Introduction to Computer Algebra and Mathematica

part 2. Using Mathematica to understand some interesting physics

Edoardo Milotti

Dipartimento di Fisica, Università di Trieste

1. Mathematica is good at defining recursivity

Consider for instance the factorial. The standard recursive definition

$$0! = 1$$
$$n! = n \times (n-1)!$$

translates to

```
Clear[fact];
fact[0] := 1;
fact[n_] := n fact[n - 1];
```

Try it !

Example: use a recursive definition for the geometric sum to implement division

```
In[•]:= Clear[sum];
        sum[1, r_] = 1;
        sum[n_, r_] := sum[n - 1, r] + r^(n - 1)
```

Recall that

$$\frac{1}{y} \approx 2^{-(\lfloor \log_2 y \rfloor + 1)} \times \sum_{k=0}^{k=k_{\max}} \left(1 - y \times 2^{-(\lfloor \log_2 y \rfloor + 1)}\right)^k$$

```
In[•]:= x = 1738; nstep = 20;
        nbit = BitLength[x];
        result = BitShiftRight[sum[nstep, BitShiftRight[BitShiftLeft[2, nbit - 1] - N[x], nbit]],
            nbit];
        Print[result, "\t", 1 / result]

        0.000575374     1738.
```

Dividere e moltiplicare per 2 in base 2 è facile

$$10 = 5 \times 2 \qquad 5 = 10 : 2$$

$$(1010)_2 = (101)_2 \times (10)_2 \qquad (101)_2 = (1010)_2 : (10)_2$$

Quindi si può calcolare il reciproco nel modo seguente:

$$\frac{1}{11} = \frac{1}{16-5} = \frac{16^{-1}}{1-5\times 16^{-1}} \approx 16^{-1} \sum_{k=0}^{k_{\max}} (5 \times 16^{-1})^k$$

```
In[•]:= x = 1738; nstep = 20;
        nbit = BitLength[x];
        result = BitShiftRight[sum[nstep, BitShiftRight[BitShiftLeft[2, nbit - 1] - N[x], nbit]],
           nbit];
        Print[result, "\t", 1 / result]

        0.000575374     1738.
```

Exercise: write a short program for the Fibonacci sequence

$$F_n = F_{n-1} + F_{n-2}$$

with the initial conditions

$$F_0 = 0; \quad F_1 = 1.$$

Fibonacci sequence

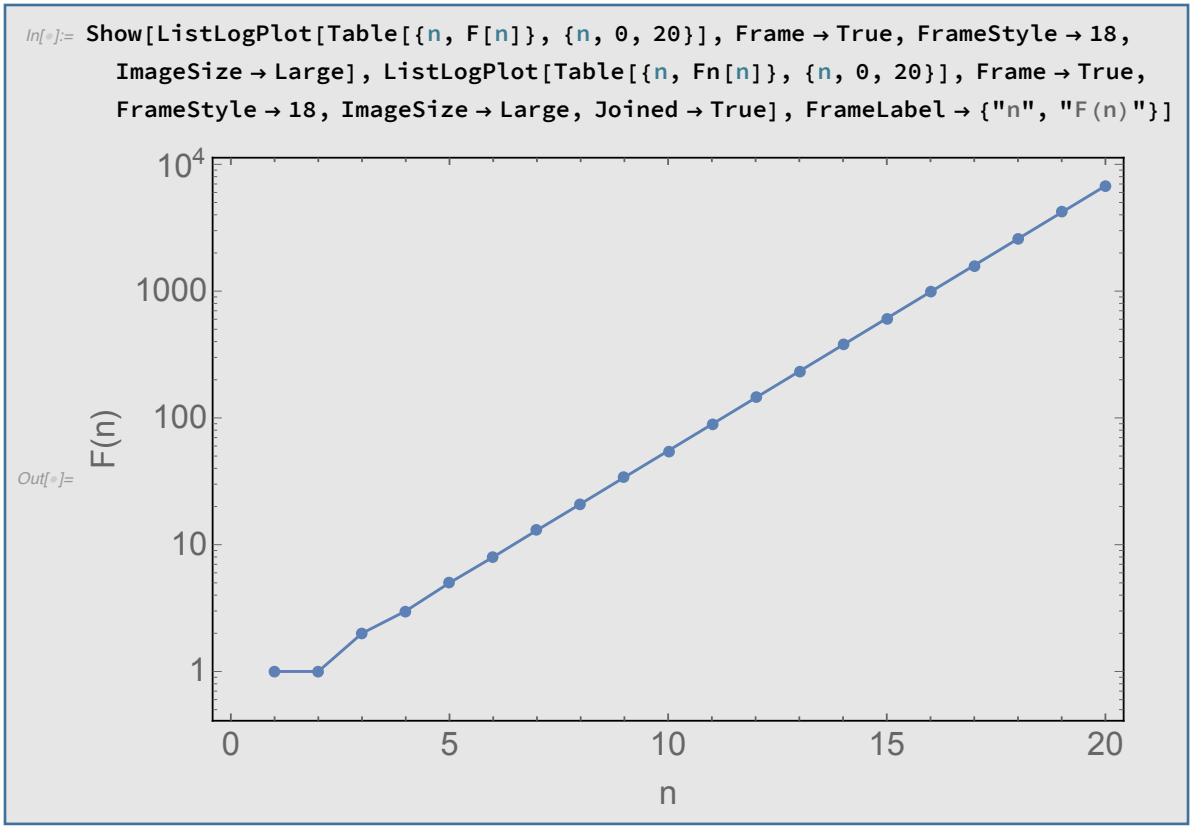
```
In[•]:= Clear[F]; (* recursive definition of the sequence *)
        F[0] := 0;
        F[1] := 1;
        F[n_] := F[n - 1] + F[n - 2];

In[•]:= TableForm[Table[{n, F[n]}, {n, 0, 20}]]
```

Out[•]//TableForm=

0	0
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34
10	55
11	89
12	144
13	233
14	377
15	610
16	987
17	1597
18	2584
19	4181
20	6765

```
In[•]:= Show[ListLogPlot[Table[{n, F[n]}, {n, 0, 20}], Frame → True, FrameStyle → 18,
          ImageSize → Large], ListLogPlot[Table[{n, Fn[n]}, {n, 0, 20}], Frame → True,
          FrameStyle → 18, ImageSize → Large, Joined → True], FrameLabel → {"n", "F(n)"}]
```

