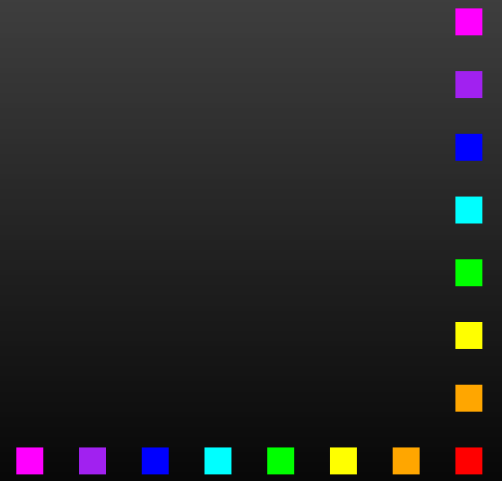
New Features in *FormCalc* 4

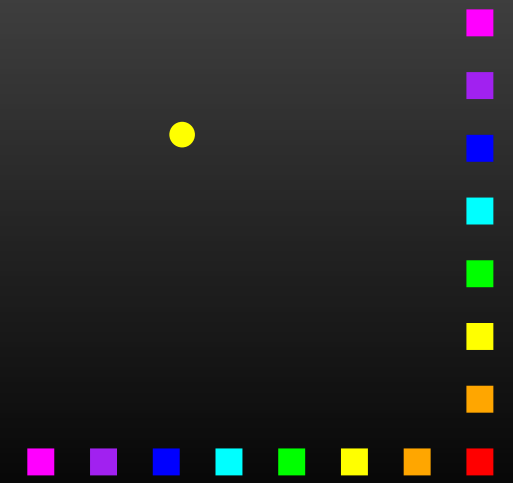
Thomas Hahn

Max-Planck-Institut für Physik
München



New Stuff

Feature	affects Mathematica part	affects Fortran part
Weyl-van der Waerden formalism	•	•
Utilities library		•
CUBA integration library		•
Parallelization, Serial numbers, and Log files		•
Modularity	•	•
Useful scripts		



External Fermion Lines

An amplitude containing external fermions has the form

$$\mathcal{M} = \sum_{i=1}^{n_F} c_i F_i \quad \textbf{where} \quad F_i = \textbf{(Product of)} \langle u | \Gamma_i | v \rangle .$$

$n_F =$ **number of fermionic structures.**

Textbook procedure: Trace Technique

$$|\mathcal{M}|^2 = \sum_{i,j=1}^{n_F} c_i^* c_j F_i^* F_j$$

where $F_i^* F_j = \langle v | \bar{\Gamma}_i | u \rangle \langle u | \Gamma_j | v \rangle = \mathrm{Tr}\left(\bar{\Gamma}_i \, |u\rangle\langle u| \, \Gamma_j \, |v\rangle\langle v|\right) .$

Problems with the Trace Technique

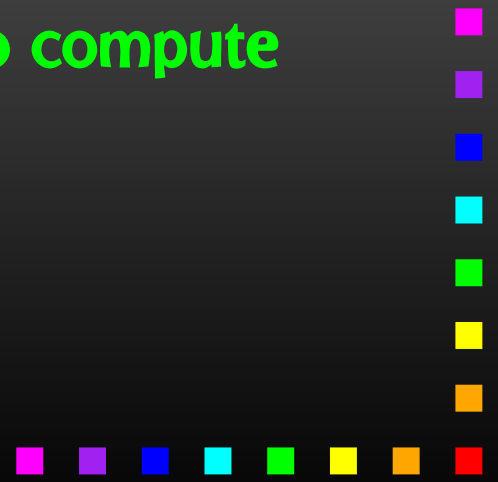
PRO: Trace technique is independent of any representation.

CON: For n_F F_i's there are n_F^2 $F_i^* F_j$'s.

Things get worse the more vectors are in the game: multi-particle final states, polarization effects . . .

Essentially $n_F \sim$ (# of vectors)! because all combinations of vectors can appear in the Γ_i.

Solution: Use Weyl-van der Waerden formalism to compute the F_i's directly.



