# GPUs for Higgs boson data analysis at the LHC using the Matrix Element Method

*Doug Schouten*[1], *Adam DeAbreu*[2], *Bernd Stelzer*[2]

[1]TRIUMF, TRIUMF, 4004 Wesbrook Mall, Vancouver, BC, Canada
[2]Department of Physics, Simon Fraser University, 8888 University Dr, Burnaby, BC, Canada

**DOI:** http://dx.doi.org/10.3204/DESY-PROC-2014-05/20

The matrix element method utilizes ab initio calculations of probability densities as powerful discriminants to extract small signals from large backgrounds at hadron collider experiments. The computational complexity of this method for final states with many particles and degrees of freedom sets it at a disadvantage compared to supervised classification methods such as decision trees, k nearest-neighbour, or neural networks. We will present a concrete implementation of the matrix element technique in the context of Higgs boson analysis at the LHC employing graphics processing units. Due to the intrinsic parallelizability of multidimensional phase space integration, dramatic speedups can be readily achieved, which makes the matrix element technique viable for general usage at collider experiments.

## 1 Introduction

The matrix element method (MEM) in experimental particle physics is a unique analysis technique for characterizing collision events. When used to define a discriminant for event classification, it differs from supervised multivariate methods such as neural networks, decision trees, $k$-NN, and support vector machines in that it employs unsupervised *ab initio* calculations of the probability density $P_i$ that an observed collision event with a particular final state arises from $2 \to N$ scattering process $i$. Furthermore, the strong connection of this technique to the underlying particle physics theory provides key benefits compared to more generic methods:

1. the probability density $P_i$ directly depends on the physical parameters of interest;
2. it provides a most powerful test statistic for discriminating between alternative hypotheses, namely $P_i/P_j$ for hypotheses $i$ and $j$, by the Neyman-Pearson lemma;
3. it avoids tuning on unphysical parameters for analysis optimization[1];
4. it requires no training, thereby mitigating dependence on large samples of simulated events.

The MEM was first studied in [1] and was heavily utilized by experiments at the Tevatron for $W$ helicity [2] and top mass [3, 4] measurements, and in the observation of single top production

[1]Rather, optimization is determined by theoretical physics considerations, such as inclusion of higher order terms in the matrix element, or improved modeling of detector resolution.

[5], for example. It has also been used in Higgs searches at the Tevatron [6] and at the Large Hadron Collider (LHC) by both CMS [7] and ATLAS [8] collaborations. The MEM has also been extended in a general framework known as MADWEIGHT [11].

The MEM derives its name from the evaluation of $P_i$:

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

where $\mathcal{M}_i$ is the the Lorentz invariant matrix element for the $2 \to n$ process $i$, $\mathbf{Y}$ is shorthand notation for all the momenta $\vec{y}$ of each of the $n$ initial and final state particles, $f_1$ and $f_2$ are the parton distribution functions (PDF's) for the colliding partons, $\sigma$ is the overall normalization (cross-section), and

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

is the n-body phase space term. The momenta of the colliding partons are given by $q_1$ and $q_2$, and the fractions of the proton beam energy are $x_1$ and $x_2$, respectively. The sum in Equation (1) indicates a sum over all relevant flavor combinations for the colliding partons.

The association of the partonic momenta $\mathbf{Y}$ with the measured momenta $\mathbf{X}$ is given by a transfer function (TF), $T(\vec{x}; \vec{y})$ for each final state particle. The TF provides the conditional probability density function for measuring $\vec{x}$ given parton momentum $\vec{y}$. Thus,

$$\hat{p}_i = \int P_i \, T(\mathbf{X}; \mathbf{Y}) \, d\mathbf{Y}, \tag{3}$$

is the MEM probability density for an observed event to arise from process $i$ assuming the parton $\to$ observable evolution provided by the TF's. For well-measured objects like photons, muons and electrons, the TF is typically taken to be a $\delta$-function. For unobserved particles such as neutrinos, the TF is a uniform distribution. The TF for jet energies is often modeled with a double Gaussian function, which accounts for detector response (Gaussian core) and also for parton fragmentation outside of the jet definition (non-Gaussian tail). To reduce the number of integration dimensions, the jet directions are assumed to be well-modeled so that $T(\theta_x, \phi_x; \theta_y, \phi_y) = \delta(\theta^x - \theta^y)\delta(\phi^x - \phi^y)$.

