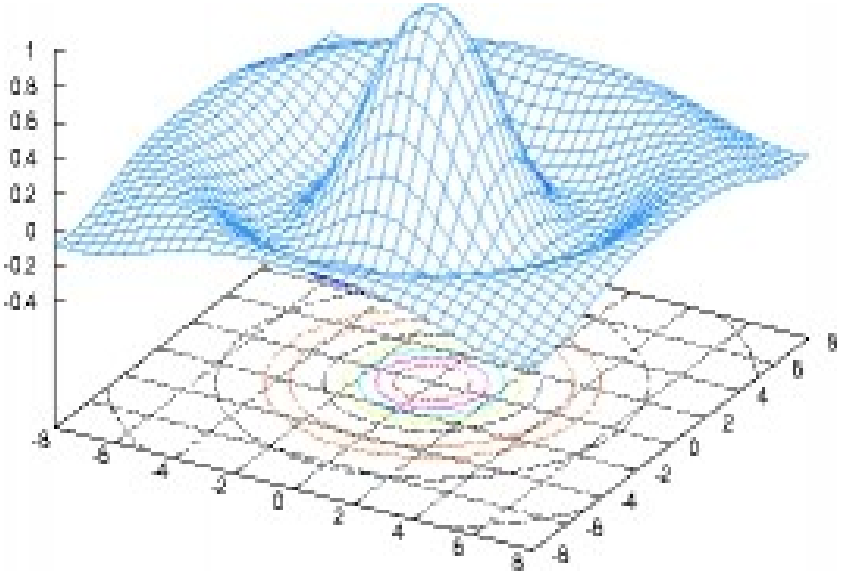
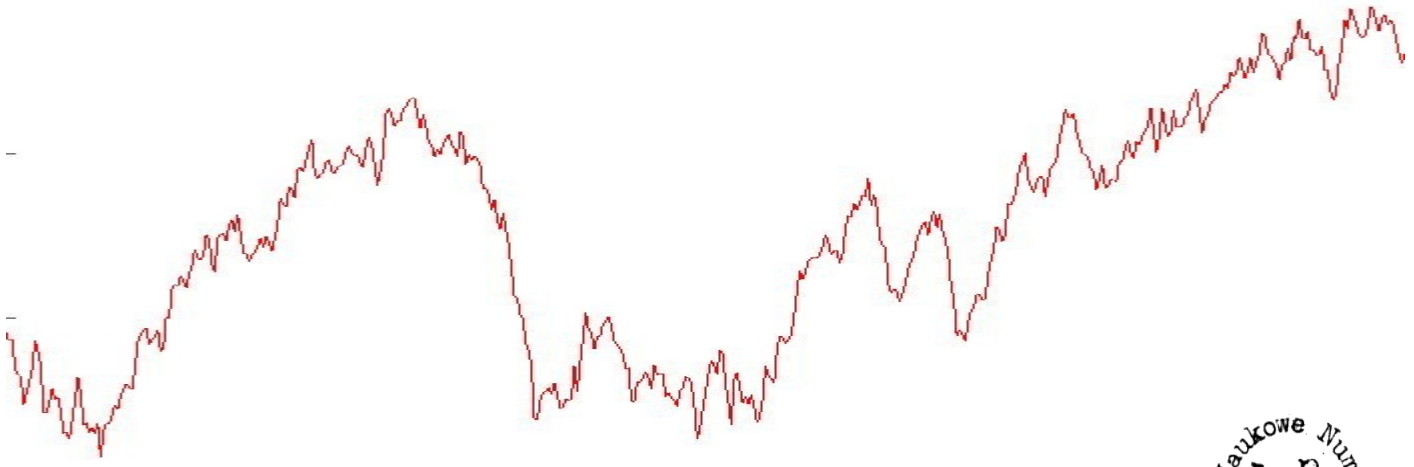
Numeryczne rozwiązywanie SRR w Octave



Piotr Dobaczewski
knn.mimuw.edu.pl

Do czego służy `sdesolve`?

