MadLoop5

Going Beyonder

Valentin Hirschi
EPFL

4th September 2012

Presentation
HP2^4@MPI München

Friday, September 7, 2012

Outline

- NLO challenges and **aMC@NLO** philosophy
- Implementation details
- Speed and stability benchmark study
- Future plans and closing words

MadGraph@NLO
Overview

Objectives for Madgraph5 at NLO

- Automation and Flexibility

  Minimize hand work while maximizing applicability.
  Also automation provides reliability by avoiding bugs.

- Unique framework and user-friendly

  It only takes to know how to efficiently use one single program to do all NLO phenomenology.
  User-guidance and on-the-fly checks insure reliable results.

- Stable and fast enough for relevant processes

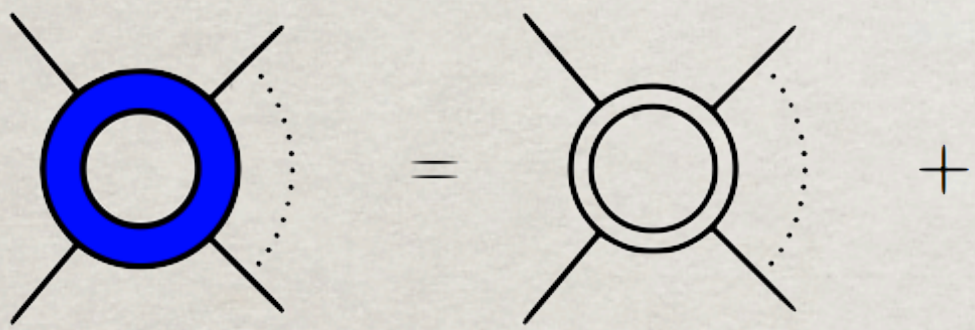
  No huge cluster needed.
  LesHouches wish list(s) covered.

NLO Basics

Fixed-order NLO contributions have two parts

NLO Basics

Fixed-order NLO contributions have two parts



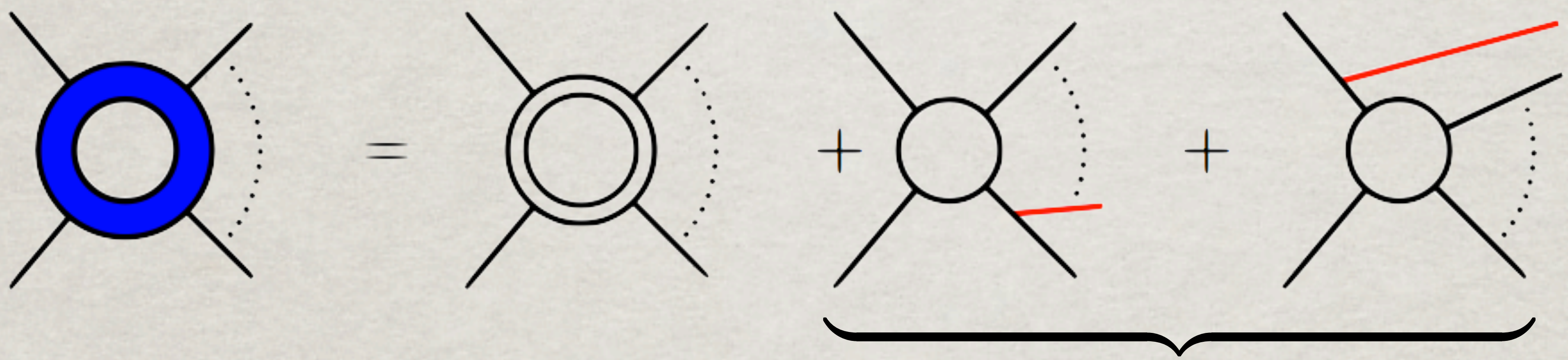
$$\sigma^{\text{NLO}} = \int_m d^{(d)}\sigma^V +$$

Virtual part

- Used to be bottleneck of NLO computations
- Algorithms for automation known in principle but needs to be efficiently implemented
- MadLoop5 in MG5 takes care of this piece

NLO Basics

Fixed-order NLO contributions have two parts



$$\sigma^{\mathrm{NLO}} = \int_m d^{(d)}\sigma^V + \int_{m+1} d^{(d)}\sigma^R +$$

Virtual part

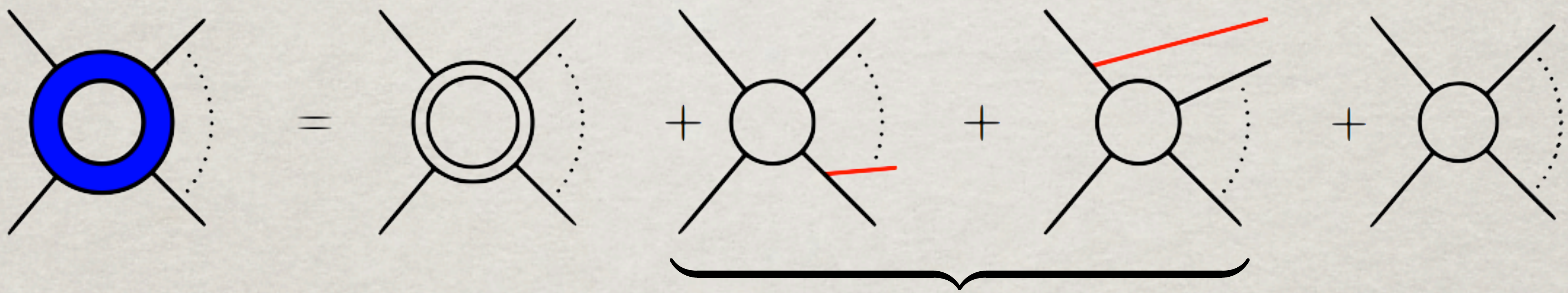
- Used to be bottleneck of NLO computations
- Algorithms for automation known in principle but needs to be efficiently implemented
- MadLoop5 in MG5 takes care of this piece

Real emission part

- Automated for different methods
- Challenge is the systematic extraction of singularities
- MadFKS5 in MG5 takes care of this piece

NLO Basics

Fixed-order NLO contributions have two parts



$$\sigma^{\text{NLO}} = \int_m d^{(d)}\sigma^V + \int_{m+1} d^{(d)}\sigma^R + \int_m d^{(4)}\sigma^B$$

Virtual part

- Used to be bottleneck of NLO computations
- Algorithms for automation known in principle but needs to be efficiently implemented
- MadLoop5 in MG5 takes care of this piece

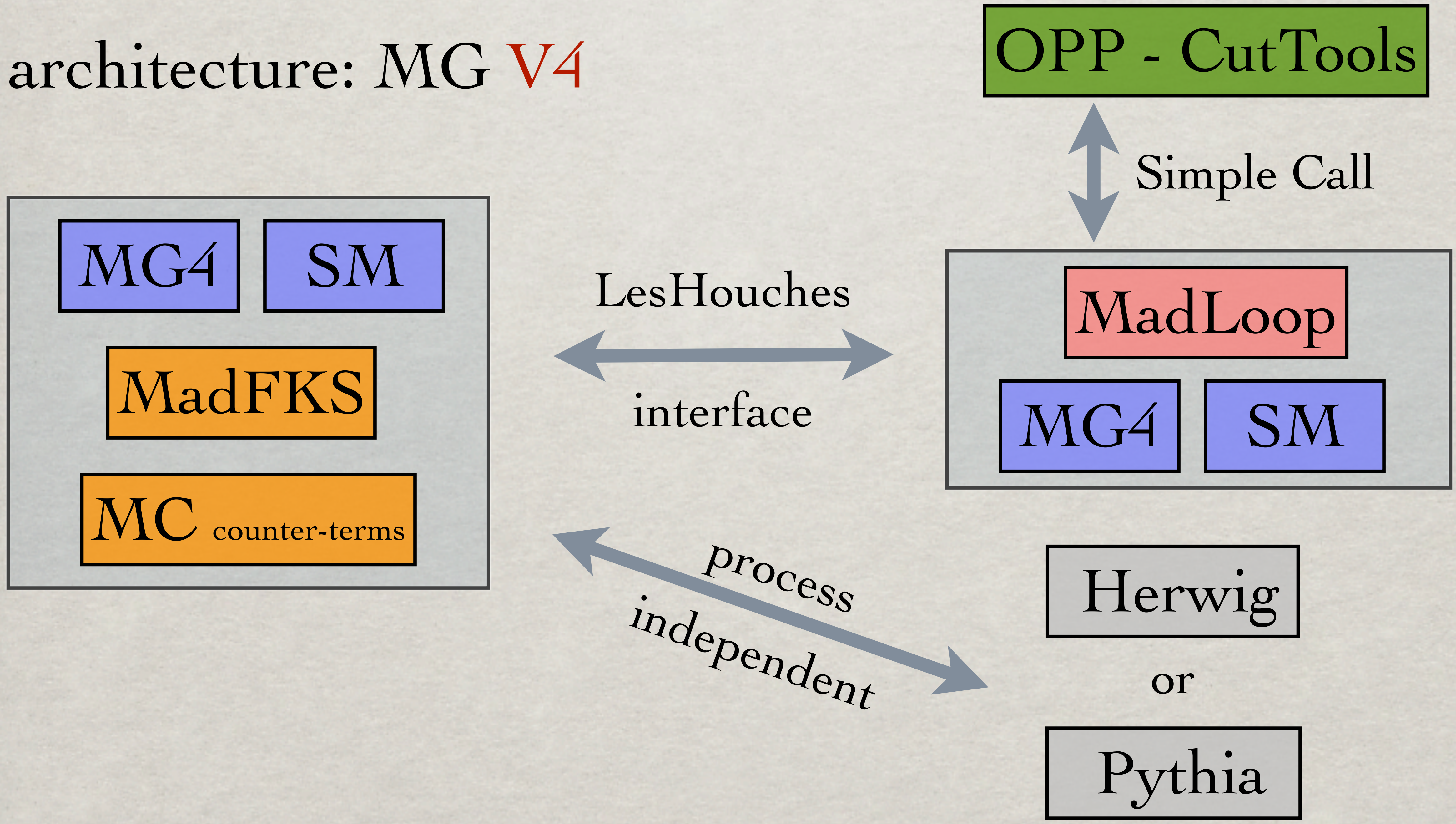
Real emission part

- Automated for different methods
- Challenge is the systematic extraction of singularities
- MadFKS5 in MG5 takes care of this piece