Despite the advantages provided by the MEM enumerated above, an important obstacle to overcome is the computational overhead in evaluating $\geq 1$ multi-dimensional integrals for each collision event. For complex final states with many degrees of freedom (eg., many particles with broad measurement resolution, or unobserved particles), the time needed to evaluate $\hat{p}_i$ can exceed many minutes. In realistic use cases, the calculations must be performed multiple times for each event, such as in the context of a parameter estimation analysis where $\hat{p}_i$ is maximized with respect to a parameter of interest, or for samples of simulated events used to study systematic biases with varied detector calibrations or theoretical parameters. For large samples of events, the computing time can be prohibitive, even with access to powerful computer clusters. Therefore, overcoming the computation hurdle is a relevant goal.

This paper presents an implementation of the MEM using graphics processing units (GPU's). The notion of using GPU's for evaluating matrix elements in a multidimensional phase space has been investigated previously [12], although not in the context of the MEM. In order to ascertain the improvements in computing time when utilizing GPU's, the MEM was applied

in the context of a simplified $t\bar{t}H(\to b\bar{b})$ search in LHC Run II. Studying the $t\bar{t}H$ process is important in its own right [13, 14, 15, 16, 17]. Due to the complexity of the final state for this process, it is also an interesting use case in which to study the feasibility of the MEM with the improvements from highly parellelized multi-dimensional integration.

The note is organized as follows: in Section 2, the applicability of GPU architectures to the computational problem at hand is briefly outlined. In Section 3 a simplified $t\bar{t}H$ analysis is presented, which will be used to benchmark the improved computational performance afforded by modern GPU's. In Section 3.1 the specific implementation is outlined together with a summary of the results obtained from a number of GPU and CPU architectures.

## 2 Parallelized Integrand Evaluation

For dimensions $\geq 3$, evaluation of multidimensional integrals is typically only feasible using Monte Carlo methods. In these methods, the integrand

$$I = \int_{V_m} f(\vec{x})\, d\vec{x} \tag{4}$$

is approximated by a sum over randomly sampled points in the $m$-dimensional integration volume $V_m$

$$S_N \equiv V_m \underbrace{\frac{1}{N}\sum_{i=1}^{N} f(\vec{x}_i)}_{\equiv \overline{f}}, \tag{5}$$

which converges to $I$ by the law of large numbers. For all Monte Carlo integration algorithms, there is a trivial parallelization that can be achieved by evaluating the integrand $f(x_i)$ at points $\{x_i\}_{i=1,..,N}$ simultaneously, since the evaluation of the integrand at each point $x_i$ is independent of all other points $\{x_j\}_{j\neq i}$.

This mode of parallel evaluation is known as data parallelism, which is achieved by concurrently applying the same set of instructions to each data element. In practice, evaluating the functions used in the MEM involves conditional branching, so that the integrand calculation at each $x_i$ does not follow an identical control flow. Nevertheless, it is instructive to proceed with the ansatz of strict data parallelism.

Data parallelism maps very well to the single instruction, multiple data (SIMD) architecture of graphics processing units (GPU's). Modern GPU's contain many individual compute units organized in thread units. Within each thread unit, all threads follow the same instruction sequence[2], and have access to a small shared memory cache in addition to the global GPU memory.

The advent of general purpose programming on GPU's (GPGPU) has vastly increased the computing capability available on a single workstation, especially for data parallel calculations such as in the MEM. Two languages have gained traction for GPGPU, namely CUDA [20] (restricted to GPU's manufactured by NVidia) and OpenCL [21]. Both languages are based on C/C++[3].

[2]This has implications for code with complicated control flow, since threads will be locked waiting for other threads in the same unit to be syncronized in the instruction sequence. Careful tuning of the MEM function control flow and the thread unit sizes may improve the performance.

[3]In this work, the AMD Static C++ extensions to OpenCL [22] are used.

