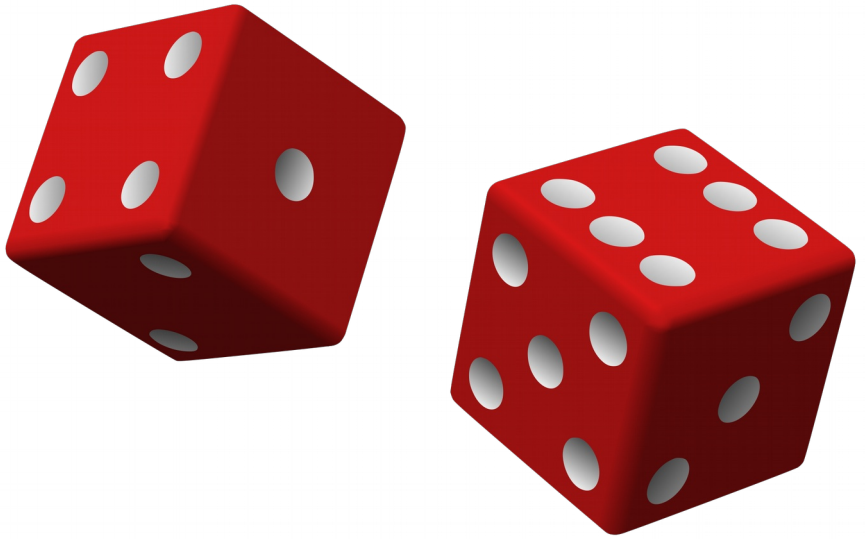
Monte Carlo Methods

Applications of dice throwing in particle physics and....



….astrodynamics

THE UNIVERSITY OF ARIZONA

David Yaylali
Department of Physics

April 19, 2017

Outline of Topics:

- Quick bit of history
- Monte Carlo Integration
- Monte Carlo in simulating particle physics interactions
- Monte Carlo in modeling error in non-linear systems
  -Transfers to Lagrange points using invariant manifolds

Monte Carlo Methods in General

Casino de Monte-Carlo, Monaco

Computational algorithms which typically utilize (pseudo)random number sampling.

Can solve problems that may be deterministic in principle or truly random in principle (quantum behavior).

- Define domain of input parameters involved
- Draw random values of input parameters based on probability distribution
- Perform deterministic computation on these random inputs
- Aggregate (average) results to find desired answer

We'll illustrate this in what follows

Output may be incorrect with some small probability. Variance of outcome about true mean decreases with larger sample numbers

Related Algorithms:

- Las Vegas: calculation time varies randomly, but correct result achieved
- Atlantic City: correct result achieved 75% (or X%) of the time.

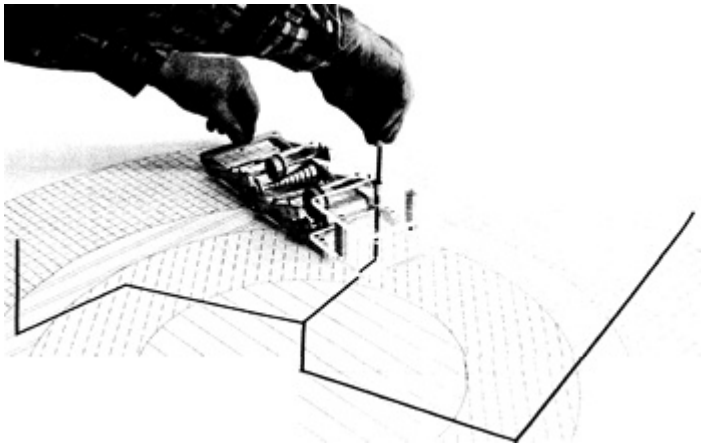
Bit of history....

First formally developed and used during the Manhattan project.

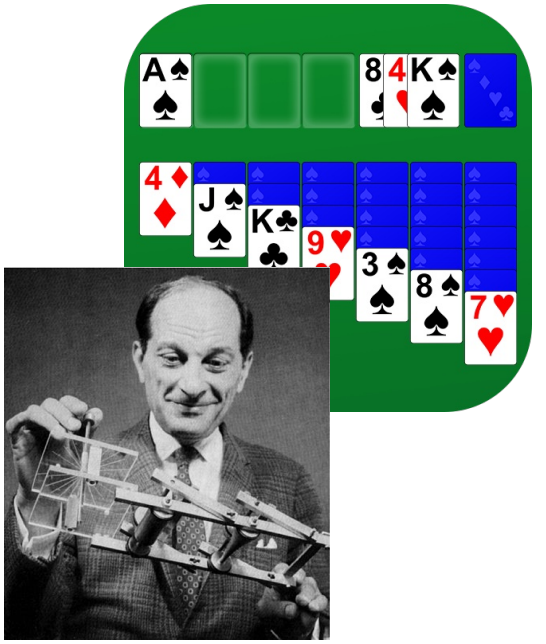
- While playing solitaire, Ulam attempted to estimate the probability of winning. *No clear way of calculating analytically.* Ulam's idea:
  - Estimate this probability by playing 1000s of games.
  - With recent development of computer (ENIAC), this approach became practical.

- Idea developed and refined along with von Neumann and Metropolis. Algorithms coded into ENIAC to compute a number of neutron transport problems.

  Common lore: Named "Monte Carlo" as a code word, to keep secrecy during wartime



FERMIAC analog MC machine, being used to model neutron transport before ENIAC was available



Stan Ulam with FERMIAC

von Neumann

Metropolis

Monte Carlo to calculate Pi

The Monte Carlo "Hello World!" program

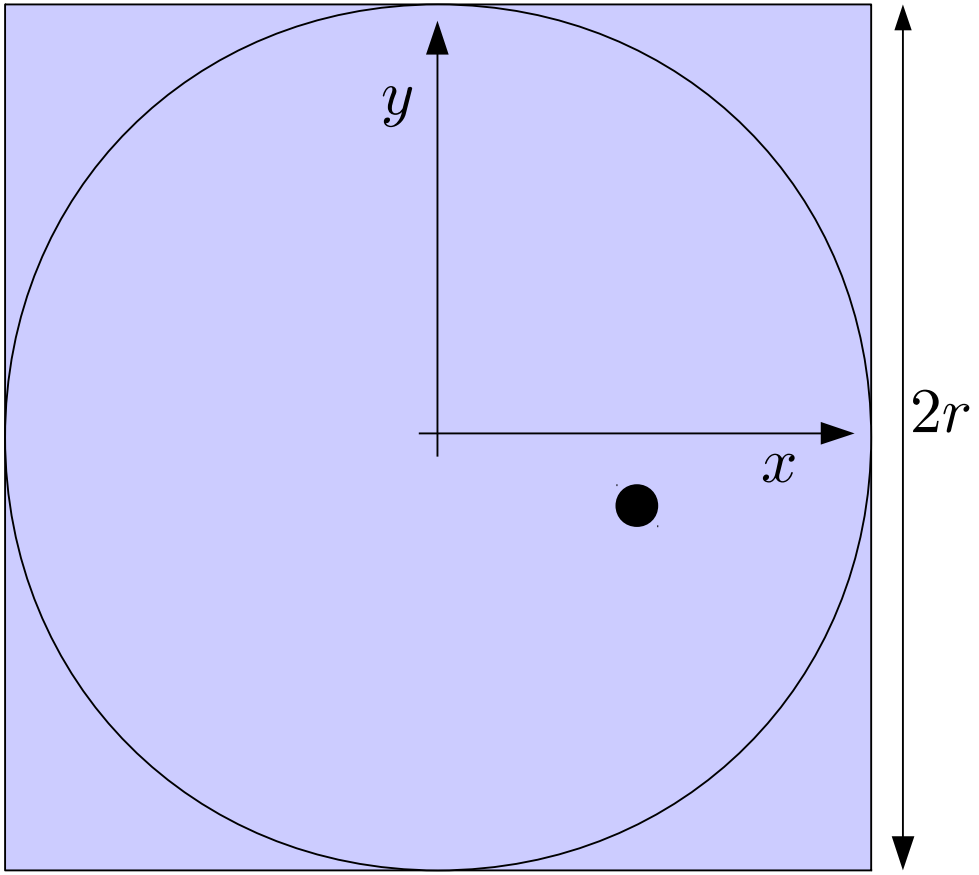
- Define a square region of area $2r \times 2r$
- Generate random coordinate (x,y) from uniform distribution:

$$\left.\begin{array}{l} x \in \mathrm{U}[-r, r] \\ y \in \mathrm{U}[-r, r] \end{array}\right\} \Rightarrow p_1 = (0.43r, -0.24r)$$

- Is point within the circle?

$$\sqrt{x^2 + y^2} < r$$

Yes ⇨ `nPass++`



Monte Carlo to calculate Pi

The Monte Carlo "Hello World!" program

- Define a square region of area $2r \times 2r$
- Generate random coordinate (x,y) from uniform distribution:

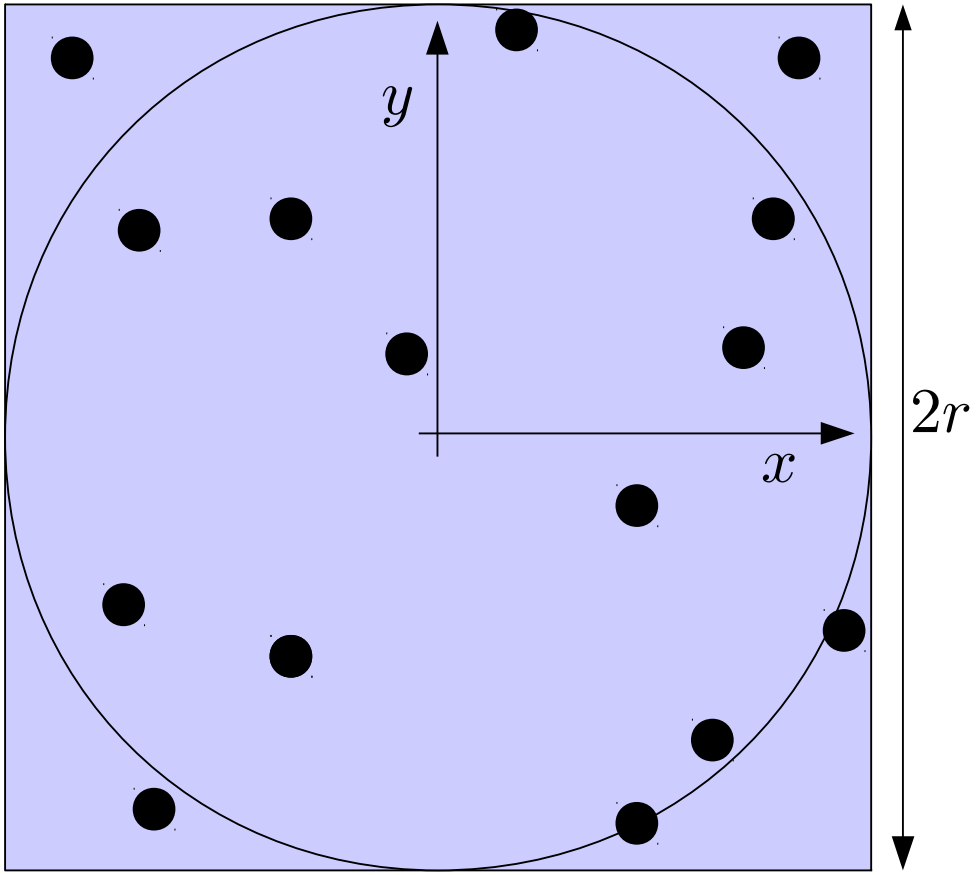
$$\left.\begin{array}{l} x \in \mathrm{U}[-r, r] \\ y \in \mathrm{U}[-r, r] \end{array}\right\} \Rightarrow p_1 = (0.43r, -0.24r)$$

- Is point within the circle?

$$\sqrt{x^2 + y^2} < r$$

Yes ⇨ `nPass++`

Repeat `nEvents` times....



Monte Carlo to calculate Pi

The Monte Carlo "Hello World!" program

- Define a square region of area $2r \times 2r$
- Generate random coordinate (x,y) from uniform distribution:

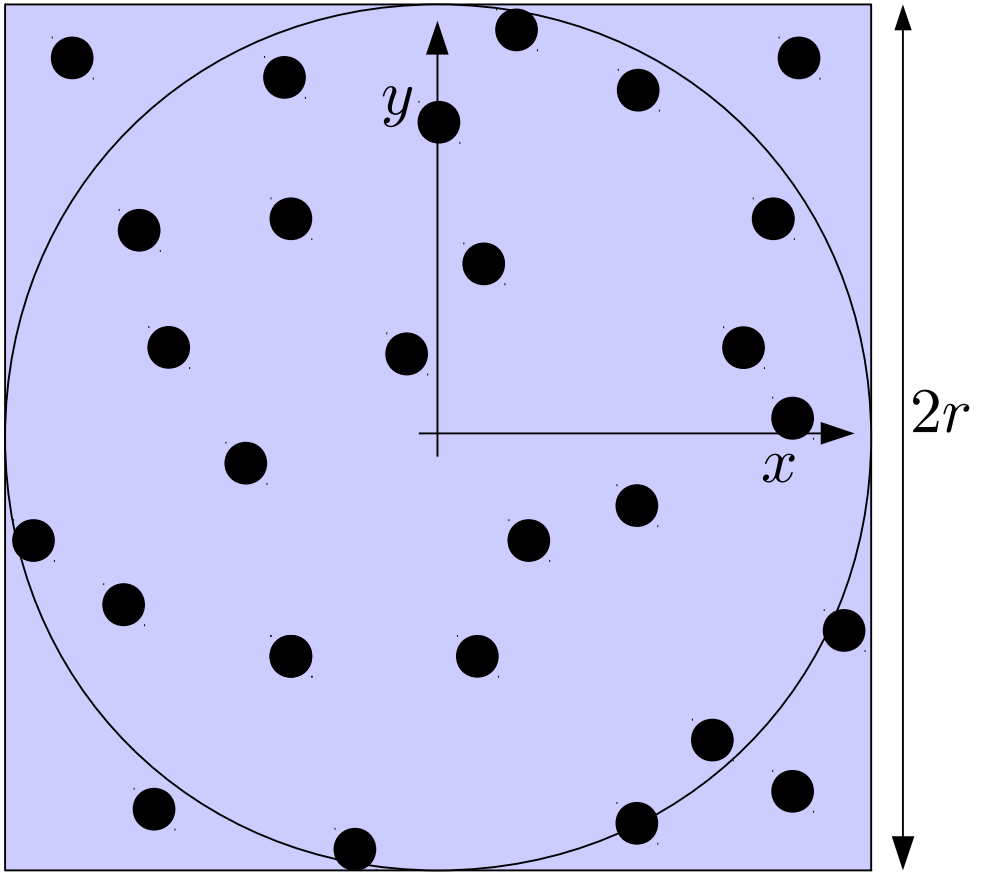
$$\left.\begin{array}{l} x \in \mathrm{U}[-r, r] \\ y \in \mathrm{U}[-r, r] \end{array}\right\} \Rightarrow p_1 = (0.43r, -0.24r)$$

- Is point within the circle?

$$\sqrt{x^2 + y^2} < r$$

Yes ⇨ `nPass++`

Repeat `nEvents` times....



Monte Carlo to calculate Pi

The Monte Carlo "Hello World!" program

- Define a square region of area $2r \times 2r$
- Generate random coordinate (x,y) from uniform distribution:

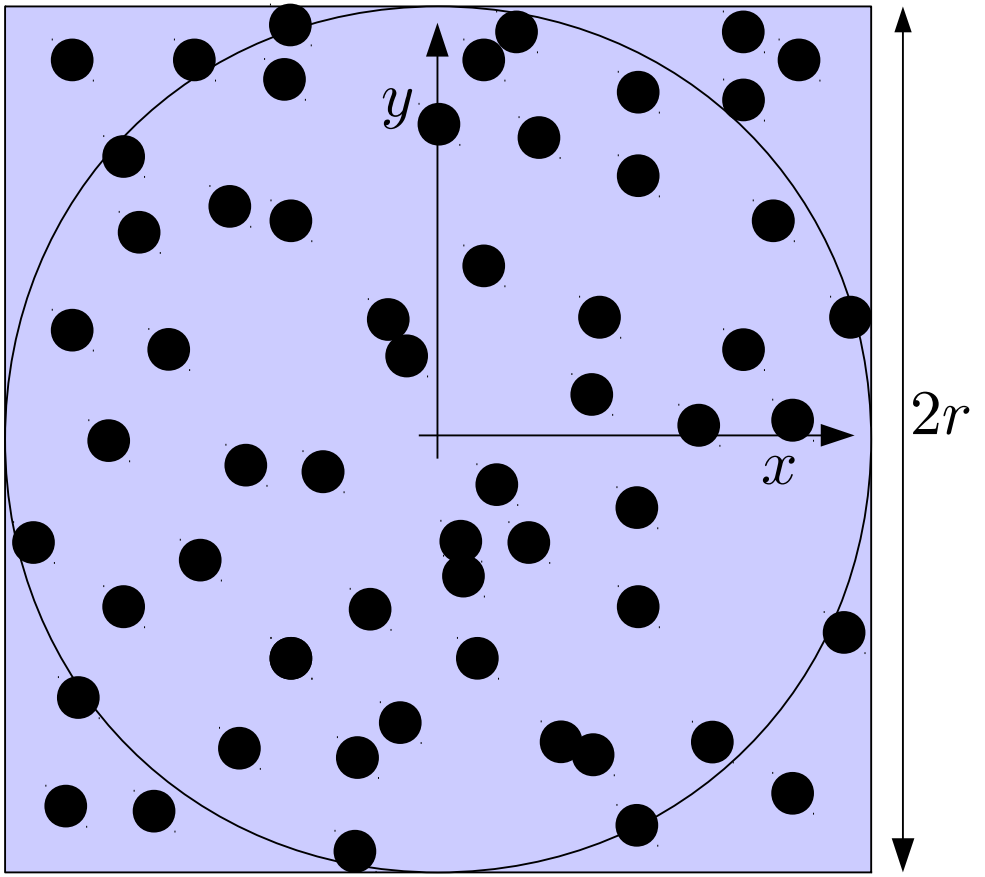
$$\left.\begin{array}{l} x \in \mathrm{U}[-r, r] \\ y \in \mathrm{U}[-r, r] \end{array}\right\} \Rightarrow p_1 = (0.43r, -0.24r)$$

- Is point within the circle?

$$\sqrt{x^2 + y^2} < r$$

Yes ⇨ `nPass++`

Repeat `nEvents` times....



Monte Carlo to calculate Pi

The Monte Carlo "Hello World!" program

- Define a square region of area $2r \times 2r$
- Generate random coordinate (x,y) from uniform distribution:

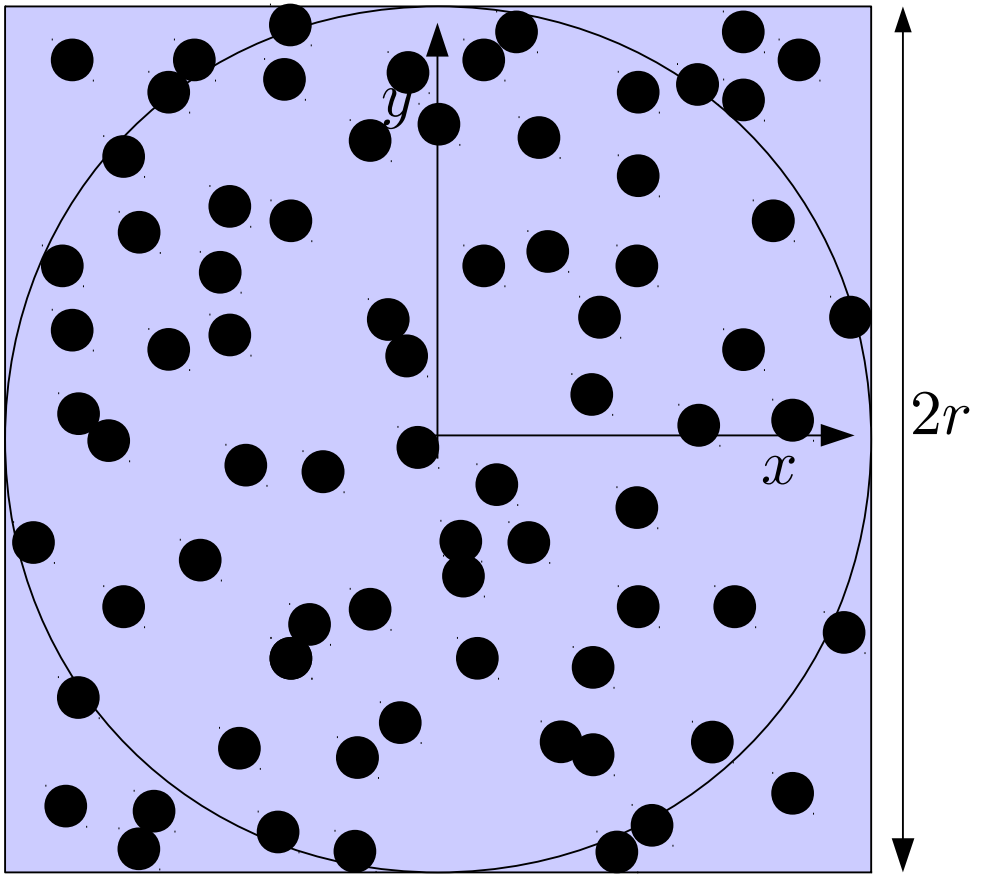
$$\left.\begin{array}{l} x \in \mathrm{U}[-r, r] \\ y \in \mathrm{U}[-r, r] \end{array}\right\} \Rightarrow p_1 = (0.43r, -0.24r)$$

- Is point within the circle?

$$\sqrt{x^2 + y^2} < r$$

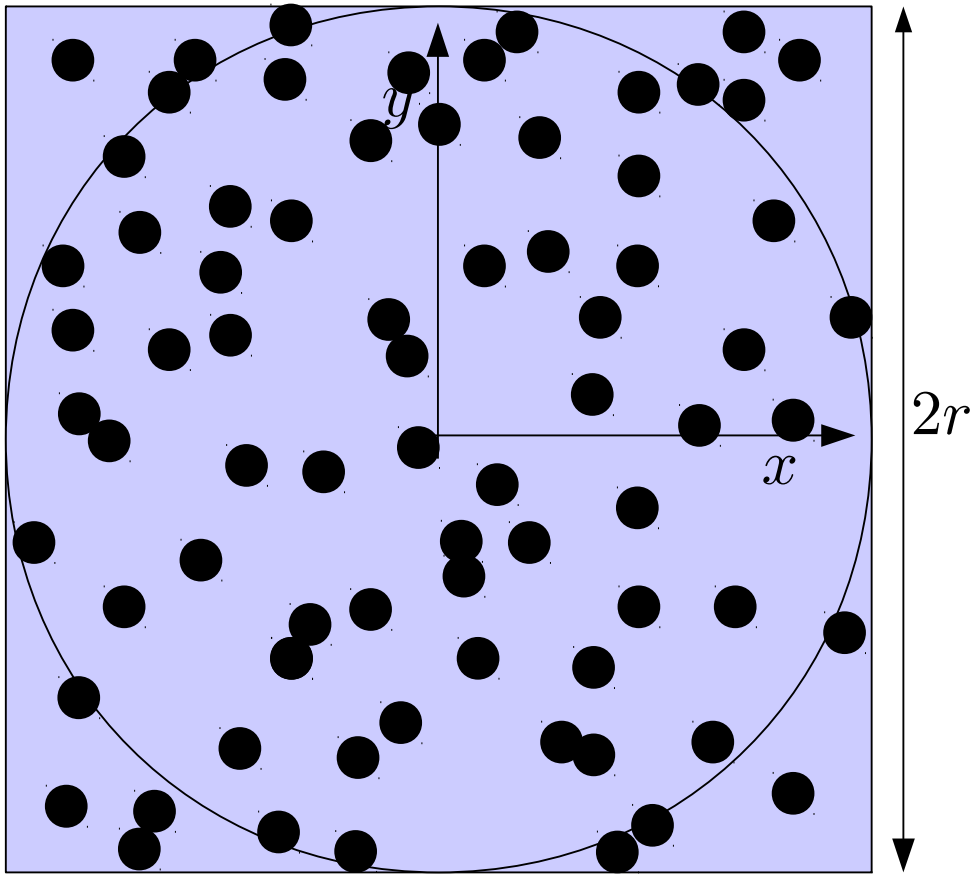
Yes ⇨ `nPass++`

Repeat `nEvents` times....