aMC@NLO

Towards full automation

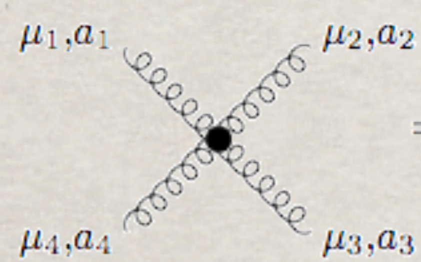
architecture: MG V4



MadLoop in MG4

What it could NOT do

✓ No four-gluon vertex at born level :



$$= -\frac{ig^4 N_{col}}{96\pi^2} \sum_{P(234)} \Bigg\{ \Bigg[\frac{\delta_{a_1a_2}\delta_{a_3a_4} + \delta_{a_1a_3}\delta_{a_4a_2} + \delta_{a_1a_4}\delta_{a_2a_3}}{N_{col}}$$

$$+ 4\,Tr(t^{a_1}t^{a_3}t^{a_2}t^{a_4} + t^{a_1}t^{a_4}t^{a_2}t^{a_3})\,(3 + \lambda_{HV})$$

$$- Tr(\{t^{a_1}t^{a_2}\}\{t^{a_3}t^{a_4}\})\,(5 + 2\lambda_{HV})\Bigg]\, g_{\mu_1\mu_2}g_{\mu_3\mu_4}$$

$$+12\frac{N_f}{N_{col}} Tr(t^{a_1}t^{a_2}t^{a_3}t^{a_4}) \left(\frac{5}{3} g_{\mu_1\mu_3}g_{\mu_2\mu_4} - g_{\mu_1\mu_2}g_{\mu_3\mu_4} - g_{\mu_2\mu_3}g_{\mu_1\mu_4}\right)\Bigg\}$$

✓ All born contribution must factorize the same power of all coupling orders.

✓ No finite-width effects of unstable massive particles also appearing in the loop.

✓ / ✕ Handle BSM model or/and EW corrections.

What ML4 could do

- Running time: Two weeks on a 150+ node cluster
- Proof of efficient EPS handling with $Zt\bar{t}$
- Successful cross-check against known results
- Large K-factors sometimes
- No cuts on b, robust numerics with small P_T

	Process	μ	n_{lf}	Cross section (pb) LO	Cross section (pb) NLO
a.1	$pp \to t\bar{t}$	m_{top}	5	123.76 ± 0.05	162.08 ± 0.12
a.2	$pp \to tj$	m_{top}	5	34.78 ± 0.03	41.03 ± 0.07
a.3	$pp \to tjj$	m_{top}	5	11.851 ± 0.006	13.71 ± 0.02
a.4	$pp \to t\bar{b}j$	$m_{top}/4$	4	25.62 ± 0.01	30.96 ± 0.06
a.5	$pp \to t\bar{b}jj$	$m_{top}/4$	4	8.195 ± 0.002	8.91 ± 0.01
b.1	$pp \to (W^+ \to)e^+\nu_e$	m_W	5	5072.5 ± 2.9	6146.2 ± 9.8
b.2	$pp \to (W^+ \to)e^+\nu_e\,j$	m_W	5	828.4 ± 0.8	1065.3 ± 1.8
b.3	$pp \to (W^+ \to)e^+\nu_e\,jj$	m_W	5	298.8 ± 0.4	300.3 ± 0.6
b.4	$pp \to (\gamma^*/Z \to)e^+e^-$	m_Z	5	1007.0 ± 0.1	1170.0 ± 2.4
b.5	$pp \to (\gamma^*/Z \to)e^+e^-\,j$	m_Z	5	156.11 ± 0.03	203.0 ± 0.2
b.6	$pp \to (\gamma^*/Z \to)e^+e^-\,jj$	m_Z	5	54.24 ± 0.02	56.69 ± 0.07
c.1	$pp \to (W^+ \to)e^+\nu_e b\bar{b}$	$m_W + 2m_b$	4	11.557 ± 0.005	22.95 ± 0.07
c.2	$pp \to (W^+ \to)e^+\nu_e t\bar{t}$	$m_W + 2m_{top}$	5	0.009415 ± 0.000003	0.01159 ± 0.00001
c.3	$pp \to (\gamma^*/Z \to)e^+e^- b\bar{b}$	$m_Z + 2m_b$	4	9.459 ± 0.004	15.31 ± 0.03
c.4	$pp \to (\gamma^*/Z \to)e^+e^- t\bar{t}$	$m_Z + 2m_{top}$	5	0.0035131 ± 0.0000004	0.004876 ± 0.000002
c.5	$pp \to \gamma t\bar{t}$	$2m_{top}$	5	0.2906 ± 0.0001	0.4169 ± 0.0003
d.1	$pp \to W^+W^-$	$2m_W$	4	29.976 ± 0.004	43.92 ± 0.03
d.2	$pp \to W^+W^-\,j$	$2m_W$	4	11.613 ± 0.002	15.174 ± 0.008
d.3	$pp \to W^+W^+\,jj$	$2m_W$	4	0.07048 ± 0.00004	0.1377 ± 0.0005
e.1	$pp \to HW^+$	$m_W + m_H$	5	0.3428 ± 0.0003	0.4455 ± 0.0003
e.2	$pp \to HW^+\,j$	$m_W + m_H$	5	0.1223 ± 0.0001	0.1501 ± 0.0002
e.3	$pp \to HZ$	$m_Z + m_H$	5	0.2781 ± 0.0001	0.3659 ± 0.0002
e.4	$pp \to HZ\,j$	$m_Z + m_H$	5	0.0988 ± 0.0001	0.1237 ± 0.0001
e.5	$pp \to Ht\bar{t}$	$m_{top} + m_H$	5	0.08896 ± 0.00001	0.09869 ± 0.00003
e.6	$pp \to Hb\bar{b}$	$m_b + m_H$	4	0.16510 ± 0.00009	0.2099 ± 0.0006
e.7	$pp \to Hjj$	m_H	5	1.104 ± 0.002	1.036 ± 0.002

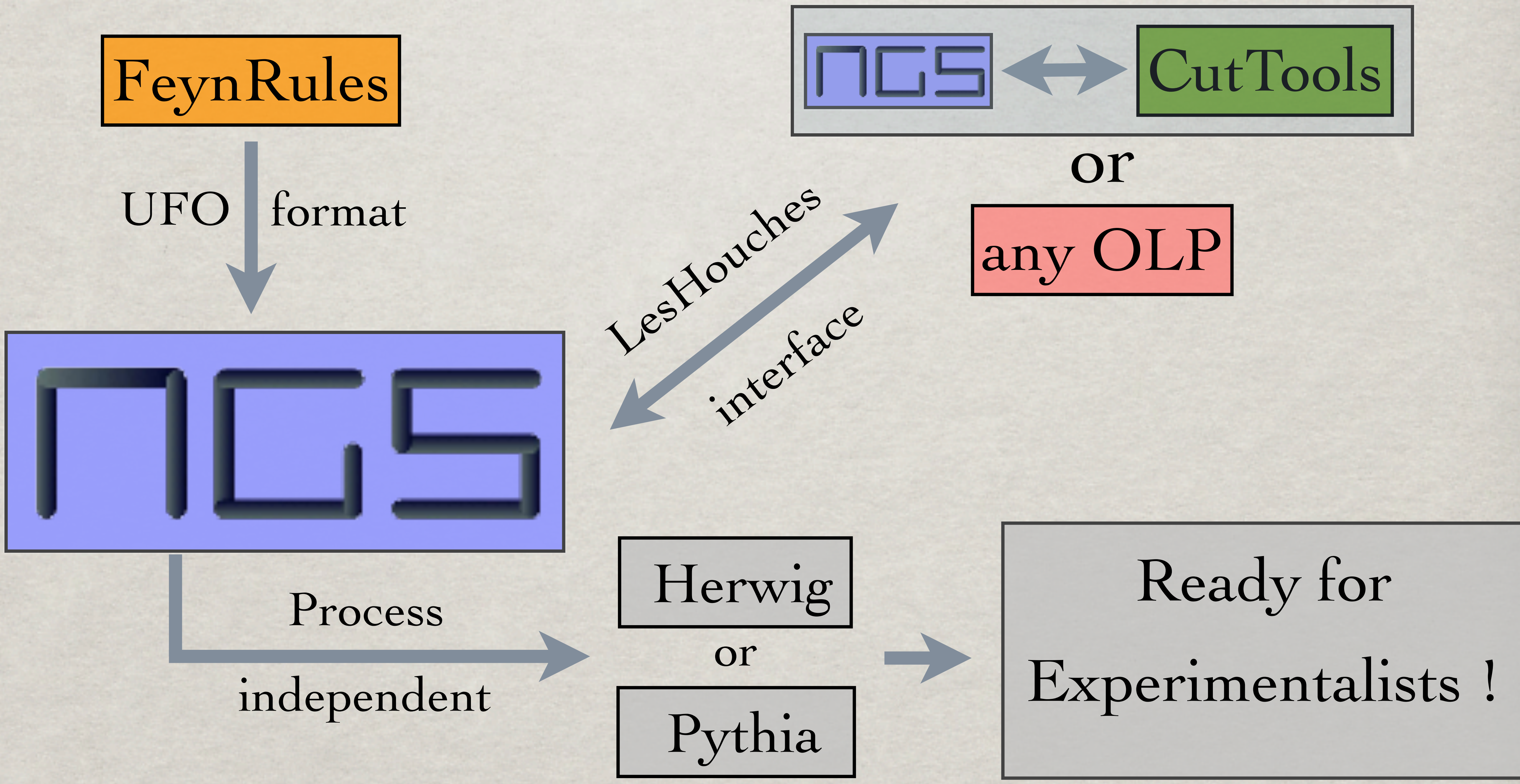
MadGraph 5 Specs

- High-level language: Python
  - Complex data-structures allow for very general objects while keeping speed where needed.
  - Involved algorithms => Performance increase
- Built-in testing suite => Reliability
- User-interface and automatic doc. => User friendly
- Flexible and Modular => Developer friendly
  All-in-one distribution

