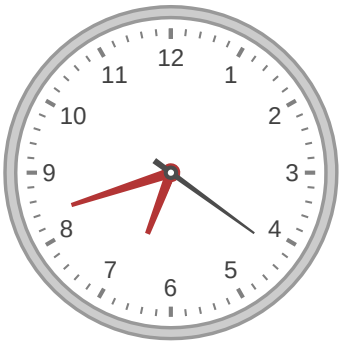
# *Avances en las matemáticas computacionales y su influencia en otras ramas de las ciencias*

Guillermo Sánchez (http://diarium.usal.es/guillermo)

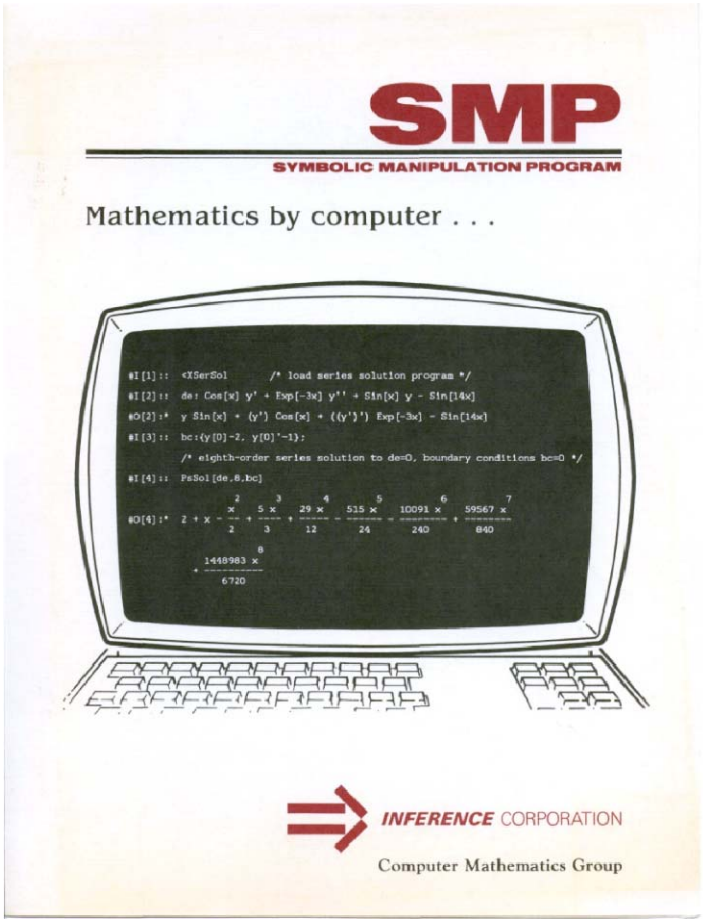
Presentación el 2013-11-11 en:


- Ejemplos de aplicación los avances de computación con Mathematica en distintos campos: astronomía, genética, finanzas...cálculos en paralelo y conexiones en malla/grid. Los ejemplos que se muestran proceden del libro **Mathematica más allá de las matemáticas** (editado por Addlink Software Científico)

## *Los albores del cálculo simbólico*

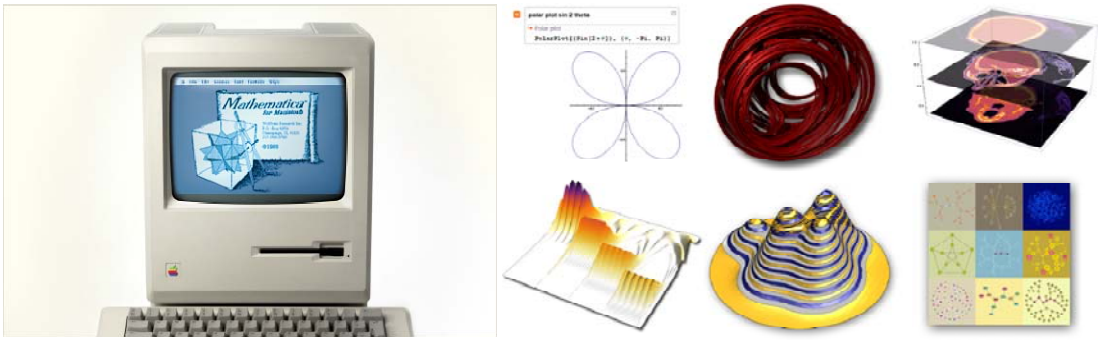
Las computadoras y el software durante años se orientaron al cálculo númerico. FORTRAN se convirtió en el rey de los lenguajes de cálculo científico. En las decadas de 1960 y 1970 nuevos surgen nuevos lenguajes (LISP, ALGOL, APL etc) que son utilizados en el desarrollo de los primeros programas de programación simbólica (REDUCE, MACSYMA). En este ambiente surge en el CALTECH em 1981 SMP (Symbolic Manipulation Program) programado en C por S. Wolfram y C. Cole.

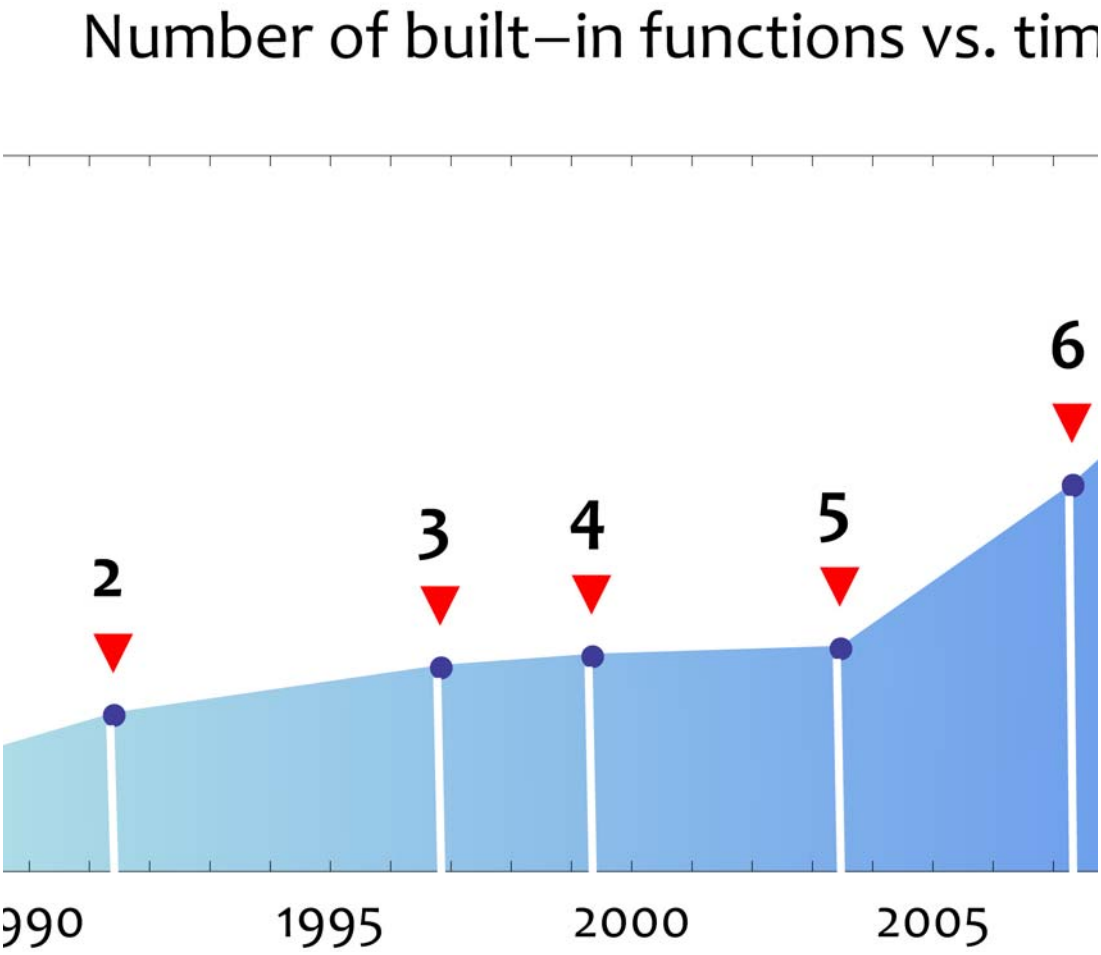


## *Mathematica*, 1988-2013

SMP fue un ensayo general del que saldría *Mathematica* (v1, 1988 ). En las primeras versiones estaba fundamentalmente orientado al cálculo simbólico.