Out[•]=



A more complex example: a Linear Congruential Generator (LCG) of uniformly distributed pseudorandom numbers

These generators are defined by the recursive formula

$$x_{n+1} = ax_n + c \mod m$$

and this translates to

```
Clear[gen];
gen[0, a_, c_, m_, n0_] := n0;
gen[n_, a_, c_, m_, n0_] := Mod[a gen[n - 1, a, c, m, n0] + c, m];
```

e.g., with the constants of the LCG described in Numerical Recipes

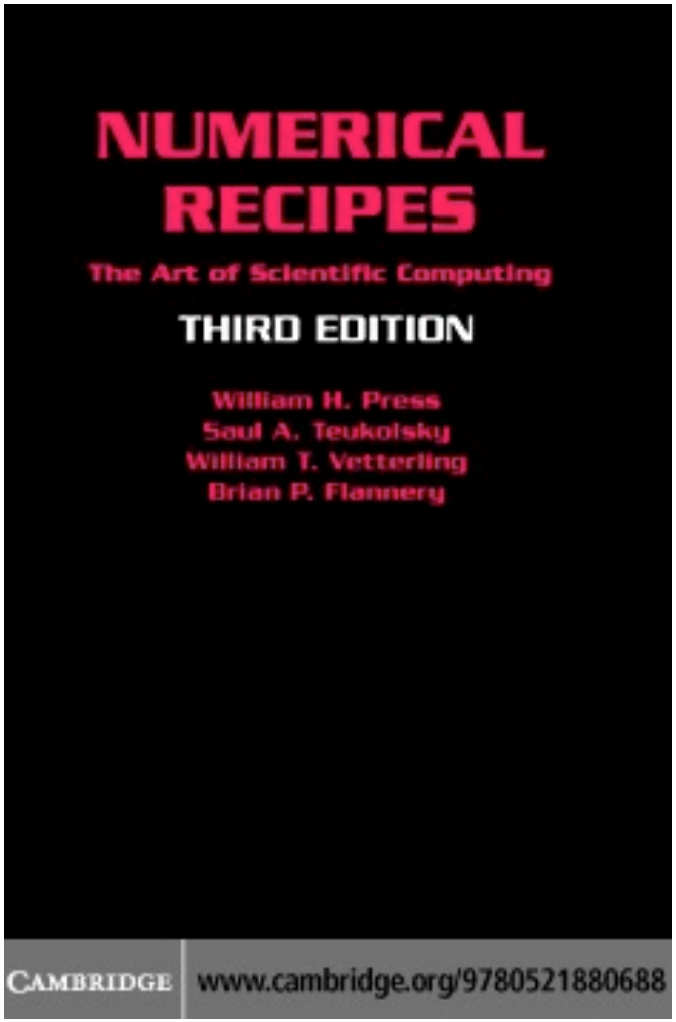
```
gen[10, 1 664 525, 1 013 904 223, 2 ^ 32, 1]
```

```
2 745 540 835
```

For a list of LCG's and some theory see

https://en.wikipedia.org/wiki/Linear_congruential_generator

See also the book Numerical Recipes for a more in-depth explanation



Random number generators are extremely important and useful in computational physics. All important computer languages have libraries of pseudorandom number generators.

For example, the result of `dir(numpy.random)` in Python is

'BitGenerator',
'Generator',
'MT19937',
'PCG64',
'Philox',
'RandomState',
'SFC64',
'SeedSequence',
'__RandomState_ctor',
'__all__',
'__builtins__',
'__cached__',
'__doc__',
'__file__',
'__loader__',
'__name__',
'__package__',
'__path__',
'__spec__',
'_bit_generator',
'_bounded_integers',
'_common',
'_generator',
'_mt19937',
'_pcg64',
'_philox',
'_pickle',
'_sfc64',
'absolute_import',
'beta',
'binomial',
'bytes',
'chisquare',
'choice',
'default_rng',
'dirichlet',
'division',
'exponential',
'f',
'gamma',
'geometric',
'get_state',
'gumbel',
'hypergeometric',
'laplace',
'logistic',
'lognormal',
'logseries',
'mtrand',
'multinomial',
'multivariate_normal',
'negative_binomial',
'noncentral_chisquare',
'noncentral_f',
'normal',
'pareto',
'permutation',
'poisson',
'power',
'print_function',
'rand',
'randint',
'randn',
'random',
'random_integers',
'random_sample',
'ranf',
'rayleigh',
'sample',
'seed',
'set_state',
'setup',
'shuffle',
'standard_cauchy',
'standard_exponential',
'standard_gamma',
'standard_normal',
'standard_t',
'test',
'tests',
'triangular',
'uniform',
'vonmises',
'wald',
'weibull',
'zipf'

```
gen[10, 1664525, 1013904223, 2^32, 1]
```

```
2745540835
```

Evaluation of the 10th pseudorandom number

```
2^32 // N
```

$$4.29497 \times 10^{9}$$

Numerical evaluation of *m*

```
gen[10, 1664525, 1013904223, 2^32, 1] // N
```

$$2.74554 \times 10^{9}$$

Numerical evaluation of the 10th pseudorandom number

```
gen[10, 1664525, 1013904223, 2^32, 1] / 2^32 // N
```

```
0.639246
```

Pseudorandom number in [0,1)

```
N[gen[10, 1664525, 1013904223, 2^32, 1] / 2^32]
```

```
0.639246
```

Pseudorandom number in [0,1)

```
N[gen[10, 1664525, 1013904223, 2^32, 1] / 2^32, 10]
```

```
0.6392460398
```

