Eur. Phys. J. C (2010) 70: 513–524
DOI 10.1140/epjc/s10052-010-1465-5

THE EUROPEAN PHYSICAL JOURNAL C

Special Article - Tools for Experiment and Theory

Calculation of HELAS amplitudes for QCD processes using graphics processing unit (GPU)

K. Hagiwara[1], **J. Kanzaki**[2,a], **N. Okamura**[2,b], **D. Rainwater**[3], **T. Stelzer**[4,c]

[1] KEK Theory Center and Sokendai, Tsukuba 305-0801, Japan
[2] KEK, Tsukuba 305-0801, Japan
[3] Space and Geophysics Laboratory, Applied Research Laboratories, University of Texas, Austin, TX 78758, USA
[4] Department of Physics, University of Illinois, Urbana, IL, USA

Received: 9 February 2010 / Revised: 14 July 2010 / Published online: 19 October 2010

Abstract We use a graphics processing unit (GPU) for fast calculations of helicity amplitudes of quark and gluon scattering processes in massless QCD. New HEGET (**H**ELAS **E**valuation with **G**PU **E**nhanced **T**echnology) codes for gluon self-interactions are introduced, and a C++ program to convert the MadGraph generated FORTRAN codes into HEGET codes in CUDA (a C-platform for general purpose computing on GPU) is created. Because of the proliferation of the number of Feynman diagrams and the number of independent color amplitudes, the maximum number of final state jets we can evaluate on a GPU is limited to 4 for pure gluon processes ($gg \to 4g$), or 5 for processes with one or more quark lines such as $q\overline{q} \to 5g$ and $qq \to qq + 3g$. Compared with the usual CPU-based programs, we obtain 60–100 times better performance on the GPU, except for 5-jet production processes and the $gg \to 4g$ processes for which the GPU gain over the CPU is about 20.

1 Introduction

In our previous report [1] we introduced a C-language [2] version of the HELAS codes [3, 4], HEGET (**H**ELAS **E**valuation with **G**PU **E**nhanced **T**echnology), which can be used to compute helicity amplitudes on a GPU (Graphics Processing Unit). Encouraging results with 40–150 times faster computation speed over the CPU performance were obtained for pure QED processes, $q\overline{q} \to n\gamma$, for $n = 2$ to 8 in pp collisions.

In this paper, we extend our study to QCD processes with massless quarks and gluons. The HEGET routines for massless quarks and gluons are identical to those of quarks and photons introduced in [1], and the qqg vertex function structure is also the same as the $qq\gamma$ functions. The only new additional routines are those for ggg and $gggg$ vertices. For the QED processes studied in Ref. [1], we found that the present CUDA compiler cannot process $q\overline{q} \to 6\gamma$ amplitude with $6! \approx 700$ Feynman diagrams, and we need to subdivide the HEGET codes into small pieces for 6γ and 7γ processes. In the case of 8γ production with $8! \approx 4 \times 10^4$ Feynman diagrams, we have not been able to compile the program even after subdivision into small pieces. We also encountered serious slow down when the program accesses global memory during the parallel processing period. Therefore, our concern for evaluating the QCD processes on a GPU is the proliferation of the number of diagrams, as well as the number of independent color amplitudes which come with different color weights.

The paper is organized as follows. In Sect. 2, we present the cross section formula for n-jet production processes in pp collisions in the quark-parton model, or in the leading order of perturbative QCD with scale-dependent parton distribution functions (PDF's). In Sect. 3, we review briefly the structure of GPU computing by using HEGET codes, and give basic parameters of the GPU and CPU machines used in this analysis. In Sect. 4, we introduce new HEGET functions for ggg and $gggg$ vertices. Section 5 gives our results and Sect. 6 summarizes our findings. Appendix lists all the new HEGET codes introduced in Sect. 4.

[a] e-mail: junichi.kanzaki@kek.jp
[b] e-mail: nokamura@yamanashi.ac.jp
[c] e-mail: tstelzer@uiuc.edu
N. Okamura is now at Faculty of Engineering, University of Yamanashi, Kofu, Japan.

2 Physics process

2.1 n-jet production in pp collisions

The cross section for n-jet production processes can be expressed as

$$\mathrm{d}\sigma = \sum_{\{a,b\}} \iint \mathrm{d}x_a\,\mathrm{d}x_b\, D_{a/p}(x_a, Q)\, D_{b/p}(x_b, Q)\, \mathrm{d}\hat{\sigma}(\hat{s}), \quad (1)$$

where $D_{a/p}$ and $D_{b/p}$ are the scale (Q) dependent parton distribution functions (PDF's), x_a and x_b are the momentum fractions of the partons a and b, respectively, in the right- and left-moving protons. For the total pp collision energy of $\sqrt{s}$,

$$\hat{s} = s x_a x_b, \quad (2)$$

gives the invariant mass squared of the hard collision process

$$\mathbf{a} + \mathbf{b} \to \mathbf{1} + \mathbf{2} + \cdots + \mathbf{n}. \quad (3)$$

The subprocess cross section is computed in the leading order as

$$\mathrm{d}\hat{\sigma}(\hat{s}) = \frac{1}{2\hat{s}} \frac{1}{2\cdot 2} \sum_{\lambda_i} \frac{1}{n_a n_b} \sum_{c_i} \left|\mathcal{M}^{c_i}_{\lambda_i}\right|^2 \mathrm{d}\Phi_n, \quad (4)$$

where

$$\mathrm{d}\Phi_n = (2\pi)^4\, \delta^4\left(p_a + p_b - \sum_{i=1}^{n} p_i\right) \prod_{i=1}^{n} \frac{\mathrm{d}^3 p_i}{(2\pi)^3 2\omega_i}, \quad (5)$$

is the invariant n-body phase space, λ_i are the helicities of the initial and final partons, n_a and n_b are the color degree of freedom of the initial partons, a and b, respectively, and c_i represents the color indices of the initial and final partons. When there are more than one gluons or identical quarks in the final states, an appropriate statistical factor should be multiplied on the phase space $\mathrm{d}\Phi_n$ in (5).