Sigma Chains

Define **Sigma matrices** and **2-dim. Spinors** as

$$\sigma_\mu = (\mathbb{1}, -\vec{\sigma})\,,$$
$$\overline{\sigma}_\mu = (\mathbb{1}, +\vec{\sigma})\,,$$

$$\langle u|_{4\mathrm{d}} \equiv \left(\langle u_+|_{2\mathrm{d}}\,, \langle u_-|_{2\mathrm{d}}\right),$$
$$|v\rangle_{4\mathrm{d}} \equiv \begin{pmatrix} |v_-\rangle_{2\mathrm{d}} \\ |v_+\rangle_{2\mathrm{d}} \end{pmatrix}.$$

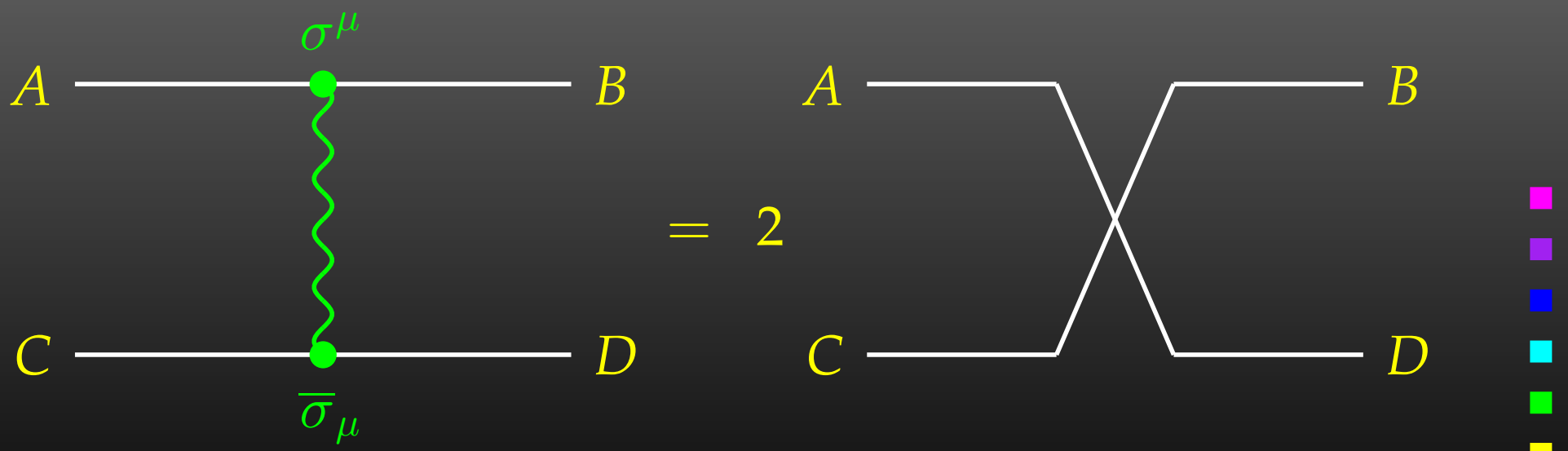
Using the chiral representation it is easy to show that **every chiral 4-dim. Dirac chain** can be converted to a ***single*** **2-dim. sigma chain:**

$$\langle u|\,\omega_-\gamma_\mu\gamma_\nu\cdots|v\rangle = \langle u_-|\,\overline{\sigma}_\mu\sigma_\nu\cdots|v_\pm\rangle\,,$$
$$\langle u|\,\omega_+\gamma_\mu\gamma_\nu\cdots|v\rangle = \langle u_+|\,\sigma_\mu\overline{\sigma}_\nu\cdots|v_\mp\rangle\,.$$

Fierz Identities

With the Fierz identities for sigma matrices it is possible to remove all Lorentz contractions between sigma chains, e.g.

$$\langle A|\,\sigma_\mu\,|B\rangle\,\langle C|\,\overline{\sigma}^\mu\,|D\rangle \;=\; 2\,\langle A|D\rangle\,\langle C|B\rangle$$



Implementation

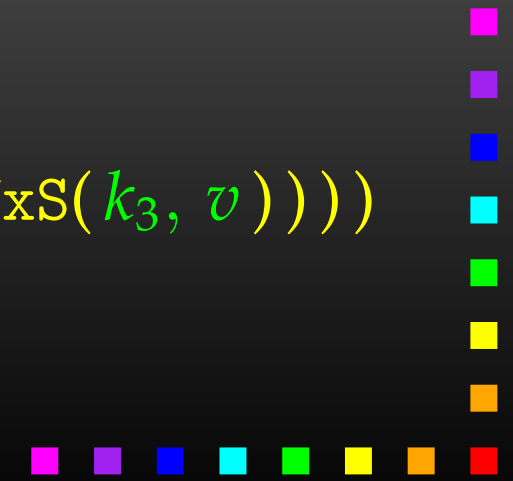
- **Objects (arrays):** $|u_\pm\rangle \sim \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \quad (\sigma \cdot k) \sim \begin{pmatrix} a & b \\ c & d \end{pmatrix}$
- **Operations (functions):**

$$\langle u|v\rangle \sim (u_1 \;\; u_2) \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \qquad \texttt{SxS}$$

$$(\overline{\sigma}\cdot k)\,|v\rangle \sim \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \qquad \texttt{VxS},\ \texttt{BxS}$$