Monte Carlo to calculate Pi

The Monte Carlo "Hello World!" program



- Define a square region of area $2r \times 2r$
- Generate random coordinate (x,y) from uniform distribution:

$$\left.\begin{array}{l} x \in \mathrm{U}[-r, r] \\ y \in \mathrm{U}[-r, r] \end{array}\right\} \Rightarrow p_1 = (0.43r, -0.24r)$$

- Is point within the circle?

$$\sqrt{x^2 + y^2} < r$$

Yes ⇨ `nPass++`

Repeat `nEvents` times....

$$\frac{\pi r^2}{(2r)^2} \approx \frac{\texttt{nPass}}{\texttt{nEvents}} \quad \Rightarrow \quad \pi \approx 4\frac{\texttt{nPass}}{\texttt{nEvents}}$$

File Edit Selection Find View Goto Tools Project Preferences

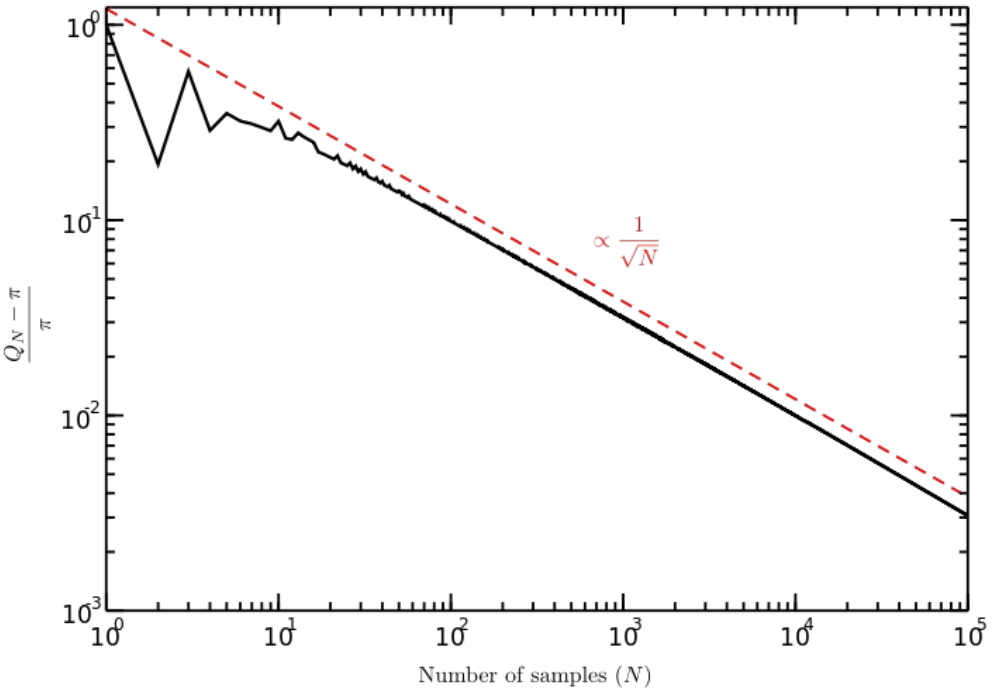
animateJC.m ● PiMC.py ×

```
1  #!/usr/bin/python
2  import random
3  import math
4
5  totEvents = 100000
6  nEvent = 0
7  nPass = 0
8  r = 1
9
10
11 for i in range(totEvents):
12     nEvent += 1
13     x = r*random.uniform(-1,1)
14     y = r*random.uniform(-1,1)
15     if x**2 + y**2 < r:
16         nPass += 1
17
18
19 piCalc = 4.0*nPass/nEvent
20
21
22 print "Events:", nEvent
23 print "Points within R: ", nPass
24 print
25 print "Pi: ", piCalc
26 print "Error: ", 100*abs(piCalc-math.pi)/math.pi, "%"
27 print
28
```

```
Events: 100000
Points within R:  78552

Pi:  3.14208
Error:  0.0155127180365 %

[Finished in 0.2s]
```

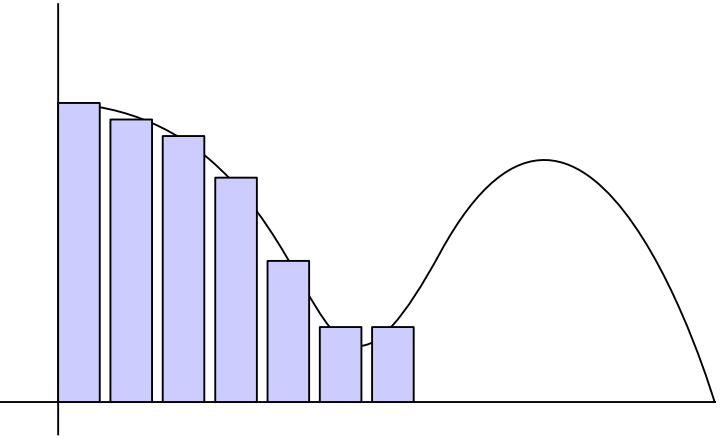


General result: From central limit theorem, MC method displays $1/\sqrt{N}$ convergence.

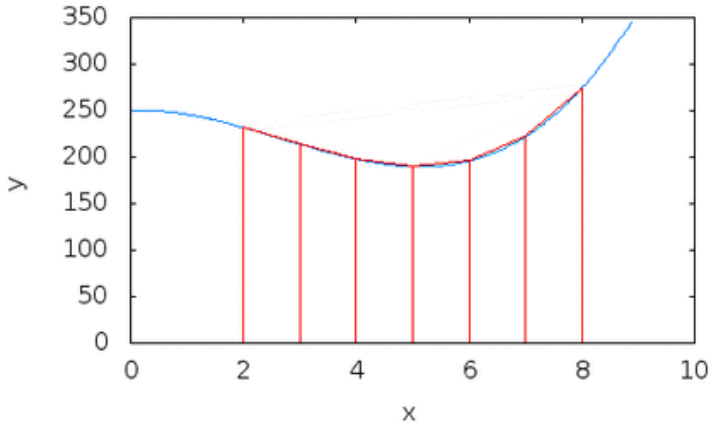
Monte Carlo Integration

A method to numerically compute large dimensional integrals.

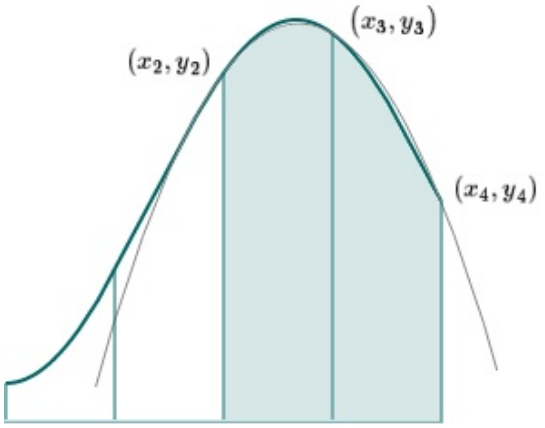
"Deterministic" numerical integration approaches



Midpoint Rule



Trapazoidal rule



Simpson's Rule

For midpoint rule: Evaluate function at each point on some predetermined grid.

$$\int_0^x f(x')dx \approx \sum_{i=1}^{n} f(x_i)\Delta x$$

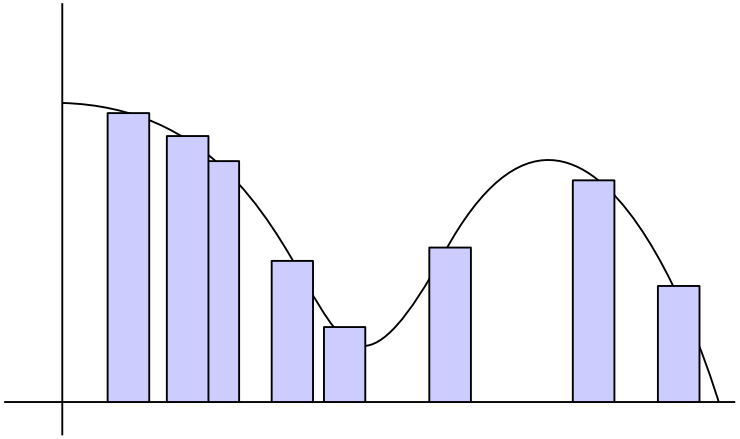
Integral Dimension	Grid Points
1	n
2	n^2
$\vdots$	$\vdots$
d	n^d

Number of required function evaluations grows geometrically with the dimension of the integral. ***Quickly becomes prohibitive***

Monte Carlo Integration

A method to numerically compute large dimensional integrals.

Illustration: Simple MC numerical integration



- Select random value from uniform distribution over the domain:

$$x_1 \in \mathrm{U}[a,b]$$

- Evaluate the function at this value

$$f(x_1)$$

- Repeat N times, average results:

$$\int_a^b f(x)dx \approx V\frac{1}{N}\sum_{i=1}^{N} f(x_i) = V\langle f\rangle$$

...with $V = \int_a^b dx$

```
#!/usr/bin/python
import random
import math

totEvents = 100
nEvent = 0
nPass = 0

ysum = 0

for i in range(totEvents):
    nEvent +=1
    x = random.uniform(0,3)
    y = x**2
    ysum = ysum+y

yInt = 3* ysum / nEvent


print "Events:", nEvent
print "ysum: ", ysum
print "Integral of x^2: ", yInt
```

```
Events: 100
ysum:  308.484534068
Integral of x^2:  9.25453602204
[Finished in 0.0s]
```

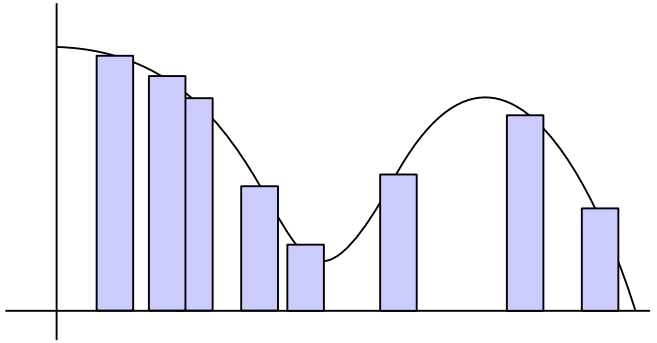
Error in the result (standard deviation of the result about the mean) $\propto 1/\sqrt{n}$, **regardless of the dimension of the integral.**

Monte Carlo Integration

A method to numerically compute large dimensional integrals.

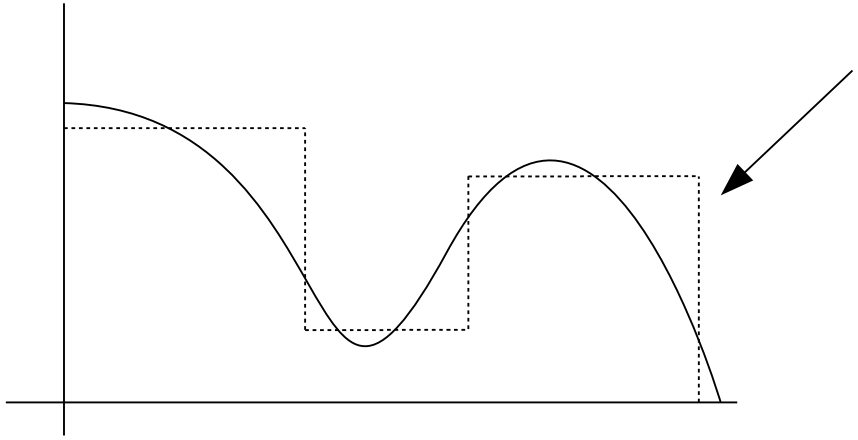
Importance sampling:

Basic idea: Values where the integrand is large contribute more to the integral.



$$\int_a^b f(x)dx \approx V\frac{1}{N}\sum_{i=1}^{N} f(x_i) = V\langle f\rangle$$

Better to sample preferentially in these areas. Use new PDF which is weighted by the function.



$$\int_a^b p(x)dx = 1$$

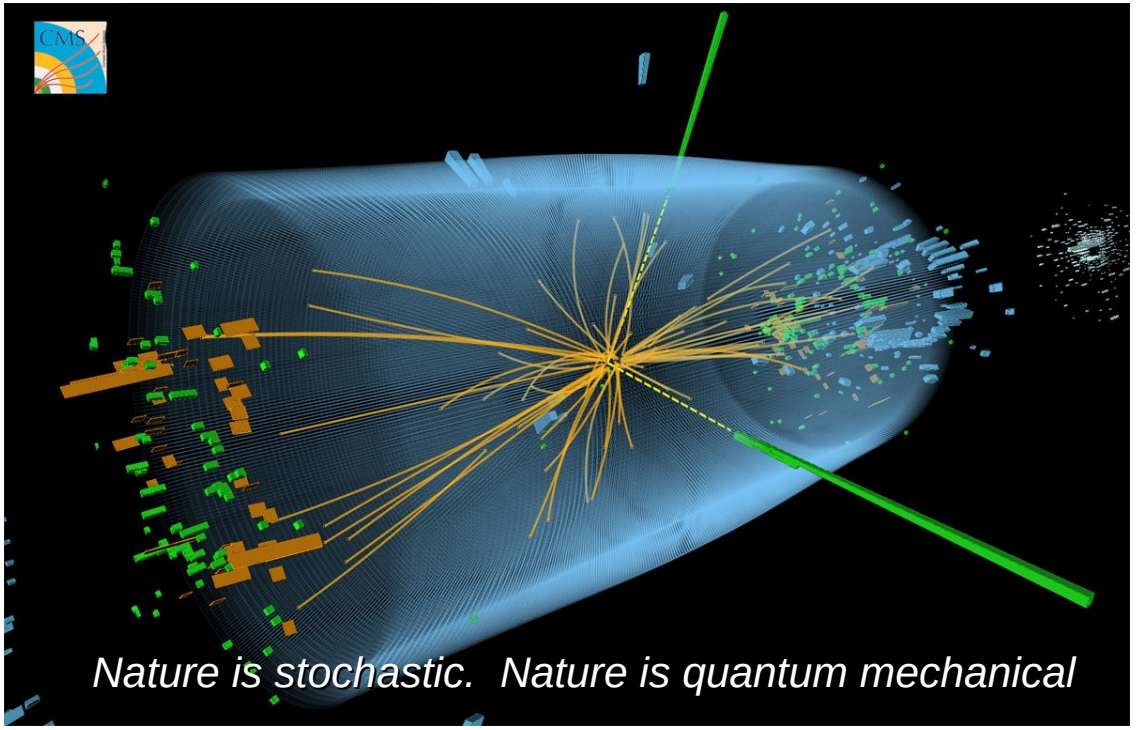
$$\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

Note that this reduces to the simple case above when p(x) is constant.

- This is essentially equivalent to **“refining” the grid** in standard grid numerical integration.
- Many algorithms which can find p(x) --- refine integration grid --- based on lower-N “survey runs”.

Monte Carlo in Particle Physics

Using Monte Carlo sampling to model stochastic processes



Applications to searches for dark matter at the Large Hadron Collider

Based on ongoing work done in collaboration with K. Dienes (UA), S. Su (UA), and B. Thomas (LaFayette)

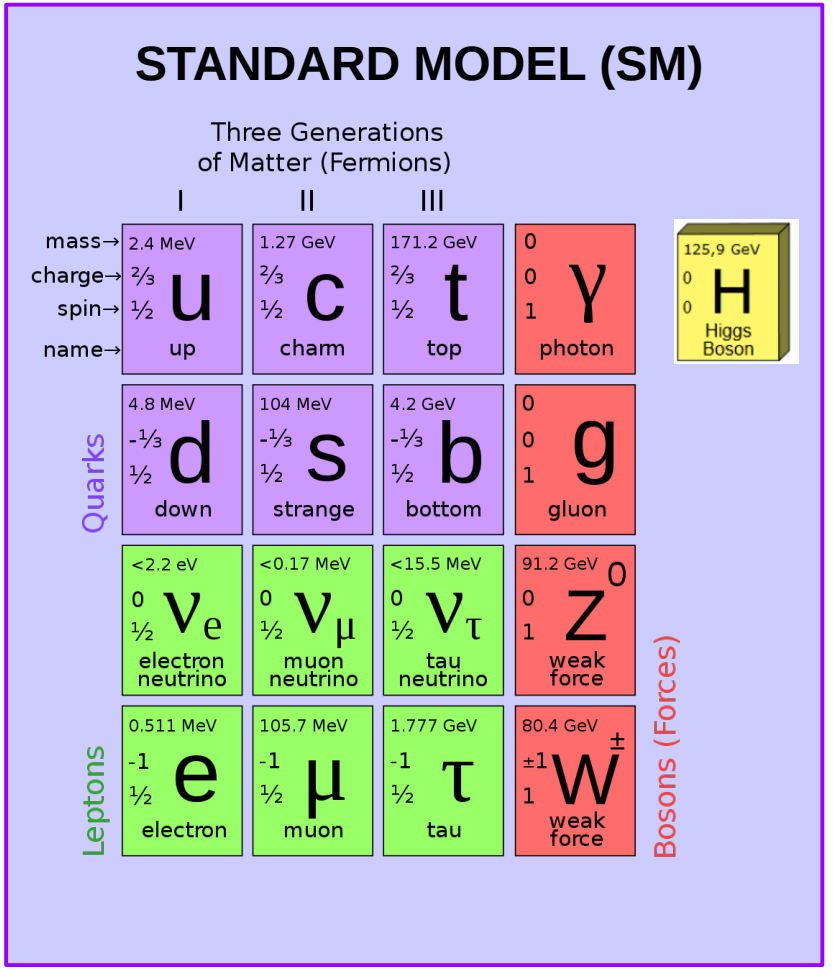
Some necessary background: Dark Matter

Dark Matter: Stable particles present in the universe which is not made up of the known “Standard Model” particles.

- Does not interact with photons/light (hence “dark”) ↔ not electrically charged
- Interacts extremely weakly with regular matter. Almost “ghost-like”.
- We know almost no details about it's properties (Mass, spectrum, forces)

Caveats...

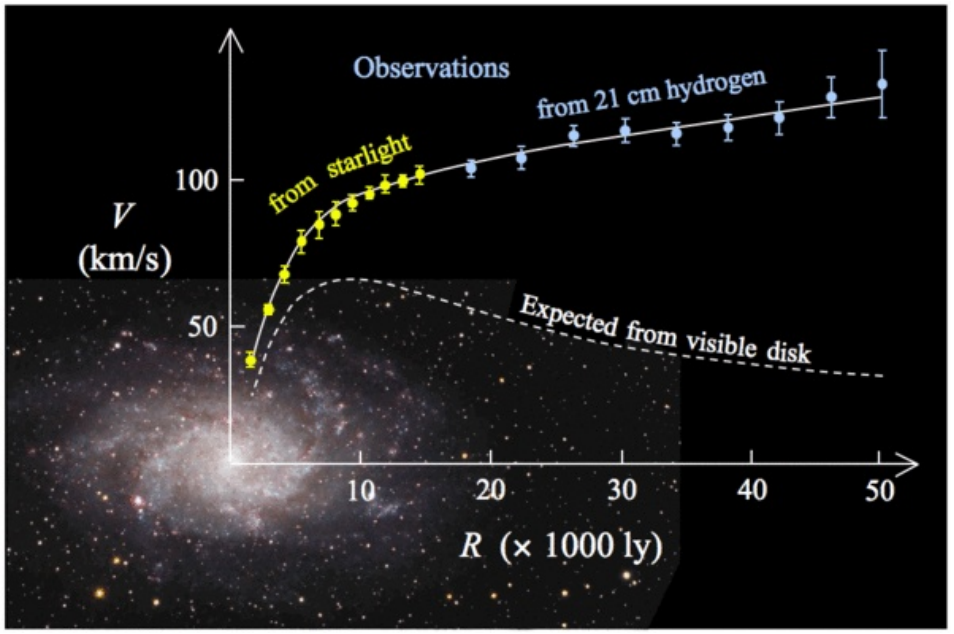
- ➔ Quantum Mech/Relativity restrict the possibilities of what the dark matter is made of.
- ➔ Certain types are “well-motivated”, theoretically.



If invisible, how do we know it's there?

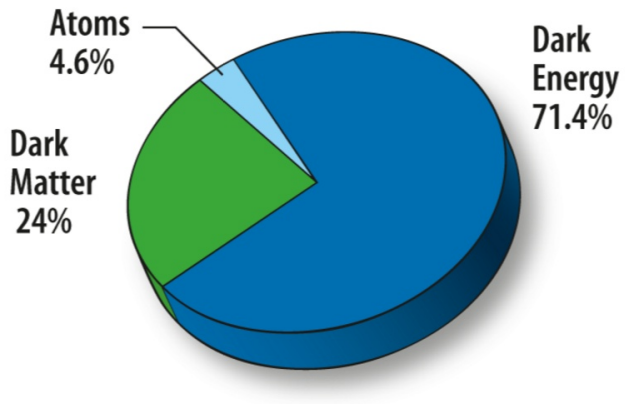
The existence of dark matter in our universe is at this point very well established.

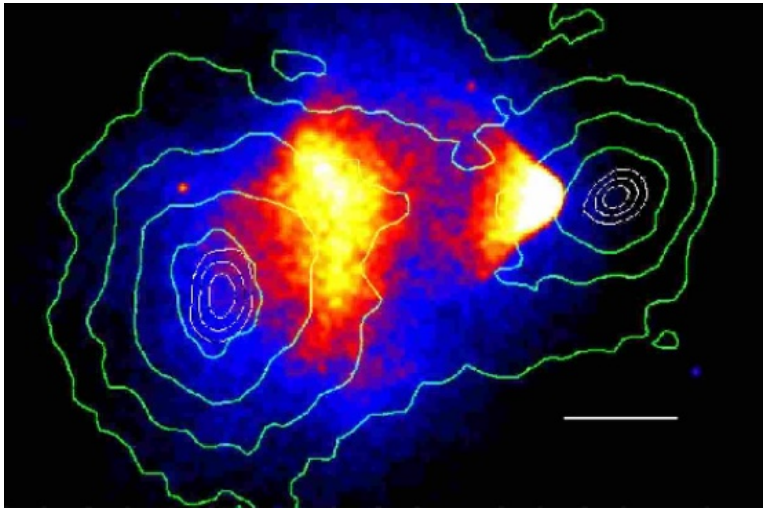
- *Local stellar velocities (Oort, 1932)*
- *Velocities of galaxies in clusters (Zwicky, 1933)*
- *Galactic rotation curves (Rubin, 1960's)*
- *Position/ratio of acoustic peaks in the CMB*
- *Large scale structure – N-body simulations*
- *Lensing in the Bullet Cluster*