- Jak na razie – do wyliczania trajektorii rozwiązania stochastycznych równań różniczkowych



Dla ustalenia uwagi

- SRR nazywamy

$$dX_t = a(X_t, t)dt + \sum_{i=1}^{m} b_i(X_t, t)dW_t^i$$

gdzie:

$X_t \in R^d \qquad a, b_1, b_2, \ldots, b_m : R^d \times R \to R^d$

$W_t^i, i = 1, 2, \ldots, m$ standardowy proces Wienera

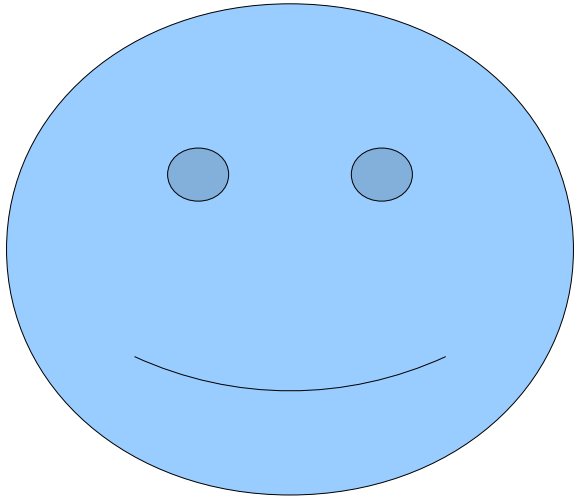
Dla jeszcze lepszego ustalenia uwagi

- Zapis macierzowy, B – macierz d x m

$$dX_t = a(X_t, t)dt + B(X_t, t)dW_t$$

Co zrobi `sdesolve?`

- Weźmie od nas
  - a
  - B
  - X_0
- Odda nam
  - X



- Przyjmijmy d=m=1

Koło Naukowe Numeryków

W świecie Octave

- a, B są funkcjami umieszczonymi w plikach

```
function y = nazwa (x, t)

y = 5*x + 0.5*t;

end
```

- Plik musi się nazywać nazwa.m i być w katalogu roboczym

W świecie Octave 2

- Rozpatrujemy nasze SRR na przedziale czasowym [0, 1]
- Załóżmy, że potrzeba nam X_t dla

  t = 0, 0.01, 0.02, ..., 1
- w Octave:

```
> t = [0 : 0.01 : 1];
```

Koło Naukowe Numeryków

W świecie Octave 3

- Wzywamy `sdesolve:`

```
X = sdesolve( {'rhsa', 'rhsb'},
                    1,
                    t,
                    20 );
```

- `{}` – lista z nazwami funkcji, **pamiętaj o ''**
- 1 – X_0
- 20 – ilość trajektorii do wylosowania