The Helicity amplitudes for the process (3)

$$\mathbf{a}(p_a, \lambda_a, c_a) + \mathbf{b}(p_b, \lambda_b, c_b) \to \mathbf{1}(p_1, \lambda_1, c_1) + \cdots + \mathbf{n}(p_n, \lambda_n, c_n) \quad (6)$$

can be expressed as

$$\mathcal{M}^{c_i}_{\lambda_i} = \sum_{l \in \text{diagram}} (M_{\lambda_i})^{c_i}_{l} \quad (7)$$

where the summation is over all the Feynman diagrams. The subscripts λ_i stand for a given combination of helicities (± 1 for both quarks and gluons in the HELAS convention [3, 4]), and the subscripts c_i correspond to a set of color indices (1, 2, 3 for flowing-IN quarks, $\overline{1}, \overline{2}, \overline{3}$ for flowing-OUT quarks, and 1 to 8 for gluons). In MadGraph [5] the amplitudes are expanded as

$$\mathcal{M}^{c_i}_{\lambda_i} = \sum_{\alpha} T^{c_i}_{\alpha} (J_{\lambda_i})_{\alpha} \quad (8)$$

in the color bases $T^{c_i}_{\alpha}$ which are made from the SU(3) generators in the fundamental representation [6]

The color factors are computed as

$$\mathcal{N}_{\alpha\beta} = \frac{1}{n_a n_b} \sum_{c_i} \left(T^{c_i}_{\alpha}\right)\left(T^{c_i}_{\beta}\right)^* \quad (9)$$

where $n_{a,b} = 3$ for q and $\overline{q}$, $n_{a,b} = 8$ for gluons, and the summation is over all $\{c_i\} = \{c_a, c_b, c_1, \ldots, c_n\}$. The color sum-averaged square amplitudes are computed as

$$\overline{\sum_{c_i}} \left|\mathcal{M}^{c_i}_{\lambda_i}\right|^2 = \sum_{a,b} (J_{\lambda_i})_{\alpha} \mathcal{N}_{\alpha\beta} (J_{\lambda_i})^*_{\beta}. \quad (10)$$

The cross sections are then expressed as

$$\mathrm{d}\hat{\sigma}(\hat{s}) = \frac{1}{2\hat{s}} \overline{\sum_{\lambda_i}} \overline{\sum_{c_i}} \left|\mathcal{M}^{c_i}_{\lambda_i}\right|^2 \mathrm{d}\Phi_n, \quad (11)$$

where we introduce the helicity sum-average symbol as

$$\overline{\sum_{\lambda_i}} \equiv \frac{1}{2}\frac{1}{2} \sum_{\lambda_i}. \quad (12)$$

In this paper the following three types of multi-jet production processes are computed:

$$gg \to gg, ggg, gggg, \quad (13a)$$

$$u\overline{u} \to gg, ggg, gggg, ggggg, \quad (13b)$$

$$uu \to uu, uug, uugg, uuggg. \quad (13c)$$

The number of contributing Feynman diagrams and the number of color bases for the above processes are summarized in Table 1, which includes those for the process, $gg \to 5g$. We note here that the number of diagrams (7245) for $gg \to 5g$ exceeds that of the $u\overline{u} \to 7\gamma$ process ($7! \approx 5040$), for which we could run the converted MadGraph codes on a GPU, only after division into small pieces [1]. In fact, we have not been able to run the $gg \to 5g$ program on GPU even after dividing the program into more than 100 pieces; as explained in Sect. 5.4.

Proliferation of the number of independent color basis vectors is also a serious concern for GPU computing, since the color matrix $\mathcal{N}$ of (9) has $m(m+1)/2$ elements when there are m independent basis vectors $T^{c_i}_{a}$. For example, the process $uu \to uuggg$ has $m = 240$ color basis vectors from Table 1, and the matrix has 3×10^4 elements. The matrix exceeding 16000 elements cannot be stored in the 64 kB

Table 1 The number of Feynman diagrams and the color bases for QCD processes studied in this paper

No. of jets	$gg \to$ gluons		$u\overline{u} \to$ gluons		$uu \to uu +$ gluons	
in the final state	#diagrams	#colors	#diagrams	#colors	#diagrams	#colors
2	6	6	3	2	2	2
3	45	24	18	6	10	8
4	510	120	159	24	76	40
5	7245	720	1890	120	786	240

constant memory, while storing it in the global memory will result in serious loss of efficiency in parallel computing. Therefore, the method to handle summation over color degrees of freedom is a serious concern in GPU computing.

2.2 Selection criteria for jets

Total and differential cross sections of the processes (13) in pp collisions at $\sqrt{s} = 14$ TeV are computed in this paper. We introduce final state cuts for all the jets as follows:

$$|\eta_i| < \eta^{\rm cut} = 2.5, \tag{14a}$$

$$p_{{\rm T}i} > p_{\rm T}^{\rm cut} = 20\ {\rm GeV}, \tag{14b}$$

$$p_{{\rm T}ij} > p_{\rm T}^{\rm cut} = 20\ {\rm GeV}, \tag{14c}$$

where η_i and $p_{{\rm T}i}$ are the rapidity and the transverse momentum of the i-th jet, respectively, in the pp collisions rest frame along the right-moving ($p_z = |p|$) proton momentum direction, and $p_{{\rm T}ij}$ is the relative transverse momentum [7] between the jets i and j defined by

$$p_{{\rm T}ij} \equiv \min(p_{{\rm T}i}, p_{{\rm T}j})\Delta R_{ij}, \tag{15a}$$

$$\Delta R_{ij} = \sqrt{\Delta\eta_{ij}^2 + \Delta\phi_{ij}^2}. \tag{15b}$$

Here ΔR_{ij} measures the boost-invariant angular separation between the jets.

As for the parton distribution function (PDF), we use the set CTEQ6L1 [8] and the factorization scale is chosen to be the cut-off $p_{\rm T}$ value, $Q = p_{\rm T}^{\rm cut} = 20$ GeV. The QCD coupling constant is also fixed as

$$\alpha_s = \alpha_s(Q = 20\ {\rm GeV})_{\overline{\rm MS}} = 0.171, \tag{16}$$

which is obtained from the $\overline{\rm MS}$ coupling at $Q = m_Z$, $\alpha_s(m_Z)_{\overline{\rm MS}} = 0.118$ [9] by using the NLO renormalization group equations with 5-flavors.

3 Computation on the GPU

3.1 GPU and its host PC

For the computation of the cross sections of QCD n-jet production processes we use the same GPU and host PC as in the previous report [1]. In particular we use a GeForce GTX280 by NVIDIA [10] with 30 Streaming Multiprocessors (SM) where each SM has 8 Streaming Processors (SP), whose parameters are summarized in Table 2. It is controlled by a Linux PC with Fedora 8 on a CPU whose properties are summarized in Table 3.