Higher precision

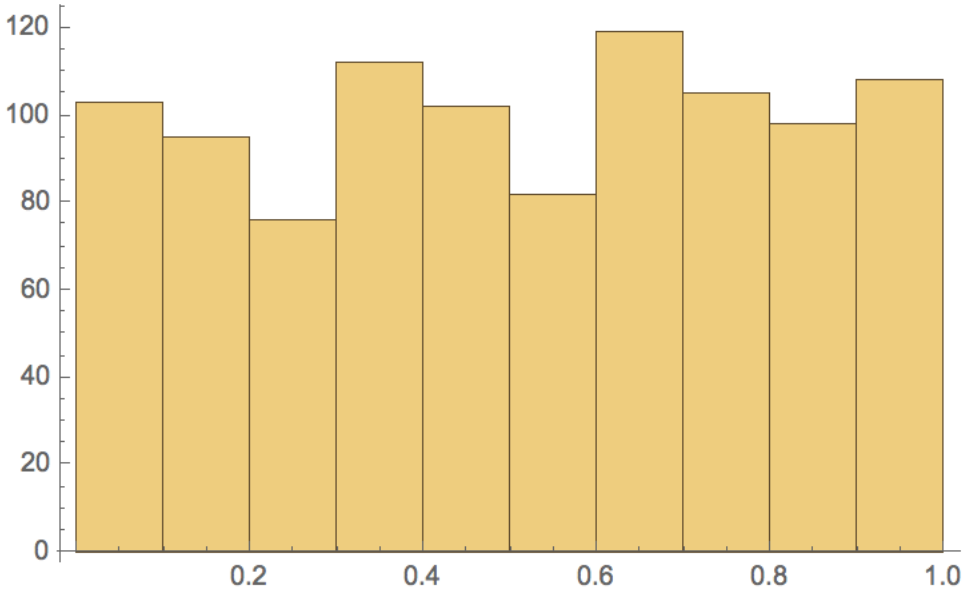
Printout of the first 10 pseudorandom numbers

```
For[k = 1, k ≤ 10,
 Print[k, "\t", gen[k, 1 664 525, 1 013 904 223, 2 ^ 32, 1], "\t",
  gen[k, 1 664 525, 1 013 904 223, 2 ^ 32, 1] / 2 ^ 32 // N];
 k ++]
```

```
1     1 015 568 748     0.236456
2     1 586 005 467     0.369271
3     2 165 703 038     0.504242
4     3 027 450 565     0.704883
5     217 083 232      0.0505436
6     1 587 069 247     0.369518
7     3 327 581 586     0.774763
8     2 388 811 721     0.556189
9     70 837 908      0.0164932
10     2 745 540 835     0.639246
```

Histogram of the first 1000 pseudorandom numbers

```
rnglist = Table[0, {1000}];

For[k = 1, k ≤ 1000,
 rnglist[[k]] = gen[k, 1664525, 1013904223, 2^32, 1] / 2^32 // N;
 k++]

Histogram[rnglist]
```



10 bins
Expected RMS fluctuation = 10

Recursion is not unlimited

```
gen[1000, 1664525, 1013904223, 2^32, 1]
```

```
645503657
```

```
gen[2000, 1664525, 1013904223, 2^32, 1]
```

```
$RecursionLimit: Recursion depth of 1024 exceeded during evaluation of
  1664525 gen[978 - 1, 1664525, 1013904223, 4294967296, 1] + 1013904223.
```

```
Hold[Mod[
  1664525 gen[2000 - 1, 1664525, 1013904223, 4294967296, 1] + 1013904223, 4294967296]]
```

```
$RecursionLimit
```

```
1024
```

```
$RecursionLimit = 2048
```

```
2048
```

```
gen[2000, 1664525, 1013904223, 2^32, 1]
```

```
2926278225
```

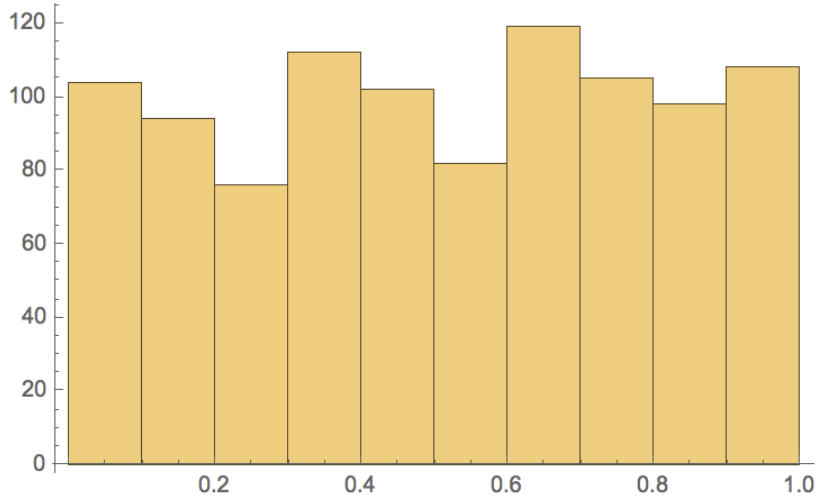
Practical recursivity

```
n = 1000;
rnglist = Table[0, {n}];
seed = 1;
rng = seed;
rnglist [[1]] = seed / 2 ^ 32 // N;
For[k = 2, k ≤ n,
 rng = Mod[a * rng + c, m] /. {a -> 1 664 525, c -> 1 013 904 223, m -> 2 ^ 32};
 rnglist[[k]] = rng / 2 ^ 32 // N;
 k ++]
```

```
Take[rnglist, {1, 20}]
```

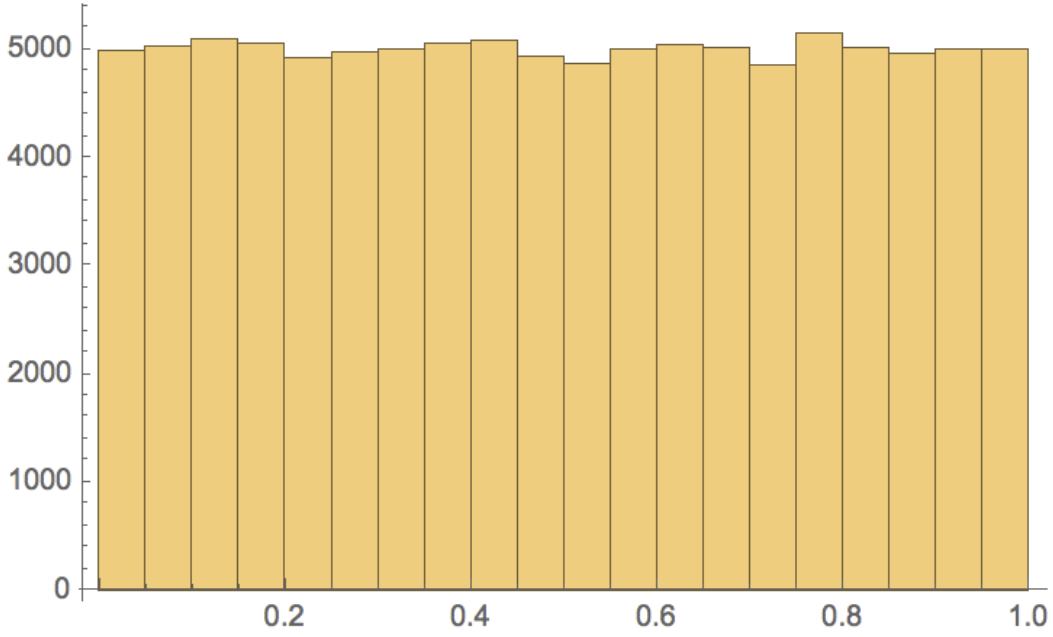
$$\{2.32831 \times 10^{-10}, 0.236456, 0.369271, 0.504242, 0.704883, 0.0505436,$$
$$0.369518, 0.774763, 0.556189, 0.0164932, 0.639246, 0.250451, 0.422378,$$
$$0.59069, 0.836934, 0.235076, 0.980846, 0.860887, 0.326876, 0.682603\}$$

```
Histogram[rnglist]
```