## 3 $t\bar{t}H(\rightarrow b\bar{b})$ Search

The fact that the Higgs coupling to the top quark is $\approx 1$ hints at a special role played by the top quark in electroweak symmetry breaking. Analysis of $t\bar{t}H(\rightarrow b\bar{b})$ production at the LHC can provide a powerful direct constraint on the fermionic (specifically, the top) couplings of the Higgs boson, with minimal model-dependence. The dominant backgrounds to this process, assuming at least one leptonic top decay, arise from the irreducible $t\bar{t}b\bar{b}$ background as well as the $t\bar{t}c\bar{c}$ and $t\bar{t}jj$ backgrounds via 'fake' $b$-tagged jets. The association of observed jets to

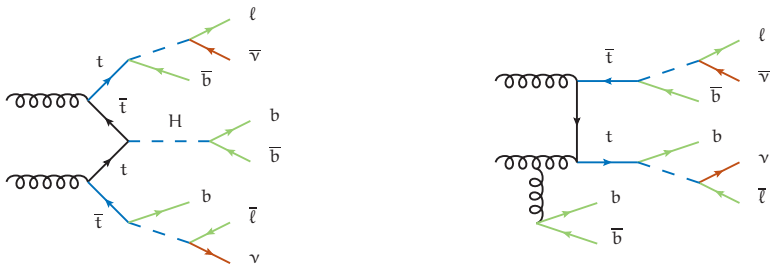


Figure 1: Representative Feynman diagrams for $t\bar{t}H(\rightarrow b\bar{b})$ production (left) and the irreducible $t\bar{t}b\bar{b}$ background (right) in the dilepton channel.

the external lines of the leading order (LO) Feynman diagrams in Figure 1 also gives rise to a combinatoric dilution of the signal, since there is an increased probability that a random pair of $b$ partons in a $t\bar{t}b\bar{b}$ event will have $m_{bb'} \approx m_H$. For the fully hadronic $t\bar{t}$ decay, there are 4! $\times$ 4! / 2= 288 combinations, assuming fully efficient $b$-tagging. This benchmark study is performed predominantly for the dileptonic $t\bar{t}$ decay mode, to showcase the application of the method for a very complex final state and also to reduce the large QCD backgrounds.

For the dilepton channel, the observable signature of the final state is $b\bar{b}\ell\bar{\ell} + \not{E}_T$, where $\not{E}_T = \vec{p}_T^{\,\nu} + \vec{p}_T^{\,\bar{\nu}}$. It is not possible to constrain the $z$ components of the neutrino momenta, and using transverse momentum balance removes only two of the remaining four degrees of freedom from the $x$ and $y$ components. Due to the broad resolution of the measured jet energy, there are also four degrees of freedom for the energy of the four $b$ quarks in the final state, so that the MEM evaluation implies an 8-dimensional integration:

$$\hat{p}_i = \frac{1}{\sigma_i} \sum_{\text{jet comb.}} \sum_{\text{flavor}} \int \mathcal{M}_i^2(\mathbf{Y}) \frac{f_1(x_1, Q^2)\, f_2(x_2, Q^2)}{|\vec{q}_1| \cdot |\vec{q}_2|}\, \Phi \cdot \tag{6}$$

$$\delta\left(p_x^{\nu} - \not{E}_T^{x} - p_x^{\bar{\nu}}\right)\, \delta\left(p_y^{\nu} - \not{E}_T^{y} - p_y^{\bar{\nu}}\right)\, d^3\vec{p}_{\nu}\, d^3\vec{p}_{\bar{\nu}} \prod_{j=1}^{N_{\text{jet}}=4} \mathrm{T}\left(E_j^{\text{jet}}; E_j\right) \cdot \left(E_j^2 \sin\theta_j\right) dE_j,$$

where $\Phi = (2\pi)^4 \delta^4(q_1 + q_2 - \sum_{i=1}^{n} p_y^i) \prod_{i=1}^{n} \frac{1}{(2\pi)^3 2E_i}$, and the integrals over the lepton momenta are removed by assuming infinitesimal measurement resolution. The outer sum is over all permutations of assigning measured jets to partons in the matrix element. A transformation to spherical coordinates has been performed $d^3\vec{p} \rightarrow p^2 \sin(\theta)\, dp\, d\theta\, d\phi$ for the jets, and $E$ is set to $|p|$.

