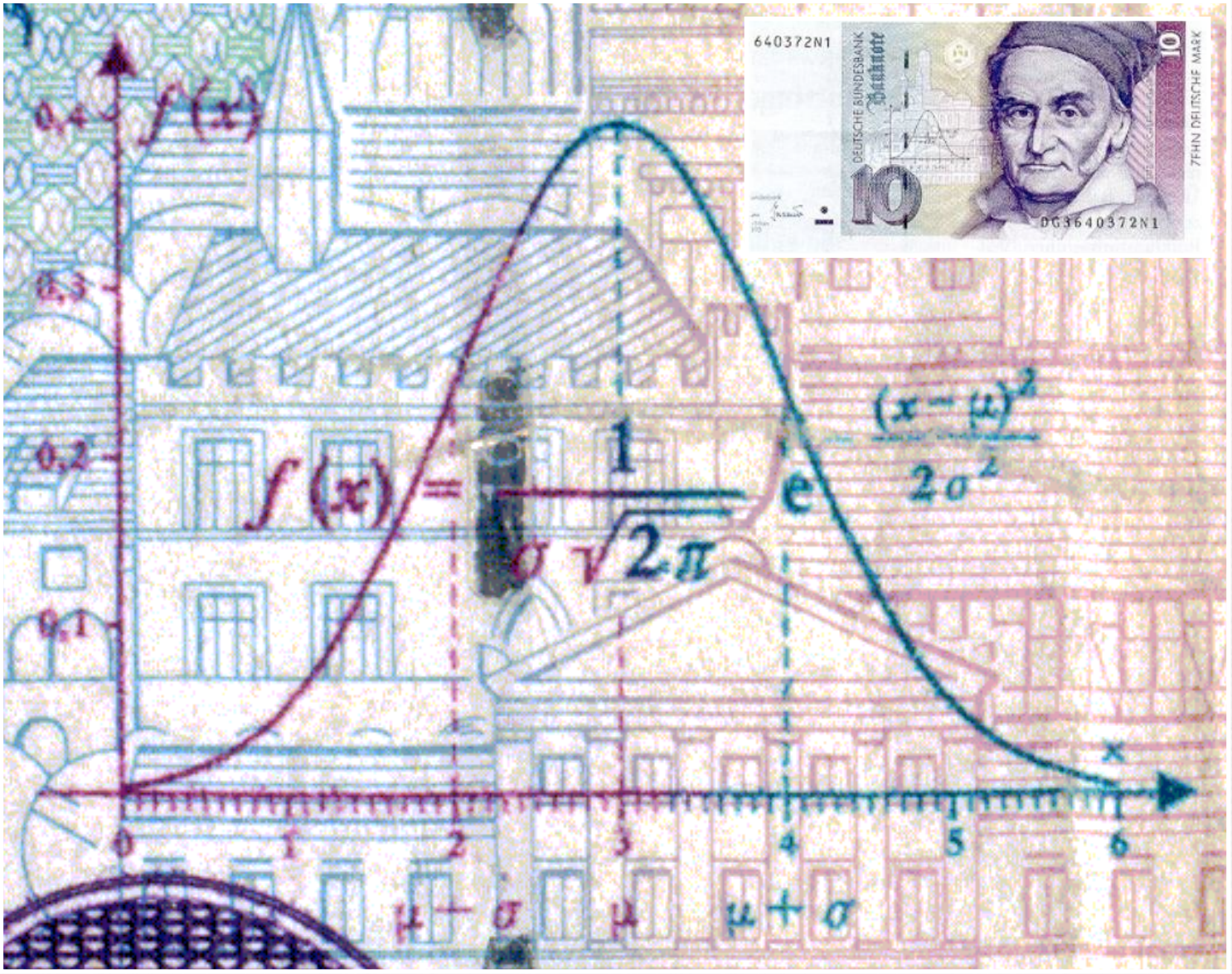
Econophysics – Theorie und Praxis

Tobias Preis

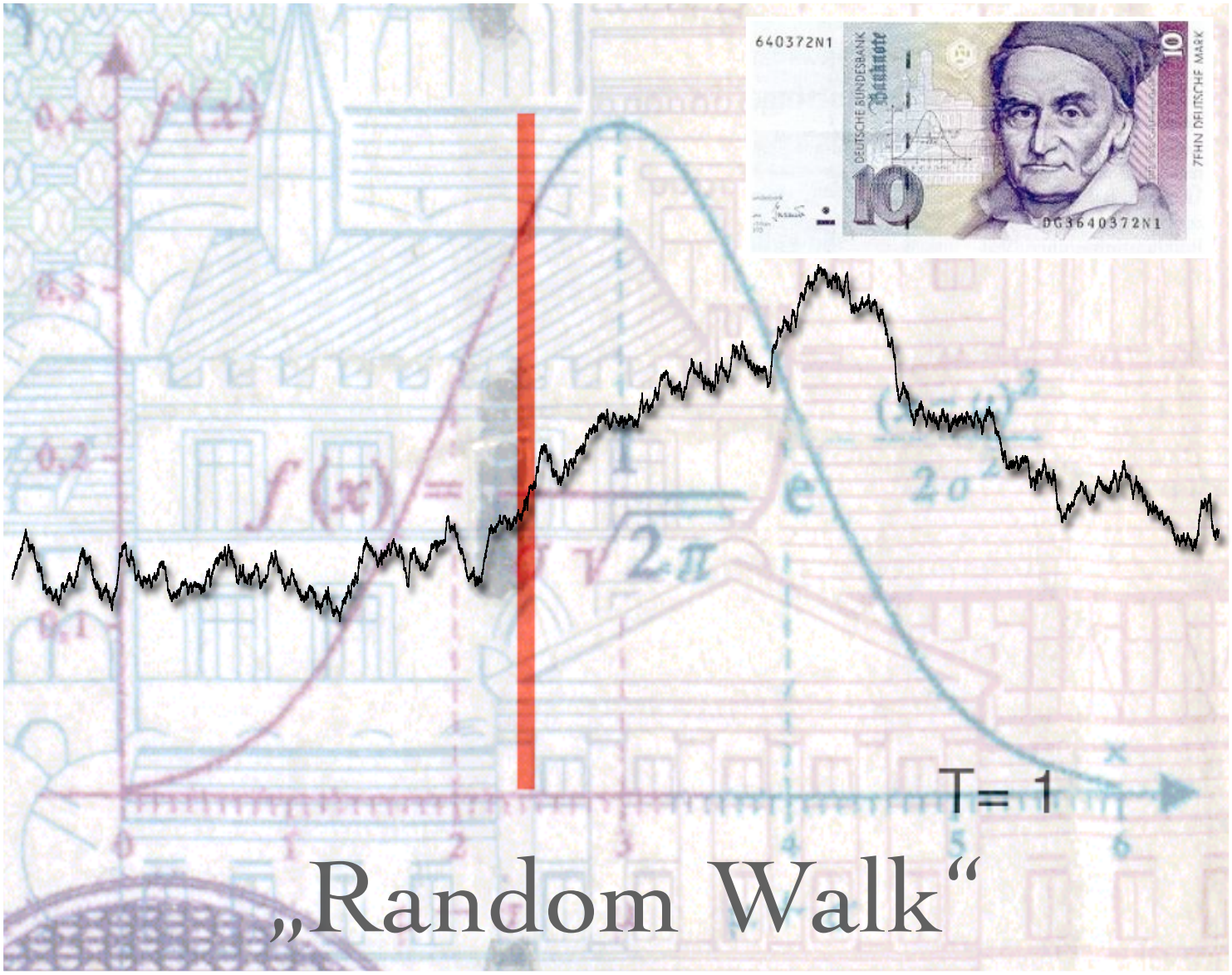
Institute of Physics – Johannes Gutenberg-University Mainz
Center for Polymer Studies – Boston University
Artemis Capital Asset Management GmbH
preis@uni-mainz.de

http://www.tobiaspreis.de

Tobias Preis



Tobias Preis



„Random Walk“

Tobias Preis

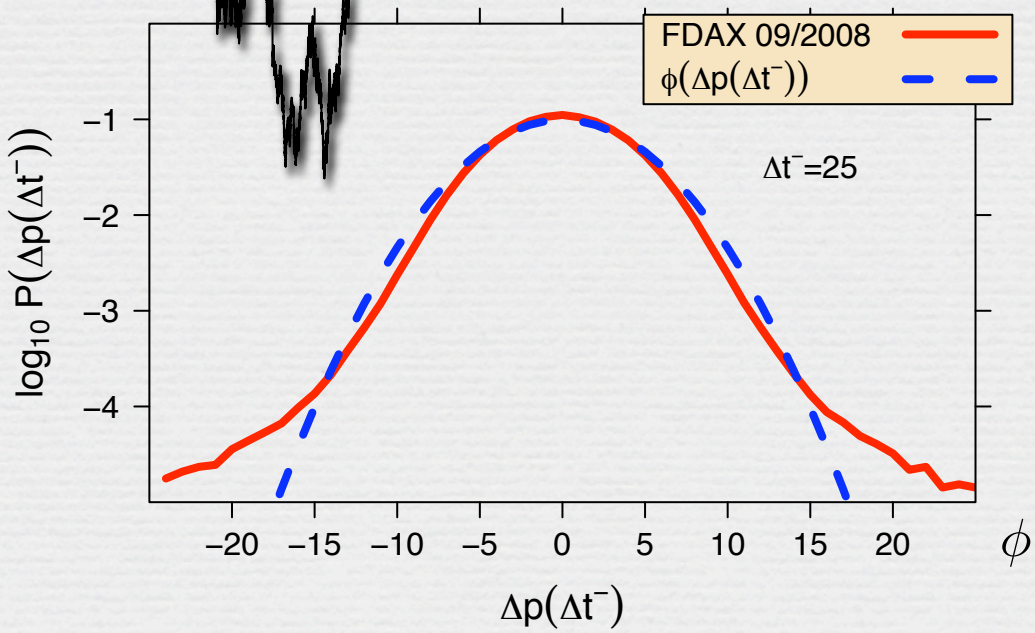
German Stock Index (Dax)

20.06.2008

15.09.2008

LEHMAN BROTHERS

19.09.2008

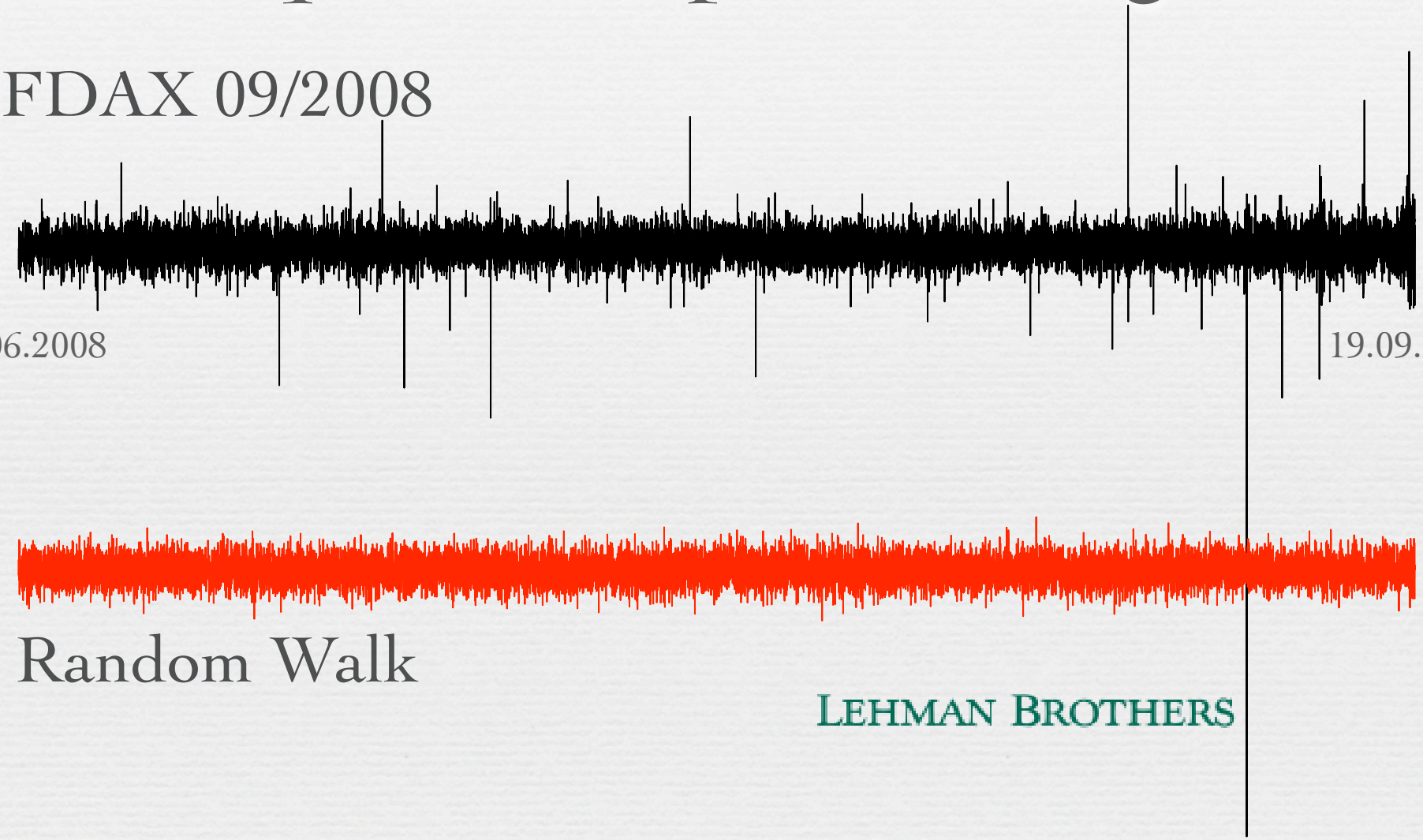


$$\phi(\Delta p(\Delta t^-)) = u \exp(-v \Delta p^2)$$

Tobias Preis

Sequence of price changes

FDAX 09/2008



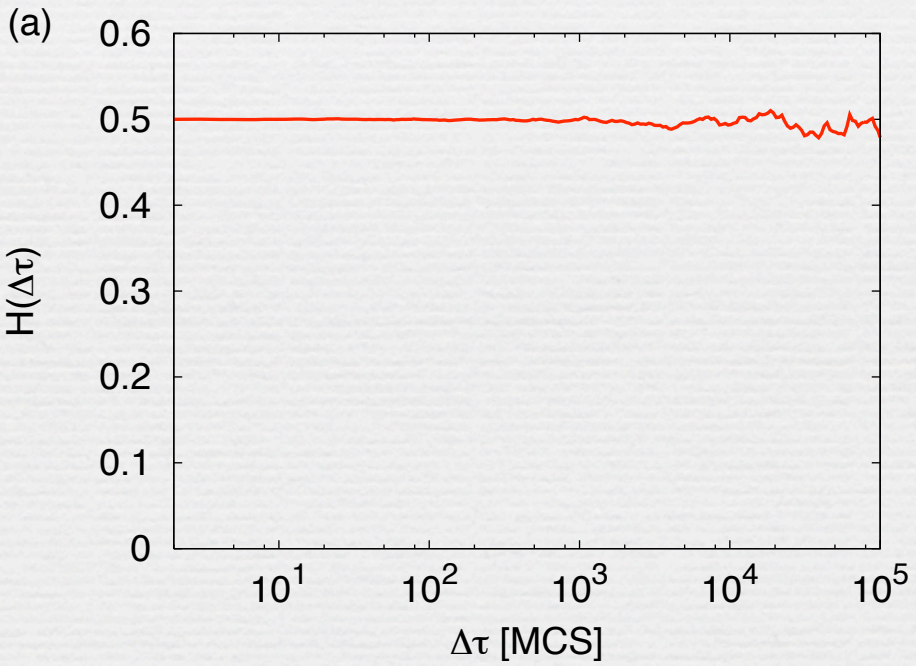
Tobias Preis

Hurst exponent

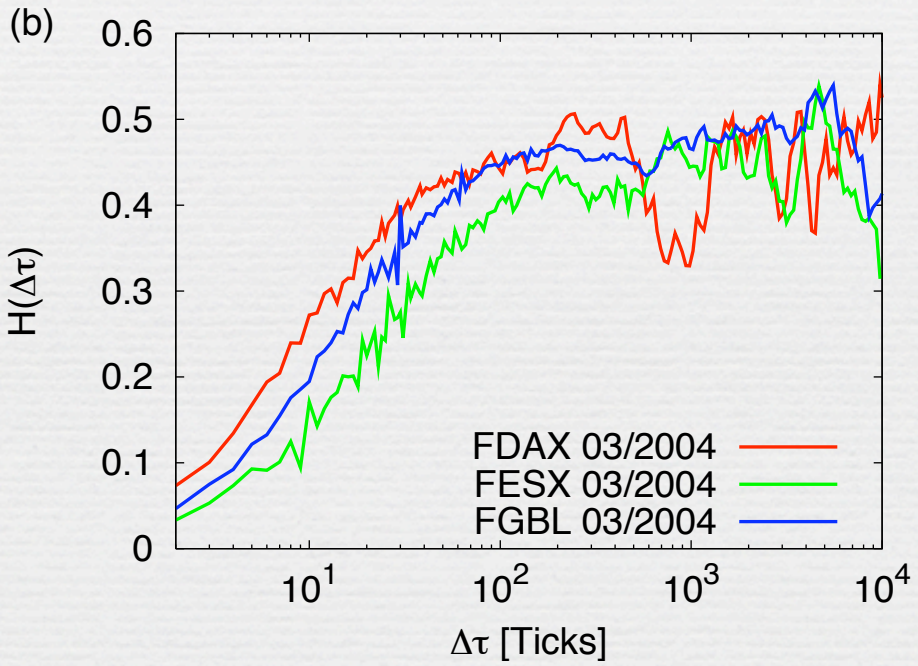
- Introduced by H. E. Hurst 1951, water levels of Nile
- Information about the scaling behavior of a process:
  - ➡ H=0.5 : diffusive regime, RW
  - ➡ H>0.5 : persistent behavior
  - ➡ H<0.5 : anti-persistent behavior
- As recently shown (McCauley et al., Physica A 369, **343** (2006)) H>0.5 implies not necessarily long time correlations.

Hurst exponent

- RW



- Market data



- In trend phases H>0.5 on medium time scales

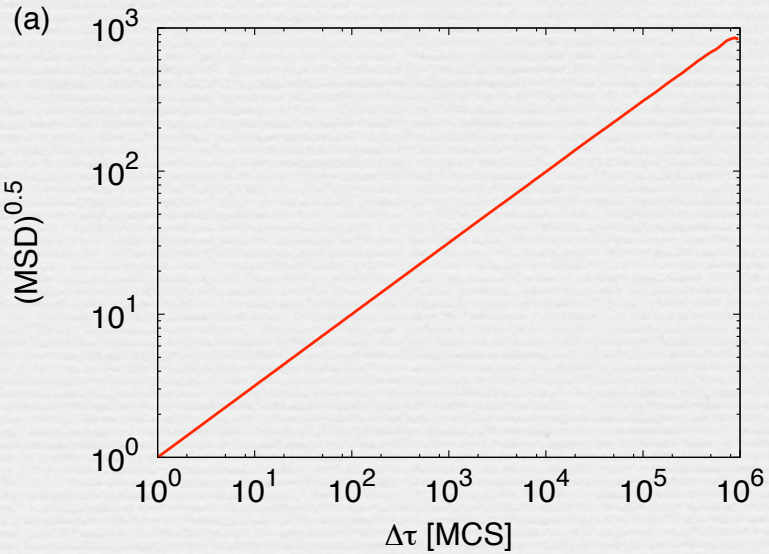
Hurst exponent

- Calculation methods ...
  - ➡ Detrended fluctuation analysis (Ausloos, Physica A **285**, 48 (2000))
  - ➡ Wavelet transformation (Simonsen et al. Phys. Rev. E **58**, 2779 (1998))
  - ➡ R/S Analysis (Hurst, Transactions of the American society of civil engineers **116**, 770 (1951))

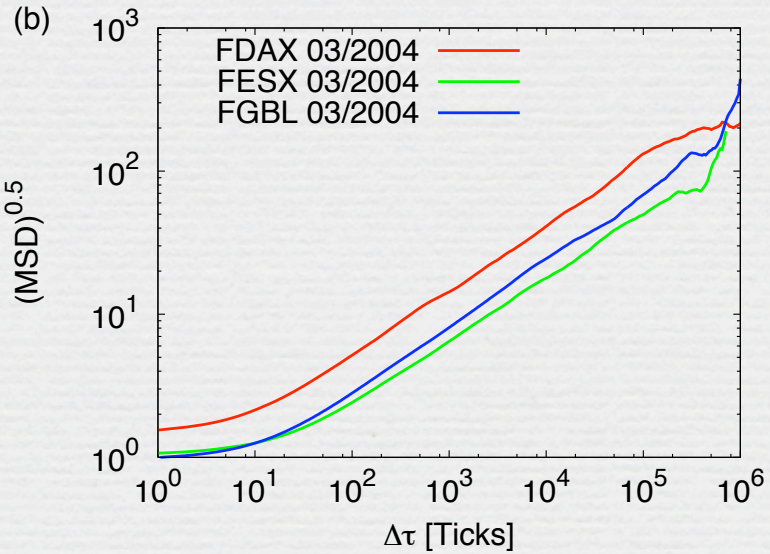
Hurst exponent

- ... or calculation with the mean square displacement (MSD)

$$\mathrm{MSD}\,(\Delta\tau) = \langle(\Delta \mathrm{p})^2\rangle_\mathrm{t} \qquad \text{with} \qquad \Delta p\,(t, \Delta\tau) = p\,(t + \Delta\tau) - p\,(t)$$



- RW



- Market data

Hurst exponent

- Hurst exponent H:

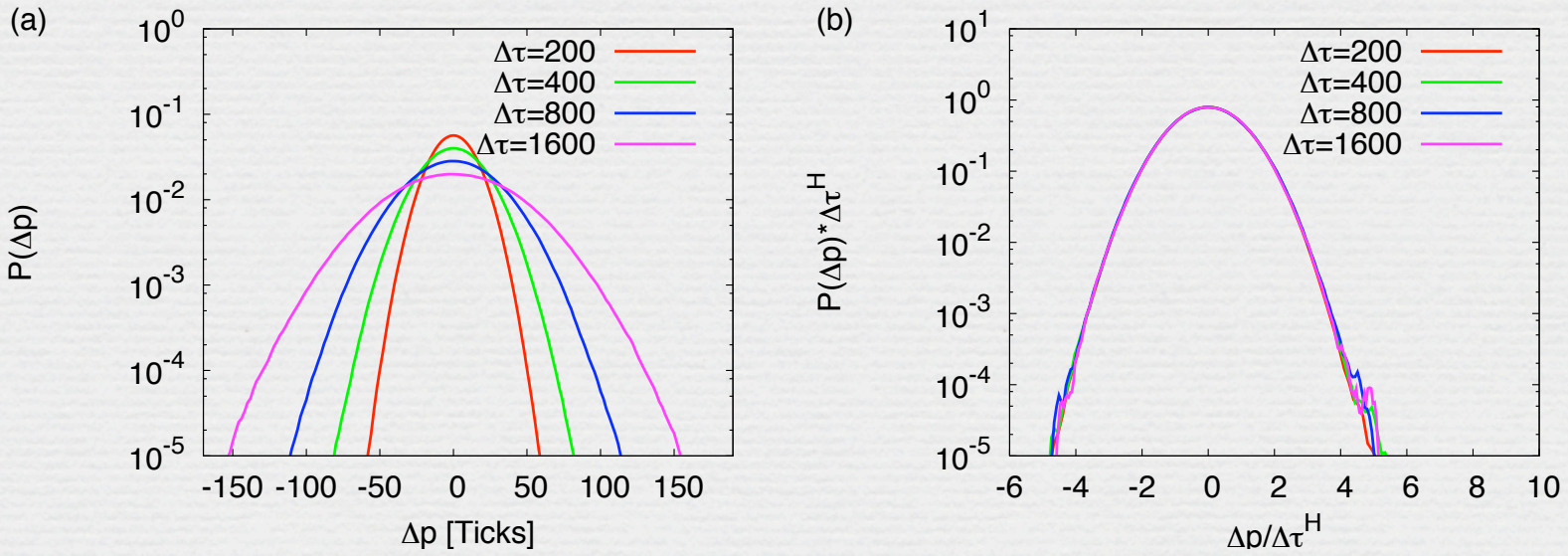
$$\langle (\Delta p)^2 \rangle_t \propto (\Delta \tau)^{2H}$$

- Universal scaling exponent:

$$\langle |\Delta p|^q \rangle_t \propto (\Delta \tau)^{\zeta(q)}$$

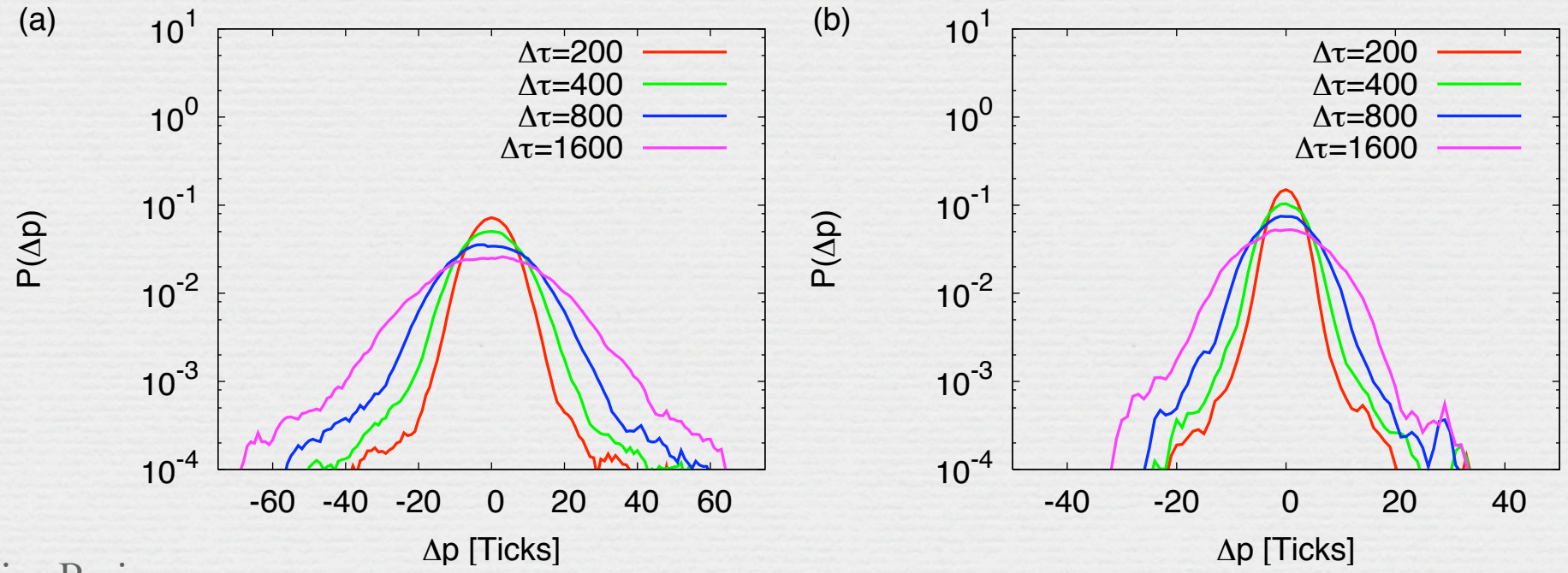
Price changes

- Price changes of a time series $\Delta p\,(t, \Delta\tau) = p\,(t + \Delta\tau) - p\,(t)$
- Returns $\Delta \ln p\,(t\Delta\tau) = \ln p\,(t + \Delta\tau) - \ln p\,(t)$
- Equivalent on short time scales if no crash events.
- RW:



Price changes

- Price changes of financial market data
  - ➡ (a) FDAX: 1.440.845 trades (2004)
  - ➡ (b) FESX: 732.485 trades (2004)



Price changes

- Truncated Lévy Flight (TLF) with smooth exponential cutoff
- Characteristic function (Fourier transformation of function $L_{\alpha,c_1,l}(\Delta p)$)

$$\Lambda_{\alpha c_1 l}(f_n) = \exp\left(c_0 - c_1 \frac{(f_n^2 + 1/l^2)^{\alpha/2}}{\cos(\pi\alpha/2)} \cos\left(\alpha \arctan(l|f_n|)\right)\right)$$

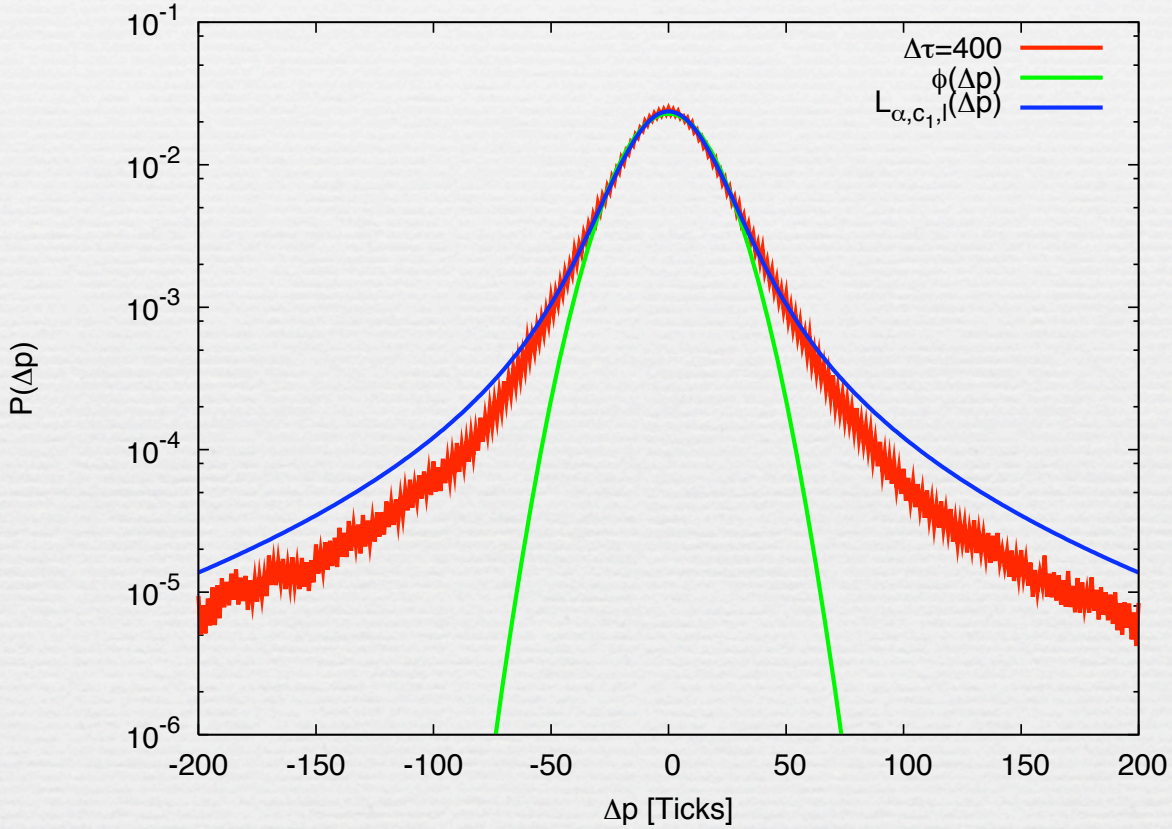
with $c_0 = \frac{l^{-\alpha}}{\cos(\pi\alpha/2)}$

- Discret Fourier transformation of the price change distributions ...

$$H(f) = \int_{-\infty}^{\infty} h(t)\, e^{2\pi i f t} dt \qquad h(t) = \int_{-\infty}^{\infty} H(f)\, e^{-2\pi i f t} df$$

Price changes

- ... fitting and inverse transformation

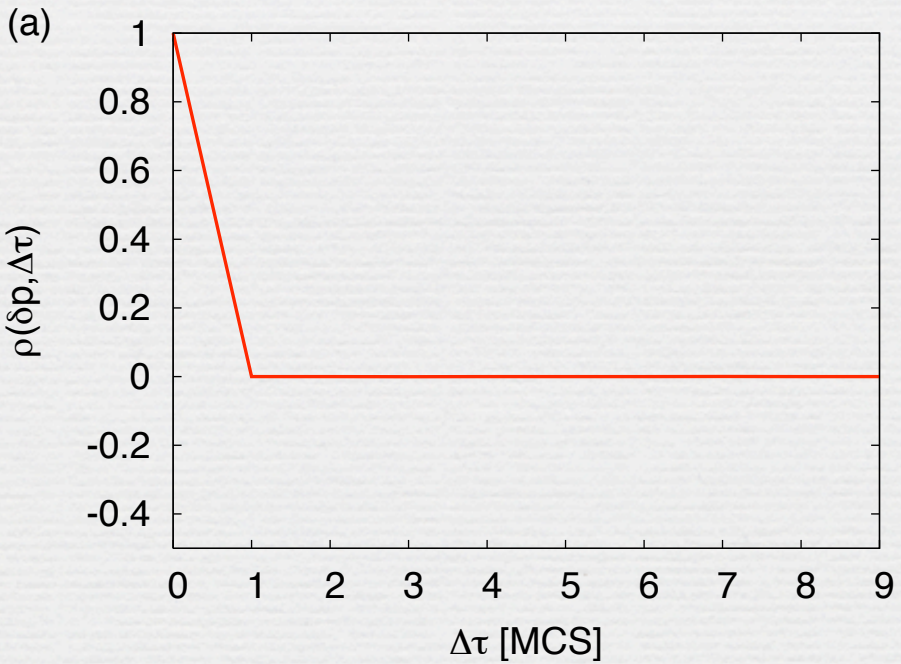


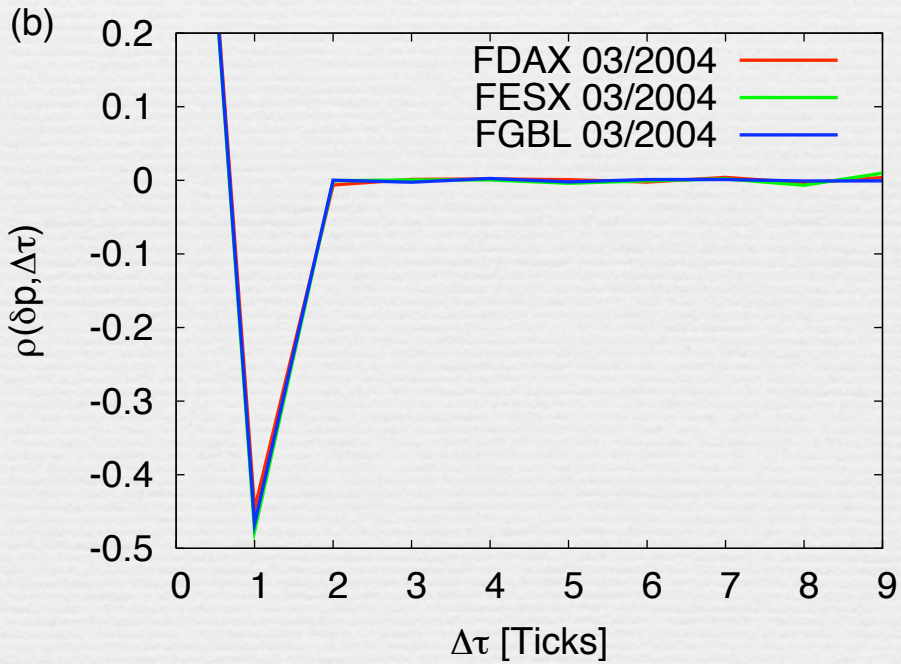
FDAX 37.933.953 trades (1991 – 2003) with Gaussian fit $\phi(\Delta p) = a \cdot \exp\left(-b \cdot \Delta p^2\right)$