All separate phenomena point to

$$m_{DM}/m_{SM} \approx 5$$

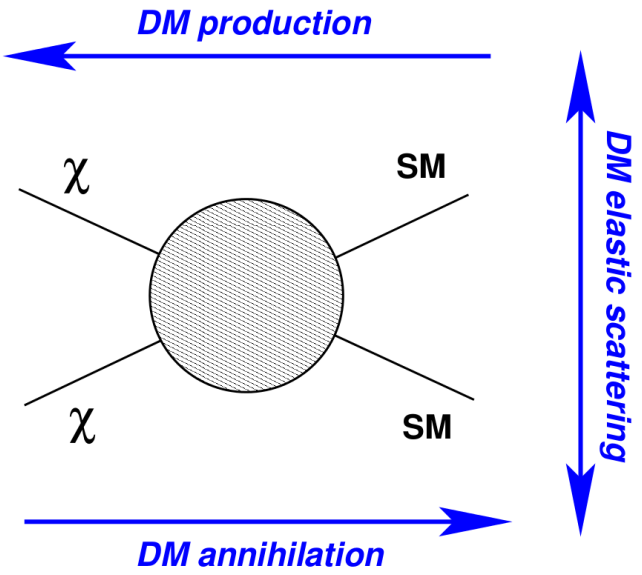


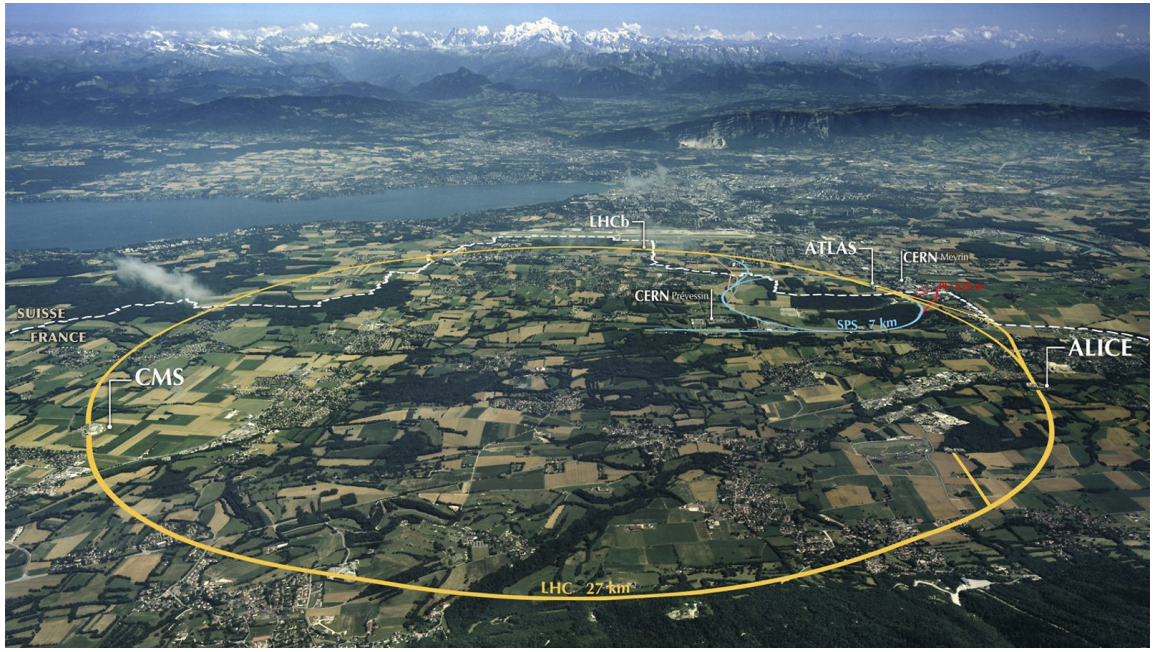


Implications

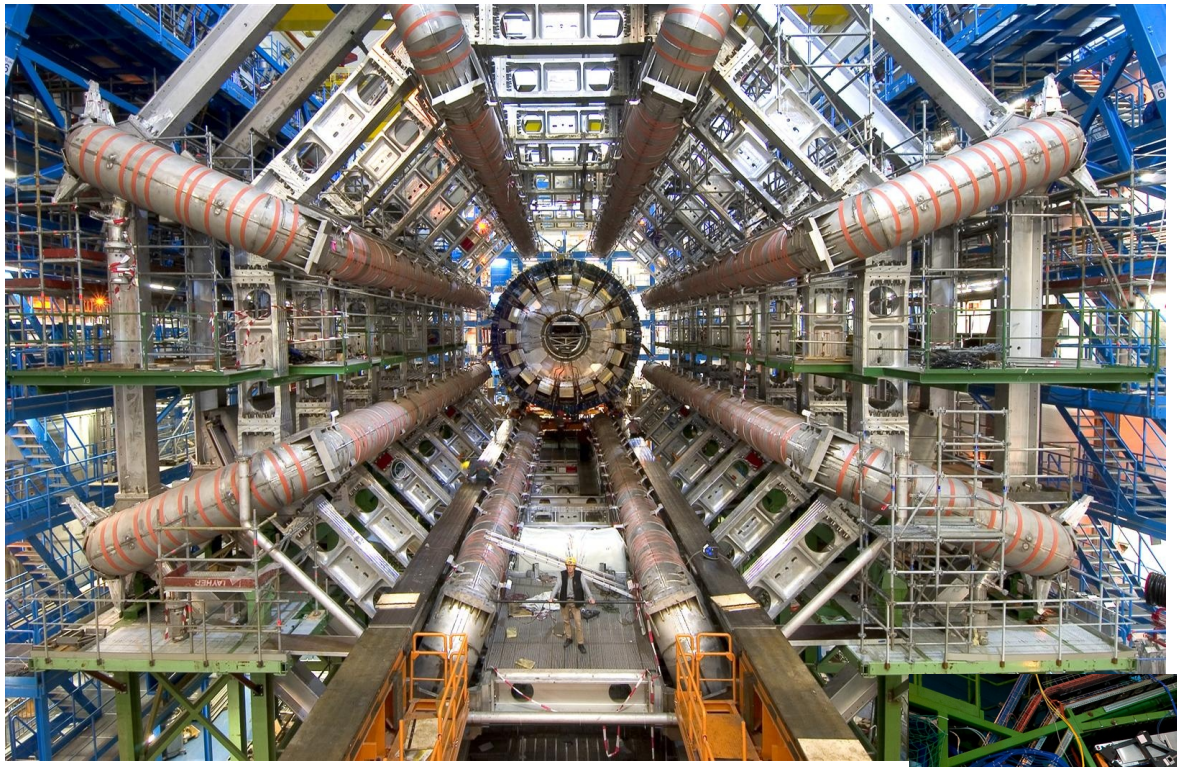
- Dark matter exists
- DM particles interact very weakly with the stuff we're made out of.
- We may be able to detect it with very sensitive instruments
- We may be able to ***create it***

If we have enough energy, we can smash two particles together and potentially create DM pairs at colliders.

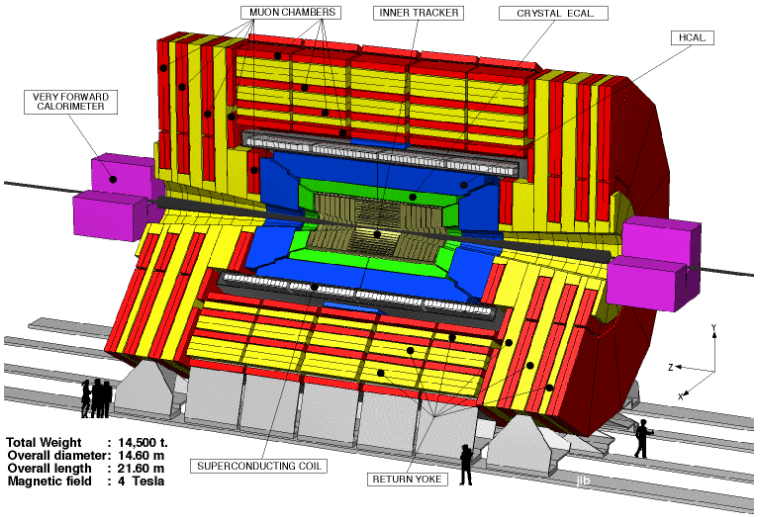




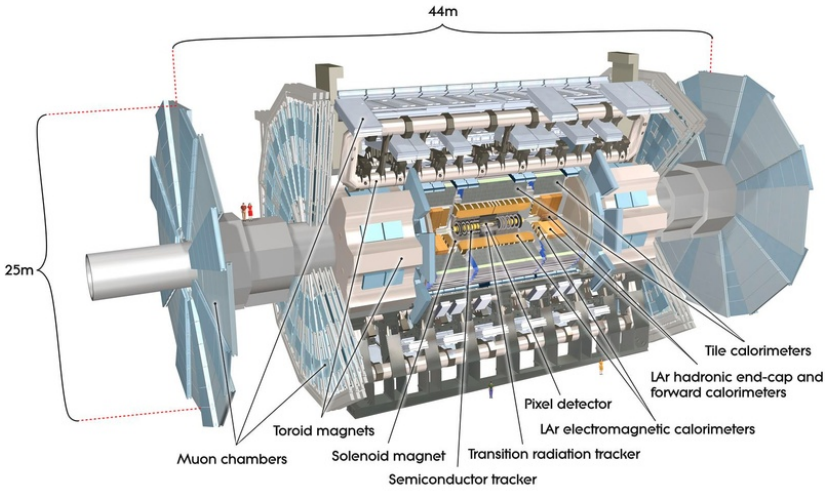
LHC Detectors

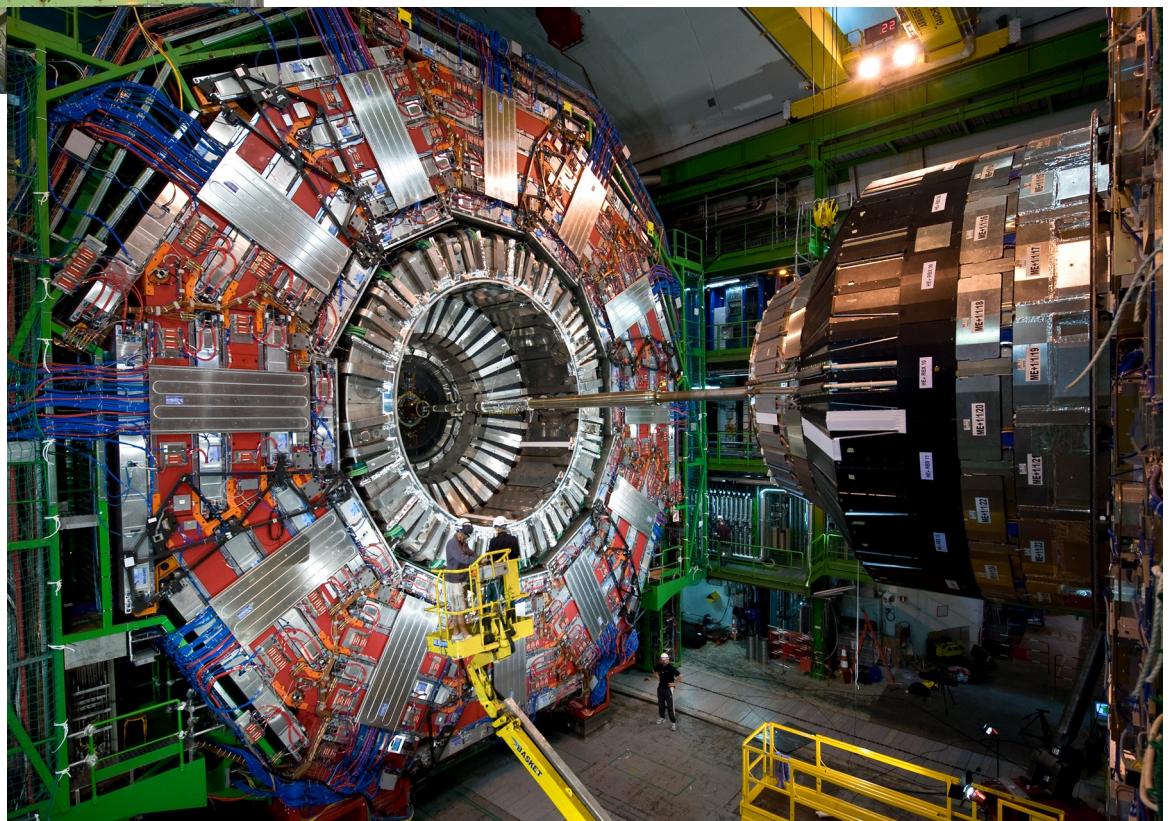


ATLAS



CMS



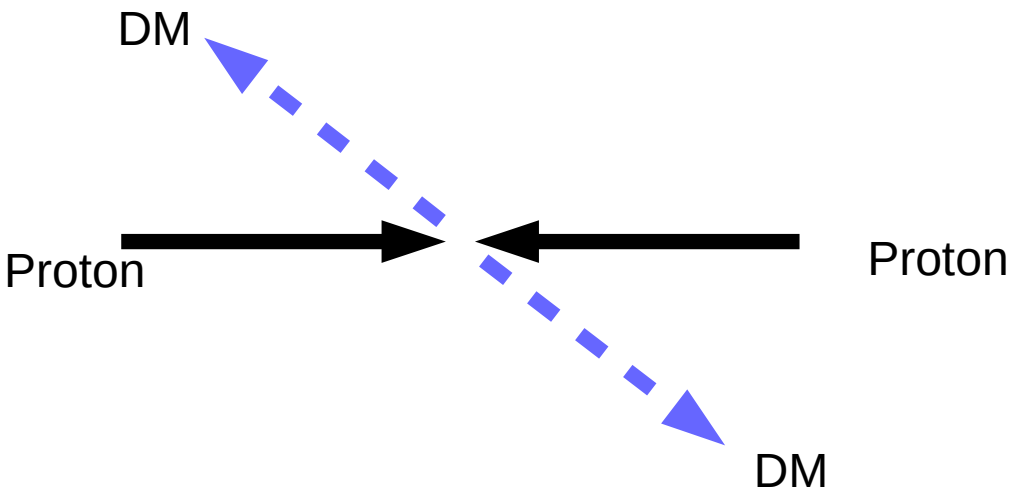


Dark Matter production at colliders

LHC Proton Beams

Bunches per proton beam: 2808
Protons per bunch: 1.2×10^{11}
Collisions per second: 10^9
Energy per collision: $13\ \mathrm{TeV}$
(~13,000 protons)

By sheer numbers, LHC is able to produce extremely rare events such as DM-SM interactions...



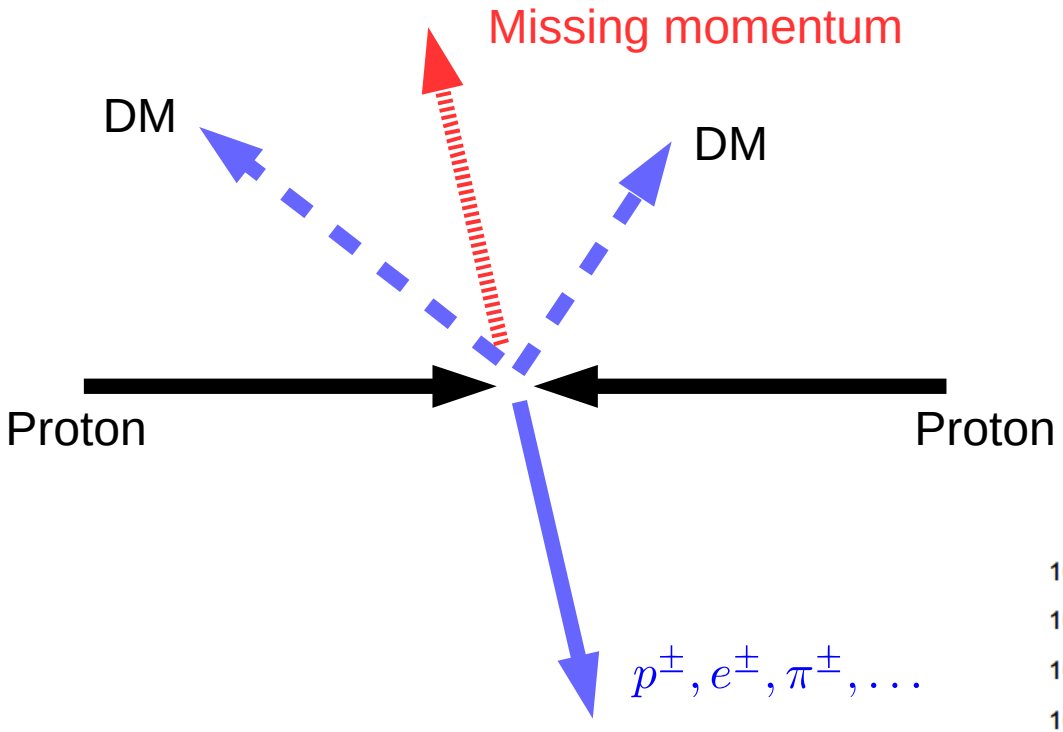
As long as the energy in the beam is such that

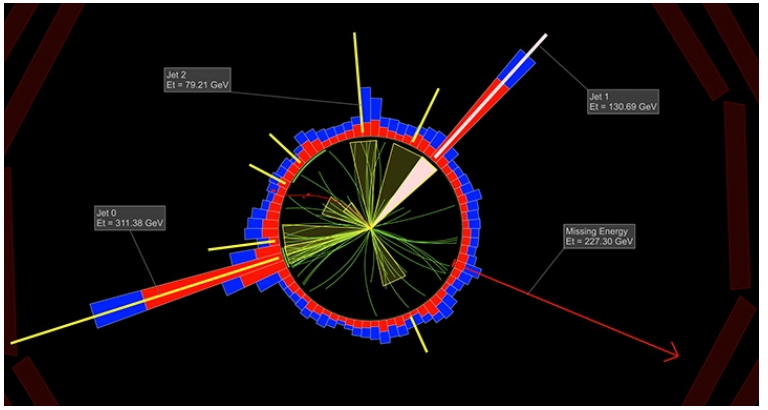
$$E_{\mathrm{c.o.m.}} \geqslant 2m_{\mathrm{DM}}$$

…we can produce DM pairs at a collider.

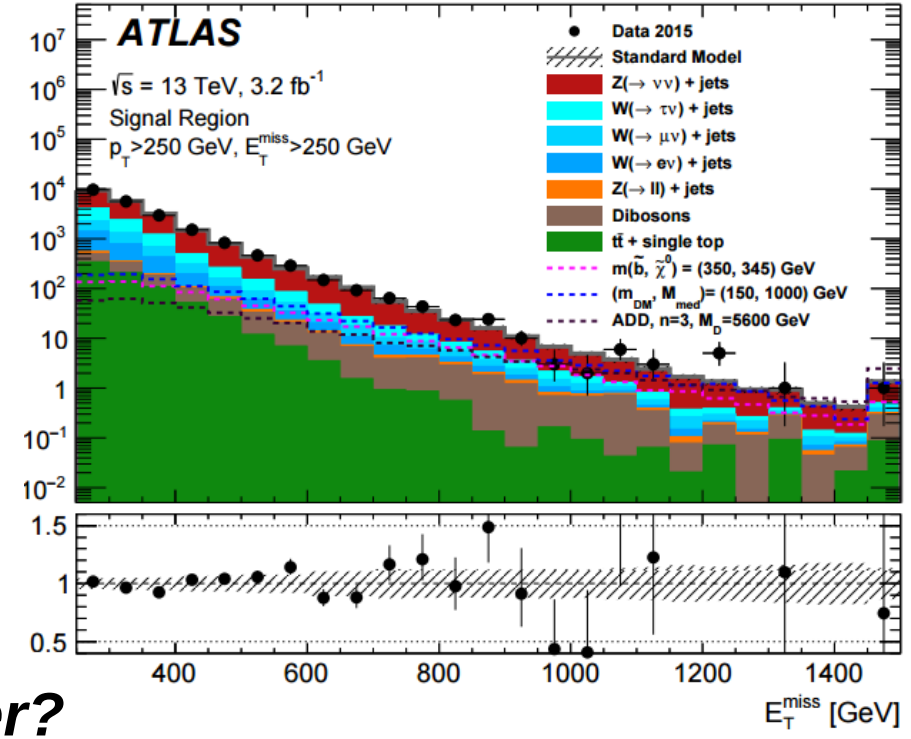
Problem: DM interacts extremely weakly... Invisible once created.... ***detectors do not register anything!***

Solution: Focus on collision events where DM and regular matter are produced together.





However, there are a ton of other processes which give rise to missing energy at colliders...



How are we sure that events with missing energy is actually dark matter?

The task is to predict what a DM production event would look like at the LHC in terms of outgoing visible particles and missing energy.

This allows us to...

- compare actual observations to these predictions.
- claim evidence of discovery, or exclude a model with some statistical significance.

Relevant points for this talk:

1) Particle interactions obey quantum mechanics (**naturally stochastic**)
2) At colliders, initial conditions (momenta of protons) are **not precisely known**
3) A **ridiculously large number** of interactions occur per second.

→ *Particle Physics RELIES on Monte Carlo methods to make predictions*

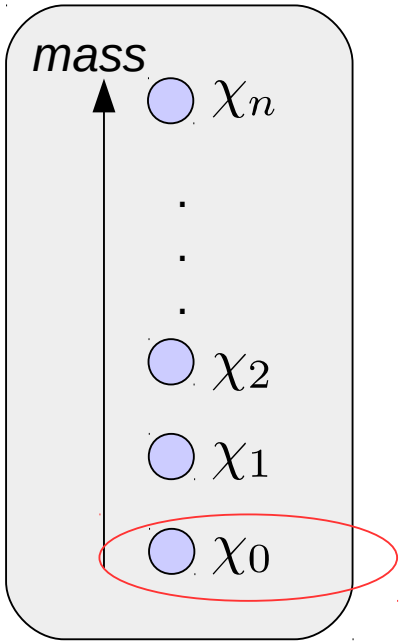
→ *Monte Carlo methods are naturally suited for these types of problems*

To illustrate, I'll outline a specific model of DM that I have been studying and describe the Monte Carlo “LHC simulator” I've built.

Theoretically Motivated Model: Dynamical Dark Matter

(K.Dienes and B.Thomas, arXiv:1106.4546)

New "tower" of exotic particles. Only the lightest one, χ_0, is stable.



Dark matter candidate

$$m_n = m_0 + n^\delta \Delta m$$

Probability for χ_n to decay to χ_m :

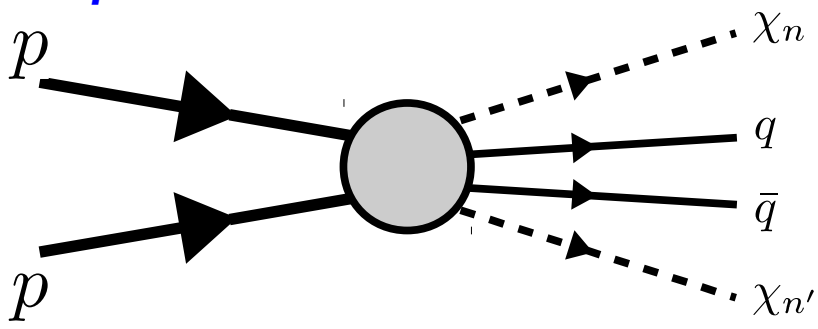
$$P(\chi_n \to \chi_m) \equiv BR(n,m)$$

"branching ratio"

$$\sum_{m=0}^{n-1} BR(n,m) = 1$$

*i.e., χ_n must decay to **some** lighter state.*

Collider production:



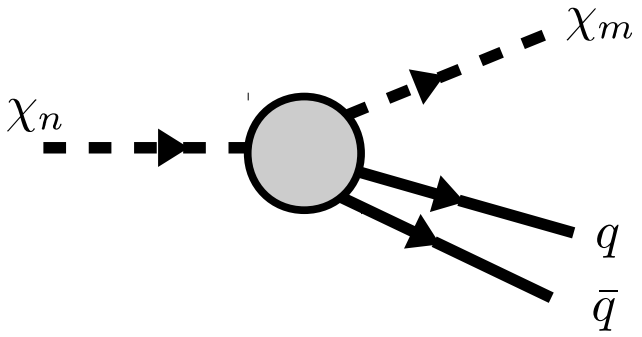
Protons can interact to produce pairs of the "mother" particle plus two "jets".

Probability for production: $\sigma(pp \to \chi_n \chi_{n'})$

"cross section"

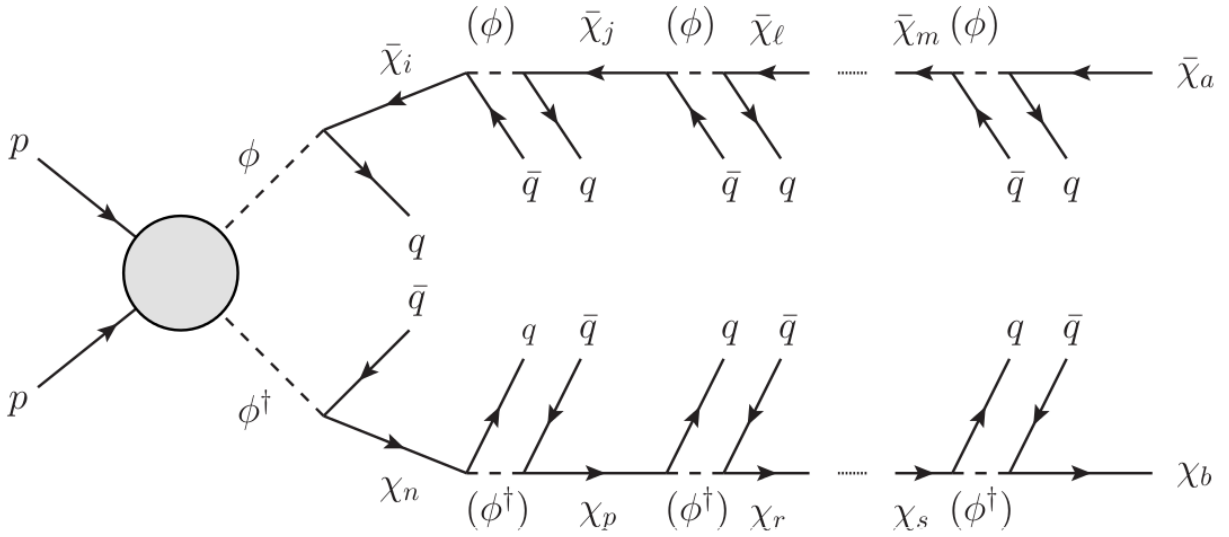
Particle decay:

A heavier state will quickly decay to a lighter state plus two visible "jets" (quarks).

