Simulación y Optimización

Utilitarios
Parte I

Enrique E. Tarifa, Facultad de Ingeniería, UNJu

Paradigmas

Paradigmas de programación

Programación imperativa

- Detalla los pasos para resolver un problema.
- Fortran, Java, C, Python, Julia, Ruby

```
// C

int total = 0;
for(int i = 0; i < 10; i++){
   total++;
   printf("%i\n", total);
}
```

Programación declarativa

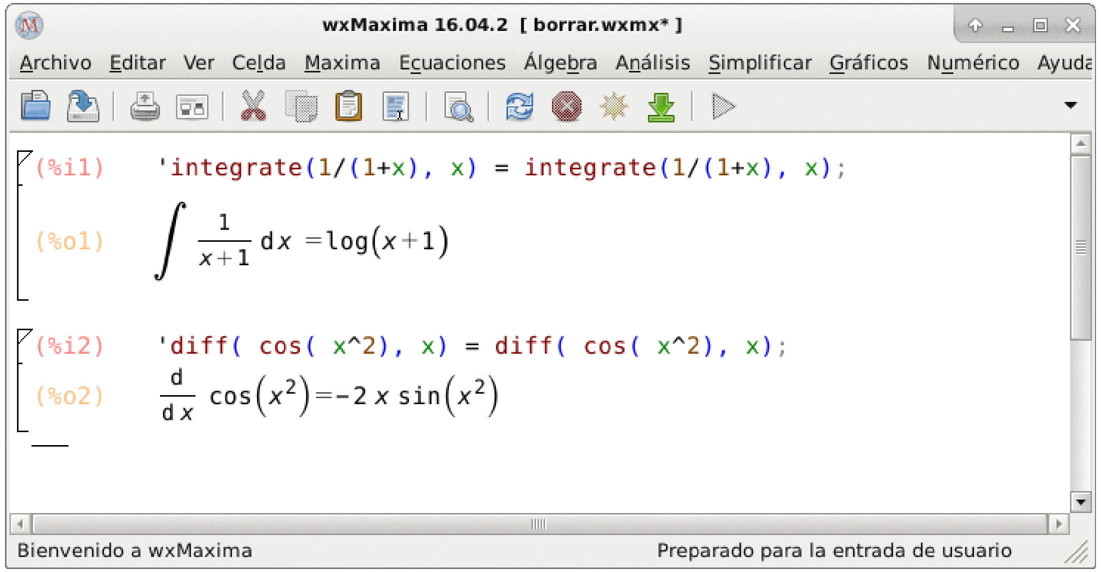
- Expresar el problema a resolver, sin dar los pasos.
- LISP, Haskell, Prolog, SQL, Elixir, XPath

```
-- Haskell

sum[1..10]
```

Aplicaciones declarativas

wxMaxima

- Cálculo simbólico
  - Integración
  - Diferenciación
  - ODEs
  - AEs
  - Transformada de Fourier
- Solución analítica



wxMaxima 22.04.0 (Windows 10 (compilación 18363), edición de 64 bits) [no guardado*]

Archivo Editar Vista Celda 'Maxima' Ecuaciones (Q) Matriz Análisis Simplificar Listado Tramado Numérico Ayuda

1 Resolución de ODEs

(%i12)
```
eqn_1: 'diff(f(x),x)='diff(g(x),x)+sin(x);
eqn_2: 'diff(g(x),x,2)='diff(f(x),x)-cos(x);
```

(eqn_1) $$\frac{d}{d\,x}\,f(x)=\frac{d}{d\,x}\,g(x)+\sin(x)$$

(eqn_2) $$\frac{d^2}{d\,x^2}\,g(x)=\frac{d}{d\,x}\,f(x)-\cos(x)$$

(%i14)
```
atvalue('diff(g(x),x),x=0,a)$
atvalue(f(x),x=0,1)$
```

(%i15)
```
desolve([eqn_1, eqn_2], [f(x),g(x)]);
```

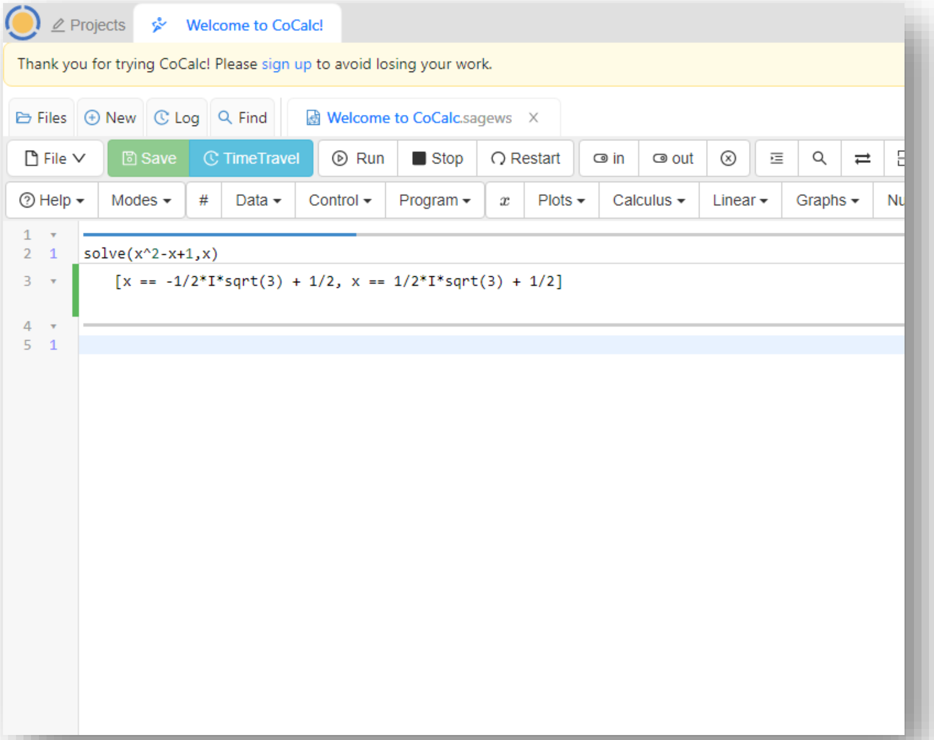
(%o15) $$\left[f(x)=a\ \%e^{x}-a+1,g(x)=\cos(x)+a\ \%e^{x}-a+g(0)-1\right]$$

'Maxima' preparada para la entrada.

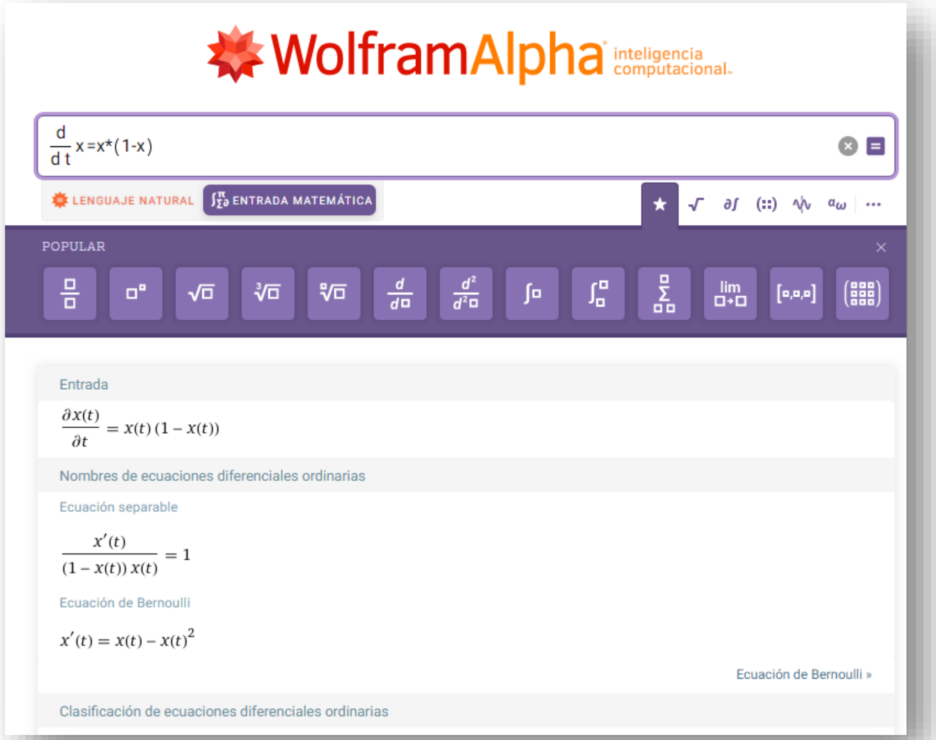
Preparado para la entrada de usuario

Otros CAS (computer algebra system)