En su desarrollo posterior ha evolucionado hasta convertirse en un lenguaje integral que puede ser aplicado en cualquier campo.




>3500 en la versión 9

# *Mathematica, el formato libre o lingüistico*

**solve x^2 + 2 x –1 = 0**

Result (1 of 2)

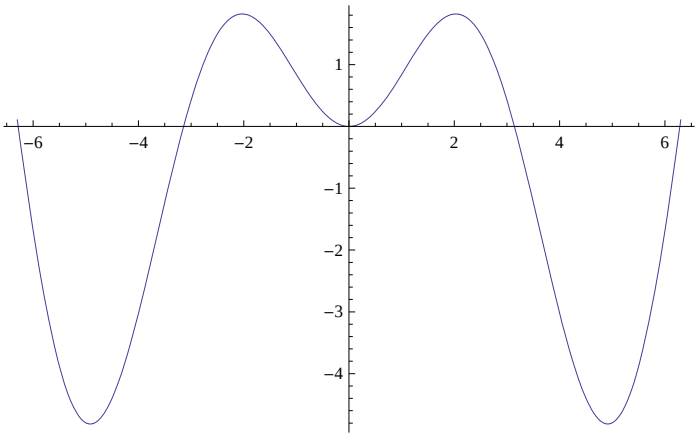
```
{Reduce[-1 + 2*x + x^2 == 0, x],
 N[Reduce[-1 + 2*x + x^2 == 0, x]]}
```

$$\left\{x == -1 - \sqrt{2} \;||\; x == -1 + \sqrt{2},\; x == -2.41421 \;||\; x == 0.414214\right\}$$

**plot x sin x**

Plots (1 of 2)

```
Plot[x*Sin[x], {x, -6.3, 6.3}]
```




Escriba en formato libre lo que desea

**volume of a cylinder** »

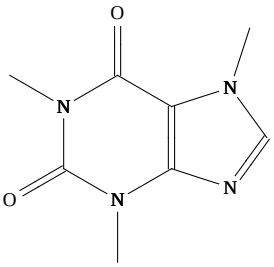
Result

$V = \pi a^2 h$

(for a circular right cylinder with center at (0, 0, 0), height *h*, radius *a*)

## caffeine molecule

```
ChemicalData["Caffeine"]
```

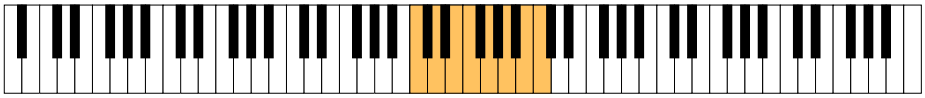



## Do major

```
WolframAlpha["Do major", IncludePods → "KeyboardDisplay",
 AppearanceElements → {"Pods"},
 TimeConstraint → {30, Automatic, Automatic, Automatic}]
```

Keyboard display:

Play sound



# Colecciones

## » Palabras

```
DictionaryLookup[{"Spanish", "añ" ~~ __ ~~ "r"}]
```

```
{añacear, añadir, añascar, añedir, añejar, añidir, añilar,
 añir, añirar, añorar, añublar, añudador, añudar, añusgar}
```

```
trascribe = SpokenString[Sqrt[x / (y + z)]]
```

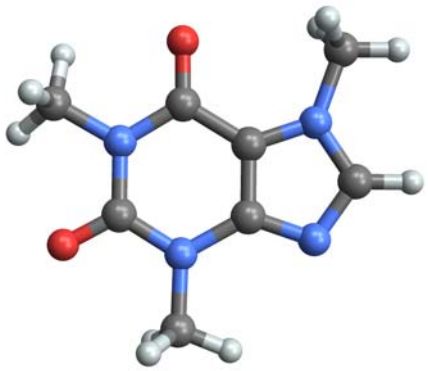
```
square root of the quantity x over the quantity y plus z
```

```
Row[WordData[#, "PhoneticForm"] & /@
  StringSplit[trascribe], "  "]
```

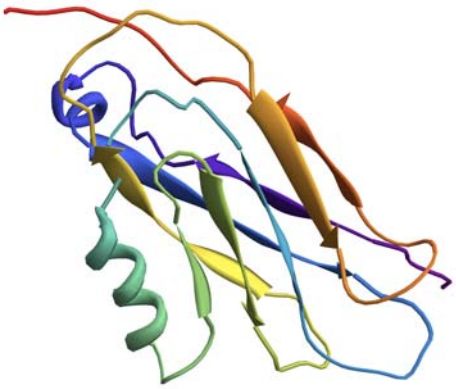
```
skwˈɛr rˈut ˈʌv ðə kwˈɒntəti ˈɛks ˈoʊvɝ ðə kwˈɒntəti wˈaɪ plˈʌs zˈi
```

## » Química, biología

```
ChemicalData["Caffeine", "MoleculePlot"]
```



```
ProteinData["A2M", "MoleculePlot"]
```



```
GenomeData["ACAA2", "BiologicalProcesses"]
```

```
{CholesterolBiosyntheticProcess,
 FattyAcidMetabolicProcess,
 LipidMetabolicProcess, MetabolicProcess}
```

## » Finanzas

Los datos se ajustan automaticamente a una distribución dada (tipicamente es utilizado metodos de máxima verosimilitud)

---

Ajustamos los datos de la cotización de Google a una distribucion lognormal :

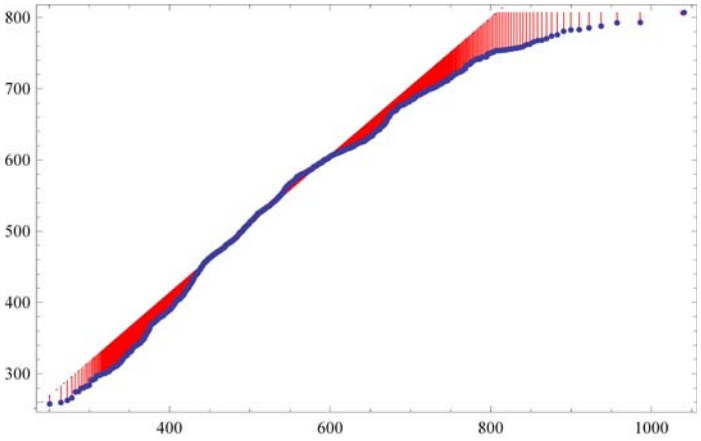
```
googleStock = FinancialData["GOOG",
    {{2006, 3, 10}, {2013, 2, 20}, "Day"}, "Value"];
```

```
dist = EstimatedDistribution[
  googleStock, LogNormalDistribution[μ, σ]]
```

```
LogNormalDistribution[6.23598, 0.212271]
```

Vemos que el ajuste es bueno excepto para altos cuantiles

```
QuantilePlot[googleStock, dist,
 Filling → Automatic , FillingStyle → Red]
```