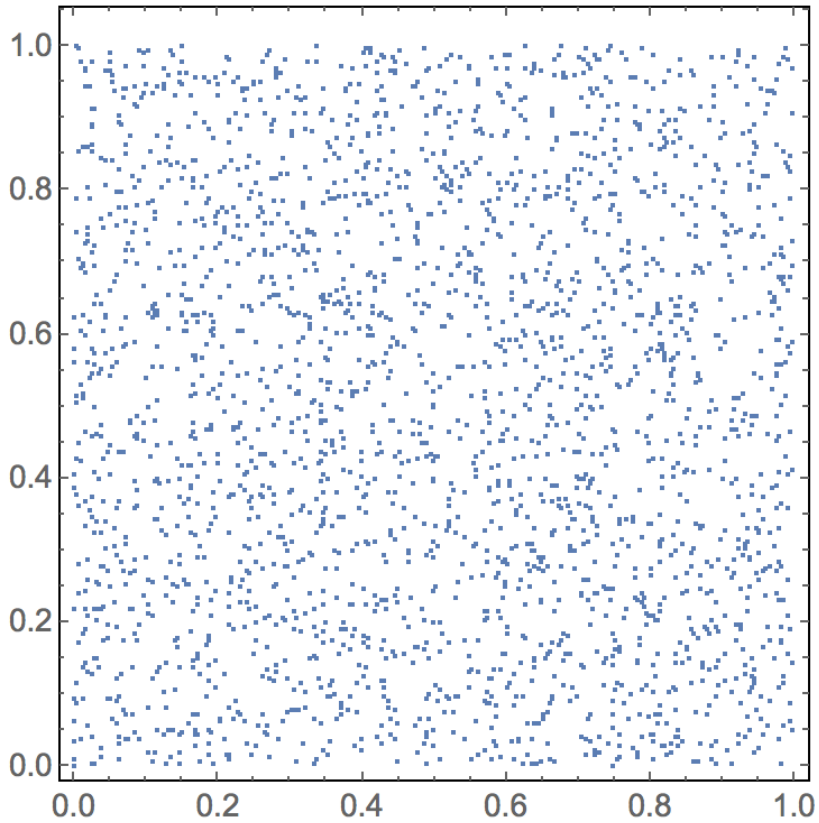


```
n = 100 000;
rnglist = Table[0, {n}];
seed = 1;
rng = seed;
rnglist [[1]] = seed / 2 ^ 32 // N;
For[k = 2, k ≤ n,
 rng = Mod[a * rng + c, m] /. {a -> 1 664 525, c -> 1 013 904 223, m -> 2 ^ 32};
 rnglist[[k]] = rng / 2 ^ 32 // N;
 k ++]
```

```
Histogram[rnglist]
```

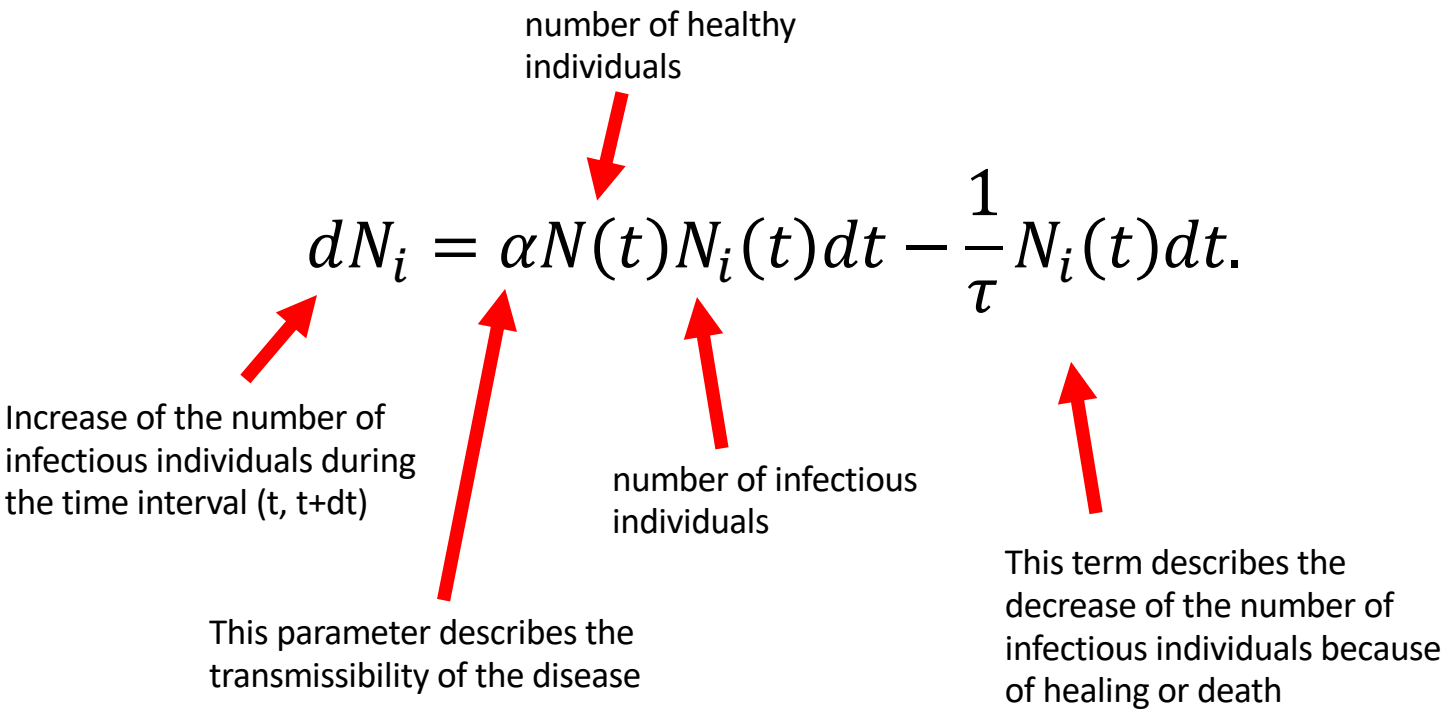


```
n = 10 000;
rnglist = Table[0, {n}];
seed = 1;
rng = seed;
rnglist [[1]] = seed / 2 ^ 32 // N;
For[k = 2, k ≤ n,
   rng = Mod[a * rng + c, m] /. {a -> 1664525, c -> 1013904223, m -> 2 ^ 32};
   rnglist[[k]] = rng / 2 ^ 32 // N;
   k += 2];
x = Drop[rnglist, -n / 2];
y = Drop[rnglist, n / 2];

ListPlot[Transpose[{x, y}], AspectRatio → Automatic, Frame → True, FrameStyle → 14]
```