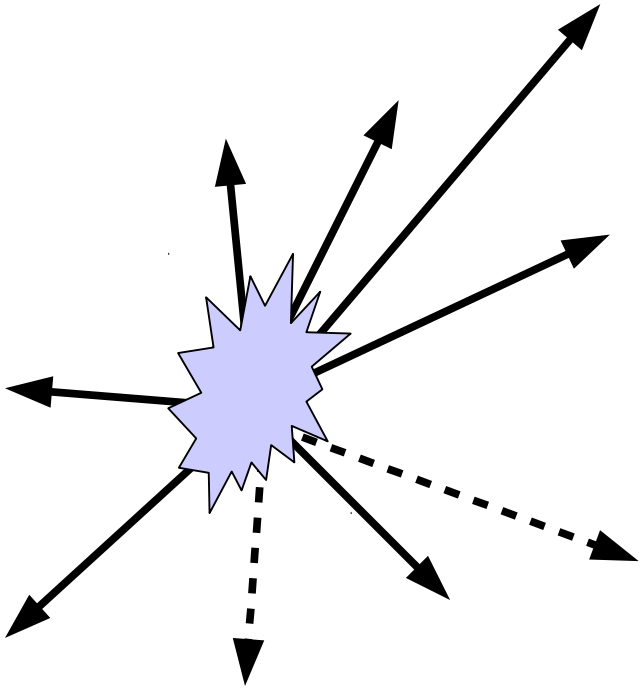


Dynamical Dark Matter Event Topology:

Mother particles will be produced, then decay down the chain to the χ_0 state.



- Each 'q' is a quark (or “jet”), which leaves a trace in the detector.
- The original χ_n decays down the chain to χ_0 **randomly**, according to the probabilities BR(n,m).
- This whole process occurs essentially instantly, producing a variable number of “jets” and missing energy (the χ_0)



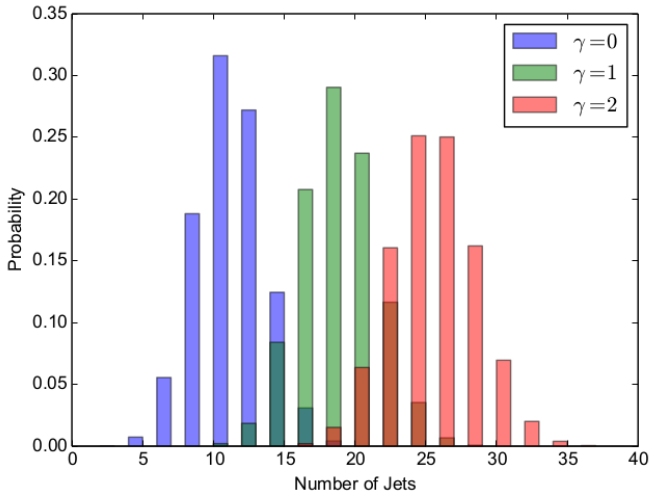
Two defining quantities in a DDM production event:

1) Number of jets

This probability distribution can actually be calculated analytically:

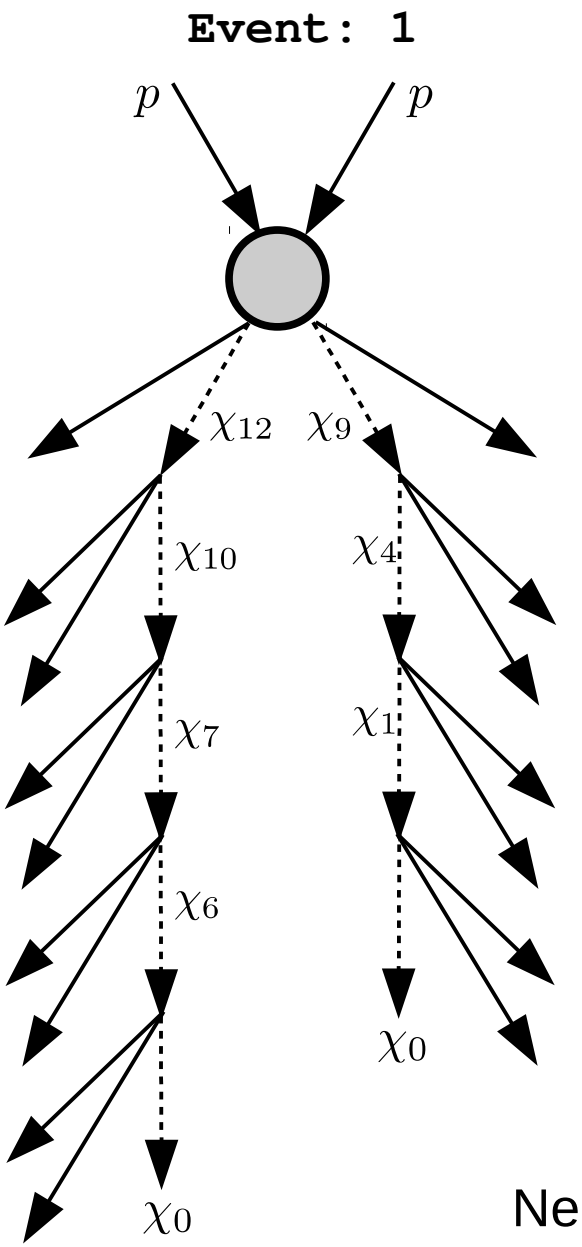
$$P(J,\chi_n) = \begin{cases} \mathrm{BR}(n,0) & \text{if } J=1, \\ \sum_{i=1}^{n-1} \mathrm{BR}(n,i)\times\mathrm{BR}(i,0) & \text{if } J=2, \\ \sum_{i_{(J-1)}=J-1}^{n-1} \mathrm{BR}(n,i_{(J-1)}) \left\{ \prod_{q=1}^{J-2} \left(\sum_{i_{(J-(q+1))}=J-(q+1)}^{i_{(J-q)}-1} \mathrm{BR}(i_{(J-q)},i_{(J-(q+1))}) \right) \right\} \mathrm{BR}(i_1,0) & \text{if } J>2. \end{cases} \tag{5.11}$$

- Straightforward, but tedious derivation.
- With this closed form, easy to compute using, e.g., Mathematica

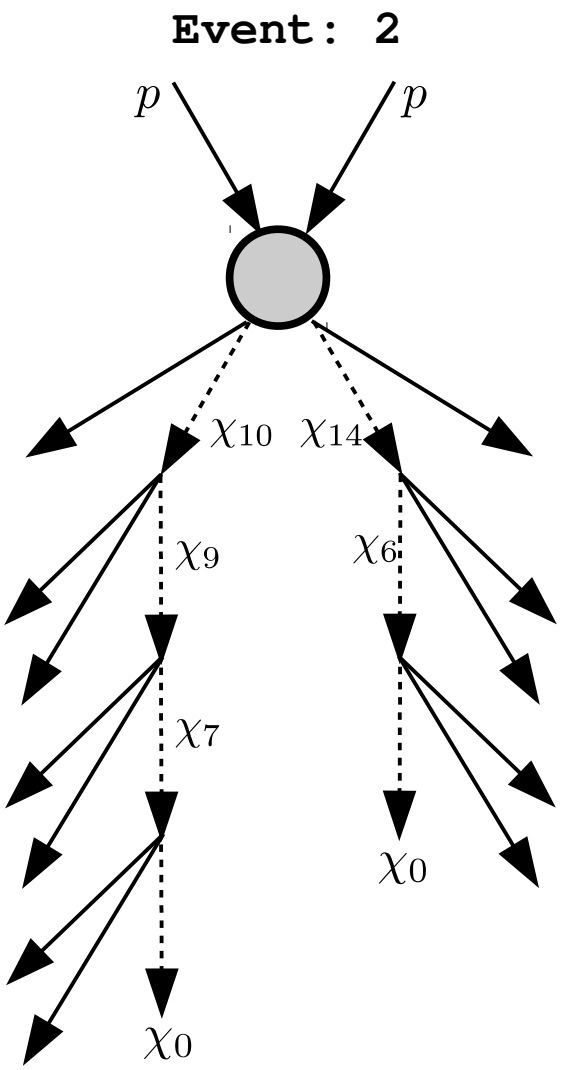


2) Missing energy and jet momentum distribution

Cannot be calculated analytically. Must use Monte Carlo methods



Variable number of jets, with differing energy and momentum distributions...



Need to predict what jet number and momentum distributions look like for search at LHC

PySwathMC

Cross sections and branching ratios:

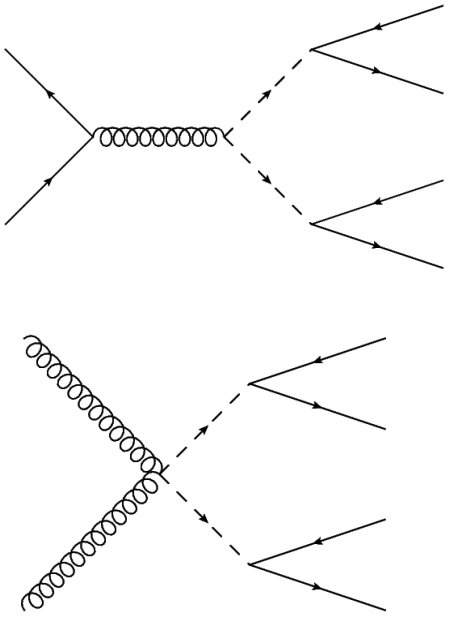
Given some choice of model parameters, one can directly calculate both cross section and all branching ratios.

Production Lagrangian:

$$\mathcal{L} = D_\mu \phi^\dagger D^\mu \phi$$

...with,

$$D_\mu \equiv \partial_\mu - ig\lambda_\alpha G_\mu^\alpha$$



Draw Feynman diagrams for process.

Calculate “amplitude” of process using QFT

$\sigma(gg \to \chi_n \chi_{n'})$

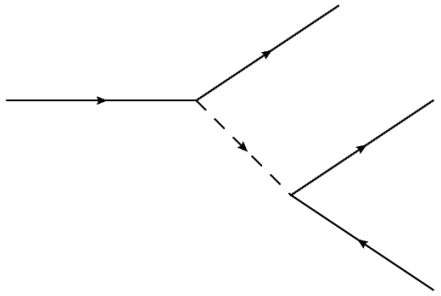
χ_n pairs actually produced by gluons within the proton

Cross sections and branching ratios:

Given some choice of model parameters, one can directly calculate both cross section and all branching ratios.

Decay Lagrangian:

$$\mathcal{L} = c_i \phi^\dagger \bar{\chi}_i P_R q$$



Draw Feynman diagrams for process.

Calculate “amplitude” of process using QFT

$$\Gamma(n,m)$$

$$BR(n,m) = \frac{\Gamma(n,m)}{\sum_{m=0}^{n-1} \Gamma(n,m)}$$

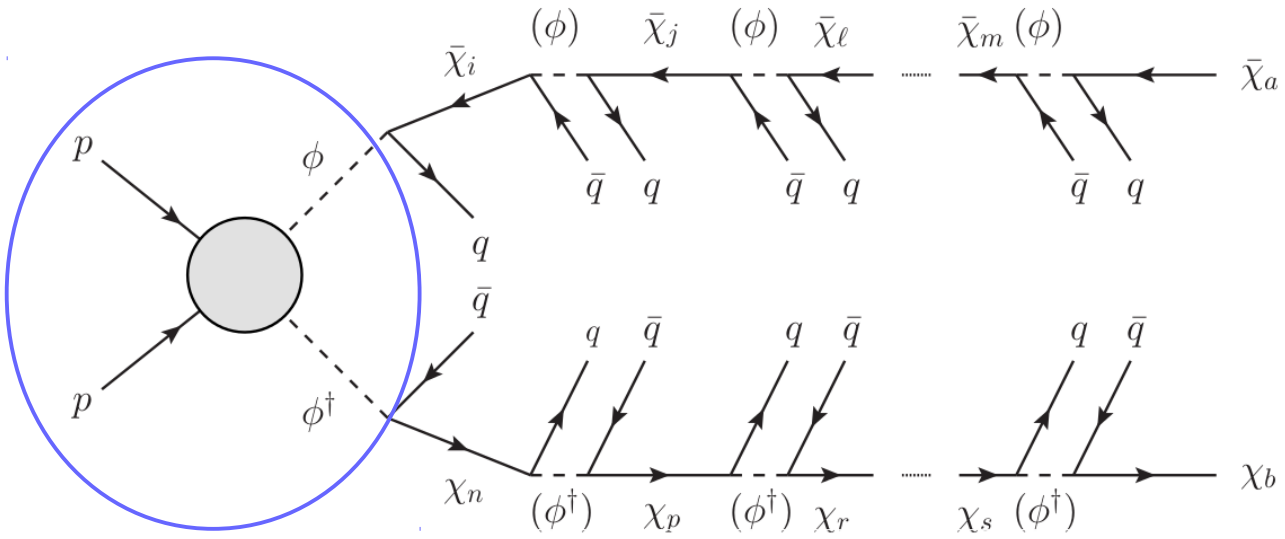
Final expressions messy...

$$\Gamma(\chi_n \to \bar{q}' q \chi_m) = \frac{3|c_{nq}|^2 |c_{mq'}|^2}{256\pi^2} \left(\frac{m_\phi^4}{m_n^3}\right) \left[f_1(m_n, m_m, m_\phi) + f_2(m_n, m_m, m_\phi) \ln\left(\frac{m_\phi^2 - m_n^2}{m_\phi^2 - m_m^2}\right) + f_3(m_n, m_m, m_\phi) \ln\left(\frac{m_n^2}{m_m^2}\right) \right]$$

...but numerically straightforward to compute

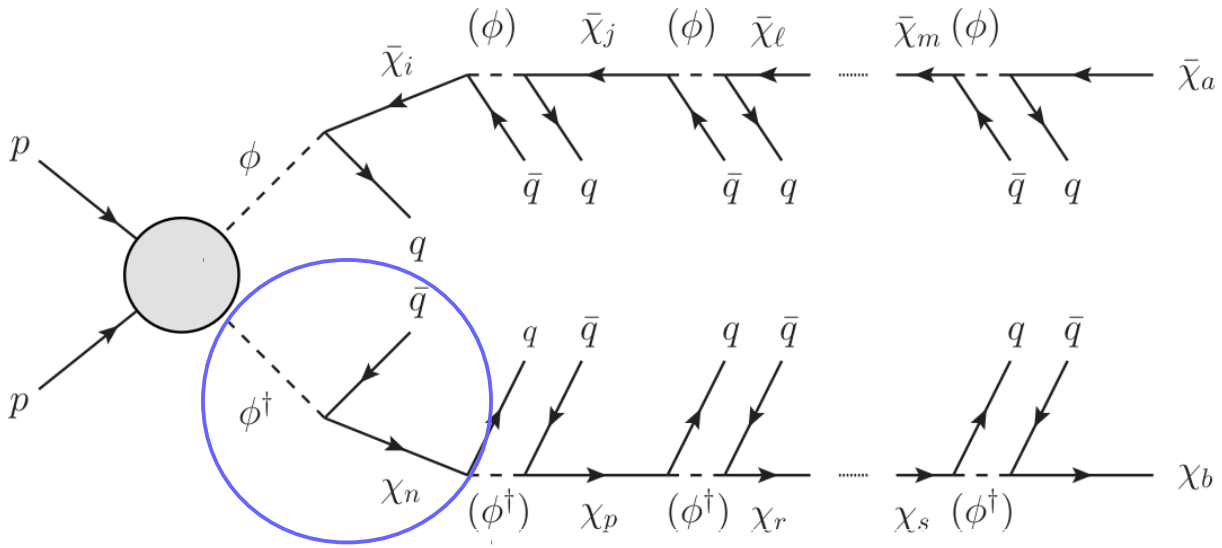
```
310  f1 = {n: np.array([
311              (6*(mX[n]**2 - mX[m]**2) / mPhi**2) - (5*(mX[n]**4 - mX[m]**4)/(mPhi**4)) + ((2*(mX[n]**2)*(mX[m]**2)*(mX[n]**2 - mX[m]**2))/(mPhi**6))
312              for m in range(n)]) for n in range(1, nX + 1)}
313  f2 = {n: np.array([
314              6 - ((8*(mX[n]**2 + mX[m]**2))/(mPhi**2)) + ((2*(mX[n]**4 + 4*(mX[n]**2)*(mX[m]**2) + mX[m]**4)/(mPhi**4)) - ((2*(mX[n]**4)*(mX[m]**4))/(mPhi**8)))
315              for m in range(n)]) for n in range(1, nX + 1)}
316  f3 = {n: np.array([ (2*(mX[n]**4)*(mX[m]**4))/(mPhi**8) for m in range(n)]) for n in range(1, nX + 1)}
317
318  # GENERAL BRANCHINGS
319
320  widthX = {n: np.array([(scipy.integrate.quad( lambda m23SqPar: ((3*cX[n]**2 * cX[m]**2) * ((mX[n]**2 - m23SqPar)**2 * (m23SqPar - mX[m]**2)**2) / (512 * PI**3 * mX[
321
```

PySwathMC Algorithm:



Step 1: Mother Particle Production

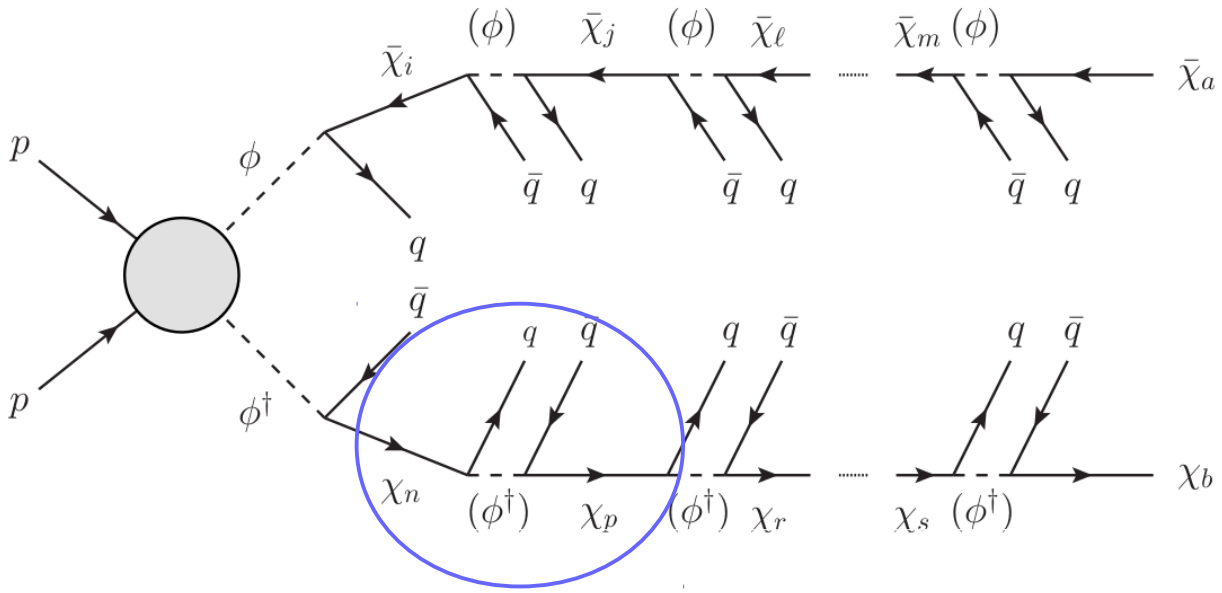
PySwathMC Algorithm:



Step 1: Mother Particle Production

Step 2: Decay to random mother χ_n plus quark (2-body decay)

PySwathMC Algorithm:

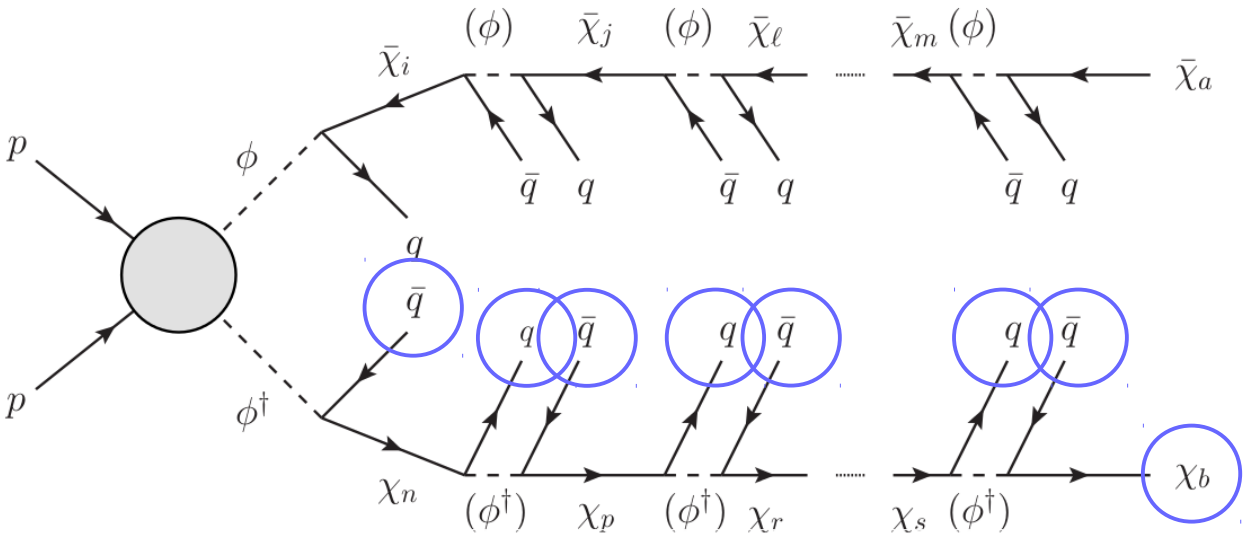


Step 1: Mother Particle Production

Step 2: Decay to random mother χ_n plus quark (2-body decay)

Step 3: Decay the mother χ_n to daughter χ_m plus 2 quarks (3-body decay) (repeat until ground state is reached)

PySwathMC Algorithm:



Repeat for desired num of events

Step 1: Mother Particle Production

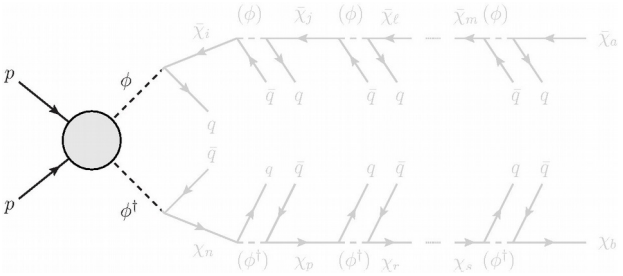
Step 2: Decay to random mother χ_n plus quark (2-body decay)

Step 3: Decay the mother χ_n to daughter χ_m plus 2 quarks (3-body decay) (repeat until ground state is reached)

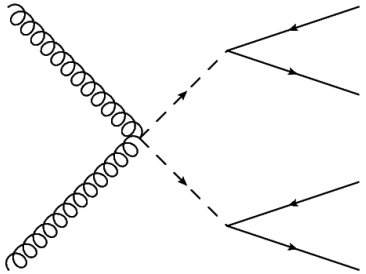
Step 4: Write event topology and momenta to file for later analysis

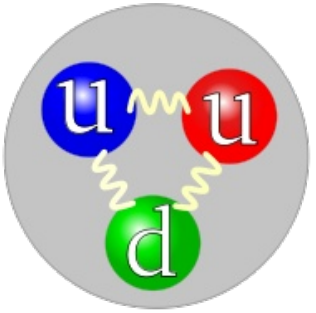
```
</init>
<event>
32  1 0.4290300E-06 0.1091187E+04 0.7818608E-02 0.9213825E-01
        21   -1    0    0  503  502    0.000000000E+00    0.000000000E+00    1.324086733E+03    1.324086733E+03    0.000000000E+00 0. 9.
        21   -1    0    0  501  503    0.000000000E+00    0.000000000E+00   -1.214629618E+03    1.214629618E+03    0.000000000E+00 0. 9.
   9000010    2    1    2  501    0   -1.426090798E+01   -4.364457722E+02   -5.920823744E+02    1.241470831E+03    1.000000000E+03 0. 9.
  -9000010    2    1    2    0  502    1.426090798E+01    4.364457722E+02    7.015394886E+02    1.297245520E+03    1.000000000E+03 0. 9.
        -1    1    4    0    0  502    2.257633555E+02   -2.506107078E+02    3.040192774E+02    4.540954189E+02    3.299999999E-01 0. 9.
         1    1    3    0  501    0   -9.731926753E+01   -4.240125012E+02   -2.923237909E+01    4.360186715E+02    3.299999999E-01 0. 9.
   9000029    2    4    0    0    0   -2.115024459E+02    6.870565293E+02    3.975202905E+02    8.431502478E+02    1.900000000E+02 0. 9.
```

Step 1: $\phi\phi^\dagger$ *production*

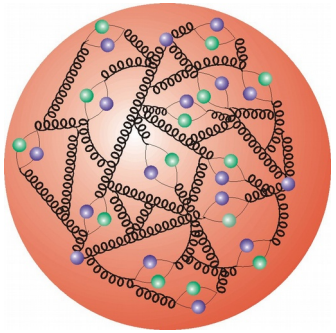


Recall: mothers created from gluons

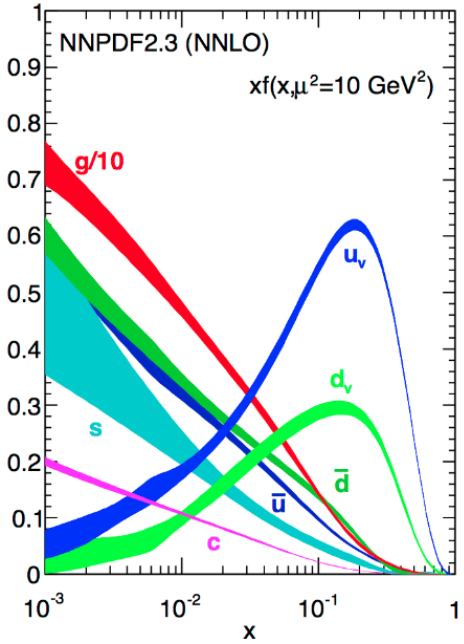




Low energy view of proton

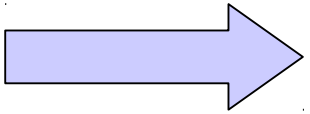


High energy view of proton



At high energy, protons look like a “soup” of particles. Virtual particle-antiparticle pairs pop in and out of existence, in accordance with the time-energy uncertainty principle.