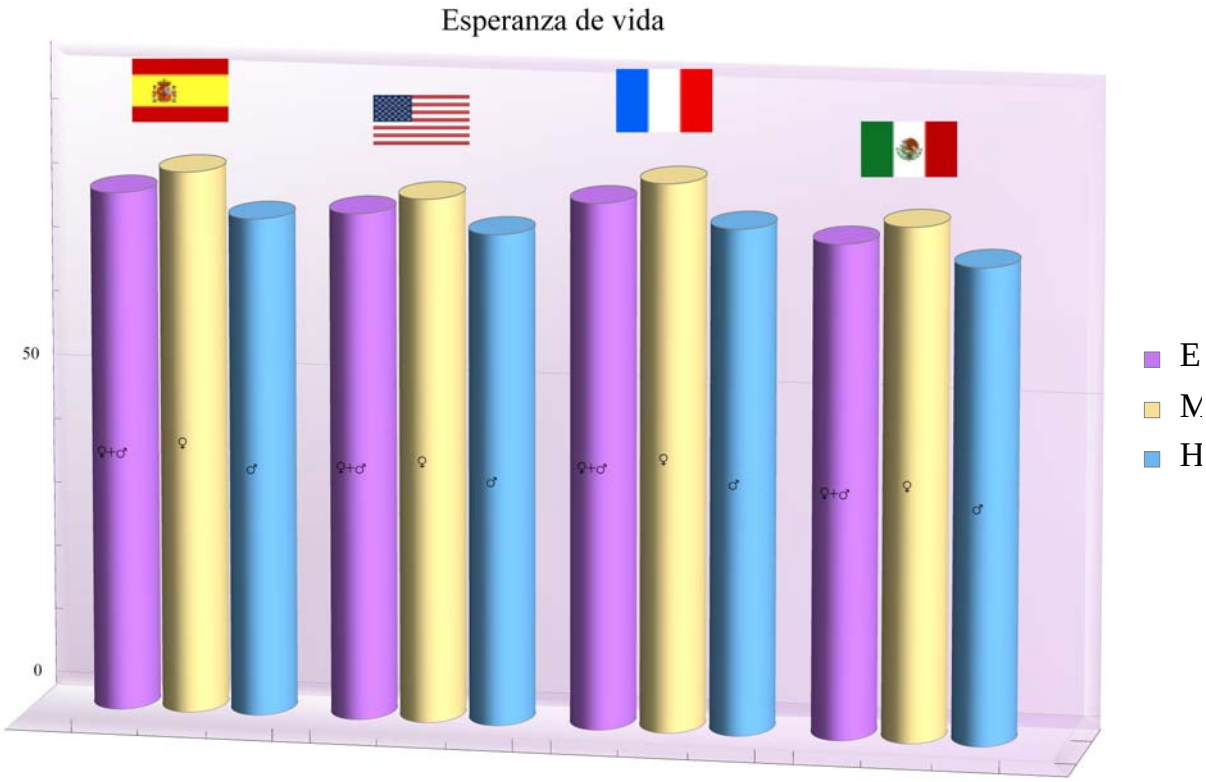

## » El mundo, ciudades, paises, economía, población, ..

Coordenadas y Localización de una ciudad en Google Map.(si pulsamos se nos abrirá en googlemap la localización de la ciudad

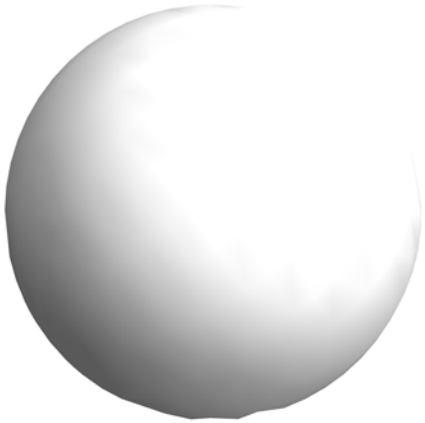
```
Hyperlink[CityData["Seville", "LocationLink"]]
```

```
http://maps.google.com/maps?q=+37.4,-5.98&z=12&t=h
```

» Esperanza de vida (pulsando en las barrar veremos los valores numéricos)




» La pobreza en el mundo (roramos la esfera y podremos ver la distribución de la riqueza)



## » Calcular la probabilidad de vivir 80 años o mas en la población española

Con CountryData se obtienen las cifras de población de hombres, mujeres y el total y la esperanza de vida.

```
{vhombre, vmujer} = Outer[CountryData, {"Spain"},
   {"MaleLifeExpectancy", "FemaleLifeExpectancy"}][[1]]
```

```
{76.74, 83.57}
```

```
{pophombre, popmujer, poptotal} =
 Outer[CountryData, {"Spain"}, {"MalePopulation",
    "FemalePopulation", "Population"}][[1]]
```

$$\{2.21394\times10^7,\ 2.31776\times10^7,\ 4.5317\times10^7\}$$

A partir de estos datos obtenemos la fracción de hombres y mujeres en el total de la población.

```
{fh, fm} = {pophombre / poptotal, popmujer / poptotal}
```

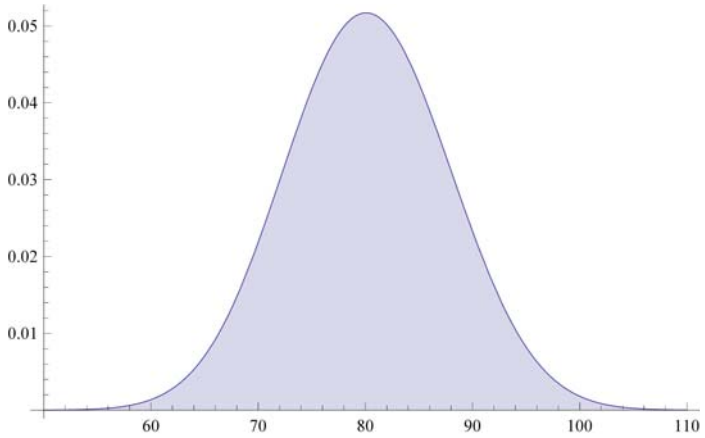
```
{0.488546, 0.511454}
```

Como desviación típica de la esperanza de vida que para edades avanzadas utilizamos (según http://entrenamiento-hector.blogspot.com): 6.7 años para hombres y 6.9 años para mujeres.

Con los datos anteriores obtenemos la distribución conjunta para la esperanza de vida de hombres y mujeres como sigue:

```
esperanzavida = MixtureDistribution[
   {fh, fm}, {NormalDistribution[vhombre, 6.7],
    NormalDistribution[vmujer, 6.9]}];
```

```
Plot[PDF[esperanzavida, x], {x, 50, 110}, Filling → Axis]
```




La probabilidad de que una persona, hombre o mujer, muera con 80 años o

más es:

```
Probability[x ≥ 80, x ≈ esperanzavida]
```

```
0.509822
```

Si consideramos exclusivamente los varones, esta probabilidad es considerablemente menor:

```
Probability[x ≥ 80, x ≈ NormalDistribution[vhombre, 6.7]]
```

```
0.313283
```

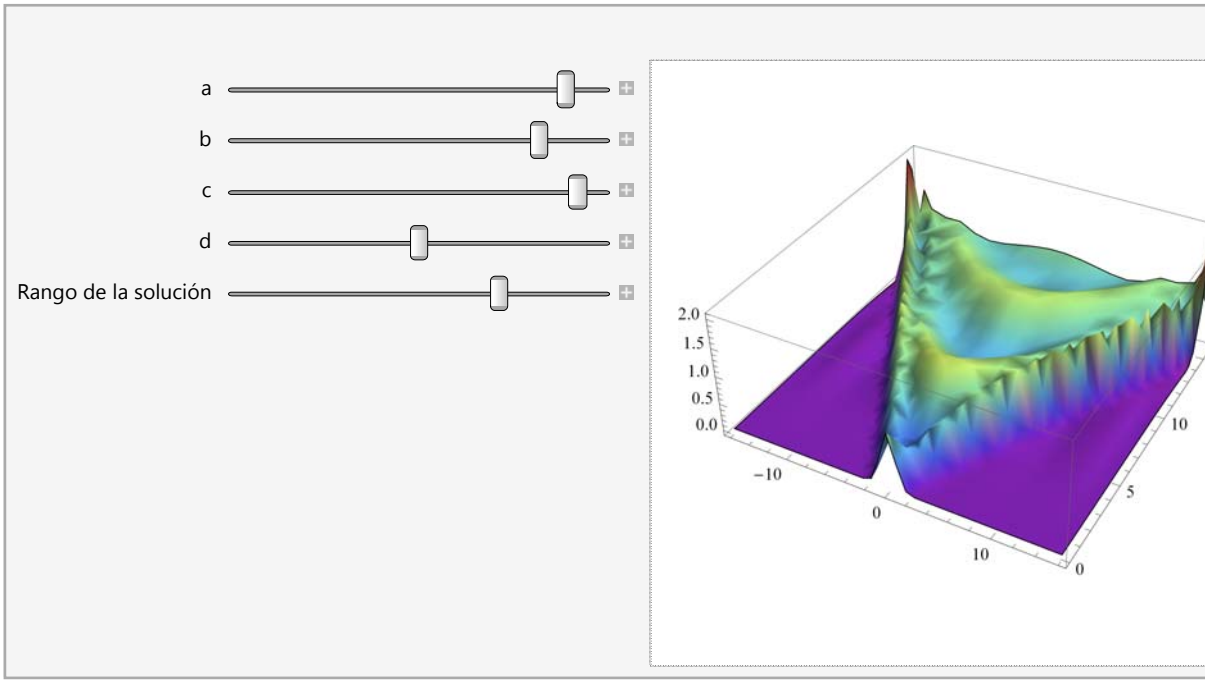
# Modelizacion

Podemos modelar fenomenos realmente complejos. Aquí lo hacemos con un tsunami

$$\frac{\partial^2 u(t, x)}{\partial t^2} = \frac{\partial^2 u(t, x)}{\partial x^2} + a\, u(t, x)^3 + b\, u(t, x)^2 + c\, u(t, x) + d,$$

con $u(0, x) = e^{-x^2}$, $\frac{\partial u(t, x)}{\partial t} = 0$ $u(t, -x_0) = u(t, x_0)$

Resolvemos la ecuación anterior con `NDSolve` dejando como parámetros $a,\ b,\ c,\ d,\ x_0$ . El comportamiento de la ecuación podemos mostrarlo dinámicamente modificando el valor de los parámetros.