2. Mathematical modeling: simple model of epidemic outbreak (SIR model)

Number of infectious (sick) individuals in a population



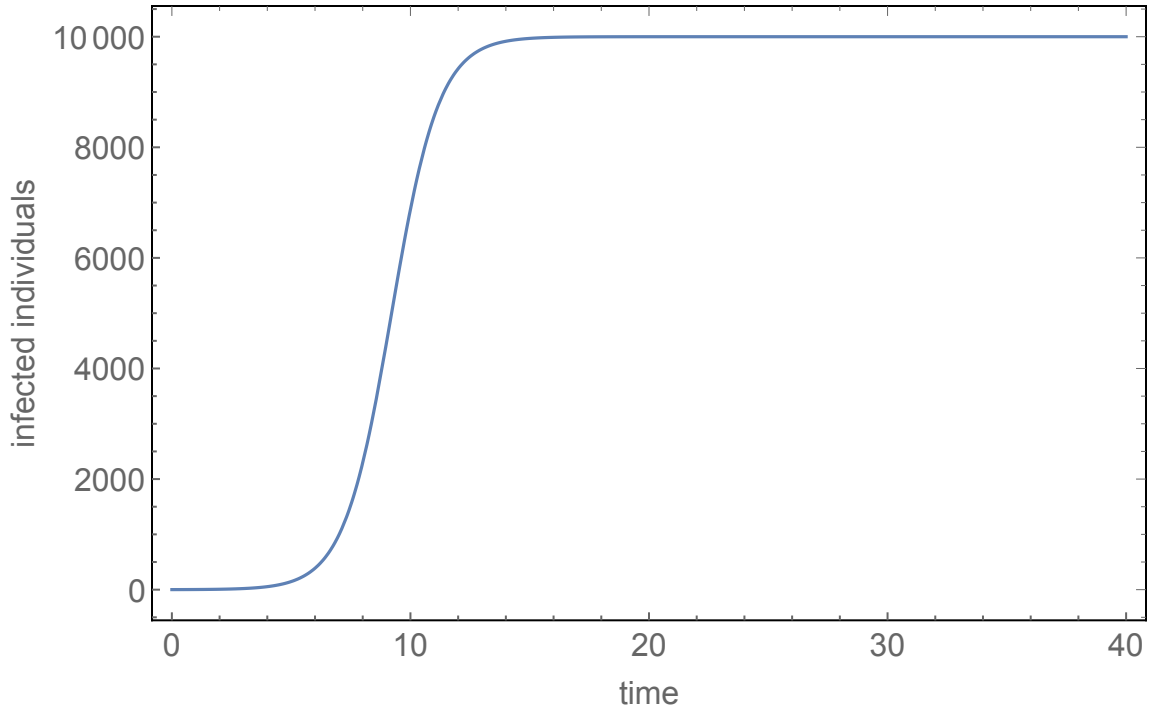
The logistic equation

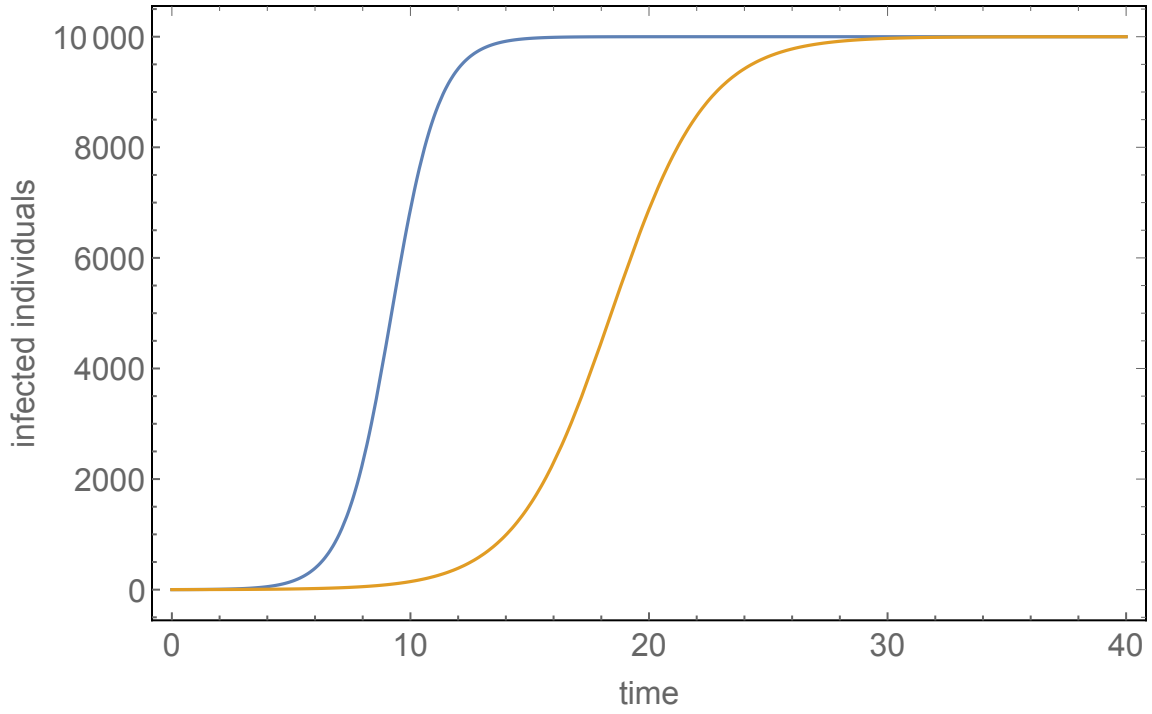
$$dN_i = \alpha N(t) N_i(t) dt - \frac{1}{\tau} N_i(t) dt.$$