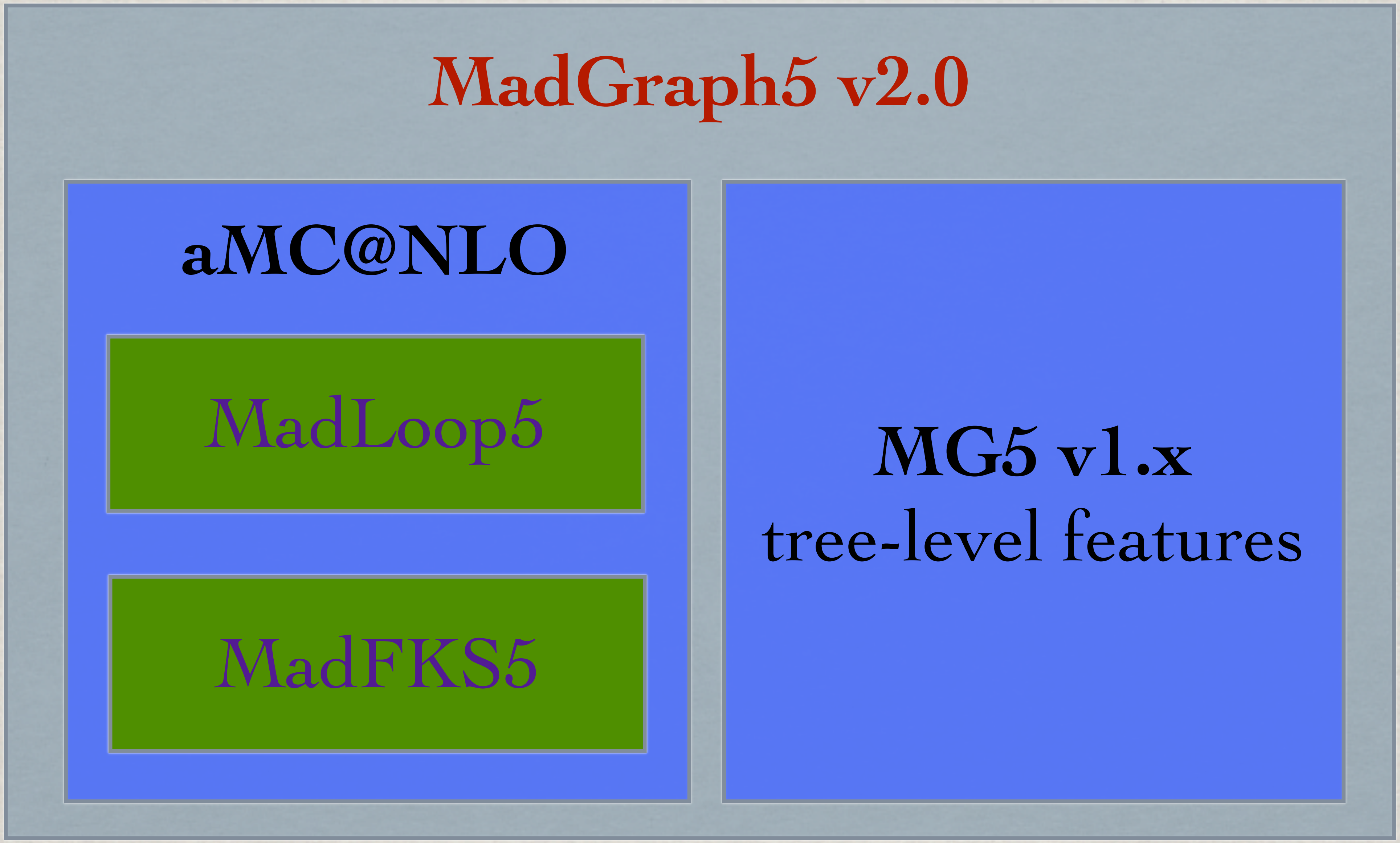
AMC@NLO

FULL AUTOMATION...

... in MadGraph5 v2.0!



Nomenclature



But this separation is now transparent to the users!

Implementation

MadLoop5 in MG5 v2.0

friend of users

- Process generation
  - `import model <model_name>-<restrictions>`
  - `generate <process> <amp_orders_and_option> [<mode>=<pert_orders>] <squared_orders>`
  - `output <format> <folder_name>`
  - `launch`
- Examples, starting from a blank MG5 interface.
  - Very simple one:

```
[ 1.54s ] generate g g > t t~ [virt=QCD]
[ 1.18s ] output
[ 44 ms*] launch
```

  - With options specified:

```
[ 0.01s ] import model loop_sm-no_bmass
[ 0.01s ] set complex_mass_scheme
[ 22.8s ] generate g g > W+ W- b b~ / z h a QED=2 [virt=QCD] QCD=6 WEIGHTED=14
[ 14.0s ] output standalone MyProc
[ 17.1s*] launch
```

* time per phase-space point, summed over helicities and colors.

Cut-Loop diagrams

With a specific example

Consider $e^+e^- \to \gamma \to u\bar{u}$:

- Loop particles are denoted with a star. When MG is asked for $e^+e^- \to u^*\bar{u}^*u\bar{u}$ it gives back eight diagrams. Two of them are:

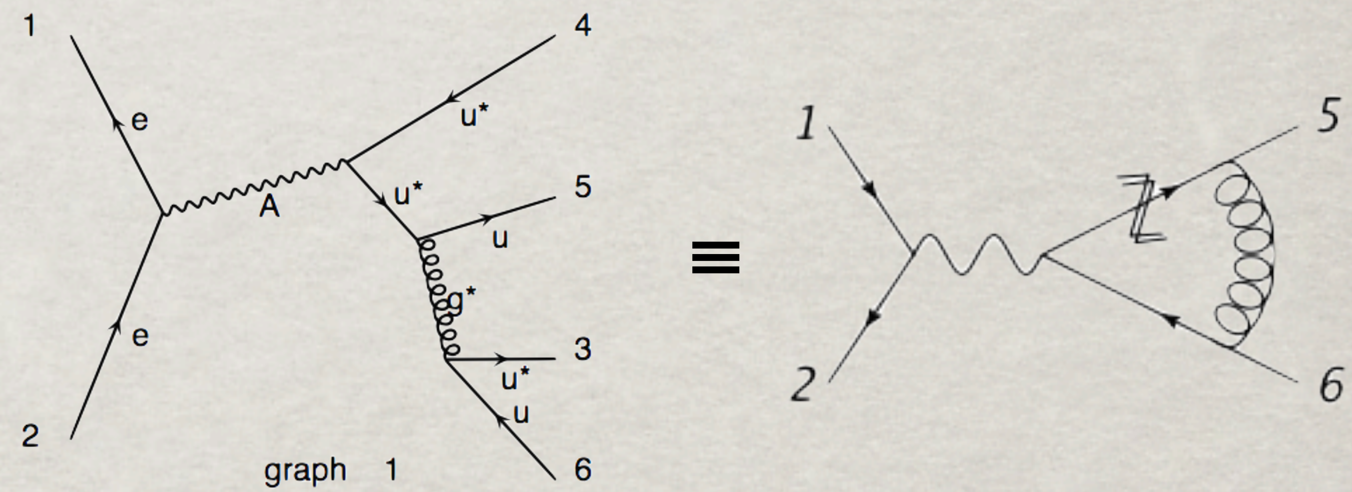
Cut-Loop diagrams

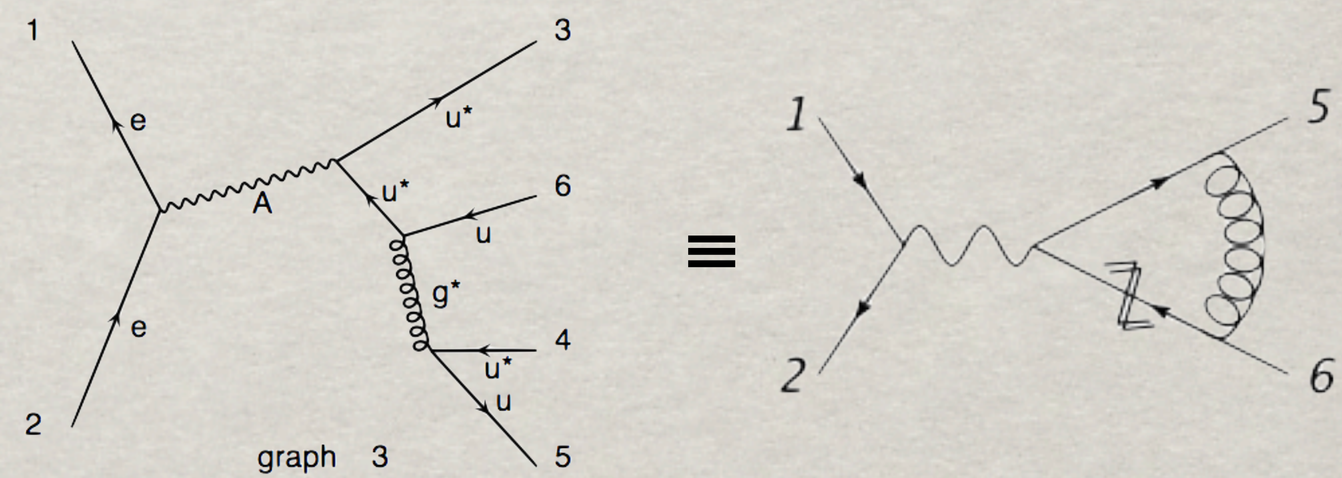
With a specific example

Consider $e^+e^- \rightarrow \gamma \rightarrow u\bar{u}$:

- Loop particles are denoted with a star. When MG is asked for $e^+e^- \rightarrow u^*\bar{u}^*u\bar{u}$ it gives back eight diagrams. Two of them are:

- Selection is performed to keep only one cut-diagram per loop contributing in the process





Cut-Loop diagrams

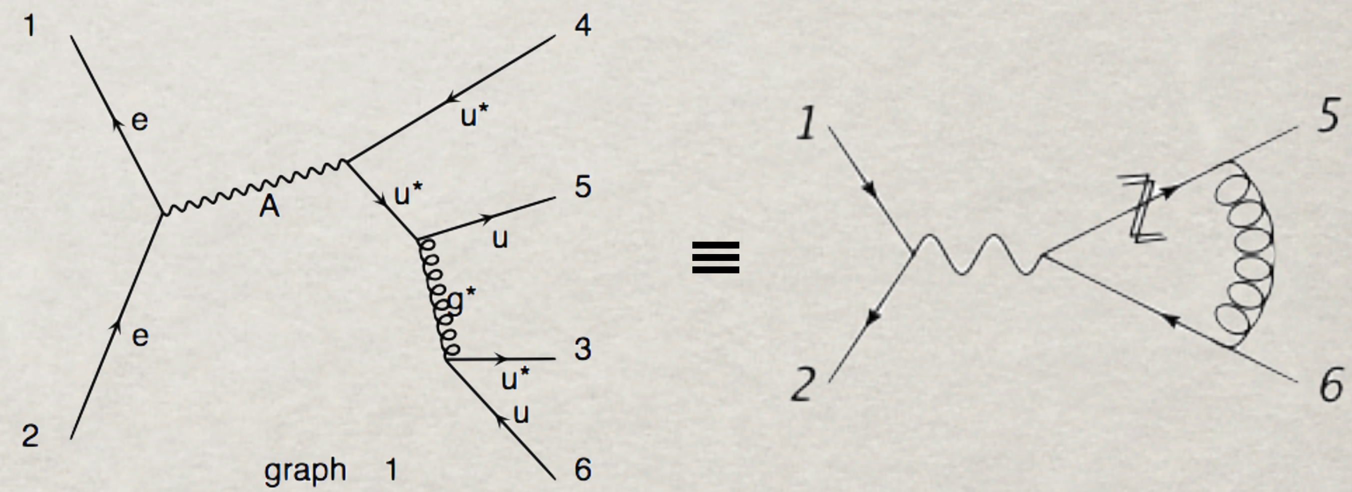
With a specific example

Consider $e^+e^- \to \gamma \to u\bar{u}$:

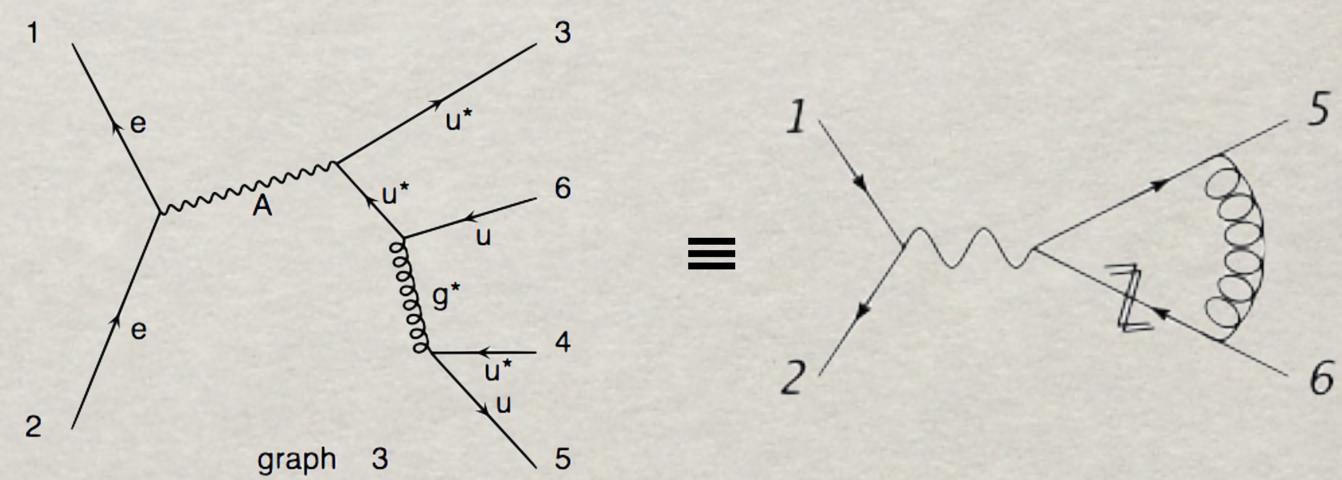
- Loop particles are denoted with a star. When MG is asked for $e^+e^- \to u^*\bar{u}^*u\bar{u}$ it gives back eight diagrams. Two of them are:

- Selection is performed to keep only one cut-diagram per loop contributing in the process

- Tags are associated to each cut-diagram. Those whose tags are **mirror and/or cyclic permutations** of tags of diagram already in the loop-basis are taken out.



Diag $_1 = [u^*(6)g^*(5)u^*(A)]$



Diag $_3 = [u^*(A)u^*(6)g^*(5)]$

Cut-Loop diagrams

With a specific example

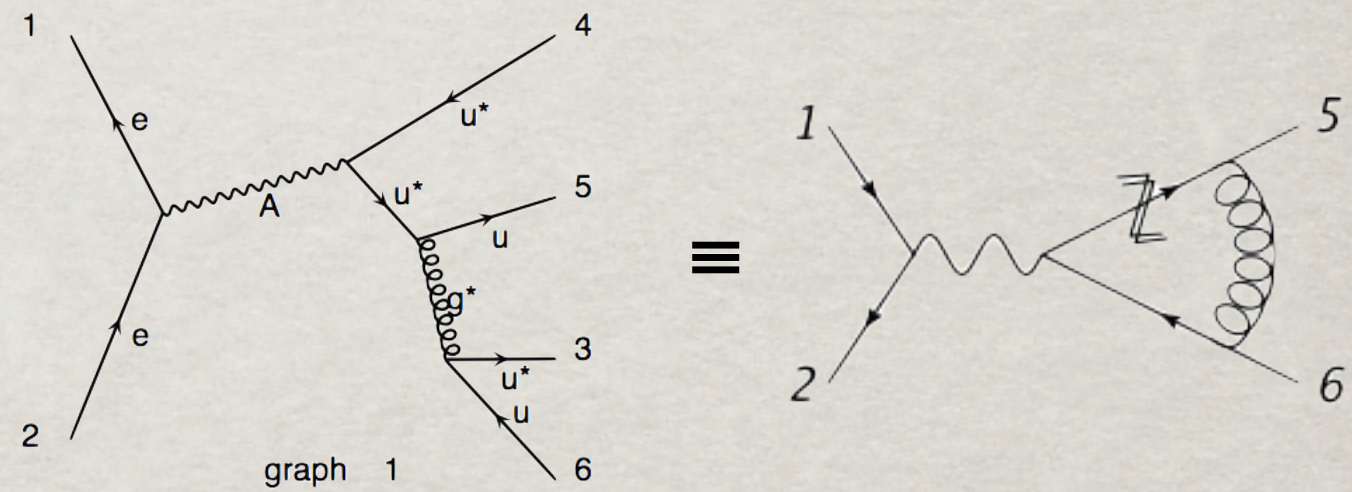
Consider $e^+e^- \to \gamma \to u\bar{u}$:

- Loop particles are denoted with a star. When MG is asked for $e^+e^- \to u^*\bar{u}^*u\bar{u}$ it gives back eight diagrams. Two of them are:

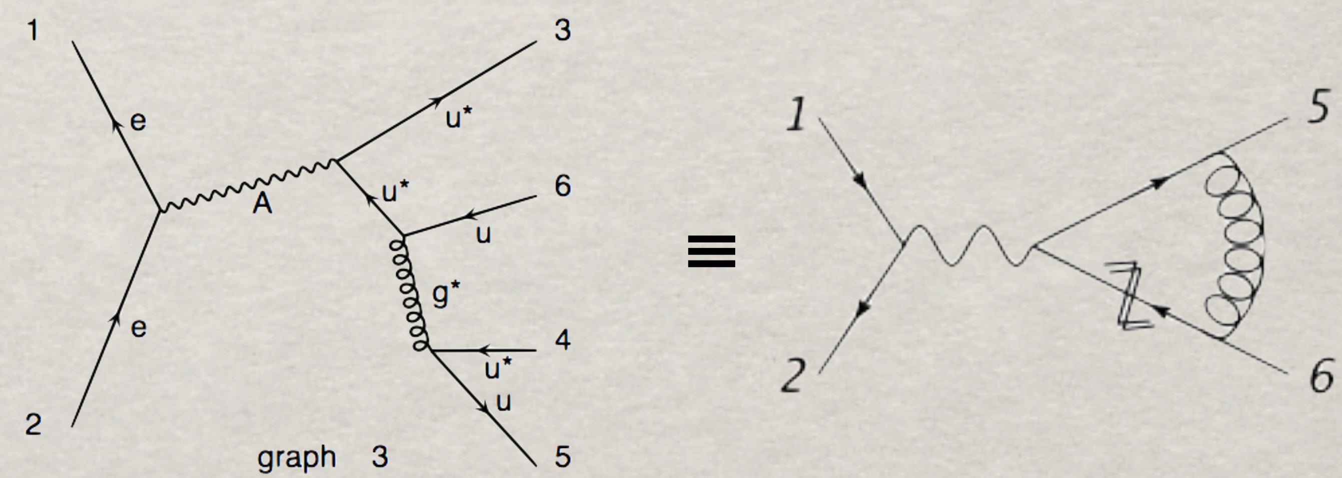
- Selection is performed to keep only one cut-diagram per loop contributing in the process

- Tags are associated to each cut-diagram. Those whose tags are **mirror and/or cyclic permutations** of tags of diagram already in the loop-basis are taken out.