# Optimización: El coste del combustible nuclear

En la naturaleza el uranio contiene tres isótopos

```
DeleteCases[
 Transpose[{IsotopeData["U", "Symbol"], IsotopeData["U",
     "IsotopeAbundance"] // Chop}], {_, _Integer}]
```

$$\{\{^{234}\text{U}, 0.000054000\}, \{^{235}\text{U}, 0.0072040\}, \{^{238}\text{U}, 0.992742\}\}$$

Los reactores nucleares normamente utilizan un uranio enriquecido (0.711<U235 %<5). Para ello se necesita uranio natural de partida (feed) y trabajo de separación (UTS) que se calculan como sigue:

$$\text{factorfeed} = \frac{X_p - X_w}{X_f - X_w}$$

$$\text{factorUTS} = \left((2X_p - 1)\,\text{Log}\left[\frac{X_p}{1 - X_p}\right] - (2X_w - 1)\,\text{Log}\left[\frac{X_w}{1 - X_w}\right]\right) - \frac{\left((2X_f - 1)\,\text{Log}\left[\frac{X_f}{1-X_f}\right] - (2X_w - 1)\,\text{Log}\left[\frac{X_w}{1-X_w}\right]\right)(X_p - X_w)}{X_f - X_w}$$

```
utsfeed[P1_, W1_] = Module[{F, uts, feed, P, W},
   P = P1 / 100; W = W1 / 100; F = 0.00711;
uts = ((2 * P - 1) * Log[P / (1 - P)] -
        (2 * W - 1) * Log[W / (1 - W)]) -
((((2 * F - 1) * Log[F / (1 - F)] - (2 * W - 1) * Log[W / (1 - W)]) *
         (P - W)) / (F - W);
feed = (P - W) / (F - W); Thread[
    {"factorUTS", "factorFeed"} -> {uts, feed}]];
```

---

En el siguiente ejemplo se calculan las feed y UTS necesarias para obtener uranio enriquecido al 5 % con una cola de 0.3 %.

```
utsfeed[5, 0.3]
```

$$\{\text{factorUTS} \to 7.19832, \text{factorFeed} \to 11.4355\}$$

*precioUe* = *factorfeed* × *pFeed* + *factorUTS* × *pUTS*, donde *pFeed* es el precio de la feed y *pUTS* el precio de la UTS .

```
precioUe[enrq_, cola_, pFeed_, pUTS_] :=
 pFeed "factorFeed" + pUTS "factorUTS" /.
  utsfeed[enrq, cola]
```

El precio de compra de la feed (UF6 natural en kgU) y de la SWU en el mercado spot (contratos a corto plazo) se pueden obtener del sitio web

http://www.uxc.com/review/uxc_Prices.aspx.

```
valoruranio = Flatten[Drop[Last[Import[
        "http://www.uxc.com/review/uxc_Prices.aspx",
        "Data"]][[2, 2, 1]], 3], 1];
```

```
TableForm[valoruranio]
```

| | | | |
|---|---|---|---|
| 1 US$ = | 0.72456 € | | |
| U 3 O 8 Price (lb) | $34.75 | [ -0.25 ] | €25.18 |
| NA Conv. (kgU) | $9.00 | [ Unch. ] | €6.52 |
| EU Conv. (kgU) | $9.50 | [ Unch. ] | €6.88 |
| NA UF 6 Price (kgU) | $99.75 | [ -0.50 ] | €72.27 |
| NA UF 6 Value* (kgU) | $99.80 | [ -0.65 ] | €72.31 |
| EU UF 6 Value* (kgU) | $100.30 | [ -0.65 ] | €72.67 |
| SWU Price (SWU) | $101.00 | [ Unch. ] | €73.18 |

Incluso podemos seleccionar un valor específico para su uso posterior. Aquí se extrae el precio de feed (EU UF6 value) en dólares.

```
{pFeed, pUTS} = ToExpression[
  {StringJoin[Drop[Characters[valoruranio[[7, 2]]], 1]],
StringJoin[Drop[Characters[valoruranio[[8, 2]]], 1]]}]
```

```
{100.3, 101.}
```

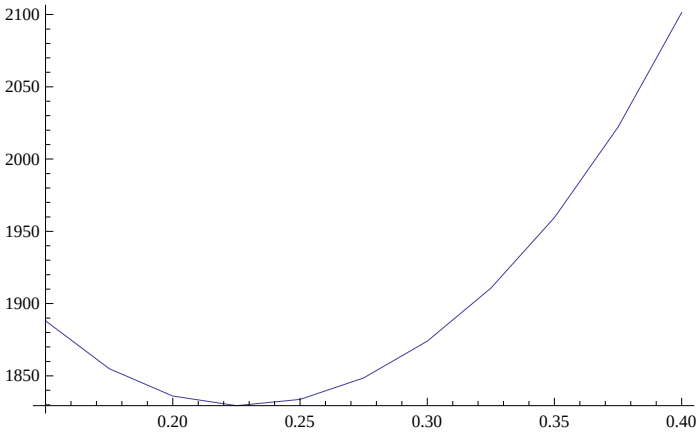
Para los precios de la feed y UTS anteriores el coste, en dólares, de 1 kg U enriquecido al 5 % y con cola 0.3 % será el siguiente:

```
precioUe[5, 0.3, pFeed, pUTS]
```

```
1874.01
```

En el siguiente gráfico dejamos como variable el valor de la cola y lo vamos modificando. Se observa que hay claramente un valor económico mínimo asociada a la cola.

```
ListPlot[Table[{i, precioUe[5, i, pFeed, pUTS]},
  {i, 0.15, 0.40, 0.025}], Joined → True]
```




```
colaoptima = Round[
  cola /. NMinimize[{precioUe[4.5, cola, pFeed, pUTS],
      0.15 ≤ cola ≤ 0.5}, cola][[2]], 0.01]
```

```
0.23
```

Ejemplo.- El coste del uranio enriquecido necesario para obtener 1 kWh.

Comment El consumo estándar de un reactor de 1000 Megawatios de potencia que funciona el 85% a potencia nominal es de 20000 kgU al año de enriquecimiento 4.5 %.

```
costeanualdolares =
 20 000 precioUe[4.5, colaoptima, pFeed, pUTS]
```

```
3.2297 × 10^7
```

```
costekwh =
 UnitConvert[Quantity[costeanualdolares, "USDollars"],
   "Euros"] / (0.85 × 1 000 000 × 24 × 365 "kWh")
```

$$\frac{€0.00323549}{\text{kWh}}$$

# $\pi$ De la antigüedad a los tiempos modernos

## » Arquímides

Supercie $S_{\text{ins}}$: descomponemos el polígono resultante en $n$ triángulos idénticos de altura $h = \text{Cos}[a/2]$ y base $b = 2\,\text{Sin}[a/2]$, donde $a$ es el ángulo $2\pi/n$. Entonces $S_{\text{ins}}$ será la suma de los triángulos que lo forman.

$S_{\text{ins}} = n \times 1/2 \times b \times h = n/2\,(2\,\text{Sin}[a/2]\,\text{Cos}[a/2]) = n/2\,\text{Sin}[a]$.

Superficie circunscrita $S_{\text{cir}}$: El area puede obtenerse por la descomposición en $n$ triángulos del polígono circunscrito. En este caso la altura de cada triángulo será 1, la misma que el radio de la circunferencia, y su base $2\,\text{Tan}[a/2]$.

Entonces el área $S_{\text{cir}} = n \times 1/2 \times 2 \times \text{Tan}[a/2] = n\,\text{Tan}[a/2]$.