discard the last term (irrealistic but interesting assumption)

$$dN_i = \alpha [N_0 - N_i(t)] N_i(t) dt,$$

Logistic equation: its exact solution is well-known


10 000
8000
6000
4000
2000
0
infected individuals
0
10
20
30
40
time


10 000
8000
6000
4000
2000
0
infected individuals
0
10
20
30
40
time

Complete differential system

$$\begin{cases} dN = -\alpha N(t) N_i(t) dt \\ dN_i = \alpha N(t) N_i(t) dt - \frac{1}{\tau} N_i(t) dt \\ dN_a = \frac{1}{\tau} N_i(t) dt \end{cases}$$

Equation for the number of healthy individuals

Equation for all those who are no longer infections (either healed or dead)

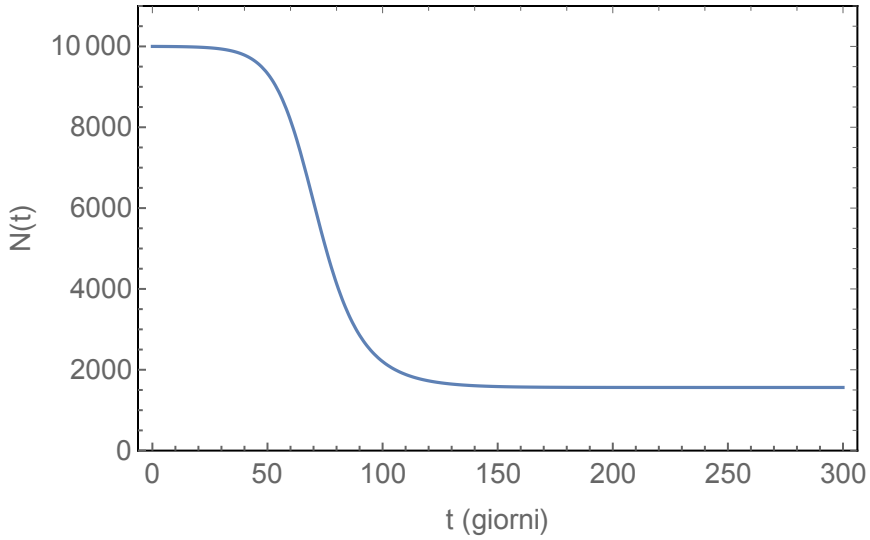
This "effective exponent" is critical, if it is negative, then the number of infectious individuals decreases spontaneously

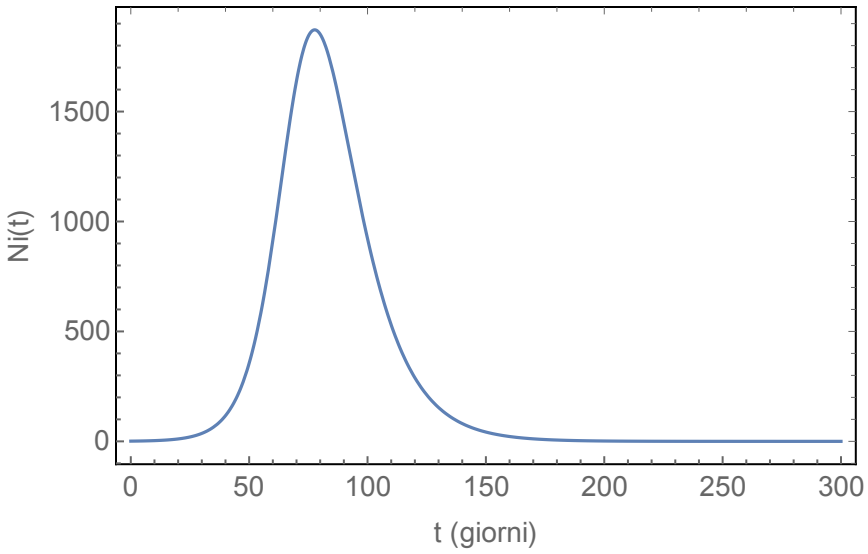
$$dN_i = \left[\alpha N(t) - \frac{1}{\tau}\right] N_i(t)dt\,,$$

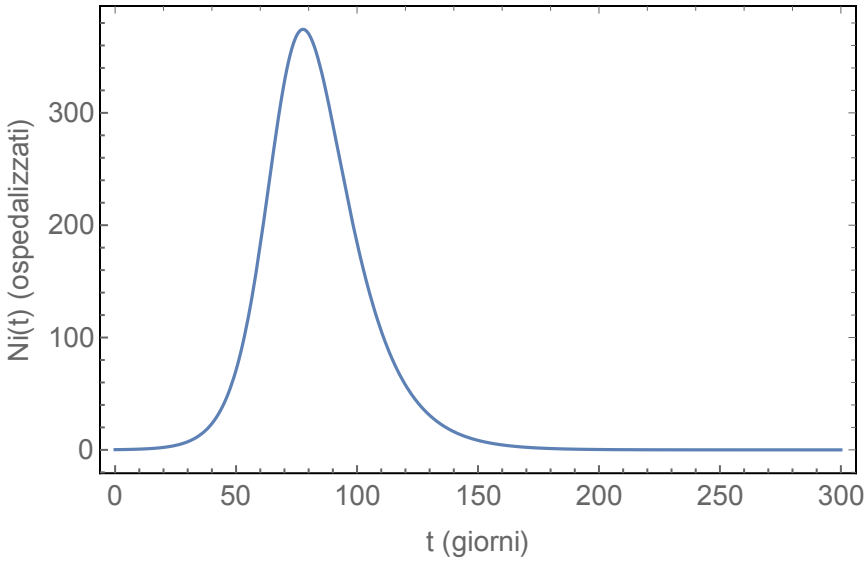
Numerical solution of the differential system with Mathematica (this snapshot also shows the numerical values of the parameters used in the integration)

