Modeling turbulence via numerical functional integration

I. Honkonen[1], J. Honkonen[2]
ilja.honkonen@fmi.fi
[1]Finnish Meteorological Institute
[2]Finnish National Defence University

Math 1/3

$$u_{i+1}(x_n) = u_i(x_n) + \left\{ -u_i(x_n) \left[\frac{u_i(x_n) - u_i(x_{n-1})}{a} \right] \right.$$

$$\left. + \left[\frac{u_i(x_{n+1}) - 2u_i(x_n) + u_i(x_{n-1})}{a^2} \right] \right\} (t_{i+1} - t_i) + F_{i+1}(x_n) - F_i(x_n)$$

$$p(\{F_i(x_n)\}) = C \exp \left[-\frac{1}{2} \sum_i \sum_{x_n} \frac{(F_{i+1}(x_n) - F_i(x_n))^2}{t_{i+1} - t_i} \right] .$$

- Stochastic difference equation with increment of a Brownian motion
  - Burgers equation toy problem
  - i: time, x: space
- Joint probability density function

Math 2/3

$$\langle u_j(x_m)u_j(x_n)\rangle = C\int \prod_i \prod_{x_n} du_i(x_n)\, u_j(x_m)u_j(x_n)$$
$$\times \exp\Bigg[-\frac{1}{2}\sum_i \sum_{x_n}\Bigg(u_{i+1}(x_n) - u_i(x_n) + \Bigg\{u_i(x_n)\left[\frac{u_i(x_n)-u_i(x_{n-1})}{a}\right]$$
$$-\left[\frac{u_i(x_{n+1})-2u_i(x_n)+u_i(x_{n-1})}{a^2}\right]\Bigg\}(t_{i+1}-t_i)\Bigg)^2 \frac{1}{t_{i+1}-t_i}\Bigg]$$

- Generating function of single-time correlation functions

Math 3/3

$$
\begin{aligned}
\langle u_1(x_m)u_1(x_n)\rangle = C \int \prod_{x_n} du_1(x_n)\, u_1(x_m)u_1(x_n) \\
\times \exp\Bigg[-\frac{1}{2}\sum_{x_n}\Bigg(u_1^2(x_n) + \Bigg\{u_1(x_n)\left[\frac{u_1(x_n)-u_1(x_{n-1})}{a}\right] \\
- \left[\frac{u_1(x_{n+1})-2u_1(x_n)+u_1(x_{n-1})}{a^2}\right]\Bigg\}^2 t_1\Bigg)\Bigg]
\end{aligned}
$$

- Simplified version: only initial condition (1st time step)

Integrand implemented in serial programs

$$S_1 = \sum_{n=1}^{D} u(n)^2$$

$$S_2 = \sum_{n=1}^{D} [-u(n)\frac{u(n+1) - u(n-1)}{2} + u(n+1) - 2u(n) + u(n-1)]^2$$

$$I = \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} u(i)u(j)exp[-(S_1 + S_2)/2]du(1)du(2)...du(D)$$