Sufficient to compute any sigma chain:

$$\langle u|\,\sigma_\mu\overline{\sigma}_\nu\sigma_\rho\,|v\rangle\,k_1^\mu k_2^\nu k_3^\rho = \texttt{SxS}(u,\ \texttt{VxS}(k_1,\ \texttt{BxS}(k_2,\ \texttt{VxS}(k_3,\ v))))$$



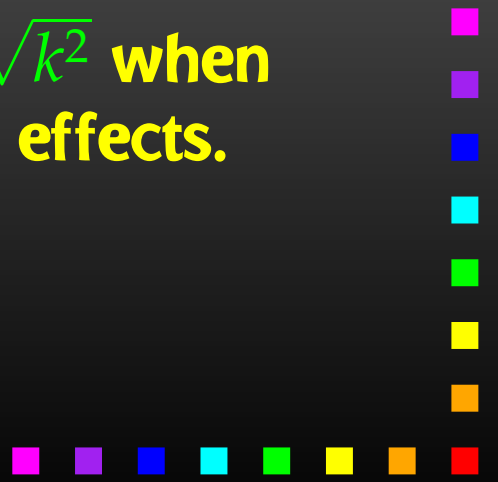
More Freebies

- Polarization does not 'cost' extra:
  = Get spin physics for free.
- Better numerical stability because components of k^μ are arranged as 'small' and 'large' matrix entries, viz.

$$\sigma_\mu k^\mu = \begin{pmatrix} k_0 + k_3 & k_1 - \mathrm{i}k_2 \\ k_1 + \mathrm{i}k_2 & k_0 - k_3 \end{pmatrix}$$

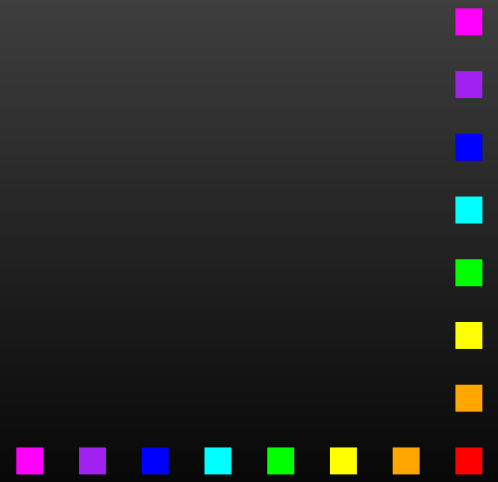
$$\downarrow$$

Large cancellations of the form $\sqrt{k^2 + m^2} - \sqrt{k^2}$ when $m \ll k$ are avoided: better precision for mass effects.



