
select,flag=twiss,column=name,s,betx,bety;
```

```
// Twiss parametrelerini hesaplamak için TWISS komutunu çalıştır.
// Hesabi bileşenlerin merkezi konumunu belirle.
// yazdır.
twiss,save,centre,file=twiss.out;

// Bir dağıtıcı magnetin 10.
// yatay ve dikey beta fonksiyonlarını hesapla.

plot, haxis=s, vaxis=betx, baxis=dx, col=twiss;
plot, haxis=s, vaxis=dx, col=twiss;

survey,file=survey.out;

stop;
```

► **SELECT** komutu ile üzerinde çalışacağımız parametreleri seçmeliyiz. Seçilen parametrelerle şunlar yapılabilir:

- ❖ Twiss parametrelerinin hesabı
- ❖ Örgü fonksiyonlarının kaydedilmesi
- ❖ Beta fonksiyonun çizdirilmesi
- ❖ ...

Kullanacağımız MADX komutları dosyası

```
TITLE, 'MAD-X deneme';

// Hızlandırıcı tanımlamasını içeren girdi dosyasını oku.
call file="sps.seq";
option,-echo,-thin_foc;

// Hızlandırıcı için kullanılacak demeti tanımla.
Beam, particle = proton, sequence=hpfbu_sps, energy = 450.0;

// hpfbu_sps isimli diziyi kullan.
use, sequence=hpfbu_sps;

// Sonucun içermesini istediğin parametreleri belirle.
select,flag=twiss,column=name,s,betx,bety;

// Twiss parametrelerini hesaplamak için TWISS komutunu çalıştır.
// Hesabi bileşenlerin merkezi için yap ve sonucu twiss.out dosyasına
// yazdır.
twiss,save,centre,file=twiss.out;

// Bir dağıtıcı magnetin 10. ve 16. kez belirlediği
// yatay ve dikey beta fonksiyonunu çizdir.

plot, haxis=s, vaxis=betx, bety,colour=100,range=qd[10]/qd[16];
plot, haxis=s, vaxis=dx, colour=100,range=qd[10]/qd[16];

survey,file=survey.out;

stop;
```

► Bir komut çalıştıralım; dairesel hızlandırıcı çevresinde örgü parametrelerinin hesaplanması:

- ❖ **twiss;** ya da
- ❖ **twiss, file=output;** ya da
- ❖ **twiss, sequence=hpfbu_sps;**

Kullanacağımız MADX komutları dosyası

```
TITLE, 'MAD-X deneme';

// Hızlandırıcı tanımlamasını içeren girdi dosyasını oku.
call file="sps.seq";
option,-echo,-thin_foc;

// Hızlandırıcı için kullanılacak demeti tanımla.
Beam, particle = proton, sequence=hpfbu_sps, energy = 450.0;

// hpfbu_sps isimli diziyi kullan.
use, sequence=hpfbu_sps;

// Sonucun içermesini istediğin parametreleri belirle.
select,flag=twiss,column=name,s,betx,bety;

// Twiss parametrelerini hesapla
// Hesabi bileşenlerin merkezi
// yazdır.
twiss,save,centre,file=twiss.out;
```

► Bir komut çalıştıralım; beta fonksiyonu ve dağılım için bir grafik çıktı oluşturalım.

```
// Bir dağıtıcı magnetin 10. ve 16. kez belirlediği aralıkta
// yatay ve dikey beta fonksiyonunu çizdir.

plot, haxis=s, vaxis=betx, bety, colour=100, range=qd[10]/qd[16];
plot, haxis=s, vaxis=dx, colour=100, range=qd[10]/qd[16];
```

```
survey,file=survey.out;
```

```
stop;
```

Kullanacağımız MADX komutları dosyası

```
TITLE, 'MAD-X deneme';

// Hızlandırıcı tanımlamasını içeren girdi dosyasını oku.
call file="sps.seq";
option,-echo,-thin_foc;

// Hızlandırıcı için kullanılacak demeti tanımla.
Beam, particle = proton, sequence=hpfbu_sps, energy = 450.0;

// hpfbu_sps isimli diziyi kullan.
use, sequence=hpfbu_sps;

// Sonucun içermesini istediğin parametreleri belirle.
select,flag=twiss,column=name,s,betx,bety;