Autocorrelation

$$\rho\left(\delta p \Delta\tau\right) = \frac{\langle \delta p(t) \cdot \delta p(t+\Delta\tau)\rangle - \langle \delta p(t)\rangle \cdot \langle \delta p(t+\Delta\tau)\rangle}{\sqrt{\mathrm{Var}\left(\delta \mathrm{p(t)}\right) \cdot \mathrm{Var}\left(\delta \mathrm{p(t+\Delta\tau)}\right)}}$$

$$\delta p\left(t\right) = p\left(t+1\right) - p\left(t\right)$$

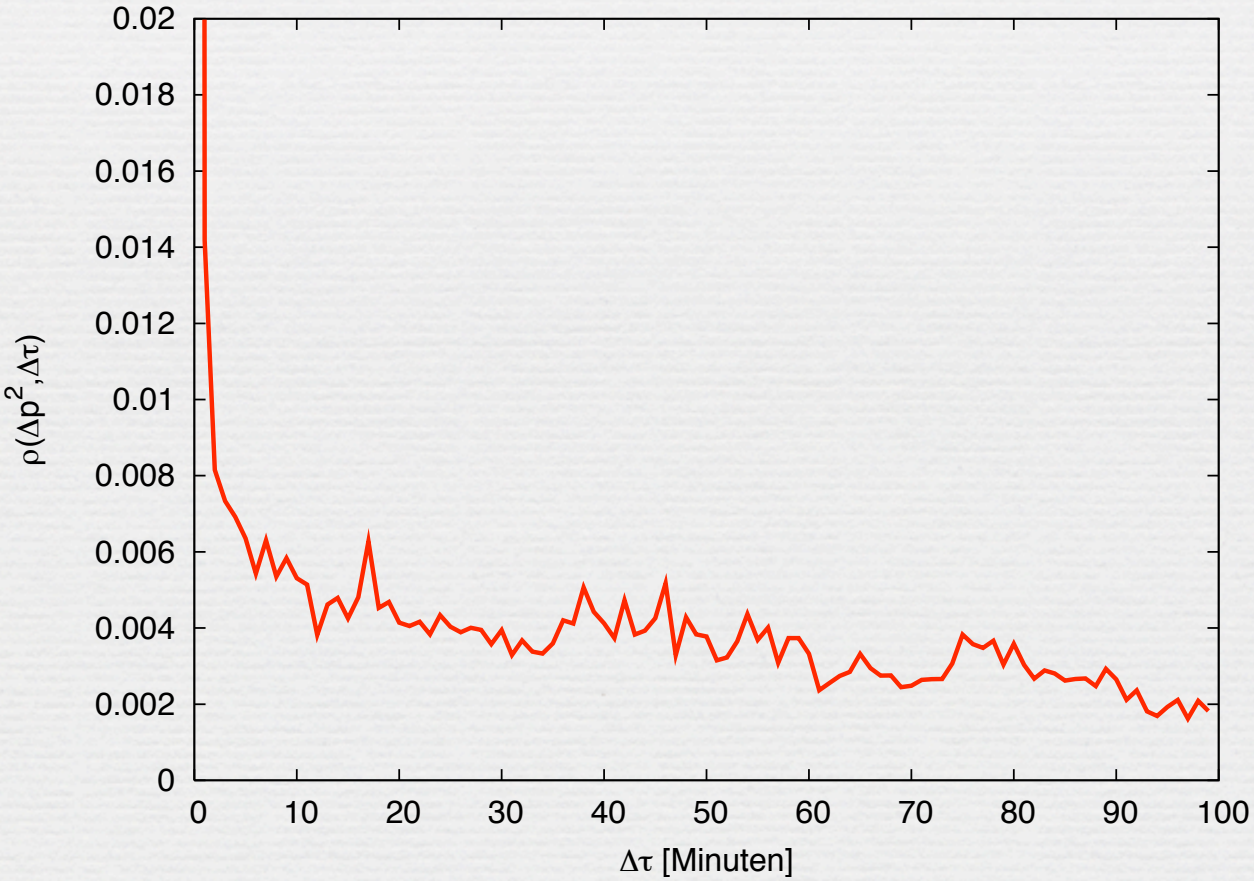




- RW
- Market data

Volatility clustering

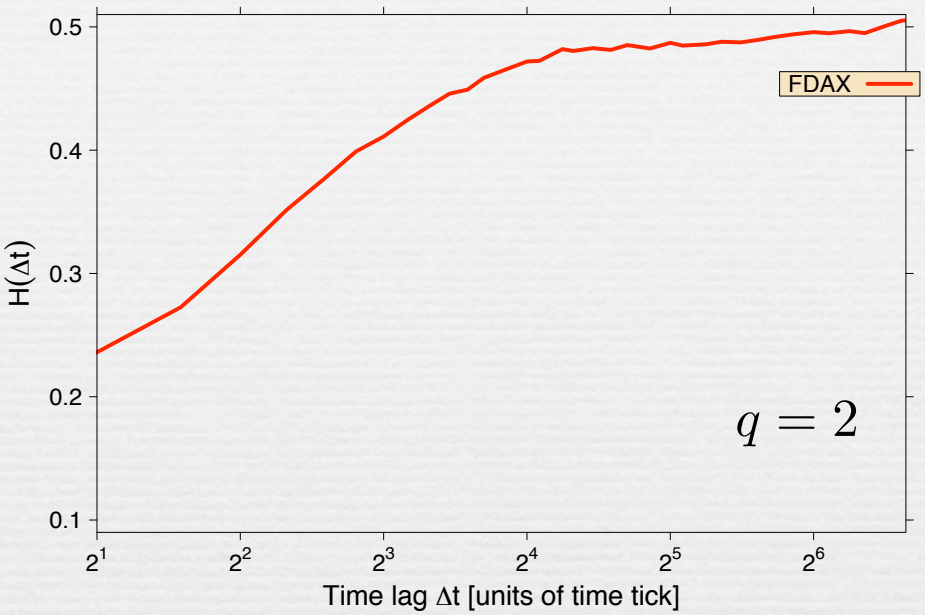
$$\rho\left(\delta p \Delta\tau\right) = \frac{\langle \delta p(t) \cdot \delta p(t+\Delta\tau)\rangle - \langle \delta p(t)\rangle \cdot \langle \delta p(t+\Delta\tau)\rangle}{\sqrt{\mathrm{Var}\left(\delta \mathrm{p(t)}\right) \cdot \mathrm{Var}\left(\delta \mathrm{p(t+\Delta\tau)}\right)}}$$



$$\delta p\left(t\right)^2 = \left(p\left(t+1\right) - p\left(t\right)\right)^2$$

Long range correlated volatility

Scaling Behavior



- Hurst exponent H

$$\langle |p(t+\Delta t) - p(t)|^q \rangle^{1/q} \propto \Delta t^{H_q(\Delta t)}$$

Time series $p(t)$

with $t = 1, 2, \ldots, T$

Synthetically anti-correlated random walk $p^*_\gamma(t) = a_\gamma(t) + b(t)$

With probability $\gamma \in [0; 1/2]$:

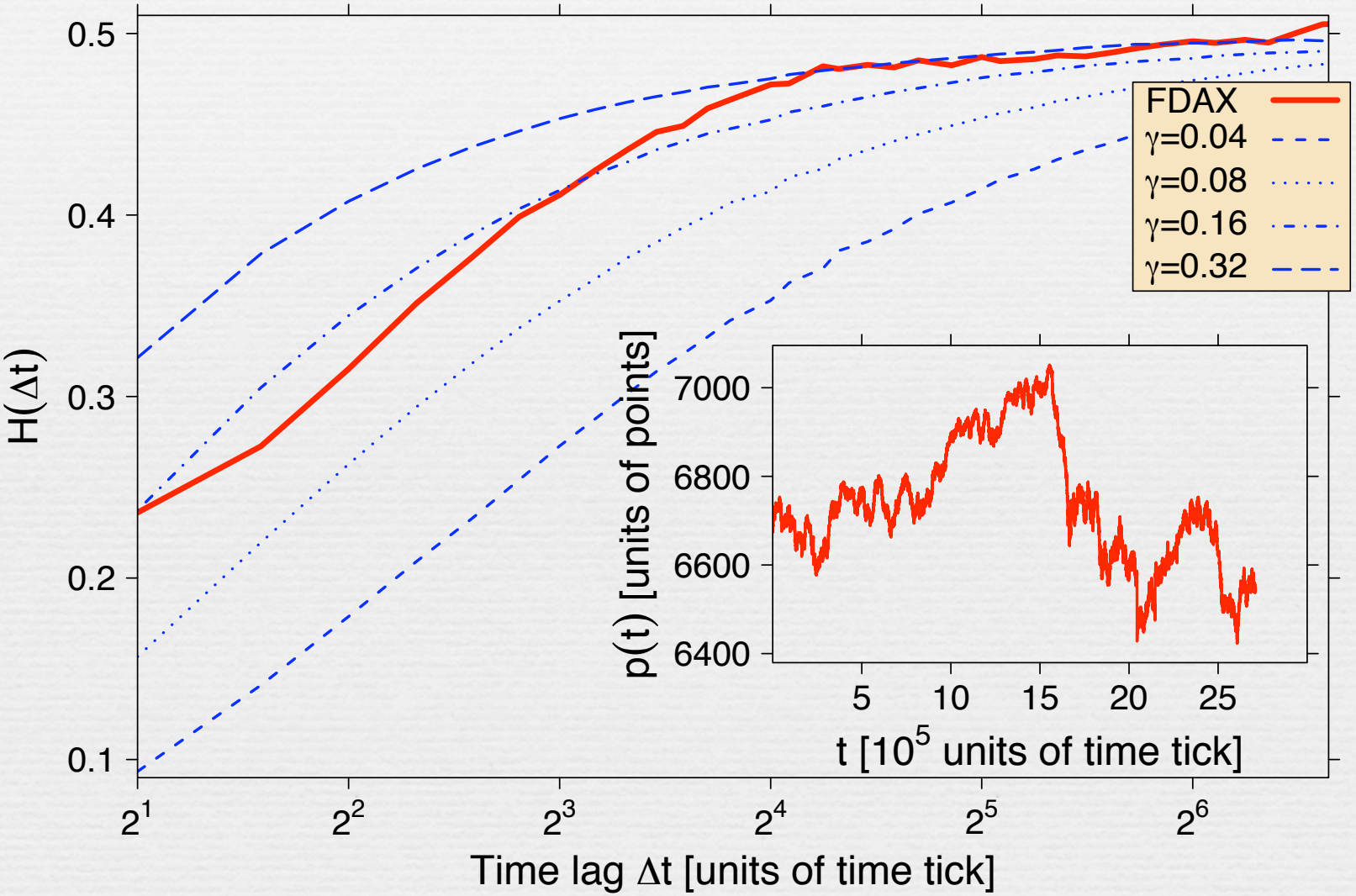
$$a_\gamma(t+1) - a_\gamma(t) = +1$$
$$a_\gamma(t+1) - a_\gamma(t) = -1$$

With probability $1 - 2\gamma$:

$$a_\gamma(t+1) = a_\gamma(t)$$

The stochastic variable $b(t)$ models the bid-ask spread and can take the value 0 and 1 in each time step, each with probability 1/2.

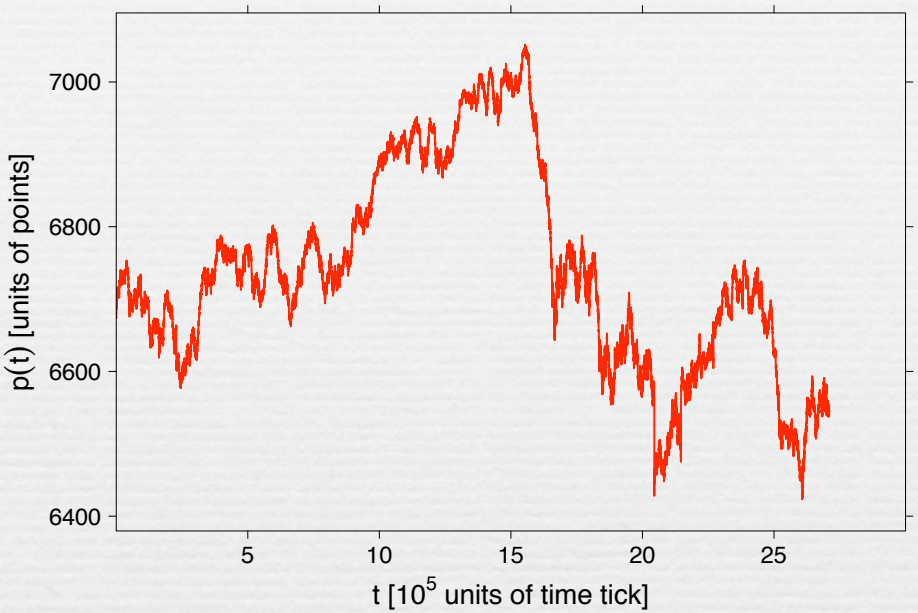
Anti-Correlated Random Walk



Synthetically anti-correlated random walk $p^*_\gamma(t) = a_\gamma(t) + b(t)$

Tobias Preis

Financial Markets Time Series



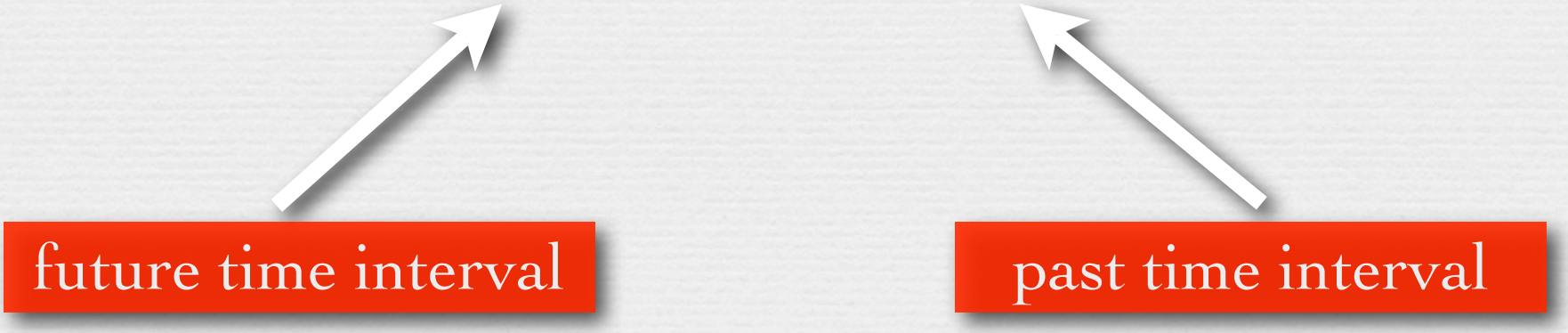
FDAX time series contains 2,709,952 trades recorded from 2 January 2007 to 16 March 2007

- Empirical stylized facts:
  - ➡ Autocorrelation
  - ➡ Volatility clustering
  - ➡ Scaling behavior / short time anti-persistence
  - ➡ Fat-tailed PDF of price changes

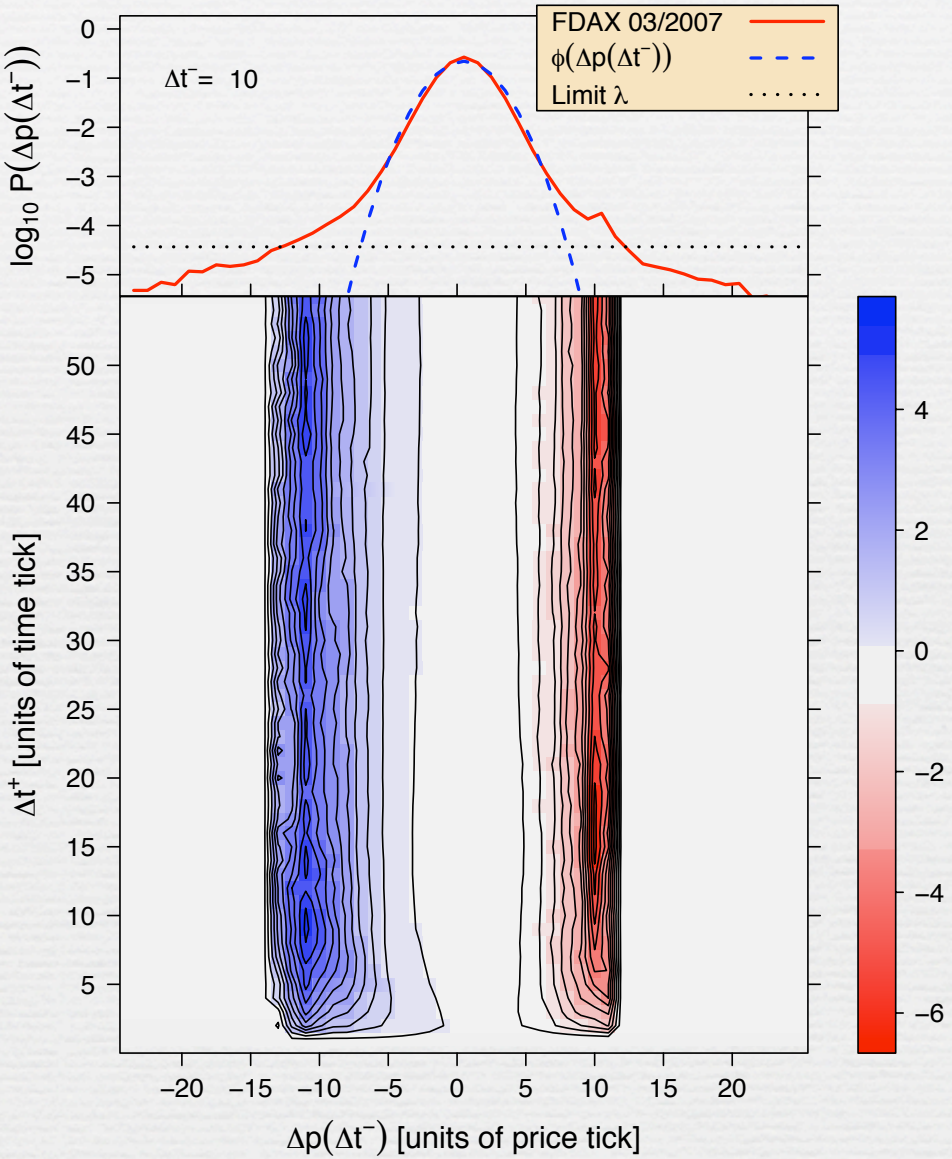
What happens after price impacts?