- Additional custom filter to eliminate tadpoles and bubbles attached to external legs.



Diag $_1 = [u^*(6)g^*(5)u^*(A)]$

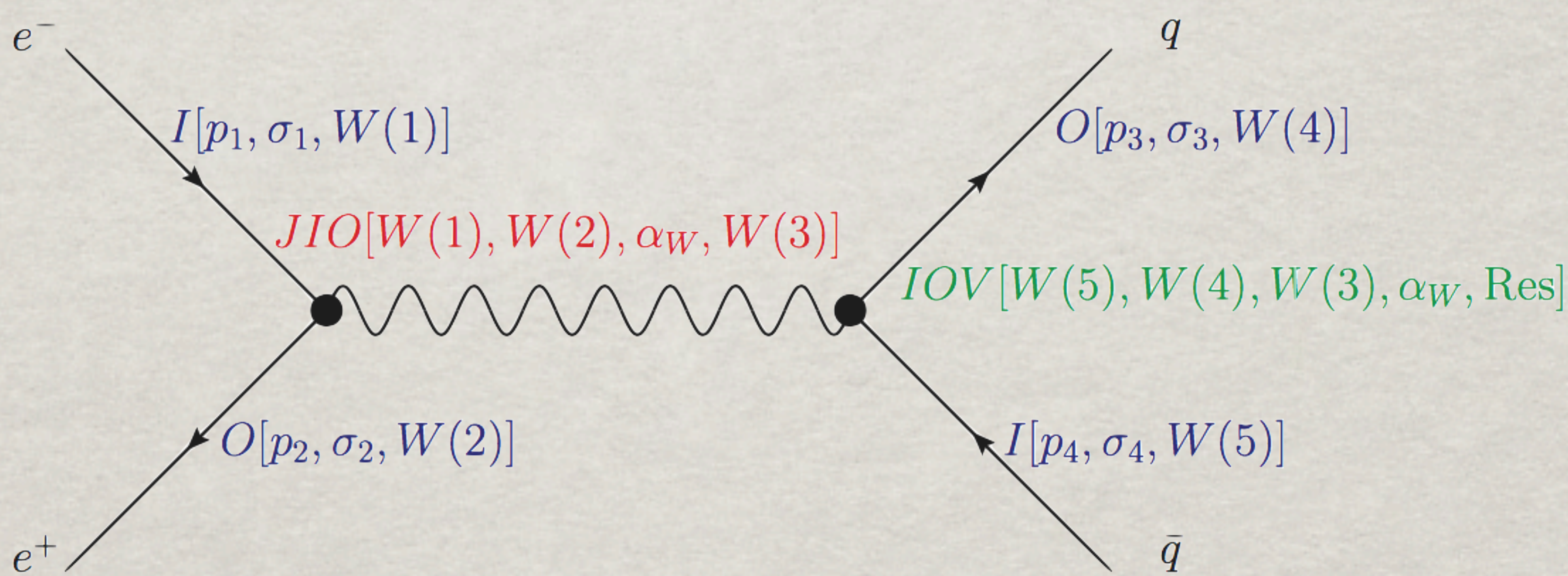


Diag $_3 = [u^*(A)u^*(6)g^*(5)]$

MadGraph

The evolutive way of computing tree-diagrams

- First generates all tree-level Feynman Diagrams
- Compute the amplitude of each diagram using a chain of calls to HELAS subroutines



- Finally square all the related amplitude with their right color factors to construct the full LO amplitude

CutTools

Or how to compute loops without doing so

- CutTools uses the OPP method for loop reduction at the integrand level

$$\bar{q}^2 = q^2 + \tilde{q}^2 \qquad (q \cdot \tilde{q}) = 0$$

$$\bar{D}_i = (\bar{q} + p_i)^2 - m_i^2 \,, \quad p_0 \neq 0 \,.$$

$$\int d^{(d)} \sigma^V = \int d^{(4+\epsilon)} \left(A(\bar{q}) + \tilde{A}(\bar{q}) \right)$$

$$A(\bar{q}) = \frac{N(q)}{\bar{D}_0 \bar{D}_1 \cdots \bar{D}_{m-1}} \quad \left(\tilde{A}(\bar{q}) \to \mathrm{R2} \right)$$

$$\begin{aligned} N(q) &= \sum_{i_0 < i_1 < i_2 < i_3}^{m-1} \left[d(i_0 i_1 i_2 i_3) + \tilde{d}(q; i_0 i_1 i_2 i_3) \right] \prod_{i \neq i_0, i_1, i_2, i_3}^{m-1} D_i \\ &+ \sum_{i_0 < i_1 < i_2}^{m-1} \left[c(i_0 i_1 i_2) + \tilde{c}(q; i_0 i_1 i_2) \right] \prod_{i \neq i_0, i_1, i_2}^{m-1} D_i \\ &+ \sum_{i_0 < i_1}^{m-1} \left[b(i_0 i_1) + \tilde{b}(q; i_0 i_1) \right] \prod_{i \neq i_0, i_1}^{m-1} D_i \\ &+ \sum_{i_0}^{m-1} \left[a(i_0) + \tilde{a}(q; i_0) \right] \prod_{i \neq i_0}^{m-1} D_i \\ &+ \tilde{P}(q) \prod_{i}^{m-1} D_i \end{aligned}$$

- R2 can be obtained with a tree-level-like computation with special Feynman-Rules.
- Evaluation of N(q) for different specific q's allows to algebraically obtain the coefficients a, b, c and d
- Reconstruction of the $\tilde{q}$ dependance of the numerator gives the cut-constructible part R1 of the finite part of the virtual amplitude

Finite part = CC + R1 + R2

Handling BSM Models

UFO models @ NLO

- Additional features in UFO@NLO:

CouplingOrder

- `expansion_order`
- `perturbative_expansion`
- `hierarchy`

CTVertices

```
V_GGZA = CTVertex(name = 'V_GGZA',
                  particles = [P.G, P.G, P.Z, P.A],
                  color = ['Tr(1, 2)'],
                  lorentz = [L.R2_GGVV],
                  loop_particles = [[[P.u], [P.c], [P.t]], [[P.d], [P.s], [P.b]]],
                  couplings = {(0, 0, 0) : C.R2_GGZAup, (0, 0, 1) : C.R2_GGZAdown},
                  type = 'R2')
```

CTParameters

```
MyCTParam = CTParameter(name = 'MyCTParam',
                        type = 'real',
                        value = {-1 : 'A', 0 : 'B'}
                        texname = 'MadRules')
```

`counterterm` attribute to Parameters and Particles

```
Param.GS.counterterm = {(1, 0, 0) : CTParam.G_UVq.value,
                        (1, 0, 1) : CTParam.G_UVb.value,
                        (1, 0, 2) : CTParam.G_UVt.value,
                        (1, 0, 3) : CTParam.G_UVg.value}
```

Automatic Language-independent Output of Helicity Amplitude

O. Mattelaer *et al.* , arXiv:1108.2041 [hep-ph]

From UFO to MG5

ALOHA translate a UFO Lorentz structure

```
VVVV6 = Lorentz(name = 'VVVV6',
          spins = [ 3, 3, 3, 3 ],
          structure = 'Metric(1,4)*Metric(2,3) -Metric(1,3)*Metric(2,4)')
```

into pseudo-HELAS subroutine in a chosen language

```
 VERTEX = COUP*( (V4(1)*( (V2(1)*( (0, -1)*(V3(2)*V1(2))
$ +(0, -1)*(V3(3)*V1(3))+(0, -1)*(V3(4)*V1(4))))+(V1(1)*( (0, 1)
$ *(V3(2)*V2(2))+(0, 1)*(V3(3)*V2(3))+(0, 1)*(V3(4)*V2(4))))))
$ +( (V4(2)*( (V2(2)*( (0, -1)*(V3(1)*V1(1))+(0, 1)*(V3(3)*V1(3))
$ +(0, 1)*(V3(4)*V1(4))))+(V1(2)*( (0, 1)*(V3(1)*V2(1))+(0,
$ -1)*(V3(3)*V2(3))+(0, -1)*(V3(4)*V2(4))))))+( (V4(3)*( (V2(3)
$ *( (0, -1)*(V3(1)*V1(1))+(0, 1)*(V3(2)*V1(2))+(0, 1)*(V3(4)
$ *V1(4))))+(V1(3)*( (0, 1)*(V3(1)*V2(1))+(0, -1)*(V3(2)*V2(2))
$ +(0, -1)*(V3(4)*V2(4))))))+(V4(4)*( (V2(4)*( (0, -1)*(V3(1)
$ *V1(1))+(0, 1)*(V3(2)*V1(2))+(0, 1)*(V3(3)*V1(3))))+(V1(4)
$ *( (0, 1)*(V3(1)*V2(1))+(0, -1)*(V3(2)*V2(2))+(0, -1)*(V3(3)
$ *V2(3))))))))
 END
```

Available in Python, C++ and F77

ALOHA available as a standalone release

New on Aloha

- ALOHA is optimizing the way it does analytical computation

Model name	Loading time, new ALOHA	Loading time, old ALOHA
SM	1.2 s	3 s
MSSM	1.4 s	5 s
Randall-Sundrum	90 s	15 min

- Abbreviation usage improves compilation and running time (up to 40%)
- Possibility to create ALOHA subroutine from the MG5 shell

```
mg5> output aloha FFV1_3
```

- New Outputs/Options in progress (Expected in the v2.0 public release)

Quadruple precision, Feynman Gauge, Spin 3/2,
Complex Mass Scheme, Open Loops techniques, anomalous couplings

Optimizations

- Summing over helicities first, then reducing the matrix element squared.

$$\mathcal{M} = \sum_{l=loop} 2\Re(\sum_{h=hel} \underbrace{\mathrm{CT}[\int \frac{d^D q \mathcal{N}_{l,h}}{D_0 D_1 \cdots D_{n-1}}]\mathcal{A}_h^*}_{\mathcal{A}_l}]) \Rightarrow \mathcal{M} = \sum_{l=loop} 2\Re(\mathrm{CT}[\int d^D q \frac{\sum_{h=hel}\sum_{b=born}\mathcal{N}_{l,h}\mathcal{A}_{b,h}^*}{D_0 D_1 \cdots D_{n-1}}])$$

Also grouping together diagrams with the same denominator structures.