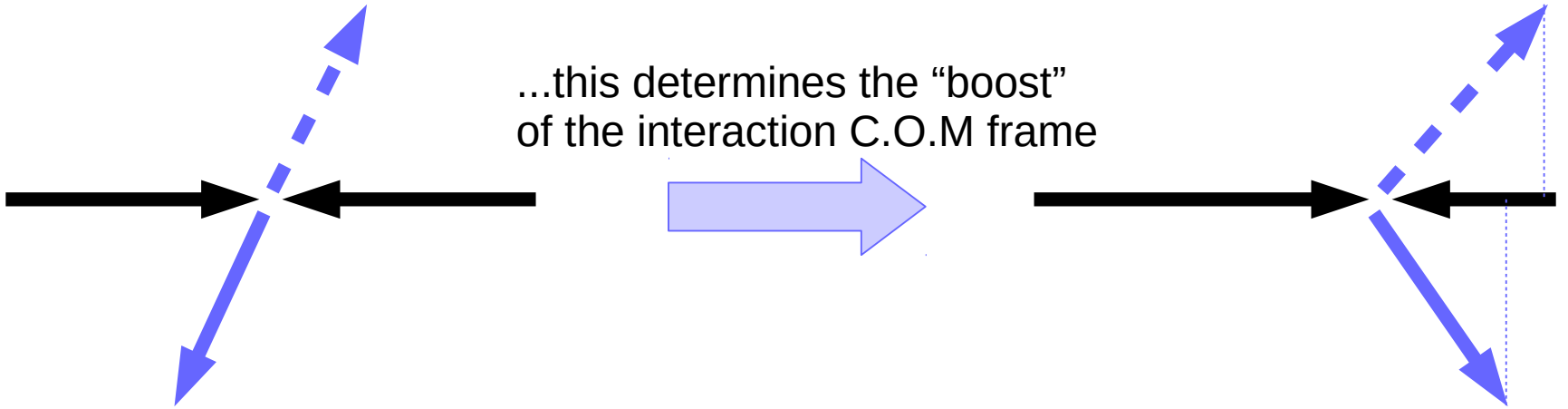
Number/energy of gluons in proton is inherantly uncertain. **Drawn from empirical parton density functions (PDF)**



Center of mass frame w.r.t the stationary lab frame for this collision is not known.

Monte Carlo the only way to simulate interactions at LHC.

Step 1: Draw energy and p_x of incoming particles randomly from PDF.



C.O.M Frame

$$\sum_i \vec{p}_i = 0$$

$$\vec{p}_{DM} = -\vec{p}_{\text{missing}}$$

Lab Frame

$$\sum_i \vec{p}_i^{\,T} = 0$$

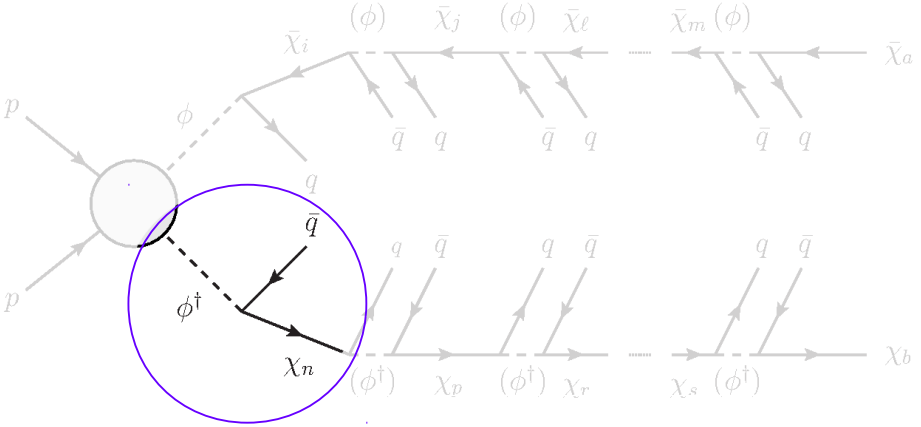
$$p_{DM}^T = -p_{\text{missing}}^T$$

Transverse momentum is still conserved, and can be used to deduce presence of invisible particles.

Note: This step needed for ALL processes at LHC – longitudinal boost of interactions is stochastic (drawn from PDF).

Step 2: Mother Particle Decay

Step 2a: Decay to random mother χ_n particle



For purposes of this talk, I've simplified the process and included this into the overall production cross section:

$$\sigma(gg \to \chi_n \chi_{n'})$$

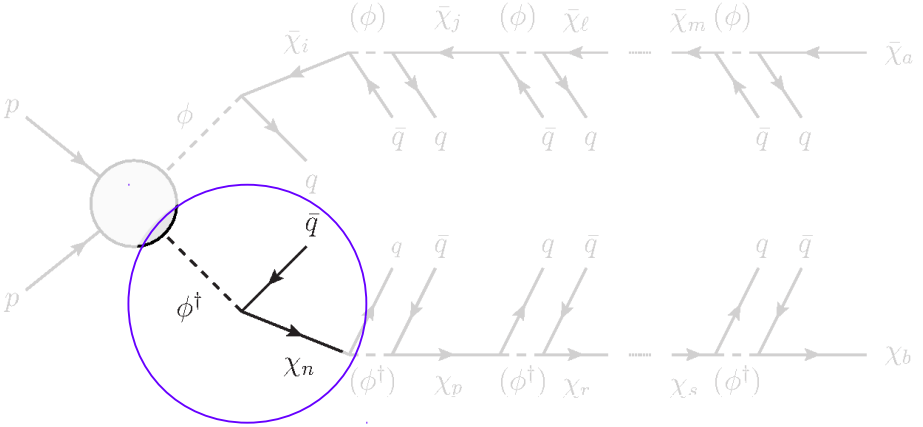
In reality, this is a combination of two separate processes:

$$\sigma(gg \to \phi\phi^\dagger) \qquad\qquad \Gamma(\phi \to \chi_n q)$$

NOT IMPORTANT... main point is that first χ_n is drawn randomly

```
#####################################
# PHIBar DECAY
#####################################
# Decay to random member Xi using random angles in CM frame
decayIndx = np.random.choice(indPhi, 1, p=brPhi)[0]
```

Step 2: Mother Particle Decay

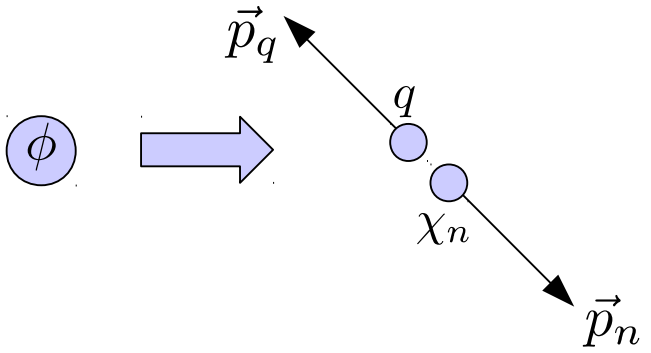


Step 2b: Solve for kinematics of χ_n and q

In the C.O.M. frame, this is a simple 2-body decay. *Relativistic kinematics easily solved....*

Cons. of Energy:

$$M = E_n + E_q$$
$$= \sqrt{(m_n c^2)^2 + (\vec{p}_n c)^2} + \sqrt{(m_q c^2)^2 + (\vec{p}_q c)^2}$$
$$\approx \sqrt{(m_n)^2 + (\vec{p}_n)^2} + p_q$$
$$\Rightarrow M^2 + 2Mp_q + p_q^2 = m_n^2 + p_n^2$$



Cons. of momentum: $\vec{p}_q = -\vec{p}_n$

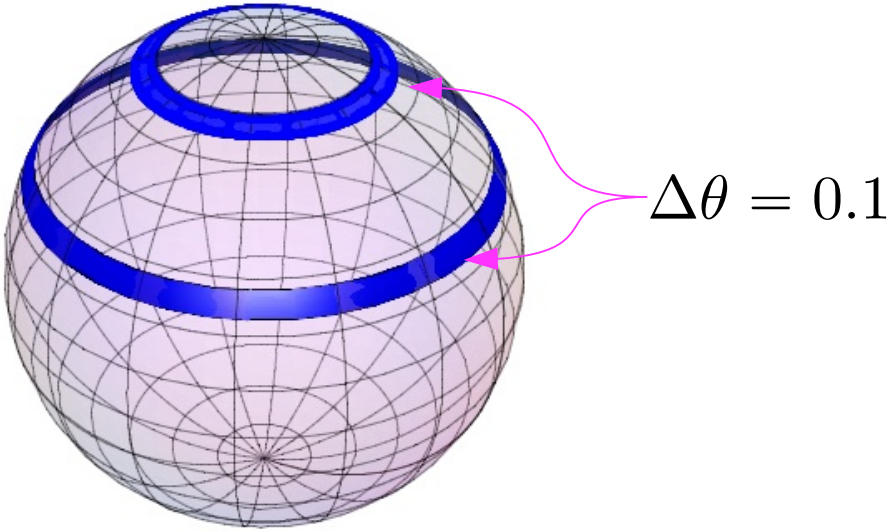
$$p = \frac{M^2 - m_n^2}{2M}$$

Energy & |momentum| determined

Direction is undetermined... the vectors above should be rotated to a random direction in 3D space

Step 2: Mother Particle Decay

Direction on sphere characterized by: $\theta,\ \phi$



Step 2c: determine random direction of decay

To pick a random direction on the sphere, we must randomly pick $\theta,\ \phi$, **properly weighted** by the phase space area.

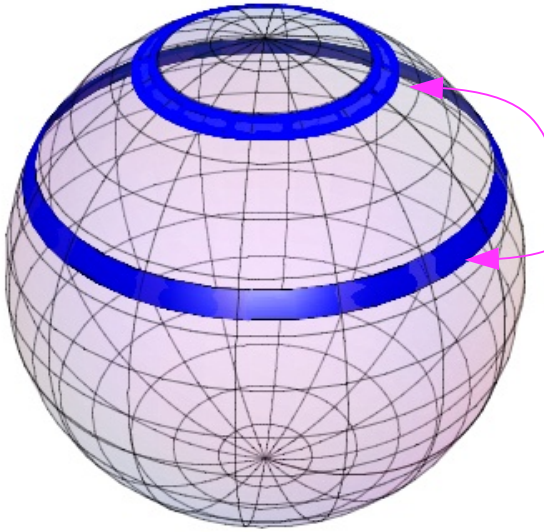
?

$$\phi \in \mathrm{U}[0, \pi]$$
$$\theta \in \mathrm{U}[0, 2\pi]$$

Probability of landing on the ring near the pole should be less than ring near equator.

Step 2: Mother Particle Decay

Direction on sphere characterized by: $\theta,\ \phi$



Step 2c: determine random direction of decay

To pick a random direction on the sphere, we must randomly pick $\theta,\ \phi$, **properly weighted** by the phase space area.

$$\phi \in \mathrm{U}[0, \pi]$$

$$\theta \in \mathrm{U}[0, 2\pi]$$

$$\cos\theta \in \mathrm{U}[-1, 1]$$

Probability of landing on the ring near the pole should be less than ring near equator.

Step 2d: Assign first daughter particles with the four-momenta

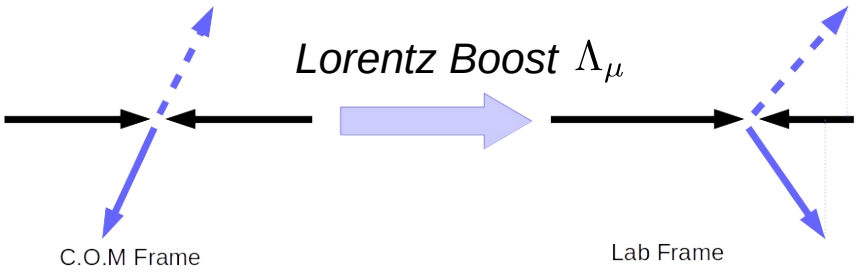
$$\boldsymbol{p} = (p, 0, 0) \quad E_i = \sqrt{m_i^2 + p^2} \quad p = (M^2 - m_n^2)/2M$$

$$p_\mu^n = (E_n, \boldsymbol{R}(\theta, \phi)\boldsymbol{p})$$

$$p_\mu^q = (E_q, -\boldsymbol{R}(\theta, \phi)\boldsymbol{p})$$

```
644     u = random.uniform(-1, 1)
645     thetaX = math.acos(u)
646     phiX = random.uniform(0, 2*PI)
647     #...and reassign the four vectors based on this Xi mass and these angles.
648     # These are momenta in the rest frame of the phi.
649     pXvec = [((mPhi**2 - mX[decayIndx]**2)/(2.0*mPhi)), thetaX, phiX]
650     pJ1Barvec = [-((mPhi**2 - mX[decayIndx]**2)/(2.0*mPhi)), thetaX, phiX]
651
```

Step 2: Mother Particle Decay



Step 2e: "boost" all vectors to the lab frame

To get the momenta we measure in the lab, we must act on the above vectors by a *Lorentz transformation.*

$$\beta = v/c$$
$$\gamma = 1/\sqrt{1-\beta^2}$$

Velocity of C.O.M. frame relative to lab frame was "determined" in Step 1. *(Momenta drawn randomly from PDFs)*

Transformation from COM frame to lab frame:

$$p_\mu^{\text{lab}} = \Lambda^\mu p_\mu^{\text{c.o.m.}}$$

$$[\Lambda^\mu] = B(\mathbf{v}) = \begin{bmatrix} \gamma & -\gamma\beta n_x & -\gamma\beta n_y & -\gamma\beta n_z \\ -\gamma\beta n_x & 1+(\gamma-1)n_x^2 & (\gamma-1)n_x n_y & (\gamma-1)n_x n_z \\ -\gamma\beta n_y & (\gamma-1)n_y n_x & 1+(\gamma-1)n_y^2 & (\gamma-1)n_y n_z \\ -\gamma\beta n_z & (\gamma-1)n_z n_x & (\gamma-1)n_z n_y & 1+(\gamma-1)n_z^2 \end{bmatrix}$$

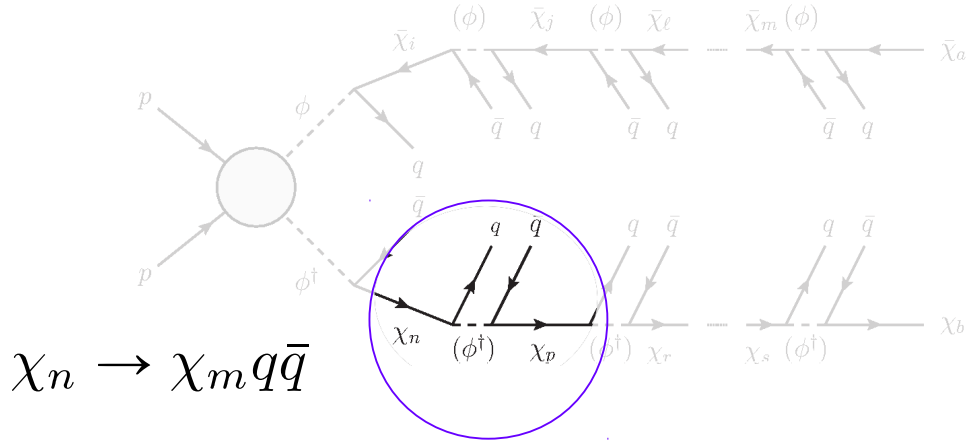
Note: before Einstein, this transformation was very simple:

$$v_{\text{lab}} = v_{\text{c.o.m}} + v_{\text{particle}}$$

```
799     ###############################################
800     # NOW BOOST TO LAB FRAME, SINCE PHI WAS BOOSTED
801     ###############################################
802     pXBarinitBoosted = boostedArray(pXBar[decayBarIndx], betaPhi)
803     pXBar[decayBarIndx].assn_p_cart([pXBarinitBoosted[1], pXBarinitBoosted[2], pXBarinitBoosted[3]])
804
805     pJ2initBoosted = boostedArray(pJ2[decayBarIndx], betaPhi)
806     pJ2[decayBarIndx].assn_p_cart([pJ2initBoosted[1], pJ2initBoosted[2], pJ2initBoosted[3]])
807
```

Moving on....

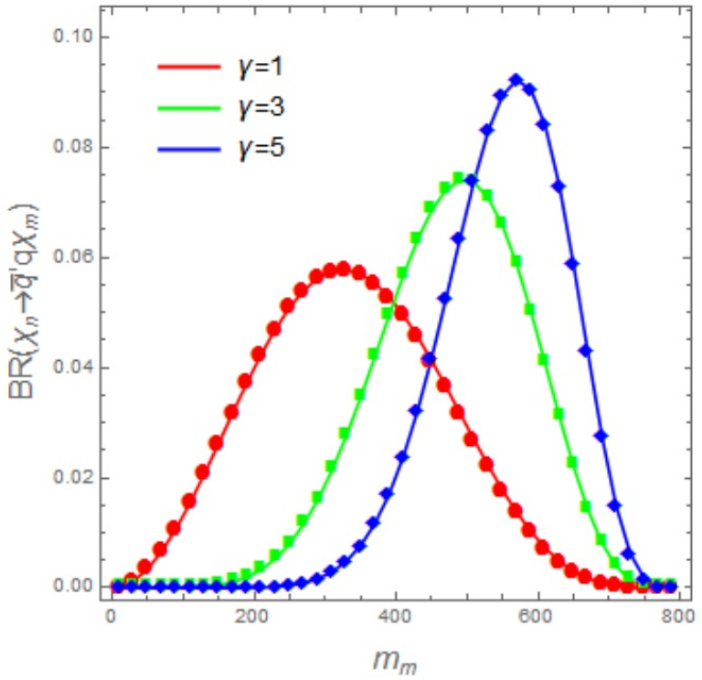
Step 3: Decay χ_n down the chain



Step 3a: determine daughter particle χ_n randomly

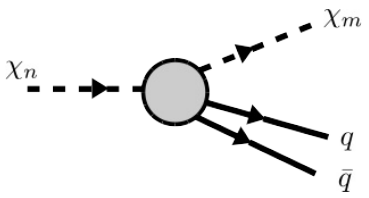
Draw from *branching ratio* probability distribution:

$$BR(n,m) = \frac{\Gamma(n,m)}{\sum_{m=0}^{n-1}\Gamma(n,m)}$$




```
684     ######################################
685     #         X-DECAY
686     # Now we want to decay the X[i] chosen above to X[j] plus two jets.
687     ######################################
688     motherIndx = decayIndx
689     colIdx = 503
690     xDecCount = 0
691
692     while motherIndx > 0:
693     #First, decay the X[i] in it's rest frame:
694         daughterIndx = np.random.choice(indX[motherIndx], 1, p=brX[motherIndx])[0]
695
```

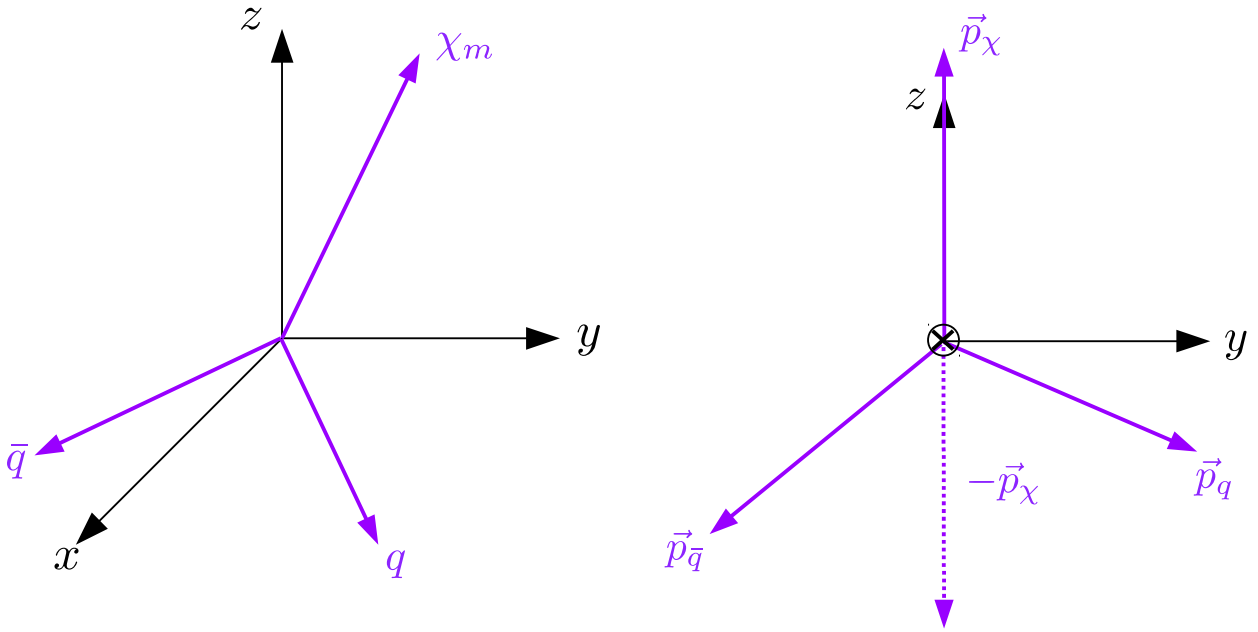
Step 3: Decay χ_n down the chain



Step 3b: determine daughter particle kinematics

In addition to overall orientation, 3-body decays have **2 undetermined degrees of freedom**.

We can see this pictorially as follows:

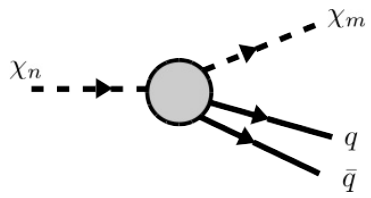


Geometrically, three particles lie in a 2D-plane.

→ Rotate frame so that X is along the z axis.

→ Now rotate about z-axis so that q is in yz-plane.