Programs which are used for the computation of the cross sections are developed with the CUDA [2] (ver.2.1) environment introduced by NVIDIA [10] for general purpose GPU computing. All programs which are executed on the CPU are compiled by using gcc4.1 with the compile option of `-O3`.

3.2 Program structure

Our program computes the total cross sections and distributions of the QCD n-jet production processes via the following procedure:

Table 2 Parameters of GeForce GTX280

Number of Streaming Multiprocessors	30
Number of Streaming Processors	240
Total amount of global memory	1 GB
Total amount of constant memory	64 kB
Total amount of shared memory per block	16 kB
Total number of registers available per block	16 k
Clock rate	1296 MHz

Table 3 Host PC environment

CPU	Core2Duo 3 GHz
L2 Cache	6 MB
Memory	4 GB
Bus Speed	1.333 GHz
OS	Fedora 8 (64 bit)

1. initialization of the program,
2. random number generation for multiple phase-space points $\{p_a, p_b, p_1, \ldots, p_n\}$ and helicities $\{\lambda_i\}$ on the CPU,
3. transfer of random numbers to the GPU,
4. generation of helicities and momenta of initial and final partons using random numbers, and compute amplitudes $(J_{\lambda_i})_a$ of (8) for all the color bases on the GPU,
5. multiplying the amplitudes and their complex conjugate with the color matrix $\mathcal{N}_{ab}$ of (9) and summing them up as in (10), and multiply the PDF's of the incoming partons on the GPU,
6. transferring momenta and helicities for external particles, computed weights and the color summed squared amplitudes to the CPU, and
7. summing up all values to obtain the total cross section and distributions on the CPU.

Program steps between the generation of random numbers (2) and the summation of computed cross sections (7) are repeated until we obtain sufficient statistics for the cross section and all distributions.

3.3 Color matrix calculation

In order to compute the cross sections of the QCD multi-jet production processes, multiplications of the large color matrix $\mathcal{N}_{ab}$ of (9), the vector of color-bases amplitudes $(J_{\lambda i})_\alpha$ of (8) and its complex conjugate have to be performed, as in (10). For large n-jet processes, like $gg \to 4$ gluons, $u\overline{u} \to 5$ gluons and $uu \to uu + 3$ gluons, the dimensions of color matrices exceed 100, and the number of multiplication becomes larger than 10^4. These matrices cannot be stored in the constant memory (64 kB for the GTX280; see Table 2) which is accessed in parallel, while storing them in the global memory (1GB for GTX280) results in serious slow-down of the GPU. We find that multiplications for the color-summation in (10) can be reduced significantly as follows.

The color matrix of (9) contains many elements with the same value. We count the number of different non-zero elements in the color matrix and find the results shown in Table 4. We find for instance that among the $240 \times (240 + 1)/2 = 28,720$ elements of the color matrix for the $uu \to uu + 3g$ process, there are only 60 unique ones.

In general, the number of different elements in the color matrix grows linearly rather than quadratically as the number of color basis vectors grows. Since the numbers in Table 4 are small enough, we can store them in the constant memory which is accessed quickly by each parallel processor.

Before we arrive at the above solution adopted in this study, we examined the possibility of summing over colors via Monte Carlo. Let us briefly report, in passing, on this exercise.

Table 4 Number of different non-zero elements in the color matrix of (9)

No. of jets	$gg \to$ gluons	$u\overline{u} \to$ gluons	$uu \to uu +$ gluons
2	3	2	2
3	7	4	7
4	15	9	19
5	45	24	60

In the Monte Carlo color summation approach, we evaluate the matrix element $\mathcal{M}^{c_i}_{\lambda_i}$ (7) for a given set of momenta $\{p_i\}$, helicities $\{\lambda_i\}$ and colors $\{c_i\}$, and sum the squared amplitudes over randomly generated sets of $\{p_i, \lambda_i, c_i\}$. This method turns out not to be efficient because in the color basis using the fundamental representation of the SU(3) generators adopted by MadGraph, most of the basis vectors $T^{c_i}_a$ vanish for a given color configuration $\{c_i\}$. As an example, $gg \to 4g$ has $5! = 120$ color basis vectors (see Table 1), which take the form

$$T^{c_i}_\alpha = \mathrm{Tr}\left(T^{a_1} T^{a_2} \cdots T^{a_6}\right) \tag{17}$$

for the configuration $\{c_i\} = (a_1, a_2, \ldots, a_6)$ where a_i denotes the color index of the gluon i taking an integer value between 1 and 8. Among the $8^6 \approx 260{,}000$ configurations, only 12% give non-zero values. Moreover, as many as 75% of the color configurations give vanishing results for all the 120 basis vectors. Although the efficiency can be improved by changing the color basis, we find that our solution of evaluating the exact summation over colors is superior to the Monte Carlo summation method for all the processes which we report in this paper.

3.4 VVV: three vector boson vertex

4 New HEGET functions

The HEGET functions for massless quarks and gluons are the same as those introduced in the previous report [1]. The qqg vertex functions are identical to the $qq\gamma$ functions of Ref. [1] except for the coupling constant;

$$eQ_q \to g_s T^a_{i\bar{j}} \tag{18}$$

for the vertex

$$\mathcal{L}_{qqg} = -g_s T^a_{i\bar{j}} A^a_\mu(x) \overline{q}_{\bar{i}}(x) \gamma_\mu q_j(x) \tag{19}$$

where $g_s = \sqrt{4\pi\alpha_s}$ is the strong coupling constant and $T^a_{i\bar{j}}$ is an SU(3) generator in the fundamental representation. For example, the qqg vertex function is computed by the HEGET function `iovxx0` as

Table 5 List of the new vertex functions in HEGET

Vertex	Inputs	Output	HEGET Function	HELAS Subroutine
VVV	VVV	Amplitude	`vvvxxx`	`VVVXXX`
	VV	V	`jvvxx0`	`JVVXXX`
VVVV	GGGG	Amplitude	`ggggxx`	`GGGGXX`
	GGG	G	`jgggx0`	`JGGGXX`

```
iovxx0(cmplx* fi, cmplx* fo, cmplx* vc,
                    float* g, cmplx vertex)
```
(20)

where the coupling constants are

$$\texttt{g[0]} = \texttt{g[1]} = g_s \tag{21}$$

following the convention of MadGraph [5] and the color amplitude is[1]

$$-T^a_{i\bar{j}}(\textbf{vertex}). \tag{22}$$

In the rest of this section, we introduce new HEGET functions for three-vector boson (VVV) and four-vector boson (VVVV) vertices. All the new HEGET functions are listed in Table 5, and their contents are given in Appendix. Also shown in Table 5 is the correspondence between the HEGET functions and the HELAS subroutines [3, 4].