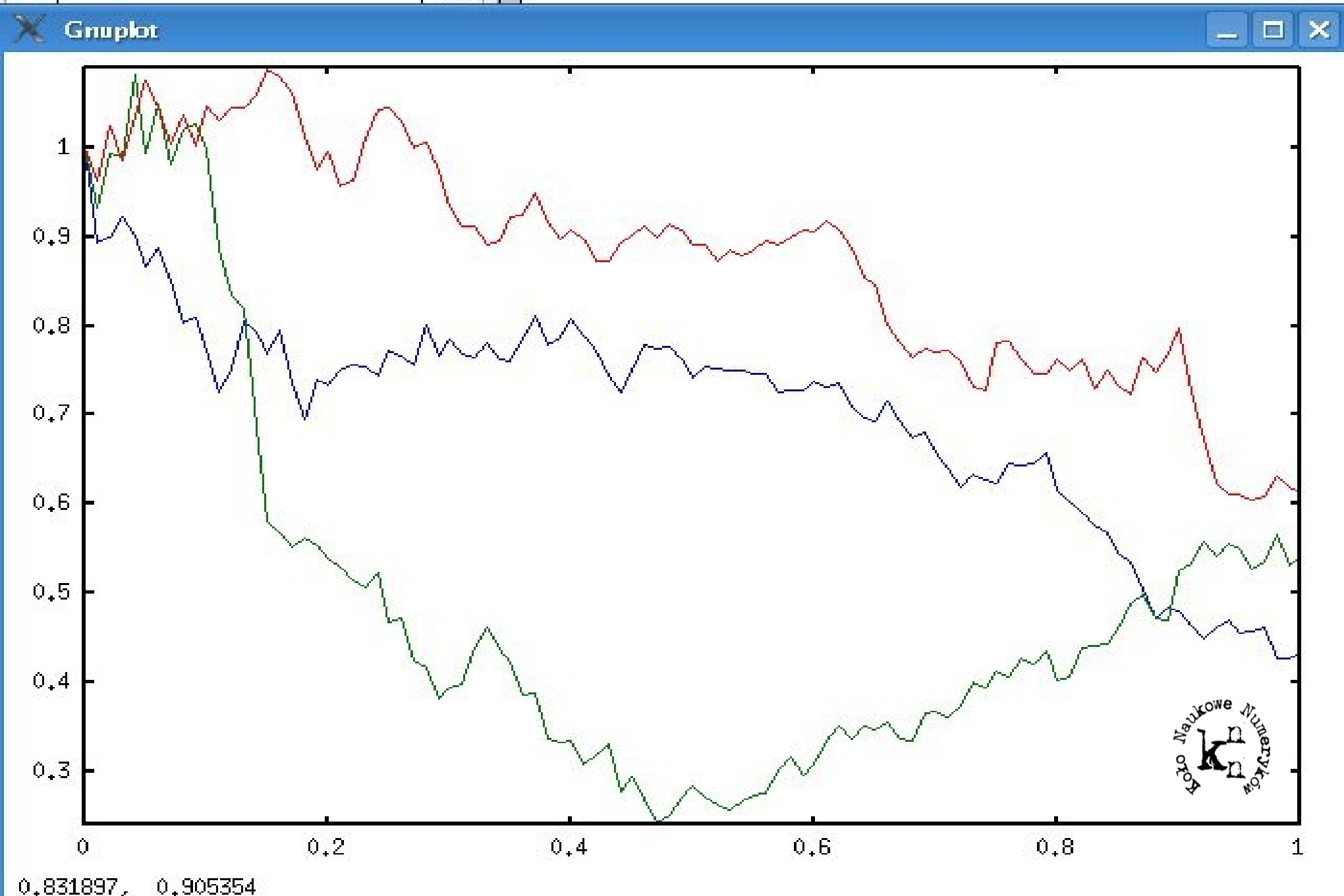
W świecie Octave 4

- Wynik: macierz X

```
> size(X)
ans : 101 20
```

- 101 wierszy – 101 momentów czasowych
- 20 kolumn – 20 trajektorii
- Błyskawiczny obrazek poglądowy:

```
> plot (t, X(:,1:3))
```


Gnuplot
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
0.2
0.4
0.6
0.8
1
Koło Naukowe Numeryków
0.831897, 0.905354

W wielu **wymiarach**

- Zwiększamy d:
- X_0 jest wektorem d-wymiarowym

```
> X0 = [1, 3.4, 55];
```

- Funkcja rhsa – wektorem stojącym!
-
```
function y = rhsa (x, t)
y = [x(2); -x(1); - x(2) * x(1)];
end
```

W wielu **wymiarach** 2

- Funkcja rhsb jest macierzą d x m:

```
function y = rhsb (x, t)
y = [0.5*x(1), 0                          ;
      0          , 0.1* x(2) * x(1);
      0          , 0                          ];
end
```

