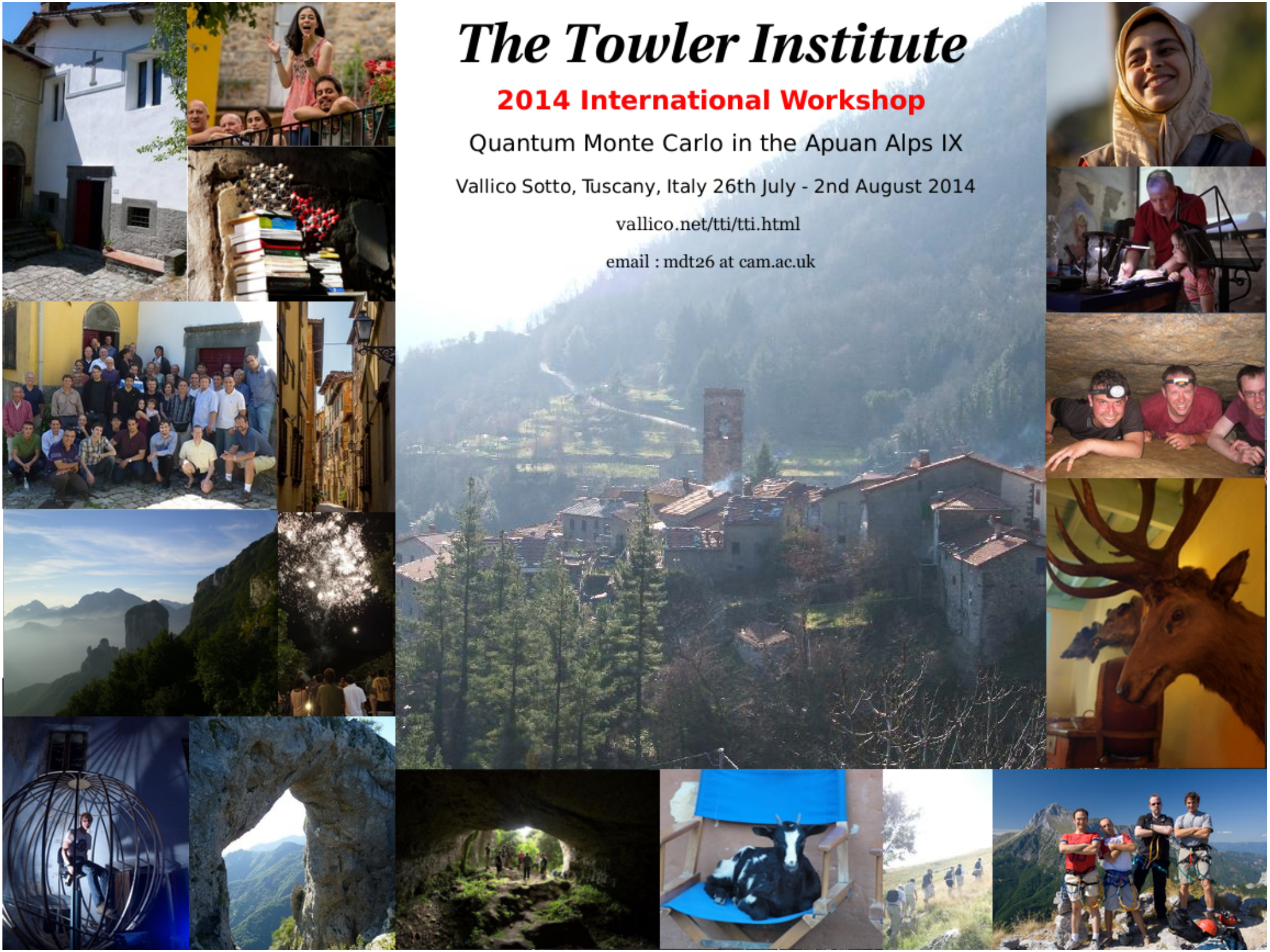
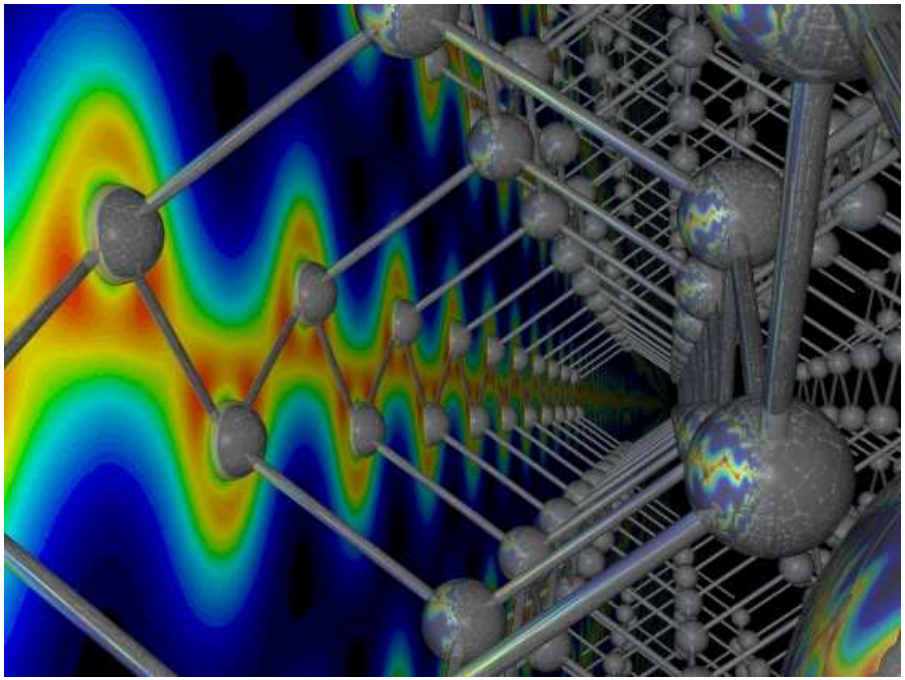

The Towler Institute
2014 International Workshop
Quantum Monte Carlo in the Apuan Alps IX
Vallico Sotto, Tuscany, Italy 26th July - 2nd August 2014
vallico.net/tti/tti.html
email : mdt26 at cam.ac.uk

High throughput QMC

Can we make CASINO more automatic..?



Mike Towler

TCM Group, Cavendish Laboratory, University of Cambridge and University College, London

QMC web page: `vallico.net/casinoqmc`

`mdt26@cam.ac.uk`

Quantum Monte Carlo in the Apuan Alps IX, 2014

Quote

"*But that's completely trivial..!*"

R.J. Needs, Ristorante Al Laghetto, 30 July 2014

Motivation I: Tim Mueller and other crazy people.

Slide borrowed from Tim Mueller's talk '*Quantum Monte Carlo for materials design*' here in Vallico Sotto last year:

Motivation II: HF/DFT codes

Wouldn't it be nice to be able to do this:

```
% vi silicon
```

```
TEST10 - SILICON BULK - BASIS SET 6-21 MODIFIED
CRYSTAL
0 0 0
227
5.42
1
14  .125  .125   .125
END
14 4
2 0 6  2.  1.
2 1 6  8.  1.
2 1 2  4.  1.
0 1 1  0.  1.
0.1233392     1.      1.
99 0
END
SHRINK
8 8
TOLDEE
9
END
```

```
% runcrystal silicon
% ls
silicon    silicon.o    silicon.w
% grep 'CYC' silicon.o
```

```
MAX NUMBER OF SCF CYCLES      50  CONVERGENCE ON DELTAP       10**-19
CYC   0 ETOT(AU) -5.784634809079E+02 DETOT -5.78E+02 tst  0.00E+00 PX  1.00E+00
CYC   1 ETOT(AU) -5.778137351820E+02 DETOT  6.50E-01 tst  0.00E+00 PX  1.00E+00
CYC   2 ETOT(AU) -5.778237340619E+02 DETOT -1.00E-02 tst  3.62E-03 PX  5.56E-02
CYC   3 ETOT(AU) -5.778259222757E+02 DETOT -2.19E-03 tst  7.80E-04 PX  1.84E-02
CYC   4 ETOT(AU) -5.778264771190E+02 DETOT -5.55E-04 tst  1.95E-04 PX  8.75E-03
CYC   5 ETOT(AU) -5.778266324836E+02 DETOT -1.55E-04 tst  5.13E-05 PX  5.75E-03
CYC   6 ETOT(AU) -5.778266800363E+02 DETOT -4.76E-05 tst  1.37E-05 PX  4.31E-03
CYC   7 ETOT(AU) -5.778266965130E+02 DETOT -1.65E-05 tst  3.68E-06 PX  2.72E-03
CYC   8 ETOT(AU) -5.778267029100E+02 DETOT -6.40E-06 tst  9.90E-07 PX  1.58E-03
CYC   9 ETOT(AU) -5.778267056625E+02 DETOT -2.75E-06 tst  2.67E-07 PX  8.77E-04
CYC  10 ETOT(AU) -5.778267069535E+02 DETOT -1.29E-06 tst  7.17E-08 PX  4.82E-04
CYC  11 ETOT(AU) -5.778267075872E+02 DETOT -6.34E-07 tst  1.96E-08 PX  2.59E-04
CYC  12 ETOT(AU) -5.778267078949E+02 DETOT -3.08E-07 tst  5.30E-09 PX  1.39E-04
CYC  13 ETOT(AU) -5.778267080513E+02 DETOT -1.56E-07 tst  1.46E-09 PX  7.44E-05
CYC  14 ETOT(AU) -5.778267081322E+02 DETOT -8.10E-08 tst  3.98E-10 PX  3.98E-05
CYC  15 ETOT(AU) -5.778267081724E+02 DETOT -4.02E-08 tst  1.10E-10 PX  2.12E-05
CYC  16 ETOT(AU) -5.778267081934E+02 DETOT -2.09E-08 tst  3.03E-11 PX  1.13E-05
CYC  17 ETOT(AU) -5.778267082040E+02 DETOT -1.06E-08 tst  8.38E-12 PX  6.04E-06
CYC  18 ETOT(AU) -5.778267082094E+02 DETOT -5.41E-09 tst  2.32E-12 PX  3.22E-06
CYC  19 ETOT(AU) -5.778267082122E+02 DETOT -2.84E-09 tst  6.46E-13 PX  1.73E-06
CYC  20 ETOT(AU) -5.778267082137E+02 DETOT -1.52E-09 tst  1.80E-13 PX  9.19E-07
CYC  21 ETOT(AU) -5.778267082145E+02 DETOT -7.57E-10 tst  5.01E-14 PX  5.11E-07
CYC  22 ETOT(AU) -5.778267082148E+02 DETOT -3.21E-10 tst  1.39E-14 PX  5.11E-07
== SCF ENDED - CONVERGENCE ON ENERGY      E(AU) -5.7782670821482E+02 CYCLES  22
```

The New Plan

So, let's make CASINO monitor the statistical error bar on the fly in QMC, run the calculation for as many moves as are necessary to reduce the error bar to a value which is 'small enough', whatever that might mean, then stop automatically.

It would also be nice if we could automatically figure out when the Metropolis and/or DMC equilibration has converged, so we don't have to specify the number of equilibration moves..

It isn't, or shouldn't be, rocket science..

But I'm not aware of any other codes that do anything like this..

Traditional way of running QMC calculations with CASINO

VMC

```
# RUN
runtype              : vmc              #*! Type of calculation (Text)
newrun               : T                #*! New run or continue old (Boolean)

# VMC
vmc_equil_nstep      : 1000             #*! Number of equilibration steps (Integer)
vmc_nstep            : 20000            #*! Number of steps (Integer)
vmc_nblock           : 1                #*! Number of checkpoints (Integer)
```

DMC

```
# RUN
runtype              : dmc              #*! Type of calculation (Text)
newrun               : T                #*! New run or continue old (Boolean)

# DMC
dmc_equil_nstep      : 1000             #*! Number of steps (Integer)
dmc_equil_nblock     : 1                #*! Number of checkpoints (Integer)
dmc_stats_nstep      : 10000            #*! Number of steps (Integer)
dmc_stats_nblock     : 10               #*! Number of checkpoints (Integer)
dmc_target_weight    : 100.d0           #*! Total target weight in DMC (Real)
```

In VMC and DMC we tell CASINO *how many samples* to take ('**nstep**' - divided up amongst all cores). In DMC also have an ensemble of configurations with a varying population of (on average) **dmc_target_weight** walkers, each independently carrying out it own random walk of **nstep** steps.

Will the error bar be small enough after nstep moves?

Nobody knows.. If it isn't, we *restart* from the saved state at the end of the previous run ('**newrun**=F') and continue for another **nstep** moves. Repeat as necessary.

VMC

```
# RUN
runtype          : vmc              #*! Type of calculation (Text)
newrun           : F                #*! New run or continue old (Boolean)

# VMC
vmc_nstep        : 20000            #*! Number of steps (Integer)
vmc_nblock       : 1                #*! Number of checkpoints (Integer)
```

DMC

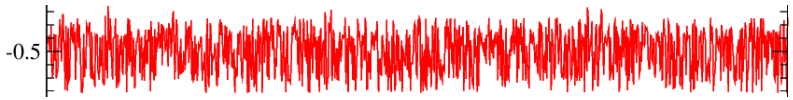
```
# RUN
runtype          : dmc_stats        #*! Type of calculation (Text)
newrun           : F                #*! New run or continue old (Boolean)

# DMC
dmc_stats_nstep  : 10000            #*! Number of steps (Integer)
dmc_stats_nblock : 10               #*! Number of checkpoints (Integer)
dmc_target_weight : 100.d0          #*! Total target weight in DMC (Real)
```

Not automatic!

Just to be clear: what is a 'block' in VMC?

Run is divided into **vmc_nblock** blocks of post-equilibration moves. Number of blocks determines how often the output, history and checkpoint files are written to disk. More specifically, at the end of each block:

- The processor- and block-averaged energies are calculated and a short 'report' are written to the *output file* ('`out`' in CASINO).
- The processor-averaged quantities for each step in the current block are appended to a *history file* ('`vmc.hist`' for energies, '`expval.data`' for everything else).



- Current VMC state plus any accumulated configs required for optimization or DMC written to a *checkpoint file* ('`config.out`' in CASINO) Configs created from 'snapshots' of sequential VMC run which are as widely spaced as possible..

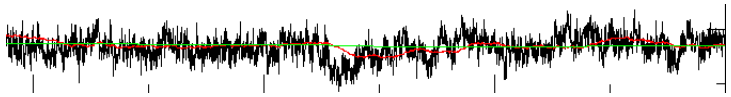
Note that:

- Total energy and error bar should be *independent* of **vmc_nblock** (though there are cases where error bar will differ, for reasons too boring to explain).
- People running CASINO often use *many* blocks, perhaps because they misunderstand what a block is. Frequent writing to disk only slows the code down - thus default **vmc_nblock**=1. Useful keyword: '**checkpoint**' - turns off writing checkpoint files, either completely or everywhere except end of run.

Just to be clear: what is a 'block' in DMC?

Pretty much the same as a VMC block, with the following differences:

- Data written to the history file ('`dmc.hist`' in CASINO) is averaged over the DMC ensemble (i.e. over configs) as well as processors.

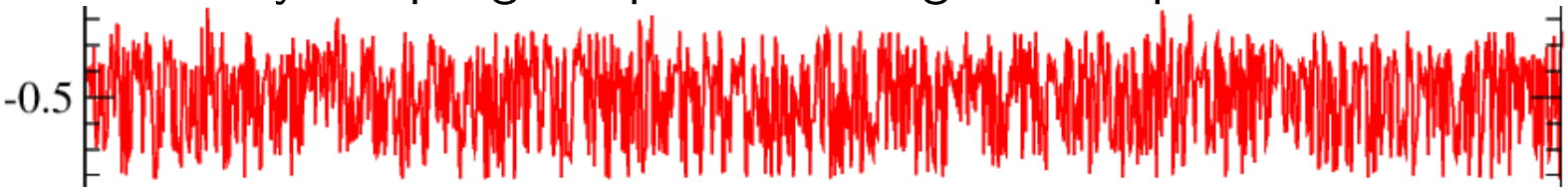


- If requested by the user, a backup copy of the checkpoint file can be made to allow for 'automatic catastrophe protection', which involves returning to the start of the previous block. This is the main reason for wanting to use multiple blocks in CASINO.
- As DMC calculations are much more likely to be cut short by time constraints, a *status file* ('`dmc.status`') is written at the end of each block containing what the final result would be if the calculation ended at that point. This is deleted and the information written to the output file at the end of the final block.