// Twiss parametrelerini hesaplamak için TWISS komutunu çalıştır.
// Hesabi bileşenlerin merkezi için yap ve sonucu twiss.out dosyasına
// yazdır.
twiss,save,centre,file=twiss.out;

// Bir dağıtıcı magnetin 10. ve 11. hücreleri için
// yatay ve dikey beta fonksiyonlarını hesapla.
plot, haxis=s, vaxis=bx, bety, colour=100, range=qd[10]/qd[16];
plot, haxis=s, vaxis=dx, colour=100, range=qd[10]/qd[16];

survey,file=survey.out;

stop;
```

► Bir komut çalıştıralım; hızlandırıcının geometrisini gözden geçirip sonucu bir çıktı dosyasına yazalım.

MADX Sonuç Çıktısına Bir Örnek

```
+++++ table: summ
      length          orbit5          alfa          gammatr
      6912            -0            0.001504942753      25.77745337
      q1              dq1              betxmax          dxmax
      26.57999204     -1.67838253          108.7763569        2.44661758
      dxrms           xcomax           xcorms           q2
      1.830638952     0              0              26.62004577
      dq2             betymax           dymax           dyrms
      -1.680294089    108.7331749        0              0
      ycomax          ycorms           deltap           synch_1
      0              0              0              0
      synch_2         synch_3         synch_4         synch_5
      0              0              0              0
```

► “madx” uzantılı dosyayı çalıştırdığımızda komut satırında görülen program çıktısı.

MADX Sonuç Çıktısına Bir Örnek