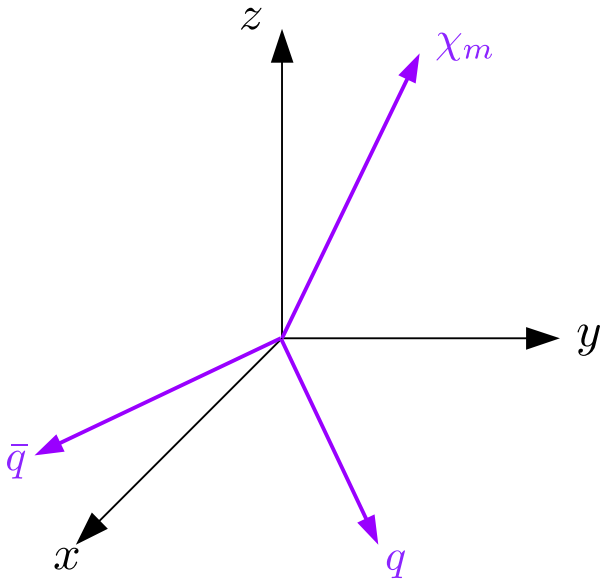
Step 3: Decay χ_n down the chain

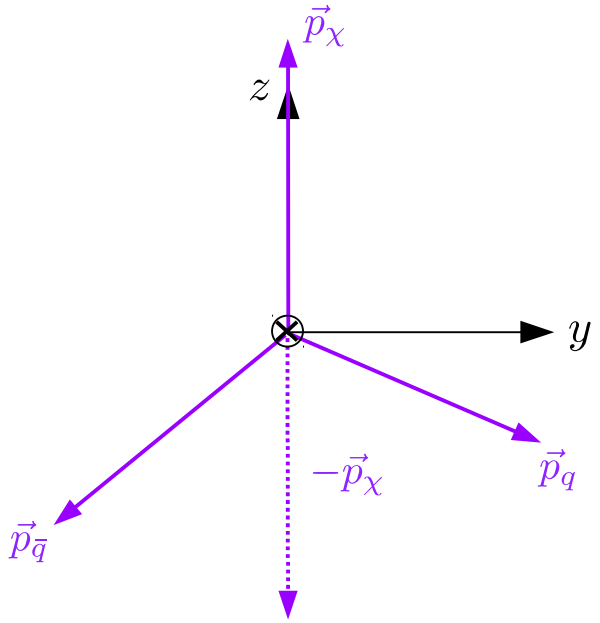


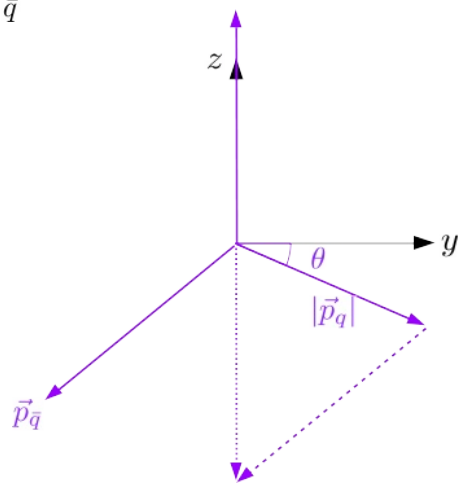
Step 3b: determine daughter particle kinematics

In addition to overall orientation, 3-body decays have **2 undetermined degrees of freedom**.

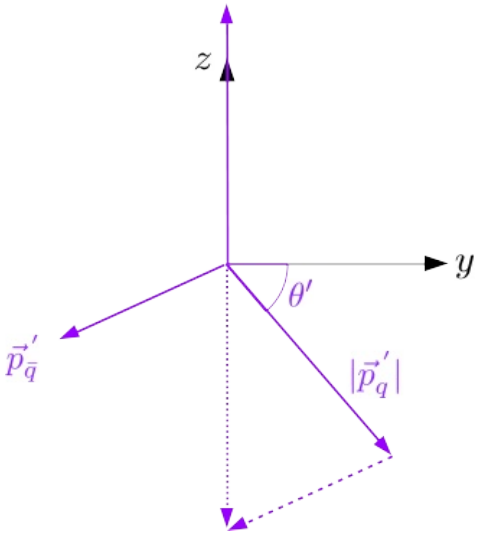
We can see this pictorially as follows:







Now, for a given $|\vec{p}_\chi|$, we are free to choose θ and $|\vec{p}_q|$



… $\vec{p}_{\bar{q}}$ is then determined.

Geometrically, three particles lie in a 2D-plane.

→ Rotate frame so that X is along the z axis.

→ Now rotate about z-axis so that q is in yz-plane.

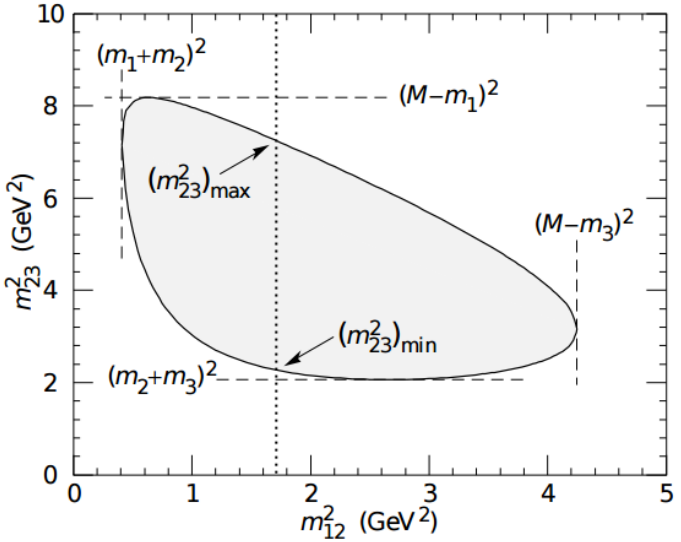
Instead of θ and $|\vec{p}_q|$, it is convenient to define m_{12} and m_{23} with $m_{ij} = (p_i + p_j)^2$

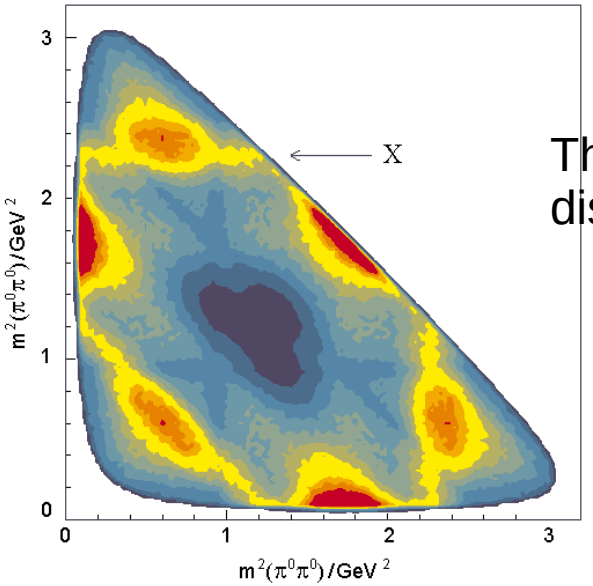
Step 3: Decay χ_n down the chain

Step 3b: determine daughter particle kinematics

The two free variables must be drawn randomly. However, they are typically NOT uniformly distributed. They are *correlated*.

Correlations (and upper/lower limits) are graphically represented by “Dalitz plots”:





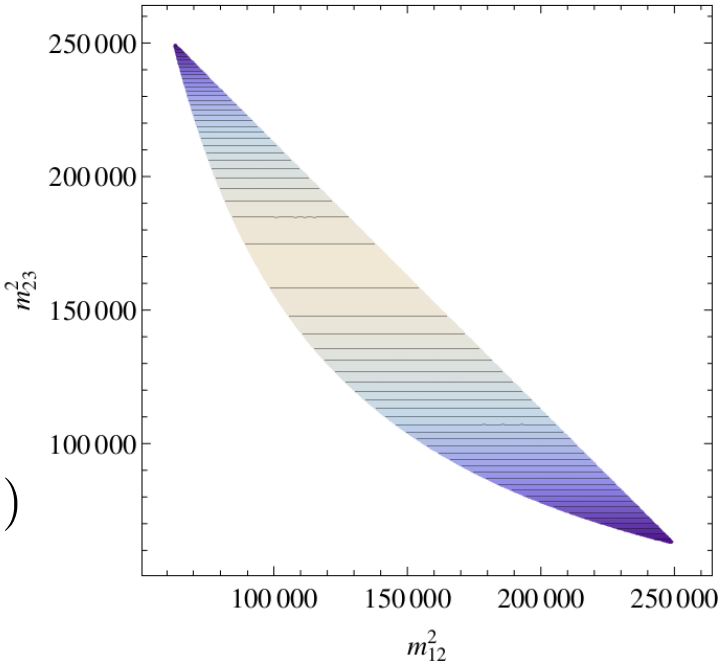
These are 2D probability distributions

Boundaries of these plots are determined by kinematic limits, e.g., total energy available in the decay: $\Delta E = m_{\text{final}} - m_{\text{initial}}$

“Height” characterizes probability. When built from experimental data, “denser” regions can indicate presence of intermediate particles.

Step 3: Decay χ_n down the chain

Step 3b: determine daughter particle kinematics



Dalitz plot calculated for our model:

$$\frac{(m_n^2 - m_{23}^2)(m_{23}^2 - m_m^2)}{(m_{23}^2 - m_\phi^2)^2} \rightarrow \int dm_{12} dm_{23} \rightarrow P(\chi_n \rightarrow \chi_m)$$

(Independent of m_{12})

- Pick m_{23} from above distribution
- Pick m_{12} from uniform distribution between min and max

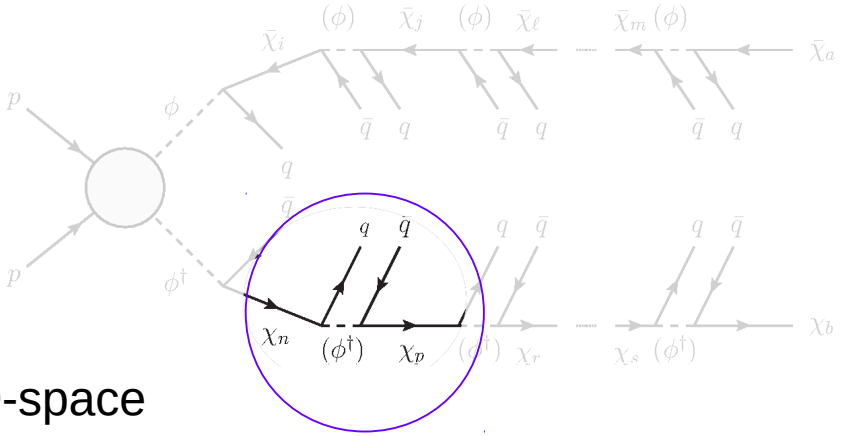
  → This sets energy and momenta of all three particles.

```
698         m23SqSampled = m23Sample(daughterIndx, motherIndx)
699         # Note, in what follows, particle 1 is the J1, particle 2 is the X, and particle 3 is the J1Bar
700         E1 = (mX[motherIndx]**2 - m23SqSampled) / (2*mX[motherIndx])  # Energy of the J1 (as opposed to J1Bar)
701         # Now select m12Sq (E3)... this is a uniformly distributed value between m12Sqmin and m12Sqmax
702         m12Sqmin = mX[motherIndx]**2 * mX[daughterIndx]**2 / m23SqSampled
703         m12Sqmax = mX[motherIndx]**2 + mX[daughterIndx]**2 - m23SqSampled
704         m12SqSampled = random.uniform(m12Sqmin, m12Sqmax)
705
706         E3 = (mX[motherIndx]**2 - m12SqSampled) / (2*mX[motherIndx]) # Energy of the J1Bar
```

$$(m_{12}^2)_{\min} = \frac{m_n^2 m_m^2}{m_{23}^2}$$

$$(m_{12}^2)_{\max} = m_n^2 + m_m^2 - m_{23}^2.$$

Step 3: Decay χ_n down the chain



Step 3c: Rotate entire system to random direction in 3D-space

Step 3d: Boost entire system by the boost factor of the mother particle in order to express the vectors in the lab-frame.

Step 3e:

- if m>0 (i.e., daughter χ_m is not the ground state χ_0), set n=m, and return to Step 3a.
- **If m=0, output all generated vectors and event information to output file.**

Step 4: Repeat 10^5 times!

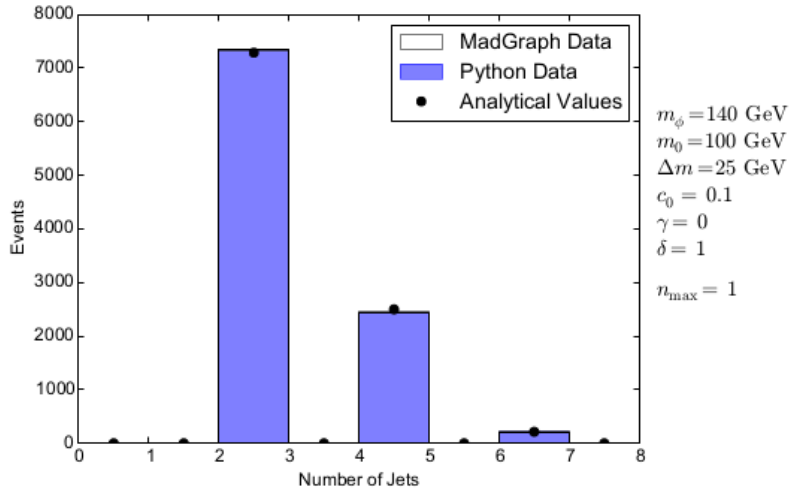
Benchmarks: $m_0 = 10\ \mathrm{GeV};\ \Delta m = 10\ \mathrm{GeV}; \delta = 1; m_\phi = 1000\ \mathrm{GeV}$

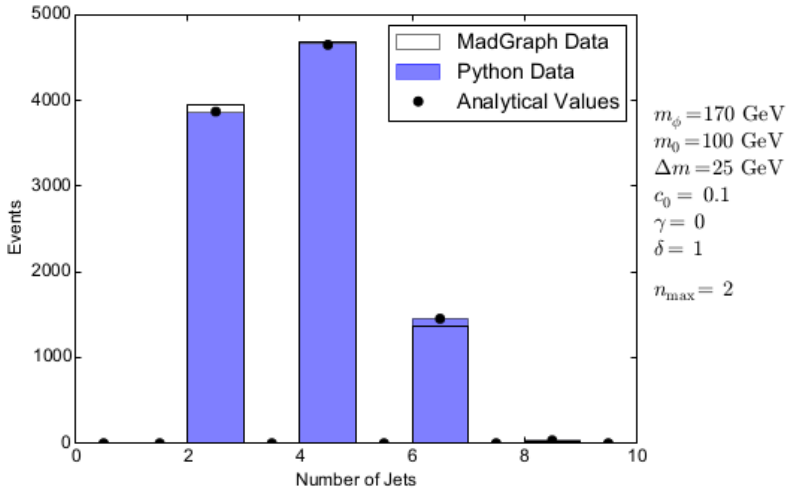
98 states in the tower above the ground state

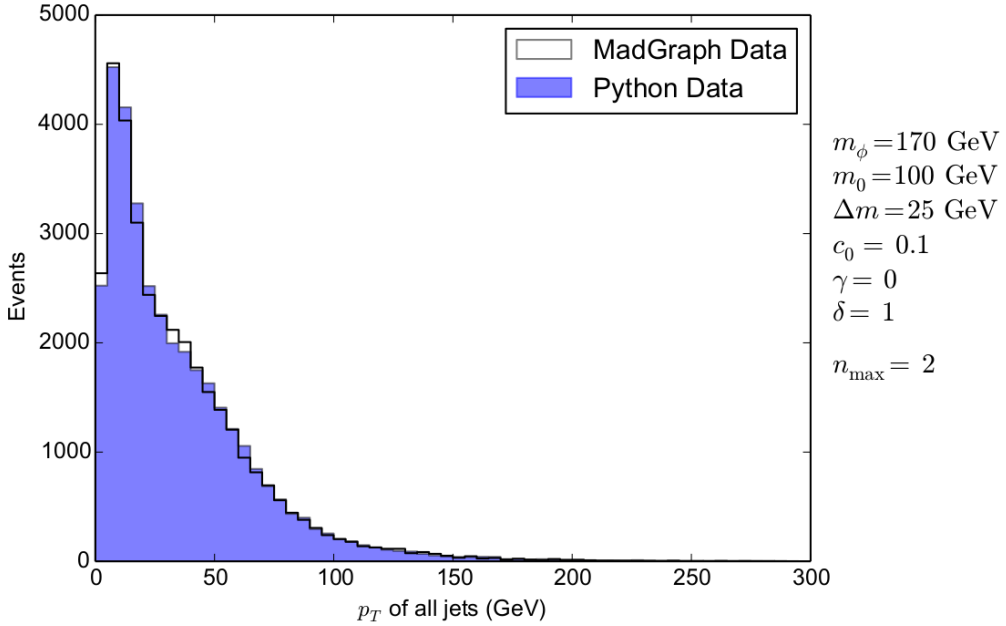
256.8s (~4 mins) on Intel Core i5-2500K, 3.30Ghz x 4

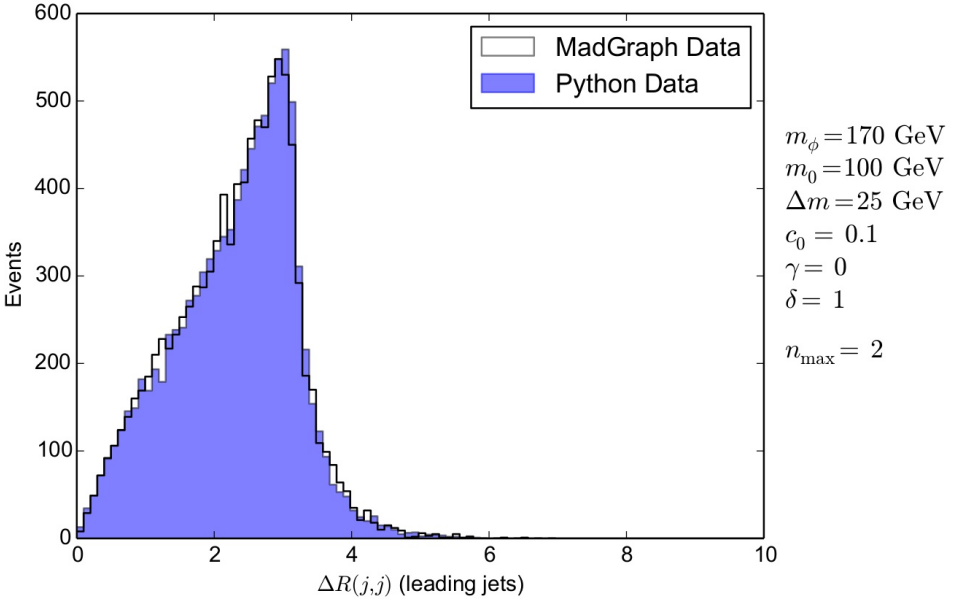
Results

Comparisons to industry standard MadGraph5 for small towers:



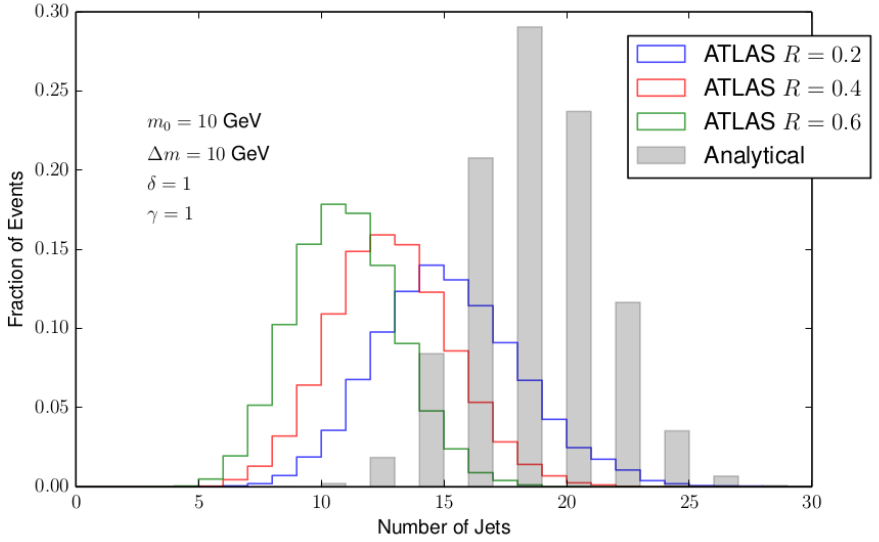


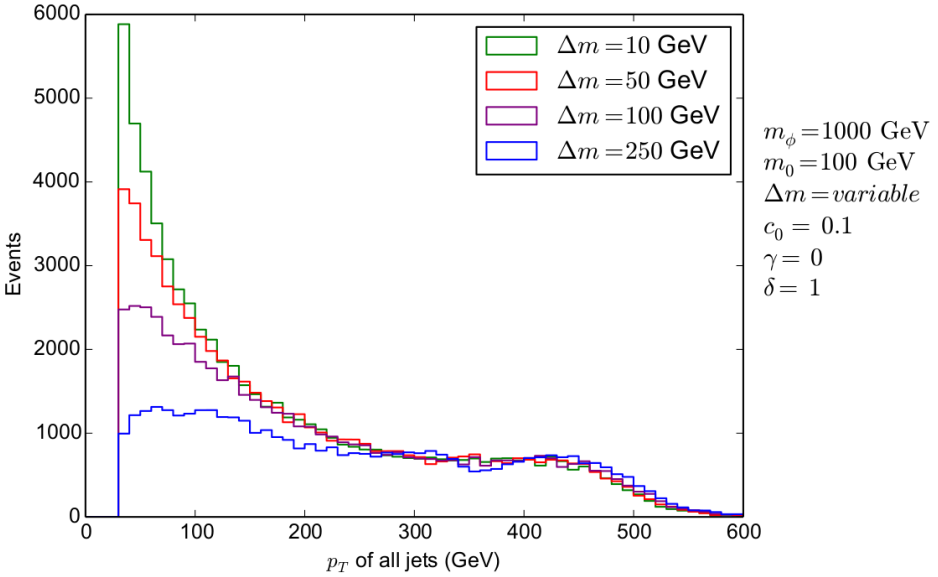


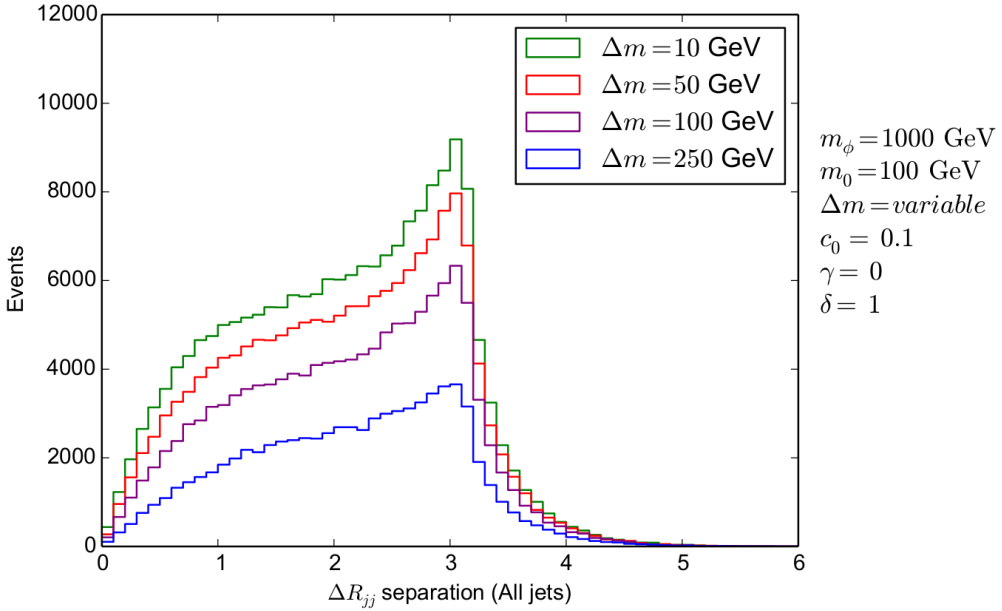


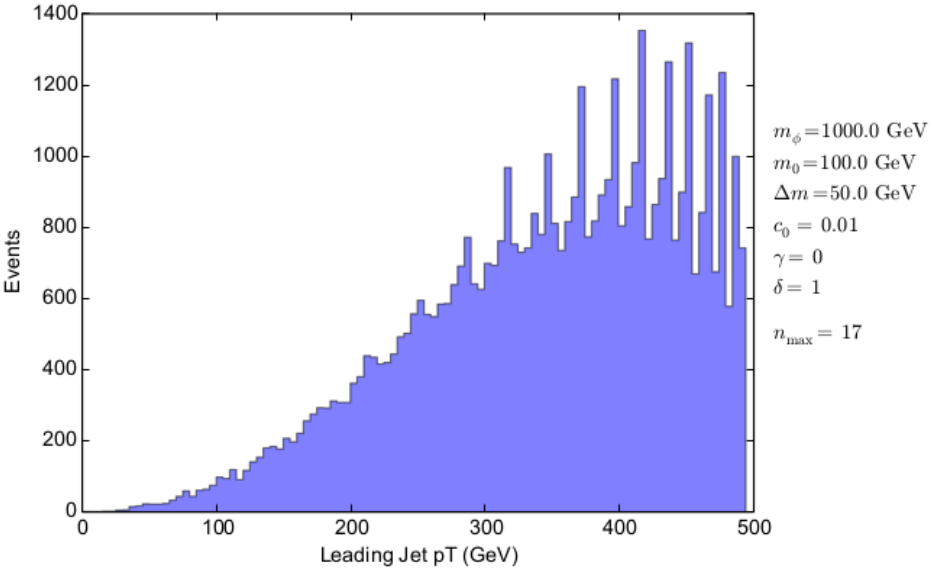
PySwath results for larger towers

Jet number, missing energy, momentum distributions,...









Deeper down the rabbit hole...