Grid spacing, viscosity, etc. = 1

Implementation

Code available at github.com/iljah/hdintegrator

Serial programs

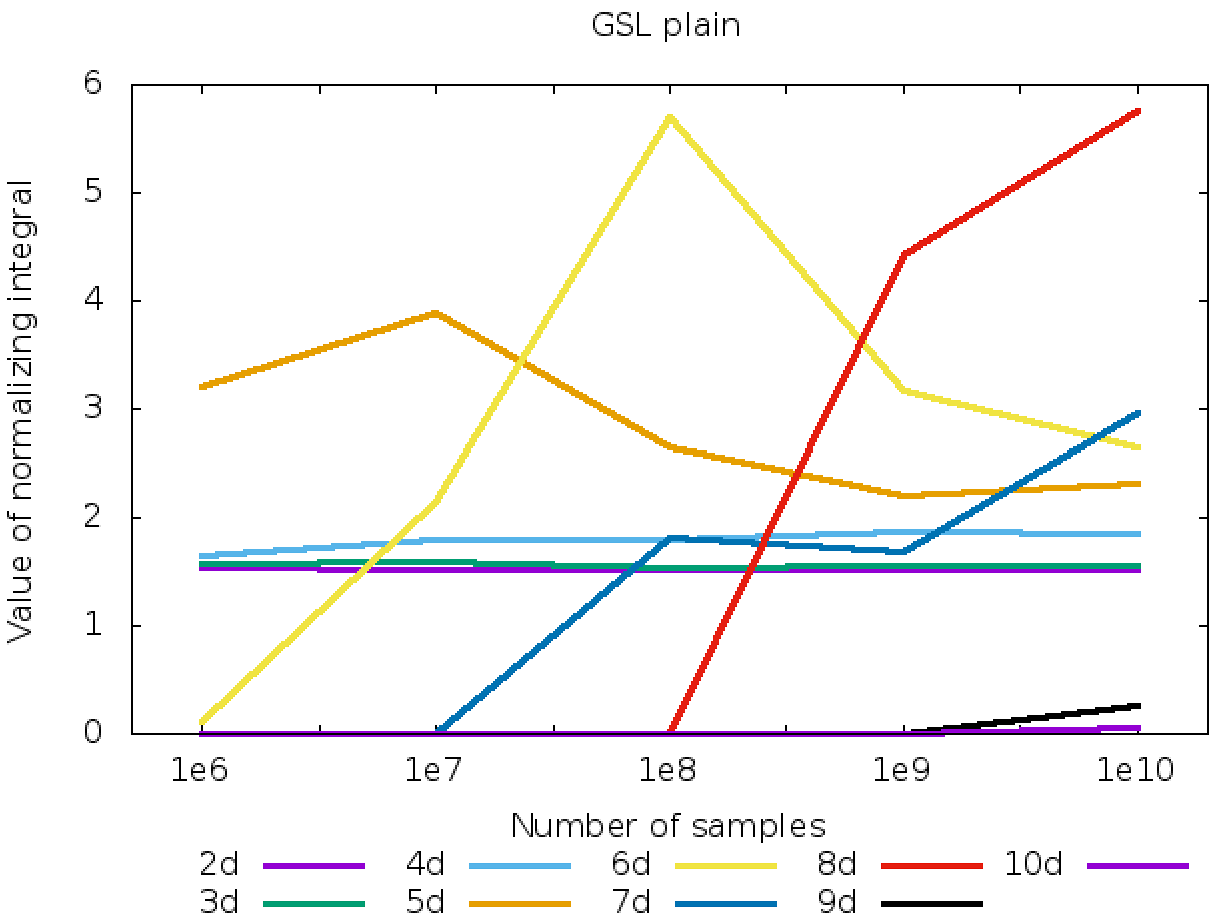
- Calculate hard-coded integrand
- Read parameters from standard input
- Write results to standard output
- Implemented with: SciPy nquad, Cubature, GSL MC methods

Parallel wrapper

- Divides volume into parts
- Calls serial programs for each part
- If no convergence, divides again
- Implemented with mpi4py
- Rank 0 coordinates work, ranks > 0 call serial progs