SageMath



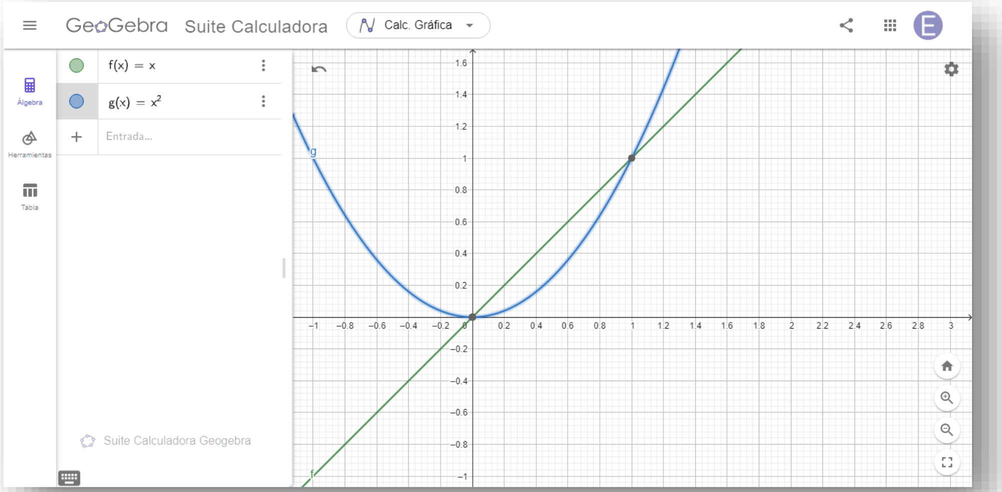
WolframAlpha

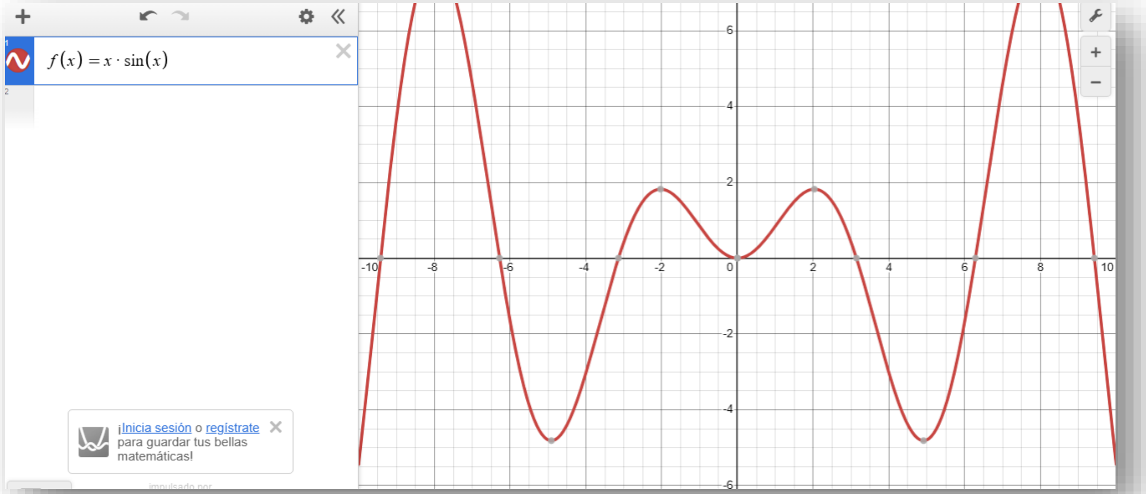


Graficadores

[Geogebra]()

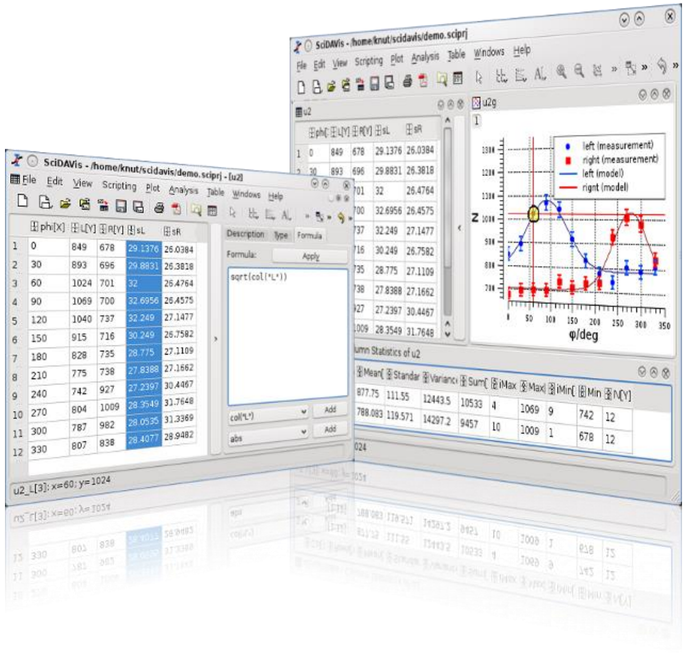
[Desmos]()



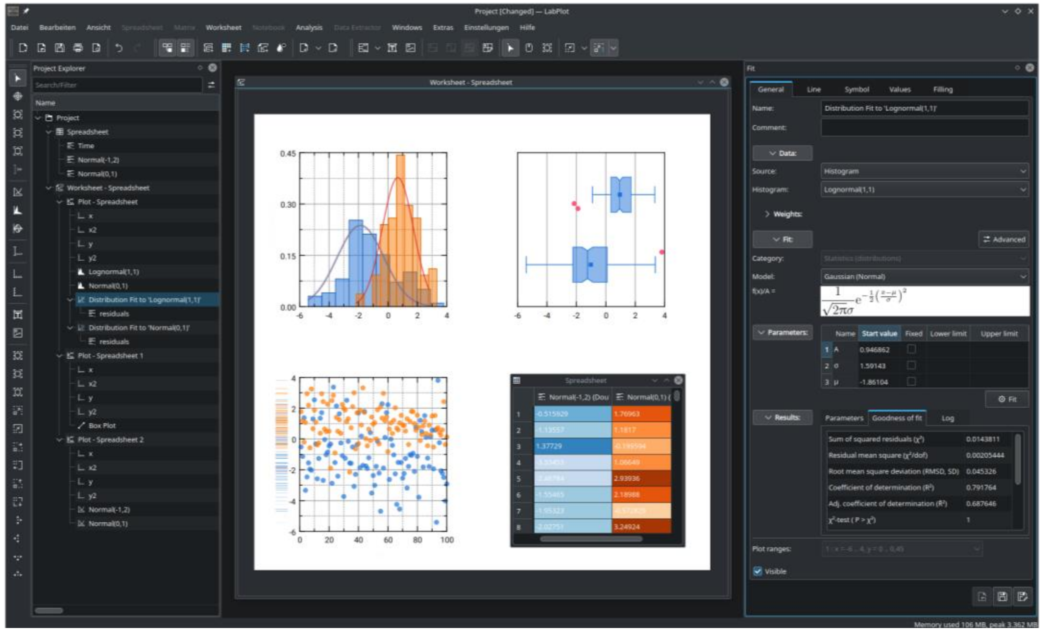


Graficadores

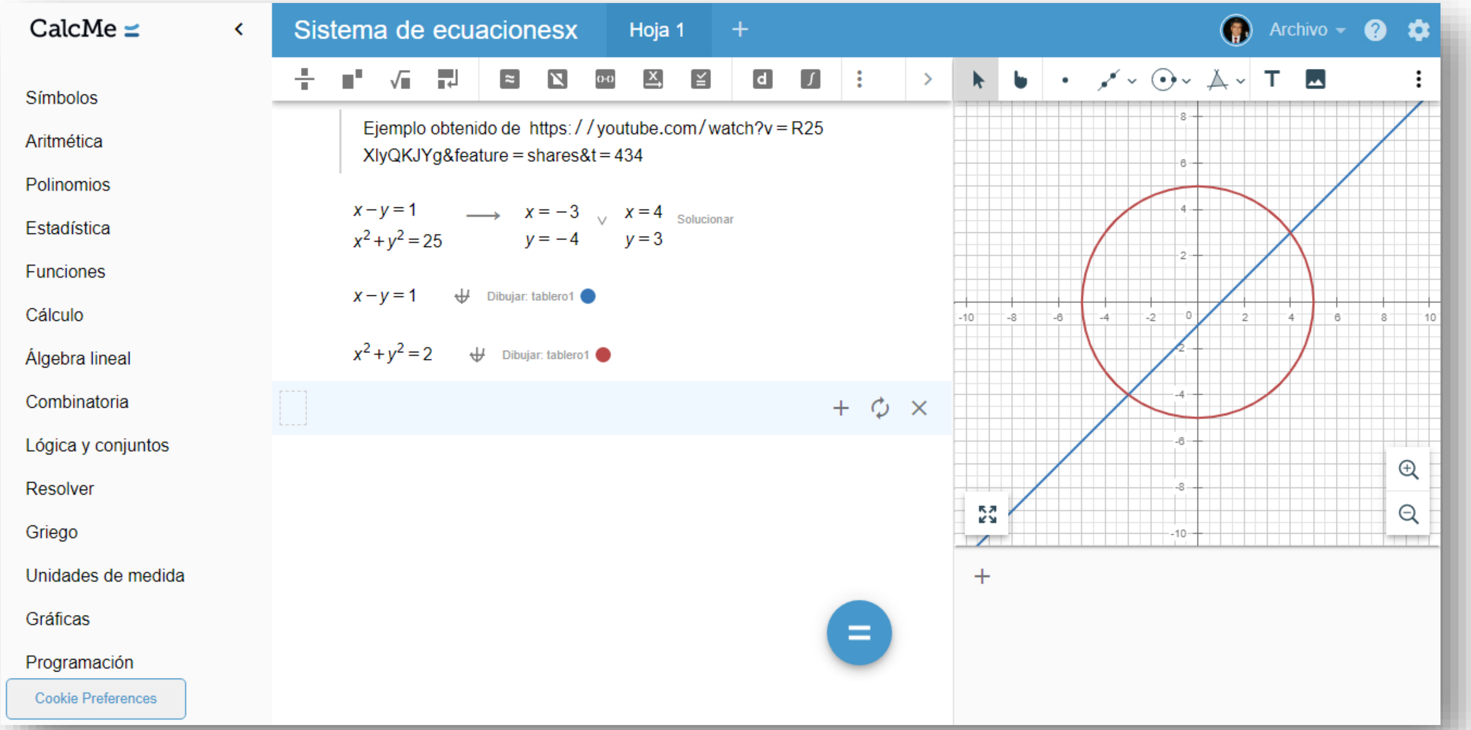
[SciDavis](#)



[LabPlot](#)