Our Monte Carlo simulation produces “parton-level” events.... that is, final state made up of quarks

PySwath must then be put through a 2nd layer of MC, which simulates “Hadronization”

- Quarks have *color charge*, and thus can never be observed (all particles in nature are necessarily colorless)
- As quarks separate, the energy in the fields becomes so large that quark-antiquark pairs are “pulled out of the vacuum... the end result is a bunch of colorless “hadrons”, which are combinations of quarks which give zero color charge
- This is simulated in PYTHIA, an industry standard

The hadronized events are then put through a 3rd layer of MC, which serves to simulate detector effects. This is done with DELPHES, another industry standard.

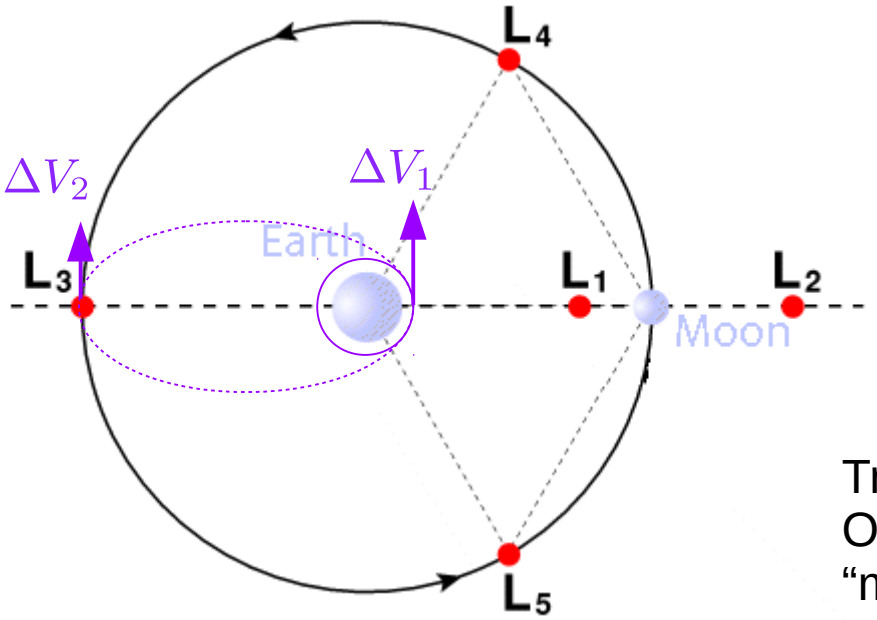
-An application in astrodynamics --- Modeling performance of orbital control system in stable manifold trajectories to Lagrange points.

Based on work done by E. Butcher, M. Nazari, et. al. and ***not DY***

E.Butcher, M.Nazari, et. al., *Optimal Transfers with Guidance to the Earth-Moon L1 and L3 Libration Points using Invariant Manifolds*, 2012, doi:10.2514/6.2012-4667

W. Anthony, E. Butcher, and J. Parker, *Statistical Fuel Budgets for Impulsive Guidance to Earth-Moon L1 Halo Orbits,* 2014, doi:10.2514/6.2014-4138

Low Energy/Ballistic Transfers to L1 and L3 Lagrange Points

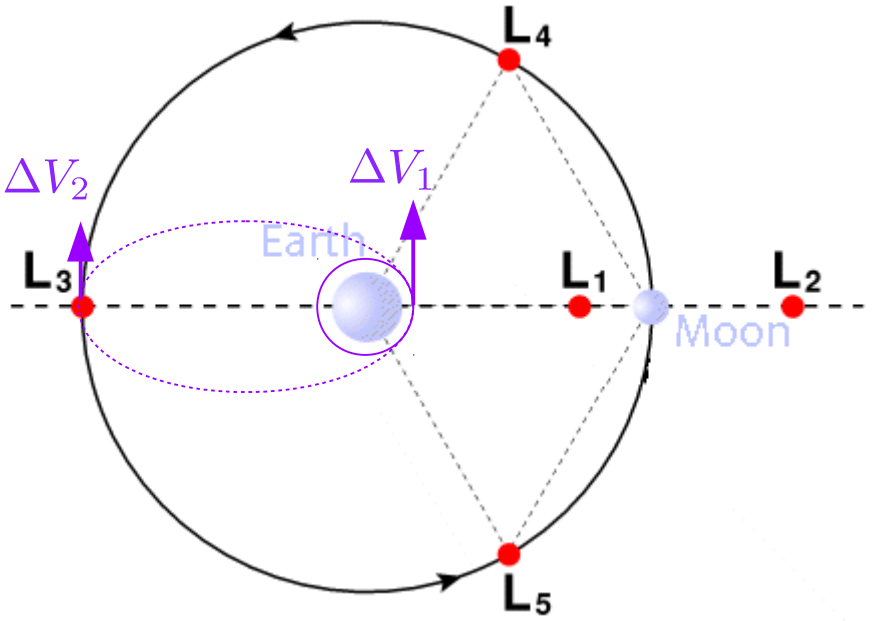


The task is to transfer from LEO to L1 or L3

High Energy Transfer, e.g, Hohmann.
One large burn to increase apoapsis to L1 or L3
Second large burn to match velocity of L1/L3 at arrival.

Transfer via **stable "invariant manifolds"**.
One large burn to leave LEO and intersect with the "manifold"
One small burn to match velocity (enter manifold).
Remaining trajectory to L1/L3 is ballistic.

Low Energy/Ballistic Transfers to L1 and L3 Lagrange Points



The task is to transfer from LEO to L1 or L3

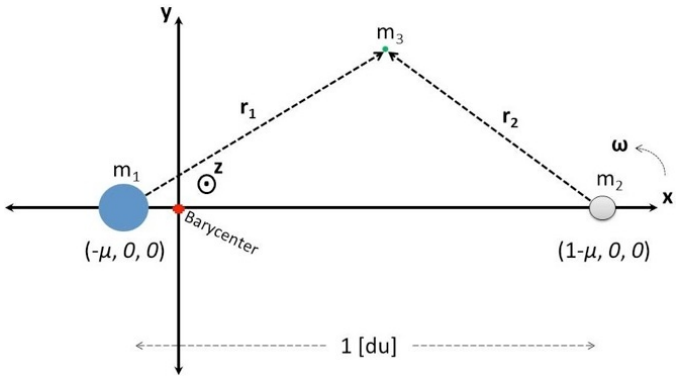
High Energy Transfer, e.g, Hohmann.
One large burn to increase apoapsis to L1 or L3
Second large burn to match velocity of L1/L3 at arrival.

Transfer via **stable "invariant manifolds"**.
One large burn to leave LEO and intersect with the "manifold"
One small burn to match velocity (enter manifold).
Remaining trajectory to L1/L3 is ballistic.

Circular restricted 3-body problem:

$$\ddot{x} - 2\dot{y} = \frac{\partial U}{\partial x}$$
$$\ddot{y} + 2\dot{x} = \frac{\partial U}{\partial y}$$
$$\ddot{z} = \frac{\partial U}{\partial z}$$

$$m_1, m_2 \gg m_3$$



$$U(x,y,z) = \frac{x^2+y^2}{2} + \frac{1-\mu}{r_1} + \frac{\mu}{r_2}$$

$$C = x^2 + y^2 + \frac{2(1-\mu)}{r_1} + \frac{2\mu}{r_2} - v^2$$

Constant of motion: determine trajectories of constant energy (i.e., ballistic trajectories)

Lagrange points found by setting all velocities and accelerations to zero. For L1, L2, L3, also set y=z=0:

$$\ddot{x} - 2\dot{y} = \frac{\partial U}{\partial x} \Rightarrow \qquad x_e - \frac{(1-\mu)(x_e+\mu)}{|x_e+\mu|^3} - \frac{\mu(x_e-1+\mu)}{|x_e+\mu-1|^3} = 0$$

$$\Rightarrow x_e = \{0.8369, 1.156, -1.005\}$$

Stability found by expanding about these libation points:

$$\vec{r} = \vec{r}_e + \delta\vec{r}$$

$$\begin{pmatrix} \delta\ddot{x} \\ \delta\ddot{y} \\ \delta\ddot{z} \end{pmatrix} + \begin{bmatrix} 0 & -2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} \delta\dot{x} \\ \delta\dot{y} \\ \delta\dot{z} \end{pmatrix} - \begin{bmatrix} U_{xx} & U_{xy} & U_{xz} \\ U_{xy} & U_{yy} & U_{yz} \\ U_{xz} & U_{yz} & U_{zz} \end{bmatrix} \begin{pmatrix} \delta x \\ \delta y \\ \delta z \end{pmatrix} = 0$$

Gyroscopic terms

Hessian

Assuming exponential solution, we find characteristic equation → eigenvalues → L1, L2, and L3 turn out to be saddle points

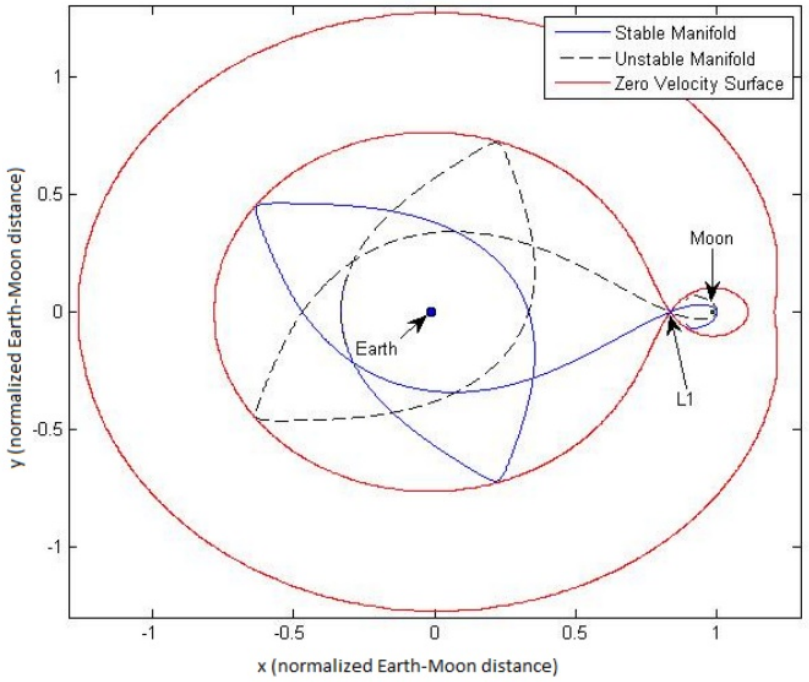
In state-space form,

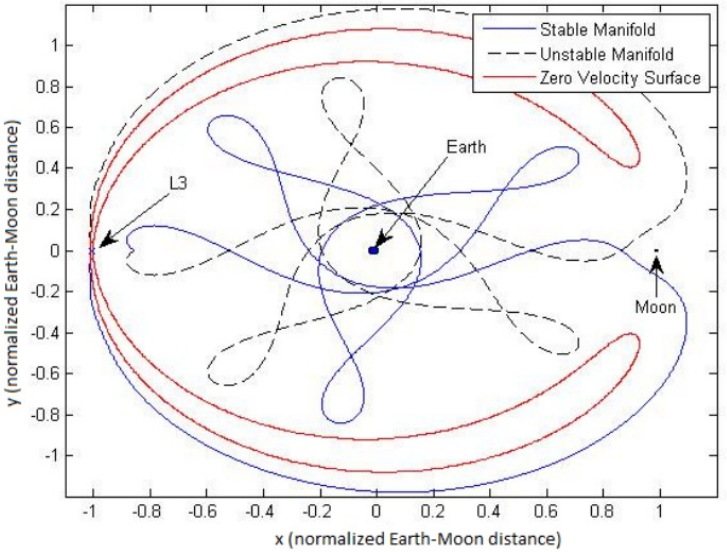
$$\delta\dot{\bar{x}} = A\delta\bar{x}$$

$$\delta\bar{x} = \begin{pmatrix} \delta\bar{r} \\ \delta\dot{\bar{r}} \end{pmatrix}, A = \begin{bmatrix} 0_3 & I_3 \\ H & -G \end{bmatrix}$$

From these we can determine stable, center, and unstable subspaces of solutions.

Es = regions around xe,ye,ze,xedot... where motion converges
Eu = “” “” where motion diverges
Ec = “” “” where motion orbits (halo orbits)

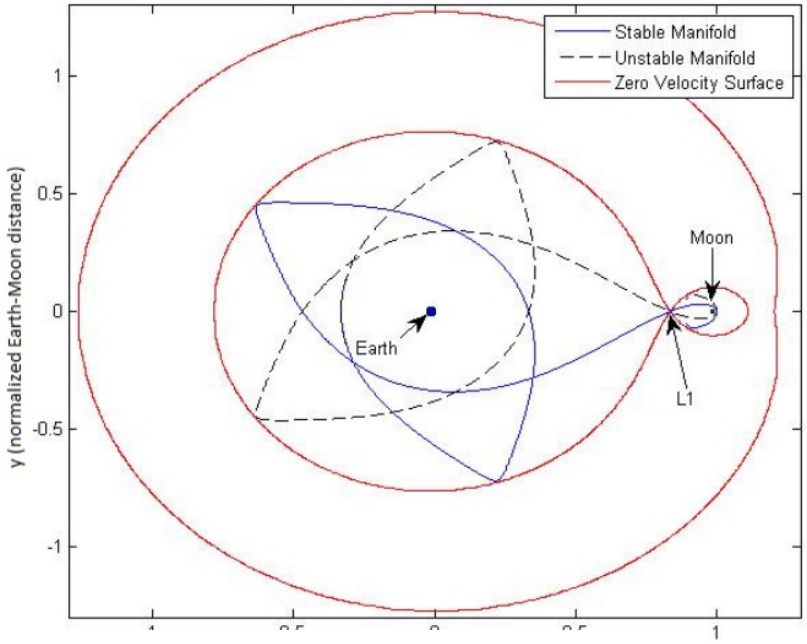


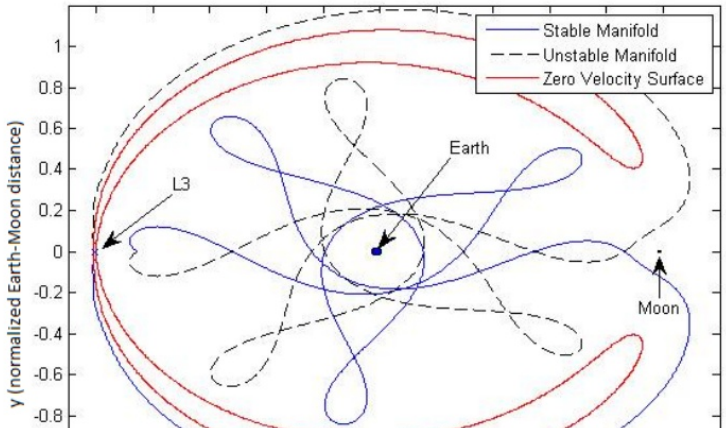


Invariant manifolds:

Regions of x,v phase space which follow ballistic trajectories to the Lagrange points.

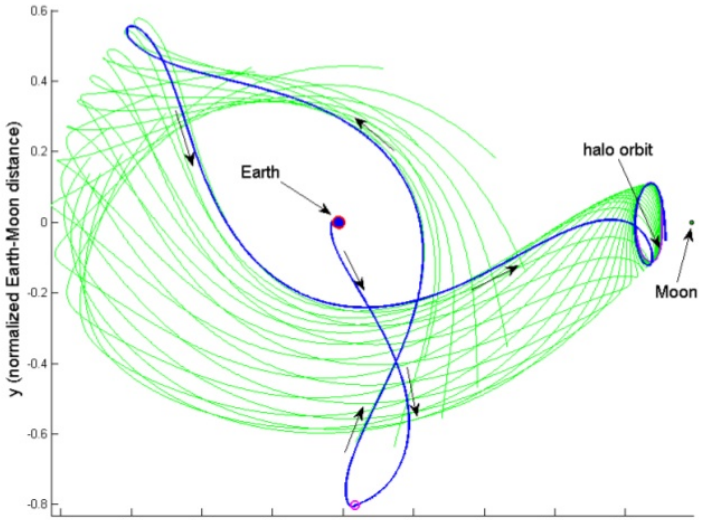
To find the L1/L3 stable manifolds numerically, we take an initial point near (but not at) r_e within the Es subspace, then integrate equations of motion backward in time.

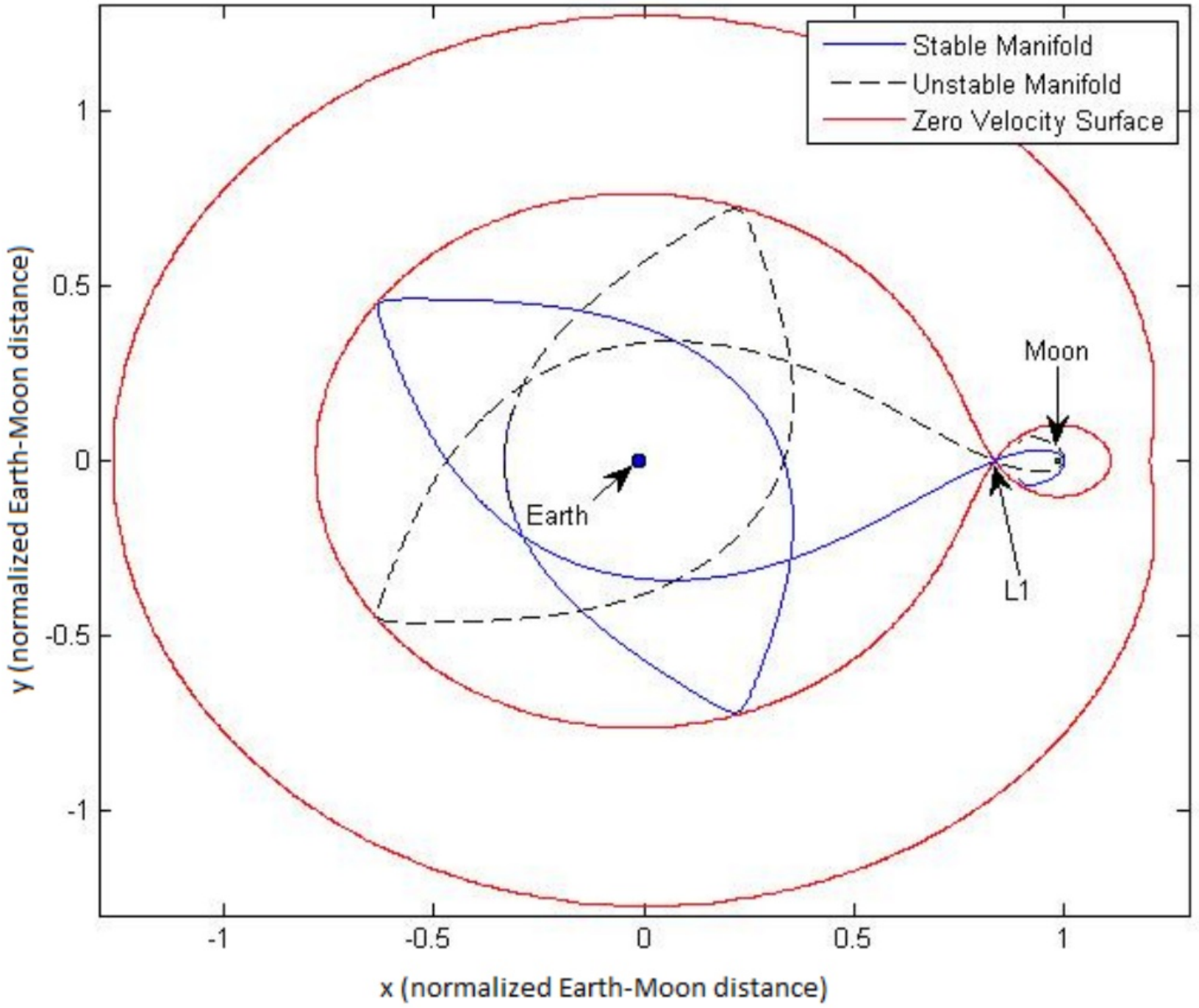




For point-like (0D) endpoint, these manifolds are 1-dimensional: lines through xyz space.

For halo (1D) endpoint, the manifolds are 2D surfaces in xyz space



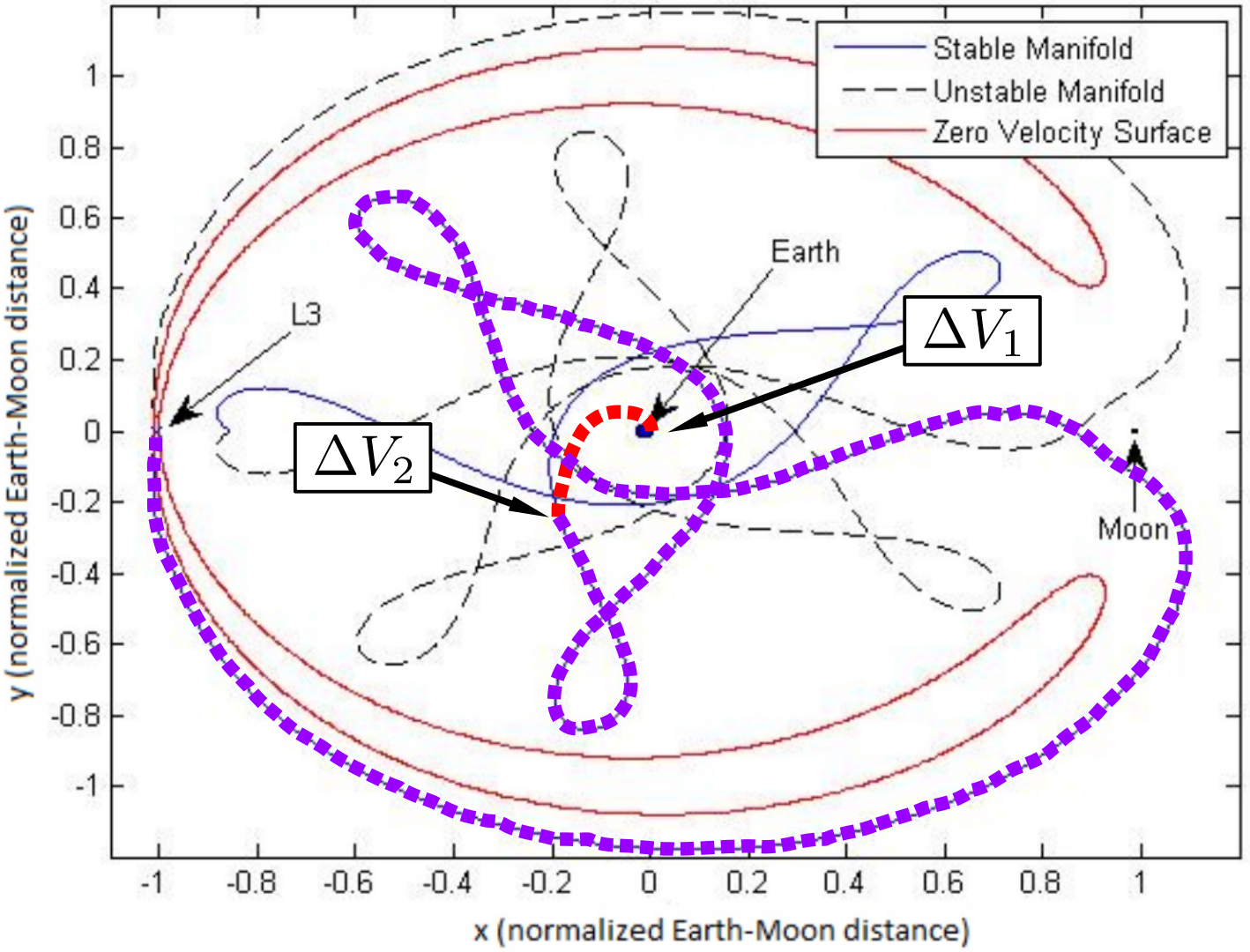

Stable Manifold
Unstable Manifold
Zero Velocity Surface
Moon
Earth
L1
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
y (normalized Earth-Moon distance)
x (normalized Earth-Moon distance)

Free variables:

$\Delta V_1 \qquad \theta$

$$\bar{r} = (r_{LEO}cos\theta - \mu)\hat{i} +$$

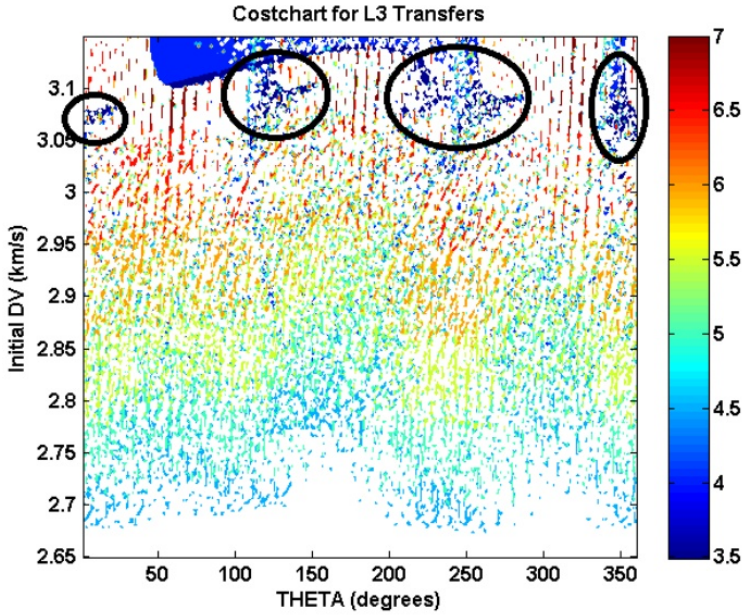
With these variables chosen, delta V2 is set.