➡ Result: Number of OPP calls decreases from N_{loops} x N_{hels} to $N_{loop\_topology}$!

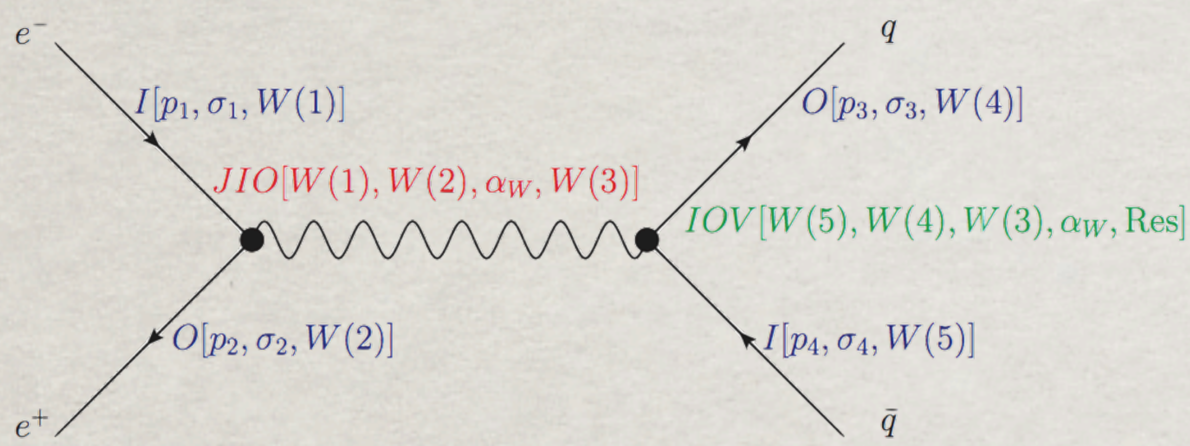
- Exploit the open-loops[F.Cascioli,P.Maierhöfer,S.Pozzorini] technology.
  - ➡ Faster numerator evaluations.
  - ➡ Optimal recycling of the loop wavefunctions.
  - ➡ Remains flexible as ALOHA outputs the building blocks [Work by O.Mattelaer].
- Automatically numerically detect zero and CP-dependent helicity configurations.
- Efficient reconstruction the missing L-cut propagator. Numerator 2 times faster for the massless fermion loops and 3 times for massive ones.

Overall speedup of a factor 10+ w.r.t ML4

Further Optimizations

Further Optimizations

- Recycling wavefunction accross helicity configurations

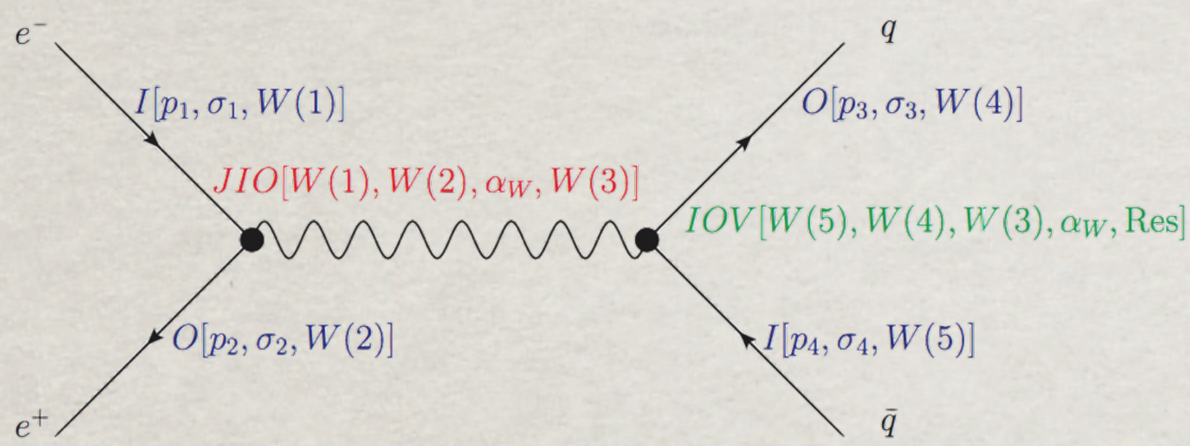


Ex. The same `JIO[e+,e-]` can be used for the two helicity configs of `q q~`

Thanks to open-loops, the loop wavefunctions can also be recycled.

Further Optimizations

- Recycling wavefunction accross helicity configurations



Ex. The same `JIO[e+,e-]` can be used for the two helicity configs of `q q~`

Thanks to open-loops, the loop wavefunctions can also be recycled.

- Grouping diagrams with similar denominator structures

$$\int d^D q \frac{\mathcal{N}_A(q)}{\bar{D}_1\bar{D}_{12}\bar{D}_{123}\bar{D}_{1234}} + \int d^D q \frac{\mathcal{N}_B(q)}{\bar{D}_1\bar{D}_{12}\bar{D}_{1234}}$$

$$= \int d^D q \frac{\mathcal{N}_A(q) + \mathcal{N}_B(q) D_{123}}{\bar{D}_1\bar{D}_{12}\bar{D}_{123}\bar{D}_{1234}}$$

A given triangle and its corresponding box can be reduced at once!

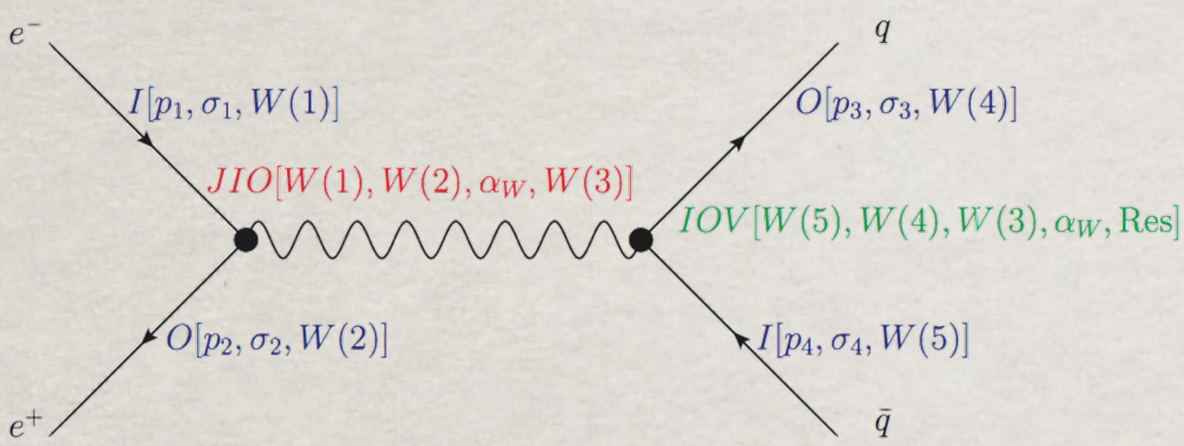
Ex: `g g > g g` would require only six calls to OPP, one per box topology!

But tedious book-keeping and also needs care with dimensionality.

Only useful if dominated by OPP!

Further Optimizations

- Recycling wavefunction accross helicity configurations



Ex. The same `JIO[e+,e-]` can be used for the two helicity configs of `q q~`

Thanks to open-loops, the loop wavefunctions can also be recycled.

- Grouping diagrams with similar denominator structures

$$\int d^D q \frac{\mathcal{N}_A(q)}{\bar{D}_1 \bar{D}_{12} \bar{D}_{123} \bar{D}_{1234}} + \int d^D q \frac{\mathcal{N}_B(q)}{\bar{D}_1 \bar{D}_{12} \bar{D}_{1234}}$$

$$= \int d^D q \frac{\mathcal{N}_A(q) + \mathcal{N}_B(q) D_{123}}{\bar{D}_1 \bar{D}_{12} \bar{D}_{123} \bar{D}_{1234}}$$

A given triangle and its corresponding box can be reduced at once!

Ex: `g g > g g` would require only six calls to OPP, one per box topology!

But tedious book-keeping and also needs care with dimensionality.

Only useful if dominated by OPP!

- Linking MadLoop5 vs Tensor Integral Reduction (TIR).

Speed and Stability

Benchmark with a case study

Four families of $2 \rightarrow 2,3,4$ processes with n=0,1,2 gluons

- `u u~ → t t~ + ng`
- `u u~ → w+ w- + ng`
- `u d~ → w+ g + ng`
- `g g → t t~ + ng`

Same choice as in arXiv:1111:5206

Aim of the study

- Performance of processes of interest from LesHouches wish list
- Benchmark choice common among many codes: easier comparison
- Study of MadLoop5 scaling with leg multiplicity.