The behaviour of the matrix element function $\mathcal{M}(\mathbf{Y})$ is strongly influenced by whether or not the internal propagators are on shell. It is difficult for numerical integration algorithms to efficiently map out the locations in momentum space of the external lines for which the internal lines are on shell. Therefore, it is advantageous to transform integration over the neutrino momenta to integrals over $q^2$ (where $q$ is the four momentum) of the top quark and $W$ boson propagators, so that the poles in the integration volume are along simple hyperplanes. This leads to the following coupled equations:

$$\begin{aligned}
\not{E}_x &= p_x^{\nu} + p_x^{\bar{\nu}} \\
\not{E}_y &= p_y^{\nu} + p_y^{\bar{\nu}} \\
q_{W^+}^2 &= (E_{\ell^+} + E_{\nu})^2 - (p_x^{\ell^+} + p_x^{\nu})^2 - (p_y^{\ell^+} + p_y^{\nu})^2 - (p_z^{\ell^+} + p_z^{\nu})^2 \\
q_{W^-}^2 &= (E_{\ell^-} + E_{\bar{\nu}})^2 - (p_x^{\ell^-} + p^{\bar{\nu}_x})^2 - (p_y^{\ell^-} + p^{\bar{\nu}_y})^2 - (p_z^{\ell^-} + p^{\bar{\nu}_z})^2 \\
q_t^2 &= (E_b + E_{\ell^+} + E_{\nu})^2 - (p_x^b + p_x^{\ell^+} + p_x^{\nu})^2 - \\
&\quad (p_y^b + p_y^{\ell^+} + p_y^{\nu})^2 - (p_z^b + p_z^{\ell^+} + p_z^{\nu})^2 \\
q_{\bar{t}}^2 &= (E_{\bar{b}} + E_{\ell^-} + E_{\bar{\nu}})^2 - (p_x^{\bar{b}} + p_x^{\ell^-} + p_x^{\bar{\nu}})^2 - \\
&\quad (p_y^{\bar{b}} + p_y^{\ell^-} + p_y^{\bar{\nu}})^2 - (p_z^{\bar{b}} + p_z^{\ell^-} + p_z^{\bar{\nu}})^2,
\end{aligned} \tag{7}$$

which have been solved analytically in [23]. For the $t\bar{t}H(\to b\bar{b})$ process, there is an additional very narrow resonance from the Higgs propagator. For the same reasoning as above, the following transformation of variables for $E_1$ and $E_2$ are employed, which are the energies of the $b$-quarks from the Higgs decay, respectively:

$$\begin{aligned}
f &= (E_1 + E_2) \\
m_H^2 &= (E_1 + E_2)^2 - |\vec{p}_1|^2 - |\vec{p}_2|^2 - 2\,|\vec{p}_1|\,|\vec{p}_2|\cos\Delta\theta_{1,2},
\end{aligned} \tag{8}$$

where $|\vec{p}| = \sqrt{E^2 - m^2}$. Figure 1 highlights the internal lines which are used in the integration.

## 3.1 Analysis and Results

The evaluation of the integrand in Equation (6) is broken into components for the matrix element $\mathcal{M}(\mathbf{Y})$, the PDF's, the TF's and the phase space factor. Each of these components is evaluated within a single GPU "kernel" program for each phase space point. Code for evaluating $\mathcal{M}$ is generated using a plugin developed for MadGraph [24]. This plugin allows one to export code for an arbitrary $2 \to N$ process from MadGraph to a format compatible with OpenCL, CUDA, and standard C++. This code is based on HELAS functions [25, 26]. Compilation for the various platforms is controlled with precompiler flags. Model parameters, PDF grids and phase space coordinates are loaded in memory and transferred to the device[4] (GPU) whereafter the kernel is executed. The PDF's are evaluated within the kernel using wrapper code that interfaces with LHAPDF [27] and with the CTEQ [28] standalone PDF library. The PDF data is queried from the external library and stored in $(x, Q^2)$ grids for each parton flavor $(d, u, s, c, b)$, which are passed to the kernel program. The PDF for an arbitrary point is evaluted using bilinear interpolation within the kernel. The precision of the interpolation is within 1% of the

[4] In the case of CPU-only computation, the transfer step is unnecessary.