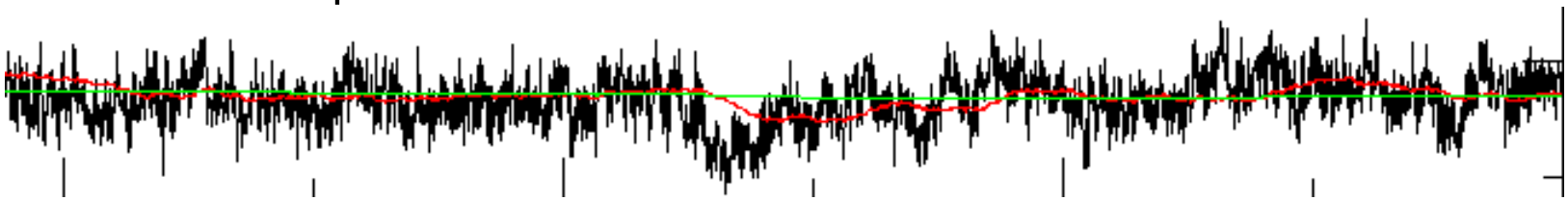
One important difference between VMC and DMC, which will be important for algorithmic purposes, is that in DMC the energies are averaged over processors at the end of every *move* but in VMC only at the end of every *block*. Clearly if we want to do a real time analysis of the error bar so we know when to stop, this is going to make VMC harder to handle. For once...

Basic quantities

Thus a QMC calculation produces potentially *millions* of data values, e.g. total energies obtained by sampling the particle configuration space.



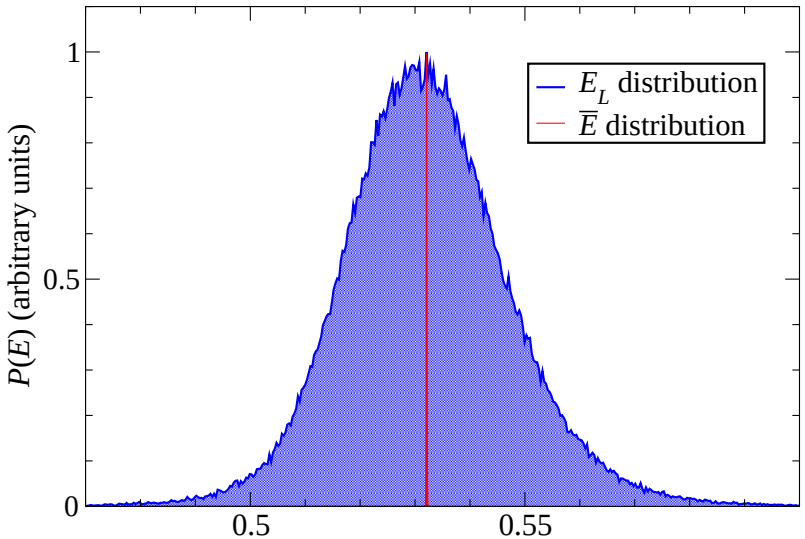
The N data values may be serially correlated, especially in DMC where we must use much smaller timesteps. Need to account for this.



Data have a *sample variance* S^2 ('width of the oscillations'). Clearly, estimate of this becomes more accurate over time and approaches non-zero constant value.

From this want single number with *error bar*: the mean energy $\bar{E}_N \pm \sigma$ where σ is the *standard error of the mean* given by $\sqrt{S^2/N}$.

The error bar therefore decreases as the *square root of the number of samples*.

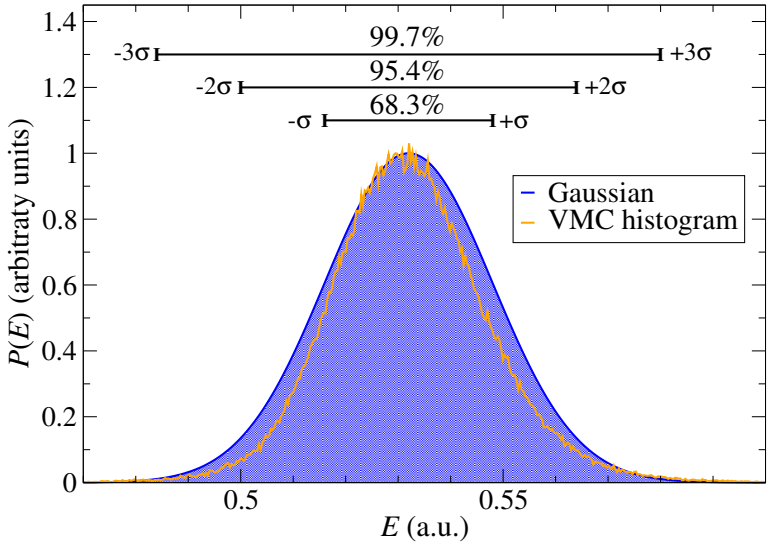


What does the error bar mean?

Let's say we've done a VMC calculation, and we get the answer $\bar{E}_N$ with an error bar of $\sigma = 0.001$. What can we conclude from this?

Definitions:

- Can define a confidence interval in this context as some range of energy, centred on your calculated mean energy $\bar{E}_N$, spanned by e.g. $\pm\sigma, \pm 2\sigma$ etc.. (in principle different for repeated runs).
- Then, if you construct many such confidence intervals, from repeated VMC runs with different random numbers, the proportion of these intervals that contain the true mean energy E is the confidence level.
- In QMC, usual to assume validity of central limit theorem (CLT), whereby the mean energies of repeated calculations are distributed *normally*, and the confidence levels are thus $0.683, 0.954, 0.997, \ldots$ for intervals of $\pm\sigma, \pm 2\sigma, \pm 3\sigma, \ldots$
- A statistically valid confidence level is one where numbers like this are actually true! Note that if so, we can just *write down* the approx. number of energy samples required to obtain a target error bar.
- Assumption that the mean energies obey a normal distribution is of course an approximation (see figure).



How well is the central limit theorem obeyed in CASINO?

The CLT is true in the large-N limit for a limited class of distributions, and so we have two non-trivial questions:

- Has the large-N limit been reached?
- VMC local energies one of the 'limited class of distributions'?

Noted by various people – see esp. Trail,Phys. Rev. E **77**, 016703 (2008) – that the local energy distribution is clearly not Gaussian, for both VMC and DMC calculations. 'Heavy-tailed' due to e.g. nodal singularities..

For most applications of QMC only single estimates are constructed, with an estimated random error calculated using the CLT. Generally, no ensemble of estimates is calculated to justify that this error is normal. Best we can do is observe that for many published results the estimated total energies and errors are consistent with exact energies where these are known in that they are higher (to within the statistical accuracy suggested by the CLT).

Trail: in general for the total energy the CLT is found to be valid in its weakest form, in that the influence of finite sample size is not obvious and must be considered on a case by case basis. Outliers significantly more likely than CLT.

For estimates of the variance in general the CLT is found to be less valid. Possesses slowly decaying tails so outliers are many orders of magnitude more likely than CLT.

Automatic QMC: problems to solve

- How to make selection of block length automatic?
- How to reliably correct error bars 'on the fly' for serial correlation, including through restarts.
- How to stop calculation when we achieve a certain error bar (this is clearly not as easy as 'stopping the first time the error bar fluctuates below a fixed target..')
- How to stop the calculation wasting resources if people request a target error bar that is 'too small'. Requires estimation of how many more samples (and hence how much more time) will be required in the future to achieve target, plus some definition of what a 'reasonable time' would be.
- Improving the error bar below a certain level will eventually be 'statistically meaningless'. How to work out what that means in practice?
- How to figure out whether 'equilibration' has been achieved (in VMC or DMC) so that the user doesn't need to specify the number of equilibration moves.?