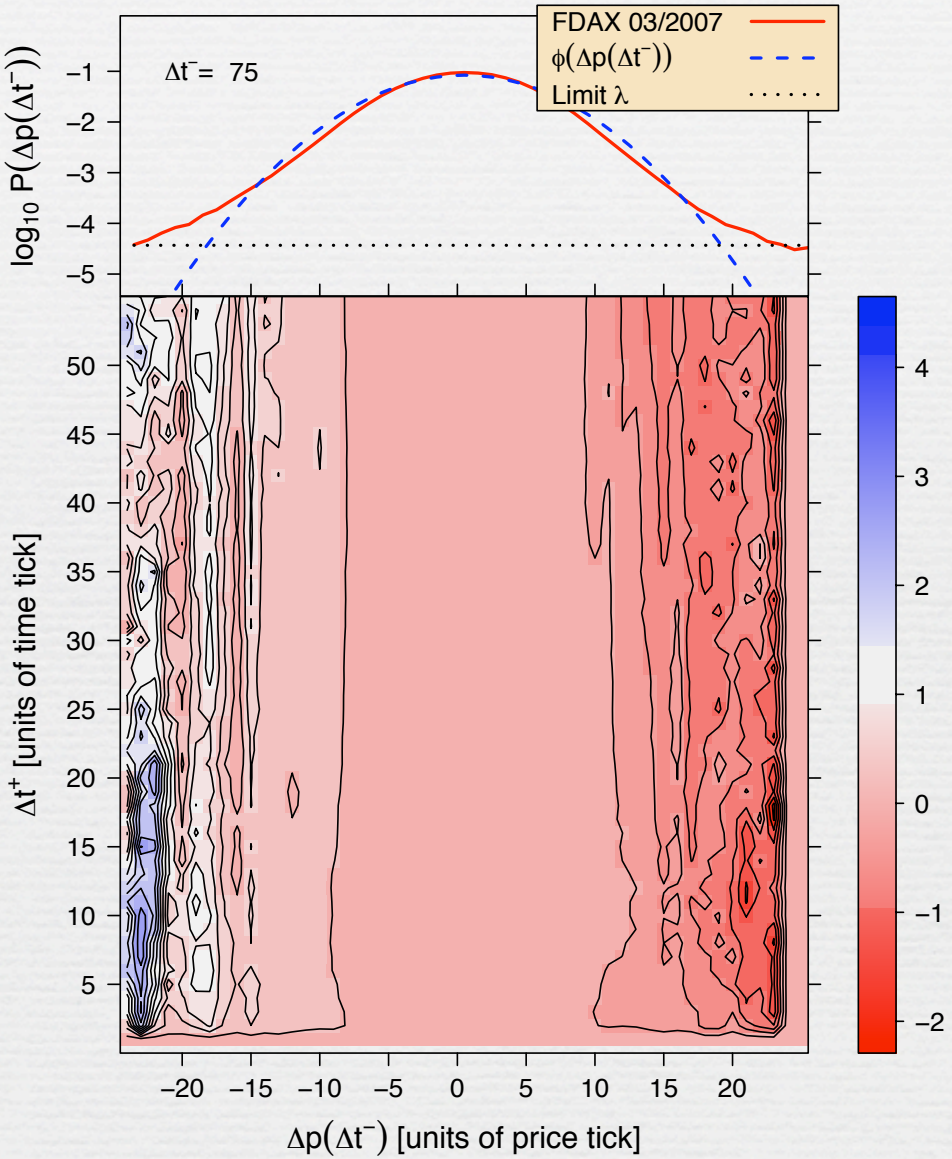
Conditional expectation value

$$\langle p(t+\Delta t^{+}) - p(t) | p(t) - p(t-\Delta t^{-}) \rangle_t$$



Conditional PDFs

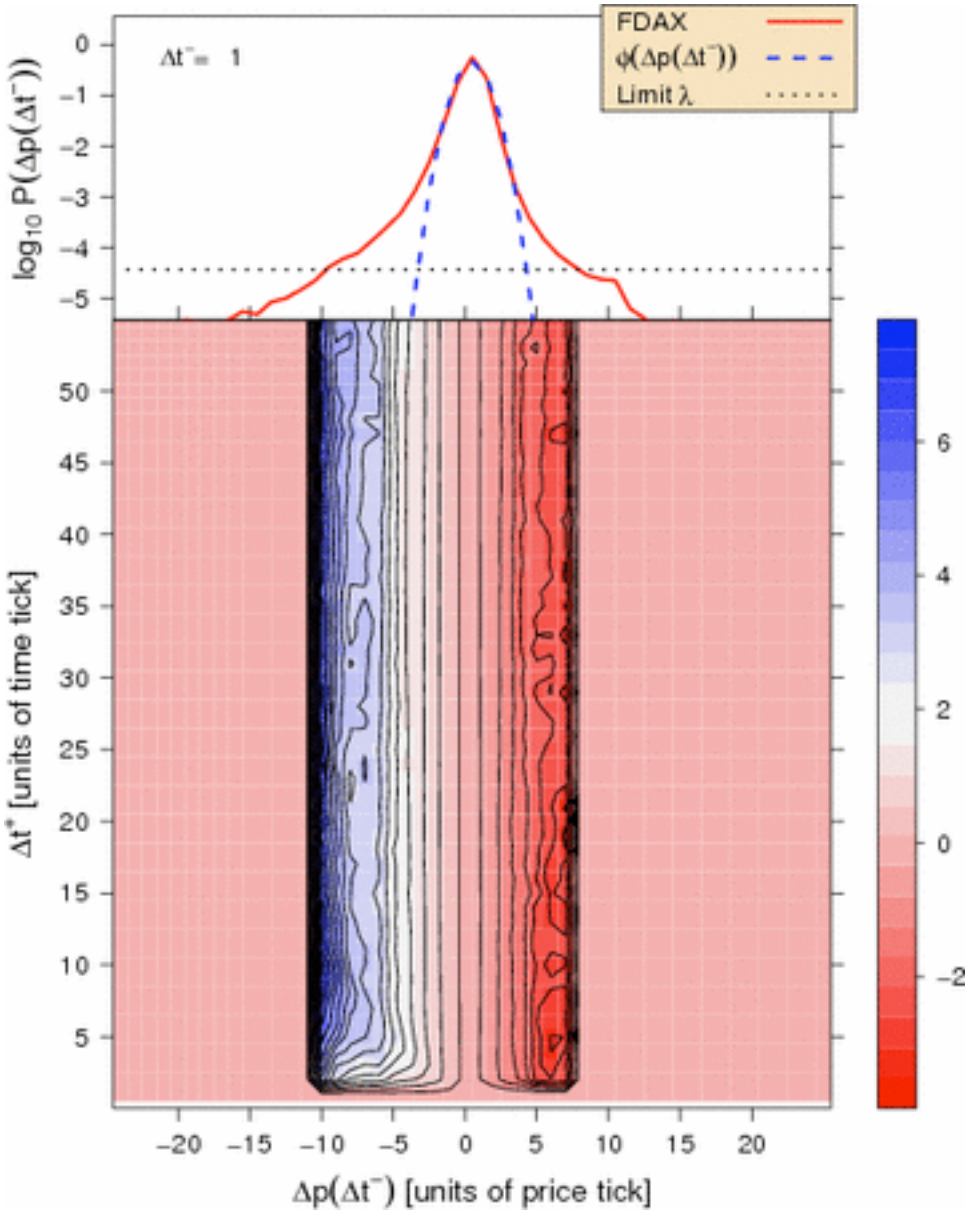


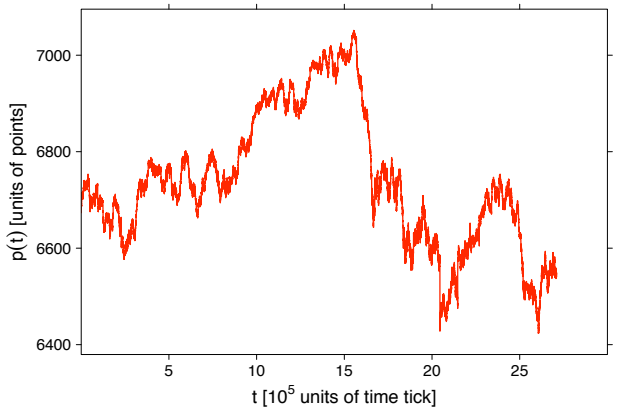


$$\phi(\Delta p(\Delta t^-)) = u \exp(-v \Delta p^2)$$

Conditional PDF

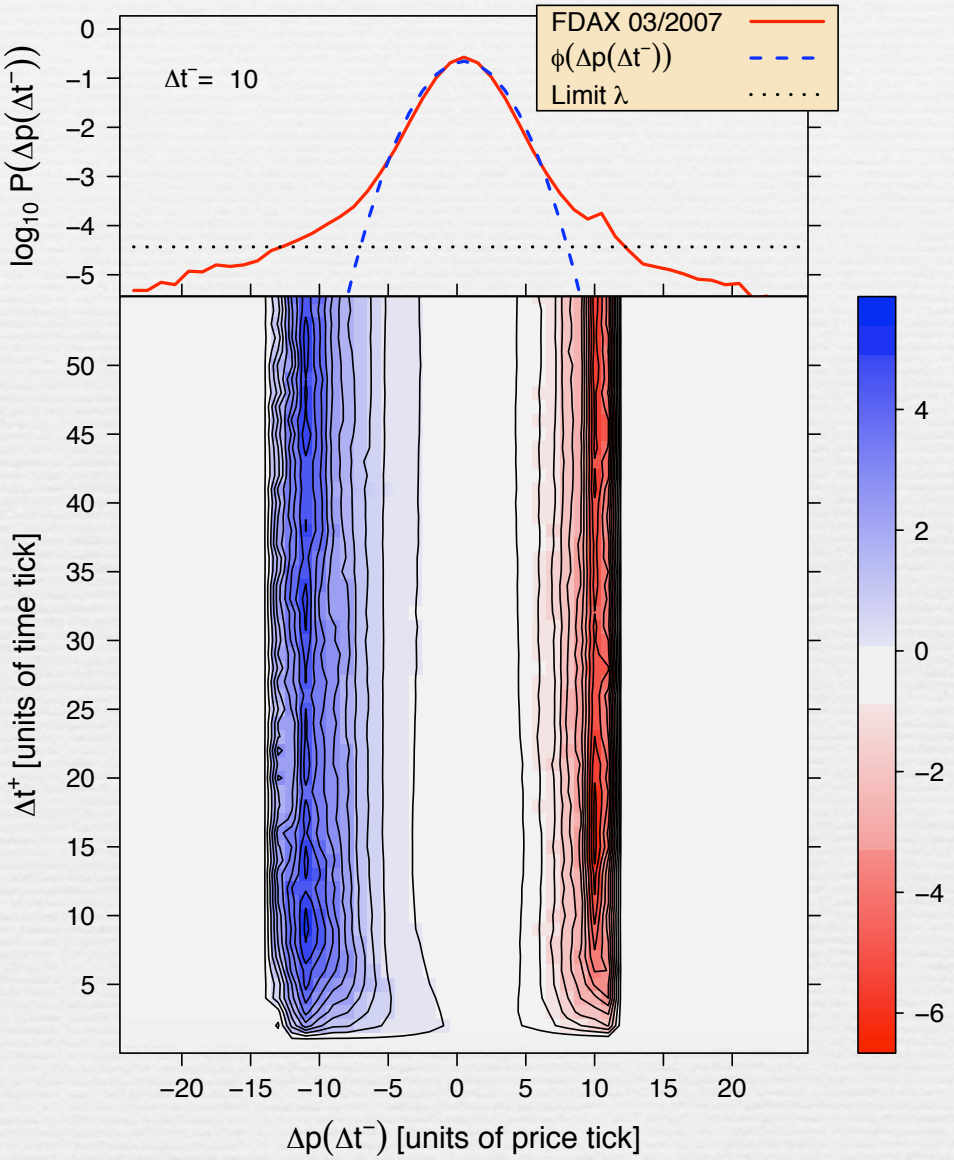
FDAX time series

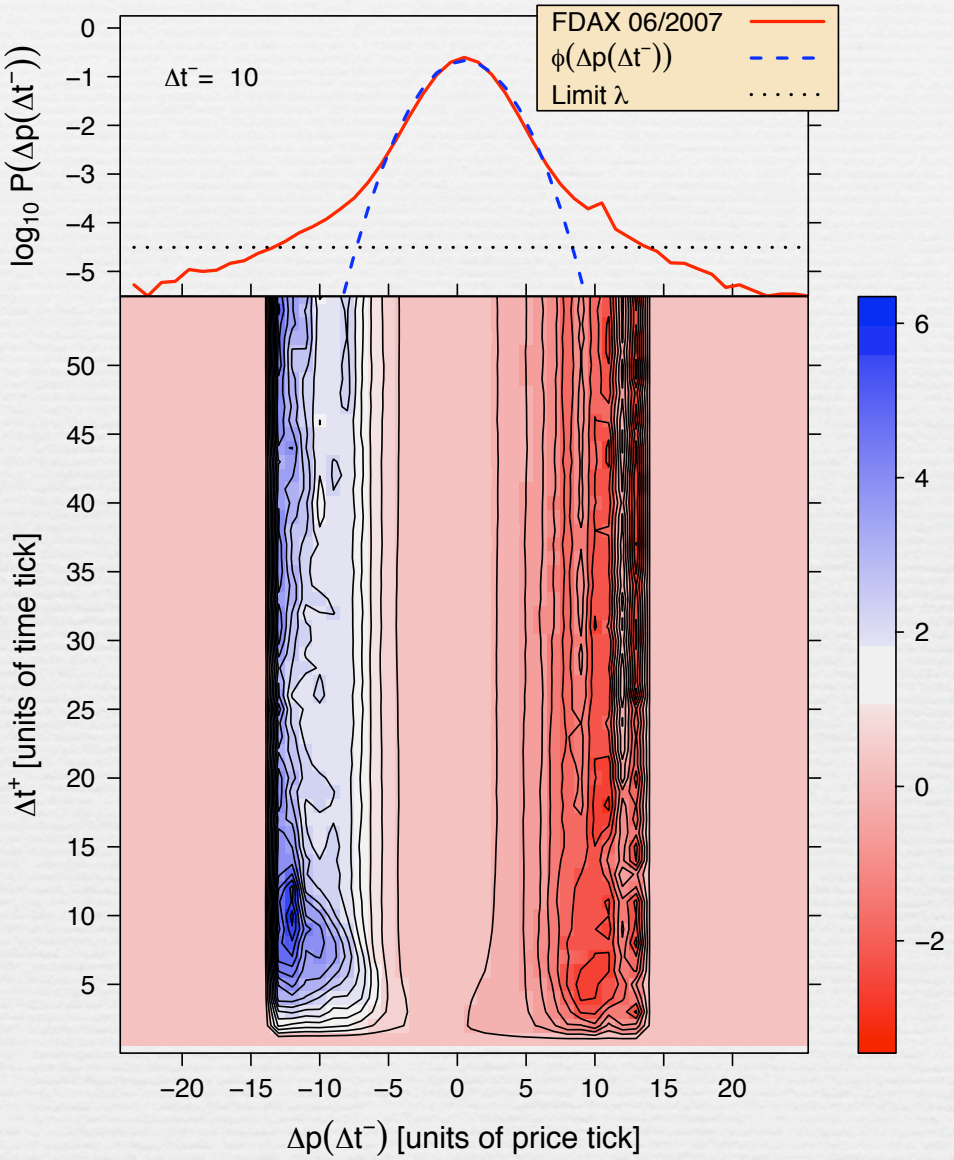




T. Preis et al., Europhys. Lett. **82**, 68005 (2008)

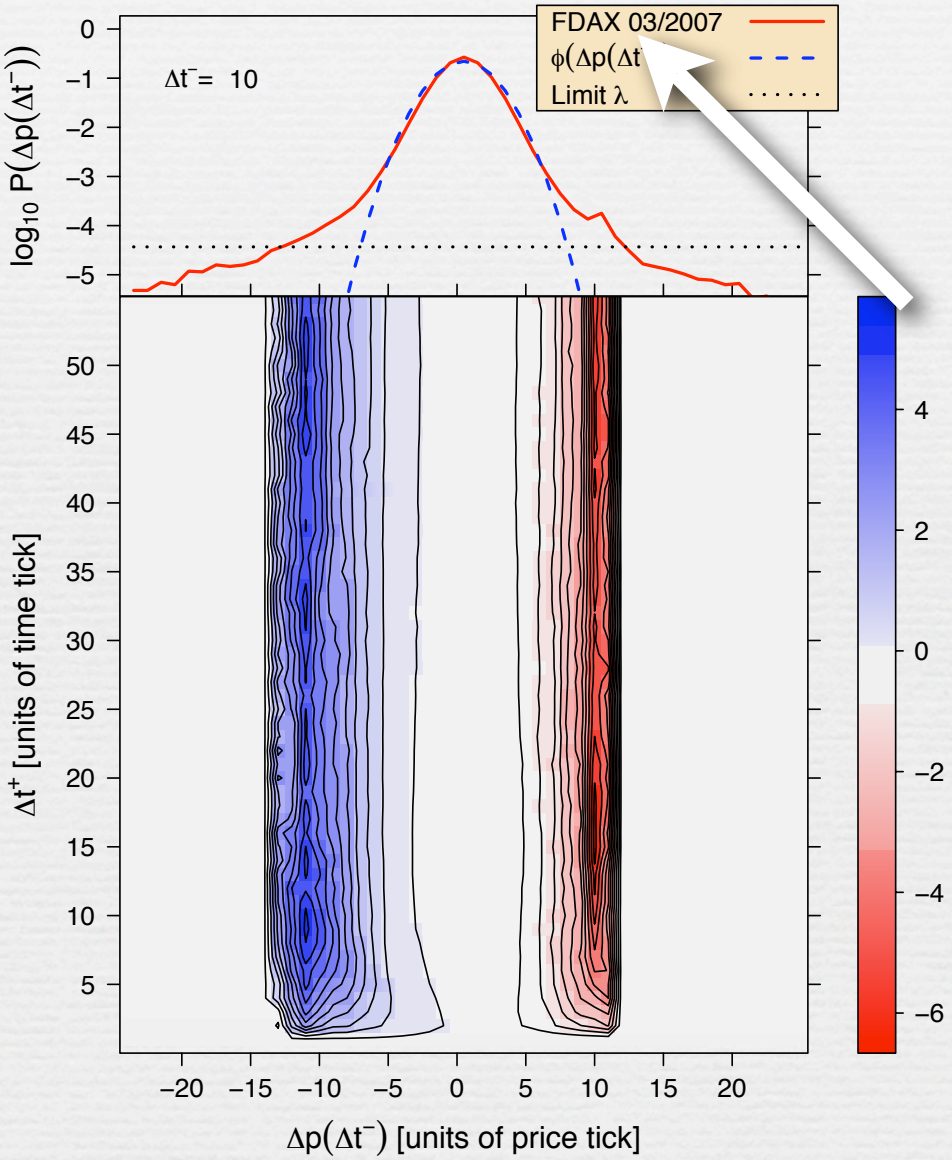
CPDFs / Market Condition

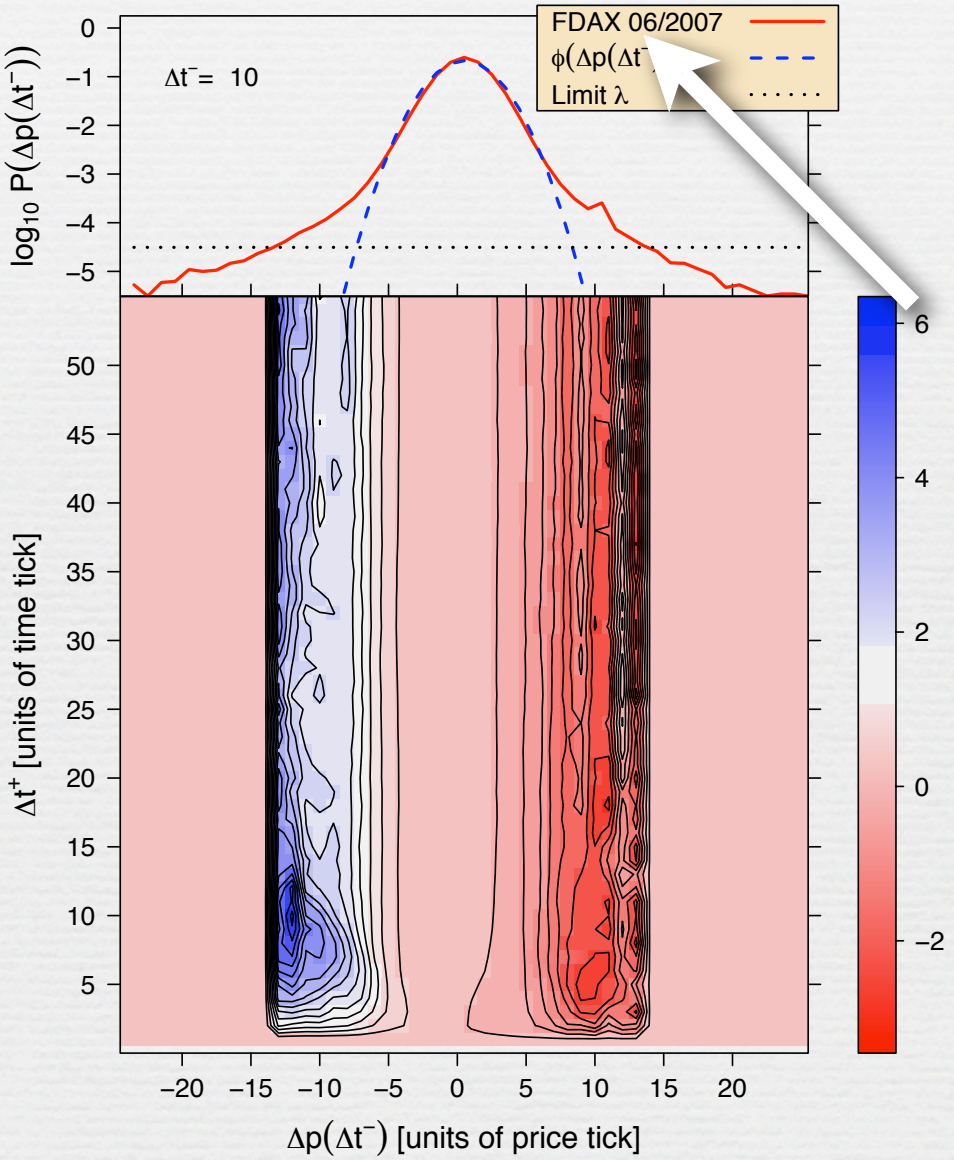




Tobias Preis

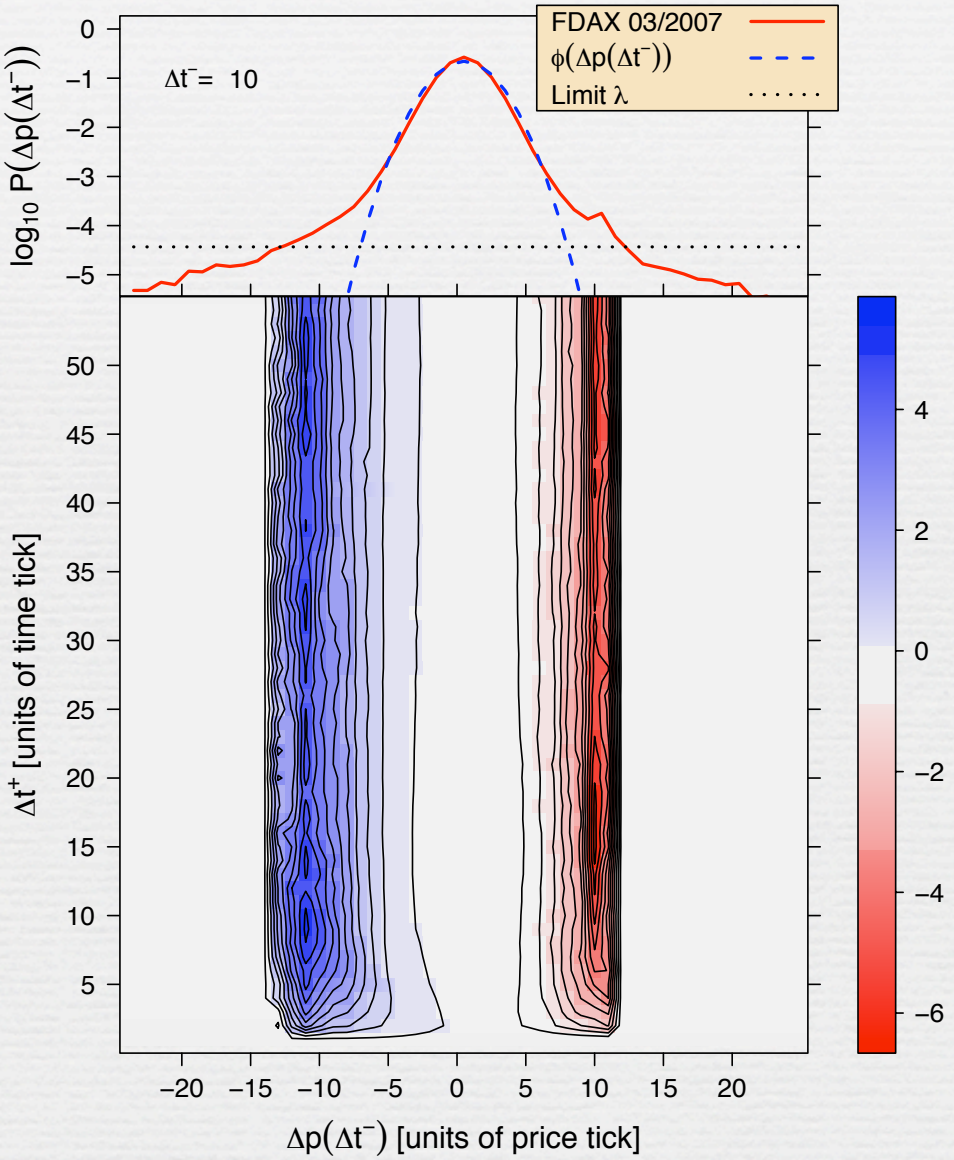
CPDFs / Market Condition

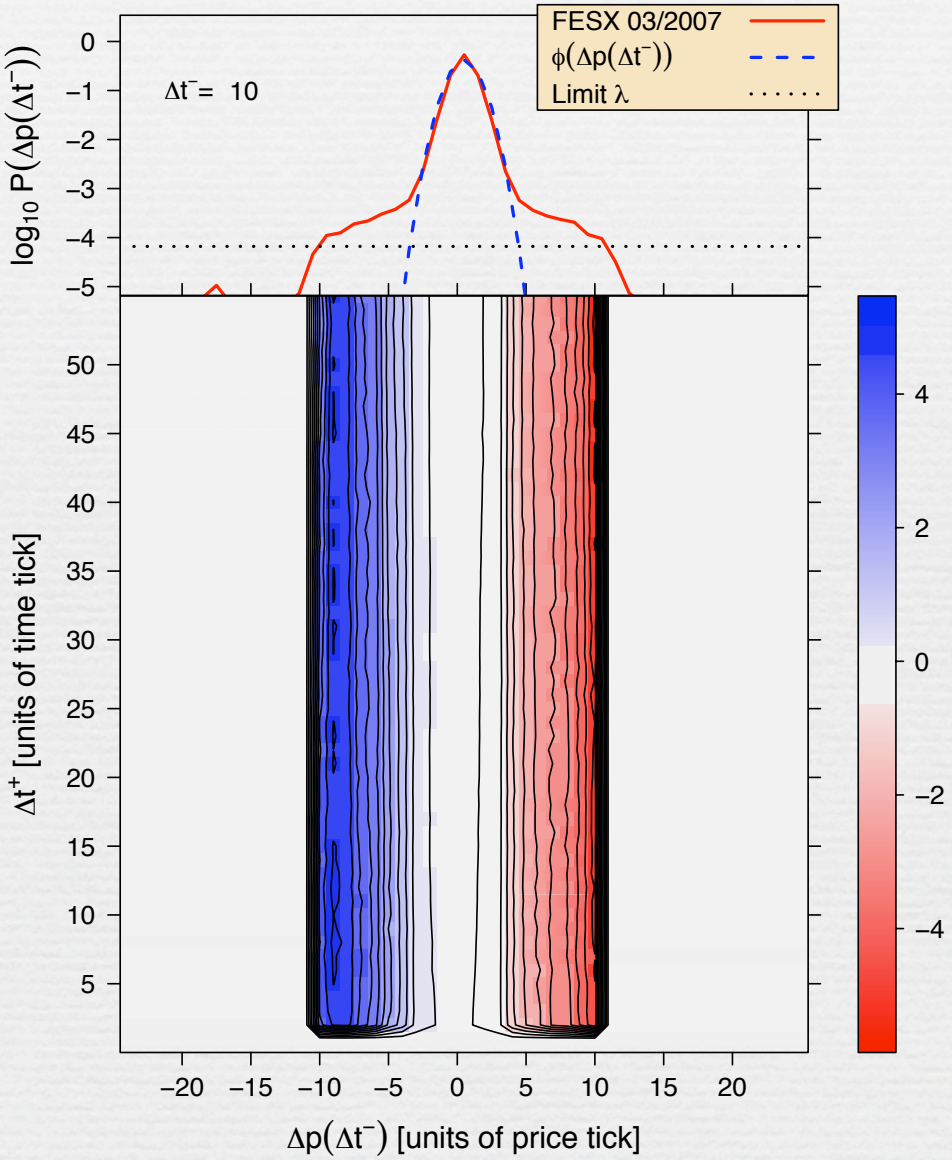




Tobias Preis

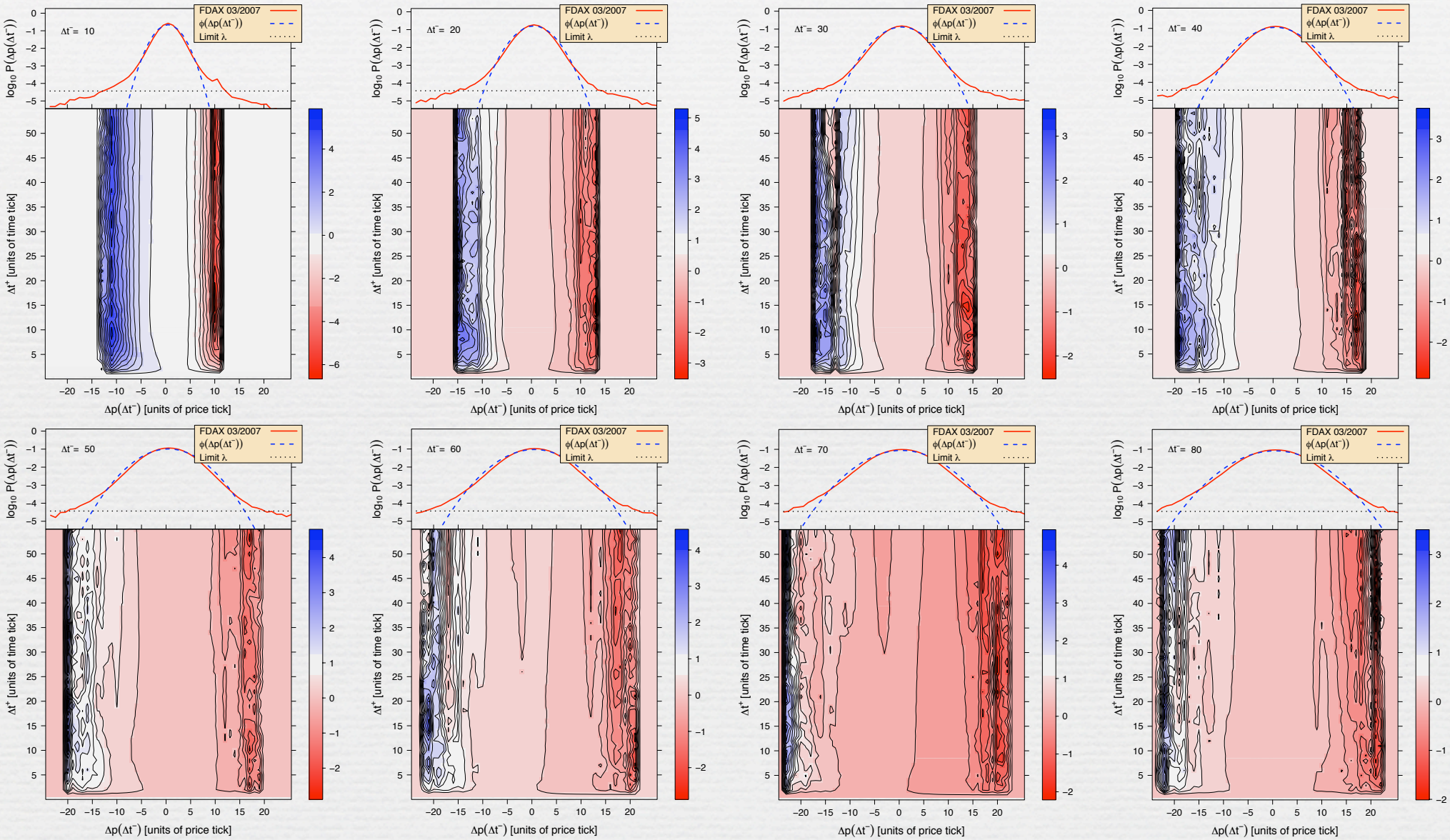
CPDFs / FDAX vs. FESX





Tobias Preis

Conditional PDFs / FDAX

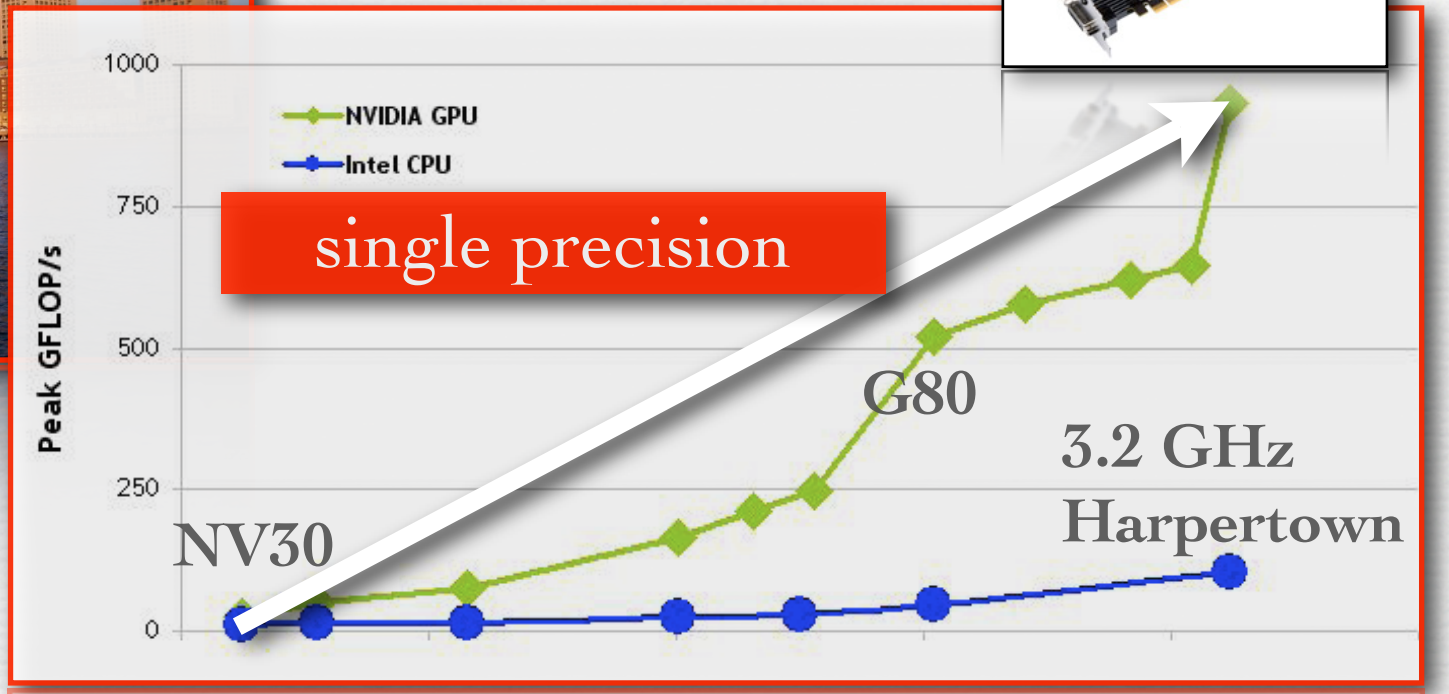


Tobias Preis

GPGPU computing in Econophysics

Realistic illustrations

GT200

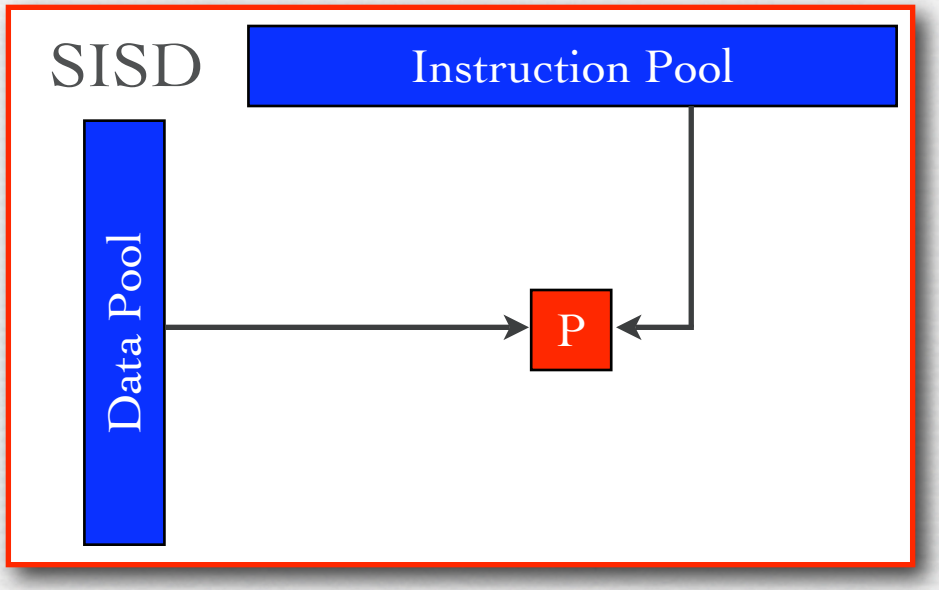


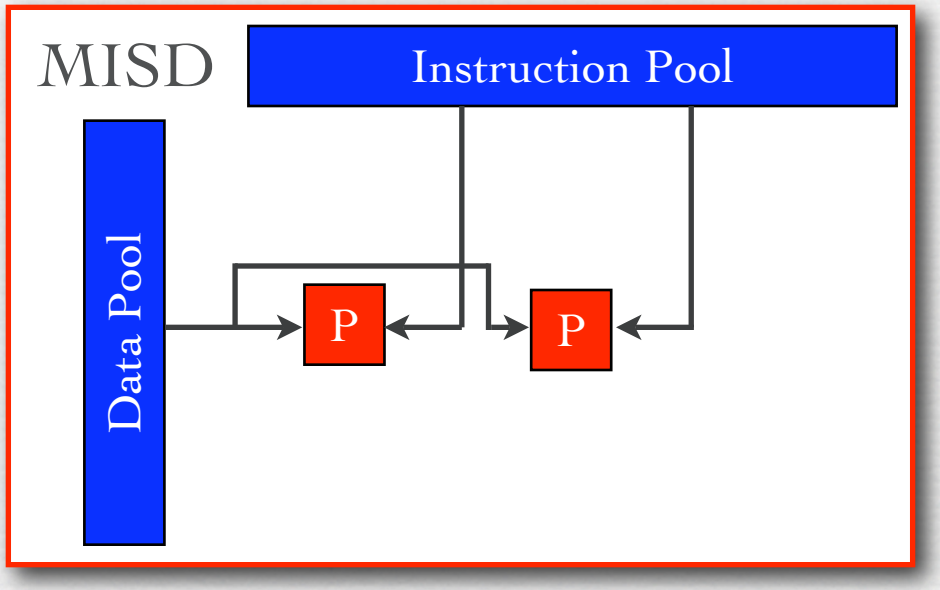
Source: NVIDIA CUDA programming guide

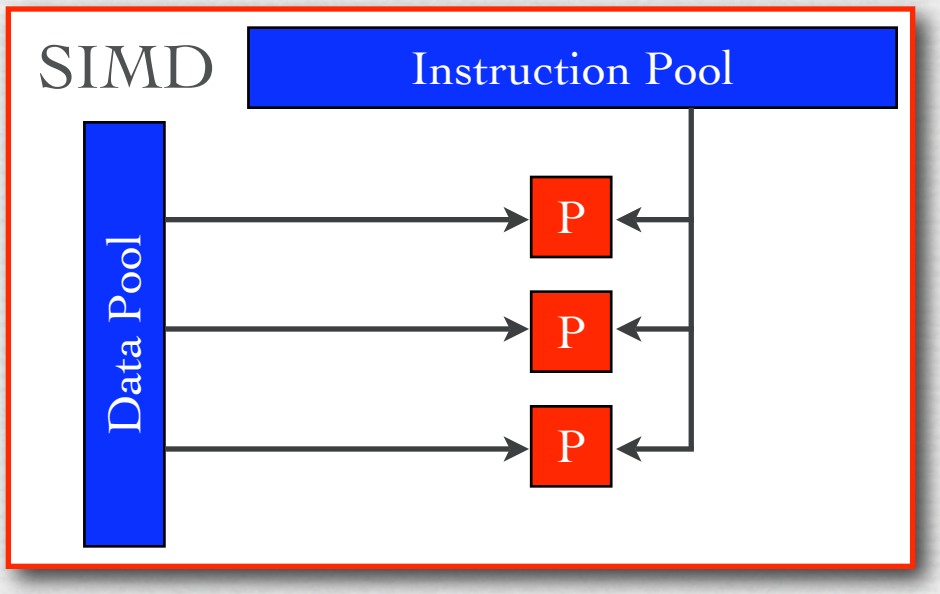
Driving force: computer game industry

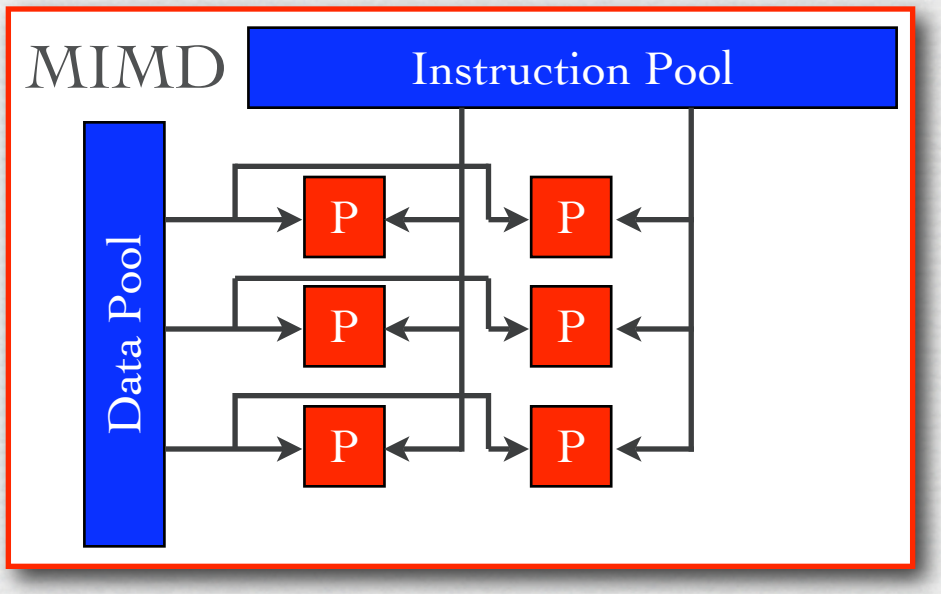
Computer Architectures

Flynn's taxonomy, proposed by Michael J. Flynn in 1966

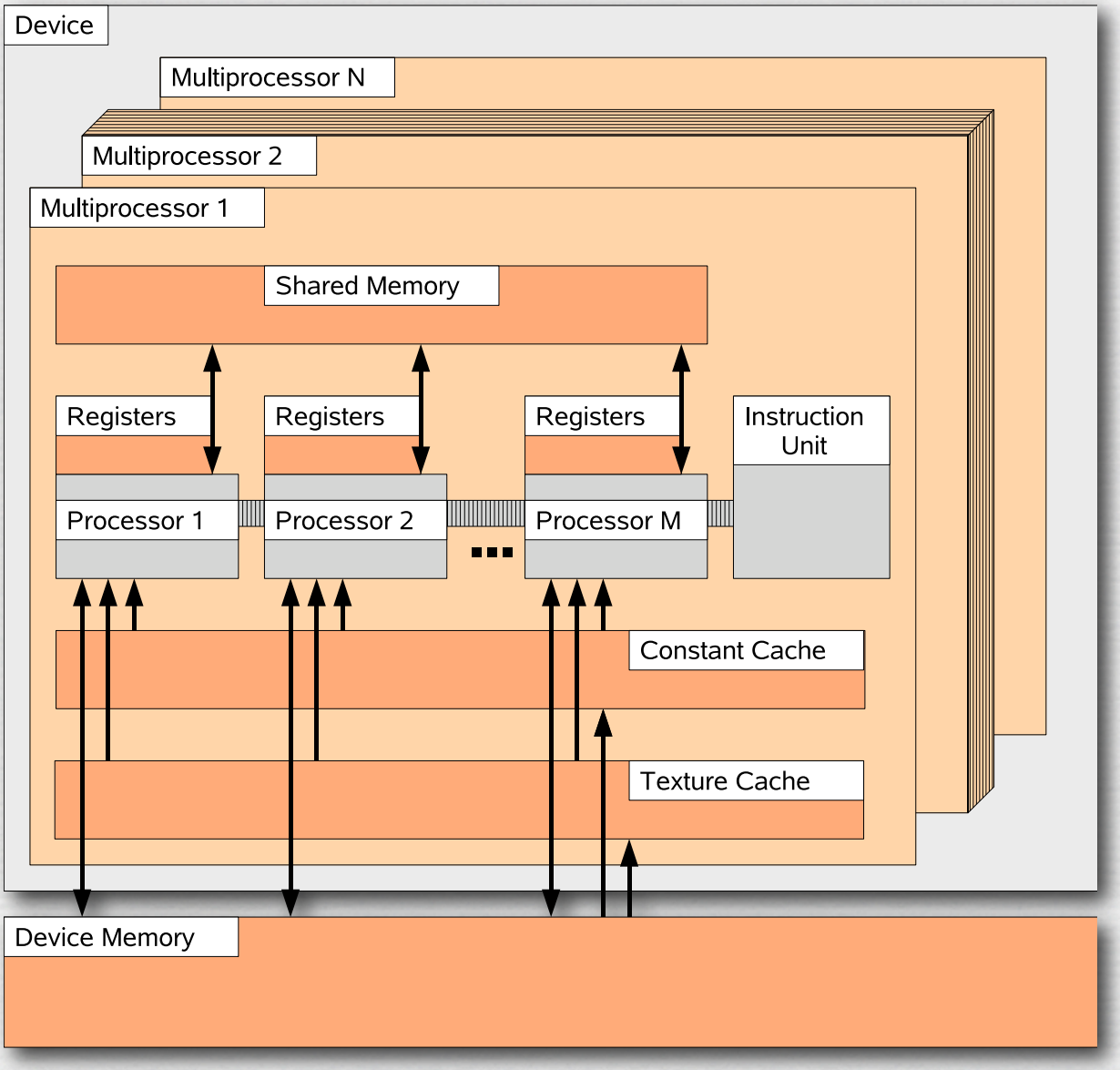








GPU device architecture



GeForce GTX 280:

Global memory 1024 MB

Number of multiprocessors 30

Number of cores 240

Constant memory 64 kB

Shared memory 16 kB

Clock rate 1.30 GHz

GPU device / Reference system

Reference CPU:

Intel Core 2 Quad 2.66 GHz

Cache size 4096 KB

GeForce GTX 280:

Global memory 1024 MB

Number of multiprocessors 30

Number of cores 240

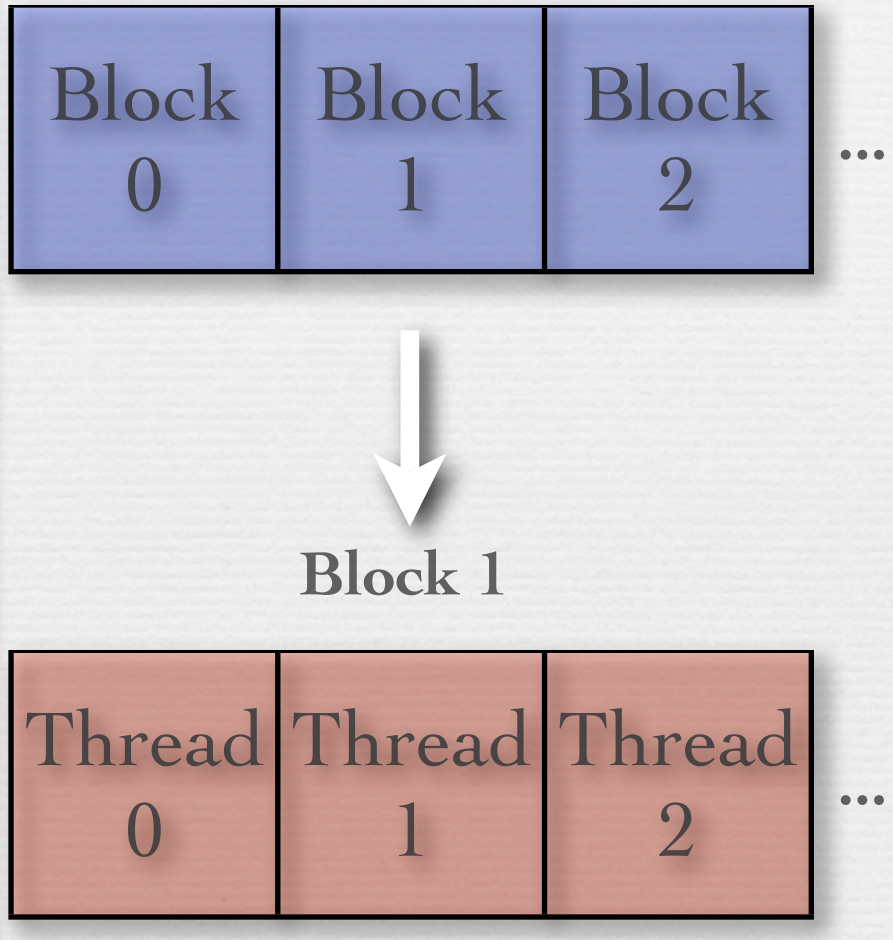
Constant memory 64 kB

Shared memory 16 kB

Clock rate 1.30 GHz

C – code with extensions

```
__global__ void gpu_function(int n, float* a, float* b) {

    //Determine array element
    int i = threadIdx.x + blockIdx.x * blockDim.x;

    if(i<n) b[i] += a[i] * a[i];
}

__host__ void cpu_function() {

    int n = 128 * 128;
    int n_blocks = 128;
    int n_threads = 128;

    gpu_function<<<n_blocks,n_threads>>>(n, a, b);

    // Global barrier between GPU functions

    gpu_function<<<n_blocks/2,n_threads*2>>>(n, a, b);
}
```



Linear congruential RNGs

$$x_{i+1,j} = (a \cdot x_{i,j} + c) \mod m$$

$$x_{0,j+1} = (16807 \cdot x_{0,j}) \mod m$$

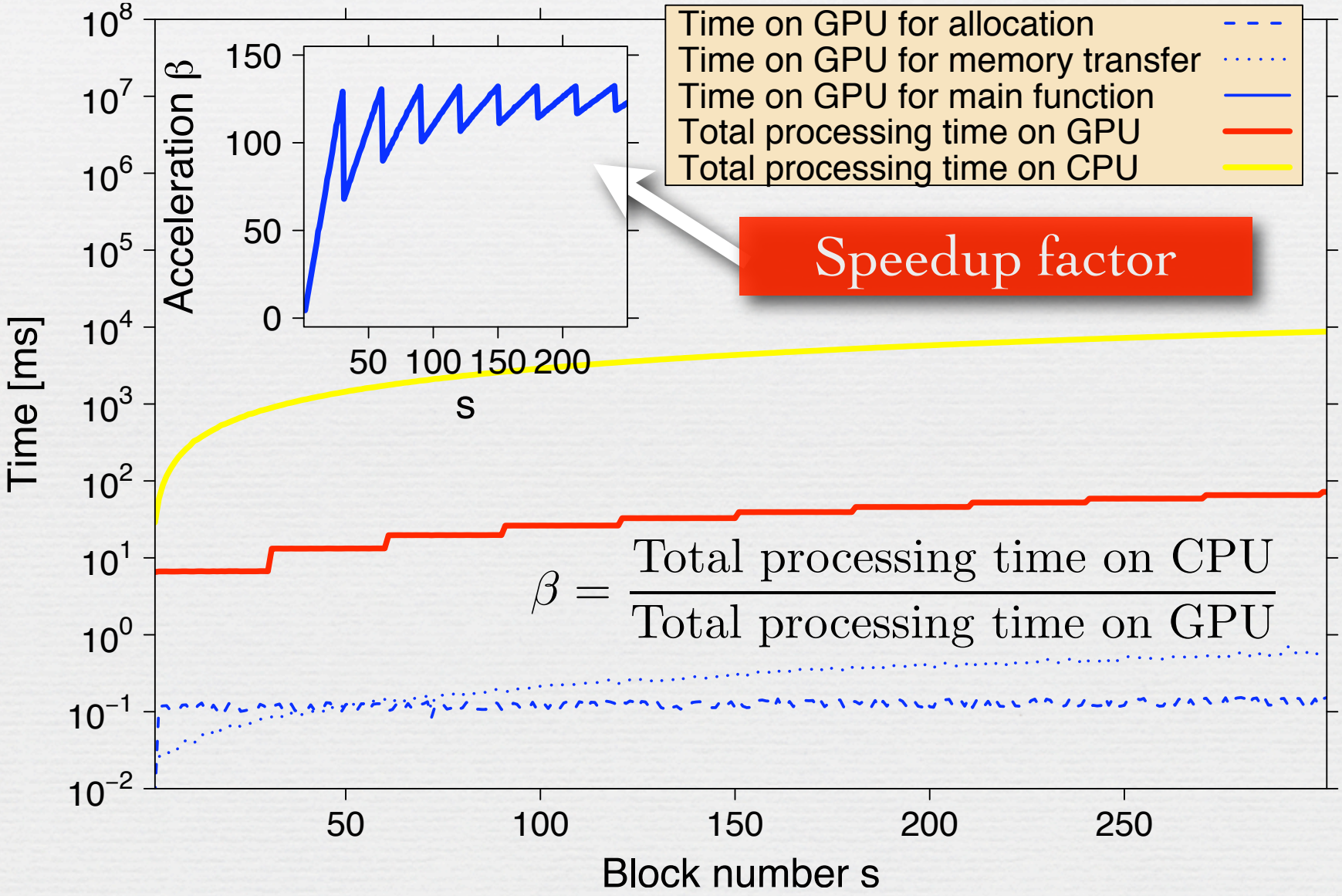
$$a = 1664525$$

$$c = 1013904223$$

32–bit architecture provided by the GPU $x_{i,j} \in [-2^{31}; 2^{31} - 1]$

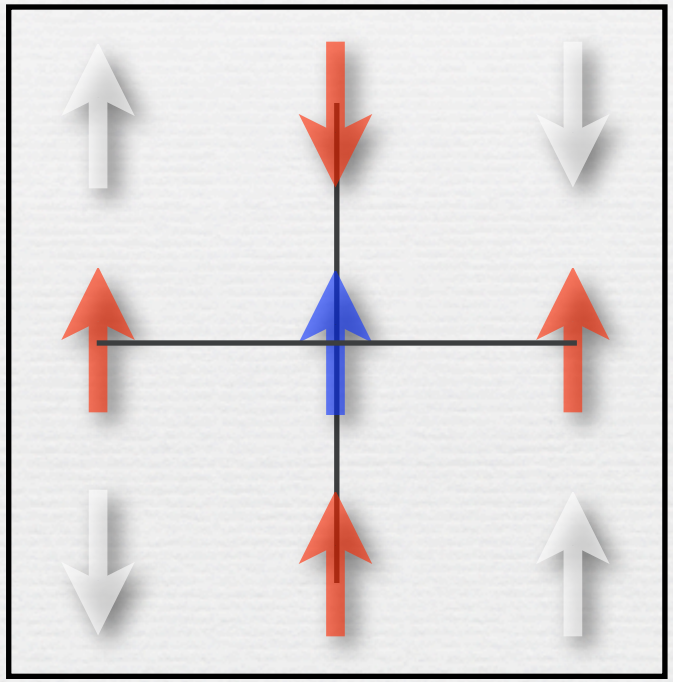
$$\longrightarrow \quad y_{i,j} = \mathrm{abs}\left(x_{i,j}/2^{31}\right) \sim 4.656612 \cdot 10^{-10} \mathrm{abs}\left(x_{i,j}\right)$$

Computation times – Random numbers



Ising model

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} S_i S_j - H \sum_i S_i$$



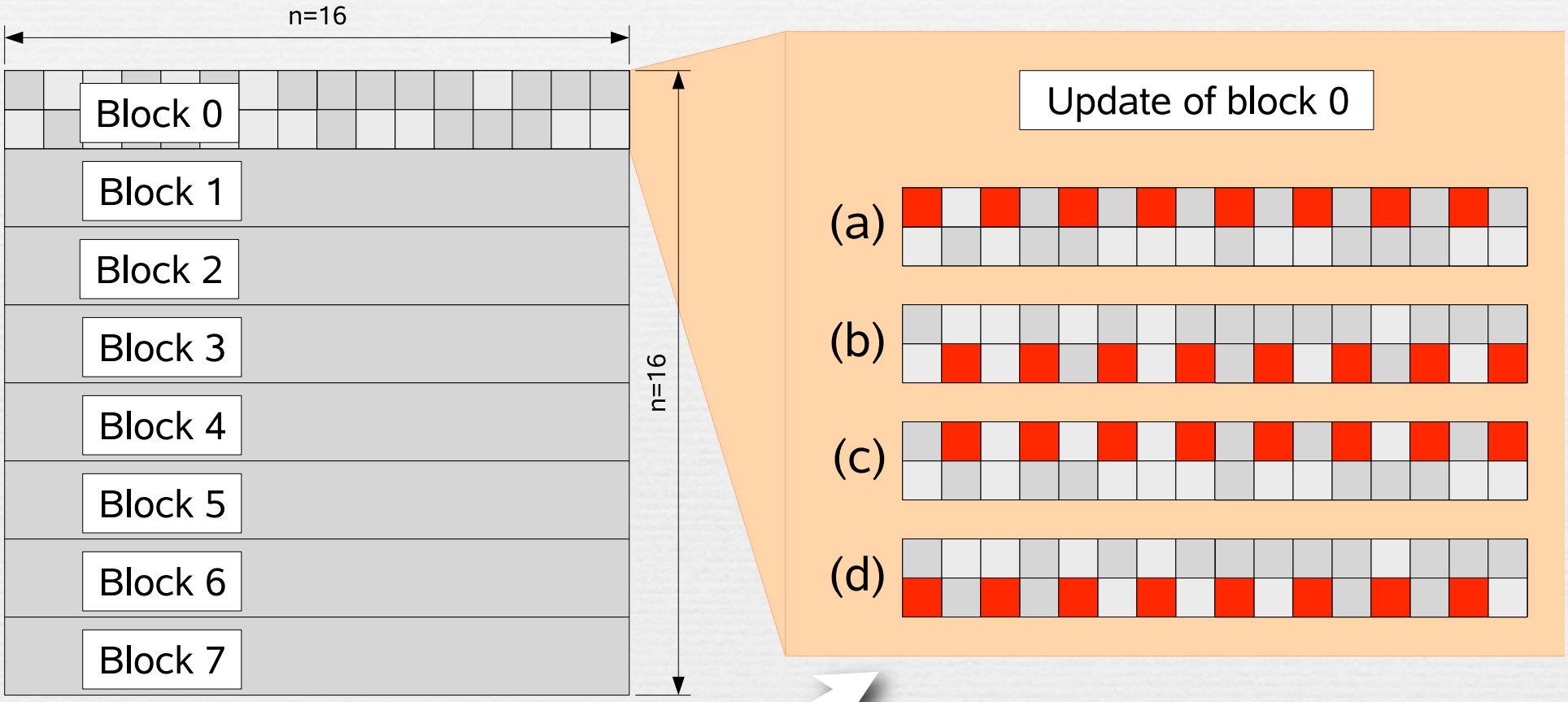
nearest neighbors

Spin update: Metropolis criterion

$$W_{a \to b} = \exp(-\Delta \mathcal{H} / k_B T) \quad \text{if} \quad \Delta \mathcal{H} > 0$$

$$W_{a \to b} = 1 \quad \text{if} \quad \Delta \mathcal{H} \leq 0$$

2D – Ising: GPU implementation

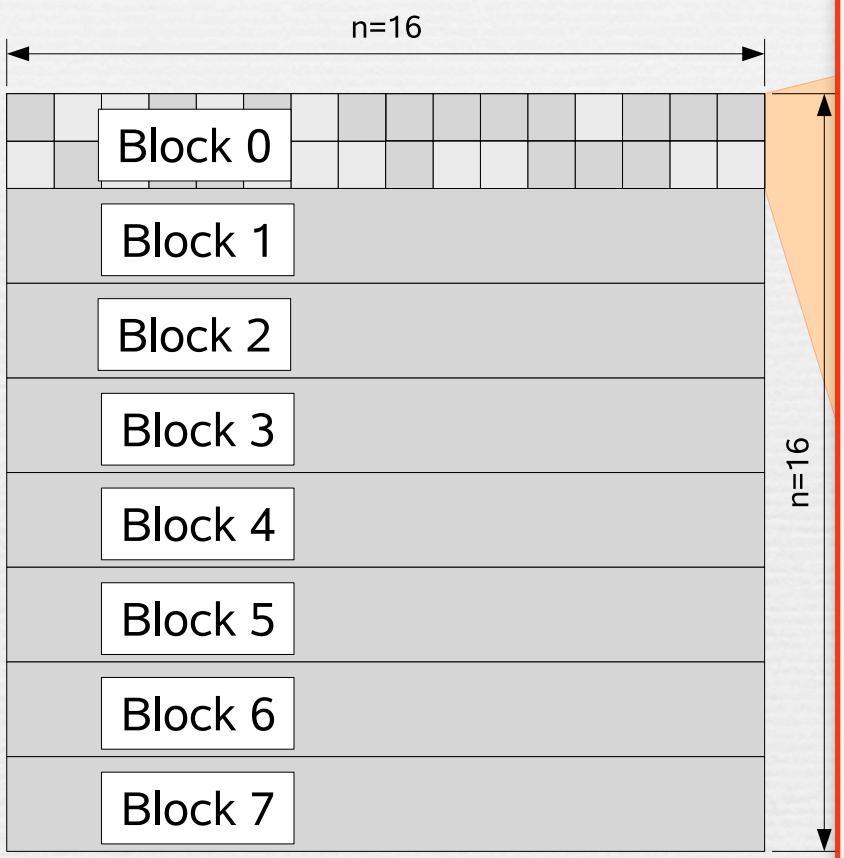


Checkerboard algoritm

Noninteracting domains where Monte Carlo moves are performed in parallel

2D Ising Model – Code

Source code available: www.tobiaspreis.de




```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <time.h>
#include <cutil.h>

#define FLAG_PRINT_SPINS 0
#define FLAG_ENERGY 0
#define T_START 3.00
#define T_FACTOR 0.9
#define T_END 2.00
#define GLOBAL_ITERATIONS 100
#define RANDOM_A 1664525
#define RANDOM_B 1013904223

#define BLOCK_SIZE 256

const unsigned int N=4*BLOCK_SIZE*BLOCK_SIZE;
const unsigned int n=2*BLOCK_SIZE;
```