CalcMe

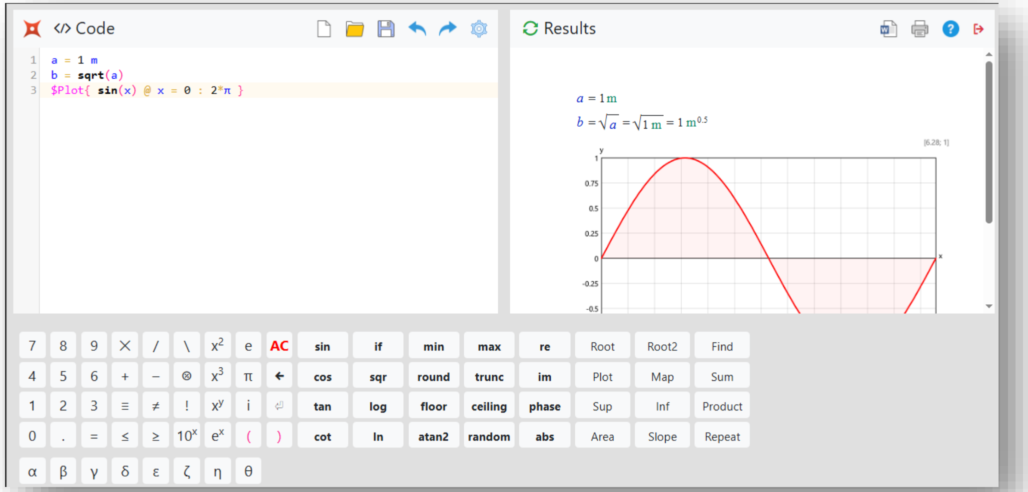


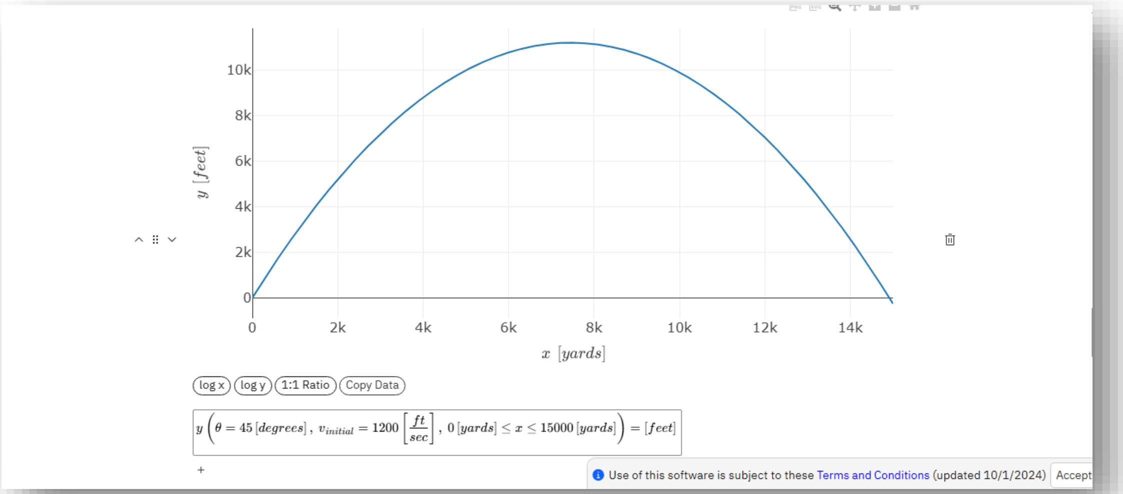
[CalcMe]()

Calculadoras

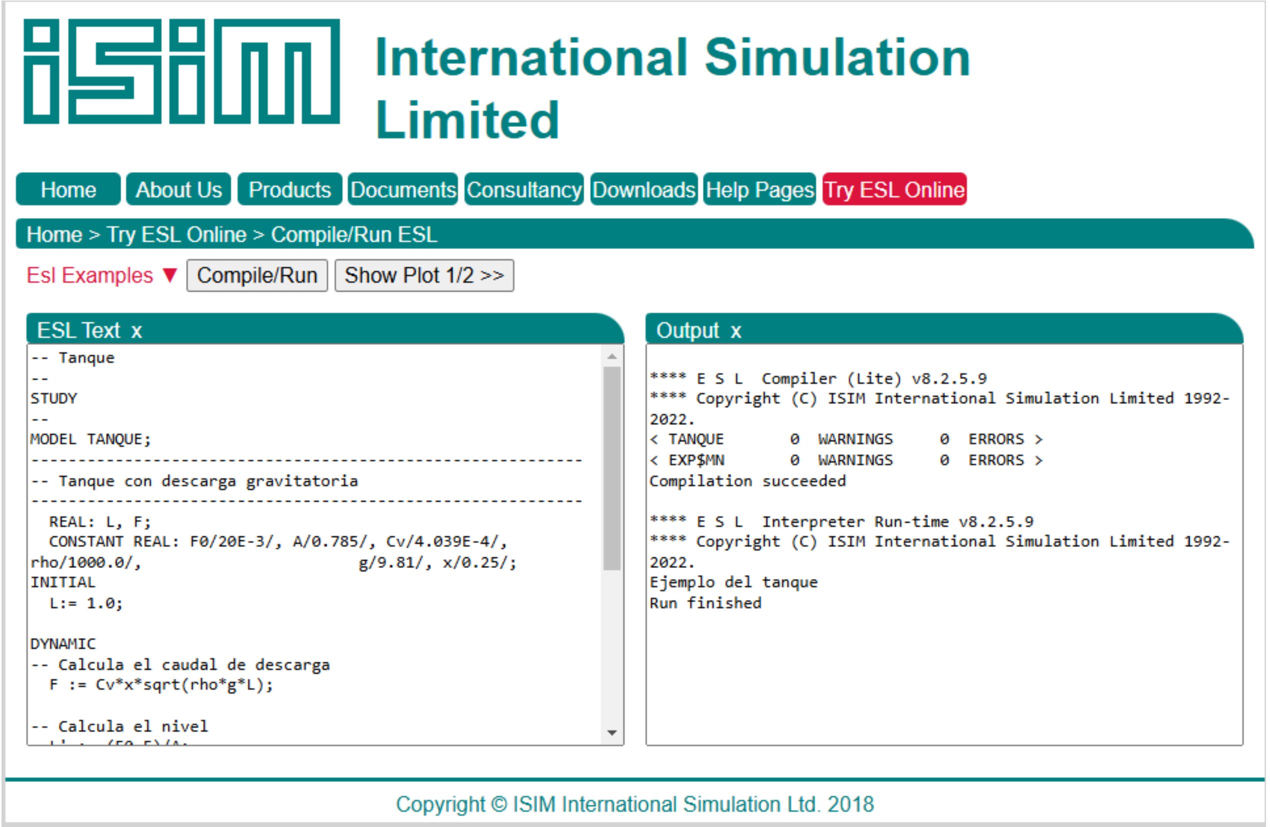
CalcPad

EngineeringPaper.xyz



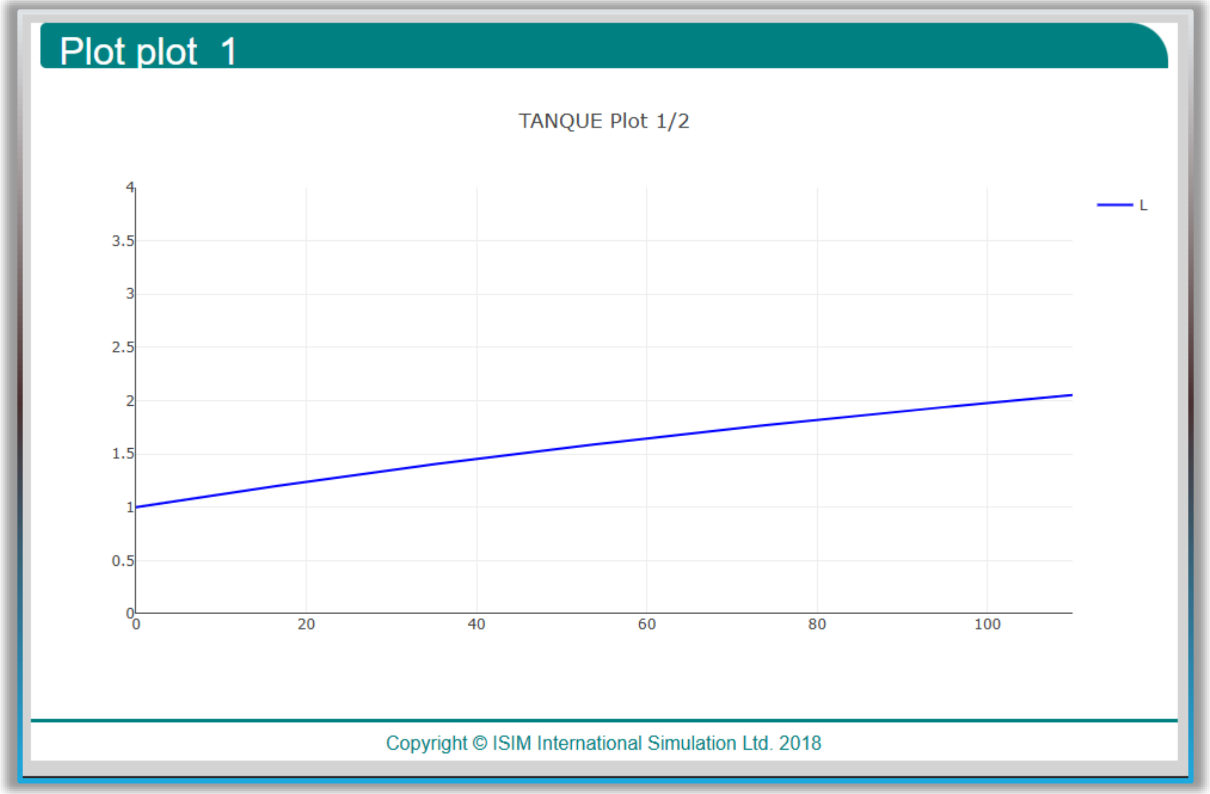


ISIM ESL



ISIM ESL

ISIM ESL




```
-- Tanque
--
STUDY
--
MODEL TANQUE;
------------------------------------------------------------
-- Tanque con descarga gravitatoria
------------------------------------------------------------
  REAL: L, F;
  CONSTANT REAL: F0/20E-3/, A/0.785/, Cv/4.039E-4/,
                 rho/1000.0/, g/9.81/, x/0.25/;

INITIAL
  L := 1.0;

DYNAMIC
-- Calcula el caudal de descarga
  F := Cv*x*sqrt(rho*g*L);

-- Calcula el nivel
  L' := (F0-F)/A;

STEP
  PLOT "TANQUE Plot 1/2", t, L, 0, TFIN,0,4;
  PLOT "TANQUE Plot 2/2", t, F, 0, TFIN,0,2*F0;
END TANQUE;
------------------------------------------------------------
--EXPERIMENT
  PRINT "Ejemplo del tanque";

  TFIN := 110.0;
  CINT := 1;
  TANQUE;

END_STUDY
```