For the ggg vertex

$$\mathcal{L}_{ggg} = g_s f^{abc}\big(\partial^\mu A^{a\nu}(x)\big)A^b_\mu(x)A^c_\nu(x) \tag{23}$$

we introduce two HEGET functions, `vvvxxx` and `jvvxx0`. They correspond to HELAS subroutines, `VVVXXX`, and `JVVXXX`, respectively.

The HEGET function `vvvxxx` (A.1.1) computes the complex amplitude (**vertex**) from three gluon wave functions `ga`, `gb` and `gc` each carrying the color a, b and c, respectively. The color amplitude corresponding to the Lagrangian (23) is

$$if^{abc}(\textbf{vertex}) \tag{24}$$

A factor of i appears because the original HELAS subroutine `GGGXXX` is identical to the `W3WXXX` subroutine [3, 4] that computes the $WW\gamma$ and WWZ vertices with real coupling constants.

The HEGET function `jvvxxx` (A.1.2) computes the off-shell wave function $\boldsymbol{j}^\mu(c;ab)$ of a gluon with color index c made from two gluon wave functions `ga` and `gb`, each carrying the color a and b, respectively. The off-shell gluon wave function is associated with the color factor

$$if^{abc}\boldsymbol{j}^\mu(c;a,b), \tag{25}$$

according to the Lagrangian (23).

4.1 VVVV: four vector boson vertex

For the $gggg$ vertex

$$\mathcal{L}_{gggg} = -\frac{g_s^2}{4} f^{abe}f^{cde}A^{a\mu}(x)A^{bv}(x)A^c_\mu(x)A^d_v(x) \tag{26}$$

we introduce two HEGET functions, `ggggxx` and `jgggx0`, listed in Table 5. They correspond to HELAS subroutines, `GGGGXX` and `JGGGXX`, respectively.

The HEGET function `ggggxx` (A.2.1) computes the complex amplitude $\boldsymbol{v}(ab,cd)$ from 4 gluon wave functions `ga`, `gb`, `gc` and `gd` each carrying the color a, b, c and d, respectively, when the associated color factor is $f^{abe}f^{cde}$. The vertex amplitude for the Lagrangian (26) is obtained by calling the function three times such that the color amplitude becomes

$$f^{abe}f^{cde}\boldsymbol{v}(ab,cd) + f^{ace}f^{dbe}\boldsymbol{v}(ac,db) + f^{ade}f^{bce}\boldsymbol{v}(ad,bc). \tag{27}$$

Finally the HEGET function `jgggxx` (A.2.2) computes the off-shell wave function $\boldsymbol{j}^\mu(d;ab,c)$ of a gluon with color d made from three gluon wave functions `ga`, `gb` and `gc`, each carrying the color index of a, b and c, respectively, when the associated color factor is $f^{abe}f^{cde}$. The off-shell wave function from the Lagrangian (26) is obtained by calling the function 3 times as

$$f^{abe}f^{cde}\boldsymbol{j}^\mu(d;ab,c) + f^{ace}f^{dbe}\boldsymbol{j}^\mu(d;ca,b) + f^{ade}f^{bce}\boldsymbol{j}^\mu(d;bc,a), \tag{28}$$

just as in (27) for the amplitude.

5 Results

5.1 Comparison of total cross sections

We have validated all the HEGET functions by comparing the helicity amplitudes of each process for many phase

[1] The sign of the color amplitudes (22) and (24) follows the sign of the Lagrangian terms (19) and (23), respectively. MadGraph [5] adopts the Lagrangian with the opposite sign, that is, $(g_s)_{\text{MadGraph}} = -g_s$. This sign difference is absorbed by the conventions (22) and (24).

Table 6 Total cross sections for $gg \to$ gluons [fb]

No. of jets	HEGET	Bases	MadGraph/MadEvent	
2	3.1929 ± 0.0010	3.1928 ± 0.0010	3.1902 ± 0.0076	$\times 10^{11}$
3	2.6201 ± 0.0023	2.6136 ± 0.0036	2.6221 ± 0.0061	$\times 10^{10}$
4	5.813 ± 0.020	5.8140 ± 0.0095	5.776 ± 0.034	$\times 10^{9}$

Table 7 Total cross sections for $u\overline{u} \to$ gluons [fb]

No. of jets	HEGET	Bases	MadGraph/MadEvent	
2	2.8981 ± 0.0007	2.8969 ± 0.0006	2.8991 ± 0.0073	$\times 10^{7}$
3	1.8420 ± 0.0012	1.8388 ± 0.0018	1.8421 ± 0.0077	$\times 10^{6}$
4	4.465 ± 0.022	4.496 ± 0.017	4.399 ± 0.038	$\times 10^{5}$
5	1.566 ± 0.057	1.589 ± 0.018	1.542 ± 0.039	$\times 10^{5}$

Table 8 Total cross sections for $uu \to uu +$ gluons [fb]

No. of jets	HEGET	Bases	MadGraph/MadEvent	
2	2.6715 ± 0.0014	2.6743 ± 0.0011	2.6689 ± 0.0047	$\times 10^{8}$
3	5.897 ± 0.004	5.889 ± 0.010	5.871 ± 0.015	$\times 10^{7}$
4	2.7754 ± 0.0130	2.7500 ± 0.0083	2.748 ± 0.042	$\times 10^{7}$
5	1.513 ± 0.024	1.560 ± 0.013	1.513 ± 0.024	$\times 10^{6}$

space points and for all helicity combinations between those computed on GPU with the HEGET functions and those computed on CPU with the FORTRAN version of HELAS subroutines. For the phase space generation, we use MadGraph/MadEvent [5] and an independent FORTRAN program which calculates total cross section and kinematical distributions with the Monte Carlo integration program BASES [11] as references. Due to the limited support for the double precision computation capabilities on the GPU, the whole computations with HEGET on a GTX280 are done with single precision, while the other programs with HELAS in FORTRAN compute cross sections with double precision.

For the calculation of the n-jet production cross sections we use the same physics parameters as the MadGraph/MadEvent for all programs, and the same final state cuts of (14) for all processes and all programs. The parton distribution functions of CTEQ6L1 [8] and the same factorization and renormalization scales, $Q = p_{\mathrm{T}}^{\mathrm{cut}} = 20$ GeV, are also used.