Problem 1: blocks

- For doing actual science, we are mainly interested in DMC calculations.
- In DMC the main purpose of a block is to implement catastrophe recovery in which case a block defines '*how much time I am willing to waste if the calculation goes wrong*', and checkpointing - so that I don't have to do everything again if the system kills my jobs because it's run out of time.
- It also has a secondary meaning, related to the impatience of the user; how often does he need reassuring that the calculations is proceeding according to plan by something being written to the output file..
- These are all measures of *time*. It is not clear *a priori* how time is related to 'number of moves in a block' (this is a function of system size, basis set, etc..).

So why not just base the block length on the time?

This will also help with automatic stopping methods, which sometimes need access to information computed only at the end of a block. We must therefore be able to control directly the time per block.

Solution 1: blocks

New keyword: **block_time**

"If **block_time** greater than 0.d0, then the number of blocks of moves implied by **vmc_nblock**, **dmc_equil_nblock**, or **dmc_stats_nblock** will be ignored. Instead, CASINO will do everything it normally does at the end of a block approximately every **block_time** seconds of CPU time."

Relatively straightforward to implement. Only things requiring thought:

- Method of computing final energy/error bar from average of block averages and block error bars previously assumed all blocks were of the same length. Variable length blocks a significant extra complication needing a fair bit of rewriting.
- Since parallel VMC processes don't communicate except at the end of a block, the elapsed CPU time needs to be measured on the *master process* and the slave processes then ordered to finish their blocks when **block_time** is exceeded on the master. If the slaves have not yet done the same number of steps as the master, they carry on until they have. If they've done more, the extra steps are discarded and the calculation 'rewound' a little bit on the corresponding slaves.
- This 'rewinding' introduces indeterminancy in the random number generator (unless I do some fairly complex modifications) meaning that the error bar on repeated identical VMC calculations will no longer be exactly reproducible.

Result 1: block_time = 10.0 s

```
Starting VMC.

================================================================
In block : 1

Acceptance ratio <level 1>        (%)  =  58.8099
Acceptance ratio <levels 1-2>     (%)  =  50.3455
Diffusion constant           (Bohr^2)  = 5.7034E-02
Correlation time               (steps)  = 2.0500E+00 +- 1.2324E-01
Efficiency               (au^-2 s^-1)  = 1.7230E+02
No. of VMC steps per process           = 667

 Total energy                      (au) =        -62.187862549370
 Standard error                    +/-            0.016612705108

Time taken in block    : : :      10.0500

================================================================
In block : 2

Acceptance ratio <level 1>        (%)  =  58.7022
Acceptance ratio <levels 1-2>     (%)  =  50.2266
Diffusion constant           (Bohr^2)  = 5.7451E-02
Correlation time               (steps)  = 2.0367E+00 +- 1.2485E-01
Efficiency               (au^-2 s^-1)  = 1.7089E+02
No. of VMC steps per process           = 667

 Total energy                      (au) =        -62.136108141186
 Standard error                    +/-            0.016465392650

Time taken in block    : : :      10.0100


================================================================
In block : 3

Acceptance ratio <level 1>        (%)  =  58.7105
Acceptance ratio <levels 1-2>     (%)  =  50.2907
Diffusion constant           (Bohr^2)  = 5.7503E-02
Correlation time               (steps)  = 1.8523E+00 +- 1.0509E-01
Efficiency               (au^-2 s^-1)  = 1.8940E+02
No. of VMC steps per process           = 666

 Total energy                      (au) =        -62.177971729737
 Standard error                    +/-            0.017131868820

Time taken in block    : : :      10.0000
```

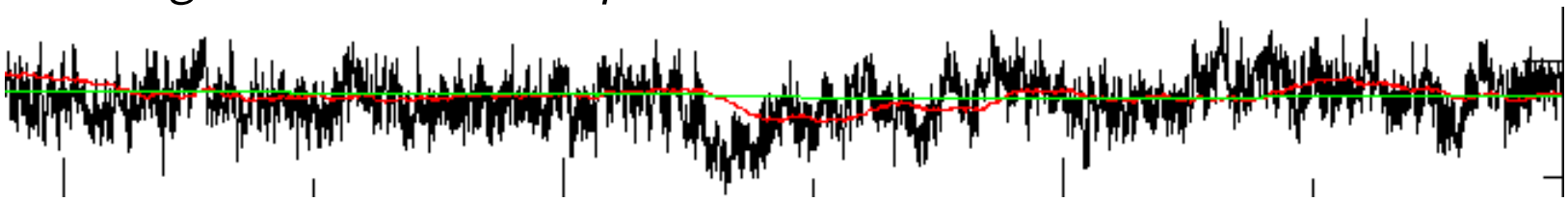
```
BEGIN DMC CALCULATION
=====================

EBEST = -7.47309826732421 (au/prim cell inc. N-N)
EREF  = -7.47309320322011

Number of previous DMC stats accumulation moves : 0

============================================================
In block : 1

Number of moves in block                 : 7014
Load-balancing efficiency (%)            : 97.853
Number of config transfers               : 379
Acceptance ratio (%)                     : 99.830
New best estimate of DMC energy (au)     : -7.47774278
Max no of attempts before accept move    : 3
New best estimate of effective timestep  : 0.00299319
Maximum distance from origin (au)        : 11.75304501

Time taken in block    : : :      10.1200

------------------------------------------------------------
In block : 2

Number of moves in block                 : 6970
Load-balancing efficiency (%)            : 97.580
Number of config transfers               : 345
Acceptance ratio (%)                     : 99.827
New best estimate of DMC energy (au)     : -7.47831676
Max no of attempts before accept move    : 3
New best estimate of effective timestep  : 0.00299317
Maximum distance from origin (au)        : 12.58361757

Time taken in block    : : :      10.1000

============================================================
In block : 3

Number of moves in block                 : 7011
Load-balancing efficiency (%)            : 98.159
Number of config transfers               : 353
Acceptance ratio (%)                     : 99.823
New best estimate of DMC energy (au)     : -7.47810008
Max no of attempts before accept move    : 3
New best estimate of effective timestep  : 0.00299315
Maximum distance from origin (au)        : 13.37695200

Time taken in block    : : :      10.1200
```

No reason **block_time** > 0 can't be the default from now on.

Problem 2: corrections for serial correlation

To introduce a stop condition based on the error bar requires the error bar to be *accurate*. In general if we simply take the square root of the raw data variance over the number of samples it will of course be *too small*, particularly in DMC where we are forced to use smaller timesteps.

This is because *successive local energies are more similar on average than they would be if the configurations were independent*.



$$\{\underbrace{E_1,\ldots,E_1}_{\tau},\underbrace{E_2,\ldots,E_2}_{\tau},\underbrace{E_3,\ldots,E_3}_{\tau},\ldots,\underbrace{E_M,\ldots,E_M}_{\tau}\}$$

Energy sequence with artificial correlation as above has no new info compared to uncorrelated set $\{E_1, E_2, E_3, \ldots, E_M\}$. Mean and error bar should not change, but in fact computed error bar of new set is $\tilde{\sigma}'_{\bar{E}} = \tilde{\sigma}_{\bar{E}}/\sqrt{\tau}$. Error bar underestimated!

Here can remove serial correlation by ignoring $\tau - 1$ of every τ consecutive E_i. For real data, τ *varies* during the run and must ignore $\tau_{\max} - 1$ of each $\tau_{\max}$ data to be safe - lots of relevant data discarded. However we may assume that $\tilde{\sigma}_{\bar{E}} = \sqrt{\tau}\tilde{\sigma}'_{\bar{E}}$ where τ is the average correlation time. Have now made CASINO print this..