| Configuration | Details | Peak Power | Cost (USD, 2014) |
|---|---|---|---|
| CPU | Intel Xeon CPU E5-2620 0 @ 2.00GHz (single core) using `gcc` 4.8.1 | 95W | 400 |
| CPU (MP) | Intel Xeon CPU E5-2620 0 @ 2.00GHz (six cores + hyperthreading) using AMD SDK 2.9 / OpenCL 1.2 | 95W | 400 |
| GPU | AMD Radeon R9 290X GPU (2,816 c.u.) using AMD SDK 2.9 / OpenCL 1.2 on Intel Xeon CPU E5-2620 | 295W | 450 |
| $GPU_x$ | same configuration as GPU, but with minor code modifications to accomodate GPU architecture | | |

Table 1: Details of the hardware configurations used to benchmark the MEM for GPU and (multicore) CPU's. The peak power is as reported by the manufacturer. The cost is listed in USD for the CPU or GPU only. For the GPU configuration, the code was identical to that used for the CPU configurations. For the $GPU_x$ configuration, the code was modified to accommodate the specific GPU architecture.

values directly queried from the PDF library. An event discriminant $D$ is constructed as

$$D = \log_{10}\left(\frac{\hat{p}_{t\bar{t}H}}{\hat{p}_{t\bar{t}b\bar{b}}}\right) \tag{9}$$

and evaluated for a sample of signal ($t\bar{t}H$) and background ($t\bar{t}b\bar{b}$) events generated in MADGRAPH and interfaced with PYTHIA for the parton shower, [29] using the so-called Perugia tune [30]. Jets are reconstructed using the anti-$k_T$ algorithm described in [31] with width parameter $d = 0.4$. Any jets overlapping with leptons within $d$ are vetoed, and $b$-tagging is performed by matching jets to the highest energy parton within $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2} < d$. A transfer function is defined for $b$-jets by relating the jet energy to the energy of the matched parton using a double Gaussian distribution.

The analysis is performed at two levels, namely

1. parton level: using the parton momenta from MADGRAPH-generated events directly (by assuming $\delta$-function TF's for all final state particles), and averaging over all permutations for the assignment of the $b$ partons in each event;

2. hadron level: using the outputs from PYTHIA and selecting events with four $b$-jets, averaging over all the permutations and integrating over the full 8-dimensional phase space as in Equation (6).

The convenient PyOpenCL and PyCUDA [32] packages are used to setup and launch OpenCL and CUDA kernels. Using OpenCL one can also compile the MEM source for a CPU target, and is thereby able to parallelize the MEM across multiple cores (see Table 1). In order to perform the numerical integration, a modified VEGAS implementation in Cython/Python [33] is used. This implementation has a number of improvements compared to previous versions and, importantly, interfaces with the provided integrand function by passing the full grid of phase space points as a single function argument. This allows one to pass the whole integration grid to the OpenCL or CUDA device at once, which facilitates the desired high degree of parallelism. The parton and hadron level MEM is performed with various hardware configurations specified in Table 1. In all configurations except the one labelled $GPU_x$, the MEM code used is identical. For the $GPU_x$ case, minor modifications were made to replace particular array variables with sets of scalar variables.

### 3.1.1 Parton Level

Here, the evaluation is performed using the parton momenta, so that all the transfer functions become $\delta$-functions, and the evaluation of $D$ does not involve any numberical integration. In

this benchmark, all possible combinations of $b$-quarks in the final state are summed. The distribution of $D$ for the signal and background samples is shown in Figure 2. A comparison of the time needed to evaluate $\hat{p}_i$ for all events is shown in Table 2 for various CPU and GPU configurations.

| Process | CPU | CPU (MP) | GPU | $\text{GPU}_x$ | CPU / $\text{GPU}_x$ |
|---|---|---|---|---|---|
| signal | 255 | 29 | 1.8 | 0.7 | 364 |
| background | 661 | 91 | 12 | 5.4 | 122 |

Table 2: Processing time, in seconds, required to evaluate the matrix elements for $10^5$ events at parton level, for the various configurations detailed in Table 1. Using GPU's reduces the processing time by a factor greater than 120× compared to a single CPU core for the $t\bar{t}b\bar{b}$ matrix element.