Results for the computation of the total cross sections are summarized in Tables 6, 7 and 8 for $gg \to$ gluons, $u\overline{u} \to$ gluons and $uu \to uu +$ gluons, respectively. We find the results obtained by the HEGET functions agree with those from the other programs within the statistics of generated number of events.

We note that multi-jet events that satisfy the final state cuts of (14), where all jets are in the central region in $|\eta| < 2.5$ (14a) and their transverse momentum about the beam direction (14b) and among each other (14c) greater than 20 GeV, are dominated by pure gluonic processes in Table 6. The cross sections for $u\overline{u} \to ng$ process in Table 7 are small because of $u\overline{u}$ annihilation. We note that the crossing-related non-annihilation processes, $ug \to u + (n-1)g$, have exactly the same number of diagrams and color bases, hence can be evaluated with essentially the same amount of computation time.

5.2 Comparison of the processing time

As already described in our previous report [1], we prepare two versions of the programs in the same structure for the computation of the total cross sections. One is written in CUDA, a C-based language, and can be executed on the GPU. The other is written in C and can be executed on the CPU. Using a standard C library function we measure the event process time for CPU and for GPU as follows. In order to include the data transfer time between CPU and GPU as an integral part of the event process time by GPU, we measure the interval between the time when CPU starts transferring random numbers to the GPU and the time when the last result from GPU is received by the CPU. This time interval is compared to the interval between the corresponding times in the CPU computation. The fraction of this interval in the total execution time of the CPU program amounts to 98.5% for $u\overline{u} \to 2g$ and 99.96% for $u\overline{u} \to 5g$.

In Fig. 1, the measured process time in μsec for one event of n-jet production processes is shown for the GPU (GTX280) and the CPU (Linux PC with Fedora 8). They are plotted against the number of jets in the final state. Be-

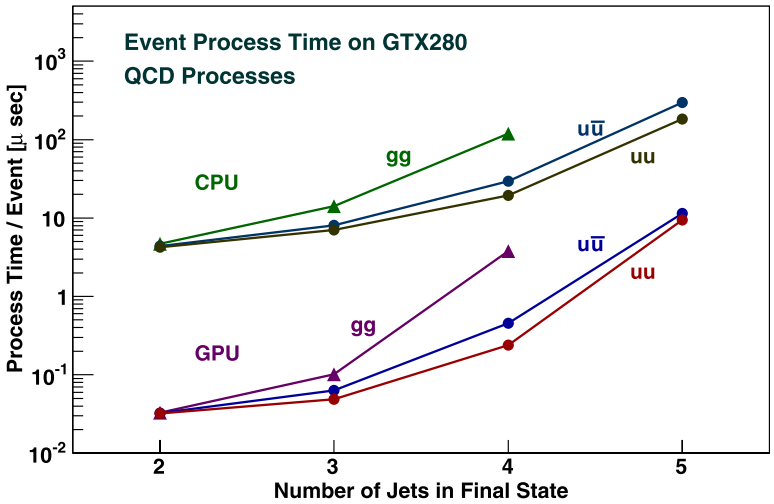

Fig. 1 Processing time for GPU and CPU

cause the process time per event on the GPU depends [1] strongly on the number of allocated registers at the compilation by the CUDA and the size of thread blocks at the execution time, we scan combination of these parameters for the fastest event process time on the GPU.

The upper three lines in Fig. 1 show the event process times on the CPU. They correspond to $gg \to n$-jets denoted as **gg**, $u\overline{u} \to n$-jets as **uū** and $uu \to uu + n$-jets as **uu**, respectively. For processes with small numbers of jets, e.g. $n_{\text{jet}} = 2$, the event process times for different processes are all around 4.5 μs. This is probably because they are dominated by computation steps other than the amplitude calculations, such as computations of the PDF factors, which are common to all physics processes. When the number of jets becomes larger, the event process time for the same number jets in the final states is roughly proportional to the number of diagrams of each process listed in Table 1.

The lower three lines in Fig. 1 show the event process times on a GTX280. They also correspond to $gg \to n$-jets denoted as **gg**, $u\overline{u} \to n$-jets as **uū** and $uu \to uu + n$-jets as **uu**, respectively. As the number of jets becomes larger, the process time on the GPU grows more rapidly than that on the CPU. For the $n_{\text{jet}} = 4$ case, the event process time of $gg \to 4$ gluons is larger than the expected time from the proportionality to the number of diagrams of the other processes, $u\overline{u} \to 4$ gluons and $uu \to uu + 2$ gluons. In other words, the event process time on GPU grows faster than what we expect from the growth of the number of Feynman diagrams.

For instance, the event process times ratio for $gg \to 4g$ and $gg \to 3g$ on the CPU are roughly 120 μs/14 μs ~ 8.6, which roughly agrees with the ratio of the numbers of Feynman diagrams (Table 1), $510/45 \sim 11$. The corresponding ratio on GPU is 3.8 μs/0.1 μs ~ 38, which is significantly larger.

For the same number of jets, we also observe that the event process times on the CPU are roughly proportional to the number of diagrams. For $n_{\text{jet}} = 4$, the ratio of the process times for $gg \to 4g$ to $u\overline{u} \to 4g$ are about 120 μs/29 μs ~ 4.1 on CPU, as compared to the ratio of the number of Feynman diagrams in Table 1, $510/159 \sim 3.2$. The same applies to $n_{\text{jet}} = 5$ between $u\overline{u} \to 5g$ and $uu \to uuggg$, where Feynman diagrams have the ratio $1890/786 \sim 2.4$ from Table 1, and the event process time on the CPU gives 300 μs/180 μs ~ 1.7, also in rough agreement.

On the other hand, the event process times on the GPU for $gg \to 4g$ and $u\overline{u} \to 4g$ have a ratio 3.8 μs/0.45 μs ~ 8.4 which is much larger than the ratio of the diagram numbers; while that for $u\overline{u} \to 5g$ and $uu \to uuggg$ has the ratio of 11 μs/9.5 μs ~ 1.15. Although we do not fully understand the above behavior of the event process time on the GPU, we find that they tends to scale as the product of the number of Feynman diagrams and the number of color bases, while the event process times on the CPU are not sensitive to the latter. This is probably because as the number of color bases grows, more amplitudes, $(J_{\lambda i})_\alpha$ in (8), should be stored and then called to compute the color sum, (10). These observations tell us that the relative weight of the color matrix computation in the GPU computing is very significant even after identifying the independent elements of the color matrix $\mathcal{N}_{\alpha\beta}$ in (9) as listed in Table 4.