La siguiente función aplica el método anterior para calcular el rango en el que se encontrará el valor de $\pi$ en función del número de lados $n$ del polígono que se considere.

```
piPolig[n_] := Module[{a}, a = 2. π / n;
  ToString[1/2 n Sin[a]] <> " ≤ π ≤ " <> ToString[n Tan[a/2]]]
```

Lo aplicamos al caso de $n = 12$.

```
piPolig[12]
```

```
3. ≤ π ≤ 3.21539
```

## » Ramanujan

Hardy le dice a Ramanujan que había tomado un taxi con un número tan anodino como el 1729. Éste le respondió que realmente era un número muy interesante pues es el más pequeño que se puede descomponer de dos maneras diferentes como suma de dos cubos.

En efecto:

```
9^3 + 10^3 == 1^3 + 12^3 == 1729
```

```
True
```

No es casual que en 1914 Ramanujan encontrase la sorprendente fórmula (que calcula 1/Pi):

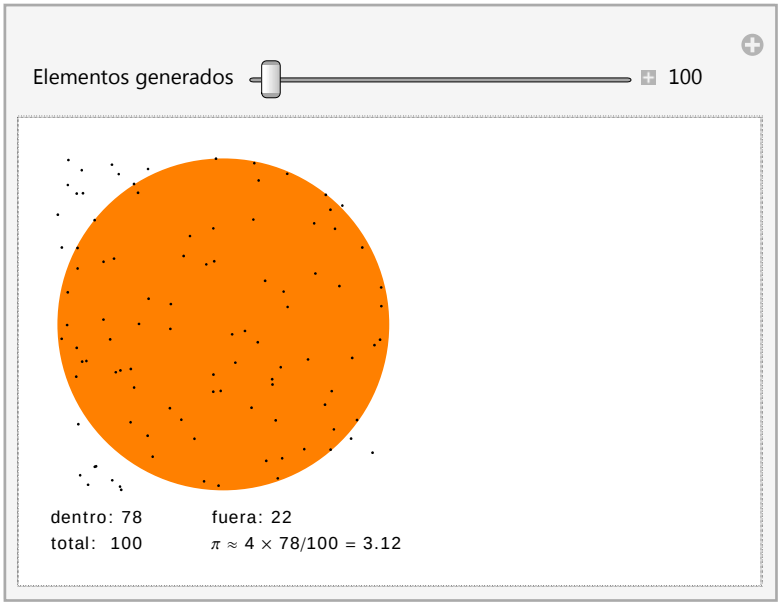
$$\text{ramapi}[n\_] := \frac{2\sqrt{2}}{9801}\sum_{j=0}^{n}\frac{(4j)!\,(26\,390\,j+1103)}{\left(4^{4j}\,j!^{4}\right)\,99^{4j}}$$

Solo para n = 3 calcula n con una precisión de mas de 15 cifras decimales

```
n[1 / ramapi[3] - Pi, 100]
```

$4.44089\times10^{-16}$

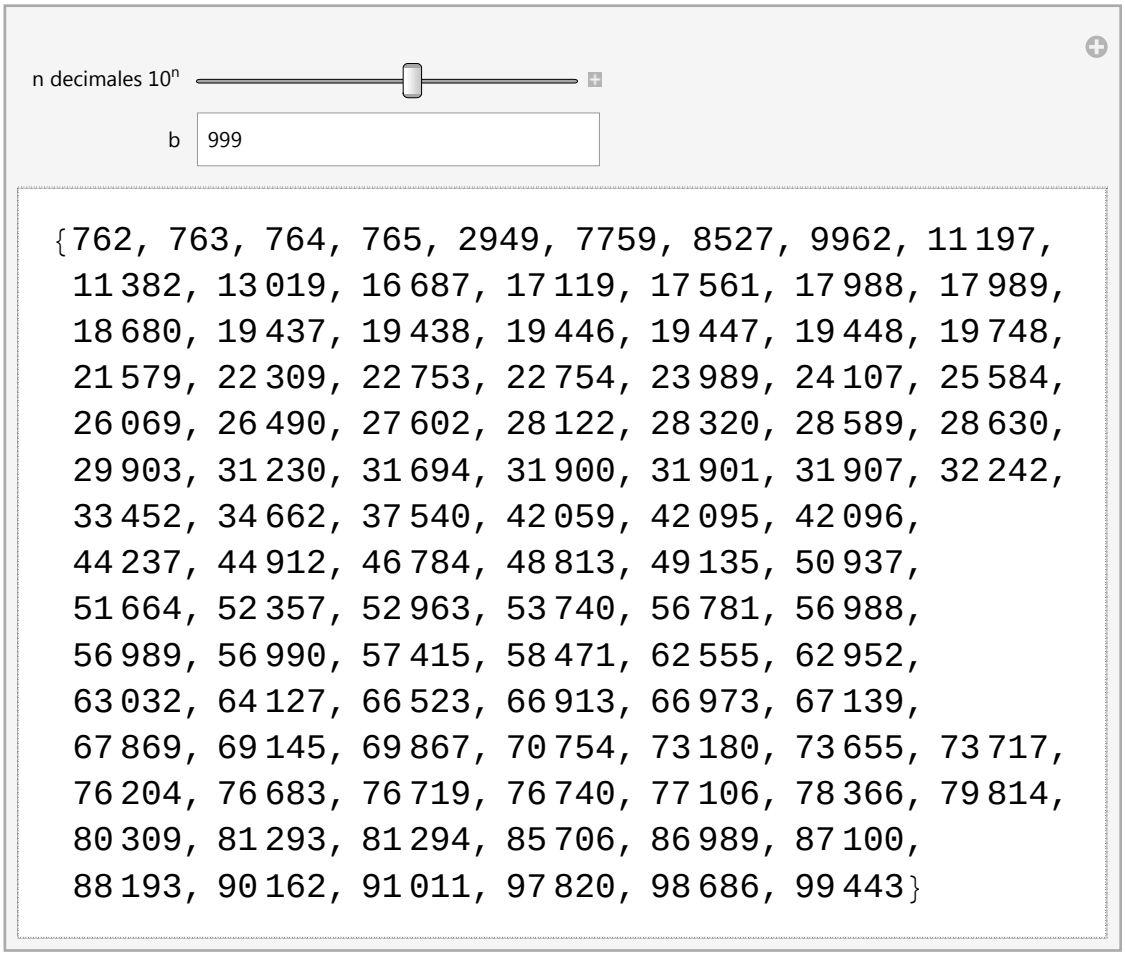
## » Métodos por simulación




## » Todo está en Pi

En el siguiente ejemplo se busca los casos (posición tras de la coma) en los que aparece una secuencia determinada (ej.: 999) en los $10^n$(ej.: n=5)

```
Manipulate[busquedaPi2[10^n, b],
 {{n, 5, "n decimales 10^n"}, 1, 8},
 {b, 999}, Initialization :→
  (busquedaPi2[n_, coincide_?IntegerQ] :=
    Module[{pi = ToString[N[Pi, n]]},
First /@ StringPosition[pi, ToString[coincide], 100] -
      2])]
```



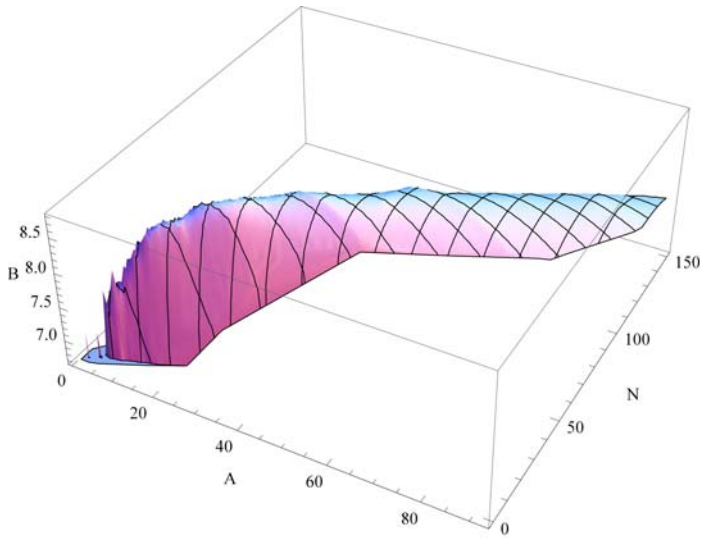
# Átomos y elementos

La siguiente expresión muestra la energía de enlace por nucleón, en función del número atómico y del número de neutrones hasta Z=92 (uranio)

```
energiaenlaceZNB = Flatten[Table[
    {z, a - z, IsotopeData[{z, a}, "BindingEnergy"]},
    {z, 1, 92}, {a, IsotopeData[#, "MassNumber"] & /@
      IsotopeData[z]}], 1];
```