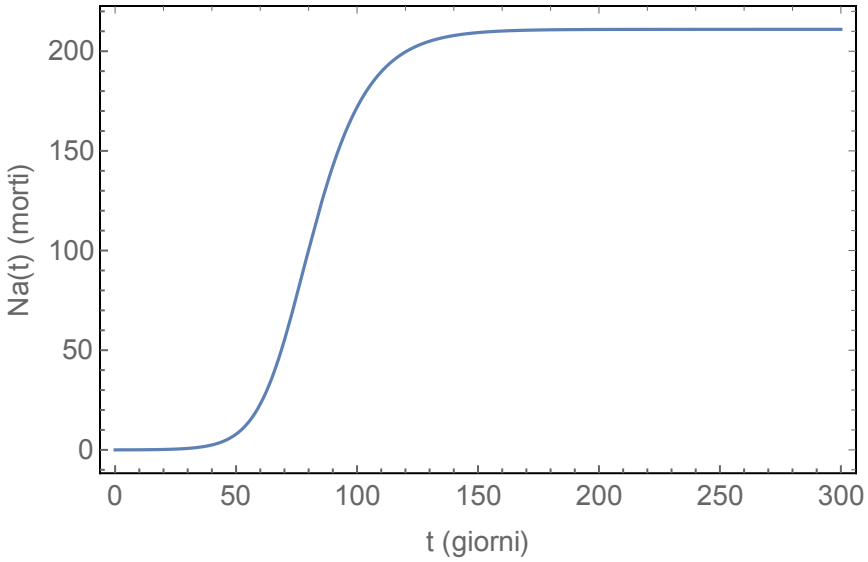
```
Clear[alpha]; (* definizione del parametro di contagiosità, con mitigazione intorno al tempo t0;
la funzione interpola in modo continuo (e con derivata centrale dadt) tra il valore iniziale
 e quello finale*)
alpha[t_, t0_] := alpha0 + 0.5 (1 + Tanh[dadt * (t - t0)]) (alpha1 - alpha0);

system := {n'[t] == -alpha[t, t0] * n[t] * ni[t], ni'[t] == alpha[t, t0] * n[t] * ni[t] - ni[t] / tau,
  na'[t] == ni[t] / tau, n[0] == n00, ni[0] == ni0, na[0] == 0};
(* definizione del sistema differenziale (equazioni + condizioni iniziali) *)

n0 = 10 000; (* numero di individui nella popolazione *)
ni0 = 1; (* numero iniziale di individui infetti *)
tau = 10; (* tempo medio di durata della malattia e di durata dell'infettività, in giorni *)
alpha0 = 2.2 / tau / (n0 - ni0); (* valore iniziale del parametro di contagiosità *)
alpha1 = 0.22 / tau / (n0 - ni0); (* valore finale del parametro di contagiosità *)
t0 = 50; (* tempo intorno al quale si mettono in atto le misure di mitigazione *)
dadt = 0.2; (* derivata centrale per la funzione alpha *)
n00 = n0 - ni0; (* numero iniziale di individui sani, mai infettati *)
tmax = 300; (* tempo massimo fino a cui si estende l'integrazione, in giorni *)
beta = 0.025; (* mortalità media *)
gamma = 0.2; (* ospedalizzazione media *)

sol = NDSolve[system, {n, ni, na}, {t, 0, tmax}] (* soluzione numerica del sistema differenziale *)
```

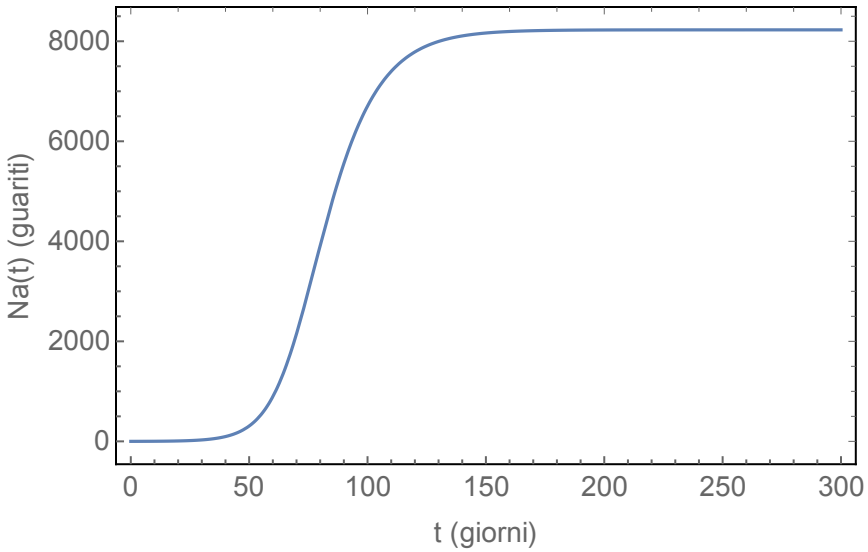
Infectious diseases with parameters similar to COVID-19, without mitigation

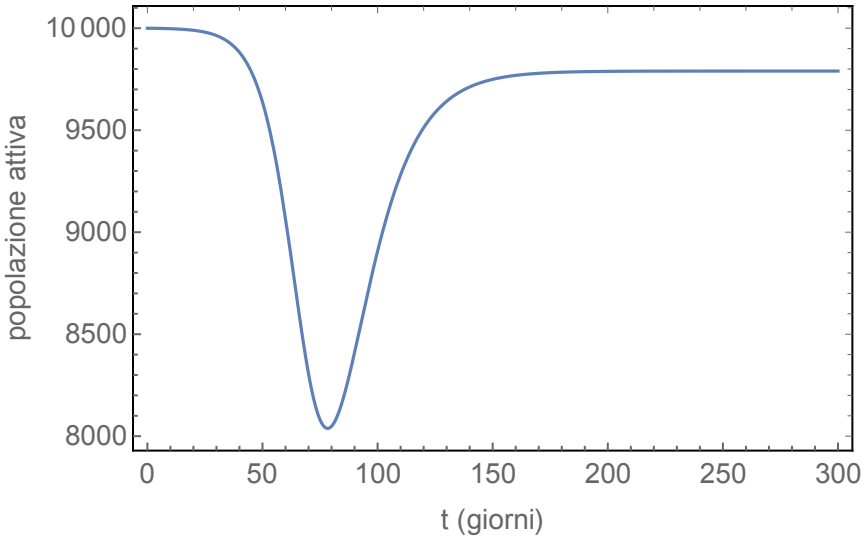



1500
1000
500
0
Ni(t)
0
50
100
150
200
250
300
t (giorni)