PolymathPlus

- Aplicación online
- Resolución de varios tipos de problemas independientes:
  - ODEs
  - AEs
    - Lineales
    - No lineales
  - Regresión

```
# Tanque con descarga gravitatoria

# Inicialización
t|0:110
L|1

# ODEs
L' = (F0-F)/A

# AEs
F = Cv*x*sqrt(rho*g*L)

# Parámetros
F0 = 20E-3
A = 0.785
Cv = 4.039E-4
rho = 1000
g = 9.81
x = 0.5
```

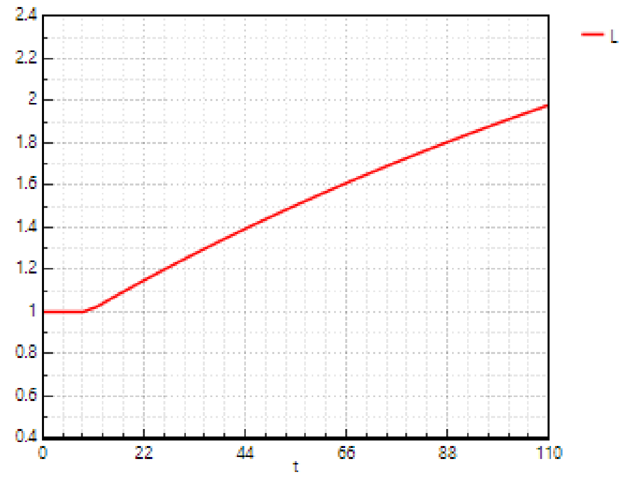
PolymathPlus 7.0.32

File Edit View Tools Examples Solve

Report Settings Help

```
# Inicialización
t|0:110
L|1

# ODEs
L' = (F0-F)/A

# AEs
F = Cv*x*sqrt(rho*g*L)

# Parámetros
F0 = 20E-3
A = 0.785
Cv = 4.039E-4
rho = 1000
g = 9.81
x = if t < 10 then 0.5 else 0.25
```

