Analisi del White Noise

Marcello Colozzo

Fissiamo le opzioni per i grafici:

In[1]:=
```
SetOptions[
  {
   Plot,
   ListPlot,
   ListLinePlot
  },
  TicksStyle -> Directive[
    Hue[5 / 6],
    8
   ],
  FrameStyle -> Directive[
    Hue[5 / 6],
    8
   ]
 ];
```

Premessa: definire la densità di probabilità

Il problema che ci poniamo ora è come costruire un'assegnata densità di probabilità relativa a una variabile aleatoria $y(t)$. Ricordiamo che la lucuzione densità di probabilità si riferisce a una funzione continua $P(y)$ tale che $P(y_0)\,\mathrm{d}y$ è la probabilità infinitesima che una misura della variabile y fornisca un valore appartenente all' intervallo $[y_0,\, y_0 + \mathrm{d}y]$. Riferiamoci in particolare a una variabile gaussiana i.e. una variabile la cui densità di probabilità è una distribuzione gaussiana di valor medio μ e varianza σ^2, che viene invocata dal comando:

In[2]:=
```
NormalDistribution[μ, σ]
```

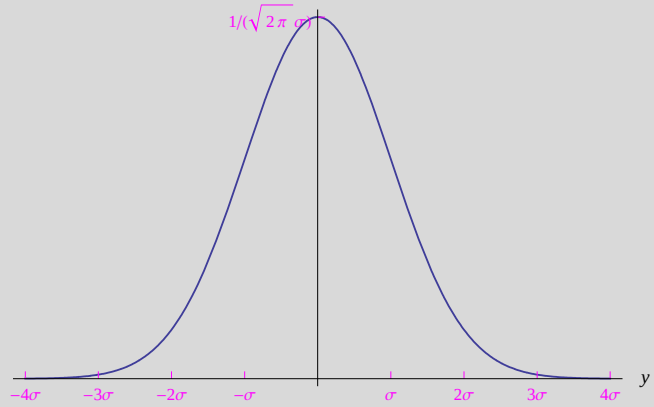
Out[2]=
```
NormalDistribution[μ, σ]
```

Il comando **`PDF[]`** restituisce la densità di probabilità per un'assegnata distribuzione. Nel caso in esame:

In[3]:=
```
gaussiana[μ_, σ_] := PDF[
  NormalDistribution[μ, σ],
  (*variabile indipendente*)
  y
 ]
```

In[4]:=
```
curvagaussiana = Plot[
  gaussiana[0, 1],
  {y, -4, 4},
  AxesLabel →
   {"y"},
  Ticks →
   {
    {
     {-4, "-4σ"}, {-3, "-3σ"}, {-2, "-2σ"}, {-1, "-σ"},
     {4, "4σ"}, {3, "3σ"}, {2, "2σ"}, {1, "σ"}
    },
    {
     {1/Sqrt[2 π], "1/(√(2 π) σ)"}
    }
   },
  PlotStyle → Thickness[0.003]
 ]
```

Out[4]=



La probabilità di misurare un valore $x \in [-n\sigma, n\sigma]$ è

In[5]:=
```
p[σ_, n_] := Integrate[
   gaussiana[0, σ],
   {y, -n σ, n σ}
  ] // N
```

Esplicitiamo alcuni valori:

In[6]:=
```
p[1, 1]
```

Out[6]=
```
0.682689
```

In[7]:=
```
p[1, 2]
```

Out[7]=
```
0.9545
```

In[8]:=
```
p[1, 3]
```

Out[8]=
```
0.9973
```

Generare numeri reali casuali

Il modo più veloce per generare un white noise consiste nel creare una lista di numeri reali casuali $y_1, y_2, \ldots, y_n$. A tale scopo utilizziamo il comando **`RandomReal[]`**