5.3 Comparison of performance of GPU and CPU

The ratios of event process times between CPU and GPU are shown in Fig. 2. Three lines correspond to $gg \to n$-jets denoted as **gg**, $u\overline{u} \to n$-jets as **uū** and $uu \to uu + (n-2)$-jets as **uu**, respectively. The performance ratios exceed 100 for the processes with small numbers of jets ($n_{\text{jet}} \leq 3$) in the final state. For $n_{\text{jet}} = 4$ and 5, the performance ratios gradually drop to less than 40. For processes with large numbers

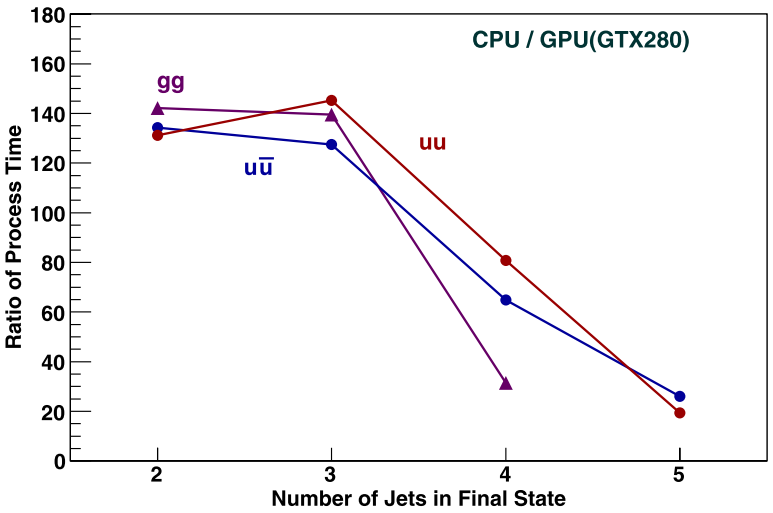

Fig. 2 Ratio of processing time. Time on CPU divided by time on GPU

of color bases, the ratios are smaller. For $gg \to 4$ gluons, which has 120 color bases, the ratio is about 30, and for $uu \to uu + 3$ gluons, which has 240 color bases, the ratio becomes about 20.

5.4 Note on $gg \to 5g$ study

Among five-jet production processes we have not been able to run the program for $gg \to 5g$. This process has 7245 diagrams and 720 color basis vectors. In order to compile the program for the computation of this process, we use the technique developed in the previous study [1]. By dividing the program into about 140 pieces we were able to compile the $gg \to 5g$ program. Compilation takes about 90 min on a Linux PC. The total size of the compiled program exceeds 200 MB, and we were not able to execute this compiled program on a GTX280.

6 Summary

We have shown the results of our attempt to evaluate QCD multi-jet production processes at hadron colliders on a GPU [10], Graphic Processing Unit, following the encouraging results obtained for QED multi-photon production processes in Ref. [1].

Our achievements and findings may be summarized as follows.

- A new set of HEGET functions written in CUDA [2], a C-language platform developed by NVIDIA for general purpose GPU computing, are introduced to compute triple and quartic gluon vertices. The HEGET routines for massless quarks were introduced in Ref. [1], and the routine for photons [1] can be used for gluons. In addition, the HEGET functions for the qqg vertex are the same as those for the $qq\gamma$ vertex introduced in Ref. [1].
- The HELAS amplitude code generated by MadGraph [5] is converted to a CUDA program which calls HEGET functions for the following three type of subprocesses: $gg \to ng$ $(n \leq 5)$, $u\overline{u} \to ng$ $(n \leq 5)$, and $uu \to uu + ng$ $(n \leq 3)$.
- Summation over color degrees of freedom was performed on a GPU by identifying the same valued elements of the color matrix of (9), in order to reduce the memory size.
- All the HEGET programs for up to 5 jets passed the CUDA compiler after division into small pieces. However, we could not execute the program for the process $gg \to 5g$. Accordingly, comparisons of performance between GPU and CPU are done for the multi-jet production processes up to 5 jets, excluding the purely gluonic subprocess.
- Event process times of the GPU program on GTX280 are more than 100 times faster than the CPU program for all the processes up to 3-jets, while the gain is reduced to 60 for 4-jets with one or two quark lines, and to 30 for the purely gluonic process. It further goes down to 30 and 20 for 5-jet production processes with one and two quark lines, respectively.
- We find that one cause of the rapid loss of GPU gain over CPU as the number of jets increases is the growth in the number of color bases. GPU programs slow down for processes with larger numbers of color basis vectors, while the performance of the CPU programs is not affected much.
- All computations on the GPU were performed with single precision accuracy. A factor of 2.5 to 4 slower performance is found for double precision computation on the GPU.[2]

[2]Please refer to Ref. [1] for the comparison of performances between single precision and double precision computations.

Acknowledgements We thank Johan Alwall, Qiang Li and Fabio Maltoni for stimulating discussions. This work is supported by the Grant-in-Aid for Scientific Research from the Japan Society for the Promotion of Science (No. 20340064) and the National Science Foundation (No. 0757889).

Appendix A: Additional HEGET functions

In the appendix, we list the HEGET functions introduced in this report. They are for the ggg and $gggg$ vertices which do not have counterparts in QED. Together with the HEGET functions listed in Ref. [1], the quark and gluon (photon) wave functions and the $qqg(qq\gamma)$ vertices, all the QCD amplitudes can be computed on GPU.

A.1 Functions for the VVV vertex

For the ggg vertex

$$\mathcal{L}_{ggg} = g_s f^{abc}\big(\partial^\mu A^{a\nu}(x)\big)A^b_\mu(x)A^c_\nu(x) \tag{29}$$

we introduce two HEGET functions, `vvvxxx` and `jvvxx0`. They correspond to HELAS subroutines, VVVXXX, and JVVXXX, respectively, for massless particles; see Table 5.

A.1.1 vvvxxx

The HEGET function `vvvxxx` (List 1 in Appendix) computes the amplitude of the VVV vertex from vector boson wave functions, whether they are on-shell or off-shell. The function has the arguments:

```
vvvxxx(cmplx* ga, cmplx* gb, cmplx* gc,
                    float g, cmplx vertex)
```
(30)

where the inputs and the outputs are:

INPUTS:

`cmplx ga[6]` wavefunction of gluon with color index, a

`cmplx gb[6]` wavefunction of gluon with color index, b

`cmplx gc[6]` wavefunction of gluon with color index, c

`float g` coupling constant of VVV vertex

OUTPUTS:

`cmplx vertex` amplitude of the VVV vertex (31)

The coupling constant is

$$\mathtt{g} = g_s \tag{32}$$

in the HEGET function (30), following the convention of MadGraph [5]. In order to reproduce the amplitudes associated with the ggg vertex Lagrangian of (29), the color factor associated with the ggg vertex is if^{abc}. More explicitly, the vertex amplitude for (29) is

$$if^{abc}(\mathbf{vertex}) \tag{33}$$

by using the output (**vertex**) in (30); see the footnote in Sect. 3.4. Also note the HELAS convention [3, 4] of using the flowing-OUT momenta and quantum numbers for all bosons.