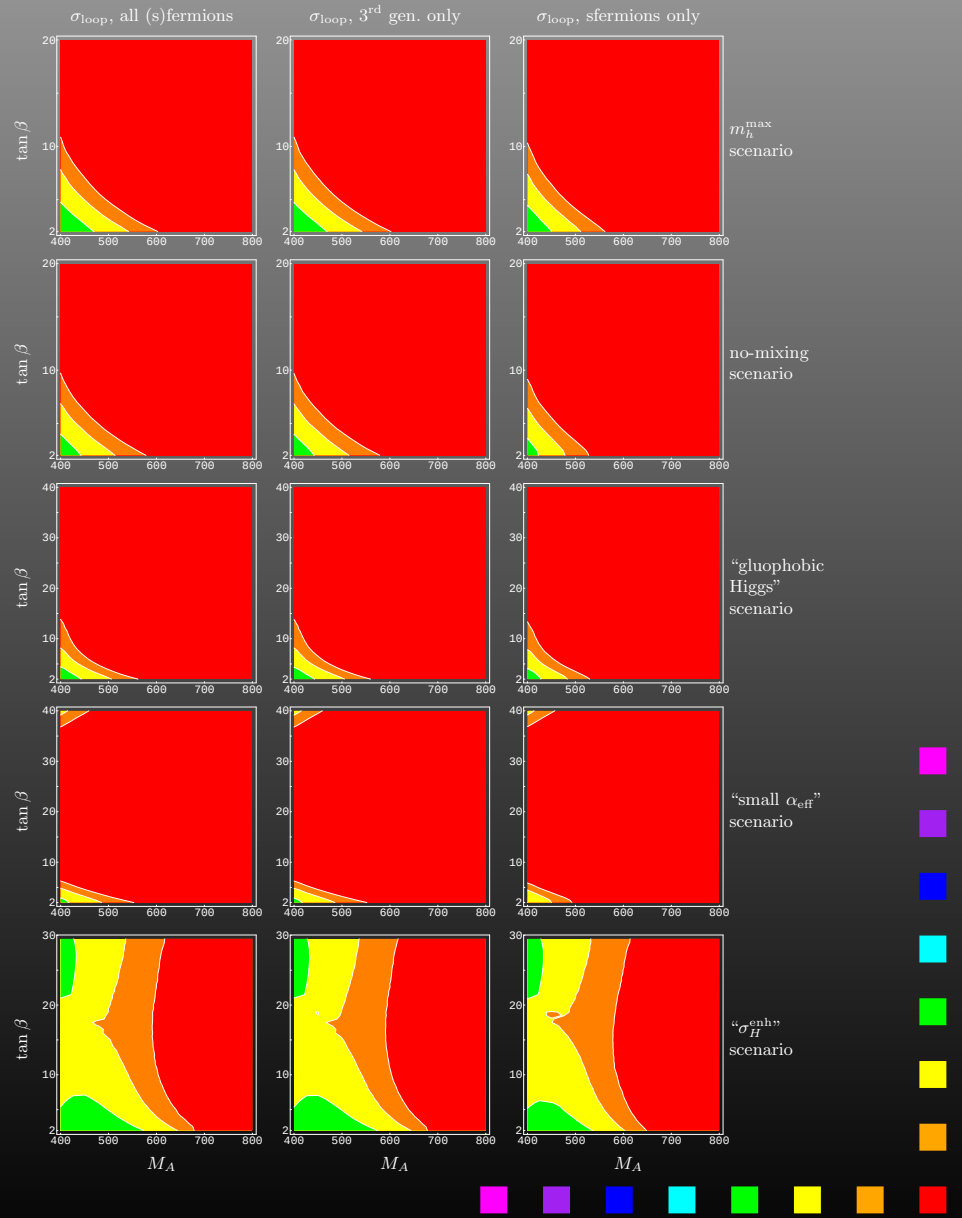
Utilities Library

- All utility functions are now collected in a library which is compiled when *FormCalc* is installed.
- Currently includes the categories
  - ▷ System utilities (log file management),
  - ▷ Kinematic functions (SxS, VxS, . . .),
  - ▷ Diagonalization routines (Eigenvalues, . . .),
  - ▷ Univariate integrators (Gauss, Patterson),
  - ▷ Multivariate integrators (CUBA library).
- Easy to add code.
- Linker selects only the necessary routines.



More Motivation

MSSM parameter scans in the M_A-$\tan\beta$ plane for $e^+e^- \to \nu\bar{\nu}H$, self-energy and vertex diagrams only



Approximate computing time: **1 CPU-Month**

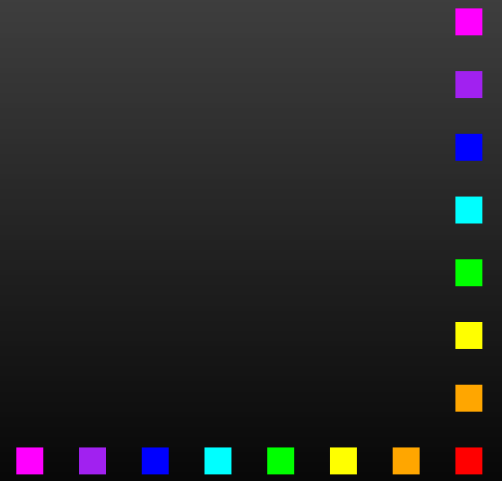
4D phase-space integration: Vegas, max. points: **100,000**

MSSM calculations $=$ SM calculations $\times\ \mathcal{O}(2) \times$ **N**

Which Screws can we Tighten?

- Phase-space integration (reduce the 100,000)
  New CUBA library offers new or improved versions of four general-purpose multidimensional integration methods.

- Parallelization (distribute the N)
  Loop unnesting via a serial number makes parallelization possible even with a shell script.