Solution 2: corrections for serial correlation - reblocking

Consider the following operation on data, where the item under each brace is the average of the two numbers above:

$$\underbrace{E_1^{(0)} \quad E_2^{(0)}} \quad \underbrace{E_3^{(0)} \quad E_4^{(0)}} \quad \underbrace{E_5^{(0)} \quad E_6^{(0)}} \quad \underbrace{E_7^{(0)} \quad E_8^{(0)}}$$

$$\underbrace{E_1^{(1)} \qquad\qquad E_2^{(1)}} \qquad\qquad \underbrace{E_3^{(1)} \qquad\qquad E_4^{(1)}}$$

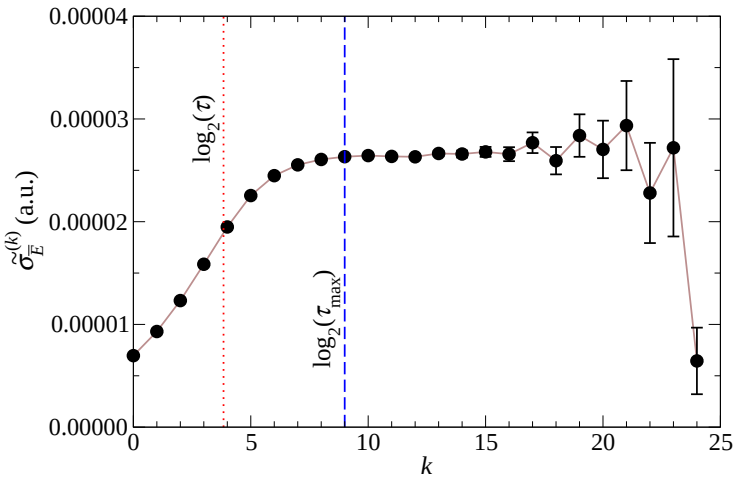
$$\ldots \qquad\qquad\qquad\qquad \ldots$$

If applied until $\tau_{\max}$ original data grouped together *resulting (smaller) data set is not serially correlated* though not we cannot compute $\tau_{\max}$ directly.

At the k-th iteration in this procedure:

$$\tilde{\sigma}_{\bar{E}}^{(k+1)2} \approx \tilde{\sigma}_{\bar{E}}^{(k)2} + \frac{2\sum_{i=1}^{M^{(k)}/2}\left(E_{2i-1}^{(k)} - \bar{E}\right)\left(E_{2i}^{(k)} - \bar{E}\right)}{M^{(k)}\left(M^{(k)} - 2\right)}$$

Last term tends to zero with no serial correlation (positive otherwise). $\tilde{\sigma}_{\bar{E}}^{(k)}$ increases toward *true error bar* as $k \approx \log_2(\tau_{\max})$. Plateau in $\tilde{\sigma}_{\bar{E}}^{(k)}$ signals convergence. Can be done 'on-the-fly' (already implemented by PLR). Only limitation - didn't work through restarts. Now it does.



Result 2: corrections for serial correlation

New CASINO output (silane, 2000000 moves then another 2000000):

```
================================================================================
FINAL RESULT:

 VMC energy (au)    Standard error      Correction for serial correlation

-6.299969221470 +/- 0.000565311037      No correction
-6.299969221470 +/- 0.000954119316      Correlation time method
-6.299969221470 +/- 0.000964564527      On-the-fly reblocking method

 Sample variance of E_L (au^2/sim.cell) : 0.638912952475 +- 0.032502473304

================================================================================
```

RESTART

↓

```
================================================================================
FINAL RESULT:

Restarted calculation. Include data from previous runs.

 VMC energy (au)    Standard error      Correction for serial correlation

-6.298869541299 +/- 0.000435284643      No correction
-6.298869541299 +/- 0.000738226000      Correlation time method
-6.298869541299 +/- 0.000774688796      On-the-fly reblocking method

 Sample variance of E_L (au^2/sim.cell) : 0.877993087814 +- 0.165833087205

================================================================================
```

Conclusion 2: corrections for serial correlation

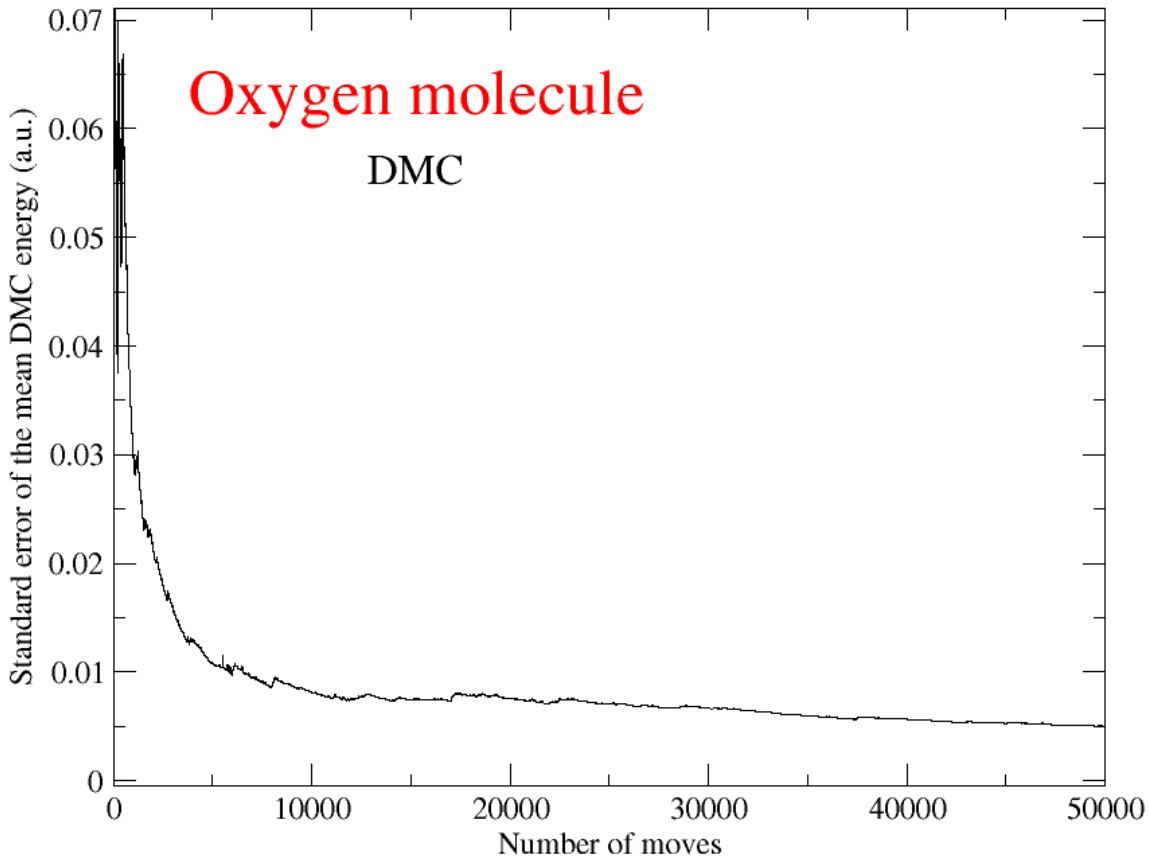
DMC

- With on-the-fly reblocking the processor-averaged energy data is added into the reblocked averages at the *end of every move* (i.e. parallel communicaton takes place after every move).
- Correlation time correction of error bar is not implemented (τ is very large, converges slowly, unnecessary cost).
- Thus have error bar corrected for serial correlation after *any arbitrary move* - with on-the-fly reblocking only. Enough info to impose stopping criterion, independent of users' arbitrary choice of block length. Implemented.

VMC - harder!