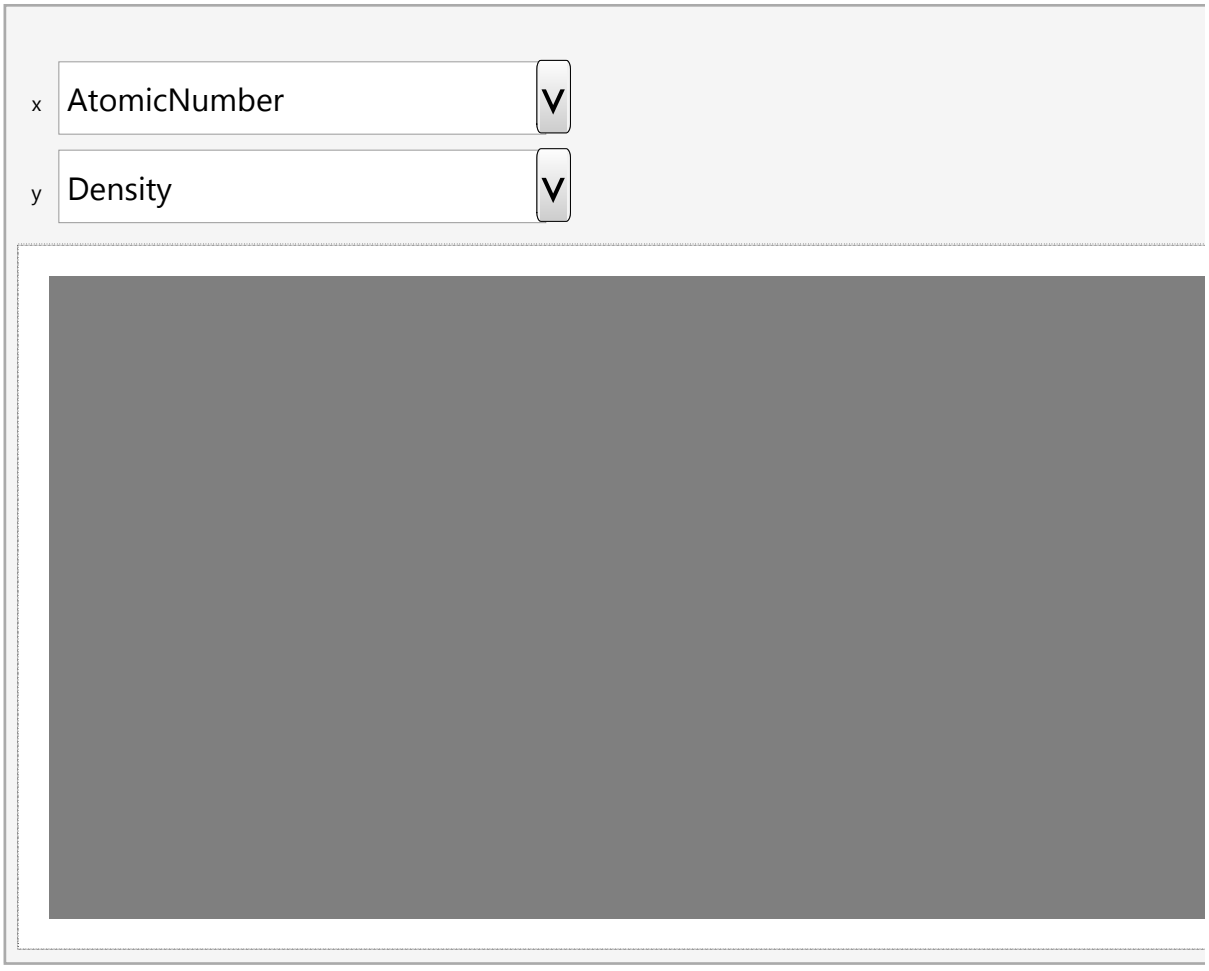
Aquí lo representamos en 3D.

```
ListPlot3D[energiaenlaceZNB, AxesLabel → {"A", "N", "B"}]
```



Explore las propiedades de los elementos

```
Manipulate[
 ListPlot[
  Table[Tooltip[{ElementData[i, x], ElementData[i, y]},
    Row[{i, " ", ElementData[i]}]],
   {i, 118}], AxesLabel → {x, y}],
 {{x, "AtomicNumber"}, ElementData["Properties"],
  ControlType → PopupMenu}, {{y, "Density"},
  ElementData["Properties"], ControlType → PopupMenu}]
```



# Modelización biocinética: Aplicación a la inyección de isótopos de iodo en el cuerpo humano

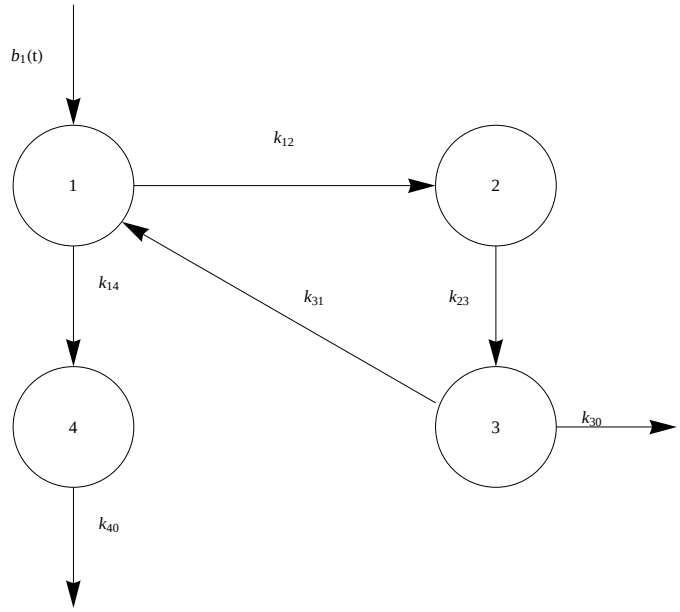
```
Clear["Global`*"]
```

Requiere instalarse la aplicación Biokmod (http://diarium.usal.es/guillermo). Utilizaremos el paquete Sysmodel (de Biokmod)

```
Needs["Biokmod`SysModel`"]
```

```
SysModel, version 1.4.1   2006-12-19
```

El modelo del iodo: (1) sangre, (2) tiroides, (3) otros organos, (4) vejiga urinaria (prescindiremos), y 0 exterior de cuerpo.




De acuerdo a la ICRP 78, en un varon estándar las tasas de trasferencia , en dias$^{-1}$, son:

```
k12 = 0.3 * (Log[2] / 0.25); k10 = 0.7 * (Log[2] / 0.25);
k23 = Log[2] / 80; k30 = 0.2 * (Log[2] / 12);
k31 = 0.8 * (Log[2] / 12);
```

Supondremos una inyección de iodo 125 en al sangre (compartimento 1) cuya constante de desintegración es:

```
λ125 = Log[2] / IsotopeData["Iodine125", "HalfLife"]
```

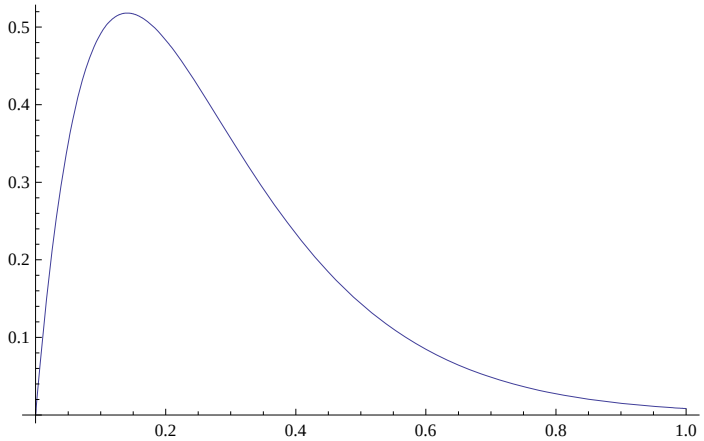
$1.35 \times 10^{-7}$

El modelo anterior podemos representarlo por la matrix compartimental siguiente (lo aplicamos al iodo 131):

```
iodine131 = CompartMatrix[3, {{1, 2, k12}, {1, 0, k10},
    {2, 3, k23} , {3, 0, k30} , {3, 1, k31}}, λ125]
{{-2.77259, 0., 0.0462098}, {0.831777, -0.00866447, 0.},
 {0., 0.00866434, -0.0577624}}
```

Supondremos como condiciones iniciales: {0, 0, 0} y el "input": $\{10.0\, t\, e^{-7.1\, t}, 0, 0\}$.

```
Plot[10.0 t e^(-7.1 t), {t, 0, 1}]
```




Observe que Biokmod construye el sistema de ecuaciones diferenciales del modelo.

