Accelerated Computation of Matrix Elements for VBF H → WW* using GPUs

Throwing Darts at the Higgs

Kevin Multani

Special Thanks: Koos van Nieuwkoop, Daniel Mori, Alexander Held

Supervisor: Dr. Bernd Stelzer

ATLAS EXPERIMENT

INTRODUCTION

The Matrix Element Method (MEM) in the world of Particle Physics is a first-principles collision event classification technique. The MEM is based upon a purely theoretical construct (Fermi's Golden Rule), where the Lorentz invariant Matrix Element is calculated and is then used to determine the probability density, P_i of a specific particle final state, from a 2 → N process ***i***. While some competing multi-variate machine learning techniques require training on Monte Carlo simulated events, the MEM is unsupervised and its optimization is dependent only on theory. The MEM becomes computationally prohibitive due to calculating several high dimensional integrals per event. General Graphics Processing Unit Programming techniques speed up this computation and is used in the present study. We have extended a previous GPU MEM software package to include VBF Higgs → WW* and compared its performance to a multiprocessor CPU; the GPU used is: Tesla K40c (NVIDIA) and the 12 Core CPU used is: Xeon E5-2620 (Intel).

Why VBF H → WW*?

- The Standard Model predicts VBF H to WW*, so it provides a method of validating the model.
- This process coupled with the MEM makes it possible to measure physical parameters, such as Higgs coupling to the W's and Higgs CP.

PROBLEM STATEMENT

The idea of the MEM is to use theory to calculate a discriminant that will help us distinguish a **signal** event from a **background** event (see figure 1). Doing this reliably is the single most important aspect of any Higgs search analyses and is the backbone of the discovery.

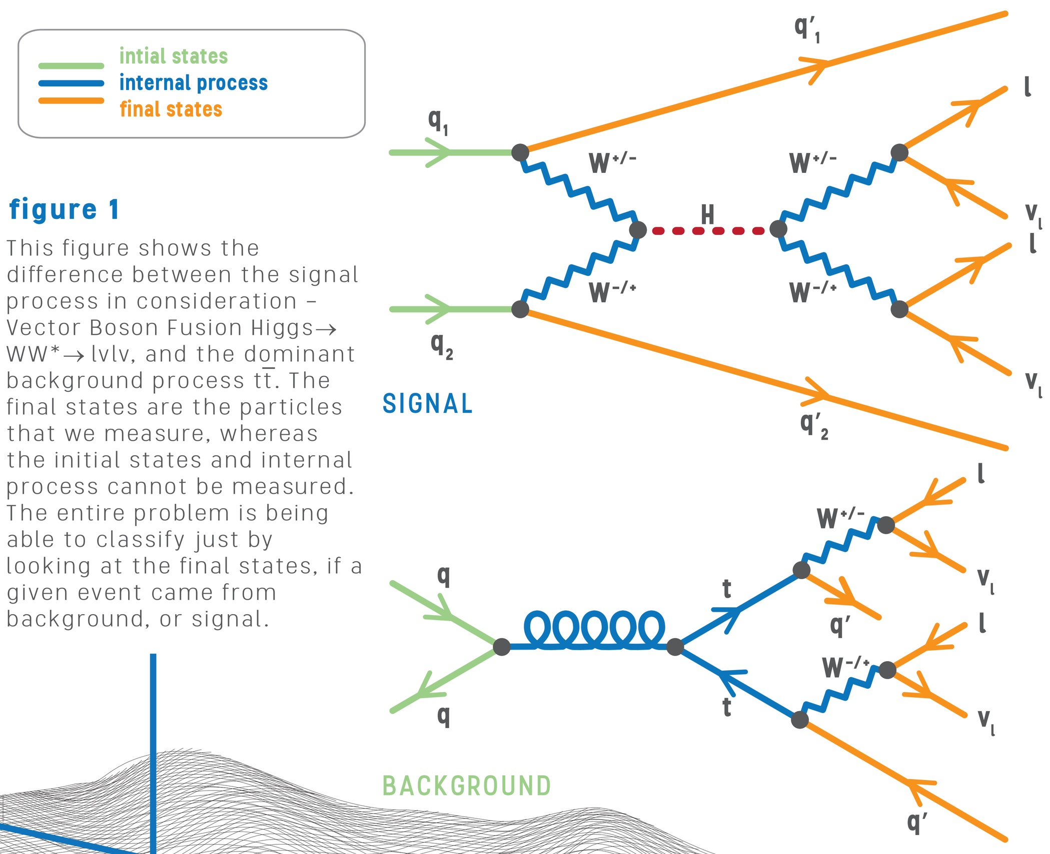


figure 1

This figure shows the difference between the signal process in consideration – Vector Boson Fusion Higgs→ WW*→ lνlν, and the dominant background process $t\bar{t}$. The final states are the particles that we measure, whereas the initial states and internal process cannot be measured. The entire problem is being able to classify just by looking at the final states, if a given event came from background, or signal.

