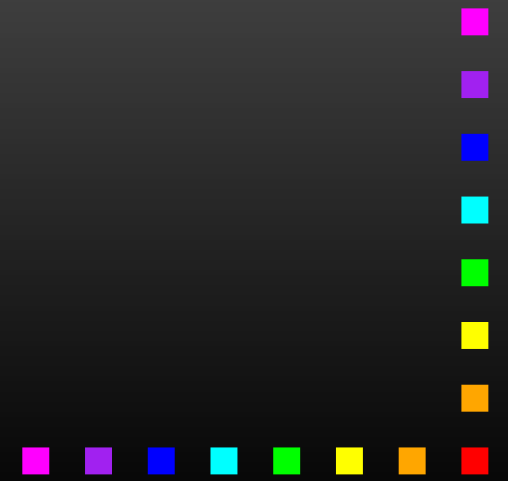
New Developments for Automatic Loop Calculations

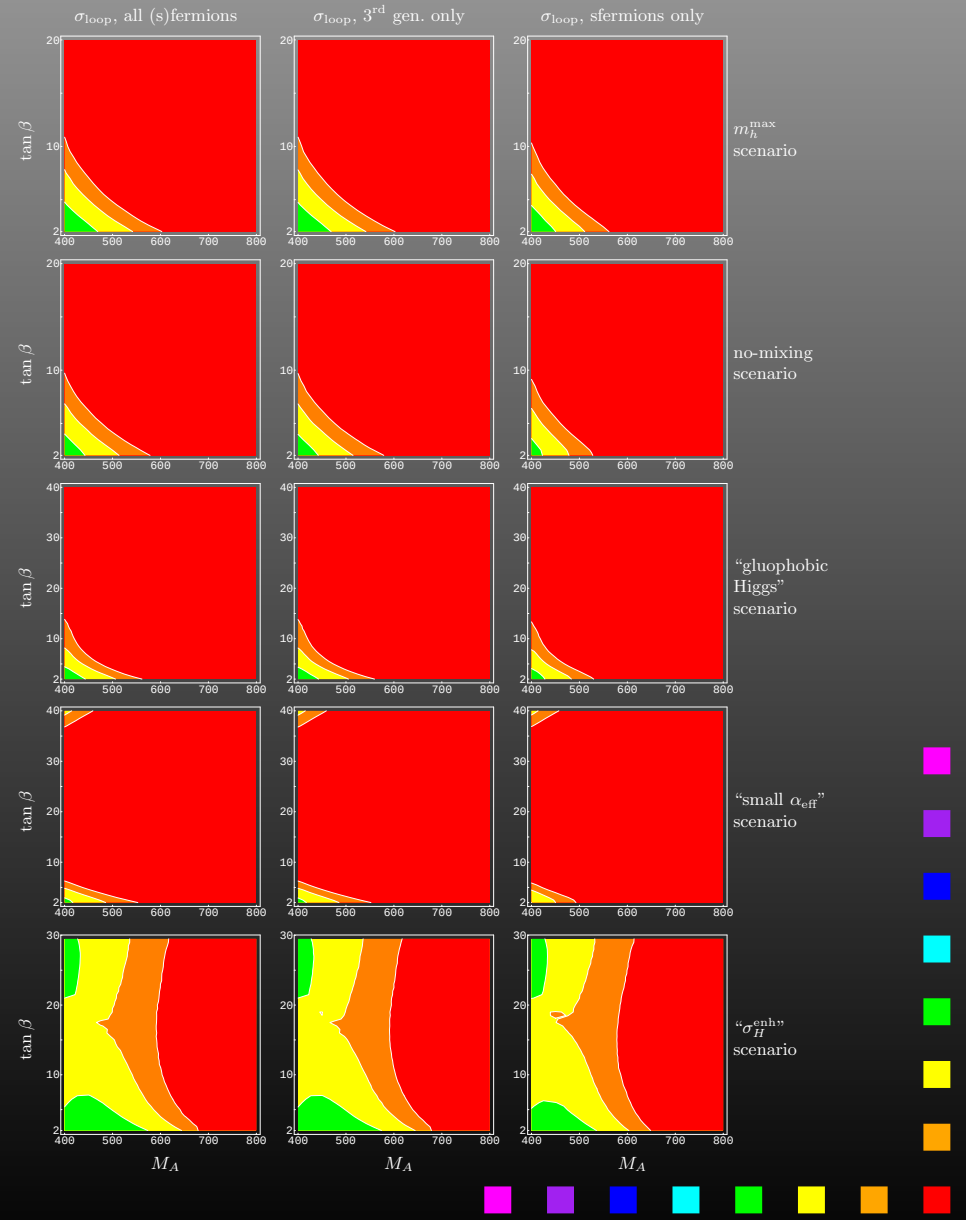
Thomas Hahn

Max-Planck-Institut für Physik
München



Problem

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.
  The flexibility of a general-purpose algorithm (compared e.g. to a multi-channel Monte Carlo) is particularly appreciated in the setting of automatically generated modules of code which the user may plug into different applications.