- UWAGA! W 3. równaniu 'nie ma' dW

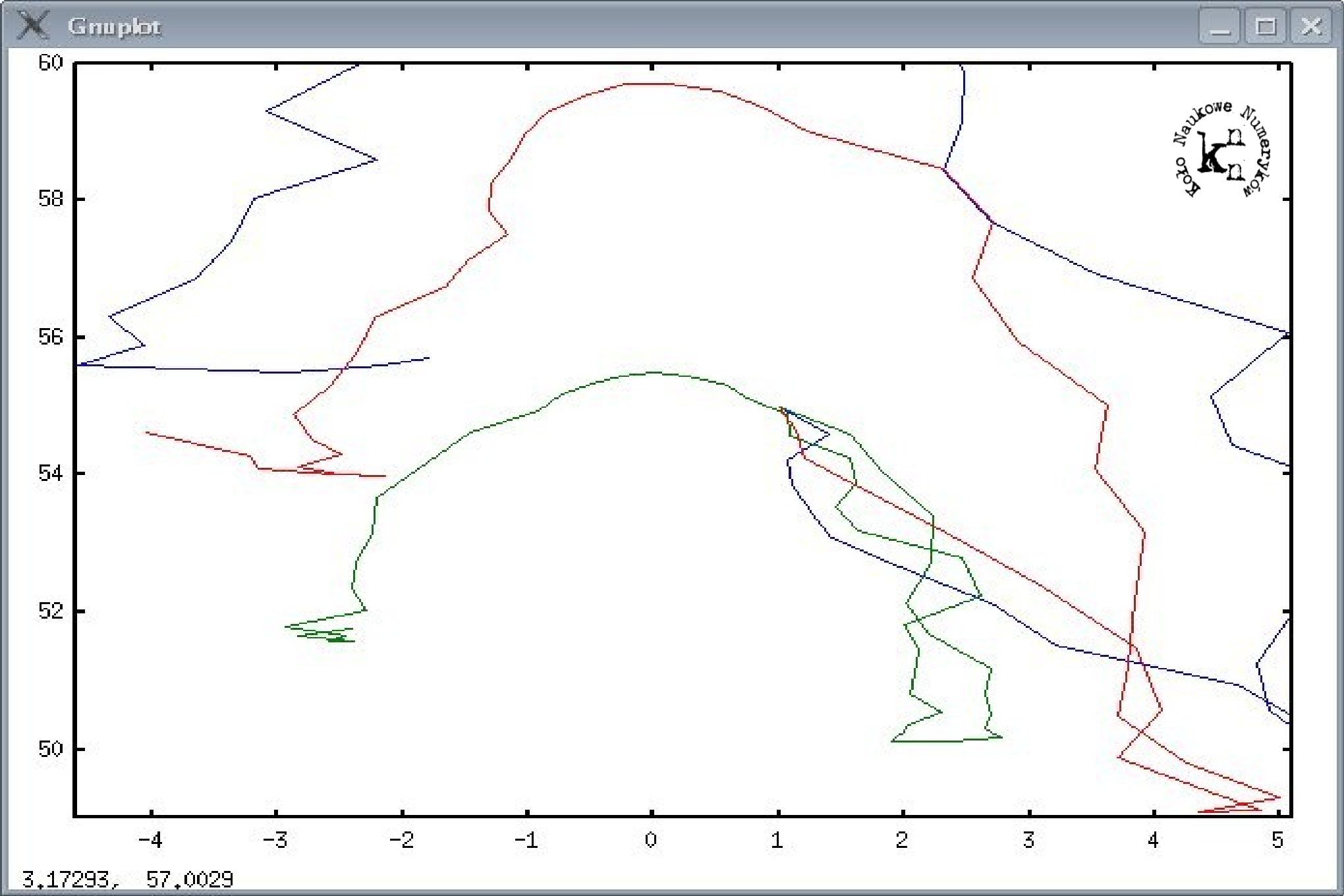
W wielu **wymiarach** 3

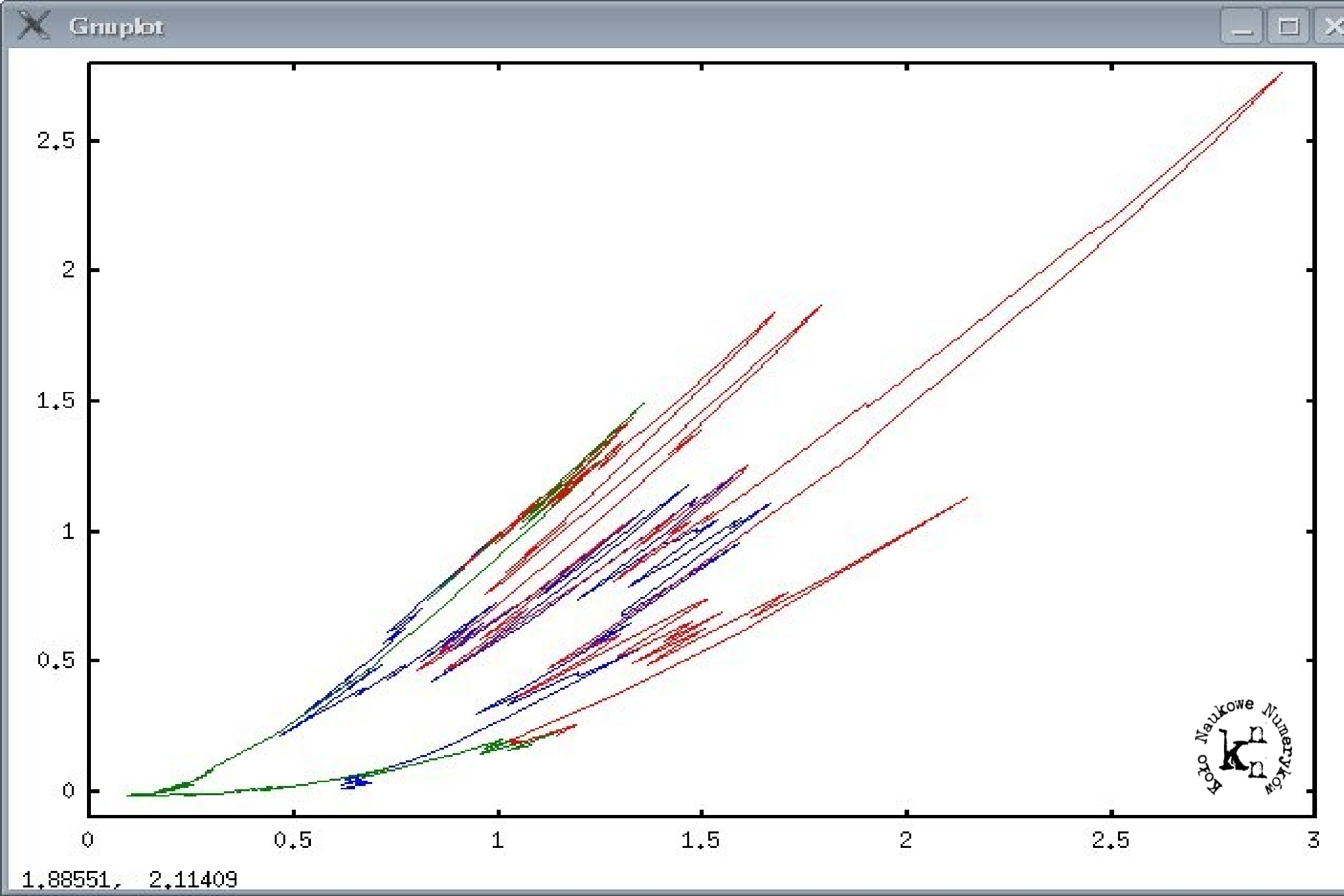
- Wynik: macierz X

```
> size(X)
ans : 101 3 20
```

- 101 wierszy – 101 momentów czasowych
- 3 kolumny – 3 współrzędne trajektorii
- 20 razy macierz 101 x 3 – 20 trajektorii
- Błyskawiczny obrazek poglądowy:

```
> plot(X(:, 1, 1:3), X(:, 3, 1:3))
```


Gnuplot
60
58
56
54
52
50
-4
-3
-2
-1
0
1
2
3
4
5
Koło Naukowe Numerycy
3,17293, 57,0029


Gnuplot
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3
Koło Naukowe Numeryków
1.88551, 2.11409

Jak to **działa**?

- Domyślnie używa
  Silnego schematu rzędu 1

  w skrócie --- EO1S

- Rzędu 1 tzn. błąd proporcjonalny do kroku

$$E\left(max_{0\le n\le n_T}|Y_n - X_{t_n}|^2\right) \le Kd^2$$

Co z wynikiem?

- średnia ścieżka – funckja `mean`
- estymator jądrowy gęstości rozkładu X_T

  funkcja (autorska, prościutka)

```
[x,y] = ke(X_t, h);
plot (x, y)
```

  albo `kernel_density` w octave-forge

- zapisać do pliku i wgrać do R

```
save -ascii nazwa_zmiennej plik
```

Gdzie znaleźć `sdesolve`?

- Weboctave
- Ściągnąć do własnego octave'a
- Kiedyś (może) trafi do octave-forge

http://knn.mimuw.edu.pl/weboctave

Gdzie znaleźć `sdesolve`?

- Weboctave
- Ściągnąć do własnego octave'a
- Kiedyś (może) trafi do octave-forge

knn.mimuw.edu.pl/sdesolve.tar

- trzeba skompilować .oct-plik
mkoctfile doubleii.cc
- i umieścić pliki w odpowiednim katalogu

Gdzie znaleźć `sdesolve`?