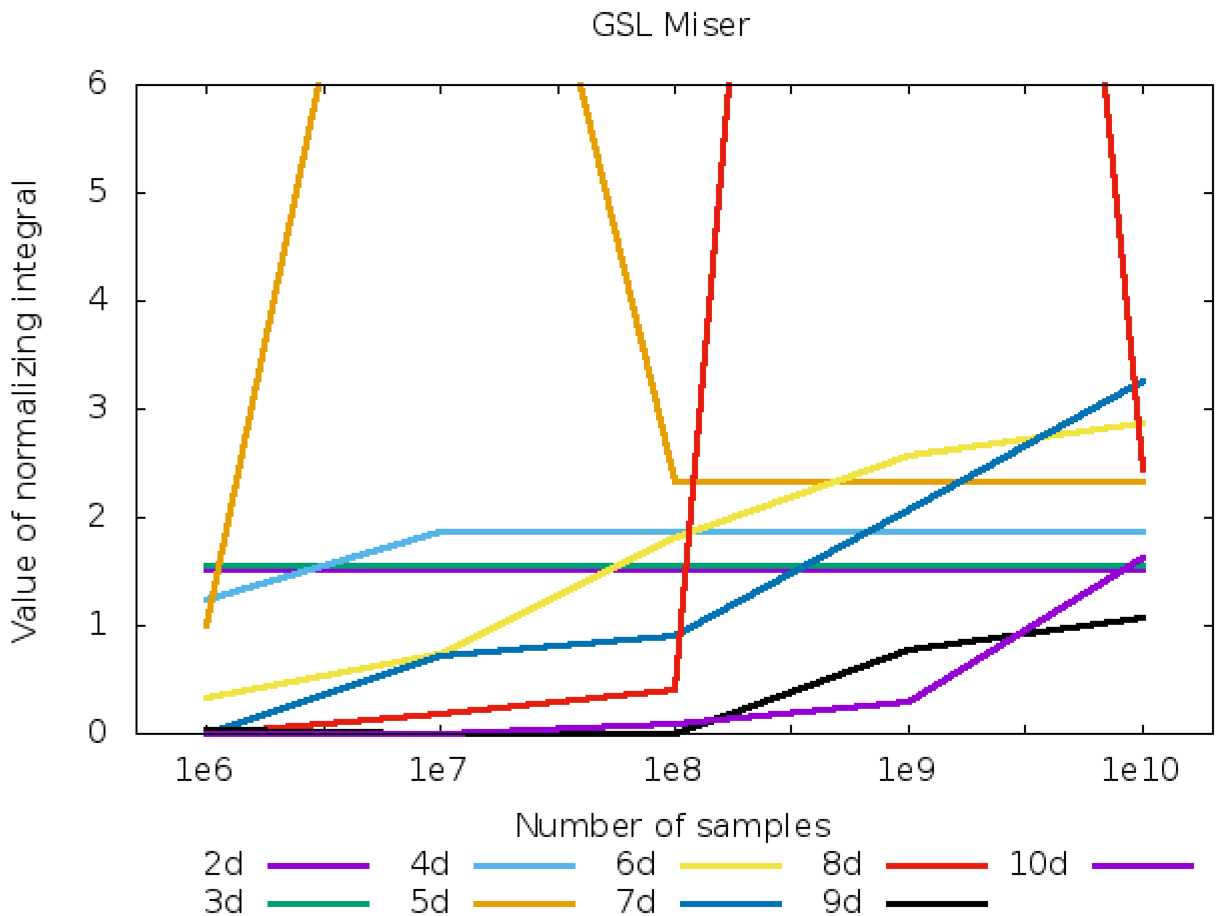
Results from serial programs: GSL plain

- GSL MC methods seem most viable at the moment
- In plain method 1e10 sample integration takes about 1 h
- Converges in < 7d