MATRIX ELEMENT METHOD OVERVIEW

The **punchline**: the MEM is attempt to encode all available kinematic/dynamic information of an event into a single observable/discriminant.

Benefits of the technique:

- The likelihood, P_i directly depends on the physical parameters in the process
- Can calculate physical parameters directly from the matrix element
- Requires no training on large Monte Carlo datasets

One aspect of the MEM is that each event is assigned a real value which indicates how **signal**-like or **background**-like it is. We will refer this value as the event probability discriminant or simply just the discriminant:

$$D(m_H)=\log_{10}\frac{P_{VBF}(m_H)}{P_{t\bar{t}}} \qquad \text{eq. 1}$$

Where $P_{VBF}(m_H)$ is the likelihood of a candidate event being consistent with the Higgs boson mass hypothesis m_H and $P_{t\bar{t}}$ corresponds to the likelihood of the same event being consistent with the $t\bar{t}$ background hypothesis. Each likelihood can be computed by directly applying the MEM:

$$P_i = \frac{1}{\sigma_i}\sum_{\text{flavor}}\int_{V_n} M^2_i(\mathbf{Y})\,\frac{f_1(x_1,Q^2)f_2(x_2,Q^2)}{|\vec{q}_1|\cdot|\vec{q}_2|}\,d\Phi_1(q_1+q_2;y_1,..,y_n), \qquad \text{eq. 2}$$

parton distribtion functions f_1 and f_2 of colliding partons; vbf or $t\bar{t}$; total cross section for process **i**; sum over all flavors of incoming partons; integrate over n-dimensional space; initial and final state 4 momenta; Lorentz invariant matrix element

$$d\Phi_n(q_1+q_2;y_1,..,y_n) = (2\pi)^4\delta^4\Big(q_1+q_2-\sum_{i=1}^{n}y_i\Big)\prod_{i=1}^{n}\frac{d^3y_i}{(2\pi)^3 2E_i} \qquad \text{eq. 3}$$

4-momenta of incoming partons; 4-momenta of final state particles; conservation of energy of initial and final states; energies of the final state particles

A few remarks:

- In the present study, the sum over all incoming parton flavors is neglected and the total cross-sections are not divided.
- The final result, however, is still proportional to the likelihood P_i.
- The final state particles y_i can be constrained to achieve the reduction shown in figure 2, finally to give equation 4.

figure 2.

$$\underbrace{p_{l_1}, p_{\nu_2}, p_{l_1}, p_{\nu_2}, p_{q'}, p_{q'},}_{\text{final state 4-momenta}} \xrightarrow{\text{constraints}} \underbrace{p^{\nu1}{}_z, p^{\nu2}{}_x, p^{\nu2}{}_y, p^{\nu2}{}_z}_{\text{neutrino momenta}}$$