### 3.1.2 Hadron Level

The analysis at hadron level is closer to what can be optimally achieved in a real world collider experiment. Only the momenta of stable, interacting particles are accessible, and the jet energy resolution must be taken into account. The calculation of $\hat{p}_i$ requires evaluating the eight-dimensional integral in Equation (6). The integration variable transformation for $t\bar{t}H$ and $t\bar{t}b\bar{b}$ matrix element integrals presented in Section 2 are used. At each phase space point in the sum of Equation (5), the $\not{E}_T$ used in Equation (7) is defined as

$$\not{E}_{x,y} = -\left( p^{\ell^+}_{x,y} + p^{\ell^-}_{x,y} + \sum_{j \,\in\, \text{jets}} p^{j}_{x,y} \right). \tag{10}$$

The processing times per event for the hadron level MEM calculation are shown in Table 3. The relative improvement for the GPU is significantly reduced compared to the parton level analysis. This arises from a number of differences for this scenario. First, the VEGAS stratified sampling and adaptive integration algorithm is run on the CPU in all cases, which damps the GPU improvements in the integrand evaluation. Second, in the evaluation of the integral of Equation (6), significant additional complexity is demanded to solve Equations (7) and (8). Due to the cancellation of large coefficients in these solutions, double floating point precision is required, which reduces the GPU advantage since double precision calculations are performed significantly slower on most GPU's. Furthermore, the number of intermediate variables is significantly larger, which is found to increase the number of processor registers used. Since the number of registers available to each thread unit (or "wavefront" in the parlance of OpenCL) is limited to at most 256 for the GPU used in this study, the overall duty factor of the GPU is significantly reduced, to as low as 10%, since the full number of threads available in each block could not be utilized. It is anticipated that careful tuning of the code to accommodate GPU architecture could greatly improve the relative performance.

## 3.2 Further Results in the Lepton + Jets Channel

A brief summary is given on the evaluation of the GPU performance on a recent implementation of the $t\bar{t}H(\to b\bar{b})$ search in the lepton + jets channel. The corresponding Feynman diagrams for

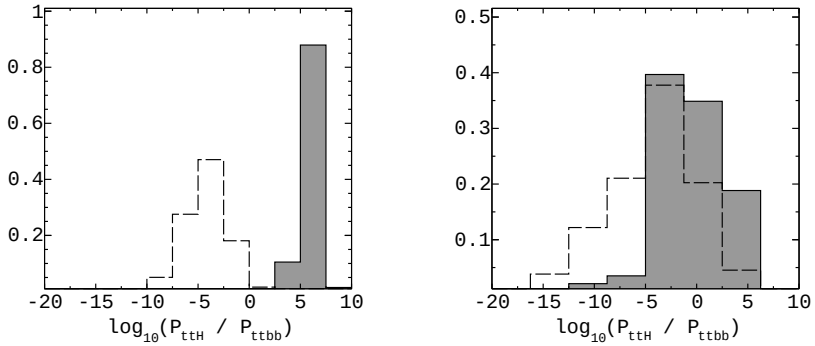


Figure 2: The event discriminant $D$ for $t\bar{t}H$ (filled) and $t\bar{t}b\bar{b}$ (dashed line) events at parton level (left) and at hadron level (right). The distributions are normalized to unit area.

| Process | CPU | CPU (MP) | GPU | GPU$_x$ | CPU / GPU$_x$ |
|---|---|---|---|---|---|
| signal | 312 | 36.2 | 7.5 | 5.9 | 52.0 |
| background | 405 | 55.1 | 9.1 | 7.1 | 57.3 |

Table 3: Processing time required to evaluate the matrix elements for a single event at hadron level, for the various configurations detailed in Table 1. Note that this includes a full 8-dimensional integration over phase space for each event. Using GPU's reduces the processing time by at least 50×.

signal and background are shown in Figure 3. In this case, only a 6-dim phase space integral has to be evaluated (instead of 8-dim in the case of the dilepton final state). The variable