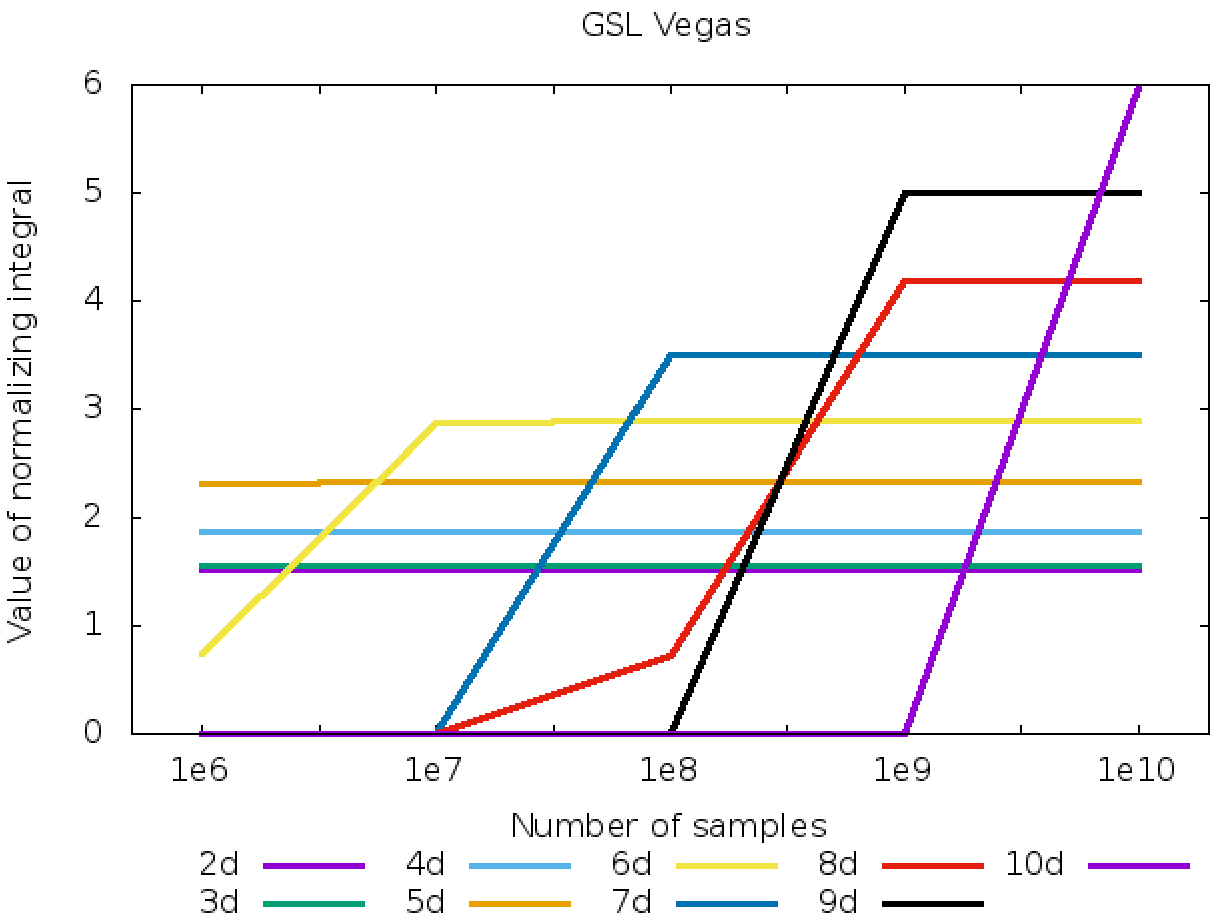
Results from serial programs: GSL Miser

- In Miser method 1e10 sample integration takes about 1 h
- Converges in < 7d
- Correlated integrals more difficult to calculate