Integration chart



Formatted equations

$$\frac{d(L)}{d(t)} = \frac{F_0 - F}{A}$$

$F_0 = 20E-3$

$A = 0.785$

$Cv = 4.039E-4$

$\rho = 1000$

$g = 9.81$

$F = Cv \cdot x \cdot \sqrt{\rho \cdot g \cdot L}$

Differential equations

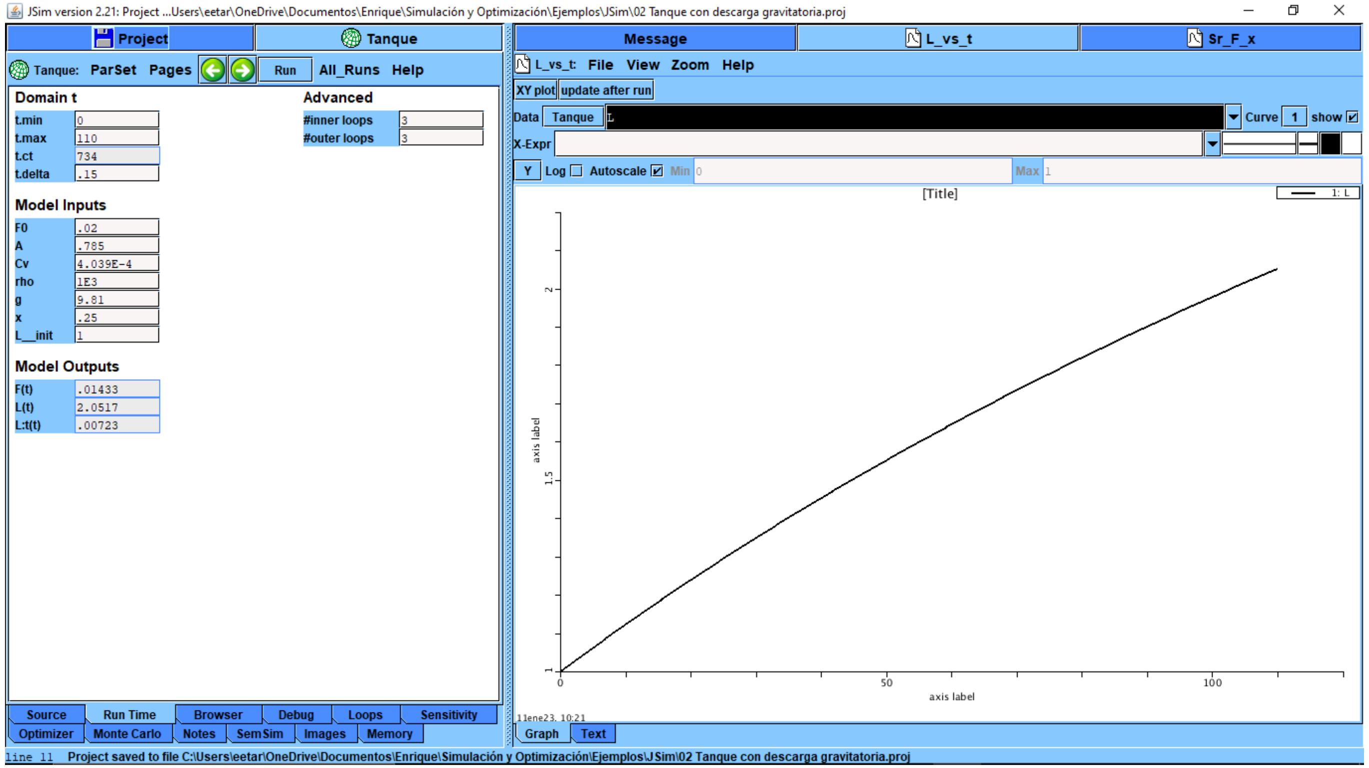
1 d(L)/d(t) = (F0-F)/A

Explicit equations

Tanque con descarga gravitatoria.pol

ENRIQUE Auto Save Ln: 12 Col: 19

JSim

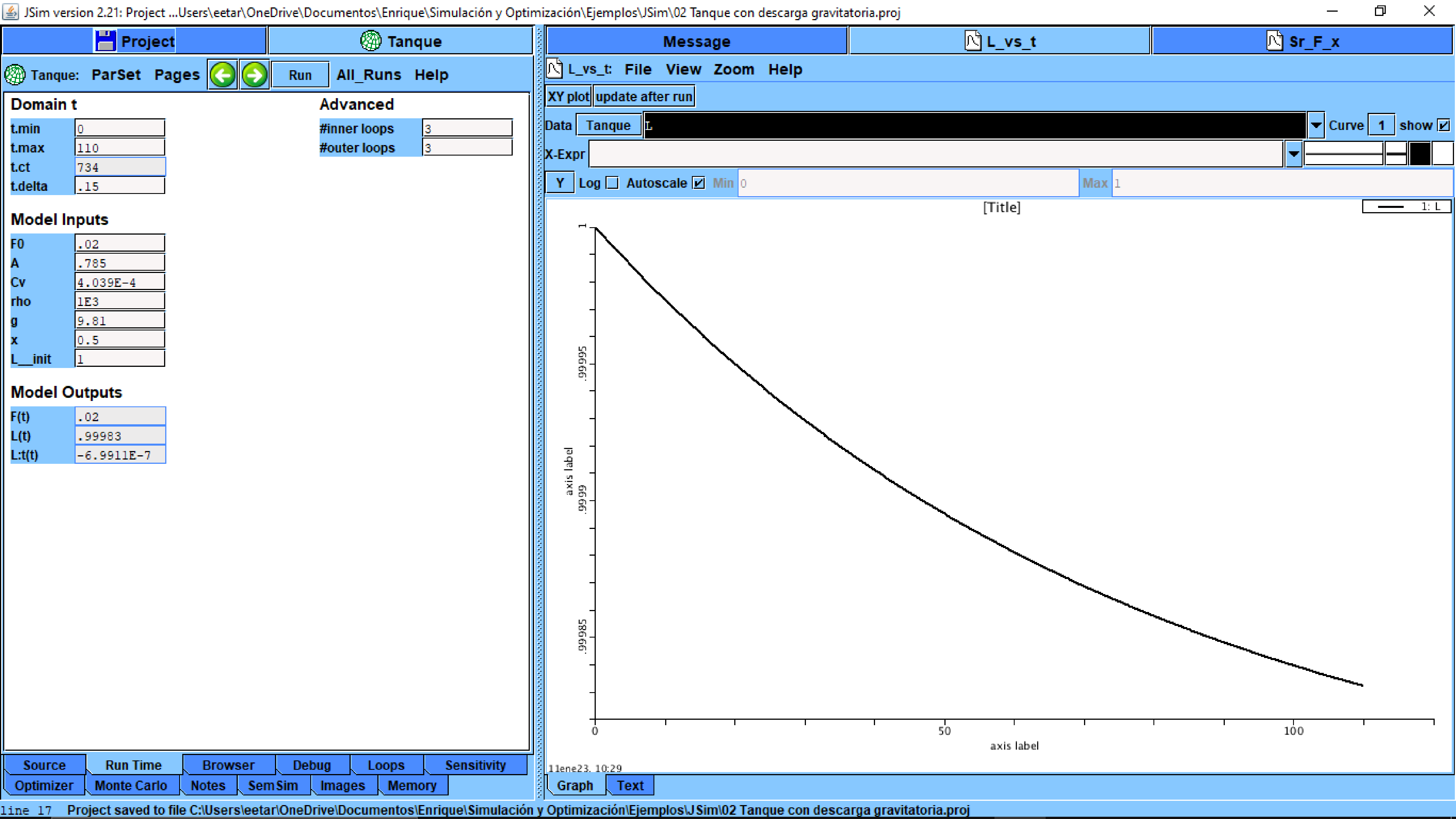


JSim

JSim

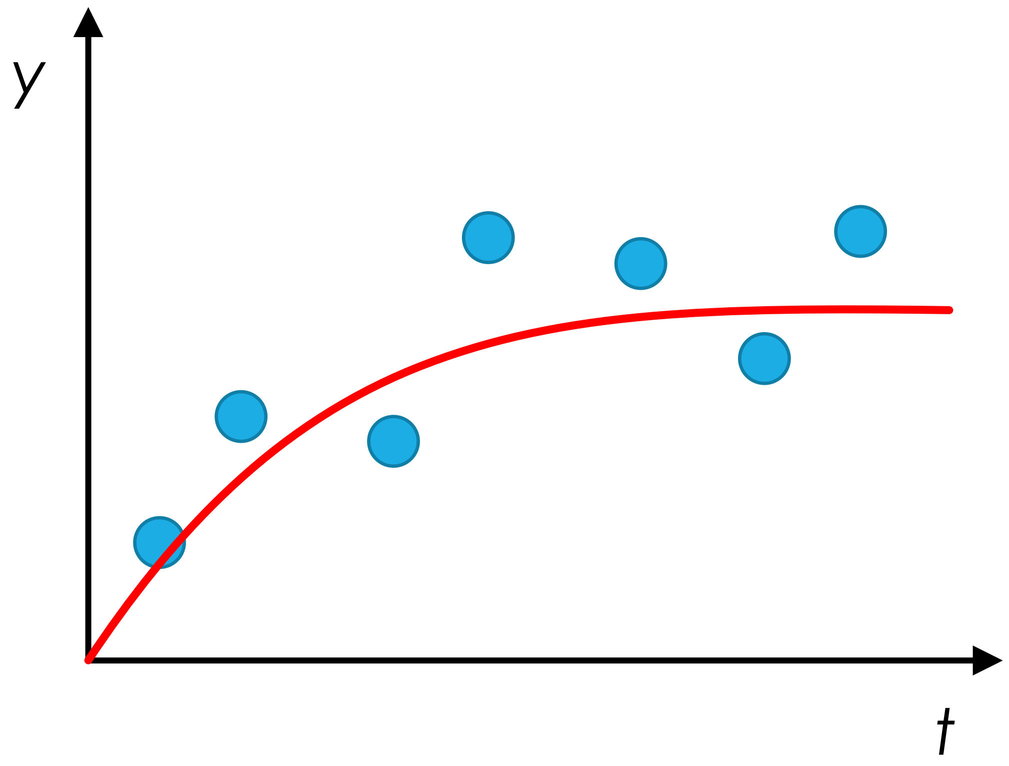
- Declarativo
- Unidades físicas
- Límites de variables
- Vectores
- Eventos
- Resolución sistemas AEs
- Integración de ODEs
- Integración de PDEs
- Optimización
- Simulación de Monte Carlo

```
// Tanque con descarga gravitatoria

math tanque {

  // Parámetros de simulación
  realDomain t; // tiempo
  t.min   = 0;
  t.max   = 110;
  t.delta = 0.15;

  // Parámetros
  real F0 = 20E-3, A = 0.785, Cv = 4.039E-4,
       rho = 1000, g = 9.81, x = 0.5;

  // Variables
  real F(t);
  real L(t); // Variable de estado

  // Condiciones iniciales
  when(t=t.min) {
    L = 1;
  }

  // ODEs
  L:t = (F0-F)/A;

  // AEs
  F - Cv*x*sqrt(rho*g*L) = 0;
}
```


JSim version 2.21: Project ...Users\eetar\OneDrive\Documentos\Enrique\Simulación y Optimización\Ejemplos\JSim\02 Tanque con descarga gravitatoria.proj
Project
Tanque
Tanque: ParSet Pages Run All_Runs Help
Domain t
t.min 0
t.max 110
t.ct 734
t.delta .15
Advanced
#inner loops 3
#outer loops 3
Model Inputs
F0 .02
A .785
Cv 4.039E-4
rho 1E3
g 9.81
x 0.5
L__init 1
Model Outputs
F(t) .02
L(t) .99983
L:t(t) -6.9911E-7
Message
L_vs_t
Sr_F_x
L_vs_t: File View Zoom Help
XY plot update after run
Data Tanque L
Curve 1 show
X-Expr
Y Log Autoscale Min 0 Max 1
[Title]
1: L
1
.99995
.9999
.99985
axis label
0
50
100
axis label
11ene23, 10:29
Graph Text
Source Run Time Browser Debug Loops Sensitivity
Optimizer Monte Carlo Notes SemSim Images Memory
line 17 Project saved to file C:\Users\eetar\OneDrive\Documentos\Enrique\Simulación y Optimización\Ejemplos\JSim\02 Tanque con descarga gravitatoria.proj