Threads per Block

```
//Init
CUT_DEVICE_INIT(argc,argv);
srand48(23);

//Allocate and init host memory for output arrays
int num_entries=0;
for(double t=T_START; t>=T_END; t=t*T_FACTOR) num_entries++;
unsigned int mem_out_size=sizeof(float)*num_entries;
float* h_T=(float*) malloc(mem_out_size);
float* h_E=(float*) malloc(mem_out_size);
unsigned int mem_ref_out_size=sizeof(double)*num_entries;
double* h_ref_E=(double*) malloc(mem_ref_out_size);
num_entries=0;
for(double t=T_START; t>=T_END; t=t*T_FACTOR) {
  h_T[num_entries]=t;
  num_entries++;
}

//Allocate and init host memory for simulation arrays
unsigned int mem_size=sizeof(int)*N;
unsigned int mem_size_random=sizeof(int)*BLOCK_SIZE*BLOCK_SIZE;
int* h_random_data=(int*) malloc(mem_size_random);
int* h_S=(int*) malloc(mem_size);
unsigned int mem_size_out=sizeof(int)*BLOCK_SIZE;
int* h_out=(int*) malloc(mem_size_out);
h_random_data[0]=1;
for(int i=1;i<BLOCK_SIZE*BLOCK_SIZE;i++) {
  h_random_data[i]=16807*h_random_data[i-1];
}
for(int i=0;i<N;i++) {
  if(drand48()>0.5) h_S[i]=-1;
  else h_S[i]=1;
}
```

Initialization of GPU

Initialization of Spin Lattice

```
//Create and start timer
float gpu_sum=0;
unsigned int timer=0;
CUDA_SAFE_CALL(cudaThreadSynchronize());
CUT_SAFE_CALL(cutCreateTimer(&timer));
CUT_SAFE_CALL(cutStartTimer(timer));
```

Start timer

```
//Allocate device memory for arrays
int* d_random_data;
int* d_S;
int* d_out;
CUDA_SAFE_CALL(cudaMalloc((void**) &d_random_data,mem_size_random));
CUDA_SAFE_CALL(cudaMalloc((void**) &d_S,mem_size));
CUDA_SAFE_CALL(cudaMalloc((void**) &d_out,mem_size_out));
```

Memory Allocation

```
//Stop and destroy timer
CUDA_SAFE_CALL(cudaThreadSynchronize());
CUT_SAFE_CALL(cutStopTimer(timer));
float gpu_dt_malloc=cutGetTimerValue(timer);
gpu_sum+=gpu_dt_malloc;
printf("\n ---------------------------- GPU ---------------------------- \n");
printf(" Processing time on GPU for allocating: %f (ms) \n",gpu_dt_malloc);
CUT_SAFE_CALL(cutDeleteTimer(timer));
```

Stop timer

2D Ising Model – Code

Memory Transfer

```
//Copy host memory to device and create mirror of d_S
CUDA_SAFE_CALL(cudaMemcpy(d_random_data,h_random_data,mem_size_random,cudaMemcpyHostToDevice));
CUDA_SAFE_CALL(cudaMemcpy(d_S,h_S,mem_size,cudaMemcpyHostToDevice));

...

//Print spins
if(FLAG_PRINT_SPINS) {
  CUDA_SAFE_CALL(cudaMemcpy(h_S,d_S,mem_size,cudaMemcpyDeviceToHost));
  for(int i=0;i<BLOCK_SIZE*2;i++) {
    for(int j=0;j<BLOCK_SIZE*2;j++) {
      if(h_S[i*BLOCK_SIZE*2+j]>0) printf("+ ");
      else printf("- ");
    }
    printf("\n");
  }
  printf("\n");
}
```

Host to Device

Device to Host

2D Ising Model – Code

GPU Kernel Calls

```
//Calc energy
num_entries=0;
dim3 threads(BLOCK_SIZE);
dim3 grid(BLOCK_SIZE);
for(float t=T_START;t>=T_END;t=t*T_FACTOR) {
  double avg_H=0;
  for(int global_iteration=0;global_iteration<GLOBAL_ITERATIONS;global_iteration++) {
    device_function_main<<<grid,threads>>>(d_S,d_out,d_random_data,t,true);
    device_function_main<<<grid,threads>>>(d_S,d_out,d_random_data,t,false);

    CUDA_SAFE_CALL(cudaMemcpy(h_out,d_out,mem_size_out,cudaMemcpyDeviceToHost));
    int energy_sum=0;
    for(int i=0;i<BLOCK_SIZE;i++) energy_sum+=h_out[i];
    avg_H+=(float)energy_sum/N;
  }
  h_E[num_entries]=avg_H/GLOBAL_ITERATIONS;
  num_entries++;
}
```

2D Ising Model – Code

```
//Reference solution
cpu_function(h_ref_E,h_S);

...

//Cleaning memory
free(h_T);
free(h_E);
free(h_ref_E);
free(h_random_data);
free(h_S);
free(h_out);
CUDA_SAFE_CALL(cudaFree(d_random_data));
CUDA_SAFE_CALL(cudaFree(d_S));
CUDA_SAFE_CALL(cudaFree(d_out));
}
```

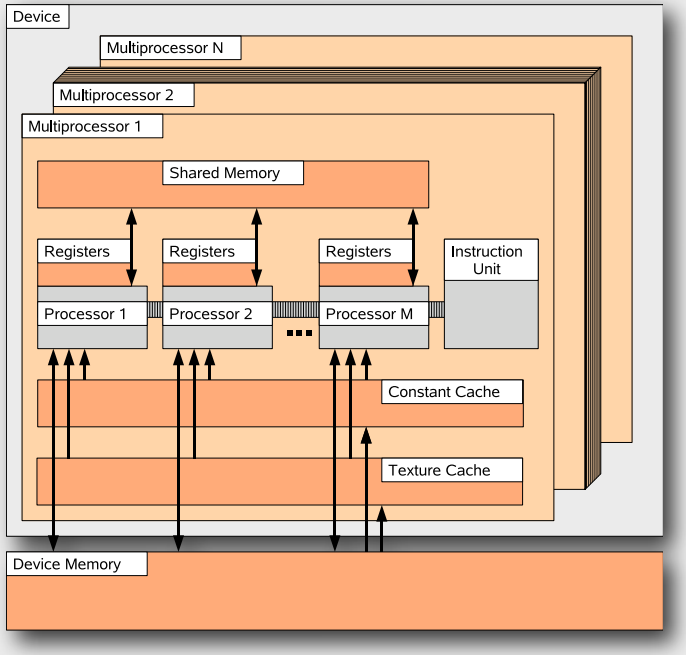
CPU Function

Clean GPU Memory

2D Ising Model – Code

GPU Kernel

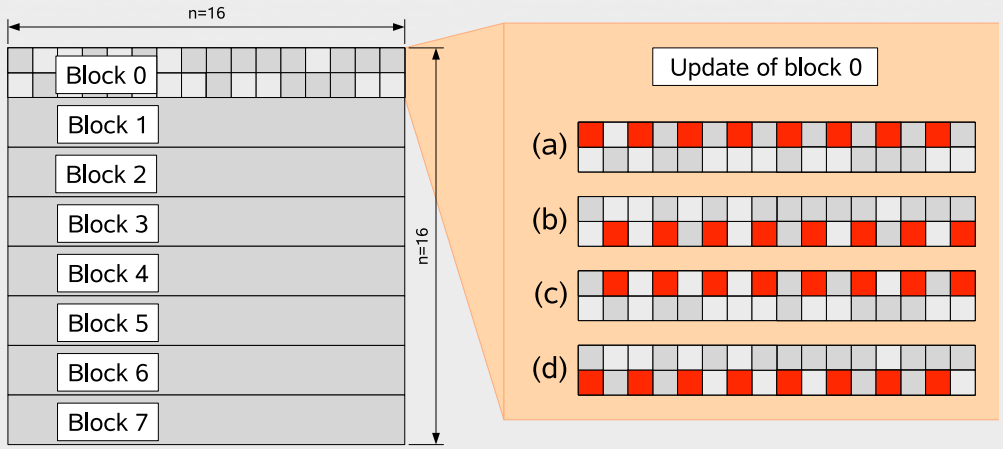
```
/****
 *
 *  Device function main
 *
 */
__global__ void device_function_main(int* S,int* out,int* R,float t,bool flag) {

  //Energy variable
  int dH=0;
  float exp_dH_4=exp(-(4.0)/t);
  float exp_dH_8=exp(-(8.0)/t);

  //Allocate shared memory
  __shared__ int r[BLOCK_SIZE];

  //Load random data
  r[threadIdx.x]=R[threadIdx.x+BLOCK_SIZE*blockIdx.x];
  __syncthreads();

  if(flag) {

    ...
```



GPU Kernel

2D Ising Model – Code

```
//Create new random numbers
r[threadIdx.x]=RANDOM_A*r[threadIdx.x]+RANDOM_B;

//Spin update top left
if(blockIdx.x==0) { //Top
  if(threadIdx.x==0) { //Left
    dH=2*S[2*threadIdx.x]*(
                    S[2*threadIdx.x+1]+
                    S[2*threadIdx.x-1+2*BLOCK_SIZE]+
                    S[2*threadIdx.x+2*BLOCK_SIZE]+
                    S[2*threadIdx.x+N-2*BLOCK_SIZE]);
  }
  else {
    dH=2*S[2*threadIdx.x]*(
                    S[2*threadIdx.x+1]+
                    S[2*threadIdx.x-1]+
                    S[2*threadIdx.x+2*BLOCK_SIZE]+
                    S[2*threadIdx.x+N-2*BLOCK_SIZE]);
  }
}
else {
  if(threadIdx.x==0) { //Left
    dH=2*S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x]*(
                                S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x+1]+
                                S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x-1+2*BLOCK_SIZE]+
                                S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x+2*BLOCK_SIZE]+
                                S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x-2*BLOCK_SIZE]);
  }
  else {
    dH=2*S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x]*(
                                S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x+1]+
                                S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x-1]+
                                S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x+2*BLOCK_SIZE]+
                                S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x-2*BLOCK_SIZE]); }}
```

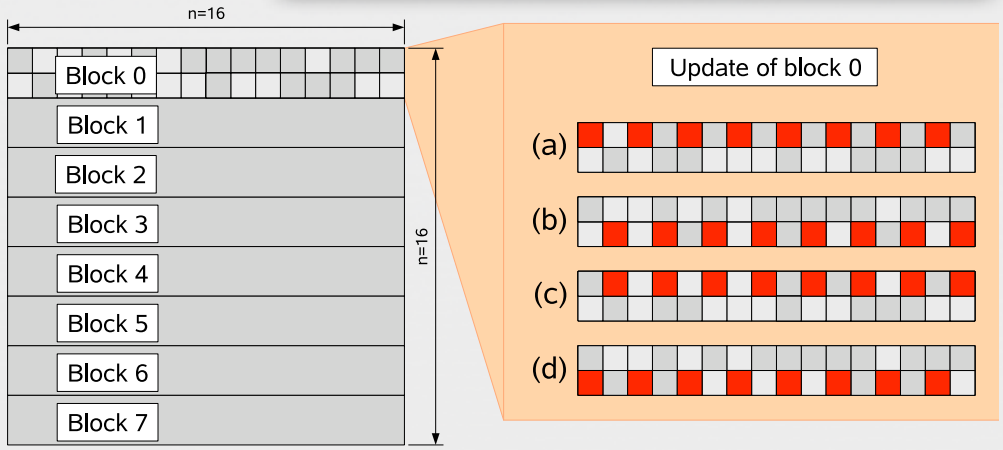


2D Ising Model – Code

GPU Kernel

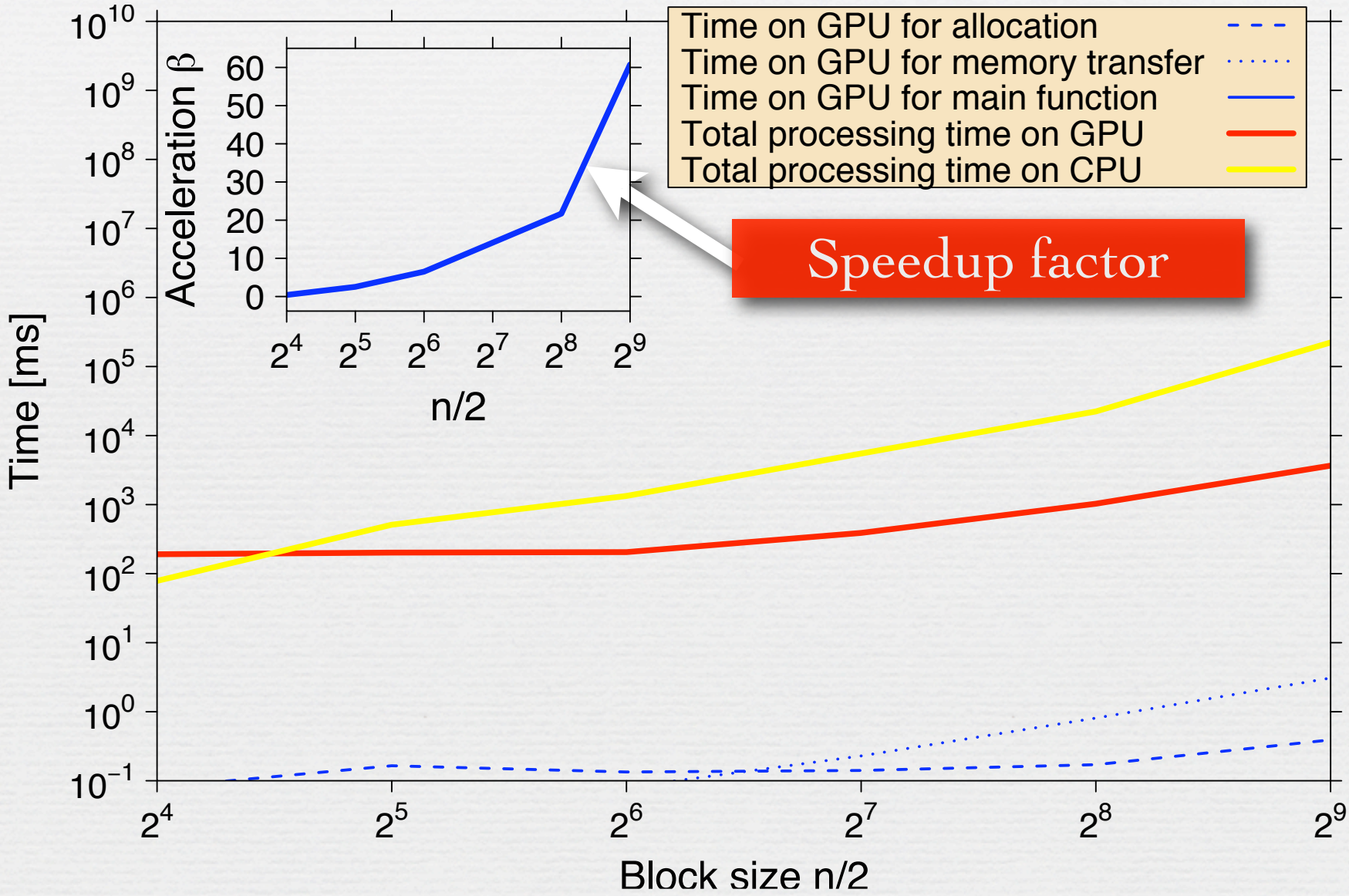
```
if(dH==4) {
  if(fabs(r[threadIdx.x]*4.656612e-10)<exp_dH_4) {
    S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x]=-S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x];
  }
}
else if(dH==8) {
  if(fabs(r[threadIdx.x]*4.656612e-10)<exp_dH_8) {
    S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x]=-S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x];
  }
}
else {
  S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x]=-S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x];
}
```

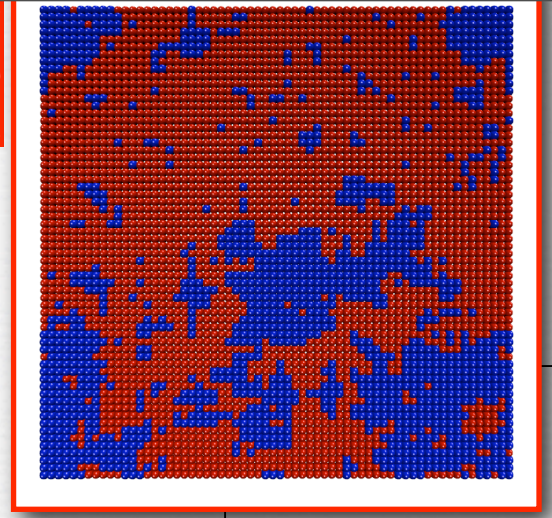
```
//Transfer random data back to global memory
R[threadIdx.x+BLOCK_SIZE*blockIdx.x]=r[threadIdx.x];

if(!flag) {

  //For reduction shared memory array r is used

    //Calc magnetisation
    dH=S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x]
       +S[2*threadIdx.x+1+4*BLOCK_SIZE*blockIdx.x]
       +S[2*threadIdx.x+4*BLOCK_SIZE*blockIdx.x+2*BLOCK_SIZE]
       +S[2*threadIdx.x+1+4*BLOCK_SIZE*blockIdx.x+2*BLOCK_SIZE];
    __syncthreads();

  //Save partial results back to shared memory in new structure
  r[threadIdx.x]=dH;

  //Reduction on GPU
  for(unsigned int dx=1;dx<BLOCK_SIZE;dx*=2) {
    if(threadIdx.x%(2*dx)==0) {
       r[threadIdx.x]+=r[threadIdx.x+dx];
    }
    __syncthreads();
  }

  //Save in out
  if(threadIdx.x==0) out[blockIdx.x]=r[0];
```



2D Ising Model – Code

Computation times – 2D





Binder cumulant – 2D

$k_B T_C = 2.269185$ J

$$U_4(T) = 1 - \frac{\langle M(T)^4 \rangle}{3\langle M(T)^2 \rangle^2}$$

Critical temperature

U_4

0.6
0.5
0.4
0.3
0.2
0.1

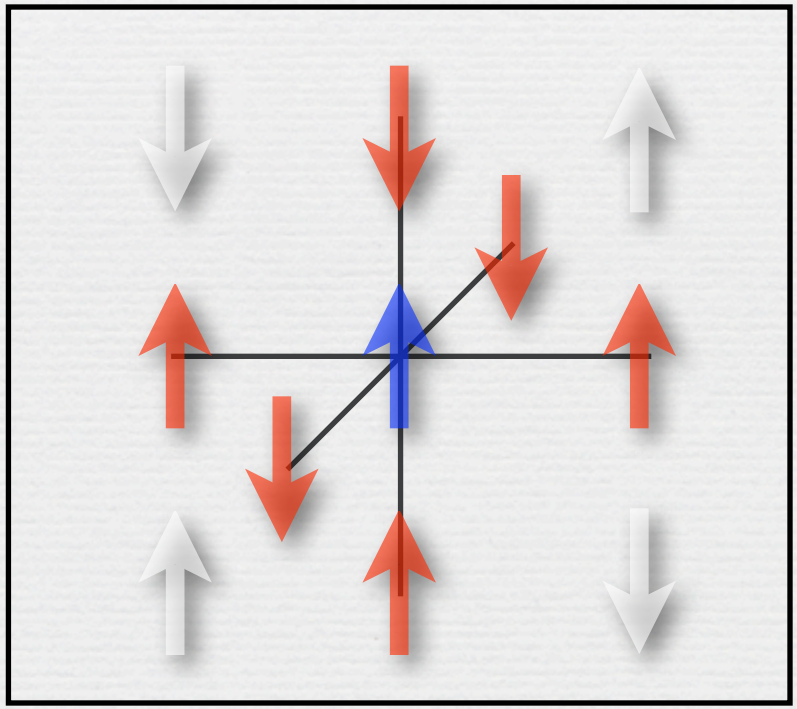
n/2 = 16
n/2 = 32
n/2 = 64
n/2 = 128
n/2 = 256

2.245 2.250 2.255 2.260 2.265 2.270 2.275 2.280 2.285 2.290 2.295

$k_B T$ [J]

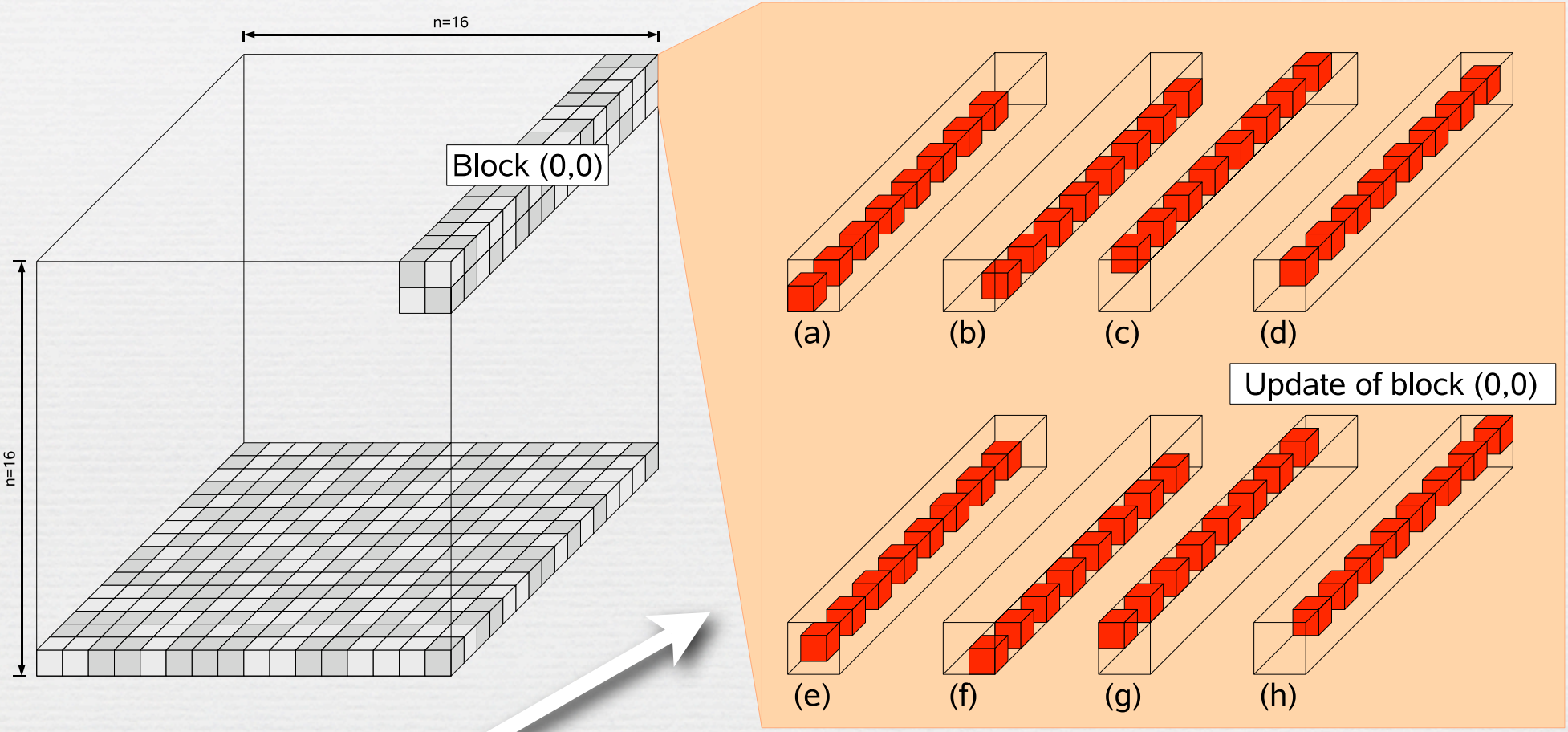
Ising model – 3D

$$\mathscr{H} = -J \sum_{\langle i,j \rangle} S_i S_j - H \sum_i S_i$$



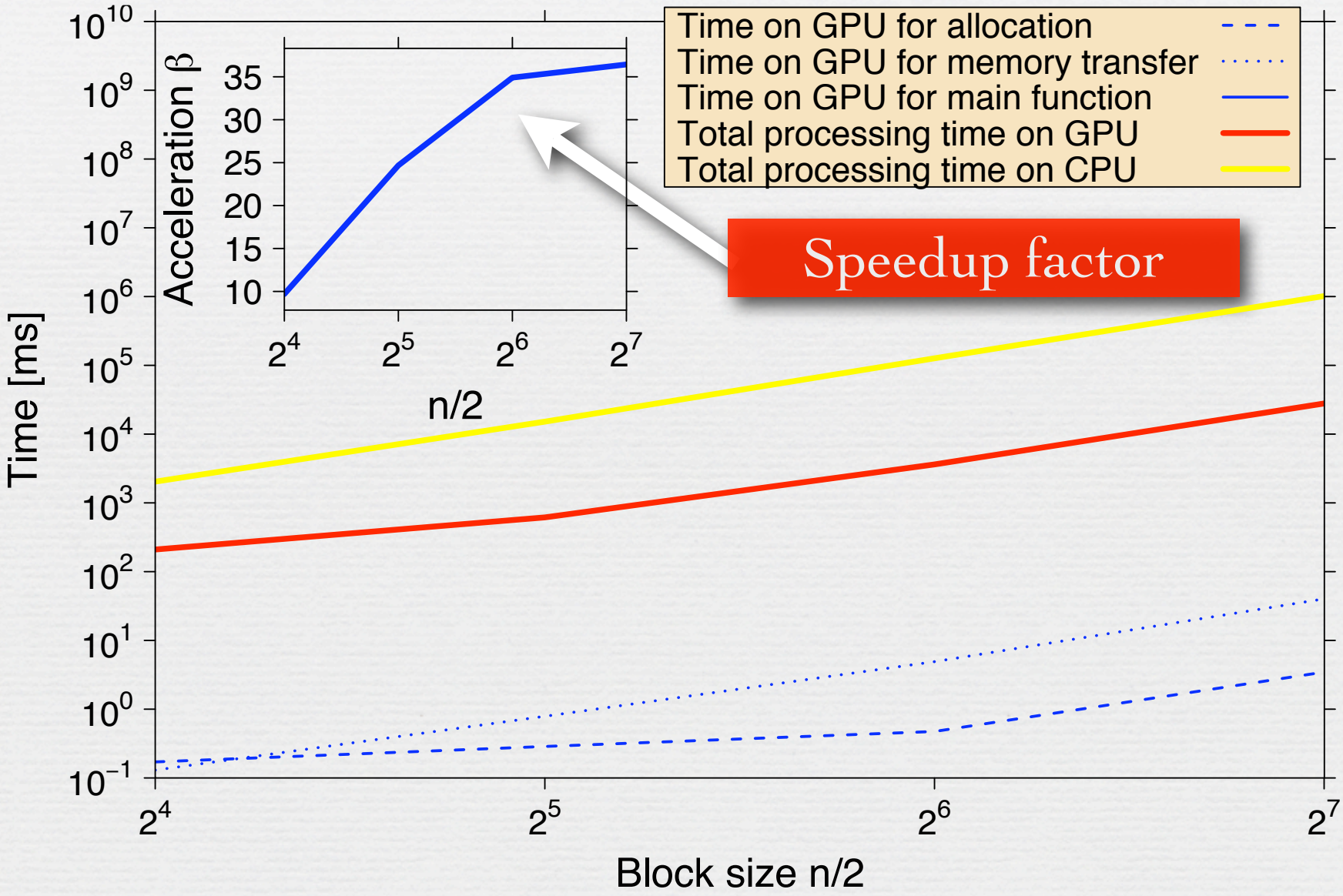
nearest neighbors

3D – Ising: GPU implementation

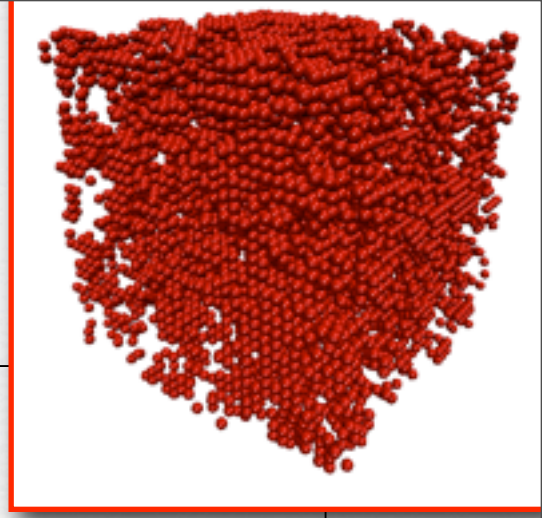


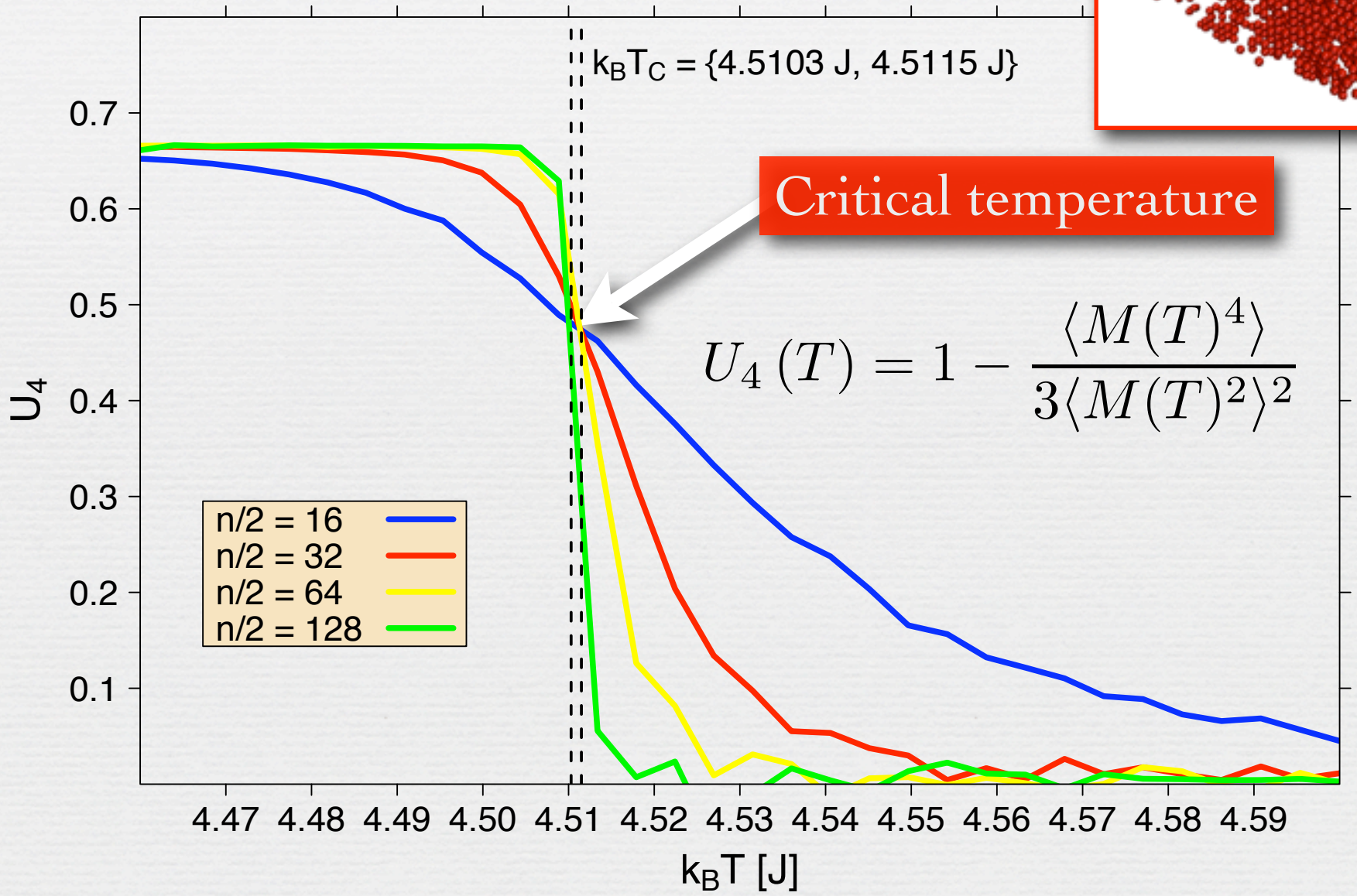
Noninteracting domains where Monte Carlo moves are performed in parallel

Computation times – 3D



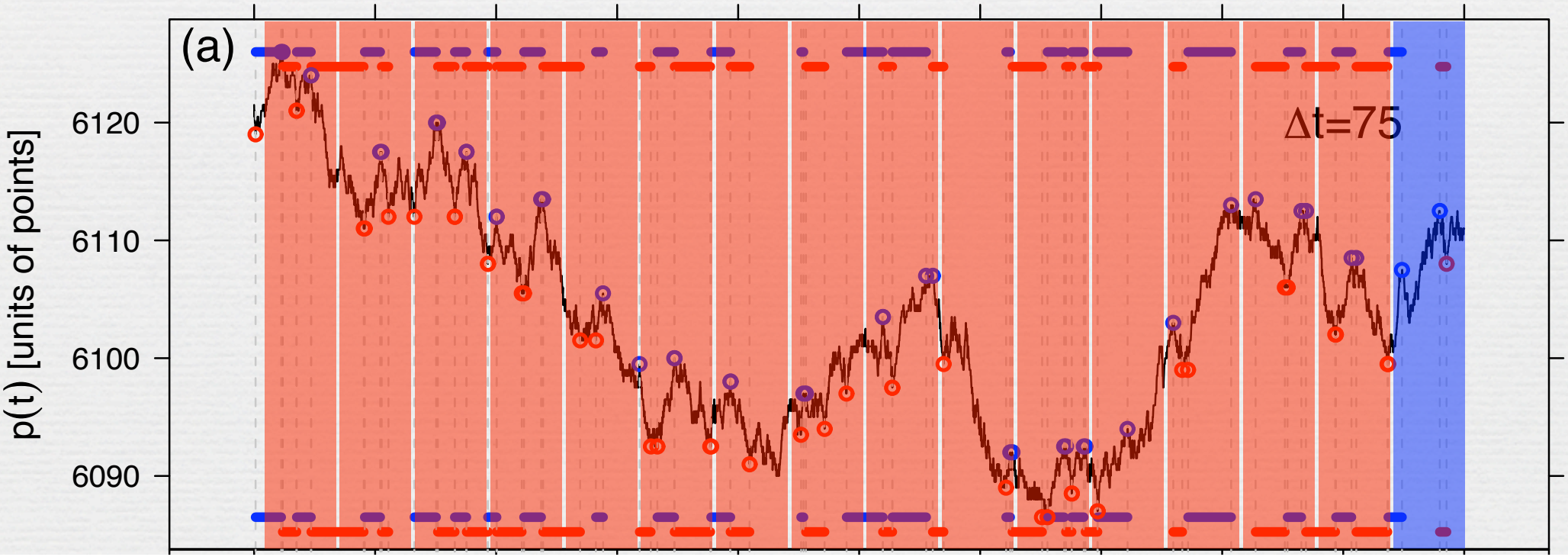
Binder cumulant – 3D





Fluctuation Patterns

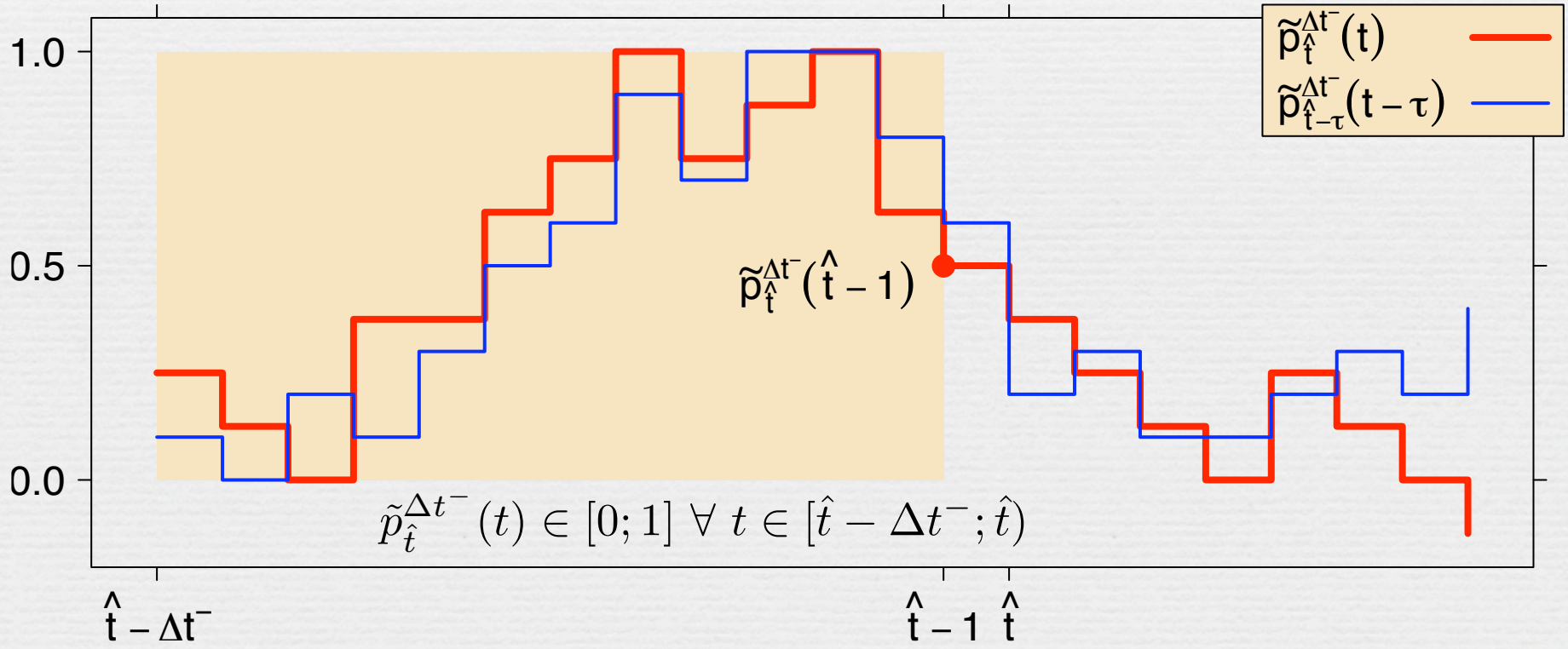
„The aim is to compare the current reference pattern of time interval length Δt^- with all previous patterns in the time series.“