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

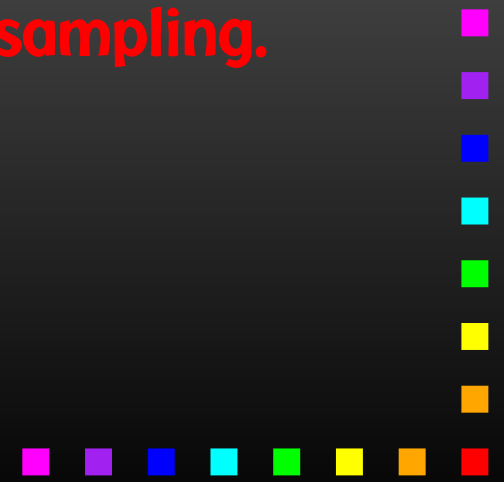
Routines in the CUBA Library

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

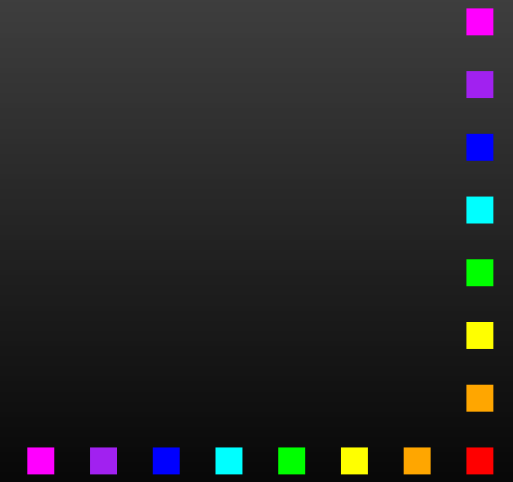
- Monte Carlo algorithm (+ cubature rules for comparison).
- Variance reduction: Stratified sampling.
- Algorithm:
  - ▷ PHASE 1: Partition the integration region such that all subregions have an approximately equal value of