```

@ NAME          %05s "TWISS"
@ TYPE          %05s "TWISS"
@ SEQUENCE      %09s "HPFBU_SPS"
@ PARTICLE      %06s "PROTON"
@ MASS          %le          0.938272013
@ CHARGE        %le          1
@ ENERGY       %le          450
@ PC            %le          449.9990218
@ GAMMA         %le          479.6050546
@ KBUNCH        %le          1
@ BCURRENT      %le          0
@ SIGE          %le          0
@ SIGT          %le          0
@ NPART         %le          0
@ EX            %le          1
@ EY            %le          1
@ ET            %le          1
@ LENGTH        %le          6912
@ ALFA          %le          0.001504942753
@ ORBITS        %le          -0
@ GAMMATR       %le          25.77745337
@ Q1            %le          26.57999204
@ Q2            %le          26.62004577
@ DQ1           %le          -1.67838253
@ DQ2           %le          -1.680294089
@ DXMAX         %le          2.44661758
@ DYMAX         %le          0
@ XCOMAX        %le          0
@ YCOMAX        %le          0
@ BETXMAX       %le          108.7763569

```

► "madx" uzantılı dosyayı çalıştırdığımızda oluşacak twiss.out dosyasından bir kesit.



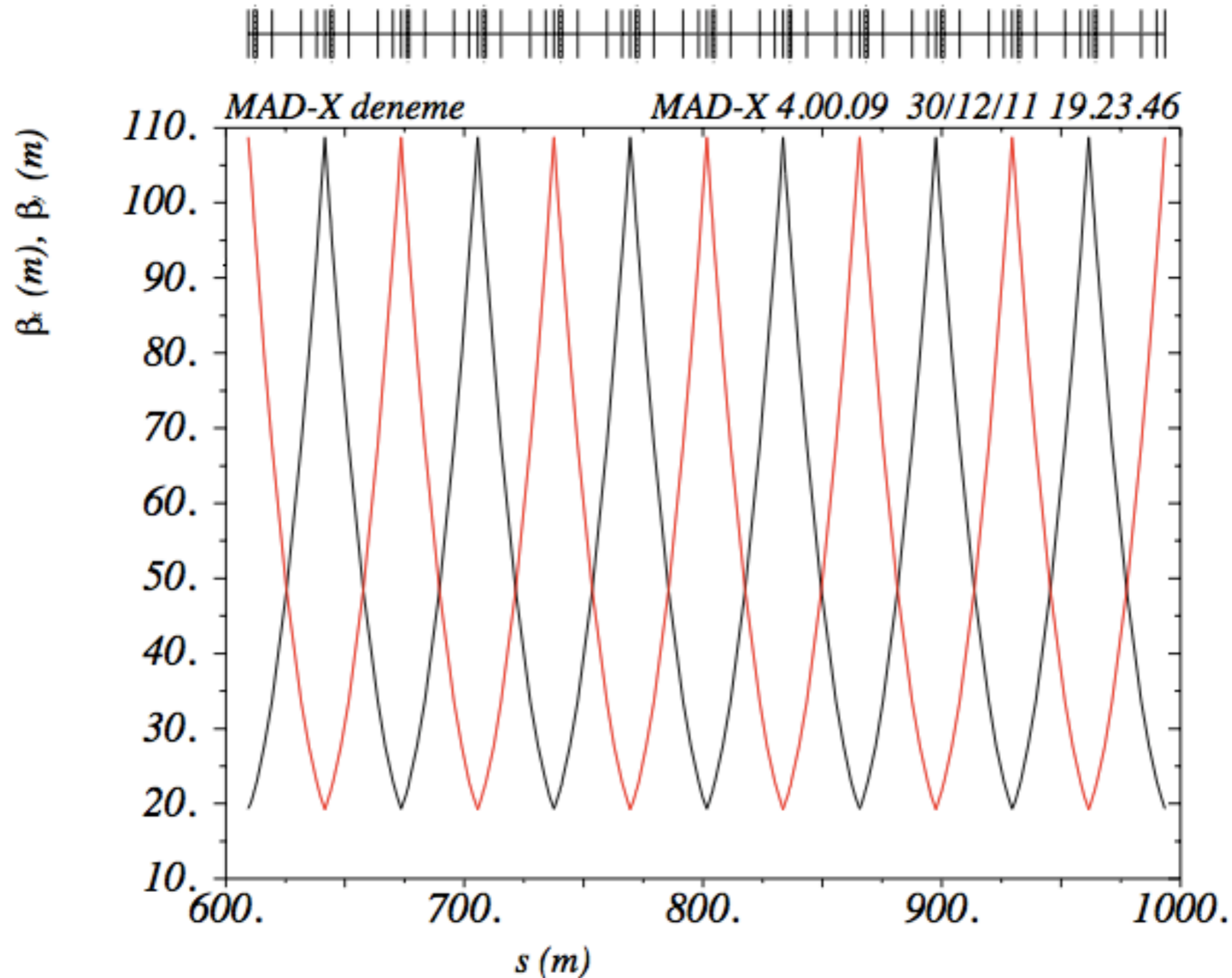
MADX Sonuç Çıktısına Bir Örnek

* NAME	S	BETX	BETY