Fluctuation Patterns

True range adapted modified time series $\tilde{p}_{\hat{t}}^{\Delta t^-}(t)$

$$\tilde{p}_{\hat{t}}^{\Delta t^-}(t) = \frac{p(t) - p_l(\hat{t}, \Delta t^-)}{p_h(\hat{t}, \Delta t^-) - p_l(\hat{t}, \Delta t^-)}$$

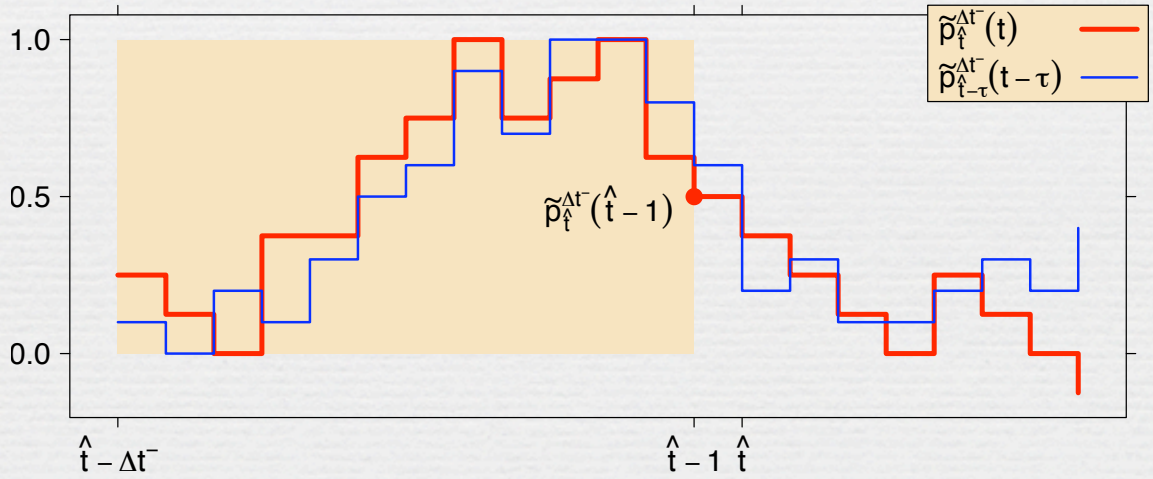


Fluctuation Patterns

Mean-square quality between current and comparison sequence

$$Q_{\hat{t}}^{\Delta t^-}(\tau) = \sum_{\theta=1}^{\Delta t^-} \frac{\left(\tilde{p}_{\hat{t}}^{\Delta t^-}(\hat{t}-\theta) - \tilde{p}_{\hat{t}-\tau}^{\Delta t^-}(\hat{t}-\tau-\theta)\right)^2}{\Delta t^-}$$

with $Q_{\hat{t}}^{\Delta t^-}(\tau) \in [0,1]$



In order to quantify the value of reference and comparison pattern relative to the reference point, one can define ...

$$\omega_{\hat{t}}^{\Delta t^-}(\tau,\Delta t^+) = \left(\tilde{p}_{\hat{t}}^{\Delta t^-}(\hat{t}-1+\Delta t^+) - \tilde{p}_{\hat{t}}^{\Delta t^-}(\hat{t}-1)\right)\cdot\left(\tilde{p}_{\hat{t}-\tau}^{\Delta t^-}(\hat{t}-\tau-1+\Delta t^+) - \tilde{p}_{\hat{t}}^{\Delta t^-}(\hat{t}-1)\right)$$

Fluctuation Patterns

Observable for pattern conformity:

$$\xi_\chi\left(\Delta t^+,\Delta t^-\right) = \sum_{\hat{t}=\Delta t^-}^{T-\Delta t^+} \sum_{\tau=\tau^*}^{\hat{t}} \frac{\operatorname{sgn}\left(\omega_{\hat{t}}^{\Delta t^-}\left(\tau,\Delta t^+\right)\right)}{\exp\left(\chi Q_{\hat{t}}^{\Delta t^-}(\tau)\right)}$$

Limitation: $\tau^* = \begin{cases} \hat{t}-\hat{\tau} & \text{if } \hat{t}-\hat{\tau}-\Delta t^- \geq 0 \\ \Delta t^- & \text{else} \end{cases}$

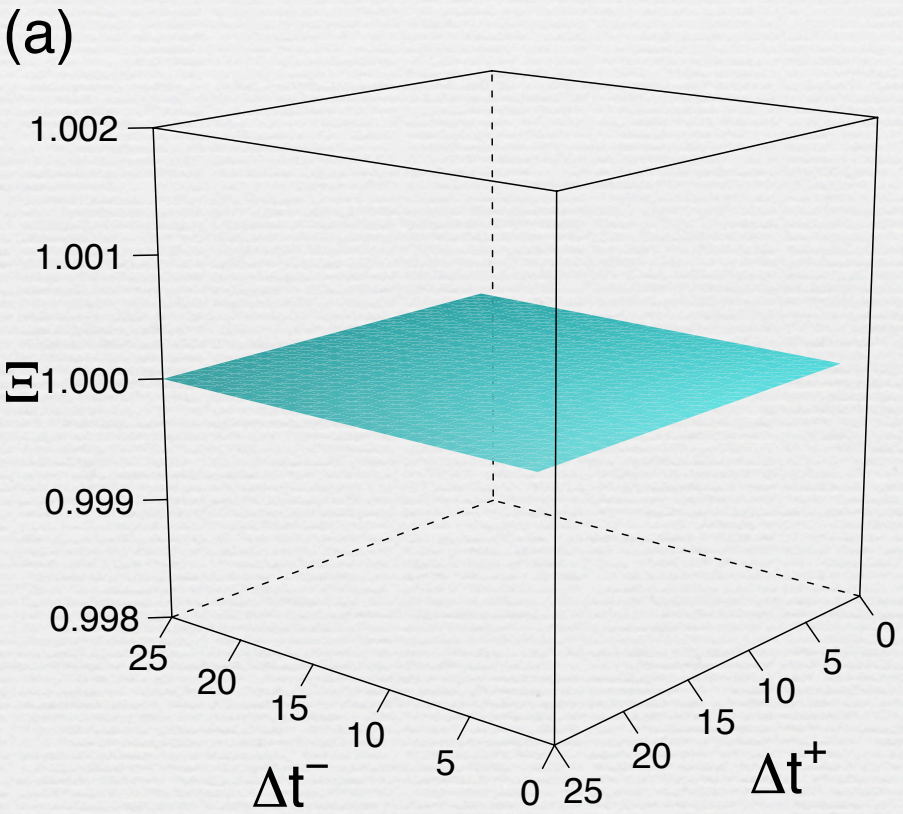
Definition: $\operatorname{sgn}(x) = \begin{cases} 1 & \text{for } x > 0 \\ 0 & \text{for } x = 0 \\ -1 & \text{for } x < 0 \end{cases}$

Normalized pattern conformity:

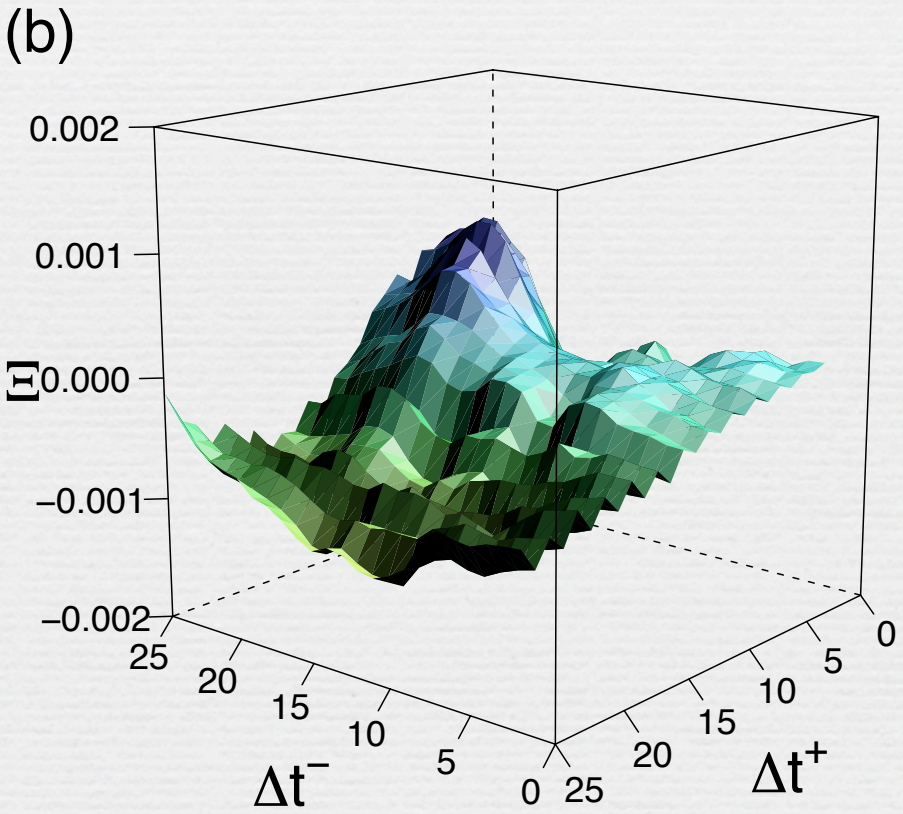
$$\Xi_\chi\left(\Delta t^+,\Delta t^-\right) = \frac{\xi_\chi\left(\Delta t^+,\Delta t^-\right)}{\sum_{\hat{t}=\Delta t^-}^{T-\Delta t^+} \sum_{\tau=\tau^*}^{\hat{t}} \frac{\left|\operatorname{sgn}\left(\omega_{\hat{t}}^{\Delta t^-}\left(\tau,\Delta t^+\right)\right)\right|}{\exp\left(\chi Q_{\hat{t}}^{\Delta t^-}(\tau)\right)}}$$

Pattern Conformity / Trivial Cases

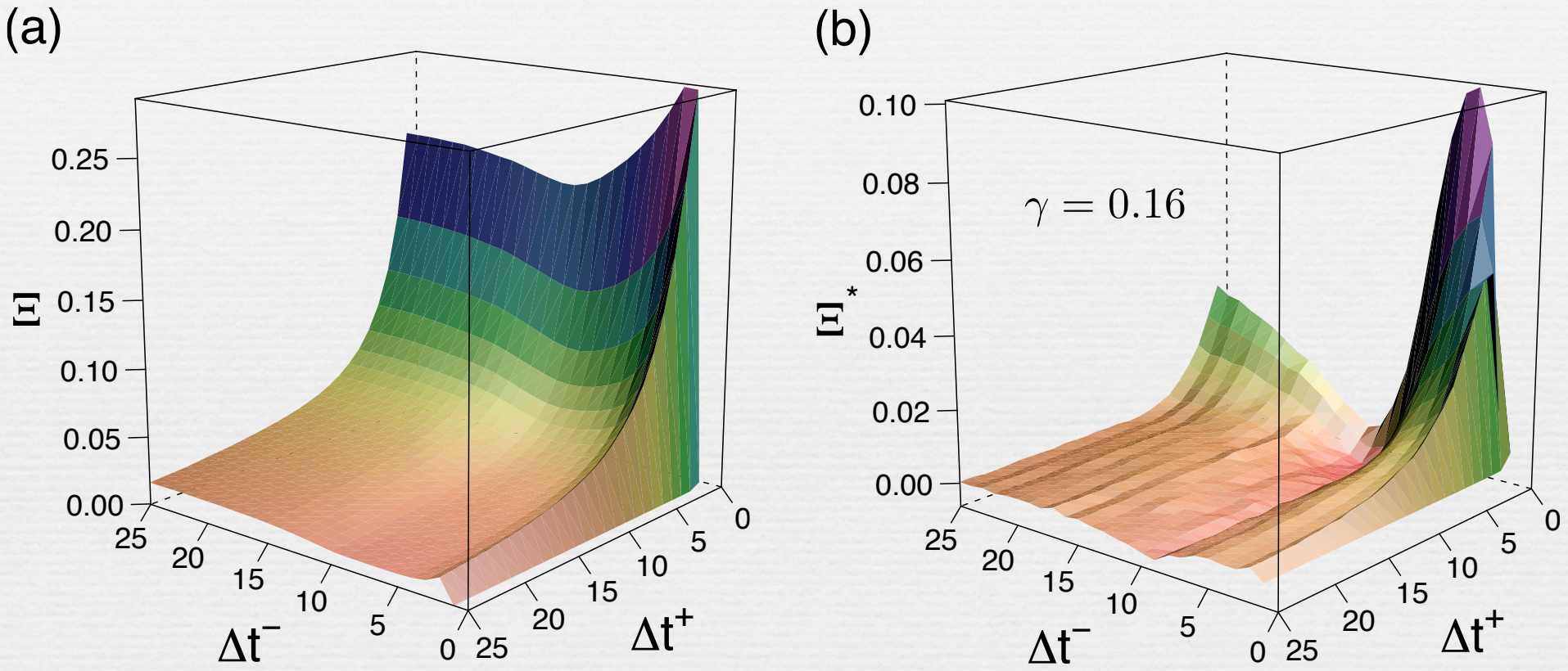
- Straight Line



- Random Walk



Pattern Conformity / FDAX



Complex correlations for financial market time series – especially for large pattern lengths.

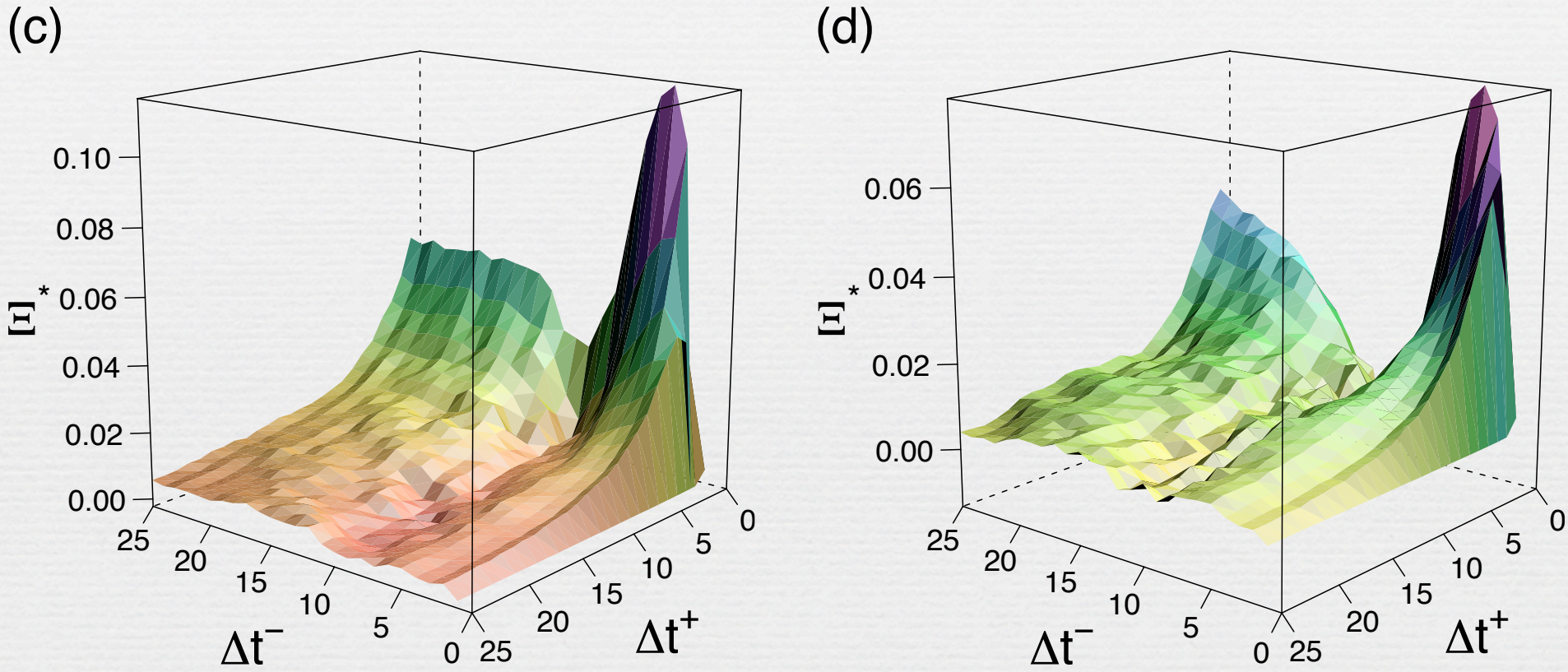
$$\Xi^{*} = \Xi^{\mathrm{FDAX}}_{\chi=100} - \Xi^{\mathrm{ACRW}}_{\chi=100}$$

$$Q_{\hat{t}}^{\Delta t^{-}}(\tau) = Q_{\hat{t}}^{p,\Delta t^{-}}(\tau)$$

Tobias Preis

Inclusion of volumes and ITWT

T. Preis et al., Europhys. Lett. **82**, 68005 (2008)

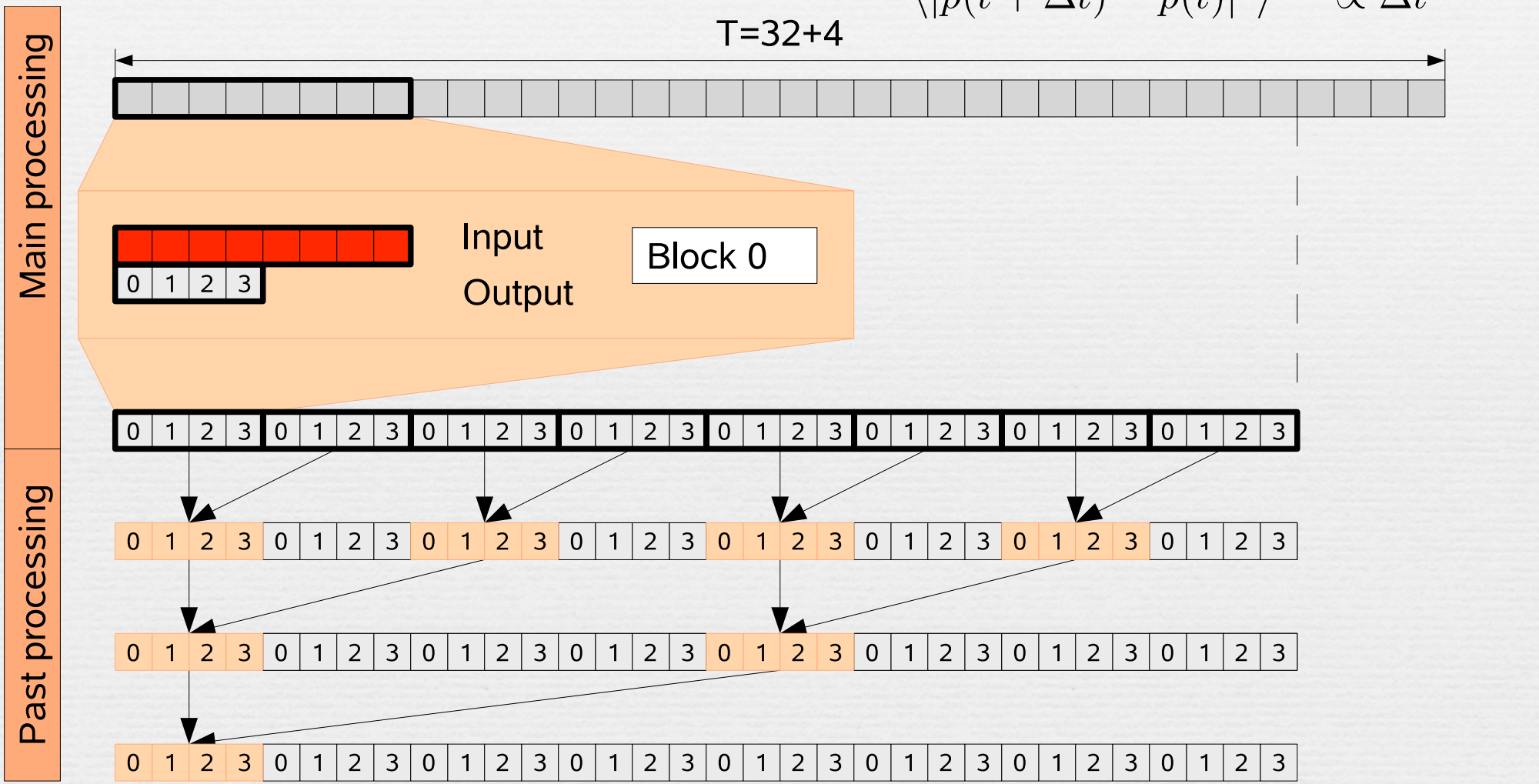


Same structure – high values of the pattern conformity

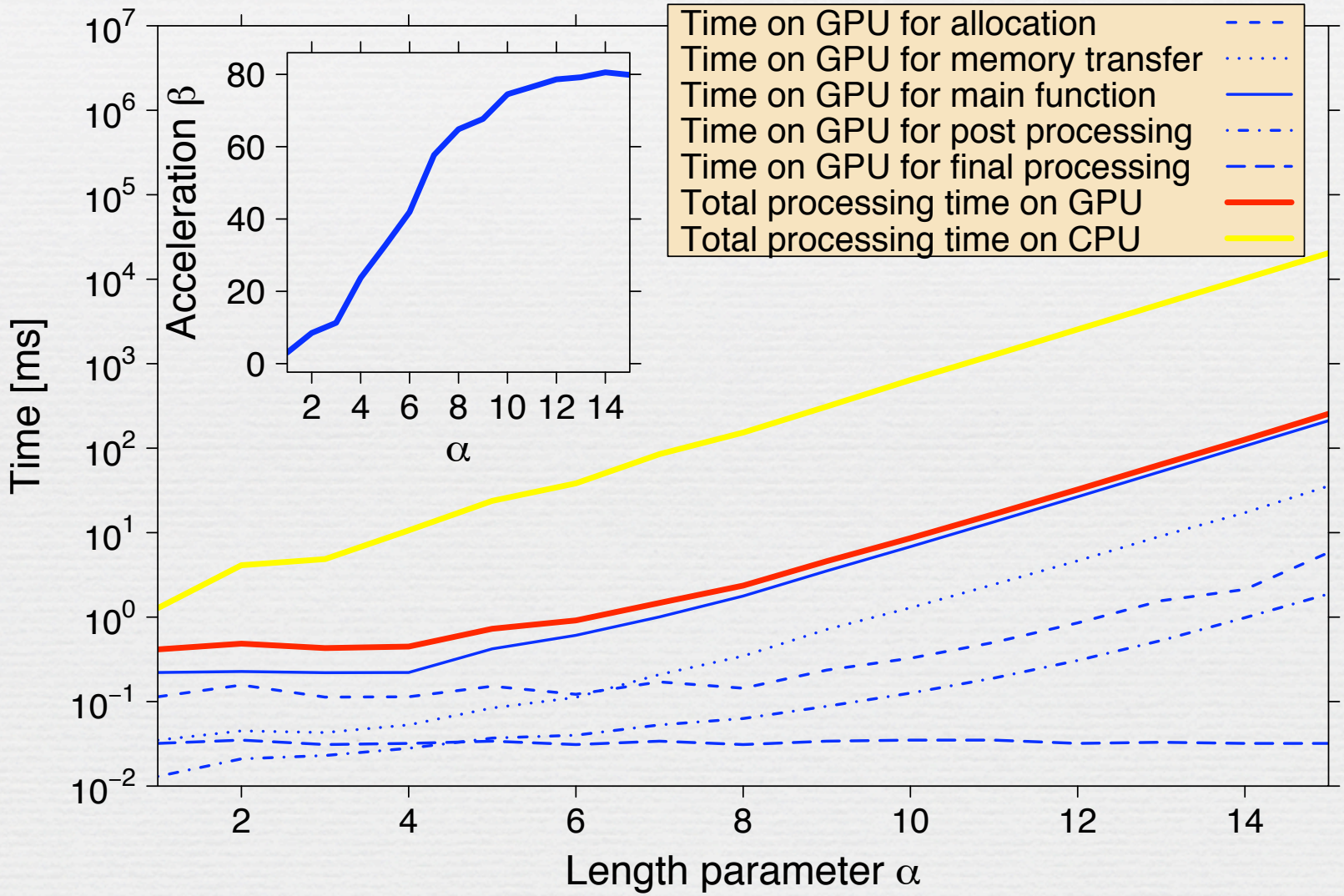
$$Q_{\hat{t}}^{\Delta t^-}(\tau) = Q_{\hat{t}}^{p,\Delta t^-}(\tau) + Q_{\hat{t}}^{v,\Delta t^-}(\tau) \qquad Q_{\hat{t}}^{\Delta t^-}(\tau) = Q_{\hat{t}}^{p,\Delta t^-}(\tau) + Q_{\hat{t}}^{\iota,\Delta t^-}(\tau)$$

GPU computing / Hurst exponent

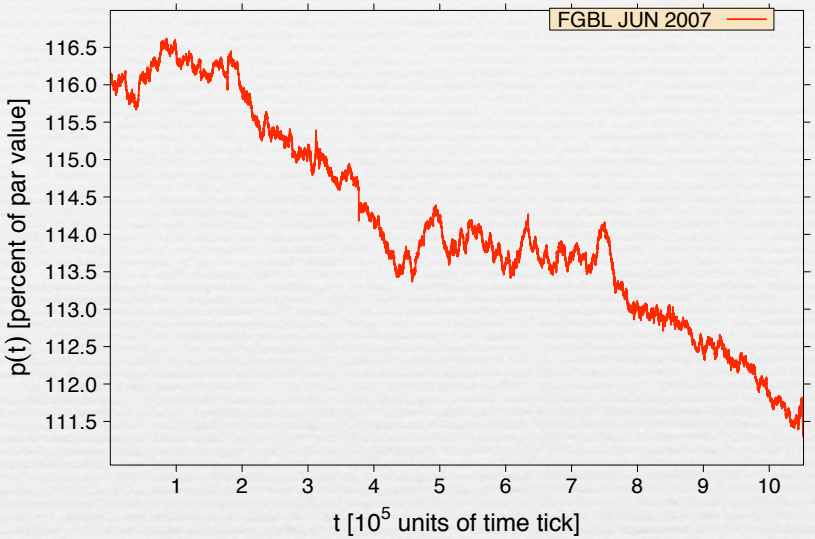
$$\langle |p(t+\Delta t) - p(t)|^q \rangle^{1/q} \propto \Delta t^{H_q(\Delta t)}$$

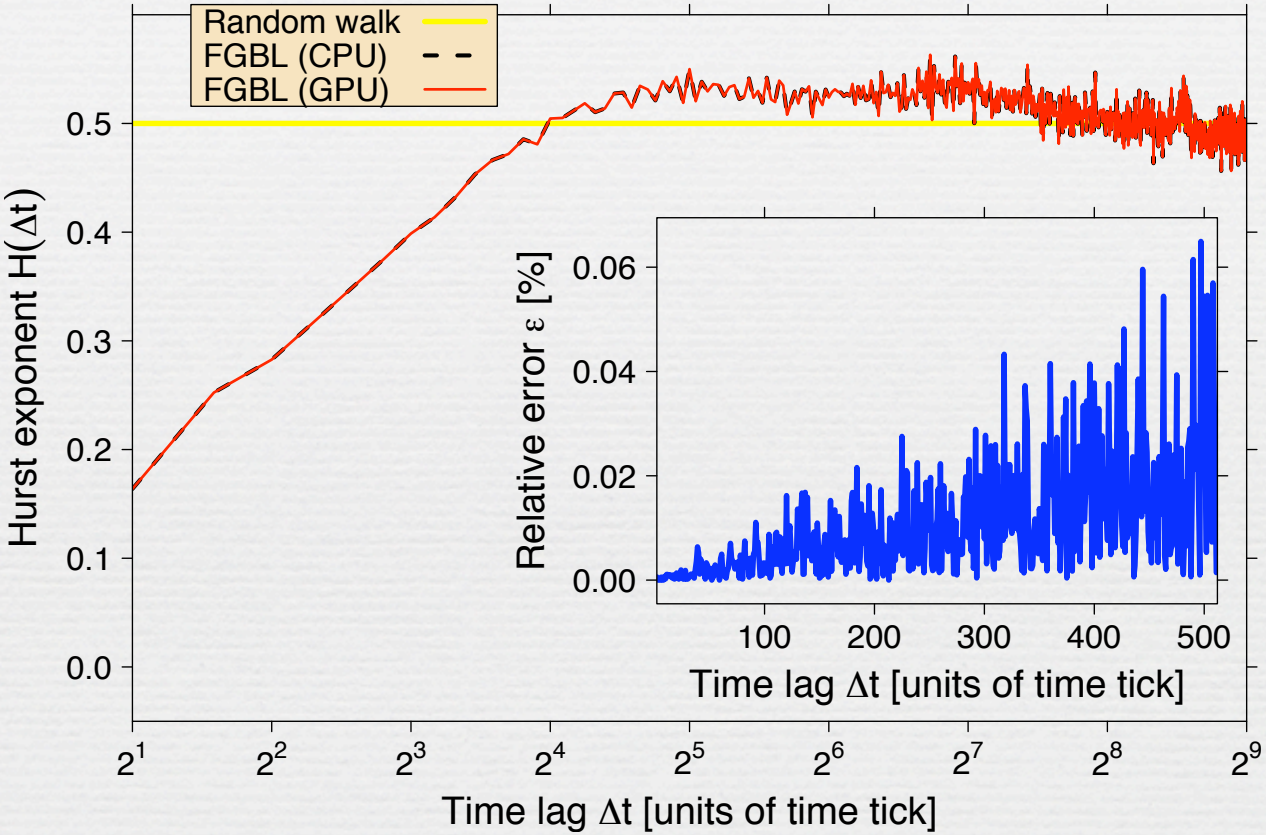


GPU computing / Hurst exponent

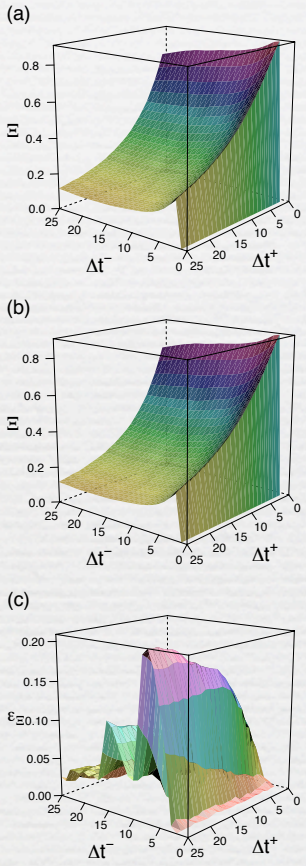


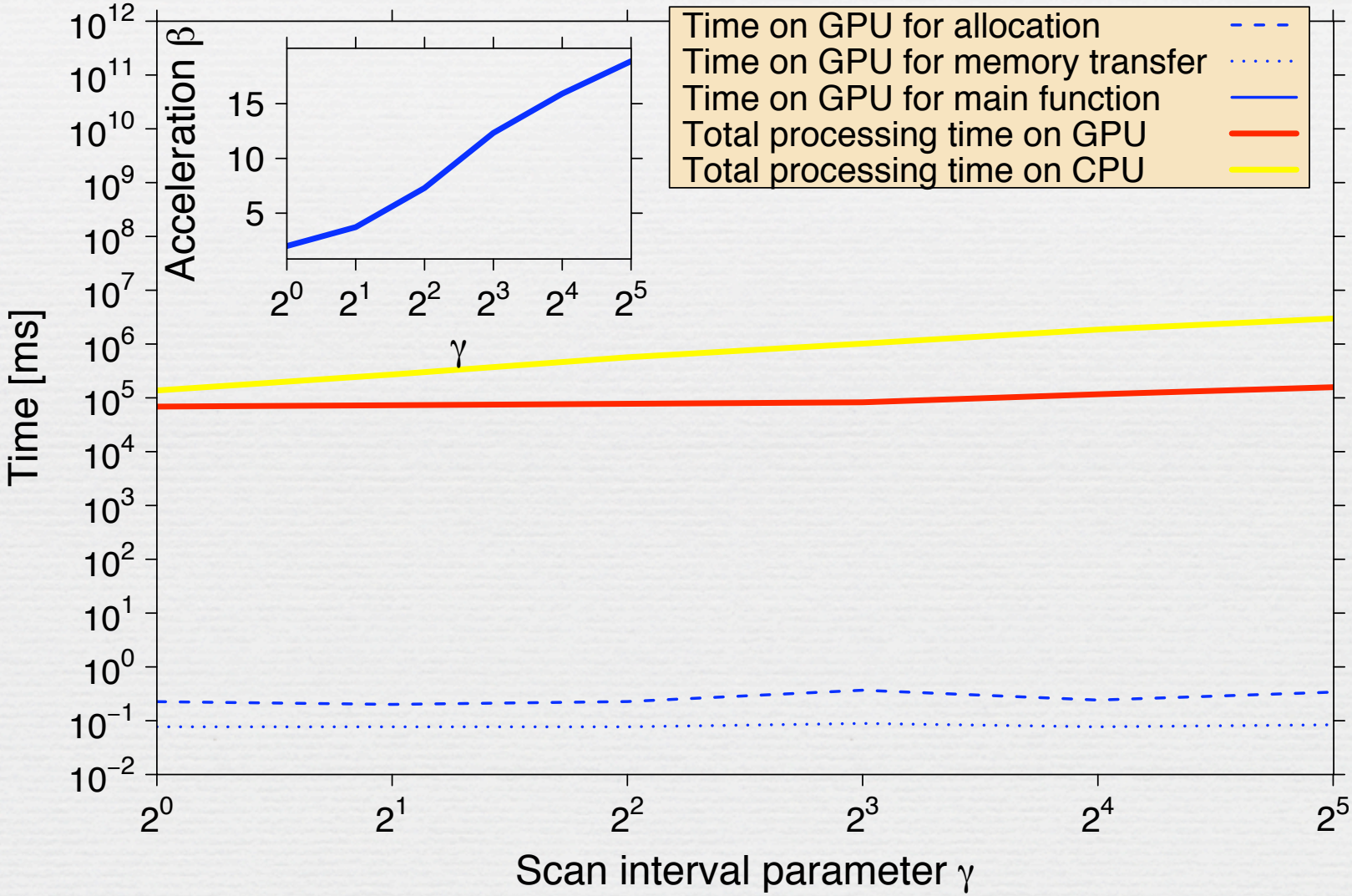
GPU computing / Hurst exponent





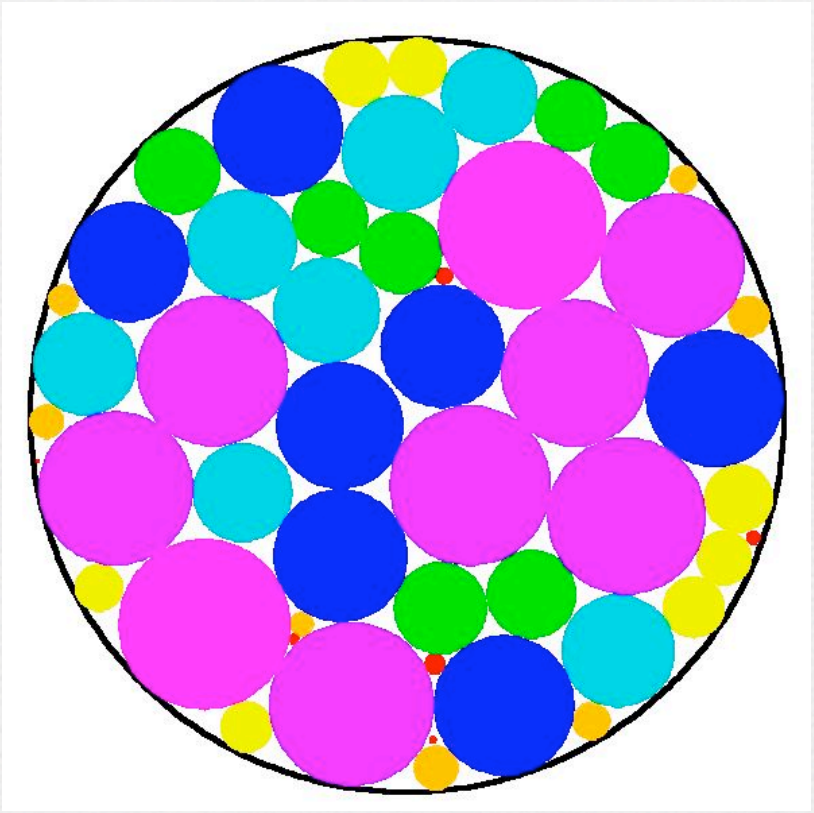
GPU computing / Pattern Conformity





Packing Problems

A. Müller, J. J. Schneider, E. Schömer, Phys. Rev. E **79**, 021102 (2009)



Packalgorithmus ausgezeichnet vom Time Magazine

Al Zimmermann: http://www.azspcs.net/

Tobias Preis

37

Packing, Improved

Johannes Schneider may not have the coolest invention on this list, but it sure is practical. The University of Mainz researcher and his team developed an algorithm that broke the record for fitting a given number of different-size discs into the smallest circle. The algorithm improves on its competitors (yes, there are competitors) in that it's better at detecting false starts and backtracking when it hits on an inelegant configuration. Schneider believes that his algorithm could benefit packaging and shipping companies by helping them use their resources more efficiently.