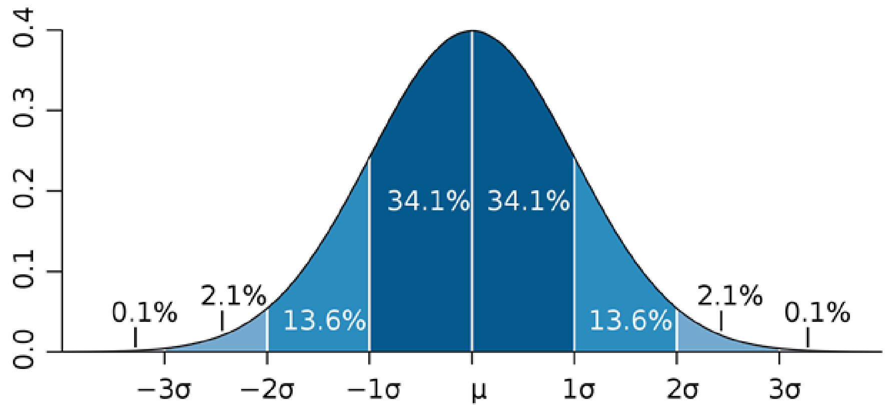
JSim

Regresión

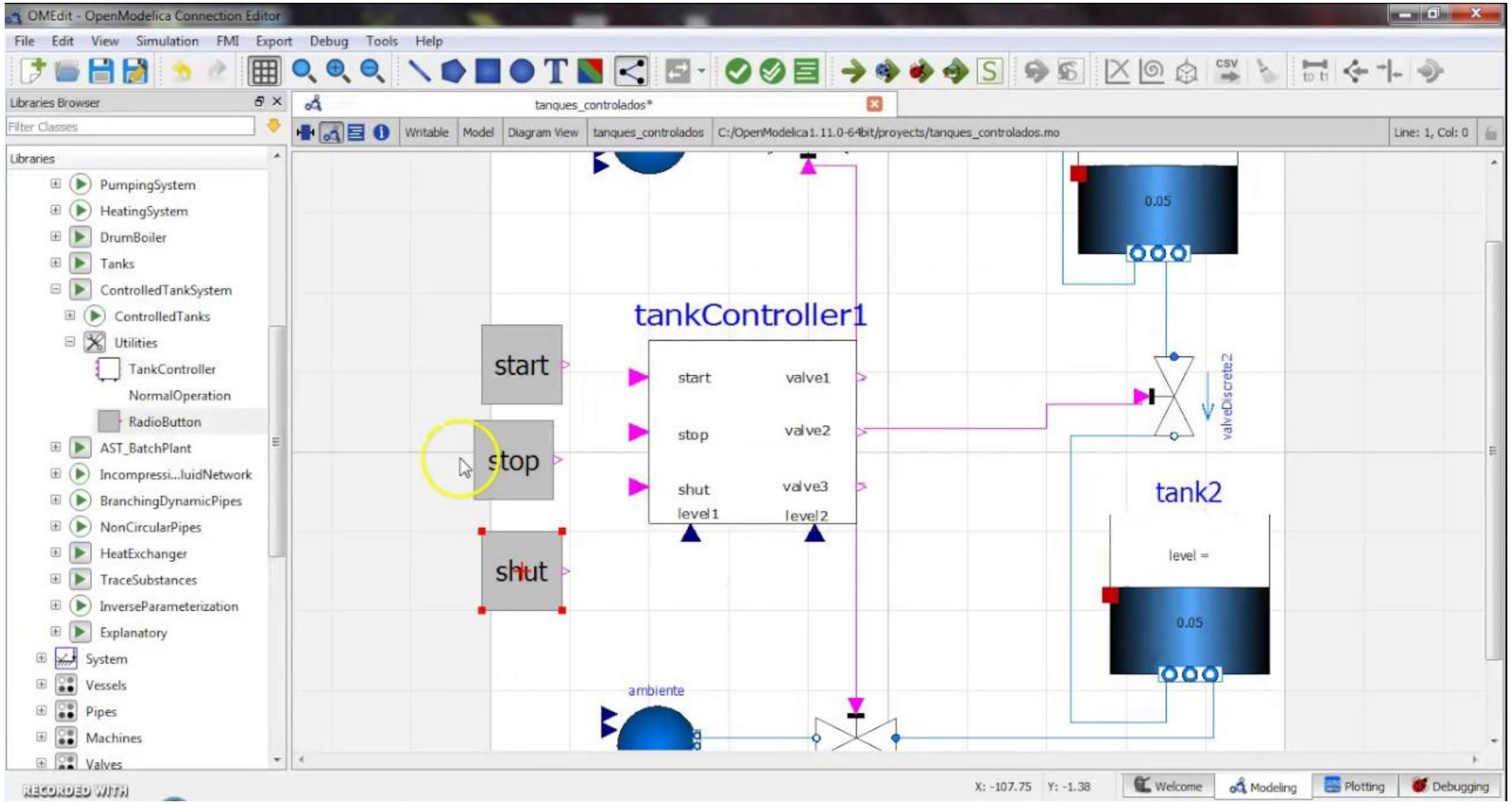


Simulación de Monte Carlo

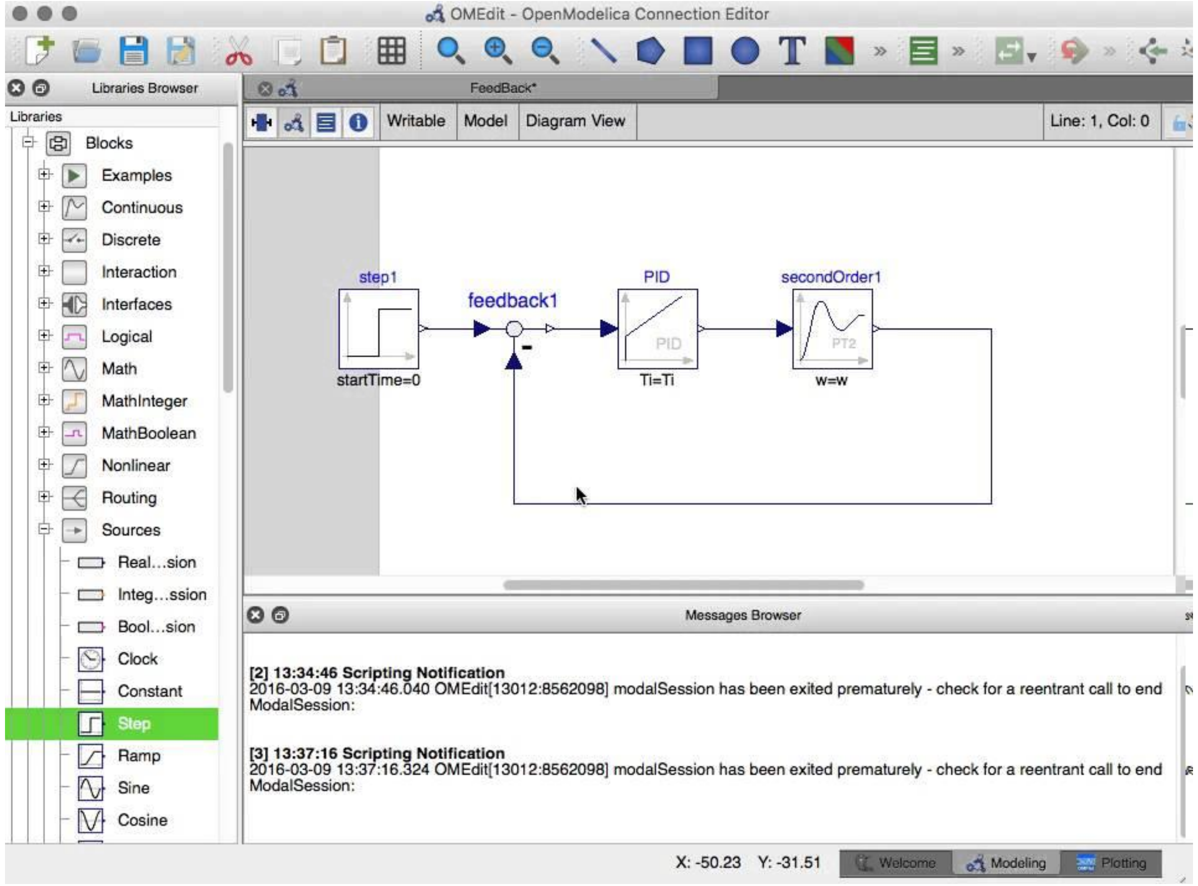
- Modelar ruidos de datos.
- Realizar la simulación de Monte Carlo.
- Determinar intervalos de confianza de los parámetros.