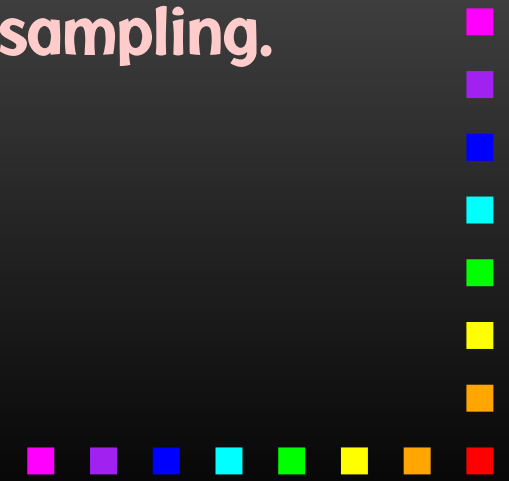
Routines in the CUBA Library

Routine	Basic method	Type	Variance reduction
Vegas	Sobol sample	Monte Carlo	importance sampling
Suave	Sobol sample	Monte Carlo	globally adaptive subdivision
Divonne	Korobov sample *or* Sobol sample *or* cubature rules	Monte Carlo Monte Carlo deterministic	stratified sampling, aided by methods from numerical optimization
Cuhre	cubature rules	deterministic	globally adaptive subdivision

- Very similar invocation (easily interchangeable)
- Fortran, C/C++, Mathematica interface provided
- Can integrate vector integrands

Vegas Cheat Sheet

- Monte Carlo algorithm.
- Variance reduction: importance sampling.
- Algorithm:
  - Iteratively build up a piecewise constant weight function, represented on a rectangular grid.
  - Each iteration consists of a sampling step followed by a refinement of the grid.
- New: Uses Sobol quasi-random numbers for sampling.



Suave Cheat Sheet

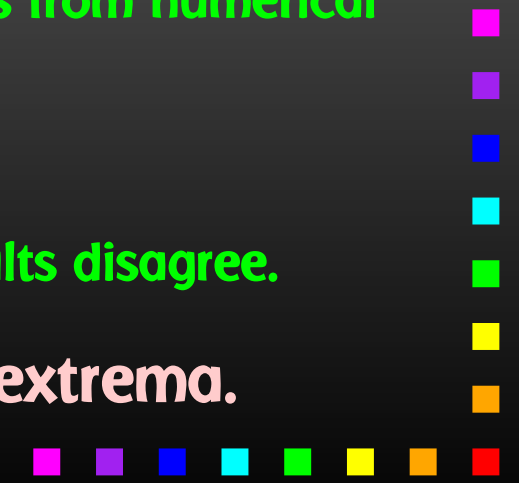
- Monte Carlo algorithm.
- Variance reduction: Vegas-style importance sampling combined with globally adaptive subdivision.
- Algorithm:
  - Until the requested accuracy is reached, bisect the region with the largest error along the axis in which the fluctuations of the integrand are reduced most.
  - Prorate the number of new samples in each half for its fluctuation.
- New: Hybrid Vegas/Miser algorithm.

Divonne Cheat Sheet

- Monte Carlo algorithm (+ cubature rules for comparison).
- Variance reduction: Stratified sampling.
- Algorithm:
  - ▷ PHASE 1: Partition the integration region such that all subregions have an approximately equal value of

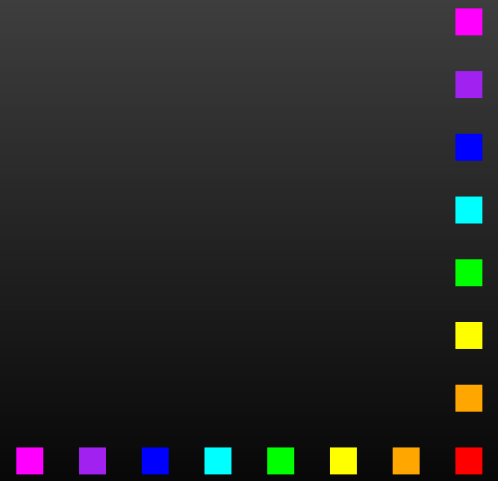
$$\text{spread}(r) = \frac{1}{2}\,\text{Vol}(r)\Big(\max_{\vec{x}\in r} f(\vec{x}) - \min_{\vec{x}\in r} f(\vec{x})\Big).$$

    Minimum and maximum are sought using methods from numerical optimization.
  - ▷ PHASE 2: Sample the subregions independently.
  - ▷ PHASE 3: Further subdivide or sample if 1 & 2 results disagree.
- New: Phase 3, Allows the user to point out extrema.

