Wall Model PLR Integration

Shaun Alsum

Typical Case

$$\mathcal{L}_{M+P}\left((\mu,\vec{k})\middle|\vec{X}\right) = \mathrm{Pois}\left(n_{\mathrm{obs}};n_{\mathrm{exp}}\right)\cdot\prod_{\vec{x_i}\in\vec{X}}\left[\mu k_{\mathcal{S}}\mathcal{P}_{\mathcal{S}}(\vec{x_i})+\sum_j k_j\mathcal{P}_{\mathcal{B}j}(\vec{x_i})\right]\cdot\prod_{k_i\in\vec{k}}\mathcal{P}(k_j)$$

Extended term | Shape term | Profile term

$$\prod_{\vec{x_i}\in\vec{X}}\left[\mu k_{\mathcal{S}}\mathcal{P}_{\mathcal{S}}(s1_i,s2_i,r_i,d_i,\phi_i,t_i)+\sum_j k_j\mathcal{P}_{\mathcal{B}j}(s1_i,s2_i,r_i,d_i,\phi_i,t_i)\right]$$

$$\prod_{\vec{x_i}\in\vec{X}}\left[\mu k_{\mathcal{S}}(c_i)\mathcal{P}_{\mathcal{S}}(s1_i,s2_i\mid c_i)\mathcal{P}_{\mathcal{S}l}(r_i,d_i,\phi_i\mid c_i)+\sum_j k_j(c_i)\mathcal{P}_{\mathcal{B}j}(s1_i,s2_i\mid c_i)\mathcal{P}_{\mathcal{B}j}(r_i,d_i,\phi_i\mid c_i)\right]$$

Implemented via RooHistPdf(r, drift, phi) X RooHistPdf(s1, s2)

Wall Model

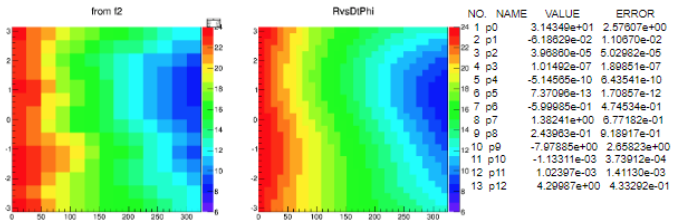
- Histogram in s1, s2, drift, phi tells you how many events are at each location in this 4d space.
- Gaussian in r tells you how they are distributed radially.

$$P\left(r,\phi,\mathrm{Drift},S2\right)=\frac{1}{\pi\sqrt{2\sigma}}\exp\left[-\frac{\left(r-r_{\mathrm{wall}}\right)^{2}}{2\sigma^{2}}\right]$$

- r_{wall} depends strongly on drift and phi
- σ depends strongly on s2 (and s2 on s1)
- Therefore the Gaussian is a fully 5d function.

```
return par[0] // Original Wall Position
       +par[1]*x[0]+par[2]*pow(x[0],2)+par[3]*pow(x[0],3)+par[4]*pow(x[0],4)+par[5]*pow(x[0],5)//
Shrinkage in drift time
       +par[6]*sin(x[1]*par[7]-par[8])// Magnitude, period and phase in phi
       +par[9]*sin(x[1]*x[0]*par[10]-par[11]*x[0]-par[12]); // cross dependency
```

x[0] : drift time;
x[1]: phi



NO.	NAME	VALUE	ERROR
1	p0	3.14349e+01	2.57607e+00
2	p1	-6.18629e-02	1.10670e-02
3	p2	3.96860e-05	5.02982e-05
4	p3	1.01492e-07	1.89851e-07
5	p4	-5.14565e-10	6.43541e-10
6	p5	7.37096e-13	1.70857e-12
7	p6	-5.99985e-01	4.74534e-01
8	p7	1.38241e+00	6.77182e-01
9	p8	2.43963e-01	9.18917e-01
10	p9	-7.97885e+00	2.65823e+00
11	p10	-1.13311e-03	3.73912e-04
12	p11	1.02397e-03	1.41130e-03
13	p12	4.29987e+00	4.33292e-01