Running environnement

- Intel i5 2.8 GHz, only one core exploited
- `gfortran -O0`, similar results with `gfortran -O5` and `ifort`

Code generation

Process	Exe. size [MB]	t_{code} [s]
u u~ → t t~	3.4	9.1
u u~ → w+ w-	3.5	12.4
u d~ → w+ g	3.5	13.9
g g → t t~	3.6	12.8
u u~ → t t~ g	3.7	18
u u~ → w+ w- g	3.9	35
u d~ → w+ g g	3.8	24
g g → t t~ g	4.2	62
u u~ → t t~ g g	4.8	180
u u~ → w+ w- g g	4.8	204
u d~ → w+ g g g	5.2	254
g g → t t~ g g	9.9*	1230
u d~ → w+ g g g g	24**	9370

*,**: Color + helicity data = 25MB , 191 MB

Executable size: **a few MB**

Mild scaling with multiplicity.

Generation time **< 1 hour**

Not a limiting factor.

Could generate

u d~ → w+ g g g g

or even

g g → g g g g

Speed of one-loop amplitudes

Color summed, with OPP

Process	t_{pol} [ms]	n_{hel}	t_{unpol} [ms]
u u~ → t t~	0.52	3/16	0.72
u u~ → w+ w-	0.43	10/36	1.00
u d~ → w+ g	0.87	6/24	1.51
g g → t t~	2.51	6/16	5.42
u u~ → t t~ g	7.44	16/32	27.5
u u~ → w+ w- g	9.3	36/72	81.8
u d~ → w+ g g	13.5	12/48	36.9
g g → t t~ g	40.8	32/32	381
u u~ → t t~ g g	142	32/64	1010
u u~ → w+ w- g g	166	72/144	2820
u d~ → w+ g g g	260	24/96	1'310
g g → t t~ g g	826	64/64	16'900
u d~ → w+ g g g g	9400	48/192	90'900

Polarized timing **competitive**

$t_{2\to2} : t_{2\to3} : t_{2\to4} \lesssim \mathbf{1 : 40 : 800\ ms}$

Unpolarized timing

Good enough for **2 → 3**

Might need further improvement for **2 → 4**

Higher multiplicity

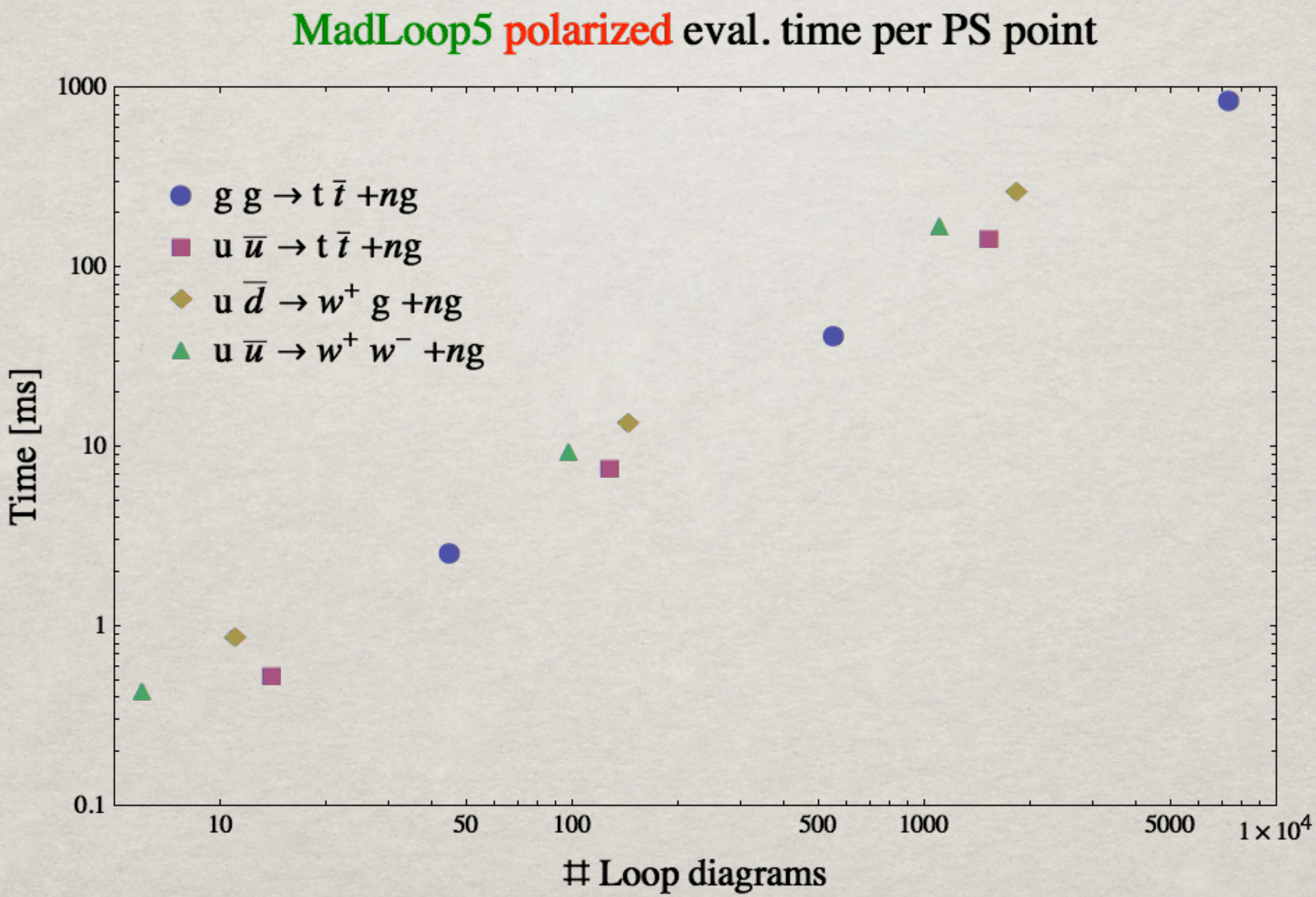
2 → 5 generation feasible

But evaluation is slow, so only useful to cross-check other codes

(Ex. gg→gggg successfully cross-checked vs NGluon[S. Badger])

Linear scaling with # loop diags

Higher rank loops appearing at larger multiplicities are no obstacle!



Numerical stability with OPP

Double precision is not always enough!

Stability probed by two methods:

- Loop reading direction : $D_0 D_1 \ldots D_{n-1} D_n \rightarrow D_n D_{n-1} \ldots D_1 D_0$
  ➥ Advantage: The coefficients of N(q) need not be recomputed.

- Two PS point rotations : (E,x,y,z) → (E,z,-x,-y) and (E,x,y,z) → (E,-z,y,x)

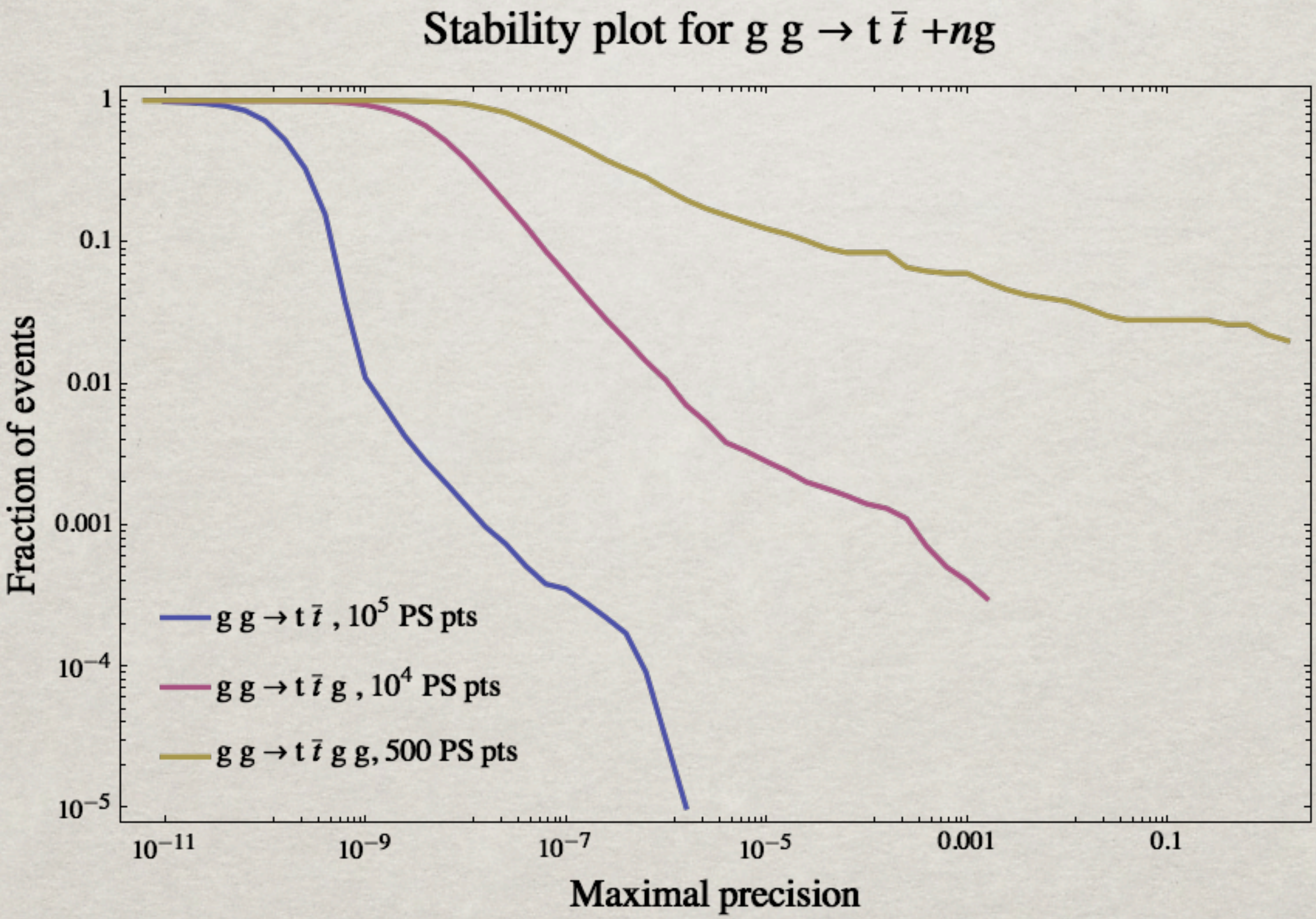
Fraction of points with less than 3 digits accuracy:

2 → 2 **<< 10^{-3} %**
2 → 3 **~ 0.01 %**
2 → 4 **~ 7 % (!)**

Further investigation necessary for **2 → 4**.