\$ %s	%le	%le	%le
"HPFBU_SPS\$START"	0	101.5961579	20.70328425
"START_MACHINE"	0	101.5961579	20.70328425
"DRIFT_0"	0.77125	105.1499566	19.94571028
"QF"	1.5425	108.7763569	19.26082066
"DRIFT_1"	2.5925	103.8571423	20.21112973
"LSF"	3.6425	99.07249356	21.29615787
"DRIFT_2"	3.9424975	97.73017837	21.6309074
"CH"	4.2425	96.39882586	21.97666007
"DRIFT_3"	4.2925	96.17800362	22.03535424
"BPM"	4.3425	95.95748651	22.0943539
"DRIFT_4"	4.6925025	94.4223997	22.51590816
"MBSPPS"	5.0425	92.90228648	22.95242507
"DRIFT_5"	8.2425	79.69728195	27.63752778
"MBSPPS"	11.4425	67.74212222	33.5738988
"DRIFT_6"	17.5425	48.41469349	48.35614376
"MBSPPS"	23.6425	33.6289371	67.68523387
"DRIFT_5"	26.8425	27.68865546	79.6433337
"MBSPPS"	30.0425	22.99821861	92.85270185
"DRIFT_7"	31.7925	20.96178735	100.6058286
"QD"	33.5425	19.29915001	108.7331749
"DRIFT_1"	34.5925	20.25187715	103.8118608

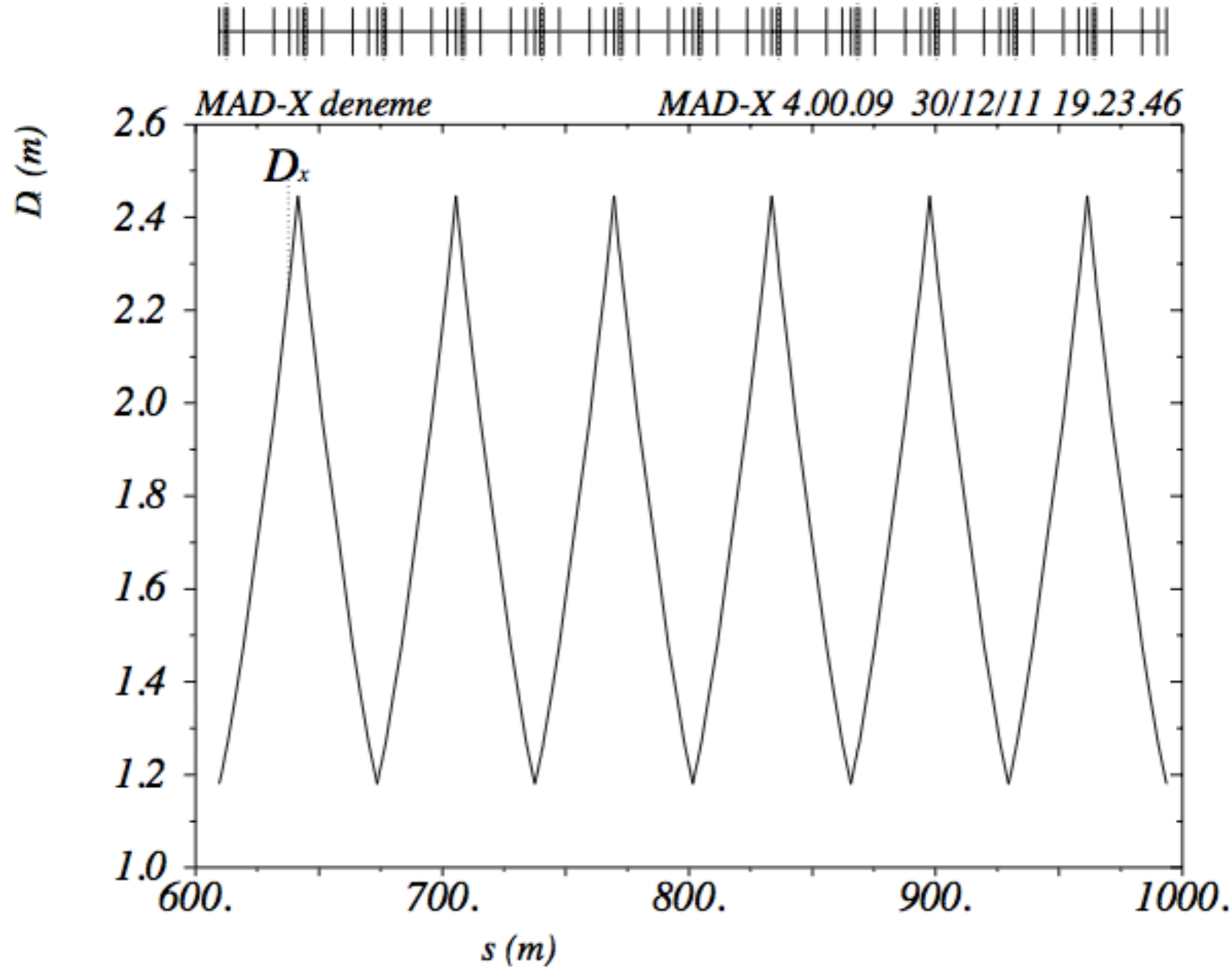
2

► "madx" uzantılı dosyayı çalıştırdığımızda oluşacak twiss.out dosyasından bir kesit.

MADX Grafik Çıktısına Bir Örnek: Beta fonksiyonları

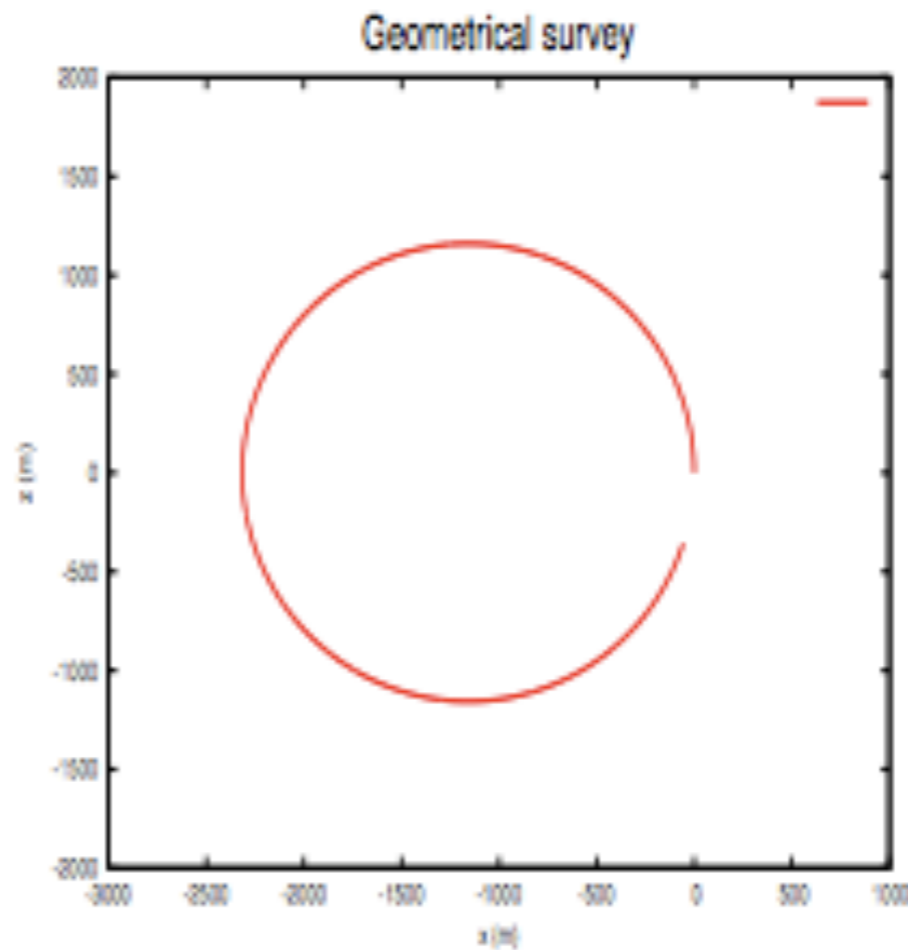
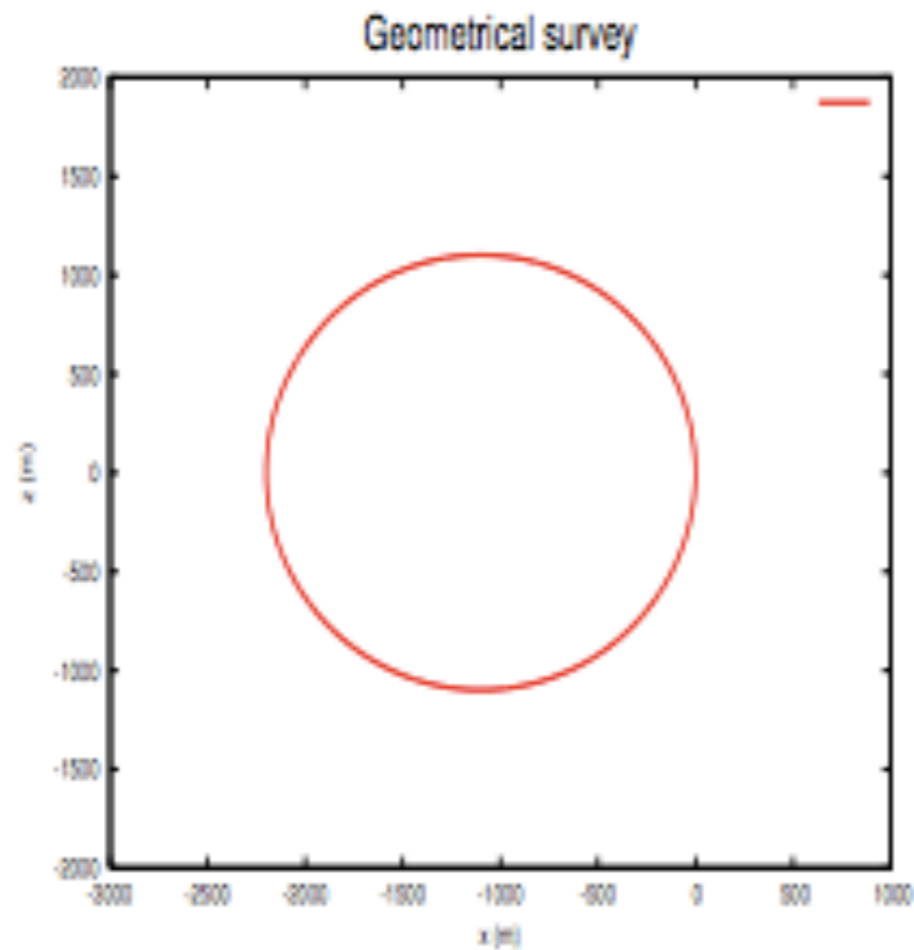


MADX Grafik Çıktısına Bir Örnek: Dağılım fonksiyonu



MADX Grafik Çıktısına Bir Örnek: Hızlandırıcı Tarama