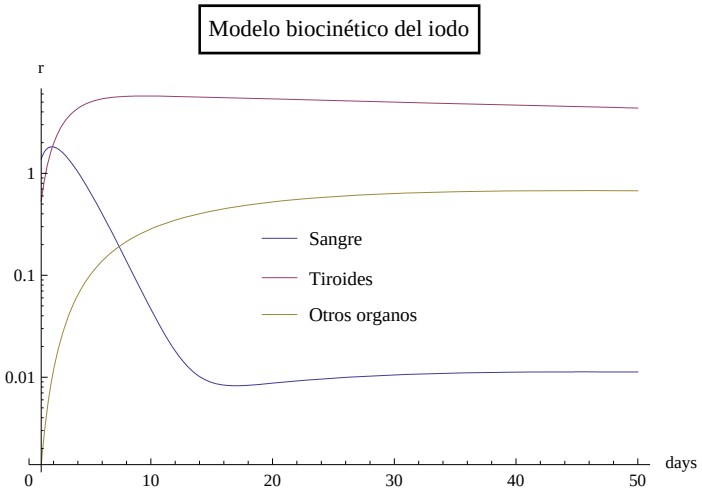
```
ShowODE[iodine131, {0, 0, 0}, {10 t e^(-7.1 t), 0, 0}, t, x]
{x1'[t] == 0. + 10 e^(-0.7 t) t - 2.77259 x1[t] + 0.0462098 x3[t],
 x2'[t] == 0. + 0.831777 x1[t] - 0.00866447 x2[t],
 x3'[t] == 0. + 0.00866434 x2[t] - 0.0577624 x3[t],
 x1[0] == 0, x2[0] == 0, x3[0] == 0}
```

Resolvemos el sistema (en este caso tiene solucion analítica, de no ser así deberiamos usar la función SystemNDSolve

```
{x1[t_], x2[t_], x3[t_]} =
 {x1[t], x2[t], x3[t]} /. SystemDSolve[iodine131,
   {0, 0, 0}, {10 t e^(-7.1 t), 0, 0}, t, t, x]
{2.32813 e^(-2.77254 t) - 2.32439 e^(-0.7 t) - 0.0205416 e^(-0.0601472 t) +
  0.016804 e^(-0.00632405 t) + 4.82663 e^(-0.7 t) t,
 -0.700638 e^(-2.77254 t) - 5.6033 e^(-0.7 t) + 0.331878 e^(-0.0601472 t) +
  5.97206 e^(-0.00632405 t) - 5.80713 e^(-0.7 t) t,
 0.00223612 e^(-2.77254 t) + 0.197578 e^(-0.7 t) - 1.20576 e^(-0.0601472 t) +
  1.00594 e^(-0.00632405 t) + 0.0783433 e^(-0.7 t) t}
```

Ahora podemos representar la evolución del contenido de iodo 131 en cada compartimento

```
LogPlot[{x1[t], x2[t], x3[t]},
 {t, 1, 50}, PlotRange → Full,
 AxesLabel → {"days", "r"}, PlotLegends -> Placed[
   {"Sangre", "Tiroides", "Otros organos"}, Center],
 PlotLabel → Style[Framed[
    "Modelo biocinético del iodo"]]]
```



En este ejemplo supondremos que el coeficiente de trasferencia k13 es el que veria .

```
Manipulate[Plot[ xTh[t, c], {t, 1, t1}, PlotRange → Full,
   AxesLabel → {"days", "r"}, PlotLegends -> "Tiroides",
   PlotLabel → Style[
     Framed["Modelo biocinético del iodo"]]],
 {t1, 2, 10}, {c, 0.0, 0.02}, Initialization :→
  (xTh[t1_, k3_] := x2[t1] /. SystemDSolve[
      CompartMatrix[3, {{1, 2, k12}, {1, 0, k10},
        {2, 3, k3} , {3, 0, k30} , {3, 1, k31}}, λ125],
      {0, 0, 0}, {10 t e^-5.9 t , 0, 0} , t, t1, x])]
```

# Compilación & Paralelización

## » Romper códigos de cifrado

```
Clear["Global`*"]
```

*Mathematica* permite la conputación paralela utilizando todos los nucleos de nuestro ordenador, incluso distribuir el cálculo entre varios ordenares que esten conectados a una red (grid).

---

Por ejemplo, aqui tenemos varios enteros de 46 digitos que trataremos de factorizar.

```
largeIntegers = {
    3 614 639 327 625 022 357 498 716 997 616 141 559 555 385 787,
    1 555 448 460 931 142 740 464 025 812 509 966 226 753 668 069,
    2 181 897 995 524 176 530 622 591 658 224 757 998 839 632 493,
    1 922 853 025 887 991 704 598 930 446 931 704 051 497 624 647,
    2 272 856 689 805 810 352 674 075 033 225 118 441 591 285 227,
    2 362 930 657 136 771 575 226 664 337 831 303 048 141 997 261,
    2 284 188 855 407 316 248 949 247 887 514 956 603 668 459 957,
    1 955 774 192 657 260 628 352 808 399 207 168 800 531 153 113
  };
```

Comprobemos el tiempo de calculo utilizando un sólo nucleo.

```
AbsoluteTiming[
 Map[FactorInteger, largeIntegers]
]
```

```
{11.789383, {{{58 492 725 308 072 529 112 393, 1},
    {61 796 391 065 508 605 932 259, 1}},
   {{38 857 083 130 167 221 288 339, 1},
    {40 029 985 156 645 721 564 071, 1}},
   {{38 237 356 848 750 206 683 279, 1},
    {57 061 946 100 374 669 843 267, 1}},
   {{38 203 009 403 916 883 124 941, 1},
    {50 332 501 441 387 630 029 667, 1}},
   {{38 857 083 130 167 221 288 339, 1},
    {58 492 725 308 072 529 112 393, 1}},
   {{38 237 356 848 750 206 683 279, 1},
    {61 796 391 065 508 605 932 259, 1}},
   {{40 029 985 156 645 721 564 071, 1},
    {57 061 946 100 374 669 843 267, 1}},
   {{38 857 083 130 167 221 288 339, 1},
    {50 332 501 441 387 630 029 667, 1}}}}
```

Lancemos todos los nucleos del ordenadar utilizando `LaunchKernels` (o abriendo el cuadro de dialogo de Parallel Kernel Status que está en el menu Evaluation).

```
LaunchKernels[]
```

```
{KernelObject[1, local], KernelObject[2, local],
 KernelObject[3, local], KernelObject[4, local]}
```

Ahora realizamos el mismo cálculo trasformandolo en un cálculo en paralelo (Parallelize).

```
AbsoluteTiming[
 Parallelize[Map[FactorInteger, largeIntegers]]
]
```

```
{6.110913, {{{58 492 725 308 072 529 112 393, 1},
    {61 796 391 065 508 605 932 259, 1}},
   {{38 857 083 130 167 221 288 339, 1},
    {40 029 985 156 645 721 564 071, 1}},
   {{38 237 356 848 750 206 683 279, 1},
    {57 061 946 100 374 669 843 267, 1}},
   {{38 203 009 403 916 883 124 941, 1},
    {50 332 501 441 387 630 029 667, 1}},
   {{38 857 083 130 167 221 288 339, 1},
    {58 492 725 308 072 529 112 393, 1}},
   {{38 237 356 848 750 206 683 279, 1},
    {61 796 391 065 508 605 932 259, 1}},
   {{40 029 985 156 645 721 564 071, 1},
    {57 061 946 100 374 669 843 267, 1}},
   {{38 857 083 130 167 221 288 339, 1},
    {50 332 501 441 387 630 029 667, 1}}}}
```