In[9]:=
```
? RandomReal
```

RandomReal[] gives a pseudorandom real number in the range 0 to 1.
RandomReal[{x_{min}, x_{max}}] gives a pseudorandom real number in the range x_{min} to x_{max}.
RandomReal[x_{max}] gives a pseudorandom real number in the range 0 to x_{max}.
RandomReal[*range*, *n*] gives a list of *n* pseudorandom reals.
RandomReal[*range*, {n_1, n_2, …}] gives an $n_1 \times n_2 \times \ldots$ array of pseudorandom reals.
RandomReal[*dist*, …] samples from the symbolic continuous distribution *dist*. ≫

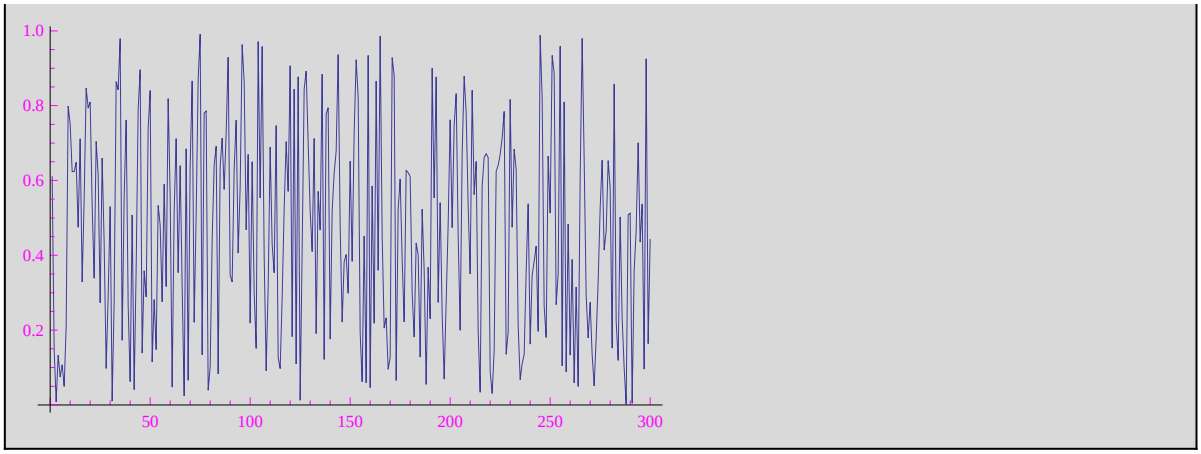
Nota: in realtà sono numeri "pseudocasuali", poiché l'algoritmo che li genera è deterministico.

Quindi:

In[10]:=
```
whitenoise[n_] := (*genera una lista di n numeri reali casuali tra 0 e 1*)Table[
  RandomReal[],
  {k, n}
 ]
```

In[11]:=
```
whitenoiseplot = ListLinePlot[
  whitenoise[300]
 ]
```

Out[11]=



Per il calcolo del valor medio e della varianza, *Mathematica* dispone dei comandi **Mean[]** e **Variance[]**:

In[12]:=
```
media[n_] := Mean[whitenoise[n]]; varianza[n_] := Variance[whitenoise[n]]
```

In[13]:=
```
media[300]
```

Out[13]=
```
0.508196
```

In[14]:=
```
varianza[300]
```

Out[14]=
```
0.0837115
```

In questo modo non abbiamo un controllo sul valor medio e sulla varianza. Ricordiamo che quest'ultima è la potenza media del segnale, e potrebbe essere assegnata a priori. Possiamo generare un white noise di potenza media assegnata, forzando *Mathematica* a generare una lista di numeri reali casuali con varianza nota. A tale scopo definiamo la funzione:

In[15]:=
```
random[μ_, σ_] := RandomReal[
  NormalDistribution[μ, σ]
 ]
```

In[16]:=
```
Clear[whitenoise, whitenoiseplot]
```

Assumendo un valor medio nullo:

In[17]:=
```
whitenoise[σ_, n_] := (*genera una lista di n numeri
   reali casuali tra 0 e 1 di valor medio 0 e varianza σ^2*)Table[
   random[0, σ],
   {n}
  ];
```