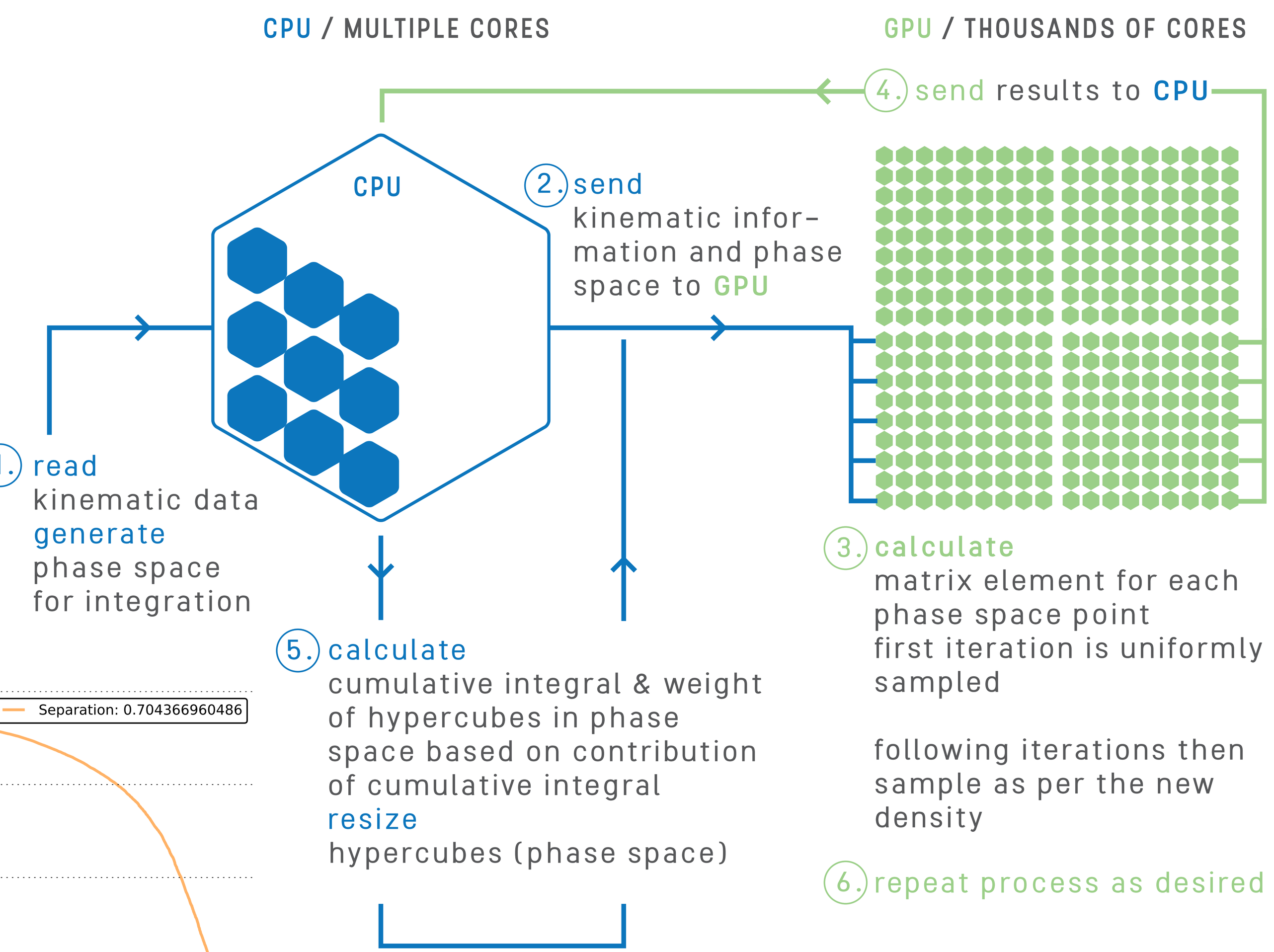
$$P_i\sigma_i \propto \hat{P}_i = \int \frac{|M_i|^2 f_1(x_1,Q^2)f_2(x_2,Q^2)}{8\cdot 16(2\pi)^8 x_1 x_2 E^2_{beam}}\,\frac{dp^{\nu1}{}_z\,dp^{\nu2}{}_x\,dp^{\nu2}{}_y\,dp^{\nu2}{}_z}{E_{l_1}E_{l_2}E_{\nu_1}E_{\nu_2}} \qquad \text{eq. 4}$$

GPU COMPUTING AND THE MEM

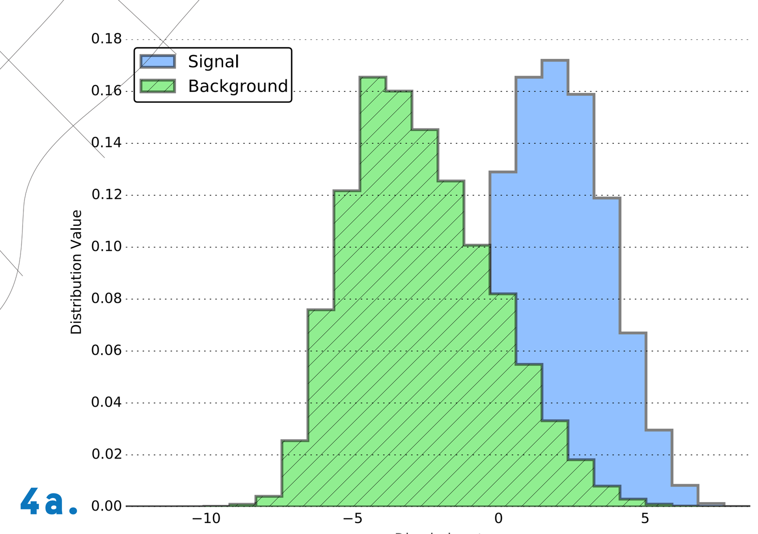
A simple way to understand the difference between a CPU and GPU is to compare how they process tasks. A CPU consists of a few cores optimized for sequential serial processing while a GPU has a massively parallel architecture consisting of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously.