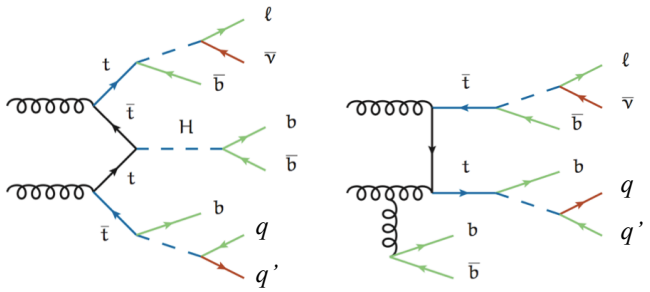


Figure 3: Representative Feynman diagrams for $t\bar{t}H(\to b\bar{b})$ production (left) and the irreducible $t\bar{t}b\bar{b}$ background (right) in the lepton + jets channel

transformation is much simpler. The results of the new study are summarized in Table 4. Using GPU's reduces the processing time by at least 100× for the lepton + jets final state. The overall duty factor of the GPU is increased compared to the dilepton analysis to about 20%.

| Process | CPU | CPU (MP) | $GPU_x$ | CPU / $GPU_x$ |
|---|---|---|---|---|
| signal | ≈1000 | 91 | 7.9 | 125 |
| background | ≈3800 | 347 | 35 | 109 |

Table 4: Processing time required to evaluate the matrix elements for a single event at hadron level, for the various configurations detailed in Table 1. Note that this includes a full 6-dimensional integration over phase space for each event. Using GPU's reduces the processing time by at least 100×.

# 4 Conclusions

The matrix element method can be computationally prohibitive for complex final states. The $t\bar{t}H(\to b\bar{b})$ benchmark study in this paper has shown that by exploiting the parallel architectures of modern GPU's, computation time can be reduced by a factor $\geq 100$ for the matrix element method, at about 10-20% utilization of the GPU. It is anticipated that careful code modifications can add significant further improvements in speed. This can be the subject of future study. However, even with the performance gains in this benchmark study, it is clear that for the MEM, the computing time required with O(10) GPU's is equivalent to a medium-sized computing cluster with O(400) cores (along with its required support and facilities infrastructure). This provides the potential to apply the method generally to searches and measurements with complex final states in experimental particle physics.

The programs described in this work are generic in nature, such that GPU-capable MEM code can be readily derived for an arbitrary $2 \to N$ process with only few modifications to accommodate transformations of variables or transfer functions. It is envisaged that future work can automate the inclusion of NLO matrix elements and transformations of variables (as in MadWeight) for the matrix element method, thereby providing an optimal methodology for classification and parameter estimation in particle physics.

# References

[1] Kunitaka Kondo. Dynamical Likelihood Method for Reconstruction of Events with Missing Momentum. I. Method and Toy Models. *J. Phys. Soc. Jap.*, 57(12):4126–4140, 1988.

[2] V.M. Abazov, et al. Helicity of the W boson in lepton + jets $t\bar{t}$ events. *Phys. Lett. B*, 617:1–10, 2005.

[3] Abulencia, A. et al. Precision measurement of the top-quark mass from dilepton events at cdf ii. *Phys. Rev. D*, 75:031105, Feb 2007.

[4] V.M. Abazov et al. A precision measurement of the mass of the top quark. *Nature*, 429:638–642, 2004.

[5] Aaltonen, T., et al. Observation of single top quark production and measurement of $|V_{tb}|$ with CDF. *Phys. Rev. D*, 82:112005, Dec 2010.

[6] Aaltonen, T., et al. Search for a Standard Model Higgs Boson in $WH \to \ell\nu b\bar{b}$ in $p\overline{p}$ Collisions at $\sqrt{s} = 1.96$ TeV. *Phys. Rev. Lett.*, 103:101802, Sep 2009.

[7] Serguei Chatrchyan et al. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys.Lett.*, B716:30–61, 2012.

[8] Search for the Standard Model Higgs boson in the $H \to WW^{(*)} \to \ell\nu\ell\nu$ decay mode using Multivariate Techniques with 4.7 fb-1 of ATLAS data at $\sqrt{s} = 7$ TeV. Technical Report ATLAS-CONF-2012-060, CERN, Geneva, June 2012.

[9] F. Fiedler, A. Grohsjean, P. Haefner, and P. Schieferdecker. The Matrix Element Method and its Application to Measurements of the Top Quark Mass. *NIM A*, 624(1):203–218, 2010.