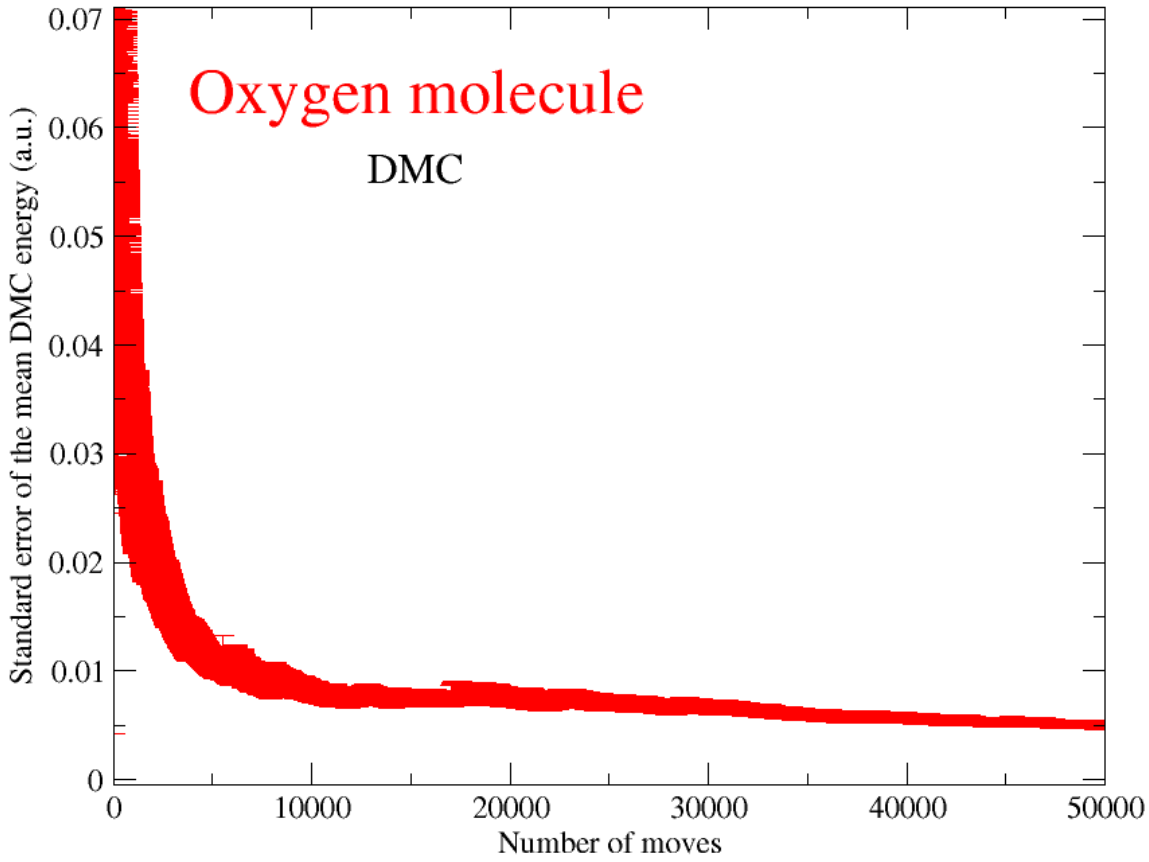
- Averaging over cores done at end of block. No parallel communication within blocks. Must assume user has set arbitrary **block_time** keyword to, e.g., 3 years thus currently corrected error bar *not* available after *any* move..
- From the user perspective, a block defines when the checkpoint file (`config.out`) is written, and when stuff is written to the main output file.
- Need to divorce 'summing energies over processors' from normal concept of a block. With target error bar stopping criterion, we need to add new time periodicity in VMC, e.g., `min[1 minute,` **block_time**`]`. Not yet implemented - ran out of time. Not difficult..

Problem 3: stop on target error bar



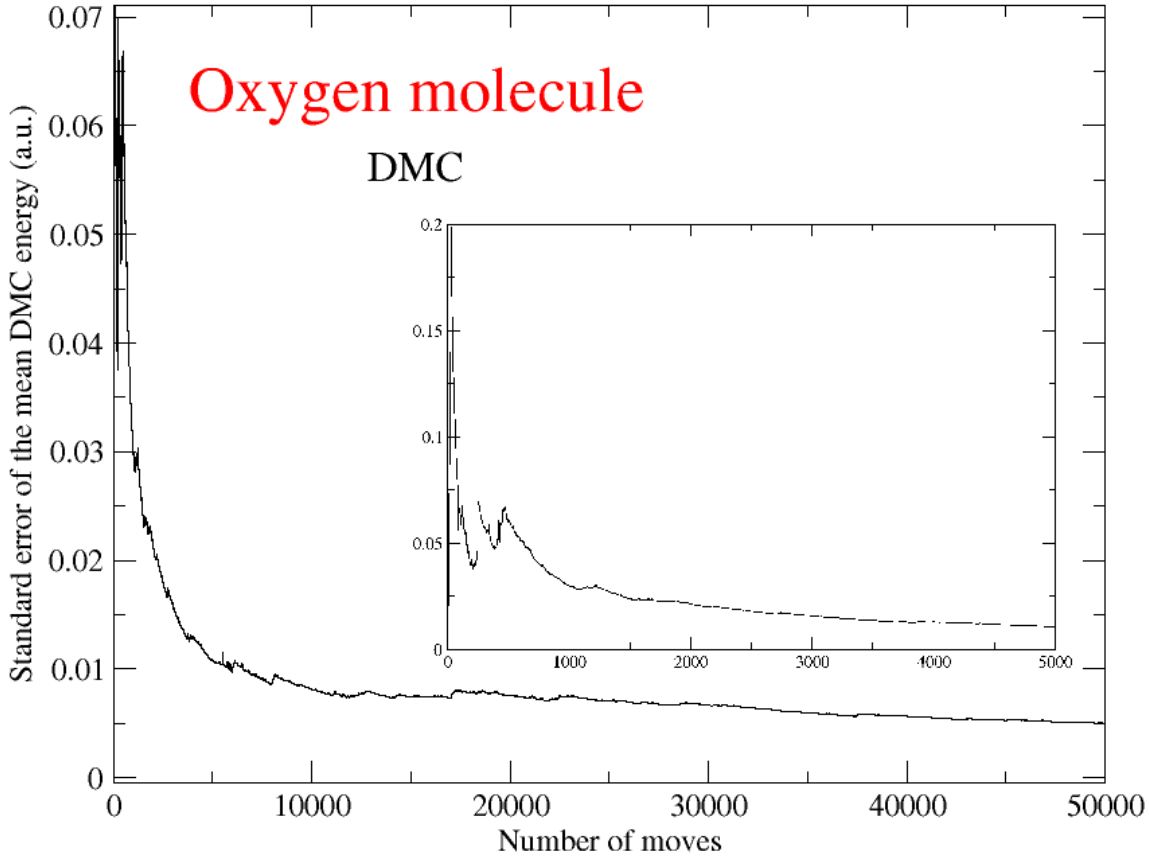
After an initial transient of 500-1000 moves, the error bar behaves nicely and decreases relatively smoothly with something like the expected $1/\sqrt{N}$.

Problem 3: stop on target error bar



After an initial transient of 500-1000 moves, the error bar behaves nicely and decreases relatively smoothly with something like the expected $1/\sqrt{N}$.

Problem 3: stop on target error bar



After an initial transient of 500-1000 moves, the error bar behaves nicely and decreases relatively smoothly with something like the expected $1/\sqrt{N}$.

How to control different stopping methods

New input keywords: **stop_method**, **target_error**, **stop_time**

- The **stop_method** keyword defines how a VMC/DMC run is to be terminated. It may take the values '`nstep`', '`target_error`', or '`small_error` '.
- The classic method is '`nstep`' which means simply perform the number of VMC/DMC steps implied by the input keywords **vmc_nstep**, **dmc_stats_nstep** then stop, and the error bar is what it is (it may be too large or smaller than required).
- If **stop_method** = '`target_error`' then the run will continue until the error bar on the total energy (corrected on the fly for serial correlation) is approximately equal to that defined by the **target_error** input keyword, subject to the constraint that the *estimated* CPU time required on the master process (summed over restarts if necessary) will not exceed **stop_time**. CASINO is able to approximately estimate the required time by analyzing how the error bar decreases as a function of the number of moves, and as soon as it is reasonably confident that the desired target error is too small and cannot be reached, then the code will stop (in a restartable condition). On halting in this manner, an estimate of the CPU time required to get a range of error bars will be written the output file. Note that the method used to estimate the required time assumes the validity of the central limit theorem, which is only approximately valid in this case.
- If **stop_method** = '`small_error`', CASINO will attempt to make the error bar as small as possible in a 'reasonable time' defined by the value of the **stop_time** keyword. '*As small as possible*' means what it says, but taking account of the fact that there is an error bar on the error bar and it is somewhat pointless to reduce the error bar below its significant precision.
- Note in both the last two cases CASINO has a minimum run length needed to get a reasonable estimate of the variance.