GPU Computing Remarks

- TP, PV, WP, and JJS, „GPU Accelerated Monte Carlo Simulation of 2D and 3D Ising Model“, J. Comp. Phys. **228**, 4468–4477(2009)
- TP, PV, WP, and JJS, „GPU Accelerated fluctuation analysis by graphic cards and complex pattern formation in financial markets“, New J. Phys. **11**, 093024(2009)
- Source code available: www.tobiaspreis.de

Milestones of trading

CBOT trading pit, Chicago, 1993

Floor trading

1602 Options on shares of East-Indian Company (NL)

1634-36 Tulip bulb speculation bubble (NL)

...

1848 CBOT: trading of future contracts (USA)

...

1978 CBOE: trading of option contracts (USA)

...

1990 DTB: futures and options (D)

...

1998 EUREX: DTB+SOFFEX (D, CH)

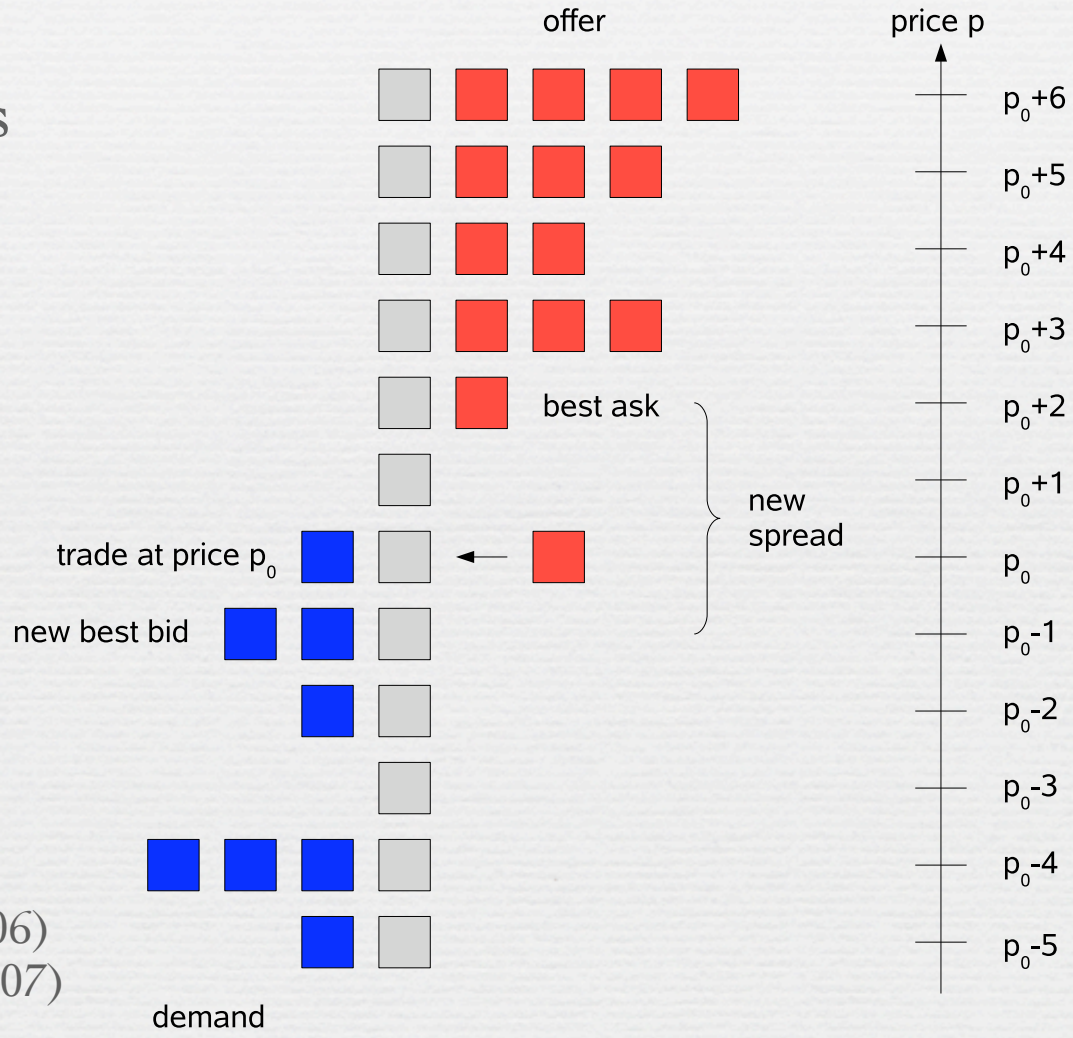
...

Electronic trading

Deutsche Bank AG headquarter, Frankfurt, 2007

Order book structure

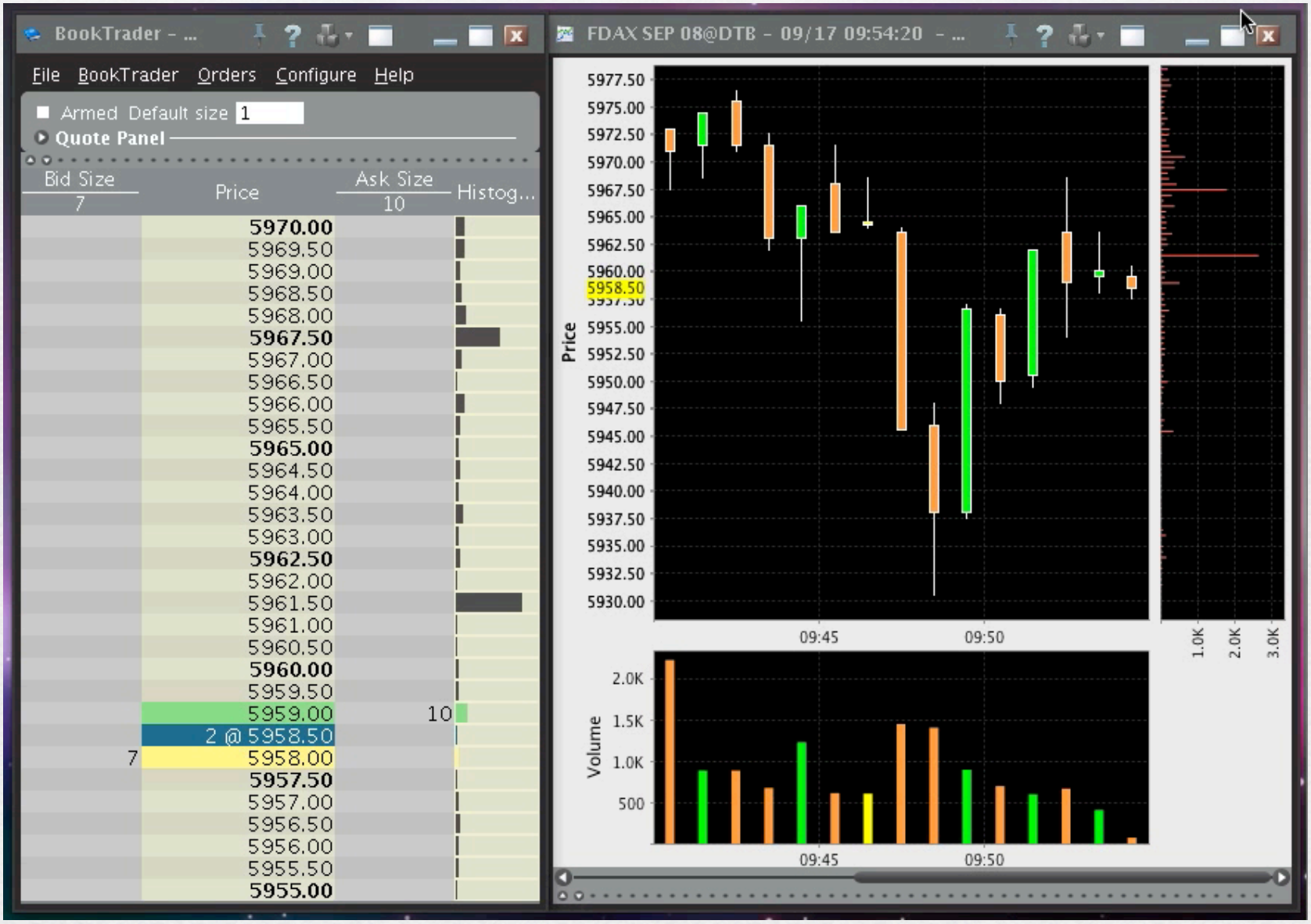
- Discrete price levels
- Limit orders
- Market orders



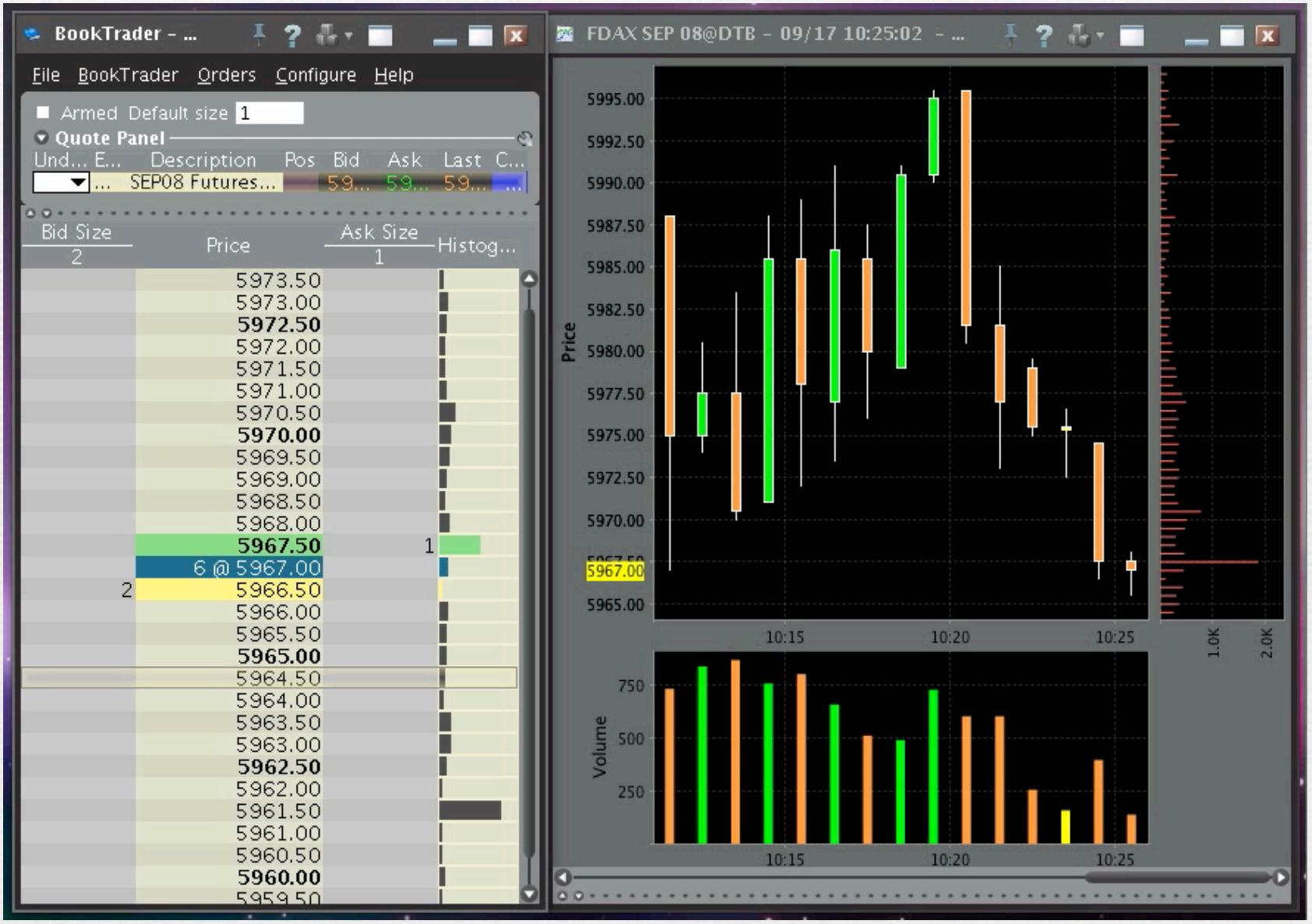
T. Preis et al., Europhys. Lett. **75**, 510 (2006)
T. Preis et al., Phys. Rev. E **76**, 016108 (2007)

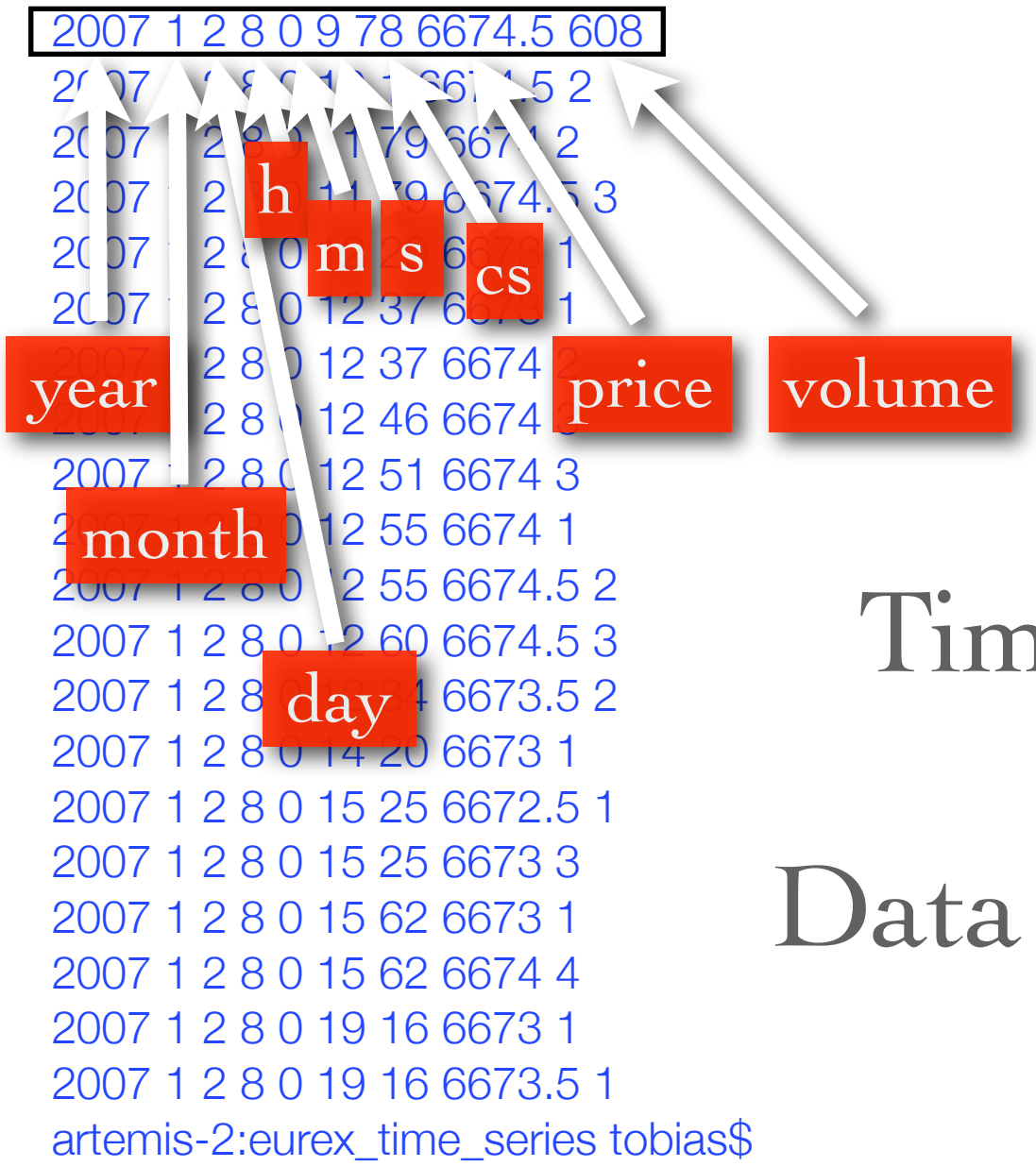
Tobias Preis

Real order book data I



Real order book data II





Time Series
–
Data Structure

Time Series – Data Sets

- Future Contracts:

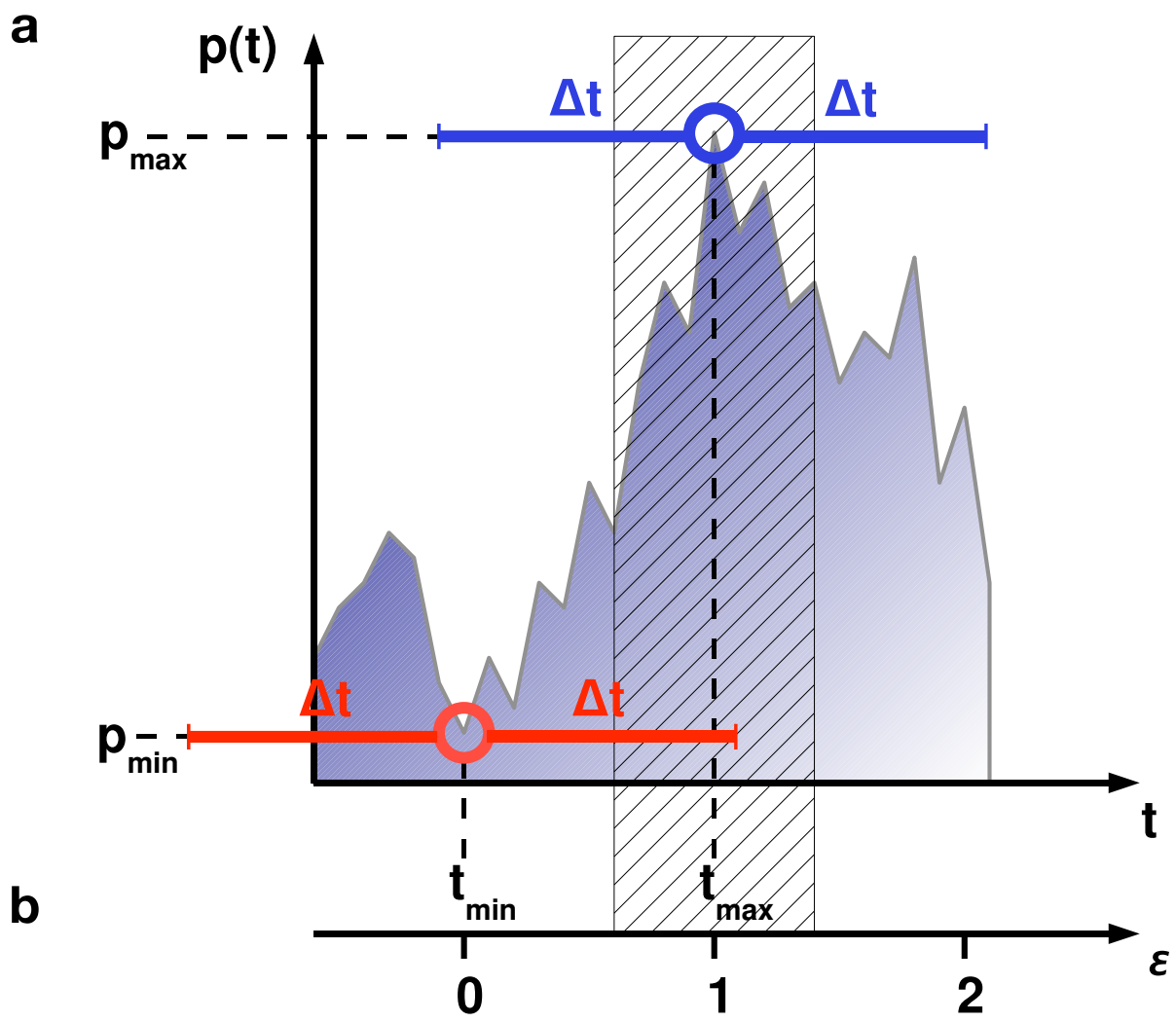
  „A future contract is a contract to buy or sell a proposed underlying asset at a specific date in the future at a specified price.“

- Advantages:

  Prices created by trading decisions for this product alone (in contrast to stock indices)

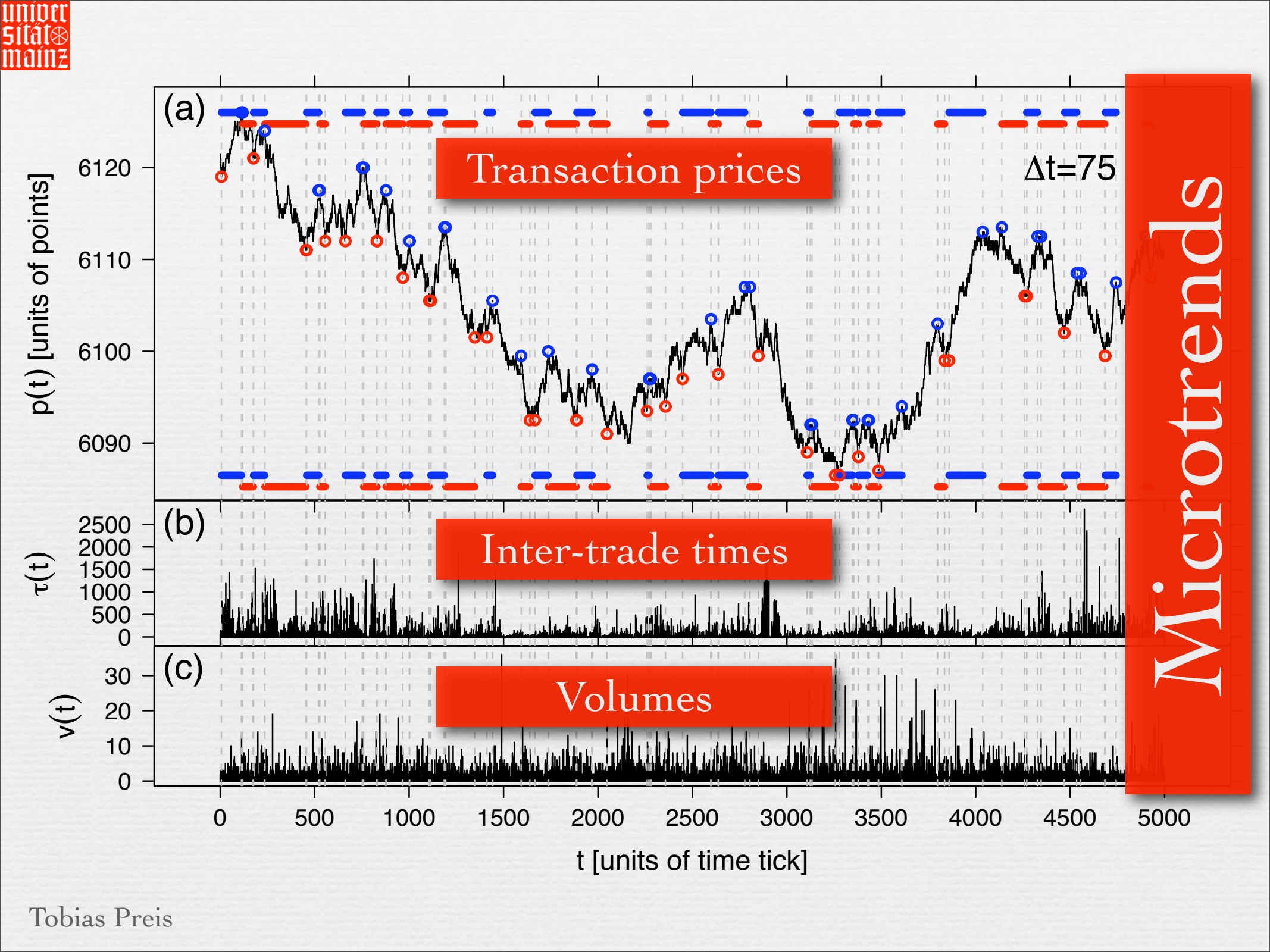
  Large liquidity, inter-trade waiting times down to centiseconds

Microtrends

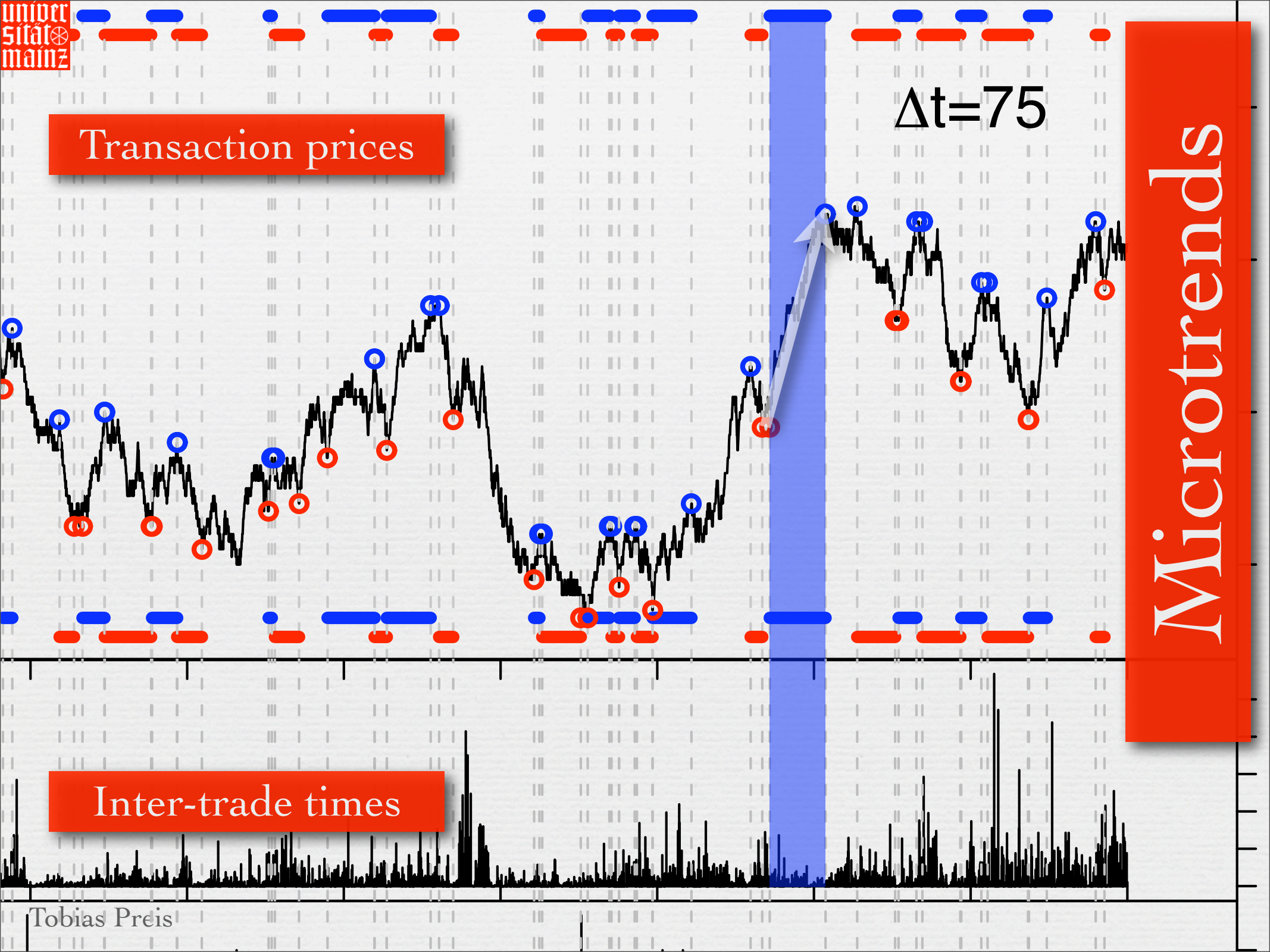


13,991,275 trades

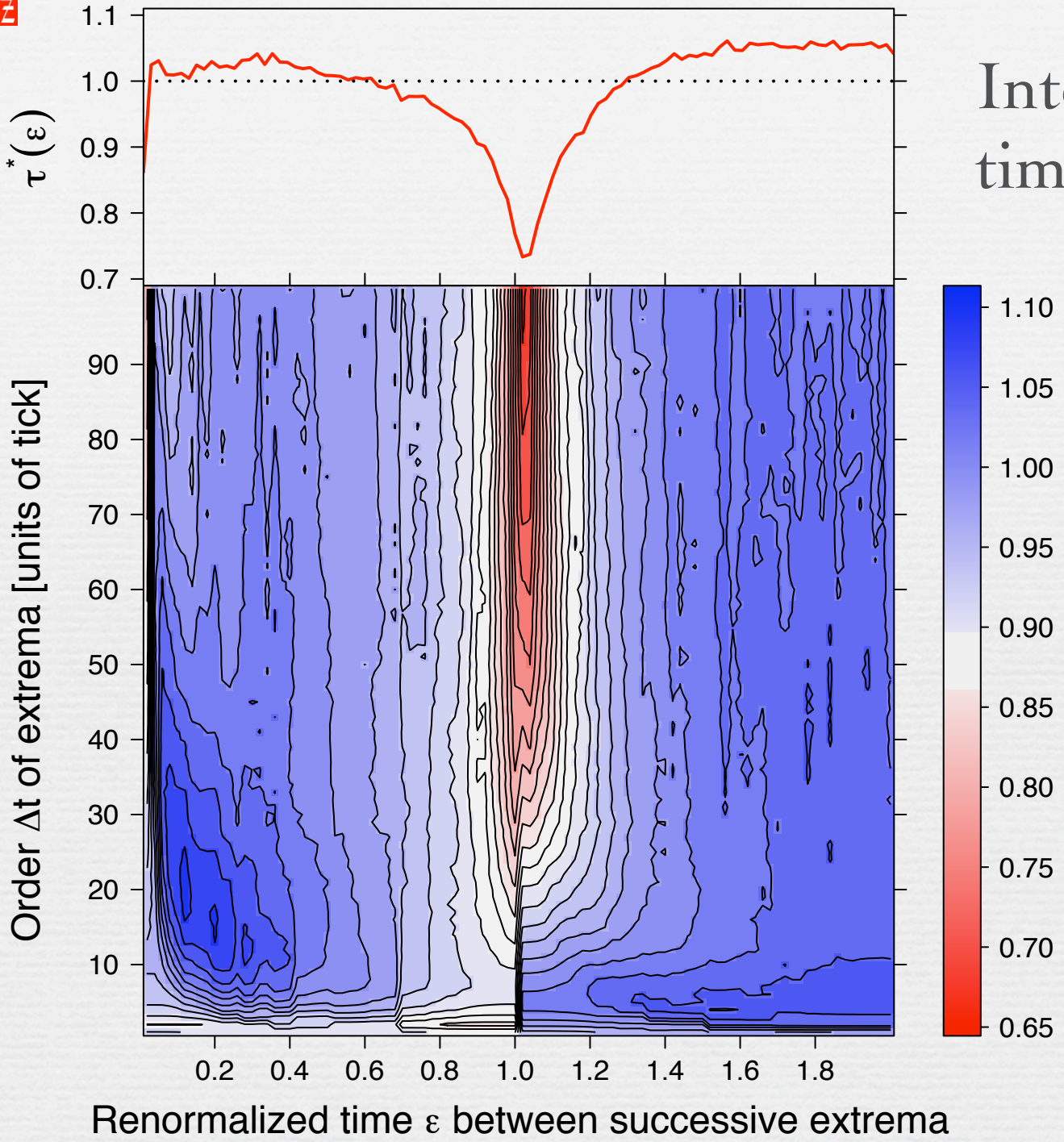
– June 2007
– September 2008
– December 2008