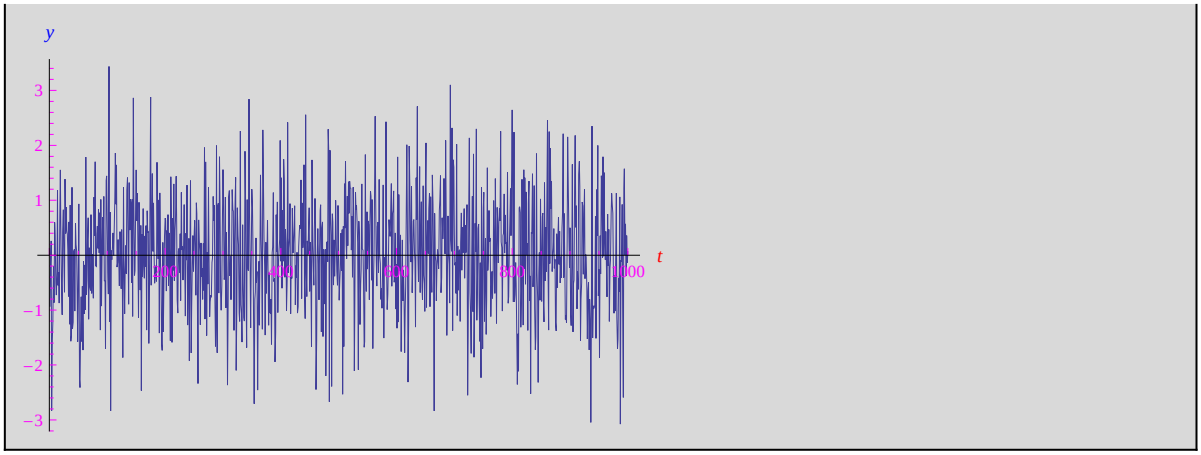
Per graficarne l'andamento, definiamo una funzione che ha per argomento l'ordine *n* della lista definita sopra.

In[18]:=
```
whitenoiseplot[n_] := ListLinePlot[
  (*assumo σ=1*)
  whitenoise[1, n],
  AxesLabel →
   {
    Style["t", Small, Red],
    Style["y", Small, Blue]
   }
 ]
```

Per $n = 1000$:

In[19]:=
```
whitenoiseplot[1000]
```

Out[19]=



In[20]:=
```
media[n_] := Mean[whitenoise[1, n]]; varianza[n_] := Variance[whitenoise[1, n]]
```

In[21]:=
```
media[10]
```

Out[21]=
```
-0.085802
```

In[22]:=
```
varianza[10]
```

Out[22]=
```
1.93308
```

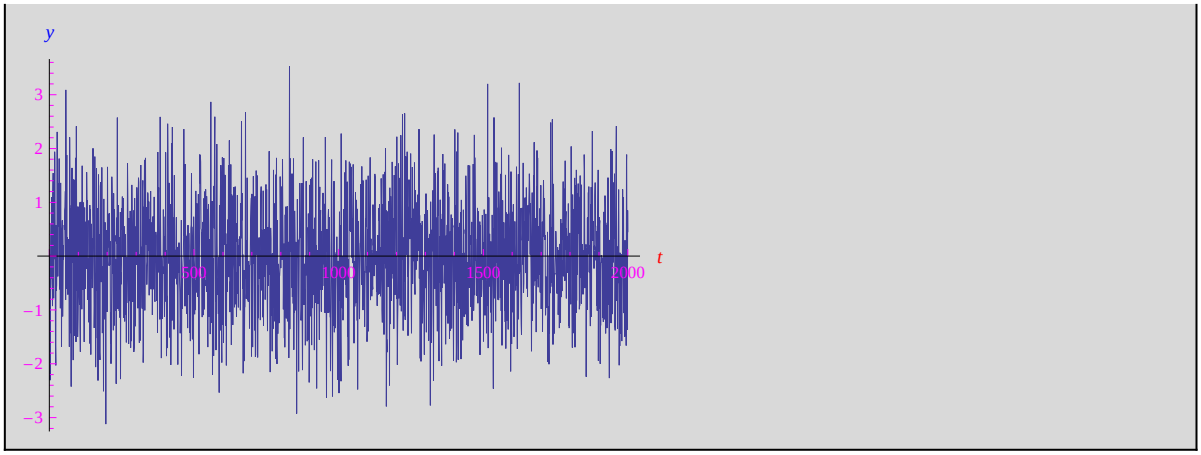
L'approssimazione migliora al crescere di n :