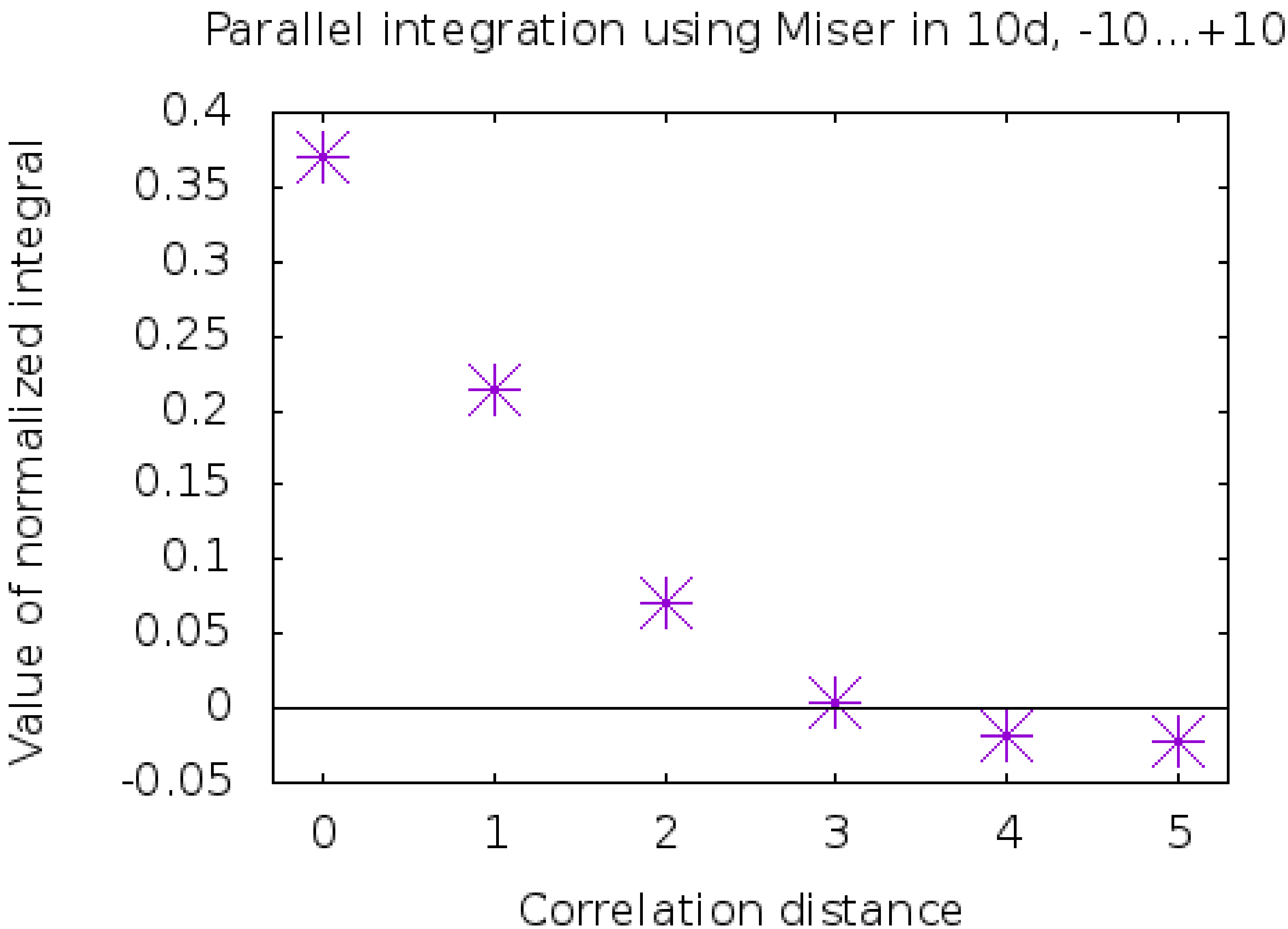


Results from serial programs: GSL Vegas

- In Vegas method 1e10 sample integration takes about 5 h
- Converges in < 10d
- Integration range in all previous from -15 to +15 in every dimension



Very preliminary result from parallel integration, still investigating convergence



Math TODO

- Add time-dependence
  - 1d → 2d grid of dims
- Integrate Navier-Stokes
- Integrate over -1...1 instead of -inf...inf?
- Increase Reynolds number via changing:
  - dx & dt (resolution), viscosity, forcing
- Only large-scale forcing instead of white spatial noise

Technical TODO

- Integrate in 100s of dimensions
- Use GPUs for integration
- Integrate in cloud
  - AWS lambda looks good
- Allow continuing integral calculation in serial progs
- Non-rectangular subvolumes?
  - Spherical coordinates?
- Find convergence in depth-first instead of breadth-first manner
  - Saves memory in coordinator process