## » Acelerando los cálculos: El conjunto de Mandelbrot

Requiere que en nuestro ordenador tenga instalado un compilador de C

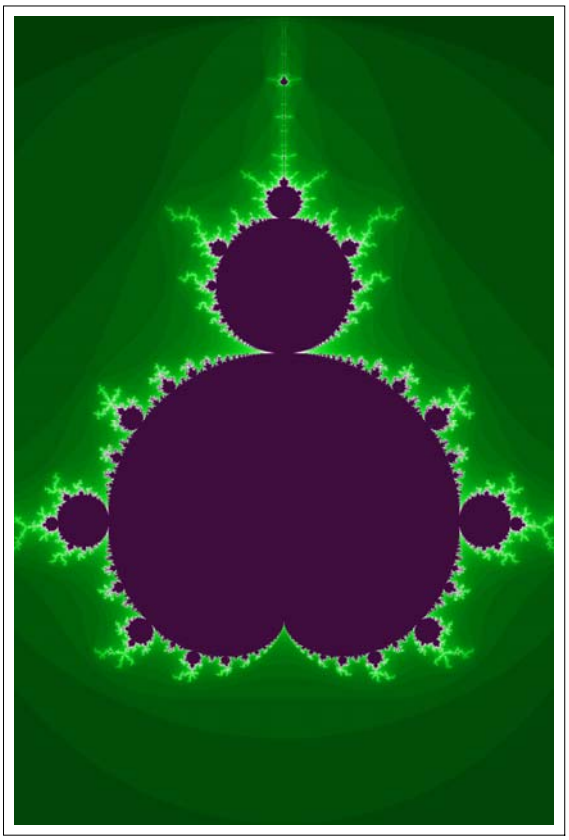
```
Needs["CCompilerDriver`"]
```

Podemos ver los que podemos utilizar

```
CCompilers[Full]
```

La función  Compile puede automaticamente generar codigo C, compilarlo y “linkarlo” dinamicamente, y ejecutarlo en paralelo (cuando es posible)

```
f = Compile[{{c, _Complex}},
    Module[{num = 1}, FixedPoint[(num++; #1^2 + c) &, 0,
      99, SameTest → (Re[#1]^2 + Im[#1]^2 ≥ 4 &)]; num],
    CompilationTarget → "C", RuntimeAttributes →
     {Listable}, Parallelization → True];
ArrayPlot[f[Table[x + ⅈ y, {x, -2, 1, 0.002},
    {y, -1, 1, 0.002}]], ColorFunction → "GreenPinkTones"]
```



# Comparando organismos genéticamente

Para calcular la separación entre especies se utiliza la distancia de Levenshtein, que corresponde al numero minimo de operaciones para convertir una secuencia en otra.

casa → caso (se sustituye 'a' por 'o')
caso → cazo (se sustituye 's' por 'z')
cazo → cazos (se inserta una 's' al final).

---

La distancia de Levenshtein puede calcularse utilizando `EditDistance`

```
EditDistance["casa", "cazos"]
```

```
3
```

En el caso anterior el tiempo de cálculo fue insignificante pero, cuando el número de caracteres a comparar aumenta, los recursos de cómputo se incrementan drásticamente. Vamos a comparar tres organismos. Descargamos su genoma.

```
picrogenes = Import["picrophilus_torridus_gene.fna"];
acidogenes =
  Import["thermoplasma_acidophilum_gene.fna"];
volcaniumgenes = Import[
   "thermoplasma_volcanium_gene.fna"];
```

Vemos el contenido (reducido) de uno de estos ficheros. Se trata del genoma de cada organismo, esto es, la relación de genes con su correspondientes secuencias de bases de ADN (A,C, G, T).

**picrogenes**

A very large output was generated. Here is a sample of it:

```
{ATGAATTCGGATAACTTTCAATCCTATTTAGGTTATTATTTAACCTGGATTT ⁞
     CCTTGGCCATATTTGCCGAGATACTATACTTTCAATCCTATTTAGGTT ⁞
     ATTATTTAACATATATCGGATTTCAAACCCTTATAGATAGACTAGACT ⁞
     TTCAATCCTATTTAGGTTATTATTTAACGCCAGAGGCATTTCTCCGTG ⁞
     CGTGGTCCTTTAATGACTTTCAATCCTATTTAGGTTATTATTTAACAC ⁞
     AACAGGCAAAAACAGAATTAAGACTGGCCATTGAGCTTTCAATCCTAT ⁞
     TTAGGTTATTATTTAACCCGTTTCTAATAAACAATATAATAAATGATT ⁞
     TTTAA, <<1596>>,
 ATGTTCACATCAATAGAT … TGCCTTAAGATGGATATAA}
```

Show Less | Show More | Show Full Output | Set Size Limit...

En este caso se trata de organismos muy simples cuyos genomas tienen el siguiente tamaño (número total de genes):

```
Length /@ {picrogenes, acidogenes, volcaniumgenes}
```

```
{1598, 1575, 1610}
```

Cargamos los datos importados para su tratamiento en paralelo.

```
With[{picrogenes = picrogenes, acidogenes = acidogenes,
   volcaniumgenes = volcaniumgenes},
  ParallelEvaluate[picro = picrogenes;
    acido = acidogenes; volcanium = volcaniumgenes;];];
```

Comparamos los primeros ocho genes del *thermoplasma volcanium* con los de *thermoplasma acidophilum* y mostramos el tiempo de cálculo y la mínima distancia gen a gen referidas al *acidophilum*.

```
acidovolcanium = ParallelMap[Function[
     {gene}, Min[EditDistance[gene, #] & /@ acido]],
    volcaniumgenes[[1 ;; 8]]]; // AbsoluteTiming
```

```
{75.505812, Null}
```

```
acidovolcanium
```

```
{304, 700, 265, 227, 497, 243, 122, 104}
```

Procedemos de la misma manera para los primeros ocho genes del *thermoplasma volcanium* y *picrophilus torridus*.

```
picrovolcanium = ParallelMap[Function[
      {gene}, Min[EditDistance[gene, #] & /@ picro]],
    volcaniumgenes[[1 ;; 8]]]; // AbsoluteTiming
```

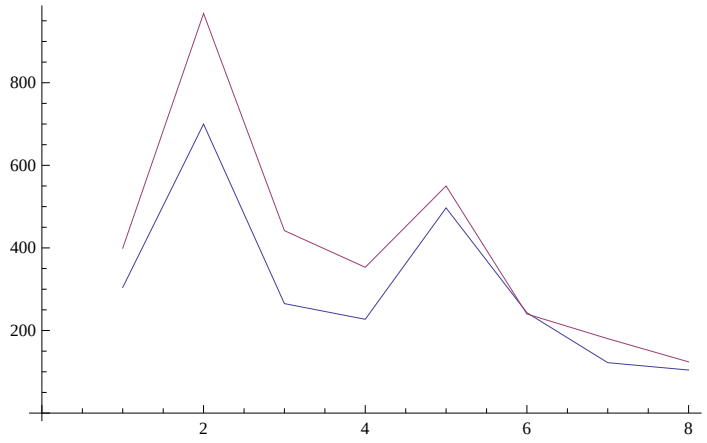
```
{76.018193, Null}
```

```
picrovolcanium
```

```
{399, 968, 442, 353, 550, 240, 180, 124}
```

Aquí mostramos gráficamente la proximidad genética de los 8 genes elegidos uno a uno del *thermoplasma volcanium* vs *thermoplasma acidophilum* y de

```
ListLinePlot[{acidovolcanium, picrovolcanium}]
```




La distancias medias entre *acidophilum volcanium* vs. *torridus* y *volcanium y thermoplasma volcanium* vs *thermoplasma acidophilum* son las siguientes:

```
{Mean[acidovolcanium], Mean[picrovolcanium]} // N
```

```
{307.75, 407.}
```

# Recursos adicionales


*Mathematica* más allá del matemáticas (http://www.addlink.es/productos/mathematica-mas-alla-de-las-matematicas-detail) por Guillermo Sánchez

En España *Mathematica*, gridMathematica, WLGM, Workbench y web*Mathematica* son distribuidos por Addlink Software Científico (http://www.addlink.es),

En http://diarium.usal.es/guillermo encontrará distintos ejemplos y vínculos a aplicaciones (algunas con webMathematica) desarrolladas por el autor.

## Celdas que se ejecutan al inicio

» General