[10] A. Grohsjean. *Measurement of the Top Quark Mass in the Dilepton Final State Using the Matrix Element Method*. 2010.

[11] P. Artoisenet, V. Lemaitre, F. Maltoni, and O. Mattelaer. Automation of the matrix element reweighting method. *JHEP*, 12(68), 2010.

[12] K. Hagiwara, J. Kanzaki, Q. Li, N. Okamura, and T. Stelzer. Fast computation of MadGraph amplitudes on graphics processing unit (GPU). *European Physical Journal C*, 73:2608, 2013.

[13] F. Englert and R. Brout. Broken symmetry and the mass of gauge vector mesons. *Phys. Rev. Lett.*, 13:321–323, Aug 1964.

[14] Higgs, Peter W. Broken Symmetries and the Masses of Gauge Bosons. *Phys. Rev. Lett.*, 13:508–509, Oct 1964.

[15] G. S. Guralnik, C. R. Hagen, and T. W. B. Kibble. Global conservation laws and massless particles. *Phys. Rev. Lett.*, 13:585–587, Nov 1964.

[16] S. Dittmaier, S. Dittmaier, C. Mariotti, G. Passarino, R. Tanaka, et al. Handbook of LHC Higgs Cross Sections: 2. Differential Distributions. 2012.

[17] C. Degrande, J.M. Gerard, C. Grojean, F. Maltoni, and G. Servant. Probing Top-Higgs Non-Standard Interactions at the LHC. *JHEP*, 1207:036, 2012.

[18] G. Peter Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27(2):192 – 203, 1978.

[19] G. Peter Lepage. VEGAS: An Adaptive Multi-dimensional Integration Program. Technical Report CLNS 80-447, Cornell, March 1980.

[20] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, March 2008.

[21] John E. Stone, David Gohara, and Guochun Shi. Opencl: A parallel programming standard for heterogeneous computing systems. *IEEE Des. Test*, 12(3):66–73, May 2010.

[22] Ofer Rosenberg, Benedict R. Gaster, Bixia Zheng, Irina Lipov. *OpenCL Static C++ Kernel Language Extension*, 04 edition, 2012.

[23] Lars Sonnenschein. Algebraic approach to solve $t\bar{t}$ dilepton equations. *Phys. Rev. D*, 72:095020, Nov 2005.

[24] Alwall, Johan and Herquet, Michel and Maltoni, Fabio and Mattelaer, Olivier and Stelzer, Tim. MadGraph 5 : Going Beyond. *JHEP*, 1106:128, 2011.

[25] H. Murayama, I. Watanabe, and Kaoru Hagiwara. HELAS: HELicity amplitude subroutines for Feynman diagram evaluations. 1992.

[26] Priscila de Aquino, William Link, Fabio Maltoni, Olivier Mattelaer, and Tim Stelzer. ALOHA: Automatic Libraries Of Helicity Amplitudes for Feynman Diagram Computations. *Comput.Phys.Commun.*, 183:2254–2263, 2012.

[27] M.R. Whalley, D. Bourilkov, and R.C. Group. The Les Houches accord PDFs (LHAPDF) and LHAGLUE. 2005.

[28] Lai, Hung-Liang and Guzzi, Marco and Huston, Joey and Li, Zhao and Nadolsky, Pavel M. and others. New parton distributions for collider physics. *Phys.Rev.*, D82:074024, 2010.

[29] Torbjorn Sjostrand, Stephen Mrenna, and Peter Z. Skands. PYTHIA 6.4 Physics and Manual. *JHEP*, 0605:026, 2006.

[30] Peter Zeiler Skands. Tuning Monte Carlo Generators: The Perugia Tunes. *Phys.Rev.*, D82:074018, 2010.

[31] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. FastJet User Manual. *Eur.Phys.J.*, C72:1896, 2012.

[32] Andreas Klöckner, Nicolas Pinto, Yunsup Lee, B. Catanzaro, Paul Ivanov, and Ahmed Fasih. PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation. *Parallel Computing*, 38(3):157–174, 2012.

[33] G. Peter Lepage. *vegas Documentation*, 2.1.4 edition, 2014.