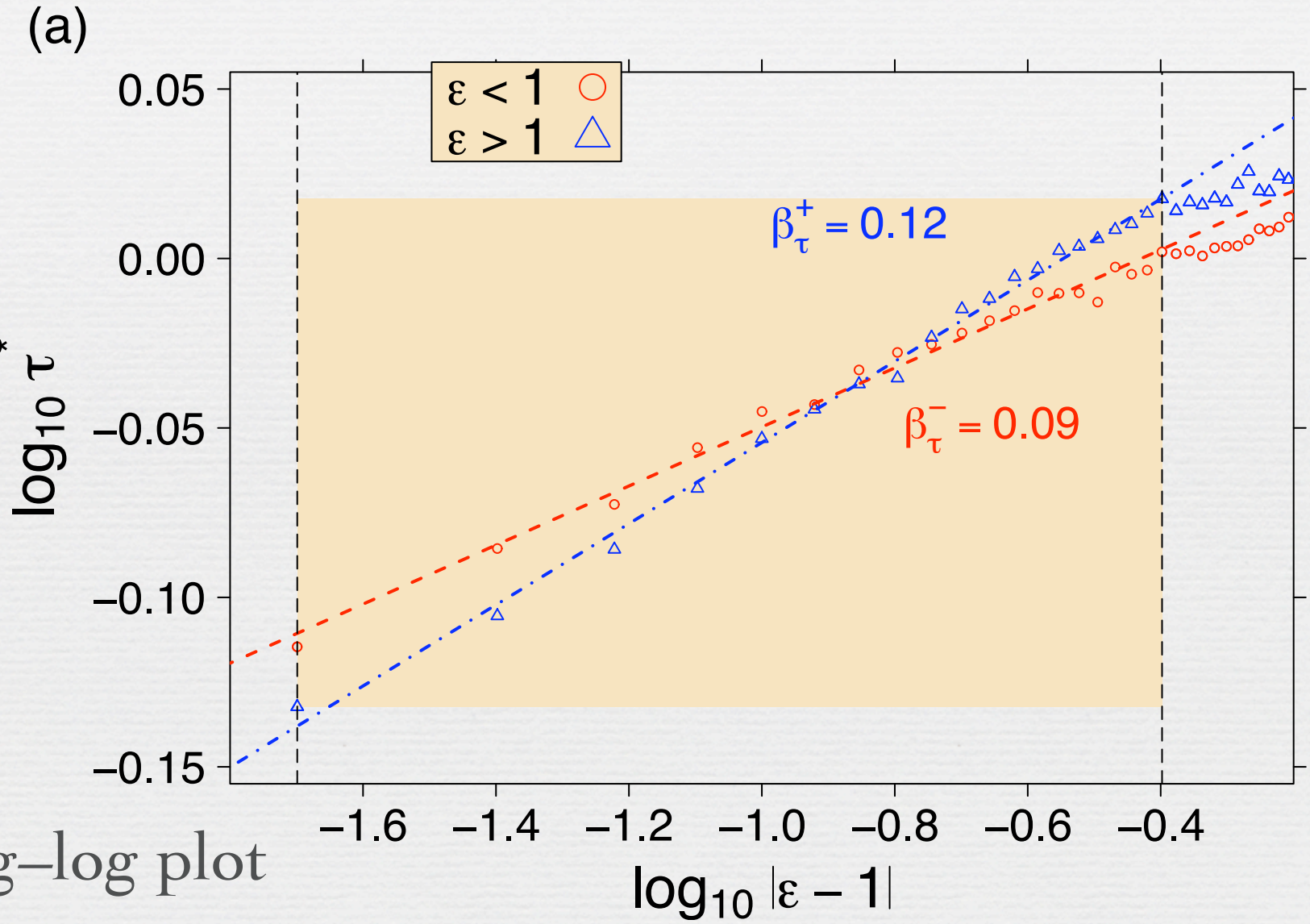
Tobias Preis


Transaction prices
Δt=75
Microtrends
Inter-trade times
Tobias Preis

Inter-trade waiting time profile

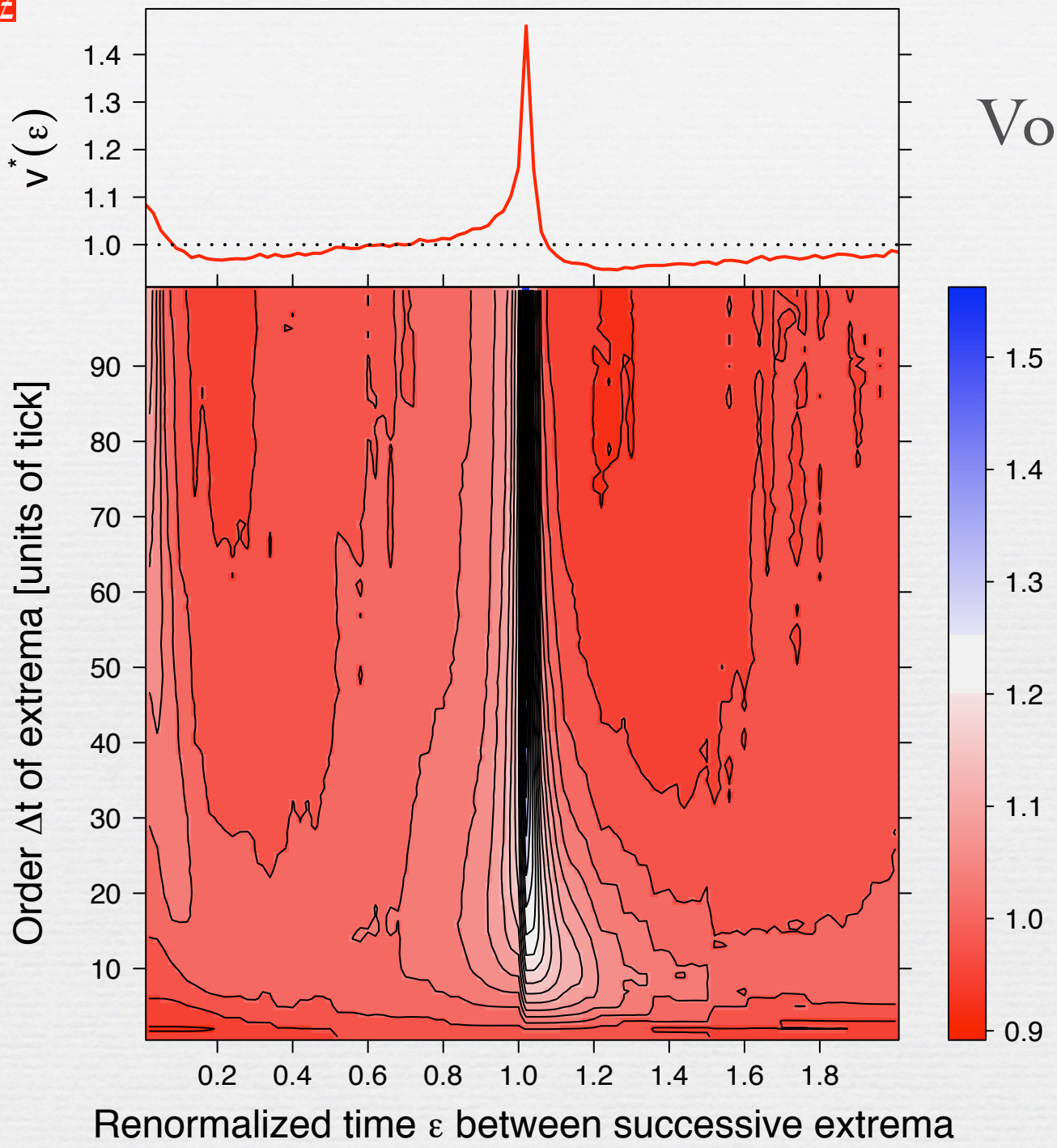


Inter-trade waiting times profile

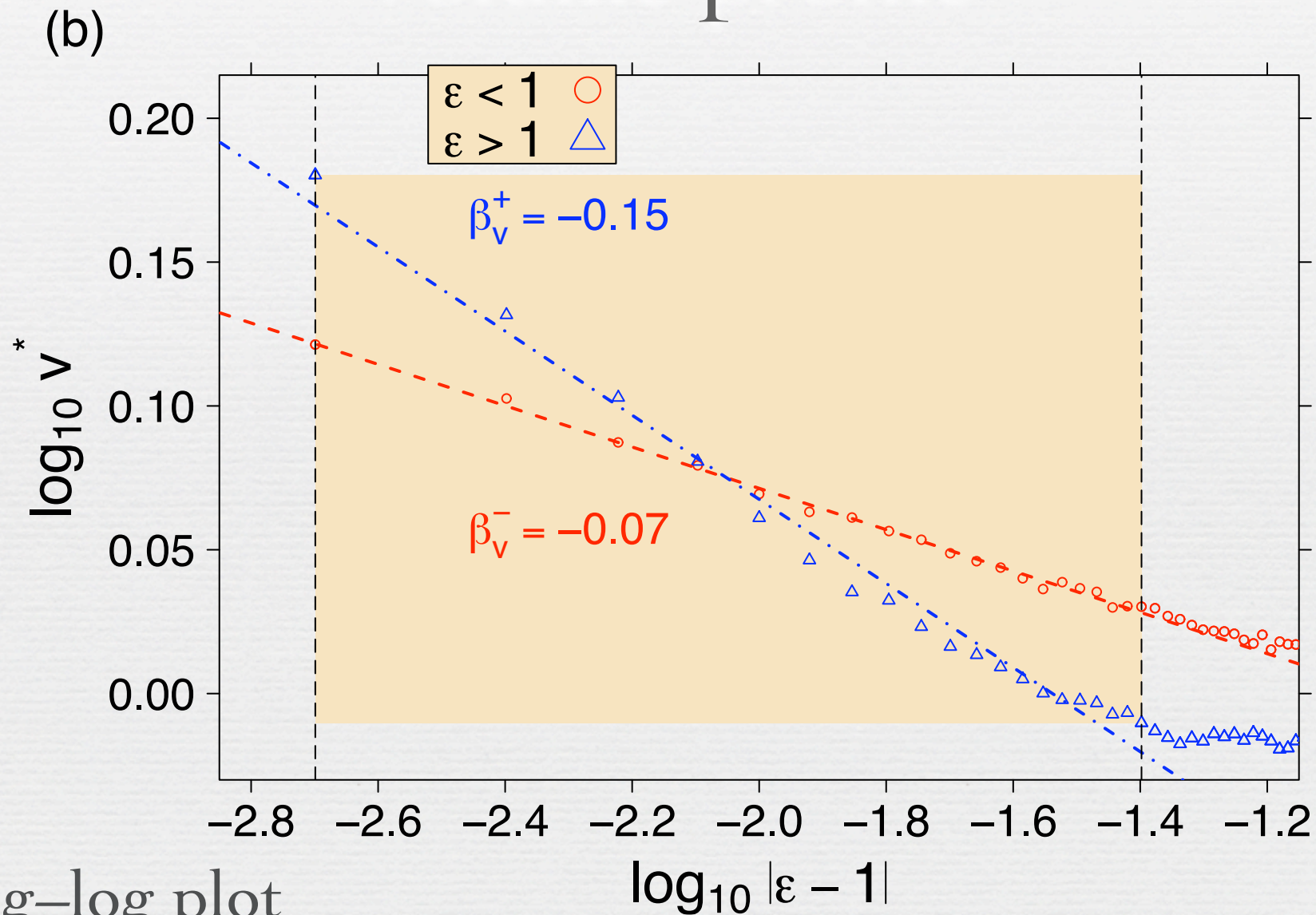


Log–log plot

Volume profile

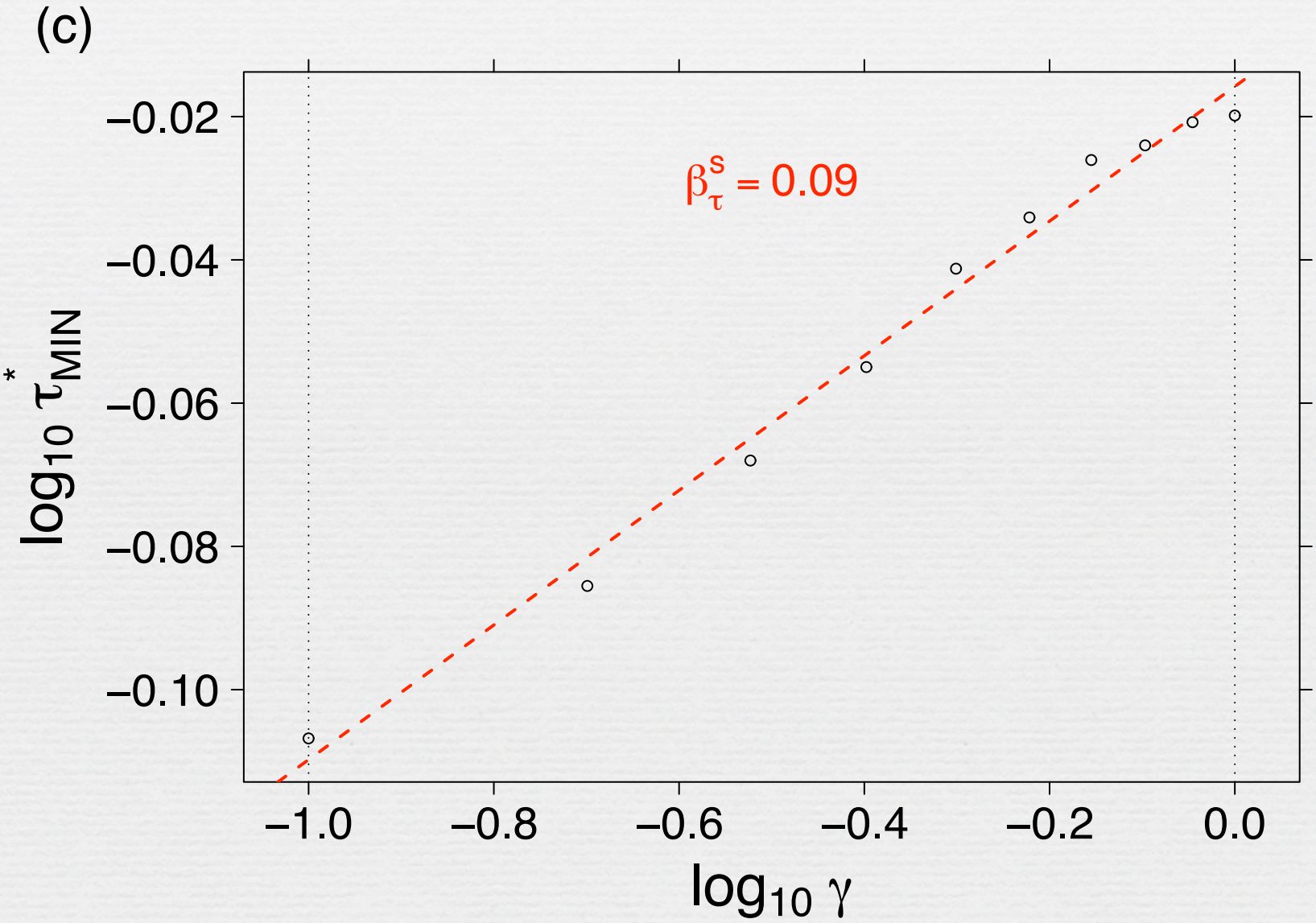


Volume profile

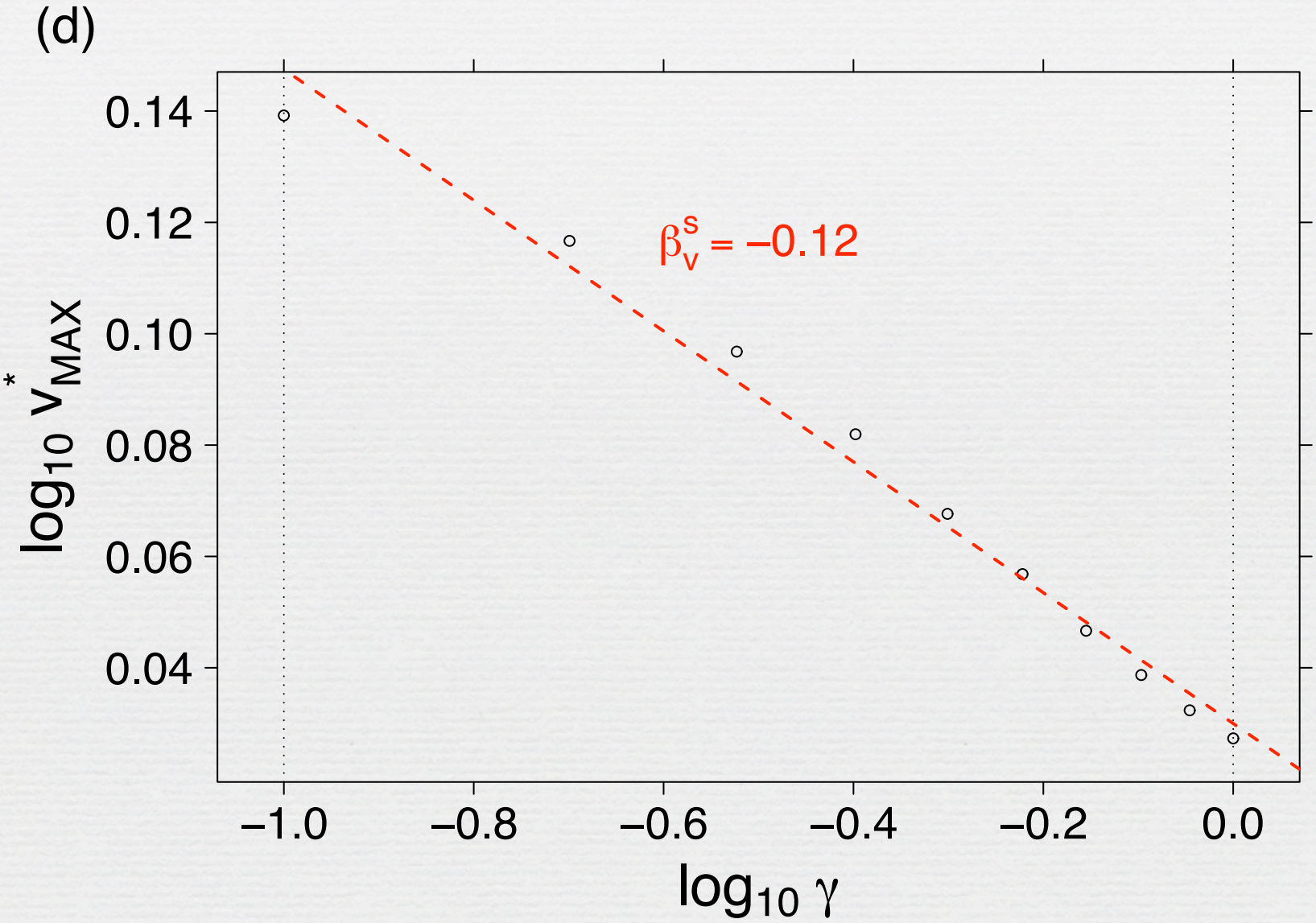


Log–log plot

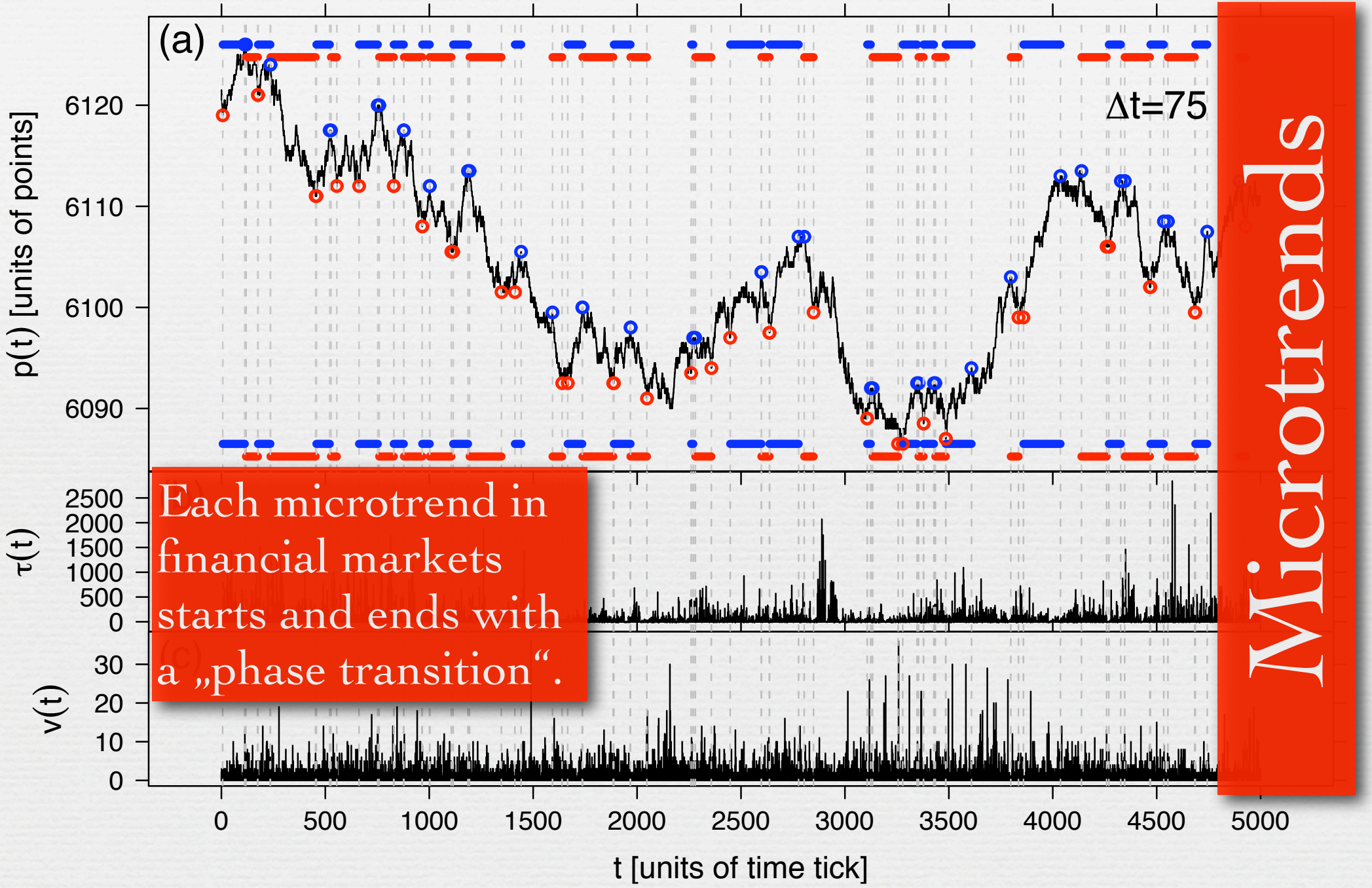
Shuffling of the ITWT time series



Shuffling of the volume time series



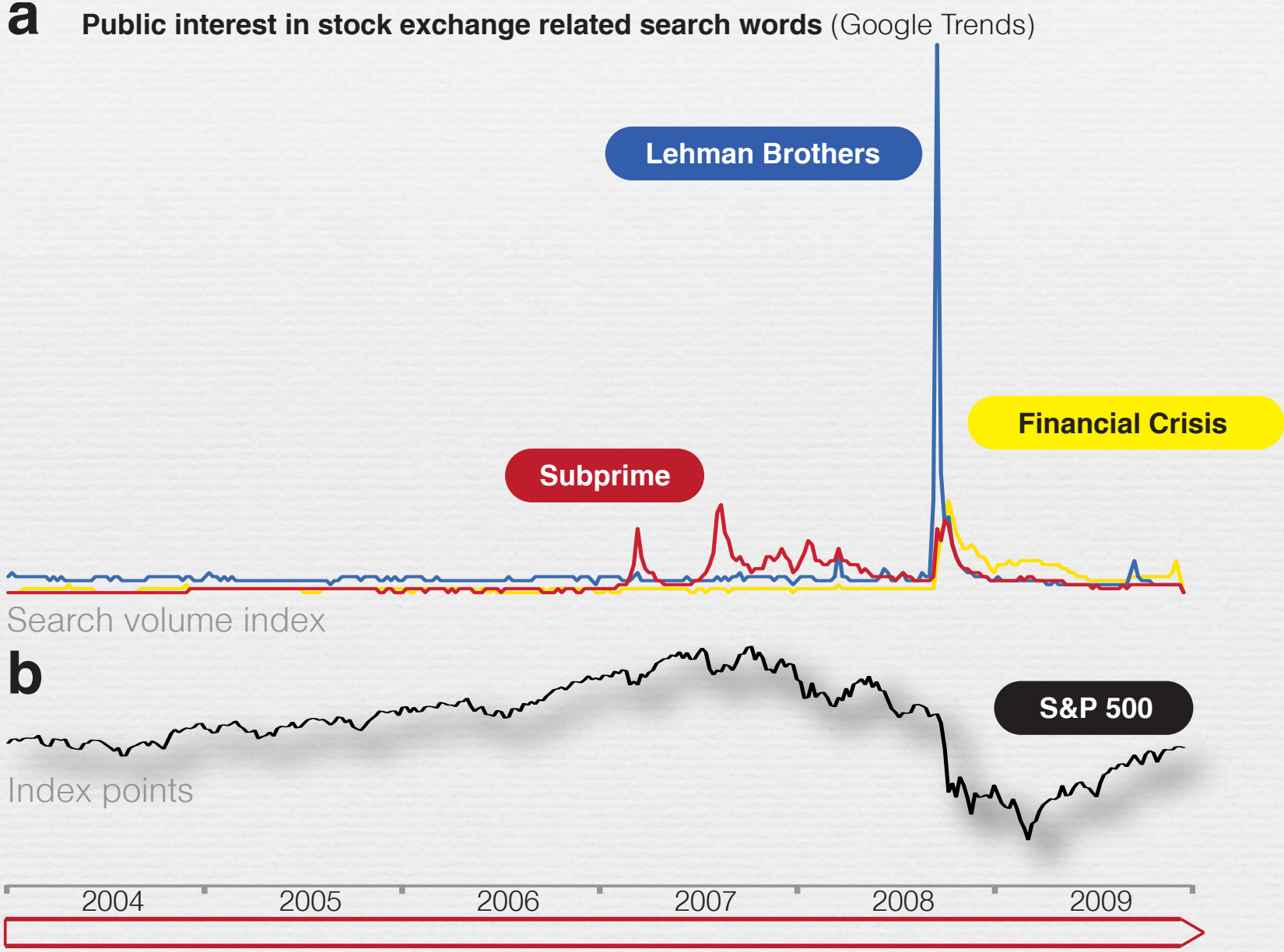
TP, J. J. Schneider, and H. E. Stanley, Preprint submitted (2009)



Each microtrend in financial markets starts and ends with a „phase transition“.

Microtrends

Google Trends



Tobias Preis

Breaking News ...

Notenbanken öffnen Geldhahn wie nie zuvor

Donnerstag, 18. September 2008, 17:18 Uhr (Quelle Reuters)

Frankfurt (Reuters) - Die wichtigsten Notenbanken weltweit haben in einer beispiellosen Aktion ihre Kräfte gebündelt und mit einer milliardenschweren Rettungsaktion den Geldmarkt vor einem Kollaps bewahrt.

Insgesamt 180 Milliarden Dollar stellte die US-Notenbank Fed am Donnerstag überraschend bereit. Dieses Geld können die anderen Zentralbanken an die Kreditinstitute weiterreichen, die sich untereinander nicht mehr trauen und deswegen kaum noch Dollar leihen. Dazu kamen Milliardenbeträge, die von der Europäischen Zentralbank (EZB), der Bank von England und anderen Notenbanken in eigener Währung ausgegeben wurden. Eine derartige Aktion hatte es selbst nach den Anschlägen in New York vom 11. September 2001 nicht gegeben.

Mit diesem Schritt wollen die Notenbanken Engpässe am Dollar-Geldmarkt lindern, die sich nach dem Zusammenbruch der US-Investmentbank Lehman Brothers, dem Notverkauf von Merrill Lynch und der 85-Milliarden-Dollar-Rettung des Versicherers AIG in den vergangenen Tagen verschärft hatten. Nach Einschätzung von Analysten beruhigte das Einschreiten der Notenbanken die Märkte. Allerdings dürften die Spannungen dennoch länger anhalten. "Das war kein großer Befreiungsschlag", sagte Rainer Sartoris von HSBC Trinkaus. "Bei den Zinsen am kurzen Ende sieht man aber, dass die Notenbanken Erfolg gehabt haben." Untereinander leihen sich die Banken inzwischen Dollar für etwa zwei Prozent, was dem Fed-Leitzins entspricht und Entspannung signalisiert. Am Morgen noch hatten die Kreditinstitute acht Prozent berechnet - ein Zeichen für massives Misstrauen.

"Die Zentralbanken arbeiten weiterhin eng zusammen und werden angemessene Maßnahmen ergreifen, um dem anhaltenden Druck entgegenzuwirken", teilte die EZB mit. Experten gehen deswegen davon aus, dass weitere Geldspritzen möglich sind. "Von Seiten der Notenbanken wird man pragmatisch vorgehen. Wenn sich die Lage wieder verschärft, wird man überlegen, wie man reagiert", sagte Sartoris.

...

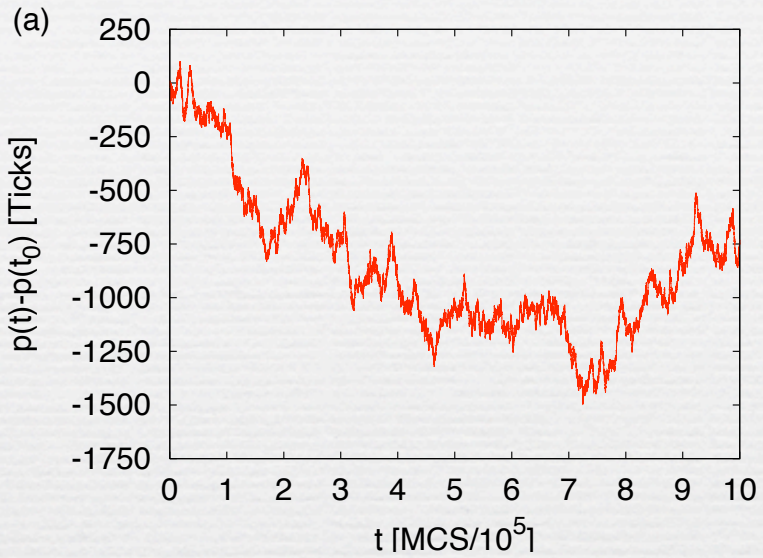
Risk aversion

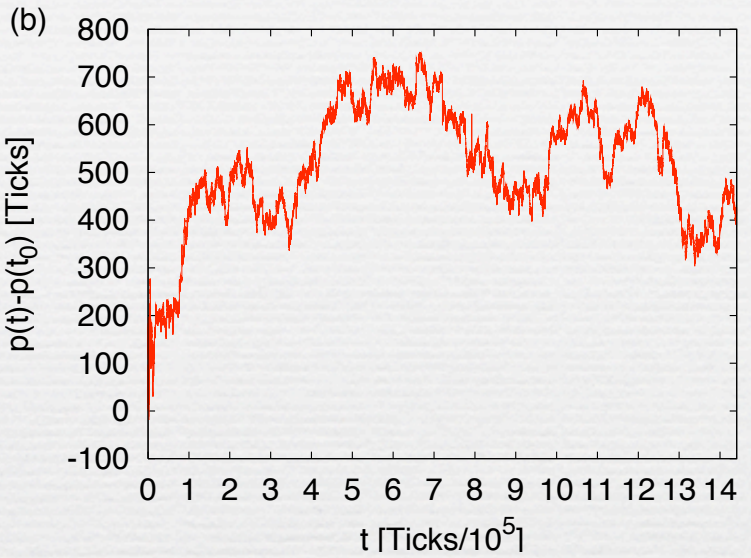
Gut funktionierende Geldmärkte sind lebenswichtig für das Funktionieren des Finanzsystems und der Wirtschaft als Ganzes. Die Banken leihen sich dabei untereinander kurzfristig Geld, um die täglichen Schwankungen in ihren Bilanzen auszugleichen. Seit dem Ausbruch der Finanzkrise vor gut einem Jahr ist der Geldmarkt jedoch stark gestört, seine Funktion wird zum Teil von den Notenbanken übernommen.

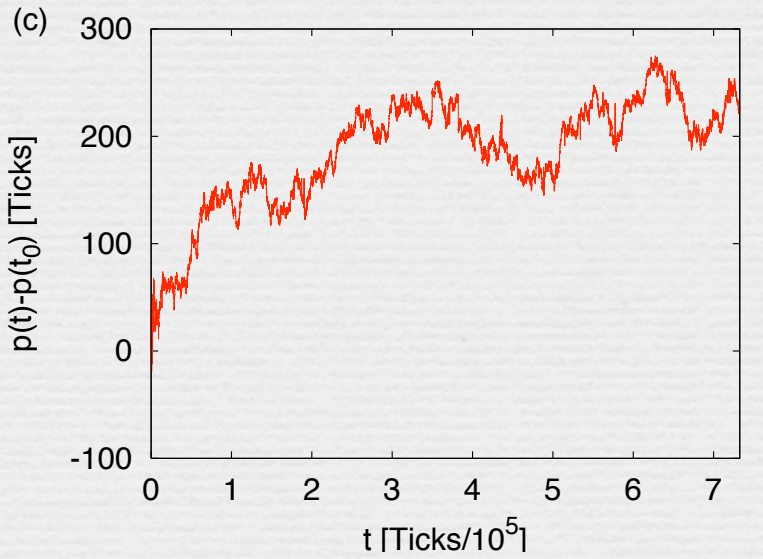
Model of financial markets

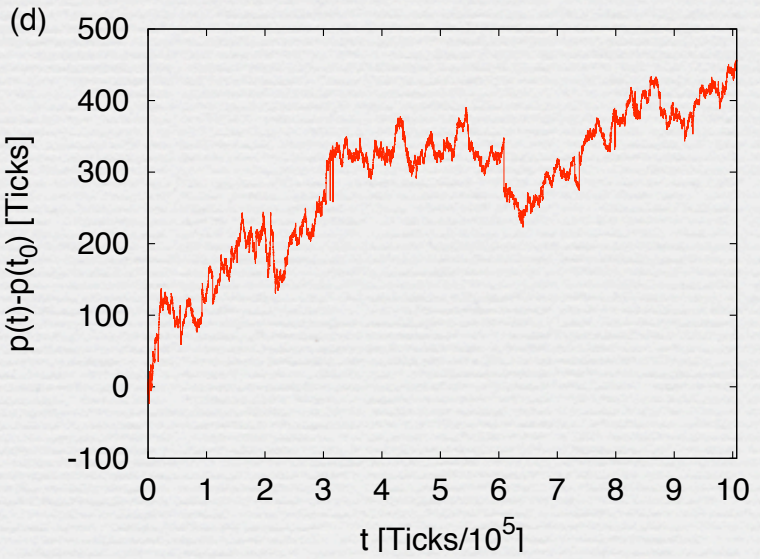
Tobias Preis

Motivation

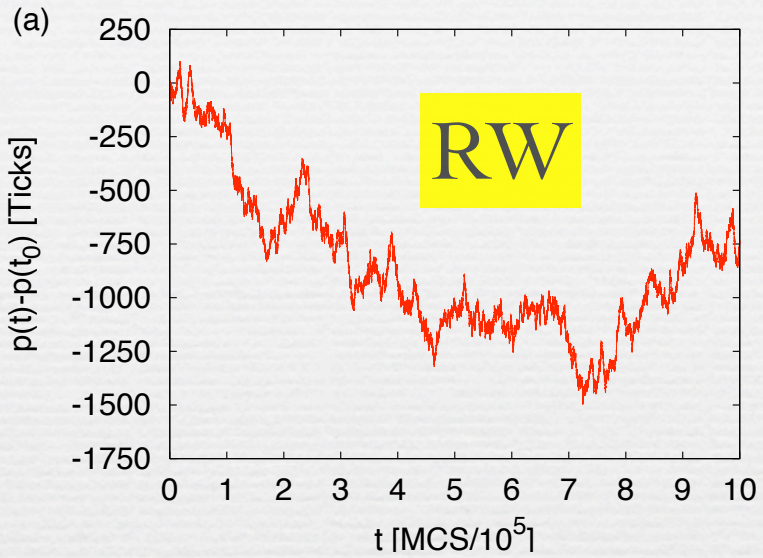

(a)
p(t)-p(t0) [Ticks]
t [MCS/10^5]

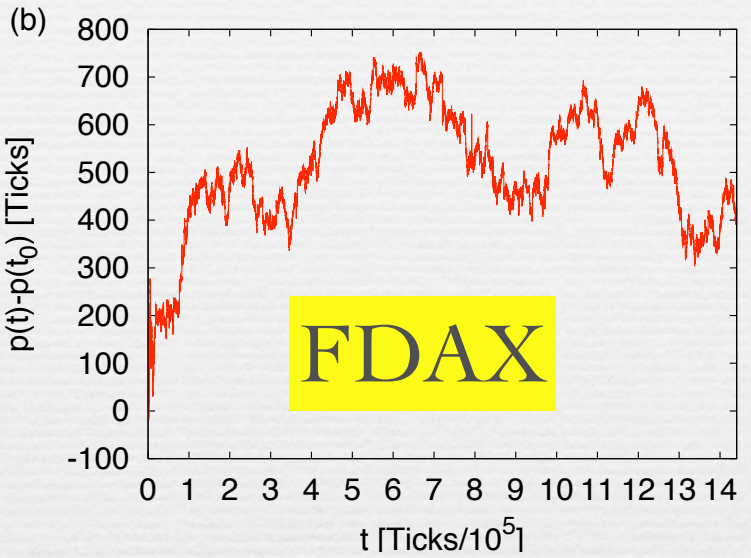

(b)
p(t)-p(t0) [Ticks]
t [Ticks/10^5]

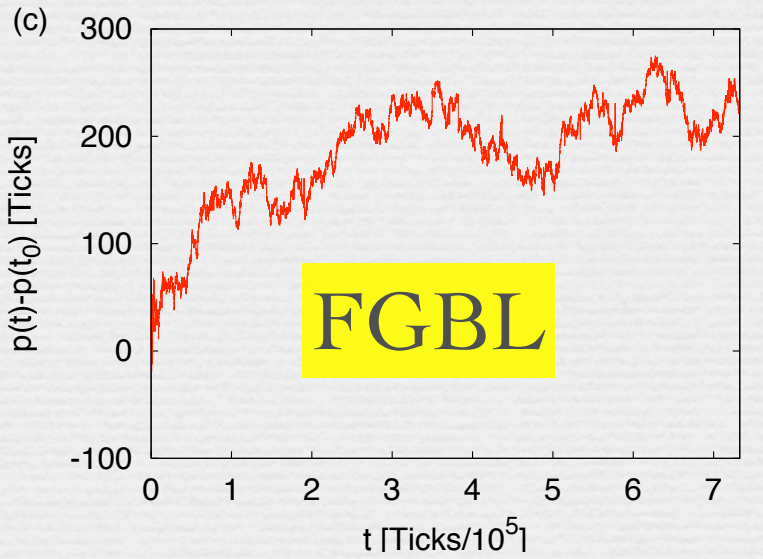

(c)
p(t)-p(t0) [Ticks]
t [Ticks/10^5]

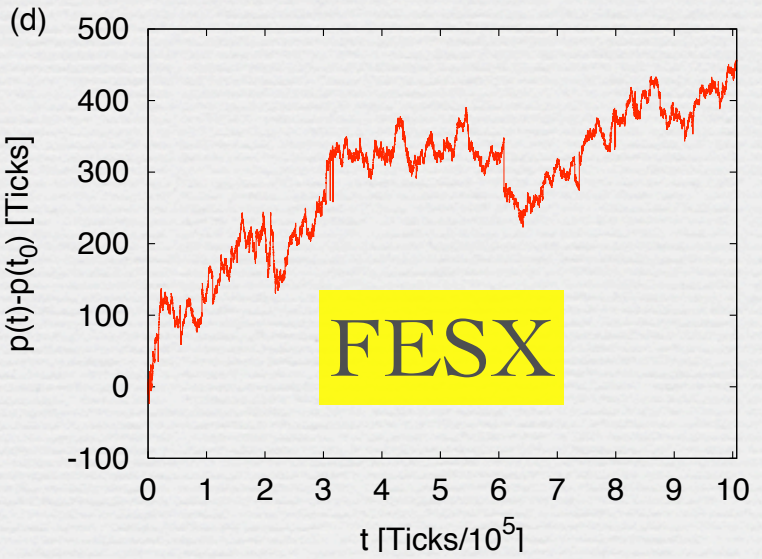

(d)
p(t)-p(t0) [Ticks]
t [Ticks/10^5]

Motivation


(a)
RW
p(t)-p(t0) [Ticks]
t [MCS/10^5]


(b)
FDAX
p(t)-p(t0) [Ticks]
t [Ticks/10^5]


(c)
FGBL
p(t)-p(t0) [Ticks]
t [Ticks/10^5]

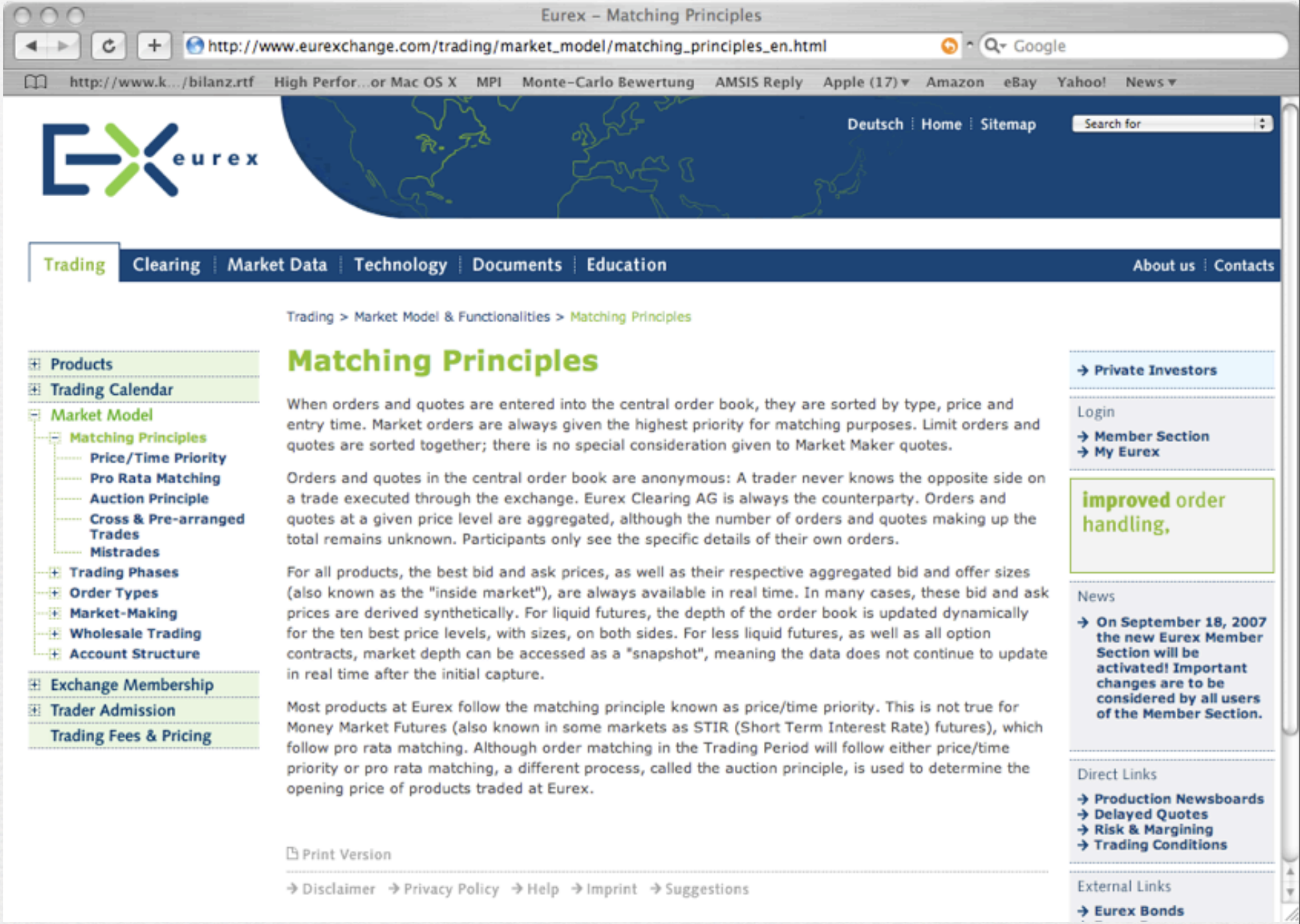

(d)
FESX
p(t)-p(t0) [Ticks]
t [Ticks/10^5]

Motivation

- Random walk (RW) vs. financial market data
- Empirical stylized facts of financial time series:
  - ➡ Fat-tailed price increment distributions
  - ➡ Significant short-term anti-correlated returns
  - ➡ Non-trivial scaling behavior of the returns
  - ➡ Volatility clustering

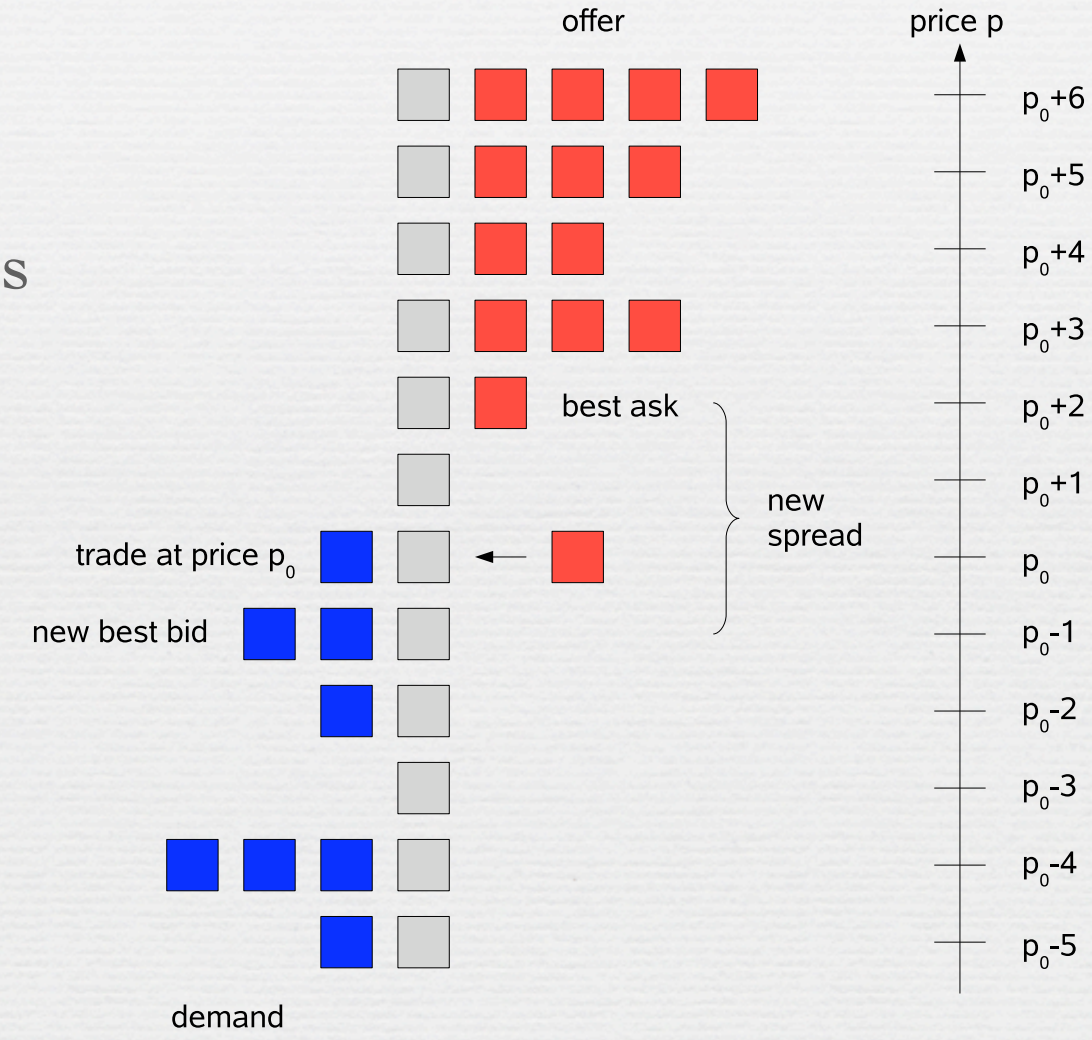
Modern financial markets

- Central electronic order book located in a computer center of the exchange; exchange members can be located anywhere in the world – connection via direct line or WWW.
- Different types of orders possible: limit orders, market order, stop orders, iceberg orders, ...
- Inter-trade waiting time: minimum ~ milliseconds!
- Transparency: order book is visible for members
- Market structure is regulated by exchange rules and these rules are published ...