figure 3

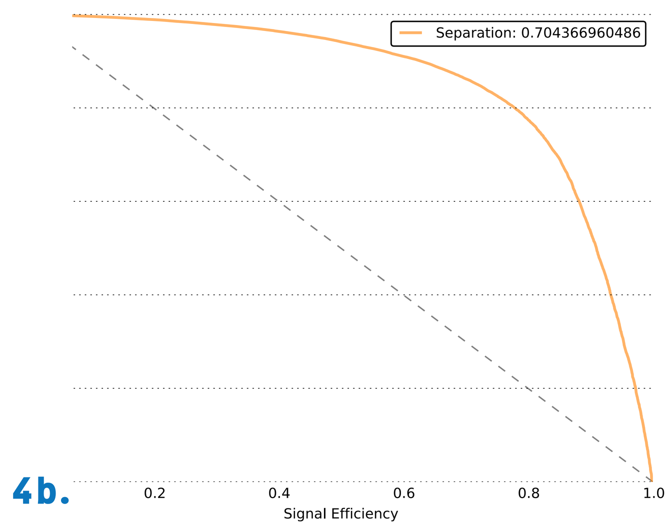
This figure shows a step by step process of the integration. The integration is spearheaded by a package called VEGAS. The final result is the result stated in equation 4. The integration begins with reading kinematic data from event files and generating the phase space associated (see figure 2). Then the kinematics and the phase space are sent to the GPU where the matrix element is calculated for each phase space point using equations 2 & 3. The results are then sent back to the CPU where VEGAS calculates weights based on the results and then resizes the hypercubes in the phase space (adaptive integration). A new set of phase space points are randomly generated from this new space and the process is repeated until desired.



RESULTS



4a.

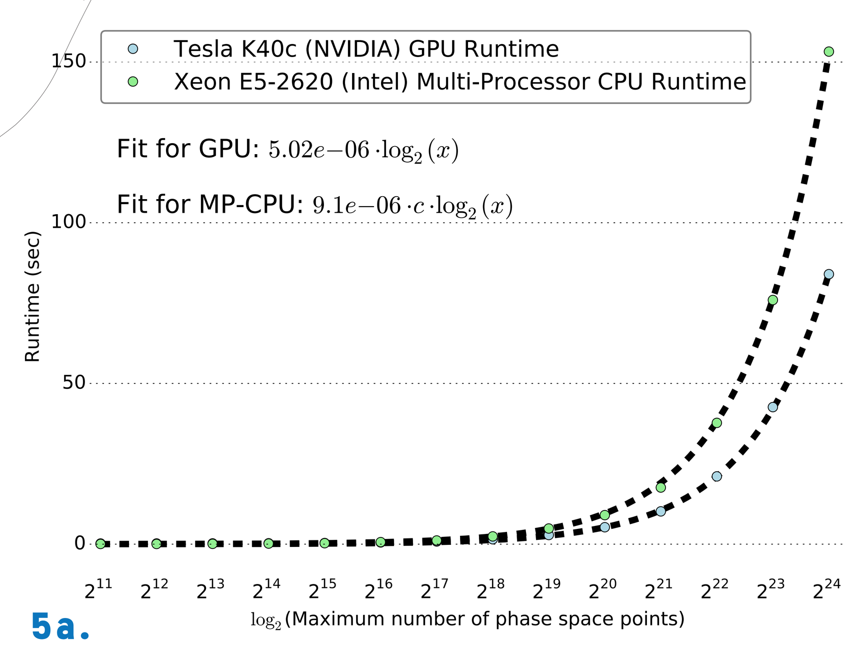


4b.