Naive DMC implementation

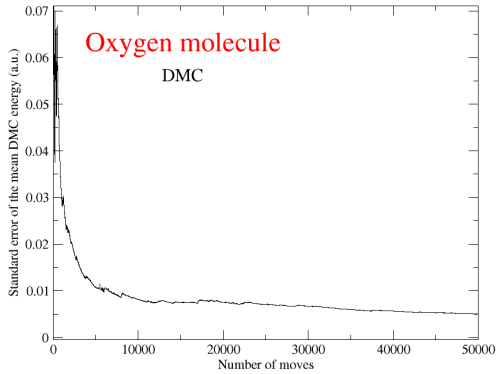
"*Stop as soon as the on-the-fly-reblocked error bar goes below* **target_error**."

stop_method=`target_error`, **target_error**=`0.005`, **stop_time**=`1 day`

```
================================================================================
In block : 6

Number of moves in block                 : 4001
Load-balancing efficiency (%)            : 96.159
Number of config transfers               : 588
Acceptance ratio (%)                     : 99.458
New best estimate of DMC energy (au)     : -150.28423236
Max no of attempts before accept move    : 4
New best estimate of effective timestep  : 0.00198152
Maximum distance from origin (au)        : 7.01240537

Time taken in block    : : :       17.2000

================================================================================
TARGET ERROR ACQUIRED
Run termination at move 4001, block 6 ( 50039 total moves ).
Target error bar:   5.0000E-03
Actual error bar:   4.9999E-03 +/- 3.5946E-04
================================================================================
```

However, doing it like this potentially introduces bias. We have to be more careful..!

How to define proper stopping condition

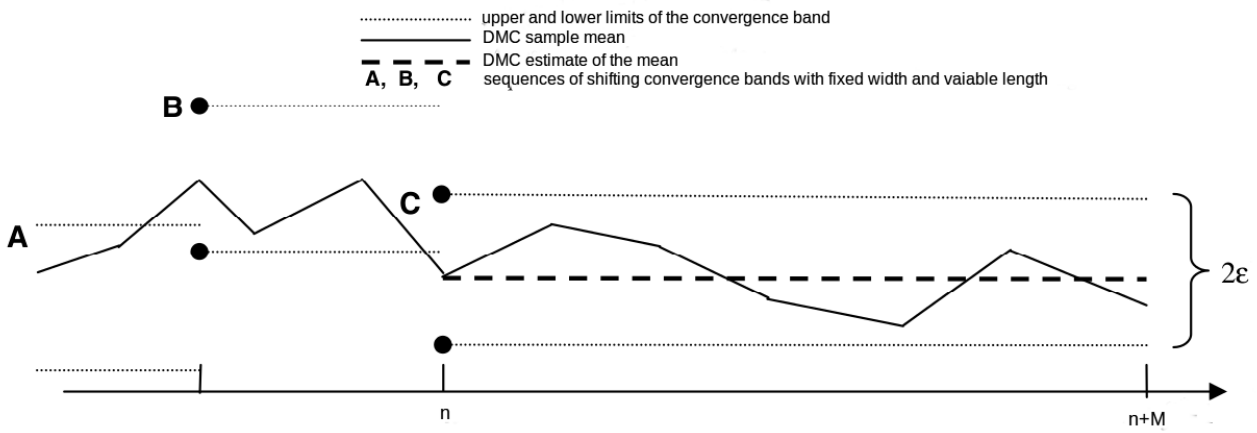


- Stopping when continuously-monitored error bar falls below target guarantees the error bar will be *underestimated* and the calculation halted too soon.
- This is because when the reblocked error bar falls below the threshold it is bound to be on a negative fluctuation below its mean, i.e., the error bar will be less than its underlying mean.
- Put another way, suppose you run a simulation and find that it halts after N steps because the error bar is smaller than some criterion. The reblocked error bar will be *significantly smaller on average* than the error bar you get if you perform a second, independent run of precisely N steps.
- Not clear *a priori* how important this is for a *general system*, but error bars on the error bars can be pretty large (see typical reblock plots) and possibly not well-defined due to the tails of the local-energy distribution.

Empirical convergence

- Usually use CLT to provide basis for assessing statistical error of approximation for large N.
- Accuracy depends on rate at which actual distribution converges to target distribution.
- Without any information on convergence, the approximation to normality is an additional source of error for any finite sample size.

However, if the upper and lower bounds are known for a random variable, this error of approximation to normality can be eliminated using a distribution-free technique. One such way is to look for a '*convergence band*' (CB) of a given width and length such that the probability of the QMC sample means to fall outside of this band is 'practically zero'. In this context the width is *twice the target error* and the length is an appropriate number of moves to ensure probability is in fact practically zero.



The convergence band

We are requiring that the range of fluctuating mid points of actual confidence intervals of the DMC sample mean energy, has reduced to an acceptable value which is certainly smaller than $\pm$**target_error**$=\epsilon$.

Construct sequential interval always covering the DMC sample means as follows:

- When the error bar first fluctuates below **target_error**, 'turn on' the convergence band, centred at the current best estimate of the DMC energy $E_{\rm DMC}$.
- Keep executing DMC moves. If the new DMC energy is in the range $E_{\rm DMC} \pm \epsilon$ then increment a counter M_{CB} and continue. Otherwise, revert the M_{CB} counter to zero, and recentre the convergence band at $E_{\rm DMC} + \epsilon$.
- Continue until M_{CB} equal to *convergence band length*, at which point stopping criterion for **stop_method**=`target_error` is attained..

What is the optimal value of the convergence band length?

- Define set of bins for convergence band lengths 1 to 100 (say). Every time mean energy fluctuates outside convergence band, increment counter for M_{CB}th bin.
- Determine empirically, averaged over a range of already equilibrated systems.
- Convert bin counters into probability (will be decreasing function of M_{CB}). Find where probability effectively zero (the *convergence band length*). About 50 seems to be good value (more investigation required; timestep dependence, etc...).

Time-proofing

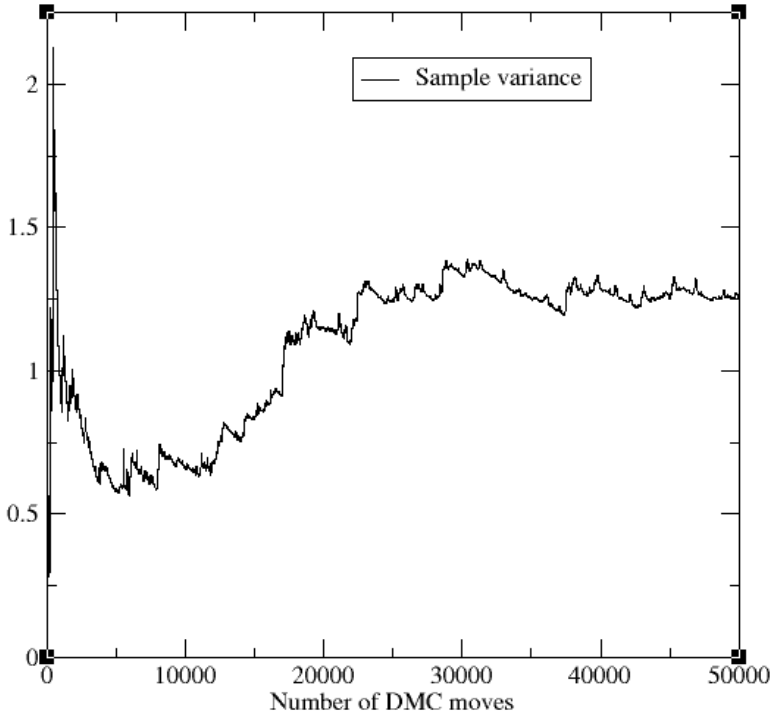
- Say it takes 100 seconds to reach **target_error**=0.001. To achieve 0.0001 we would expect 10000 seconds (nearly 3 hours - reasonable), and to achieve 0.00001 would require around 1000000 seconds (nearly 2 weeks - probably not reasonable).
- The definition of 'reasonable' is now handled by the new **stop_time** keyword.
- Hence introduce new procedure to stop as quickly as possible (in a restartable condition) if target error bar is unreasonable, with a suggestion for what target errors are possible to achieve.
- Formula for number of required moves to get error bar = **target_error**:

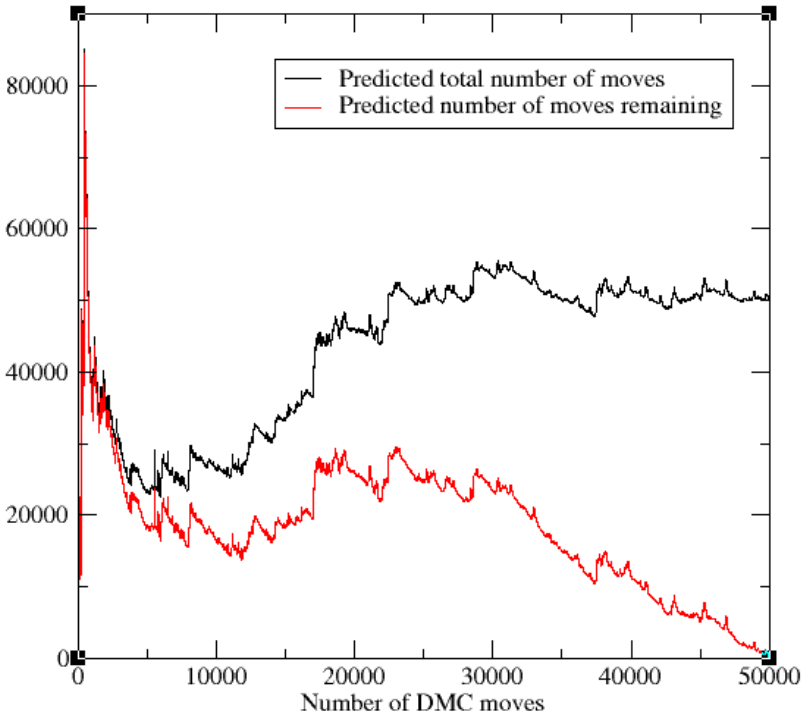
$$N = z_{C/2}^2 \times \frac{\text{sample variance}}{\text{target\_error}^2}$$

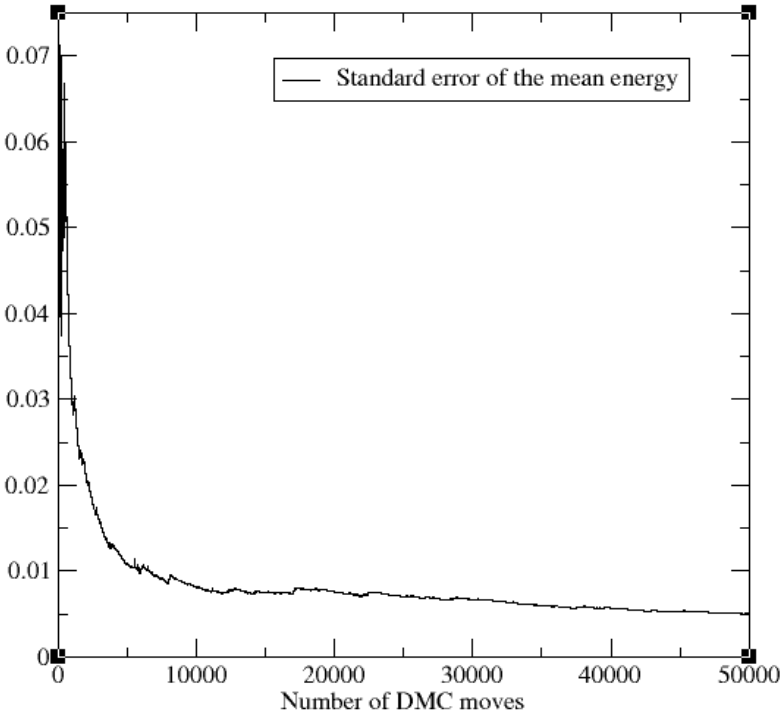
where C = confidence level (e.g. 0.9), and $z_{C/2}$ is the inverse normal distribution, which maps e.g. 0.68 onto 1 (sigma) and 0.99 onto 2 (sigma) etc. We usually set this to 1.

- Can't do this at the start because we don't know the sample variance.

How long does it take sample variance to behave?







A lot longer than the error bar!

Time proofing

Now implemented. Bit inaccurate, but obviously useful..

```
======================================================================
In block : 5

Number of moves in block               : 9095
Load-balancing efficiency (%)          : 95.679
Number of config transfers             : 1347
Acceptance ratio (%)                   : 99.460
New best estimate of DMC energy (au)   : -150.28423292
Max no of attempts before accept move  : 4
New best estimate of effective timestep : 0.00198152
Maximum distance from origin (au)      : 7.16089277
Average time per move (sec)            : 0.0044
Number of moves to target error        : 5514 +/- 339
Approx. CPU time to target error (sec) : 24.0320

Time taken in block    : : :      40.1600

======================================================================
In block : 6

Number of moves in block               : 4001
Load-balancing efficiency (%)          : 96.159
Number of config transfers             : 588
Acceptance ratio (%)                   : 99.458
New best estimate of DMC energy (au)   : -150.28423236
Max no of attempts before accept move  : 4
New best estimate of effective timestep : 0.00198152
Maximum distance from origin (au)      : 7.01240537
Average time per move (sec)            : 0.0044
Number of moves to target error        : 0
Approx. CPU time to target error (sec) : 0.0 sec

Time taken in block    : : :      17.2000

======================================================================
TARGET ERROR ACQUIRED
Run termination at move 4001, block 6 ( 50039 total moves ).
Target error bar:   5.0000E-03
Actual error bar:   4.9999E-03 +/- 3.5946E-04
======================================================================
```

CASINO reports 'Insufficient data' until the variance starts behaving itself.

Implementation of stop_method = 'small_error'

NO TIME!

Just run the calculation for **stop_time** seconds/minutes/days, and monitor that the error bar is statiscally meaningful.

Automatic detection of equilibration

NO TIME

Implementation

NO TIME

Results for equilibration

NO TIME

Other ways of achieving high throughput

Development of 'Recipe' which can be scripted for simplified high-throughput workflow.

- Input file (better defaults for keywords).
- Basis set
- Pseudopotentials
- Finite-size effects (twist averaging, etc).
- Jastrow factor and optimization
- Zero-point energies

Scripts must check for problems..

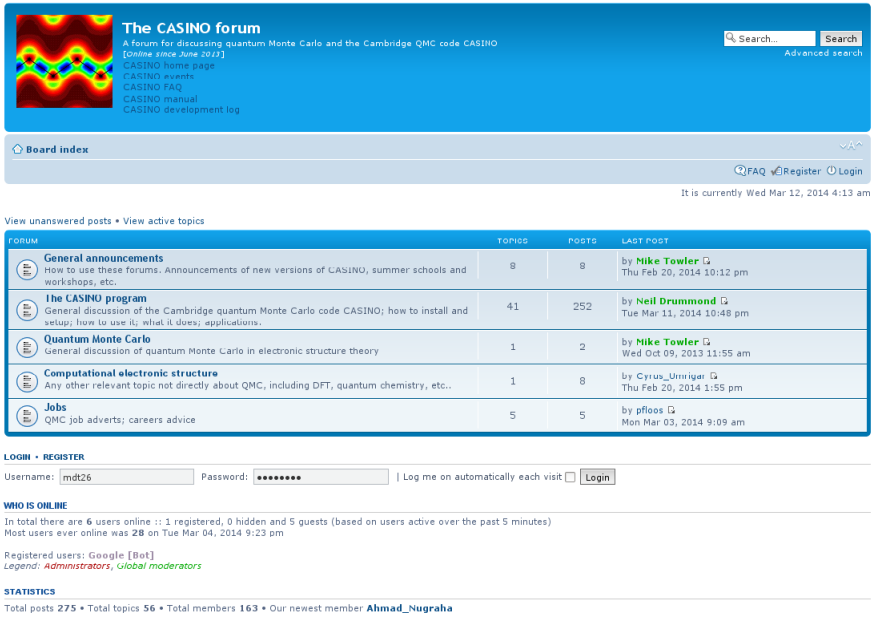
Thought required!

Conclusions

- In roughly about two weeks when I publish the all new CASINO version 2.14, there will be the option to run calculations automatically so that they either achieve a desired error bar or warn that they can't.
- This should greatly improve the rate at which calculations of large datasets can be done.
- Much more work to be done to make QMC as automatic as e.g. DFT calculations..

Websites

Some community tools for interacting QMC researchers once you've all gone home..



QMC site: `vallico.net/casinoqmc`

- Any registered user can make posts or create content.
- People with sufficiently elevated privileges can edit virtually anything.
- Database of papers with links (add yours..)

QMC forum: `vallico.net/casino-forum`

- Discussion forum and source of help for CASINO
- Various other general discussion forums.
- Can add others on request..

Final slide of the conference

THANKS FOR COMING TO VALLICO SOTTO!