List 1 vvvxxx.cu

```
#include "cmplx.h"

__device__
void vvvxxx(cmplx* ga, cmplx* gb,
    cmplx* gc,
                  float g, cmplx&
                      vertex)
{
   cmplx v12 = ga[0]*gb[0]
     - ga[1]*gb[1] - ga[2]*gb[2] - ga[3]*
         gb[3];
   cmplx v23 = gb[0]*gc[0]
     - gb[1]*gc[1] - gb[2]*gc[2] - gb[3]*
         gc[3];
   cmplx v31 = gc[0]*ga[0]
     - gc[1]*ga[1] - gc[2]*ga[2] - gc[3]*
         ga[3];

   float pga[4];
   float pgb[4];
   float pgc[4];

   pga[0] = ga[4].re;
   pga[1] = ga[5].re;
   pga[2] = ga[5].im;
   pga[3] = ga[4].im;

   pgb[0] = gb[4].re;
   pgb[1] = gb[5].re;
   pgb[2] = gb[5].im;
   pgb[3] = gb[4].im;

   pgc[0] = gc[4].re;
   pgc[1] = gc[5].re;
   pgc[2] = gc[5].im;
   pgc[3] = gc[4].im;

   cmplx p12 = pga[0]*gb[0]
    - pga[1]*gb[1] - pga[2]*gb[2] - pga
        [3]*gb[3];
   cmplx p13 = pga[0]*gc[0]
    - pga[1]*gc[1] - pga[2]*gc[2] - pga
        [3]*gc[3];
   cmplx p21 = pgb[0]*ga[0]
    - pgb[1]*ga[1] - pgb[2]*ga[2] - pgb
        [3]*ga[3];
   cmplx p23 = pgb[0]*gc[0]
    - pgb[1]*gc[1] - pgb[2]*gc[2] - pgb
        [3]*gc[3];
   cmplx p31 = pgc[0]*ga[0]
    - pgc[1]*ga[1] - pgc[2]*ga[2] - pgc
        [3]*ga[3];
   cmplx p32 = pgc[0]*gb[0]
```

```
    - pgc[1]*gb[1] - pgc[2]*gb[2] - pgc
        [3]*gb[3];

  vertex = -(v12*(p13-p23)
           +v23*(p21-p31)
           +v31*(p32-p12) )*g;

  return;
}
```

A.1.2 jvvxx0

The HEGET function `jvvxx0` (List 2) computes the off-shell vector wavefunction from the three-point gauge boson coupling in (29). The vector propagator is given in the Feynman gauge for a massless vector bosons like gluons. It has the arguments:

```
jvvxx0(cmplx* ga, cmplx* gb, float g,
                              cmplx* jvv)
```

(34)

where the inputs and the outputs are:

INPUTS:

`cmplx ga[6]`	wavefunction of gluon with color index, a
`cmplx gb[6]`	wavefunction of gluon with color index, b
`float g`	coupling constant of the VVV vertex

OUTPUTS:

`cmplx jvv[6]`	vector current $j^{\mu}(\mathtt{gc}:\mathtt{ga},\mathtt{gb})$ which has a color index, c

(35)

As in (24) the color amplitude for the off-shell current is

$$if^{abc}(\mathbf{jvv}). \tag{36}$$

List 2 jvvxx0.cu

```
#include "cmplx.h"

__device__
void jvvxx0(cmplx* ga, cmplx* gb,
    float g,
        cmplx* jvv)
{
  jvv[4] = ga[4] + gb[4];
  jvv[5] = ga[5] + gb[5];

  float p1[4];
  float p2[4];
  float q[4];

  p1[0] = (ga[4].re);
  p1[1] = (ga[5].re);
  p1[2] = (ga[5].im);
  p1[3] = (ga[4].im);

  p2[0] = (gb[4].re);
  p2[1] = (gb[5].re);
  p2[2] = (gb[5].im);
  p2[3] = (gb[4].im);

  q[0] = -(jvv[4].re);
  q[1] = -(jvv[5].re);
  q[2] = -(jvv[5].im);
  q[3] = -(jvv[4].im);

  float s = q[0]*q[0]
     -q[1]*q[1] -q[2]*q[2] -q[3]*q[3];

  cmplx gab = ga[0]*gb[0]
    -ga[1]*gb[1] -ga[2]*gb[2] -ga[3]*gb
        [3];


  cmplx sga =
     (p2[0]-q[0])*ga[0] - (p2[1]-q[1])*ga
         [1]
    - (p2[2]-q[2])*ga[2] - (p2[3]-q[3])*ga
        [3];

  cmplx sgb =
    - (p1[0]-q[0])*gb[0] + (p1[1]-q[1])*gb
        [1]
    + (p1[2]-q[2])*gb[2] + (p1[3]-q[3])*gb
        [3];

  float gs = -g*(1.0f/s);

  jvv[0] = gs*((p1[0]-p2[0])*gab
           + sga*gb[0] + sgb*ga[0]);
  jvv[1] = gs*((p1[1]-p2[1])*gab
           + sga*gb[1] + sgb*ga[1]);
  jvv[2] = gs*((p1[2]-p2[2])*gab
           + sga*gb[2] + sgb*ga[2]);
  jvv[3] = gs*((p1[3]-p2[3])*gab
           + sga*gb[3] + sgb*ga[3]);

  return;
}
```

A.2 Functions for the VVVV vertex

For the $gggg$ vertex

$$\mathcal{L}_{gggg} = -\frac{g_s^2}{4} f^{abe} f^{cde} A^{a\mu}(x) A^{bv}(x) A^{c}_{\mu}(x) A^{d}_{v}(x) \tag{37}$$

we introduce two HEGET functions, `ggggxx` and `jgggx0`, listed in Table 5. They correspond to HELAS subroutines, GGGGXX and JGGGXX, respectively, for massless particles.