Eurex – Matching Principles
http://www.eurexchange.com/trading/market_model/matching_principles_en.html
Google
http://www.k.../bilanz.rtf
High Perfor...or Mac OS X
MPI
Monte-Carlo Bewertung
AMSIS Reply
Apple (17)
Amazon
eBay
Yahoo!
News
eurex
Deutsch
Home
Sitemap
Search for
Trading
Clearing
Market Data
Technology
Documents
Education
About us
Contacts
Trading > Market Model & Functionalities > Matching Principles
Products
Trading Calendar
Market Model
Matching Principles
Price/Time Priority
Pro Rata Matching
Auction Principle
Cross & Pre-arranged Trades
Mistrades
Trading Phases
Order Types
Market-Making
Wholesale Trading
Account Structure
Exchange Membership
Trader Admission
Trading Fees & Pricing
Matching Principles
When orders and quotes are entered into the central order book, they are sorted by type, price and entry time. Market orders are always given the highest priority for matching purposes. Limit orders and quotes are sorted together; there is no special consideration given to Market Maker quotes.
Orders and quotes in the central order book are anonymous: A trader never knows the opposite side on a trade executed through the exchange. Eurex Clearing AG is always the counterparty. Orders and quotes at a given price level are aggregated, although the number of orders and quotes making up the total remains unknown. Participants only see the specific details of their own orders.
For all products, the best bid and ask prices, as well as their respective aggregated bid and offer sizes (also known as the "inside market"), are always available in real time. In many cases, these bid and ask prices are derived synthetically. For liquid futures, the depth of the order book is updated dynamically for the ten best price levels, with sizes, on both sides. For less liquid futures, as well as all option contracts, market depth can be accessed as a "snapshot", meaning the data does not continue to update in real time after the initial capture.
Most products at Eurex follow the matching principle known as price/time priority. This is not true for Money Market Futures (also known in some markets as STIR (Short Term Interest Rate) futures), which follow pro rata matching. Although order matching in the Trading Period will follow either price/time priority or pro rata matching, a different process, called the auction principle, is used to determine the opening price of products traded at Eurex.
Print Version
Disclaimer
Privacy Policy
Help
Imprint
Suggestions
Private Investors
Login
Member Section
My Eurex
improved order handling,
News
On September 18, 2007 the new Eurex Member Section will be activated! Important changes are to be considered by all users of the Member Section.
Direct Links
Production Newsboards
Delayed Quotes
Risk & Margining
Trading Conditions
External Links
Eurex Bonds

Order book structure

- Price-time priority
- Discrete price levels
- Limit orders
- Market orders



Model definition

- Liquidity provider N_A
- Liquidity taker N_A
- Limit order rate α
- Market order rate μ
- Order cancel rate δ
- Buy/Sell probability $q_{\text{provider}} = q_{\text{taker}} = \frac{1}{2}$
- Best bid p_b
- Best ask p_a
- Midpoint $p_m = \frac{p_a + p_b}{2}$
- Spread $s = p_a - p_b$

Model definition

- Exponentially distributed order entry depth
  - Limit price of a limit buy order

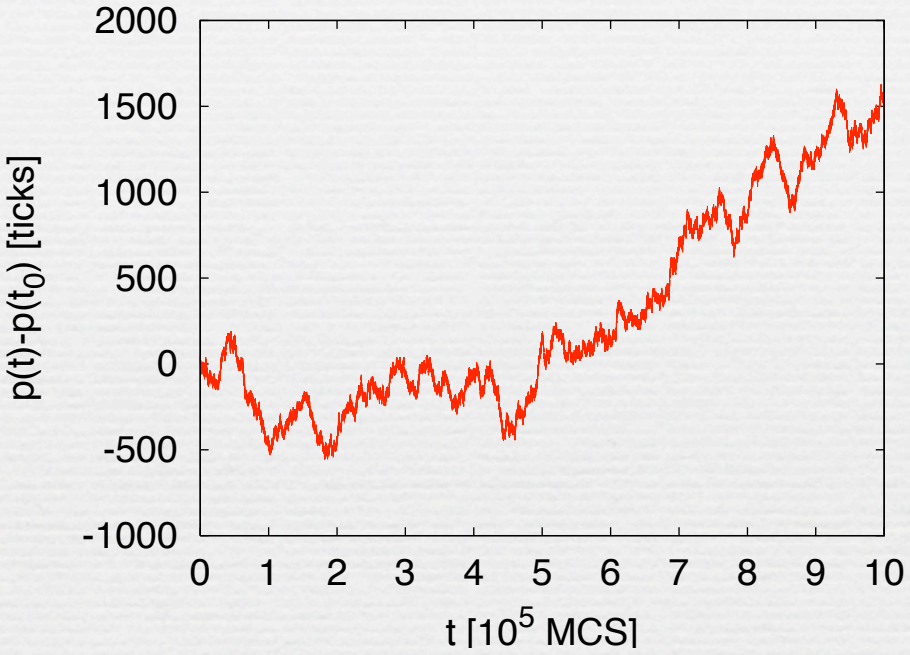
$$p_i^l = p_a - 1 - \eta$$

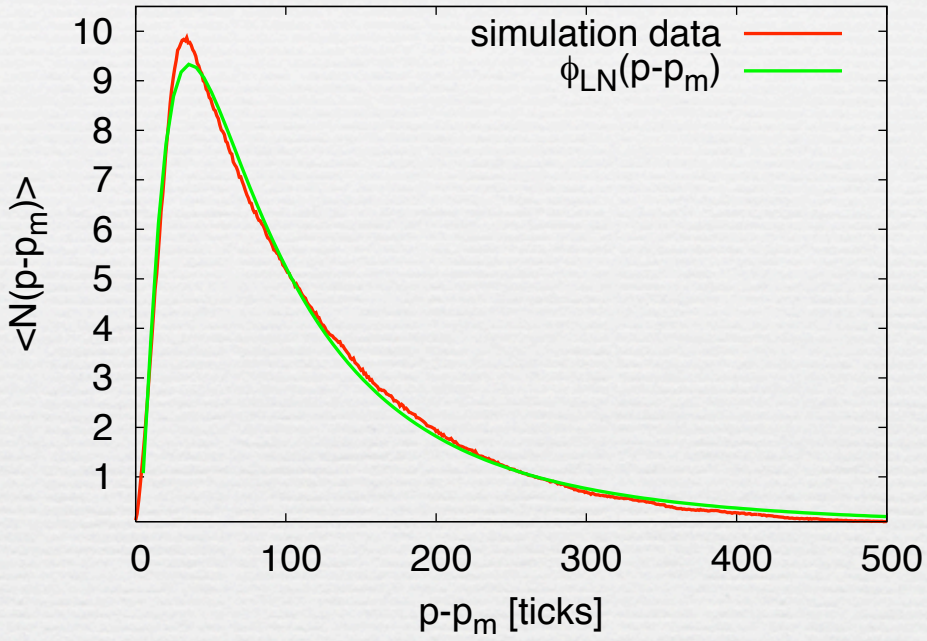
  - Limit price of a limit sell order

$$p_i^l = p_b + 1 + \eta$$

  - with stochastic variable $\eta = \lfloor -\lambda_0 \times \log(x) \rfloor$

Results of the model

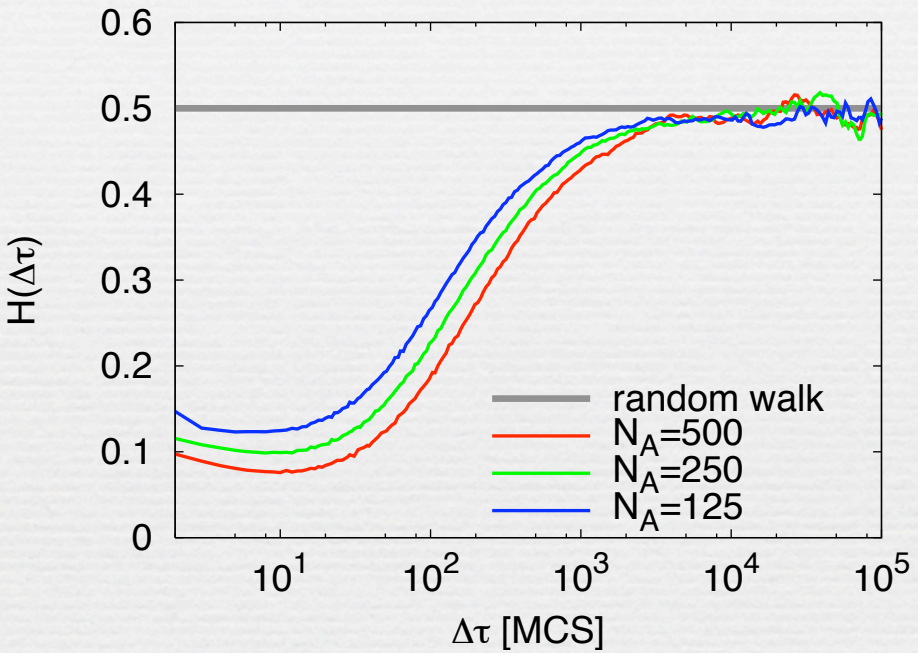


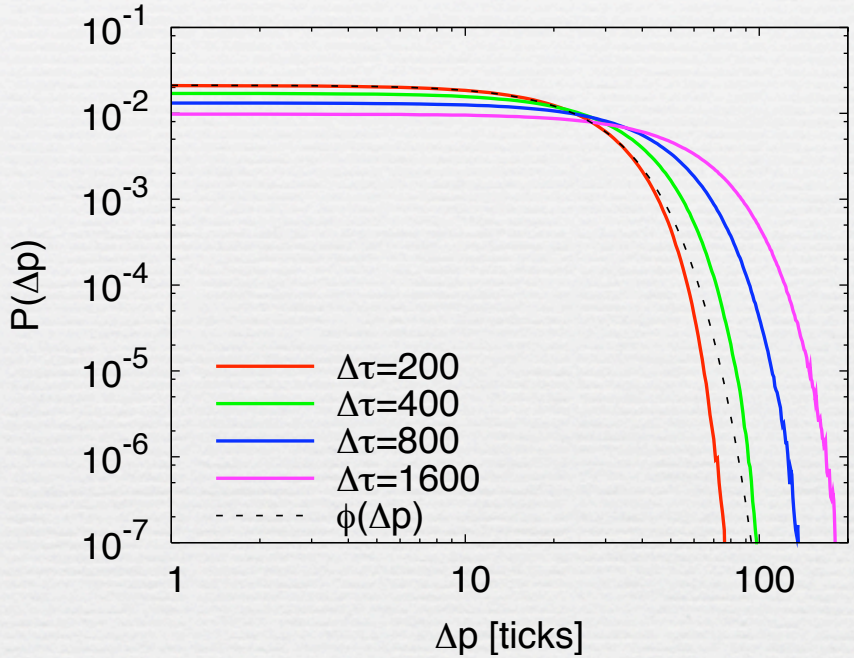


$$P_{\mathrm{LN}}(x) = A\,\frac{1}{Sx\sqrt{2\pi}}\,\exp\left(-\frac{(\ln x - M)^2}{2S^2}\right)$$

- Order book depth: log-normal distribution

Results of the model





- Hurst exponent $H(\Delta\tau)$: $\langle(\Delta p)^2\rangle^{1/2}(\Delta\tau) \propto \Delta\tau^H$
- Random walk: $H = 1/2$

Parameter space

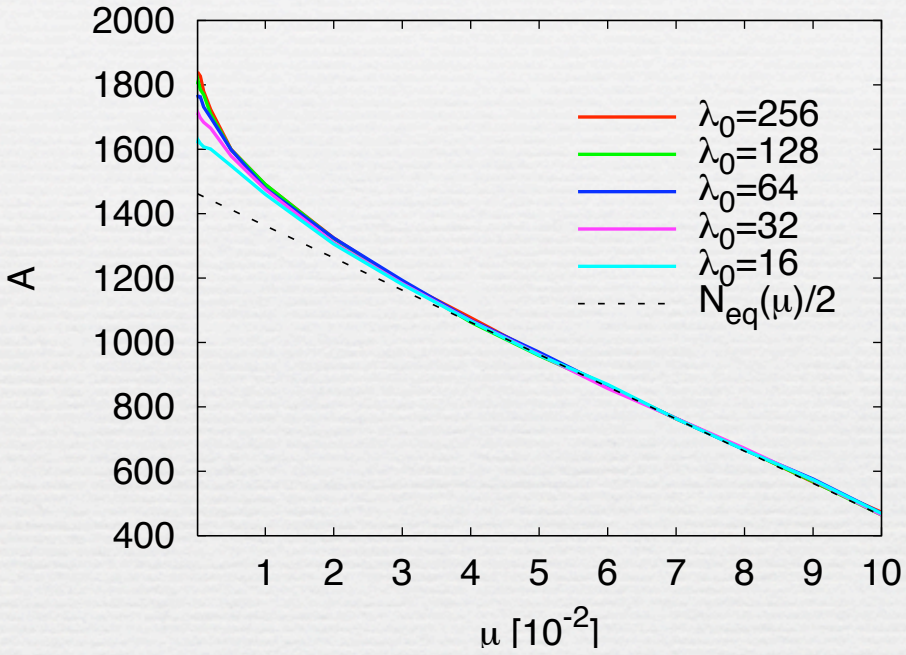
- Total number of orders stored in the order book at time t+1:

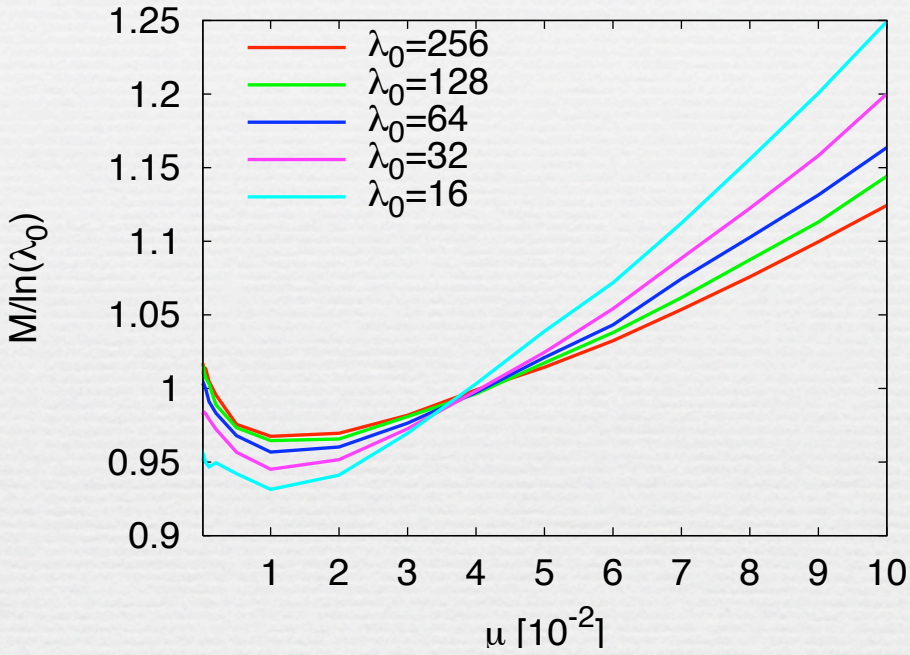
$$N(t+1) = N(t) + \alpha N_A - (N(t) + \alpha N_A)\,\delta - \mu N_A$$

- In equilibrium, one arrives at

$$\frac{N_{\mathrm{eq}}}{N_A} = \alpha\left(\frac{1}{\delta} - 1\right) - \frac{\mu}{\delta}$$

Parameter space





- Parameter of the log-normal distribution in dependence of the market order rate for different order entry depths

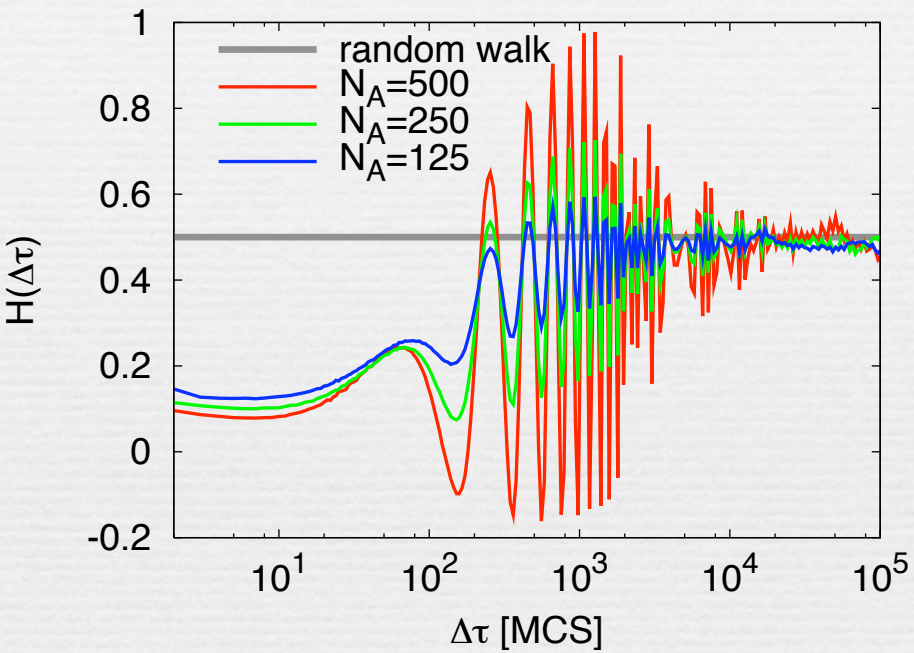
Asymmetric order flow

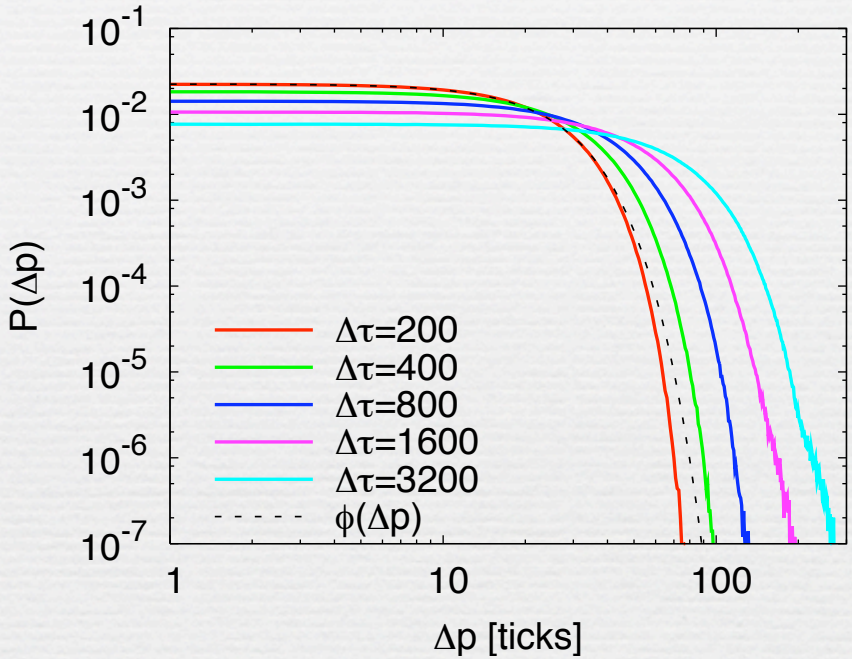
- Time dependent buy/sell probability (micro trends):
  - Deterministic perturbation: Sawtooth
  - Stochastic perturbation: Feedback random walk (FRW) – RW with mean reversion tendency

$$\langle q_{\text{taker}}(t+1)\rangle = (q_{\text{taker}}(t) - \Delta S) q_{\text{taker}}(t) + (q_{\text{taker}}(t) + \Delta S)(1 - q_{\text{taker}}(t))$$

$$\langle q_{\text{taker}}(t+1)\rangle - q_{\text{taker}}(t) = \Delta S (1 - 2 q_{\text{taker}}(t)) \begin{cases} > 0 & \text{if } q_{\text{taker}}(t) < 1/2 \\ < 0 & \text{if } q_{\text{taker}}(t) > 1/2 \end{cases}$$

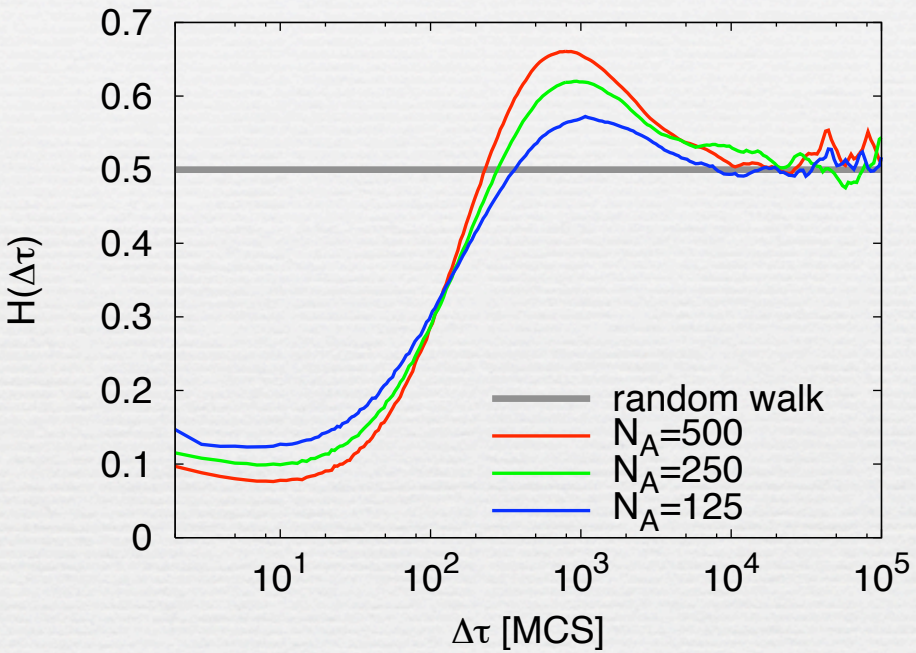
Sawtooth

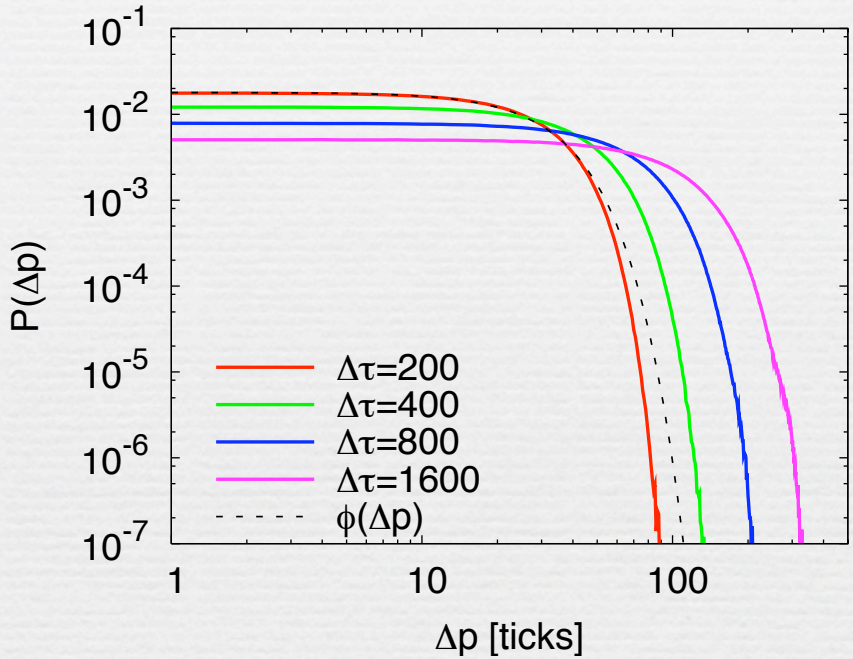




- Deterministic modulation with a period of 200 MCS is reflected in the mean square displacement, leading to quasi-periodic oscillations of the Hurst exponent.

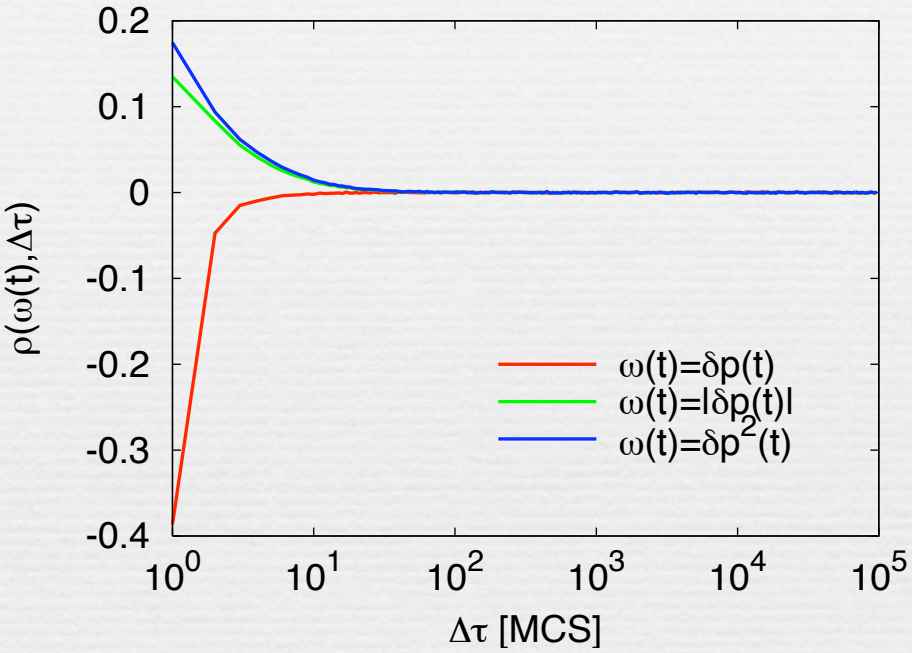
Feedback random walk





- Again an anti-persistent behavior on short time scales, a persistent behavior on medium time scales, and a diffusive regime on long time scales

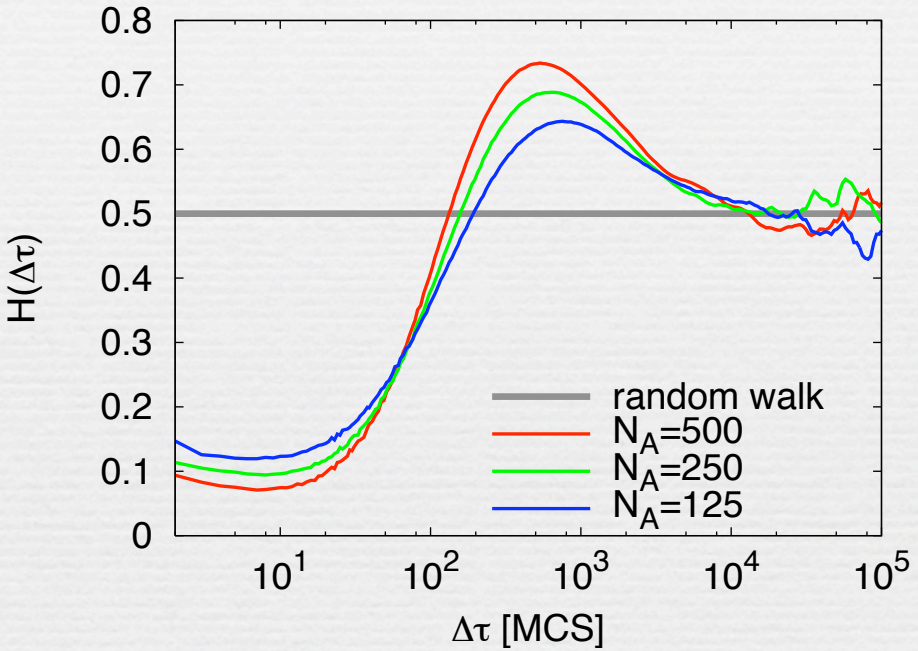
Autocorrelations

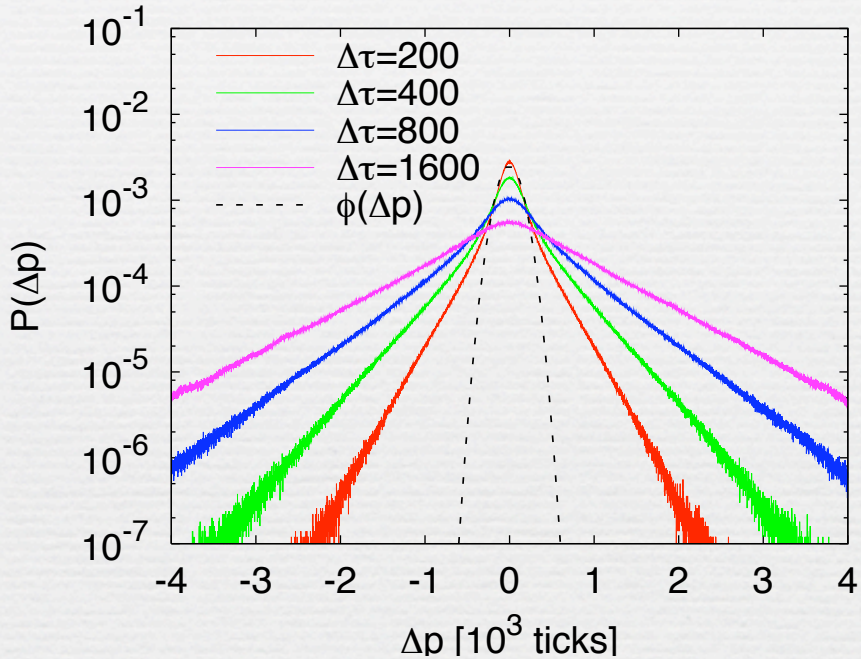


$$\varrho(\omega(t)\tau) = \frac{\langle \omega(t+\tau)\omega(t)\rangle - \langle \omega(t)\rangle^2}{\langle \omega(t)^2\rangle - \langle \omega(t)\rangle^2}$$

- Negative autocorrelation on short time scales
- Volatility clustering

Dynamic order depth





$$\lambda(t) = \lambda_0 \left(1 + \frac{\left| q_{\text{taker}}(t) - \frac{1}{2} \right|}{\sqrt{\left\langle \left(q_{\text{taker}}(t) - \frac{1}{2} \right)^2 \right\rangle}} \times C_\lambda \right)$$

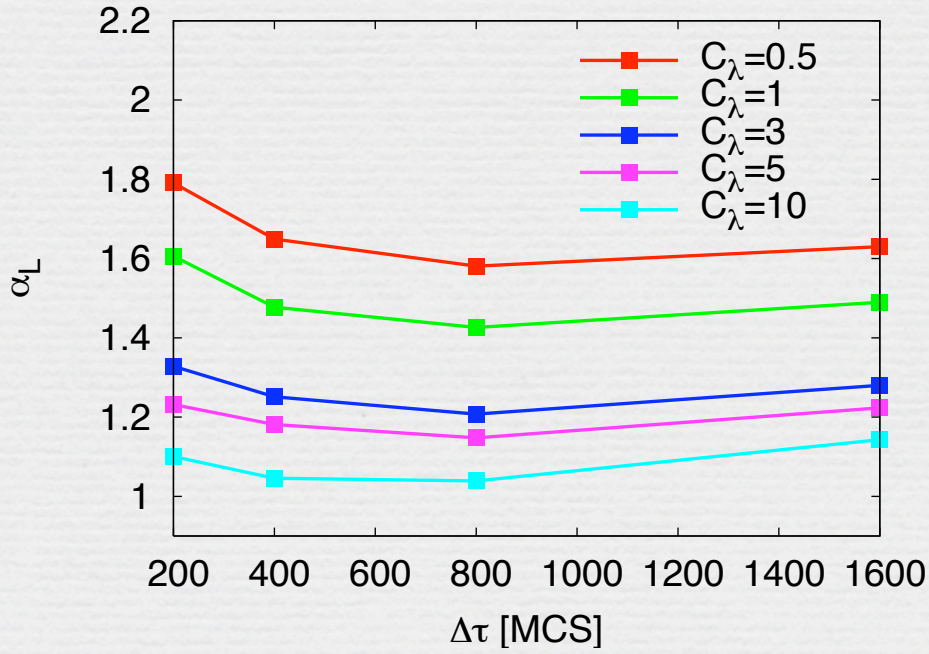
- Dynamic order entry depth leads to fat-tailed returns

Return distributions

- Return distributions can be fitted with truncated Lévy distribution

$$\Lambda_{\alpha_L c_1 l}(f_n) = \exp\left(c_0 - c_1 \frac{(f_n^2 + 1/l^2)^{\alpha_L/2}}{\cos(\pi\alpha_L/2)} \cos\left(\alpha_L \arctan(l|f_n|)\right)\right)$$

with $c_0 = \dfrac{l^{-\alpha_L}}{\cos(\pi\alpha_L/2)}$



OBM remarks

- Introduction of a simple model, based on an order book structure
- Reproduction of anti-persistent scaling behavior on short time scales, persistent on medium time scales, and diffusive behavior on long time scales using the feedback random walk
- If one additionally couples the order entry depth to the prevailing trend, one obtains fat-tailed price increments

universität mainz

Thank you!

Tobias Preis

http://www.tobiaspreis.de