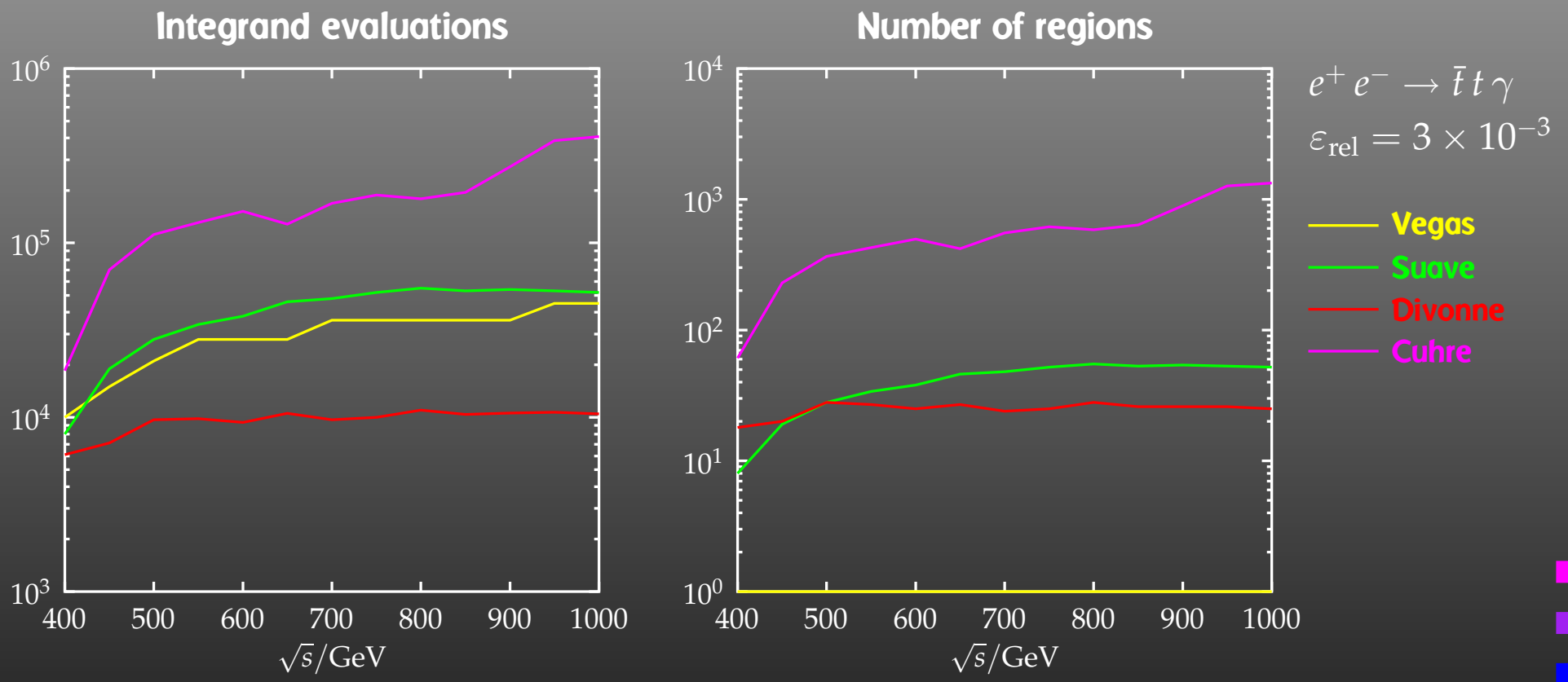


Cuhre Cheat Sheet

- Deterministic algorithm (uses cubature rules of polynomial degree).
- Variance reduction: Globally adaptive subdivision.
- Algorithm:
  - Until the requested accuracy is reached, bisect the region with the largest error along the axis with the largest fourth difference.
- New: Consistent interface only, same as DCUHRE.