In[23]:=
```
Table[
 {
  media[n],
  varianza[n]},
 {n, 2, 9000, 1000}
]
```

Out[23]=
```
{{1.89759, 0.123445}, {0.0226948, 0.988648}, {-0.020998, 1.01758},
 {-0.014359, 1.03678}, {0.0131261, 0.972019}, {-0.0198803, 1.00431},
 {0.00575883, 0.983967}, {0.0071523, 1.00958}, {-0.00555671, 1.03456}}
```

Assumendo $n = 2000$:

In[24]:=
```
whitenoiseplot[2000]
```

Out[24]=


In[25]:=
```
y = Interpolation[whitenoise[1, 2000]]
```

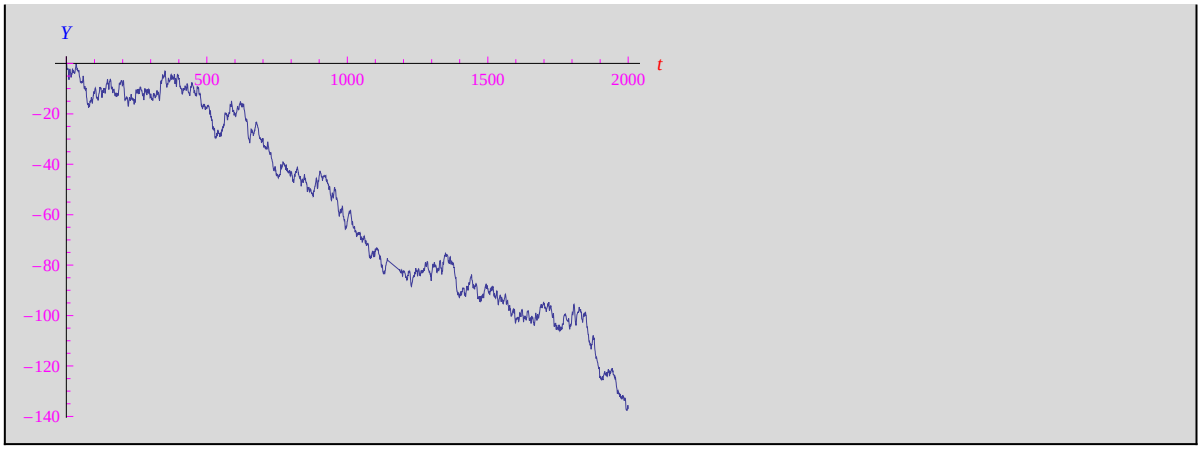
Out[25]=
```
InterpolatingFunction[{{1., 2000.}}, <>]
```

In[26]:=
```
Y[t_] = Integrate[y[t], t]
```

Out[26]=
```
InterpolatingFunction[{{1., 2000.}}, <>][t]
```

In[27]:=
```
plotY = Plot[
  Y[t],
  {t, 1, 2000},
  AxesLabel →
   {
    Style["t", Small, Red],
    Style["Y", Small, Blue]
   }

 ]
```

Out[27]=


In[28]:=
```
Dy[t_] = D[y[t], t]
```

Out[28]=
```
InterpolatingFunction[{{1., 2000.}}, <>][t]
```

In[29]:=
```
Plot[
 Dy[t],
 {t, 1, 2000}
]
```

Out[29]=