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

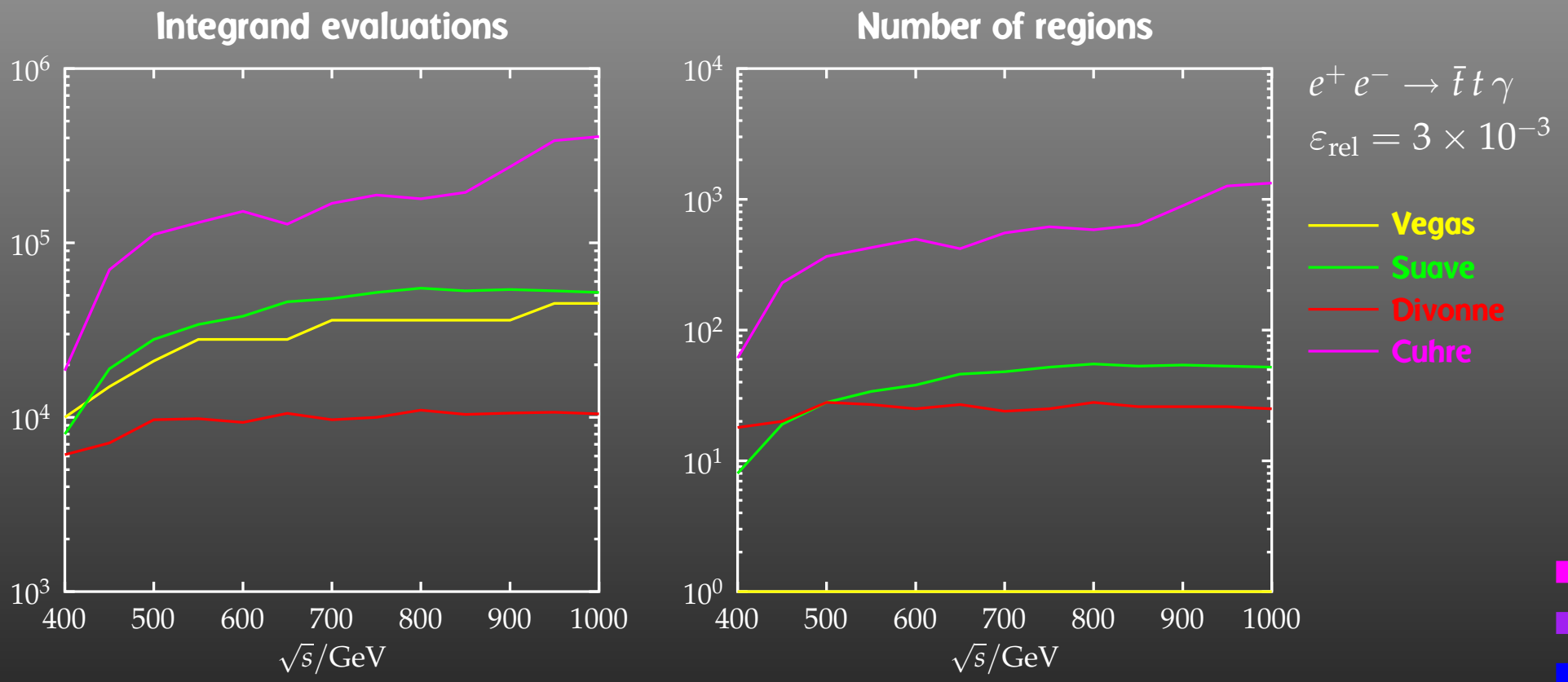
    Minimum and maximum are sought using methods from numerical optimization.
  - ▷ PHASE 2: Sample the subregions independently.
  - ▷ PHASE 3: Further subdivide or sample if 1 & 2 results disagree.
- New: Phase 3, Allows the user to point out extrema.

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!

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) speedup.
  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!

Unraveling 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

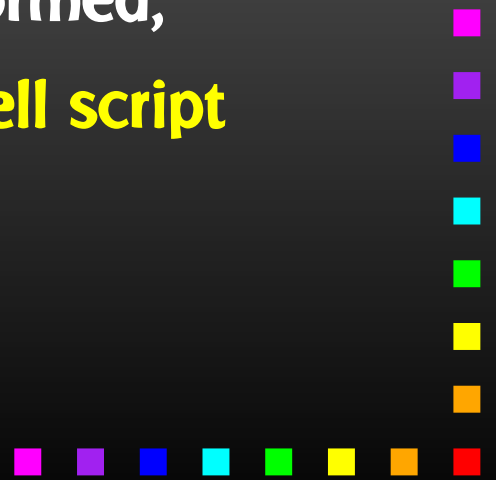
Distribution of loop iterations is now trivial:

- Send serial numbers 1...n on machine 1,
- Send serial numbers (n+1)...2 n 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).



Summary

Two new developments can dramatically reduce computing time, in particular of parameter scans:

- New CUBA library provides four independent algorithms for multidimensional numerical integration.
  Available at http://www.feynarts.de/cuba (LGPL).
- New simple parallelization mechanism.
  Available already in the latest *FormCalc* version, but straightforward to implement anywhere.

Work in progress:
Parallelization of integration routines, e.g. conceptually easy to parallelize Divonne's Phase 2.