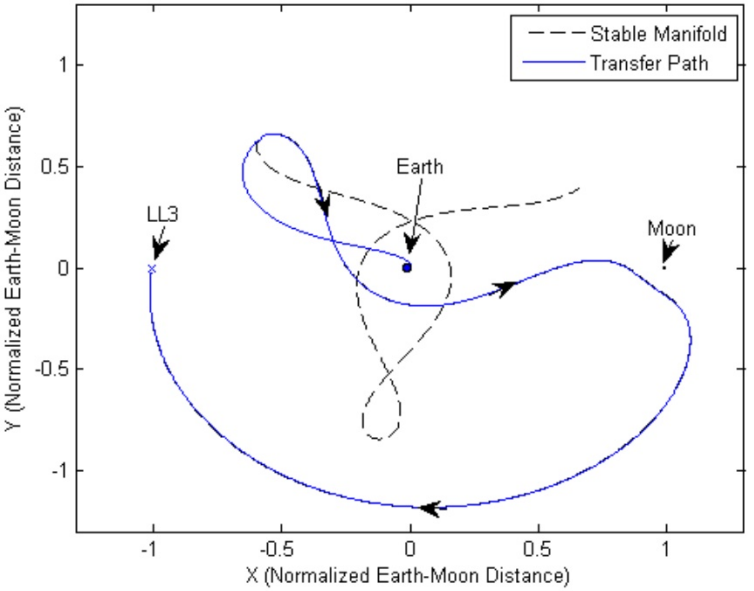


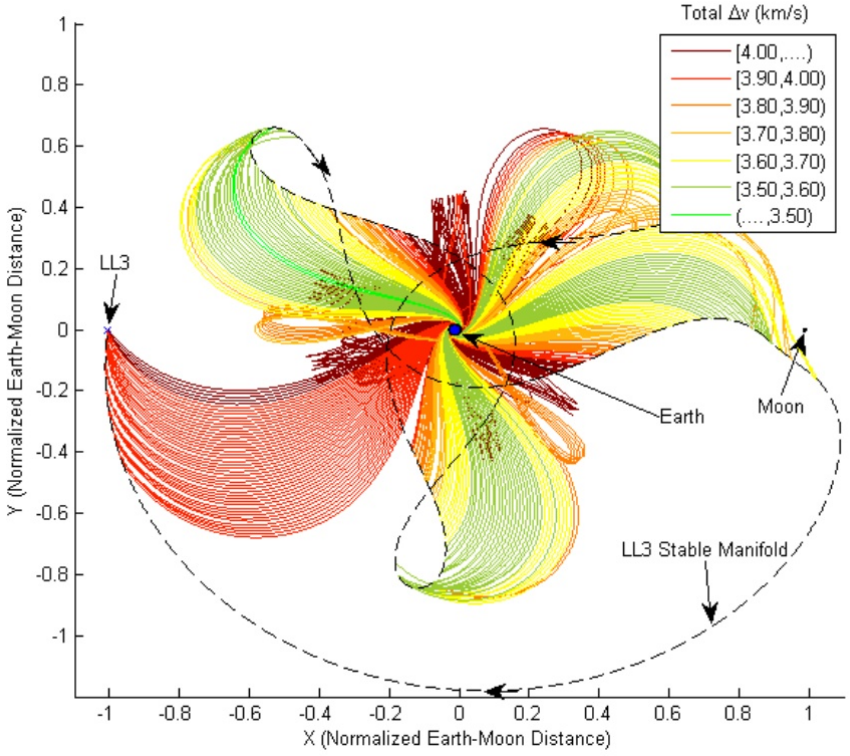
ΔV_1 First impulse: LEO to transfer trajectory

ΔV_2 Second impulse: Transfer trajectory to L3 manifold trajectory

(purple and red sketched by hand for illustration purposes... not based on actual simulation)


Costchart for L3 Transfers
Initial DV (km/s)
THETA (degrees)
3.1
3.05
3
2.95
2.9
2.85
2.8
2.75
2.7
2.65
50
100
150
200
250
300
350
7
6.5
6
5.5
5
4.5
4
3.5


Stable Manifold
Transfer Path
Earth
LL3
Moon
Y (Normalized Earth-Moon Distance)
X (Normalized Earth-Moon Distance)
1
0.5
0
-0.5
-1


Total Δv (km/s)
[4.00,....)
[3.90,4.00)
[3.80,3.90)
[3.70,3.80)
[3.60,3.70)
[3.50,3.60)
(....,3.50)
LL3
Earth
Moon
LL3 Stable Manifold
Y (Normalized Earth-Moon Distance)
X (Normalized Earth-Moon Distance)

	D1	D2		

Modeling and compensating for error and uncertainty in thrust magnitude and direction

Unavoidable errors in thrust magnitude and direction during Delta V1 and Delta V2 will be present. Imperfect knowledge of exact location, thruster imperfections, shutoff time, etc...

Since E.O.M are non-linear, small errors at Delta V1,2 can lead to large trajectory errors

→ missing target lagrange points by substantial amounts

→ overshoot the stability subset Es, so S/C does not settle around L1/L3

Need control system to correct for errors durring Delta V1,2.

Again, since E.O.M are non-linear, not possible to analytically relate errors in Delta V1,2 to final error at destination...

Impliment Monte Carlo methods to generate errors during impulse burns, then analyze performance of controller.

A few points...

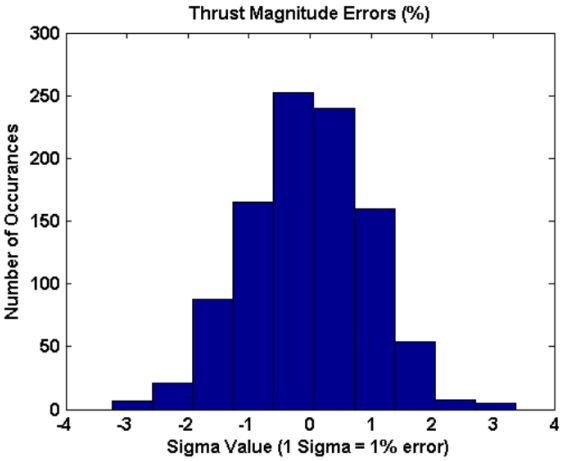
1) Different controllers have been considered, e.g.,
LQR (continuous thrust) for stationary L-point destination
Impulse corrections for halo orbits around L-points

2) Fuel used in control effort typically negligible compared to transfer impulse burns.

3) Monte Carlo analysis allows us to identify errors which have more impact on trajectory → can tell us how to design our S/C (i.e., directional thrust more imporant than burn time, etc)

Monte Carlo generation of errors in thrust magnitude and direction

Need to analyze the effects of errors in DeltaV1,2 impulse burns.

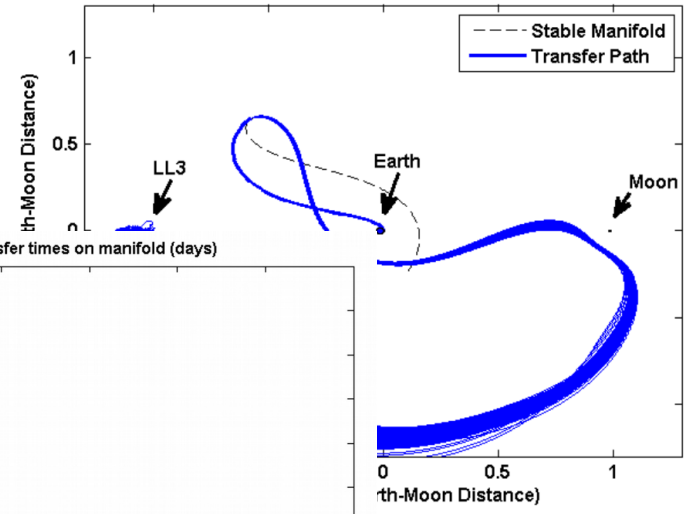


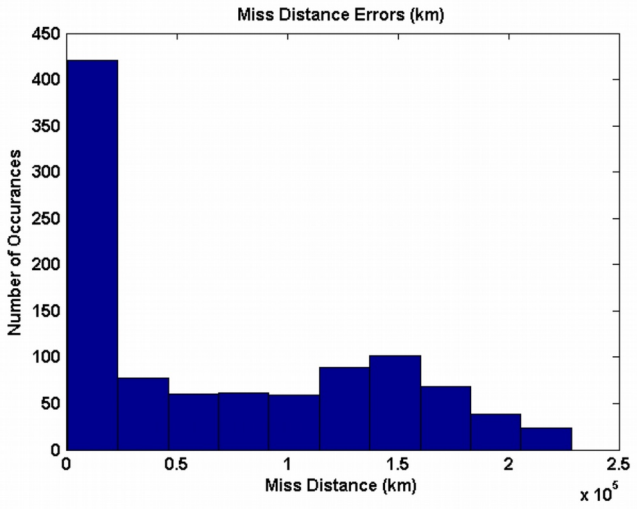
Consider the DeltaV2 burn first

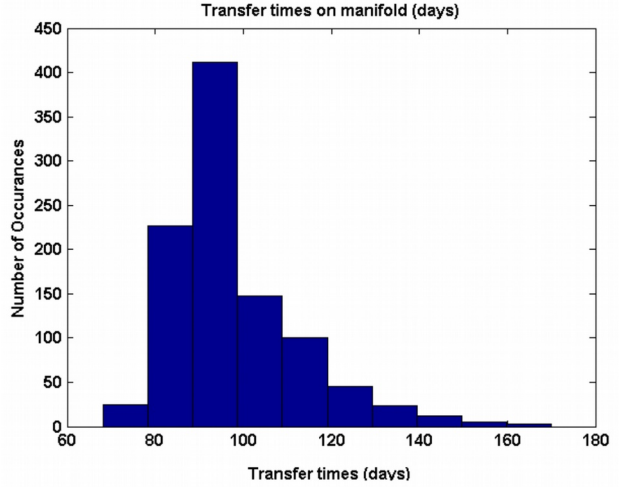
Burn can be described by three variables:
Magnitude: $|\Delta V_2^n + \delta\Delta V_2|$
Direction (two angles... as seen in particle decay)
Direction out of the plane: $\theta^{(n)} + \delta\theta$
Direction in the plane: $\phi^{(n)} + \delta\phi$

Step 1: Assume propogate trajectory to manifold injection point.

Step 2: Draw errors on nominal values from some probability d n, wit



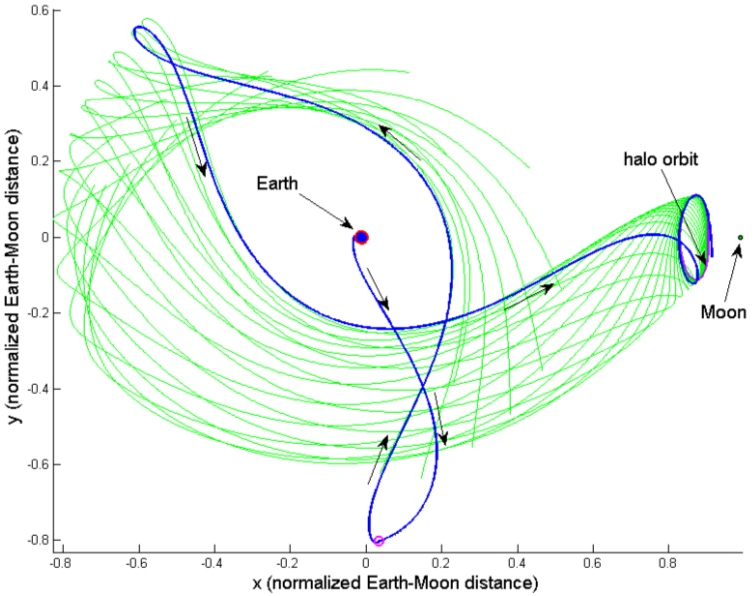


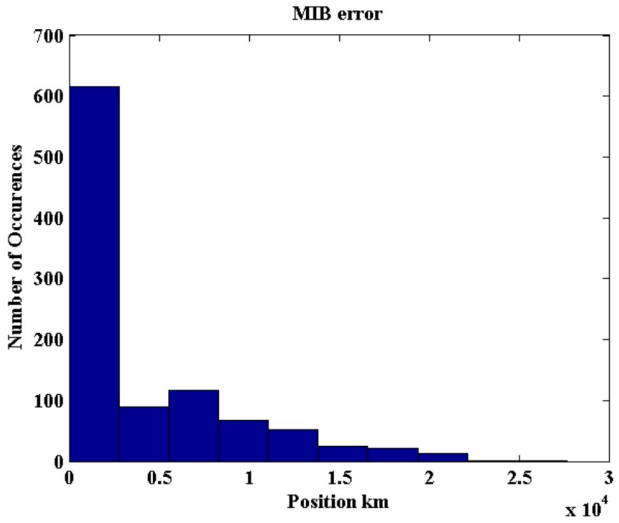


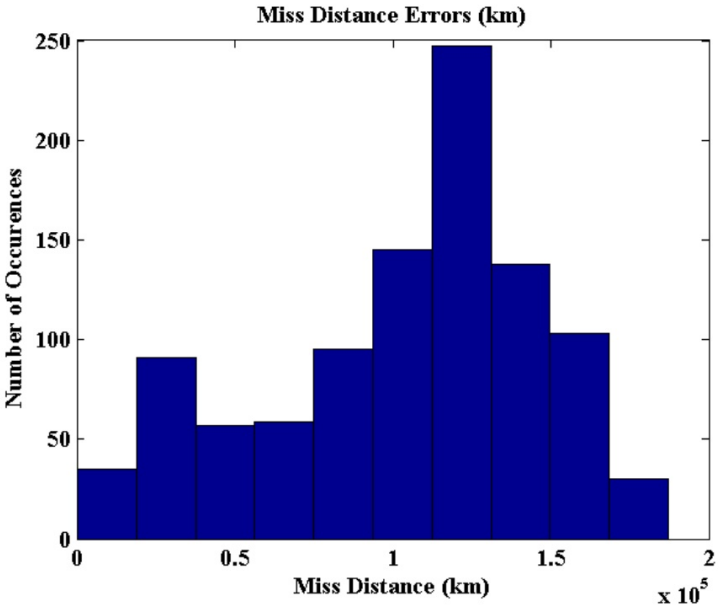
Step 3: Run
perturbed b iiss
distance, tra

Step 4: Return to Step 1.

Errors to halo orbits around Earth-Moon L1

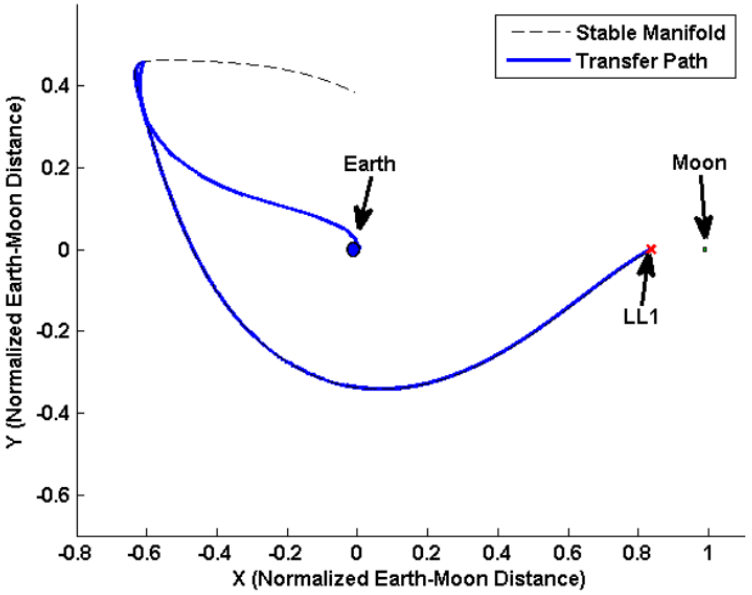


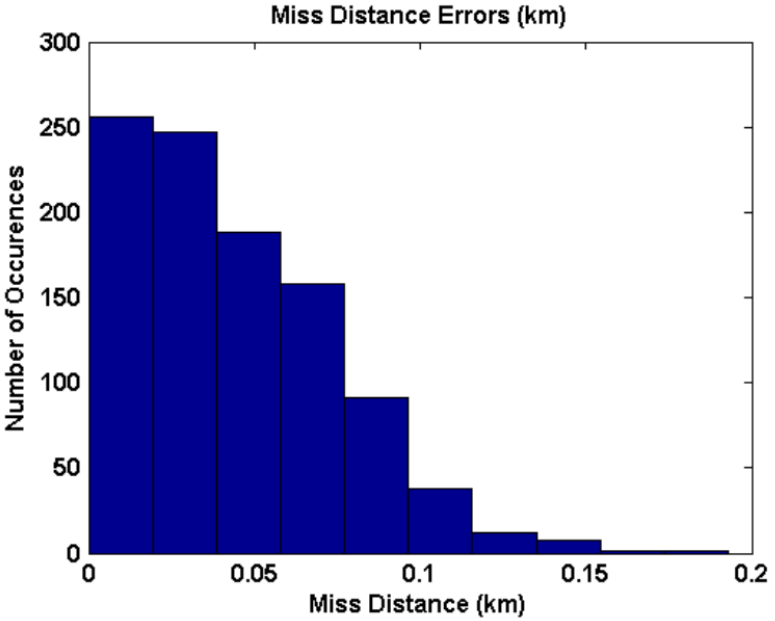


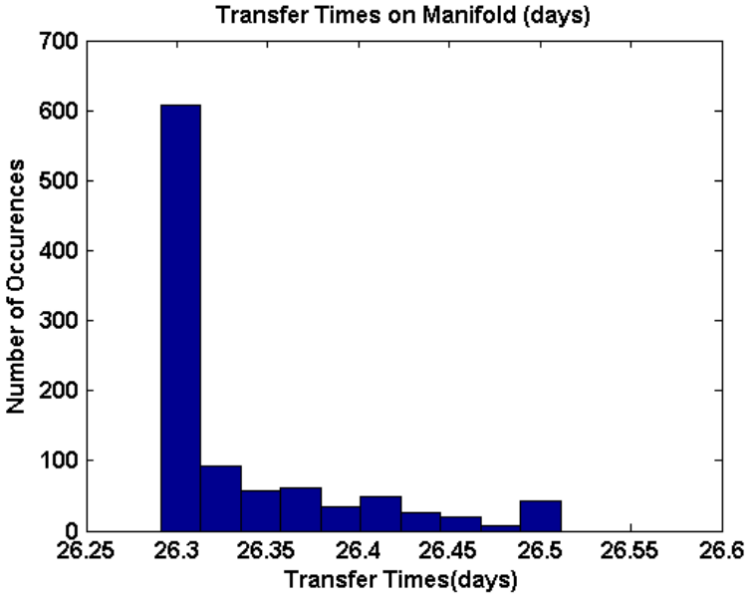


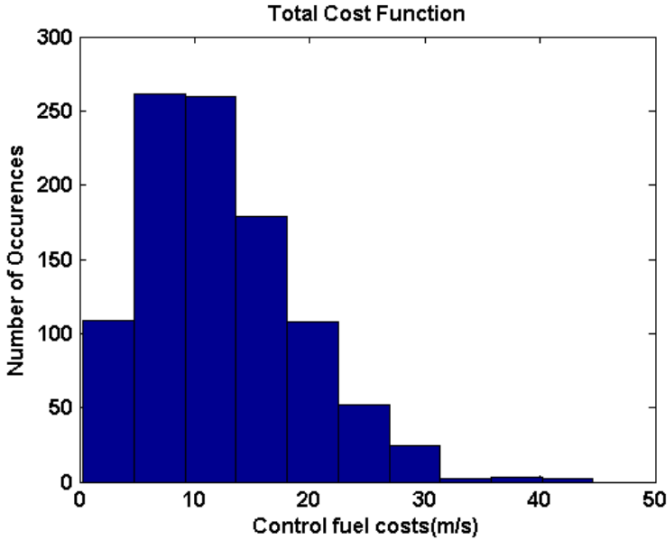
The above results necessitate a guidance control system

LQR

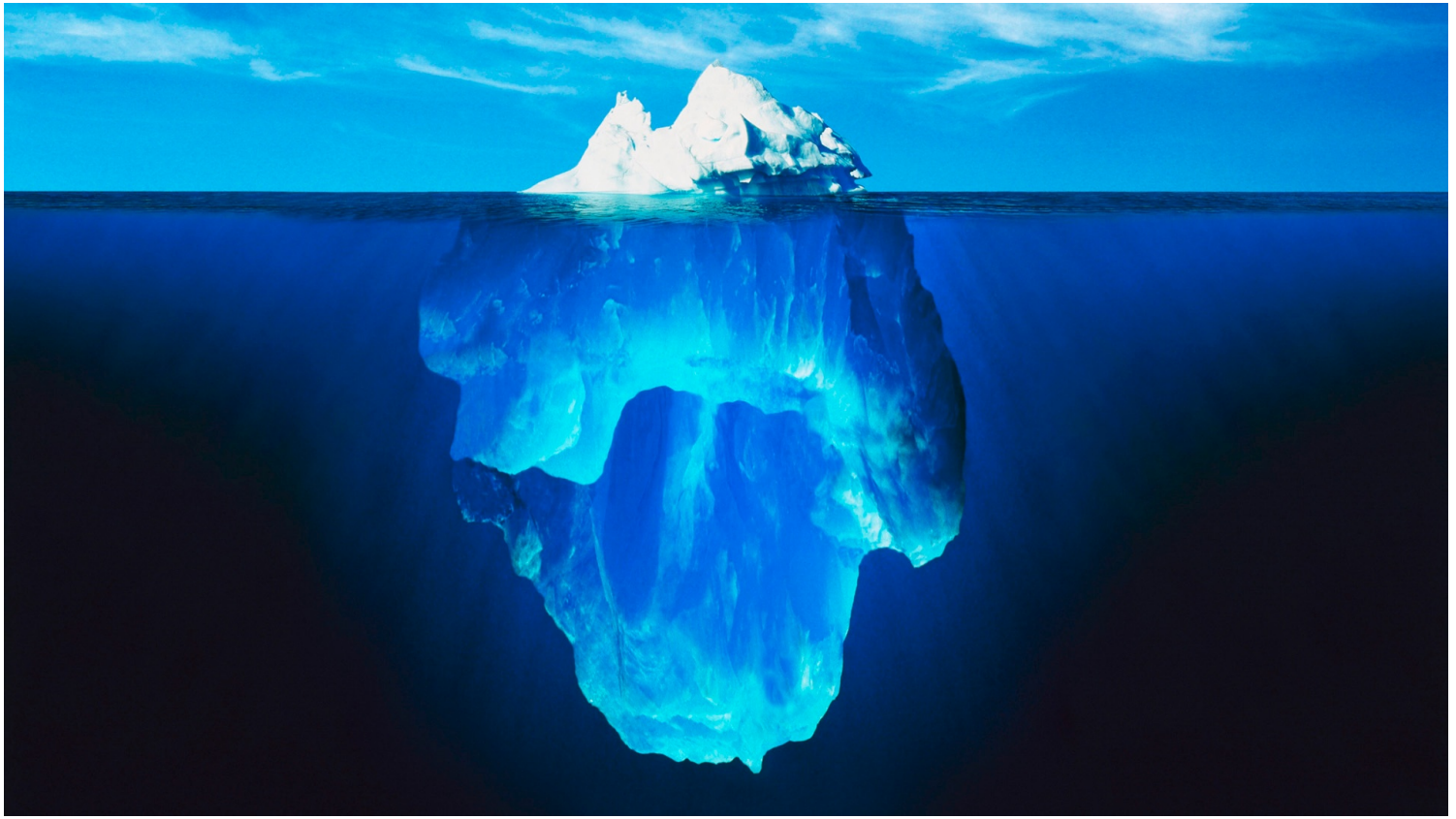








Conclusions



Monte Carlo methods are vast and have applications throughout mathematics, physics, engineering, chemistry, biology, economics, data science,...

With powerful computers literally in our pockets, these methods should be present in any problem-solver's toolbox