Test Run

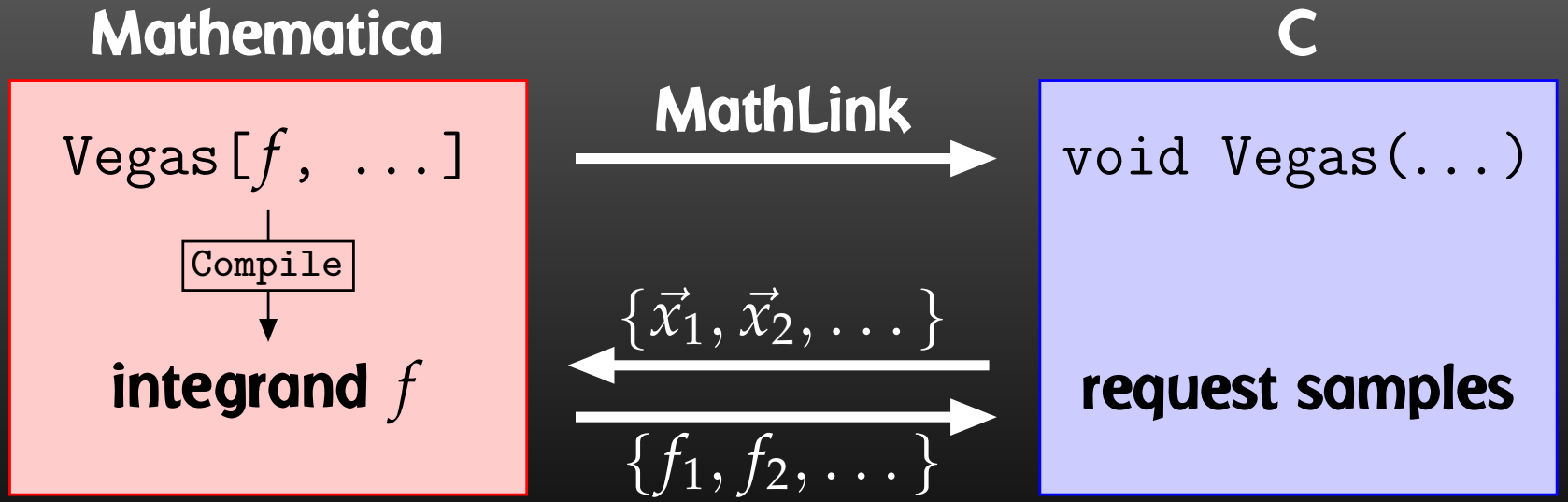


Above all: Very important to have several methods for cross-checking the results!

Mathematica interface

- **Used almost like** `NIntegrate`.
- **The integrand is evaluated completely in Mathematica. Can do things like**

```
Cuhre[PolyLog[2, x y], {x,.2,.3}, {y,.4,.5}]
```



Parallelization

- Network Parallelization
  Usually requires MPI or similar library.

  PRO: Low cost, institutes often have a sizeable cluster of PCs installed already – think O(50) speed-up.

  CON: Slow inter-process communication via network.

- Symmetric Multiprocessing (SMP)
  OS-supported (threads) in C/C++, Java, etc. Must use fork/wait in native Fortran 77 due to static variables, I/O.

  PRO: Fast inter-process communication via shared memory.

  CON: Still expensive, might change with Opteron/Itanium.

Very roughly:

1	2	3	4	8	CPUs
1	2	60	80	180	kEUR

Parameter Scans

With the preprocessor definitions in `run.F` one can either

- assign a parameter a fixed value, as in

```
#define LOOP1 TB = 1.5D0
```

- declare a loop over a parameter, as in

```
#define LOOP1 do 1 TB = 2,30,5
```

which computes the cross-section for `TB` values of 2 to 30 in steps of 5.

Main Program:

```
      LOOP1
      LOOP2
        ⋮
      (calculate
      cross-section)
1     continue
```

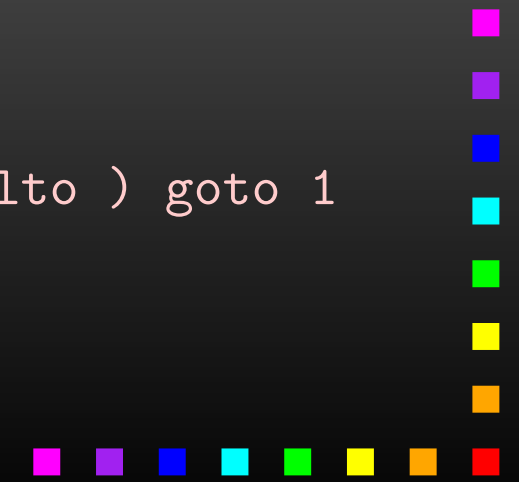
Scans are perfect for parallelization:
Each iteration of the loops can be computed independently!

Unravelling Parameter Scans

How can the distribution of iterations be automated if the loops are a) user-defined b) usually nested?