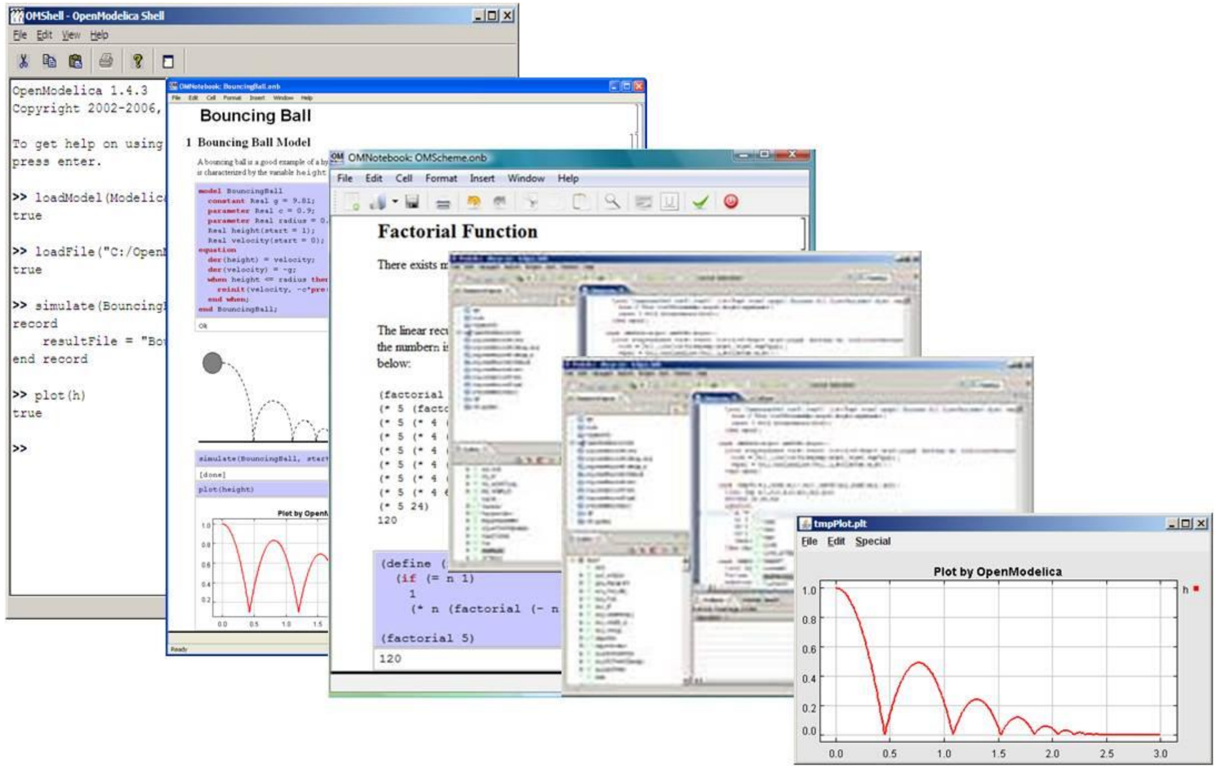
Open Modelica



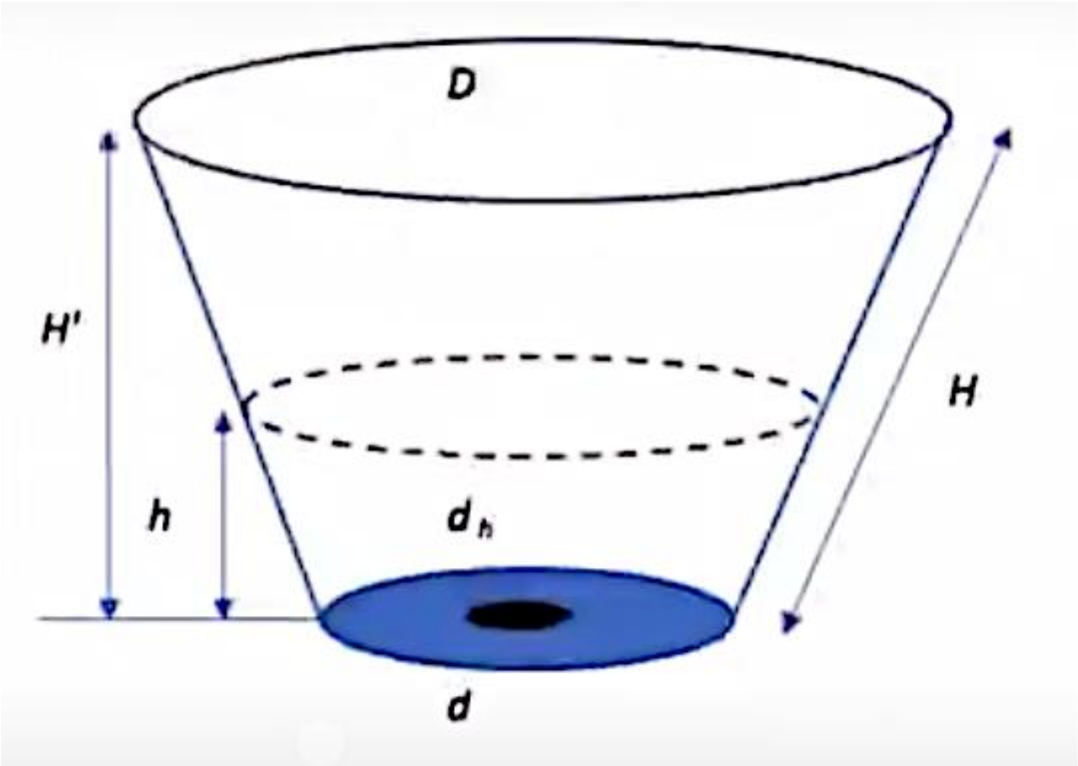
Open Modelica



Open Modelica

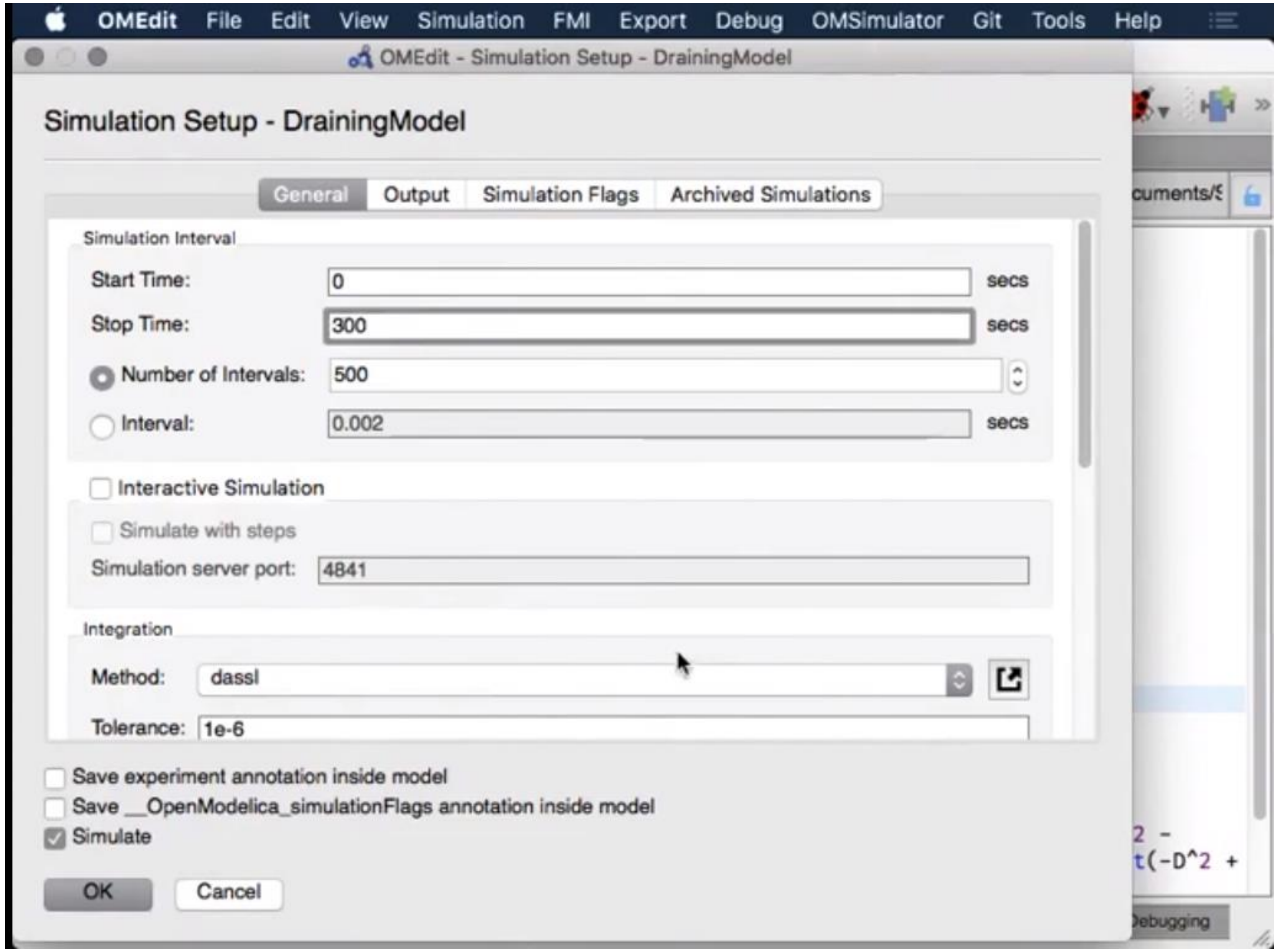


Tanque cónico

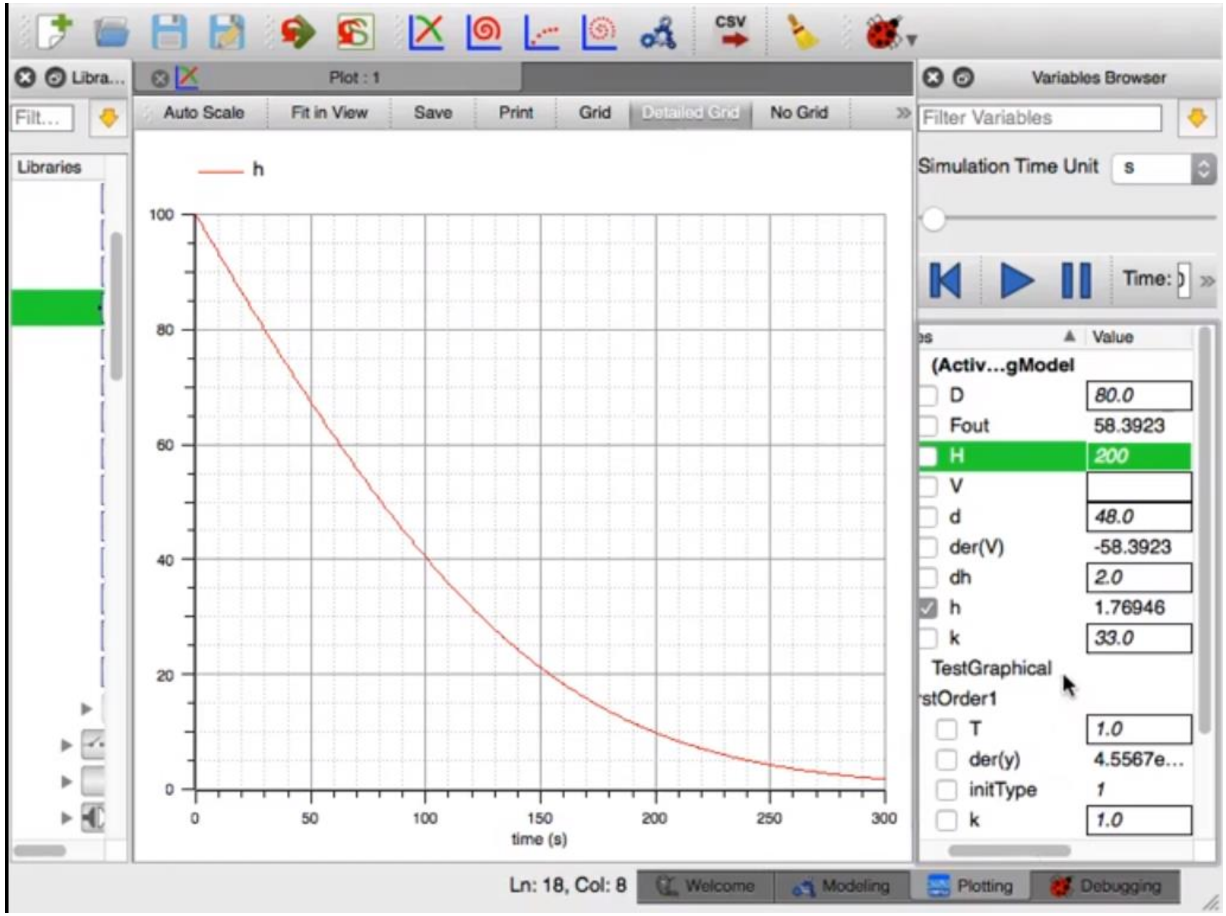


Tanque_conico.docx

Open Modelica



Open Modelica



Sistema mezclador

Mixing system

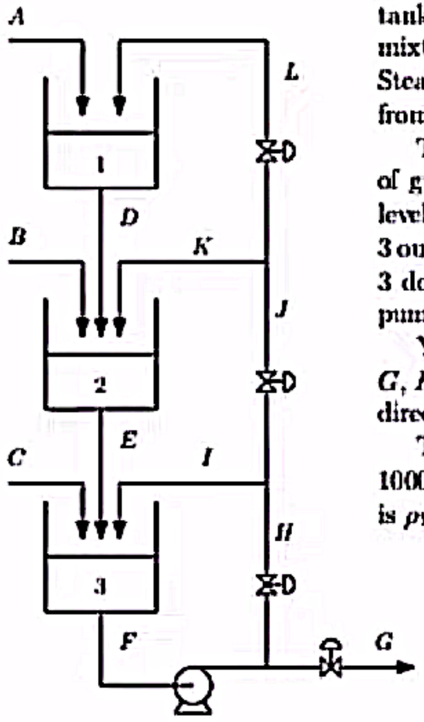


Figure 1: Mixing system

Figure 1 shows a set of well-mixed mixing tanks. All the streams contain a binary mixture of substance X and substance Y. Steams A, B and C are fed into the system from an upstream process.

Tanks 1 and 2 are drained by the force of gravity (assume flow is proportional to level), while the pump attached to the tank 3 output is sized such that the level in tank 3 does not affect the flowrate through the pump.

You may assume that the valves in lines G, H, J and L can manipulate those flows directly.