Numerical stability with OPP

2 > 4, problems ahead...



Uniformly distributed points with $\sqrt{s}$ = 1TeV, p_t > 50 GeV and ΔR_{ij}>0.5

Numerical stability with OPP

Quadruple precision solves

→ In general, accuracy is **worse** than with Tensor Integral Reduction

→ Quadruple precision cures the Unstable PS (UPS) points but...

- ... is 100 times slower! (This is for complete qd, but double-double would be only 8 times slower)
  - ➥ So 1% of UPS is already enough to double the integration time.
- ... a very (very) small fraction of the points will remain unstable.
  - ➥ What to do with these Exceptional PS points (EPS)?

→ Need to assess that the stability tests used are accurate.

→ Also need to investigate possible correlation between small weight of the ME and the unstability of its evaluation.

MadLoop V4 to V5

Great improvements

✓ = non-optimal | ✓ = done optimally | ✗ = not done | ✗ = not done YET

Task	MadLoop V4	MadLoop V5
Generation of L-Cut diagrams, loop-basis selection	✓-	✓++
Color Factor computation	✓-	✓
Counter-term (UV/R2) diagrams generation	✓-	✓
Mixed order perturbation (generation level)	✗	✓
File output and run-time speed	✓--	✓++
Drawing of Loop diagrams	✗	✓
4-gluon R2 computation	✗	✓
Automated parallel tests	✗	✓
Automatic output sanity checks (Ward, ε^{-2})	✓	✓
EPS handling	✓-- (no qp)	✓- (qp)
Virtual squared	✓-	✓
Decay Chains	✗	✗
Automatic loop-model creation	✗	✗
Complex mass scheme and massive bosons in the loop	✗	✓/✗

Future plans and conclusion

NEXT ON PIPE-LINE

- Complete the Stability study of MadLoop5.
- Publicly release MadGraph5 v2.0!
- Exploit the tool for phenomenology studies.
- Implement a UFO loop model for ElectroWeak corrections.
- Implement some of the further optimizations discussed
- Automatic Loop UFO Model generation with FeynRules
- Decay chains specifications
- Case-study SUSY ? (If not already irrelevant by then)

Thought-to-be Final Word

Be ready to try the MadGraph v2.0 by yourself

MadLoop5 in **MadGraph5 v2.0**, a new 1-loop generator

- Numerical, diagrammatic, some recursive features
- Open-loops method exploited, *i.e.* loop-momentum polynomials
- PUBLIC release very soon (keep an eye on launchpad.net/madgraph5)

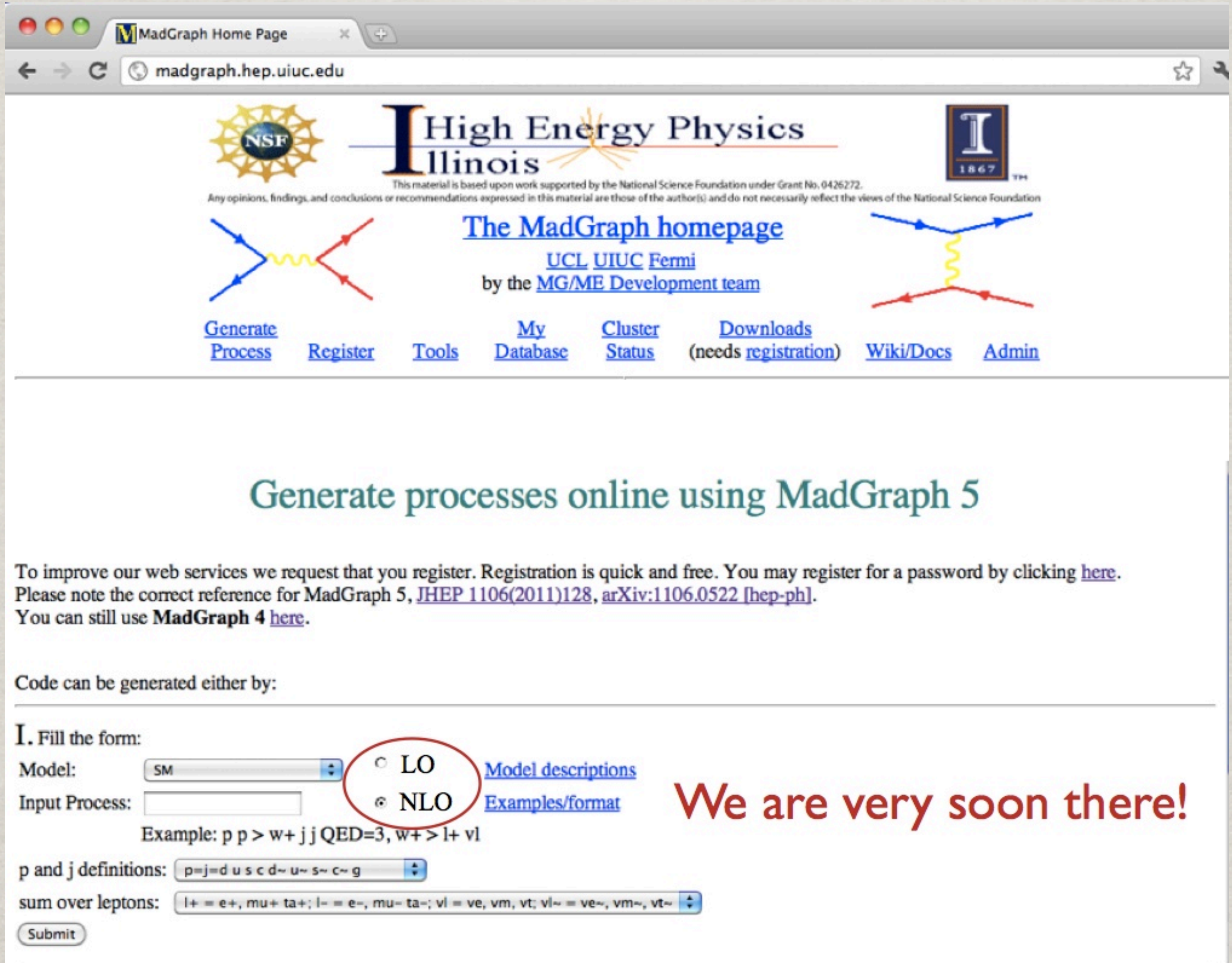
User-friendly, Automated, Flexible, Unique framework

- BSM model covered thanks to UFO and ALOHA flexibility.
- User-friendly thanks to MG5 interfaces.
- Fully automated, from the hard process output to event generation.

Fast, Stable

- Fast enough to cover today's processes of interest, $2 \to 4$ takes O(1s-3s)
- Stable thanks to quadruple precision when needed.

Thanks


MadGraph Home Page
madgraph.hep.uiuc.edu
High Energy Physics
Illinois
The MadGraph homepage
UCL UIUC Fermi
by the MG/ME Development team
Generate Process
Register
Tools
My Database
Cluster Status
Downloads (needs registration)
Wiki/Docs
Admin
Generate processes online using MadGraph 5
To improve our web services we request that you register. Registration is quick and free. You may register for a password by clicking here.
Please note the correct reference for MadGraph 5, JHEP 1106(2011)128, arXiv:1106.0522 [hep-ph].
You can still use MadGraph 4 here.
Code can be generated either by:
I. Fill the form:
Model: SM
LO
NLO
Model descriptions
Input Process:
Examples/format
Example: p p > w+ j j QED=3, w+ > l+ vl
p and j definitions: p=j=d u s c d~ u~ s~ c~ g
sum over leptons: l+ = e+, mu+ ta+; l- = e-, mu- ta-; vl = ve, vm, vt; vl~ = ve~, vm~, vt~
Submit
We are very soon there!

Additional Slides

Process Details

Process	unpol t_{coef} / t_{tot}	pol t_{coef} / t_{tot}	n_{loops} / n_{loop_groups}
u u~ → t t~	42%	20%	8 / 14
u u~ → W^+ W^-	69%	21%	5 / 6
u d~ → W^+ g	52%	16%	9 / 11
g g → t t~	66%	25%	26 / 45
u u~ → t t~ g	78%	18%	54 / 128
u u~ → W^+ W^- g	91%	24%	40 / 98
u d~ → W^+ g g	69%	17%	61 / 144
g g → t t~ g	92%	29%	164 / 556
u u~ → t t~ g g	88%	22%	374 / 1530
u u~ → W^+ W^- g g	95%	25%	260 / 1108
u d~ → W^+ g g g	84%	20%	405 / 1827
g g → t t~ g g	97%	35%	1168 / 7356
u d~ → W^+ g g g g	94%	21%	3255 / 25666

Default vs Open-loop timings