- ▶ Geometrik tarama sonucu x , y , z , theta cinsinden mutlak koordinatları verecektir.
- ▶ Tarama sonucunun olumlu olması için x 'in z 'ye karşı çizimi bir halka vermelidir.



Optik denkleştirme

- ▶ İstenilen optik düzenlemenin gerçekleştirilebilmesi için optik denkleştirme yapılır.
- ▶ Temel uygulamaları:
 - ▶ Global optik parametrelerin ayarlanması (ayar, renksellik ...).
 - ▶ Yerel optik parametrelerin belirlenmesi (beta fonksiyonu ...) --> **gelecek derslerde**
 - ▶ Kusurların düzeltilmesi --> **gelecek derslerde**

Genel (global) parametrelerin denkleştirilmesi

- ▶ İstenilen genel parametreleri (ör: ayar, renksellik) elde edebilmek için magnetlerin kuvvetleri ayarlanır.
- ▶ Denkleştirme yaparken, istediğimiz hedef **parametre değerlerini** ve bunları elde etmek için ayarlanması gereken **bileşenleri** tanımlarız.
- ▶ MADX'de aşağıdaki parametreler geleneksel olarak genel parametreler olarak kabul edilirler:
 - ▶ **Q1, Q2**: Yatay ve dikey ayar.
 - ▶ **dQ1, dQ2**: Yatay ve dikey renksellik.

Genel (global) parametrelerin denkleştirilmesi

- ▶ Örnek olarak, hpfbu_sps dizisinde yatay (Q1) ve dikey (Q2) ayarları denkleştirelim.
- ▶ Bunu yaparken odaklayıcı magnetlerimizin kuvvetlerini (kqf, kqd) değiştirelim.