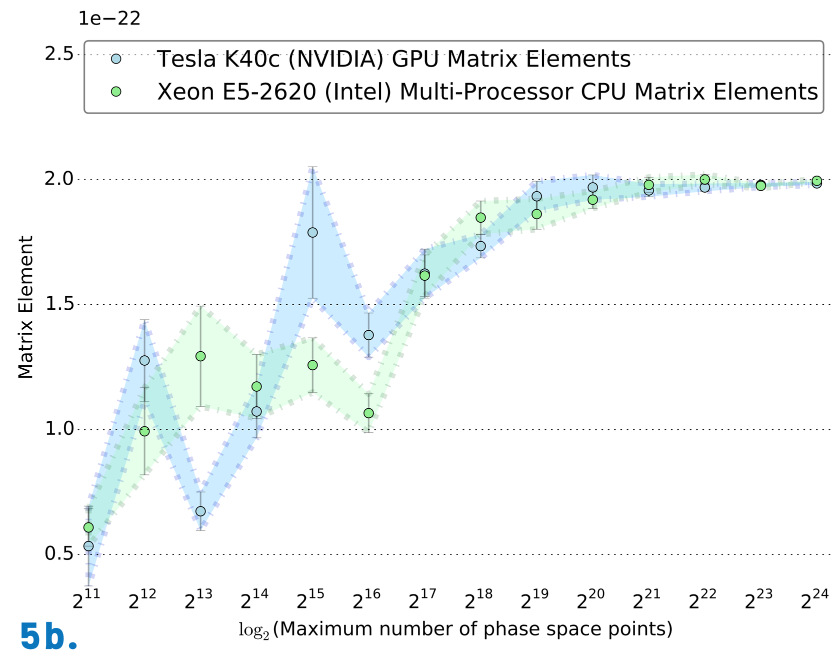
figure 4 a & b

a. This plot shows a normalized histogram of signal and background, with event weights considered. The discriminant is $D = \log_{10}(\hat{p}_{sig}/\hat{p}_{bg})$ (see equations 1 & 4). This distribution is proportional to the probability that a particular event is signal or background given a particular discriminant D. Given this histogram, the flattened background significance was calculated: Total S/sqrt(B) = 1.94 +/- 0.03.

b. This figure shows a ROC (receiving operating characteristic) curve. This plot illustrates the performance of the classification as the discriminant threshold is varied. The separation is calculated as the bounded area of the ROC curve and the lower triangle divided by the total area of the upper triangle. The separation in this case is 0.7044.



5a.



5b.

figure 5 a & b

a. This plot shows the GPU versus CPU performance. The x-axis is in a $\log_2$ scale for clarity. The fits indicate that the time-complexity within the CPU and GPU are both O(maximum number of phase space points). The speed up can be calculated by simply taking the ratio of the two functions: Speed up factor = MP-CPU/GPU = 1.813*c, where c is a scaling factor which depends on how many cores the MP-CPU was using. We found that, on average that c = 10, since 10 cores were being fully utilized during the duration of the profiling session. Also note that this constant, c, is a lower bound due to the fact that the MP-CPU has additional optimizations (hyperthreading, etc) which makes the effect of adding one more core non-linear. So, Speed Up Factor >= 18.13.

b. This plot just shows that the ME code seems to run properly in both the CPU and GPU, as the results for the ME converges for high enough phase space points. The calculations that do not agree within error bars can be neglected, as the end behavior is as expected

CONCLUSION AND OUTLOOK

The MEM allowed us to achieve a separation of approximately 70% considering only the $t\bar{t}$ background and a sensitivity of 1.94 +/- 0.03. A speed up of a factor of at least 18.13 was also observed (comparing the GPU to only 1 core of the CPU). The MEM analysis presented in this study is promising and can be matured to a fully-fledged first-principles technique.

Outlook:

- Improve convergence of the integral via a variable transformation
- Investigate other definitions of a discriminant to further improve separation and sensitivity
- Implement a Transfer Function to take measured four-momenta to truth four-momenta.
- Further GPU and GPU architecture optimization for greater speed up
- Consider all incoming parton flavors during the ME calculation

REFERENCES

http://www.nvidia.ca/object/what-is-gpu-computing.html
http://on-demand.gputechconf.com/gtc/2012/presentations/S0271-Fast-Adaptive-Sampling-Multi-Dimensional-Integral-Estimation-GPUs.pdf
Accelerated Matrix Element Method with Parallel Computing - Doug Schouten, Adam DeAbreu, Bernd Stelzer arXiv:1407.7595 [physics.comp-ph]
Koos Van Nieuwkoop – M. Sc. Thesis: Bringing the Higgs Boson to Rest

WestGrid UBC NSERC CRSNG SFU