Solution: Introduce a serial number

```
      subroutine ParameterScan(serialfrom, serialto)
      integer serialfrom, serialto, serial

      serial = 0

      LOOP1
      LOOP2
        ⋮

      serial = serial + 1
      if( serial.lt.serialfrom .or. serial.gt.serialto ) goto 1

      (calculate cross-section)

1     continue
      end
```



Shell-script Parallelization

Distribution of loop iterations on N machines is now trivial:

- Send serial numbers 1, N+1, 2N+1, . . . on machine 1,
- Send serial numbers 2, N+2, 2N+2, . . . on machine 2, etc.

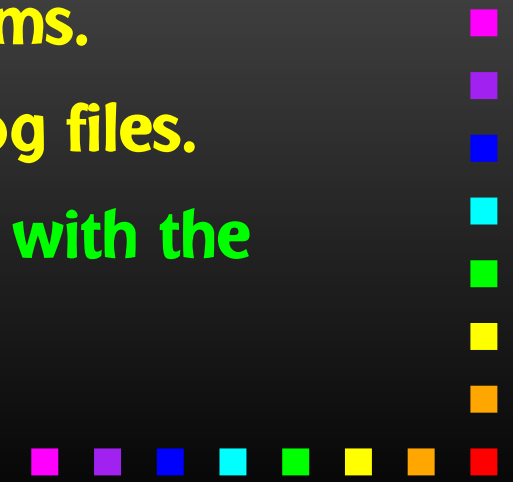
With a little interfacing to the OS,

- redirect each iteration's output to a separate file,
- enter range of serial numbers on command line,
- exit value = actual number of iterations performed,

parallelization can be controlled from a simple shell script (and of course with any batch system).

Log file management

- *All* output written to $*$ (Fortran's default unit).
- 'Real' data marked by @ in column 1.
- Easier to pinpoint errors:
  Error messages appear right next to data.
- C function redirects output to different log file for each iteration.
- Log file gets perms 744 while running, 644 when finished. Skip existing log files with 644 perms.
- Simple shell script collects 'real' data from log files.
- Can resume aborted calculation by invoking with the *same* command-line parameters!



Modularity

- Generated code written to its own subdirectory.
- A sub-makefile is generated for each subdirectory.
- Master makefile no longer overwritten, user can save a modified copy together with the customized drivers.
- Compilation by sub-make (rather than direct include), thus largely independent from master.
- Optional compile-time prefix for all externally visible symbols avoids namespace conflicts
  = Can link several generated modules together.

  Example: different partonic processes contributing to the cross-section.

Useful Shell Scripts

- `sfx` **packs all files into a self-extracting archive, e.g.**

A
```
(myhost:mydir)> ./sfx
(myhost:mydir)> echo "Joe, here's the code" | \
                pine joe@joeshost -attach mydir.sfx
```

B
```
(joeshost:joesdir)> pine    (save attachment)
(joeshost:joesdir)> ./mydir.sfx x
```

- `turnoff` **switches off/on the evaluation of modules, e.g.**

```
./turnoff box      (turn off the boxes)
./turnoff          (restore)
```

- `pnuglot` **makes a customizable plot from a data file.**
- `submit` **automatically distributes a job on a cluster, e.g.**

```
submit run uuuu 0,1000
```

Summary

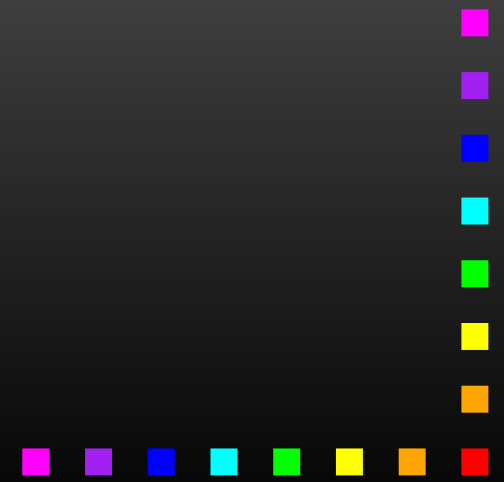
Lots of new features in *FormCalc* 4!

New Concepts:

- Weyl-van der Waerden formalism radically simplifies multi-leg calculations.
- CUBA library provides four independent algorithms for multidimensional numerical integration.
  Available at http://www.feynarts.de/cuba (LGPL).

Improved Software Engineering:

- Simple parallelization mechanism.
- Log file management allows easy restart.
- Modular code generation, master makefile.



Outlook

Next on the to-do list:

- Update the manual,
- 5-point functions,
- Subtraction formalism for QED,
- $2 \to 4$ kinematics.

- MSSM Model file including counter terms finished (T. Fritzsche), write-up in progress.