- Weboctave
- Ściągnąć do własnego octave'a
- Kiedyś (może) trafi do octave-forge

... nieprędko

W świecie R

- W laboratorium:
  trzeba załadować paczkę „sde” (niestety stara wersja)

```
d <- expression (-1*x)
s <- expression (x)
s.x = expression (1)
```

- Tak określamy prawą stronę SRR

W świecie R 2

- `set.seed(1)` – dla powtarzalności
-
```
X_mi <- sde.sim (t0=0, T = 1,
                 X0 = 1, N = 1024,
                 drift = d,
                 sigma = s,
                 sigma.x = s.x,
                scheme = "milstein")
```
- `plot(X_mi)`

W świecie R 3

- Przy nowej paczce sde!
-
```
X_mi <- sde.sim (t0=0, T = 1,
                 X0 = 1, N = 1024,
                 M = 20,
                 drift = d,
                 sigma = s,
                 sigma.x = s.x,
                method = "milstein")
```
-
```
Plot( X_mi [, 1:2] )
```

W świecie R 4

W porównaniu z Octave'em:

- Wolniejsze (6 razy!), jednowymiarowe sde
- Poza tym często szybszy, ale... niekoniecznie
- Bogatsze narzędzia statystyczne
- Trudniej zrównoleglić

Równolegle

Po co?

- W wielu wymiarach dużo wolniej!
- Może być i 10 razy szybciej (moją metodą)
- Łatwo zrównoleglić

Równolegle w labie

Potrzebne jest:

- Trochę miejsca na koncie wydziałowym
- Wygenerowanie klucza dla SSH
- Jak najwięcej włączonych komputerów (Linux)
- Odrobina znajomości Unixa i Basha
- Napisany .m-plik z zadaniem obliczenia konkretnej próbki trajektorii
- Zrozumienie jak działa sdesolve.m

Równolegle w labie

Potrzebne jest:

- Trochę miejsca na koncie wydziałowym
- Wygenerowanie klucza dla SSH
- Jak najwięcej włączonych komputerów (Linux)
- Odrobina znajomości Unixa i Basha
- Napisany .m-plik z zadaniem obliczenia konkretnej próbki trajektorii
- Zrozumienie jak działa sdesolve.m

Równolegle w labie

Potrzebne jest:

- Trochę miejsca na koncie wydziałowym
- Wygenerowanie klucza dla SSH
- Jak najwięcej włączonych komputerów (Linux)
- Odrobina znajomości Unixa i Basha
- Napisany .m-plik z zadaniem obliczenia konkretnej próbki trajektorii
- Zrozumienie jak działa sdesolve.m

Równolegle w labie

Potrzebne jest:

- Trochę miejsca na koncie wydziałowym
- Wygenerowanie klucza dla SSH
- Jak najwięcej włączonych komputerów (Linux)
- Odrobina znajomości Unixa i Basha
- Napisany .m-plik z zadaniem obliczenia konkretnej próbki trajektorii
- Zrozumienie jak działa sdesolve.m

Równolegle w labie

Potrzebne jest:

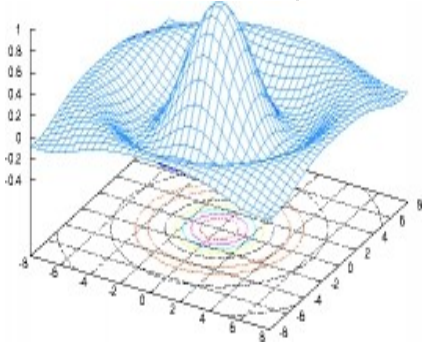
- Trochę miejsca na koncie wydziałowym
- Wygenerowanie klucza dla SSH
- Jak najwięcej włączonych komputerów (Linux)
- Odrobina znajomości Unixa i Basha
- Napisany .m-plik z zadaniem obliczenia konkretnej próbki trajektorii
- Zrozumienie jak działa sdesolve.m

Równolegle w labie

Potrzebne jest:

- Trochę miejsca na koncie wydziałowym
- Wygenerowanie klucza dla SSH
- Jak najwięcej włączonych komputerów (Linux)
- Odrobina znajomości Unixa i Basha
- Napisany .m-plik z zadaniem obliczenia konkretnej próbki trajektorii
- Zrozumienie jak działa sdesolve.m

Numeryczne rozwiązywanie SRR w Octave



Dziękuję za uwagę.

Piotr Dobaczewski
knn.mimuw.edu.pl