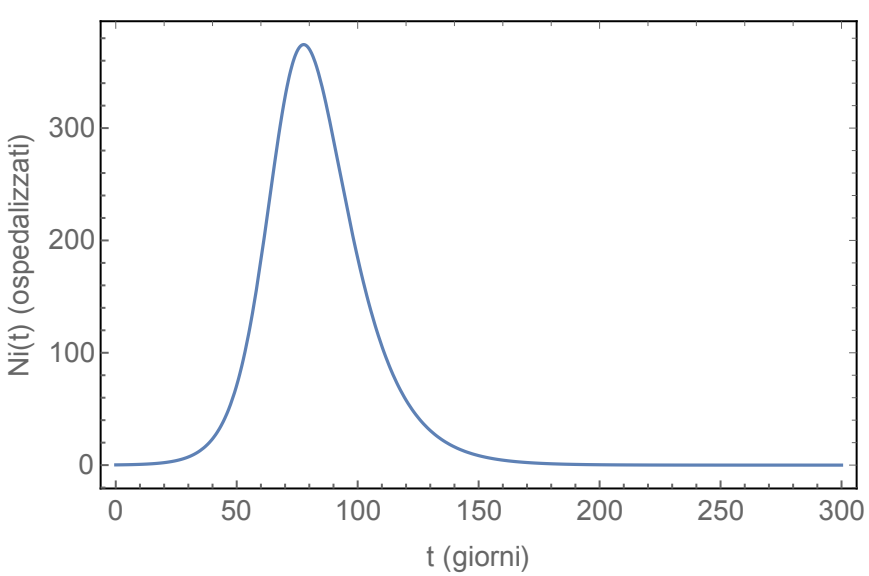

Ni(t) (ospedalizzati)
300
200
100
0
0
50
100
150
200
250
300
t (giorni)

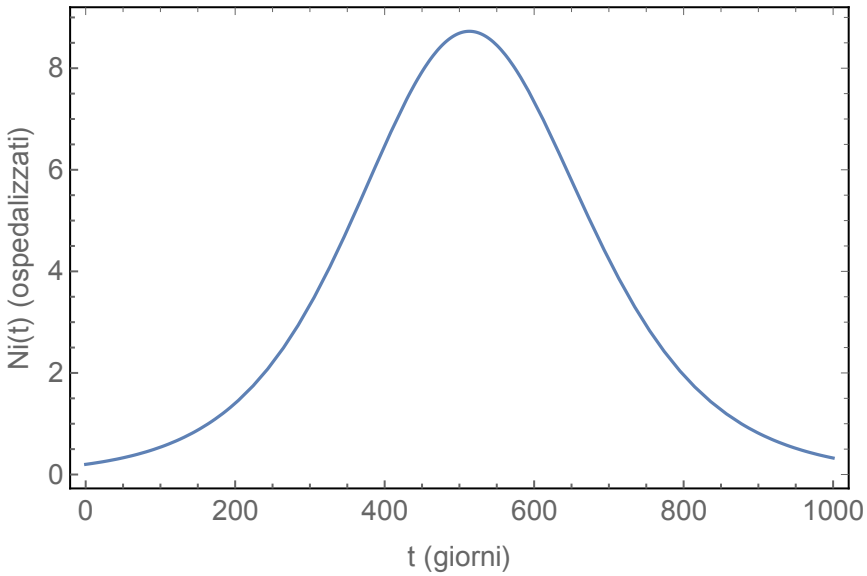

200
150
100
50
0
Na(t) (morti)
0
50
100
150
200
250
300
t (giorni)


8000
6000
4000
2000
0
Na(t) (guariti)
0
50
100
150
200
250
300
t (giorni)


10 000
9500
9000
8500
8000
popolazione attiva
0
50
100
150
200
250
300
t (giorni)

Effect of halving the contagiousness (no mitigation)





https://demonstrations.wolfram.com

WOLFRAM Demonstrations Project

12,000+ Open Interactive Demonstrations
Powered by Notebook Technology »

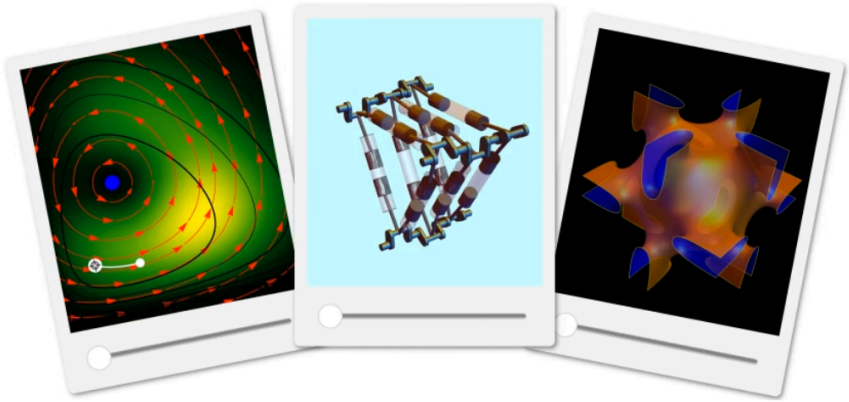
Search | TOPICS | LATEST | ABOUT | PARTICIPATE | AUTHORING AREA

Download

Bringing Ideas to Life

12,000+ Interactive Wolfram Notebooks for education, research, recreation and more

Selected and curated by Wolfram Research



View flyover video »

BROWSE TOPICS

Mathematics
Algebra | Calculus & Analysis ...

Business & Social Systems
Economics | Finance

Creative Arts
Art | Architecture | Music ...

https://demonstrations.wolfram.com/