A.2.1 ggggxx

The HEGET function `ggggxx` (List 3) computes the amplitude from 4 gluon wave functions `ga`, `gb`, `gc` and `gd`, each with the color index a, b, c and d, respectively, when the associated color factor is $f^{abe} f^{cde}$, whether the gluons are on-shell or off-shell. The function has the arguments:

```
ggggxx(cmplx* ga, cmplx* gb, cmplx* gc,
    cmplx* gd, float gg, cmplx vertex)
```

(38)

where the inputs and the outputs are:

INPUTS:

`cmplx ga[6]`	wavefunction of gluon with color index, a
`cmplx gb[6]`	wavefunction of gluon with color index, b
`cmplx gc[6]`	wavefunction of gluon with color index, c
`cmplx gd[6]`	wavefunction of gluon with color index, d
`float gg`	coupling constant of VVVV vertex

OUTPUTS:

`cmplx vertex`	amplitude of the VVVV vertex with the color factor $f^{abe}f^{cde}$.

(39)

The coupling constant `gg` for the $gggg$ vertex is

$$\texttt{gg} = g_s^2. \tag{40}$$

In order to obtain the complete amplitude, the function must be called three times (once for each color structure) with the following permutations:

$$\texttt{ggggxx(ga, gb, gc, gd, gg, v1)} \tag{41a}$$

$$\texttt{ggggxx(ga, gc, gd, gb, gg, v2)} \tag{41b}$$

$$\texttt{ggggxx(ga, gd, gb, gc, gg, v3)} \tag{41c}$$

The color amplitudes are then expressed as

$$f^{abe}f^{cde}(\texttt{v1}) + f^{ace}f^{dbe}(\texttt{v2}) + f^{ade}f^{bce}(\texttt{v3}). \tag{42}$$

List 3 ggggxx.cu

```
#include "cmplx.h"

__device__
void ggggxx(cmplx* ga, cmplx* gb,
    cmplx* gc,
        cmplx* gd, float gg, cmplx&
            vertex)
{
   cmplx gad = ga[0]*gd[0]
     -ga[1]*gd[1]-ga[2]*gd[2]-ga[3]*gd[3];

   cmplx gbc = gb[0]*gc[0]
     -gb[1]*gc[1]-gb[2]*gc[2]-gb[3]*gc[3];

   cmplx gac = ga[0]*gc[0]
     -ga[1]*gc[1]-ga[2]*gc[2]-ga[3]*gc[3];

   cmplx gbd = gb[0]*gd[0]
     -gb[1]*gd[1]-gb[2]*gd[2]-gb[3]*gd[3];

   vertex = gg*(gad*gbc-gac*gbd);

   return;
}
```

A.2.2 jgggx0

The HEGET function `jgggx0` (List 4) computes an off-shell gluon current from the four-point gluon coupling, including the gluon propagator in the Feynman gauge. It has the arguments:

```
jgggx0(cmplx* ga, cmplx* gb, cmplx* gc,
                    float gg, cmplx* jggg)
```

(43)

where the inputs and the outputs are:

INPUTS:

`cmplx ga[6]`	wavefunction of gluon with color index, a
`cmplx gb[6]`	wavefunction of gluon with color index, b
`cmplx gc[6]`	wavefunction of gluon with color index, c
`float gg`	coupling constants of the VVVV vertex.

OUTPUTS:

`cmplx jggg[6]`	vector current $\boldsymbol{j}^{\mu}(d:ab,c)$ which has the color index d associated with the color factor $f^{abe}f^{cde}$.

(44)

The function (43) should be called three times as

$$\texttt{jgggx0(ga,gb,gc,gg,j1)} \tag{45a}$$

$$\texttt{jgggx0(gc,ga,gb,gg,j2)} \tag{45b}$$

$$\texttt{jgggx0(gb,gc,ga,gg,j3)} \tag{45c}$$

as in (41), and the off-shell gluon current with the color index d is obtained as

$$f^{abe}f^{cde}(\texttt{j1}) + f^{ace}f^{dbe}(\texttt{j2}) + f^{ade}f^{bce}(\texttt{j3}). \tag{46}$$

List 4 jgggx0.cu

```
#include "cmplx.h"

__device__
void jgggx0(cmplx* ga, cmplx* gb,
    cmplx* gc,
                float gg, cmplx* jggg)
{
   jggg[4] = ga[4]+gb[4]+gc[4];
   jggg[5] = ga[5]+gb[5]+gc[5];

   float q[4];
   q[0] = -jggg[4].re;
   q[1] = -jggg[5].re;
   q[2] = -jggg[5].im;
   q[3] = -jggg[4].im;

   float fact = gg*(1.0f/(q[0]*q[0]
```

```
        - q[1]*q[1] - q[2]*q[2] - q[3]*q[3])
            );

    cmplx gcb = gc[0]*gb[0]
     - gc[1]*gb[1] - gc[2]*gb[2] - gc[3]*gb
         [3];
    cmplx gac = ga[0]*gc[0]
     - ga[1]*gc[1] - ga[2]*gc[2] - ga[3]*gc
         [3];

    jggg[0] = fact*( ga[0]*gcb - gb[0]*gac
          );
    jggg[1] = fact*( ga[1]*gcb - gb[1]*gac
          );
    jggg[2] = fact*( ga[2]*gcb - gb[2]*gac
          );
    jggg[3] = fact*( ga[3]*gcb - gb[3]*gac
          );

    return;
}
```

References

1. K. Hagiwara, J. Kanzaki, N. Okamura, D. Rainwater, T. Stelzer, Eur. Phys. J. C **66**, 477 (2010)
2. http://www.nvidia.com/object/cuda_home.html
3. K. Hagiwara, H. Murayama, I. Watanabe, Nucl. Phys. B **367**, 257 (1991)
4. H. Murayama, I. Watanabe, K. Hagiwara, KEK-Report 91-11 (1992)
5. T. Stelzer, W.F. Long, Comput. Phys. Commun. **81**, 357 (1994)
6. M.L. Mangano, S.J. Parke, Phys. Rep. **200**, 301 (1991)
7. S. Catani, Y.L. Dokshitzer, M.H. Seymour, B.R. Webber, Nucl. Phys. B **406**, 187 (1993)
8. H.L. Lai et al. (CTEQ Collaboration), Eur. Phys. J. C **12**, 375 (2000)
9. C. Amsler et al. (Particle Data Group), Phys. Lett. B **667**, 1 (2008) and 2009 partial update for the 2010 edition
10. http://www.nvidia.com/page/home.html
11. S. Kawabata, Comput. Phys. Commun. **41**, 127 (1986)