```
match, sequence=hpfbu_sps;  
  vary,name=kqf, step=0.00001; → deđişecek  
  vary,name=kqd, step=0.00001; → deđişecek  
  global,sequence=hpfbu_sps,Q1=26.58; → hedef deđer  
  global,sequence=hpfbu_sps,Q2=26.62; → hedef deđer  
  Lmdif, calls=10, tolerance=1.0e-21;  
endmatch;
```

sps_denklestir_genel.madx

Bazı ek bilgiler

- ▶ İlk bakışta girdi hazırlama dili karmaşık mı geldi?
- ▶ Bu biçimde iş yapmamızın bazı sebepleri:
 - ❖ Veri tabanı ve diğer programlarla arayüz olasıdır (mathematica, matlab, ...).
 - ❖ MADX için girdi dosyası hazırlayan programlar var.
 - ❖ Ağ-tabanlı uygulamalara olanak verir.
 - ❖ Daha karmaşık uygulamaların sağlanmasına olanak verir.

Neredeyiz?

► Buraya kadar edindiğimiz bilgilere (enine dinamik I ve II, MADX I dersleri) dayanarak:

Alıştırma 1

❖ Düzenli bir örgü hesaplarını ve tasarımını yapabiliriz.

Alıştırma 2

❖ Temel hızlandırıcı parametreleri ile oynayabiliriz (ayar, renksellik, beta fonksiyonu, ...).

► Bundan sonrası için bizi neler bekliyor?

❖ Hızlandırıcılarda bulunabilecek kusurlar ve bunların düzeltilmesi.

❖ Dağılım bastırıcı tasarımı.

❖ Çok düşük beta fonksiyonu gerektiren eklentilerin düzenli örgü içine yerleştirilmesi.