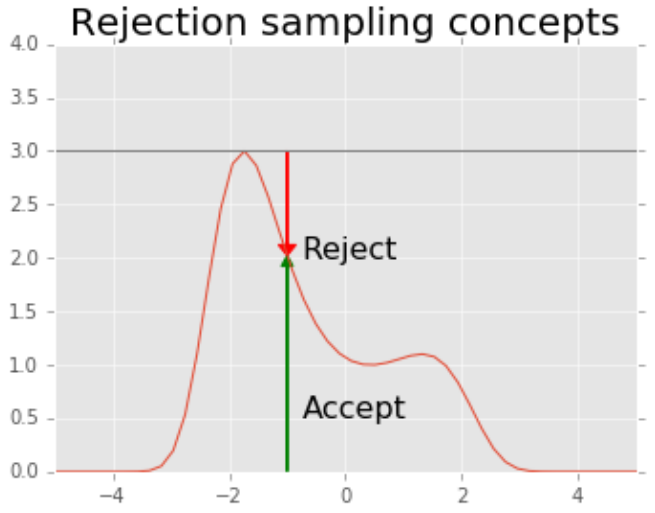
r_{wall} dependence on drift and phi

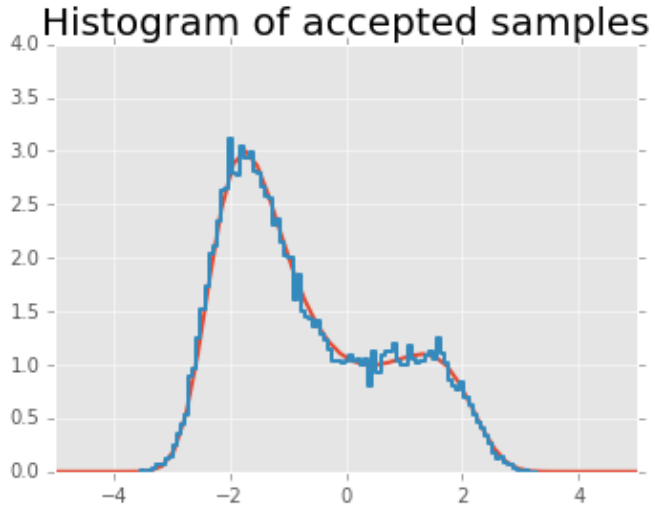
Wall Model Correlations

- s1, s2, drift strongly correlated.
- phi weakly correlated to drift

First Pass Implementation

- Try [Hist(s1, s2, drift) X Hist(phi)] x f(r | s1, s2, drift, phi)
- Implemented via RooHistPdf X RooHistPdf x Conditional(customRooPdf)
- Unfortunately, integration rather slow, event generation veeeeeery slow.
  - Uses numeric integration
  - Uses “Accept Reject” event generation

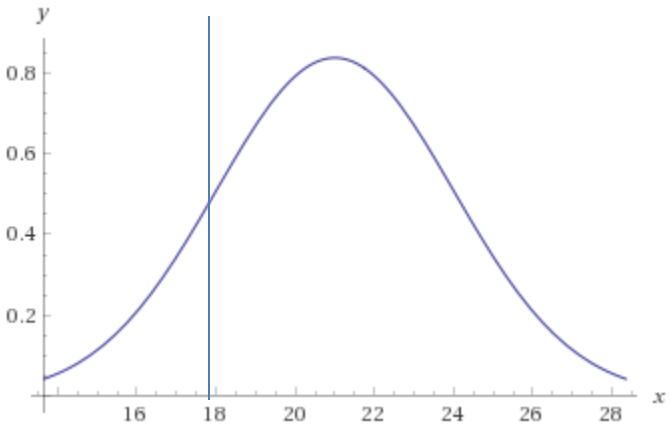




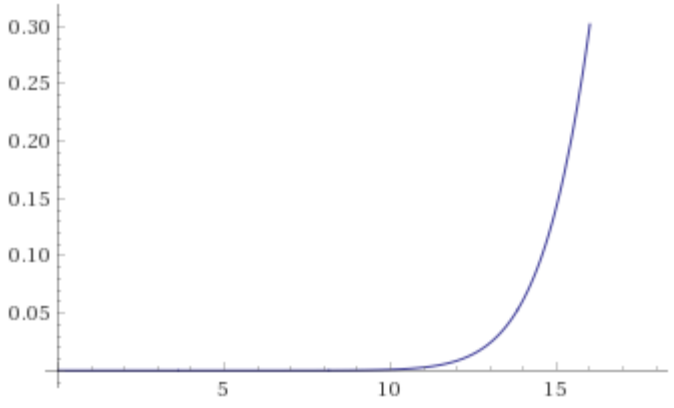
New Implementation

- Create 1 pdf object
  - Store a 3d vector array of number of events (s1, s2, drift)
  - Store a 1d vector that is a linearized version of this with each bin added to the previous to form a cumulative distribution functin (cdf)
  - Store a 1d vector array of number of events (phi)
  - Store the corresponding cdf
  - Define a function for the wall position r_{wall} = f(s1, s2, drift, phi)
  - Define a function for the std of the distribution r_{std} = f2(s1, s2, drift, phi)
  - Define the radial pdf as a function of the other observables
    - f(r | r_{wall}, r_{std}) = {Gaus(r | r_{wall}, r_{std}) if r < r_{wall} – 3, 0 otherwise}/Normalization factor
  - Define the inverse cdf of the radial function

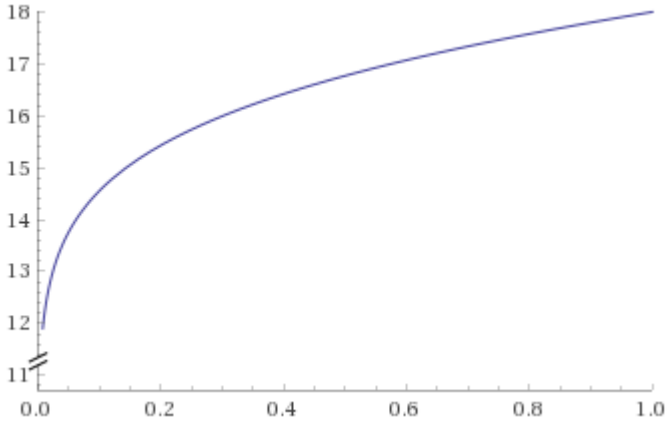
Gaussian



CDF



Inverse CDF



Integration

- Radial analytically integrates to 1 for any values of r_{wall} and r_{std} so integral is simply Sum(vector(s1, s2, drift)) * Sum(vector(phi))

- Event

Event Generation

- Generate 3 random numbers form 0 to 1:
  - s1S2Drift number
  - Phi number
  - r number
- Step through s1S2Drift cdf vector until we exceed the s1S2Drift random number, and randomly draw an event from the corresponding bin in (s1, s2, drift)
- Step through phi cdf vector until we exceed the phi random number, then draw an event from the corresponding bin
- Calculate the wall and std for these values of s1, s2, drift, and phi.
- Evaluate the inverse cdf function at these values with the r random number to obtain a value for r.

- This is at least 10^5 times faster than the default accept reject method, probably more. And is in fact faster than the 3d HistPdf generation.