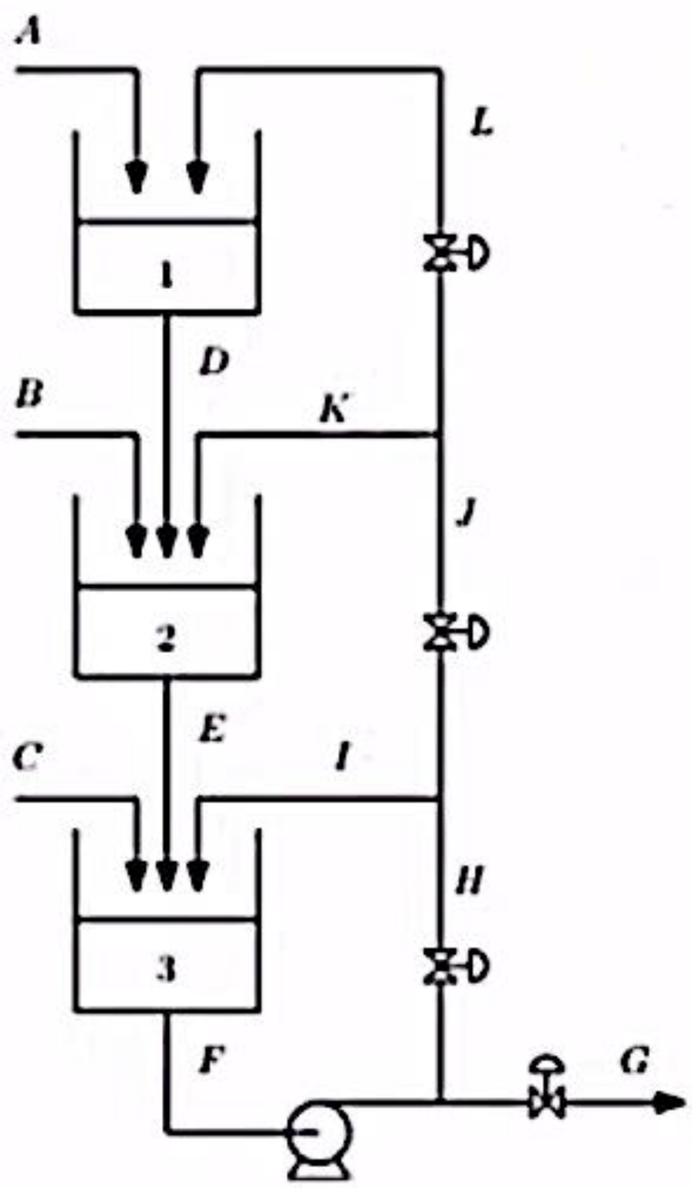
The density of substance X is $\rho_X = 1000\,\mathrm{kg/m^3}$ and the density of substance Y is $\rho_Y = 800\,\mathrm{kg/m^3}$.

Sistema Mezclador

- Tanques bien agitados.
- Dos componentes: X e Y.
- Los caudales de descarga de los tanque 1 y 2 son proporcionales a los correspondientes niveles.
- Debido a la bomba de descarga, el caudal de descarga del tanque 3 es independiente del nivel.
- Las válvulas instaladas pueden manipular directamente los caudales G, H, J y L.
- La densidad de la substancia X es 1000 kg/m^3 y la densidad de la substancia Y es 800 kg/m^3.

Sistema Mezclador

- Sistema isotérmico
- *F* es independiente de V_3.
- Componentes: 2
- Balances dinámicos:
  - 1 global por tanque
  - 1 de componente por tanque
- Balances seudoestacionarios:
  - 1 global por divisor
  - 1 de componente por divisor



[Análisis de grados de libertad]()

Tut3.mo

Users > alchemyst > Documents > Development > Dynamics-and-Control-admin > cpn321 > 2019_T3

```
model Tut3

// Flow rates, mass / time
Real A, B, C, D, E, F, G, H, I, J, K, L;
// Mass fraction X in each stream
Real xA, xB, xC, xD, xE, xF, xG, xH, xI, xJ, xK, xL;
// For each tank:
Real M1, M2, M3; // Mass
Real x1, x2, x3; // Mass fraction x
Real h1, h2, h3; // Height
Real V1, V2, V3; // Volume
parameter Real A1=3, A2=3, A3=3; // Cross-sectional area

parameter Real rhox=1000, rhoy=800; // Density of x and Y

// Setting fixed=false causes these values to be calculated at init
// Discharge coefficients
parameter Real k1(fixed=false), k2(fixed=false);
// Flowrates of the flows with valves
parameter Real Gstart(fixed=false), Hstart(fixed=false), Jstart(fix

initial equation

```

master* 0 0 18 mins Ln 9, Col 36 Spaces: 4 UTF-8 LF Modelica

Tut3.mo ×

Users > alchemyst > Documents > Development > Dynamics-and-Control-admin > cpn321 > 2019_T3

```
// Discharge coefficients
parameter Real k1(fixed=false), k2(fixed=false);
// Flowrates of the flows with valves
parameter Real Gstart(fixed=false), Hstart(fixed=false), Jstart(fix

initial equation

h1 = 1;
h2 = 1;
h3 = 1;

H = G;
H = 2*J;
J = 2*L;

der(M1) = 0;
der(M2) = 0;
der(M3) = 0;

der(x1*M1) = 0;
der(x2*M2) = 0;
der(x3*M3) = 0;

equation
```

master* ⊗ 0 ⚠ 0 Spaces: 4 UTF-8 LF Modelica

Tut3.mo

Users > alchemyst > Documents > Development > Dynamics-and-Control-admin > cpn321 > 2019_T3

```
equation

// Flowrates in
A = (if time <= 0 then 1 * rhox else 1.5 * rhox);
B = 1 * rhoy;
C = 1 * rhoy;

// Fractions in
xA = 1;
xB = 0;
xC = 0;

// Flowrates with valves
// Right now fixed at starting point, but could be changed here
G = Gstart; // For instance try this: G = A + B + C
H = Hstart;
J = Jstart;
L = Lstart;

// MB over tanks
der(M1) = A + L - D "MB Tank 1";
der(M2) = B + D + K - E "MB Tank 2";
der(M3) = C + E + I - F "MB Tank 3";
```

master* ⊗ 0 ⚠ 0 23 mins Ln 38, Col 15 Spaces: 4 UTF-8 LF Modelica

Tut3.mo

Users > alchemyst > Documents > Development > Dynamics-and-Control-admin > cpn321 > 2019_T3

```
L = Lstart;

// MB over tanks
der(M1) = A + L - D "MB Tank 1";
der(M2) = B + D + K - E "MB Tank 2";
der(M3) = C + E + I - F "MB Tank 3";

// X balance over tanks
der(x1*M1) = xA*A + xL*L - xD*D "CB Tank 1";
der(x2*M2) = xB*B + xD*D + xK*K - xE*E "CB Tank 2";
der(x3*M3) = xC*C + xE*E + xI*I - xF*F "CB Tank 3";

//Junctions
F = H + G;
H = I + J;
J = K + L;

// Fractions are the same for all the junctions
xF = xG;
xF = xH;
xF = xI;
xF = xJ;
xF = xK;
```

master* 0 0 23 mins Ln 38, Col 15 Spaces: 4 UTF-8 LF Modelica

Tut3.mo

Users > alchemyst > Documents > Development > Dynamics-and-Control-admin > cpn321 > 2019_T3

```
xF = xL;

// Fractions coming out of tanks
xD = x1;
xE = x2;
xF = x3;

// Discharge
D = k1 * h1;
E = k2 * h2;

// Geometry
V1 = A1*h1;
V2 = A2*h2;
V3 = A3*h3;

// Mass to volume
V1 = M1*(x1/rhox + (1 - x1)/rhoy);
V2 = M2*(x2/rhox + (1 - x2)/rhoy);
V3 = M3*(x3/rhox + (1 - x3)/rhoy);

annotation(
    experiment(StartTime = 0, StopTime = 20, Tolerance = 1e-6, Inte
```

master* 0 0 23 mins Ln 80, Col 9 Spaces: 4 UTF-8 LF Modelica

Aplicaciones imperativas

Sistema DAE

- Método robusto
- *P*: Vector de parámetros.
- *U*: Vector de variables manipulables.
- *D*: Vector de variables de perturbación.
- *X*: Vector de variables de estado.
- *Y*: Vector de variables de salida.
- *t*: Vector de tiempo.
- *tspan*: Intervalo de integración.
- *X0*: Vector de valores iniciales de *X*.

```
% Integración de las ODEs
[t X] = ode15s(odefun,tspan,X0)

[P U D Y] = AEs_solution(t,X)
...

% Cálculo de las ODEs
function dXdt = odefun(t,X)
% X y dXdt son vectores columna

% Cálculo de las AEs para el valor
% puntual de t
[P U D Y] = AEs_solution(t,X)
dXdt = ...;
end

% Resolución del sistema AEs
function [P U D Y] = AEs_solution(t,X)
...
end
```

Sistema DAE

- Método específico de MATLAB y GNU OCTAVE
- *Y*: Vector de variables del sistema.
- *t*: Vector de tiempo.
- *tspan*: Intervalo de integración.
- *Y0*: Vector de valores iniciales de *Y*.
- M: Matriz de masa.

```
% Resolución del DAEs
M = [1 0 0; 0 1 0; 0 0 0]; % 2 ODE, 1 AE
options = odeset('Mass',M);
[t,Y] = ode15s(daefun,tspan,Y0,options);
...

% Evaluación del DAEs
function out = daefun(t,X)
% X y out son vectores columna
dXdt = ODEs;
residuos = AEs; // Igualadas a cero.
out = [dXdt, residuos];
end
```

Ejemplo de sistema DAE

Ecuaciones

$$\frac{dy_1}{dt} = -0.04y_1 + 10^4 y_2 y_3$$

$$\frac{dy_2}{dt} = 0.04y_1 - 10^4 y_2 y_3 - 3\times 10^7 y_2^2$$

$$0 = y_1 + y_2 + y_3 - 1$$

Variables

- Variables de estado: y_1, y_2
- Variable de salida: y_3
- Matriz de masa:

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Implementación en GNU Octave

Método robusto

```
% ODEs
function dX = ODEs(t,X)
  [y1 y2] = num2cell(X'){1,:};
  Y = AEs(t,X);
  [y3] = num2cell(Y){1,:};

  dy1 = -0.04*y1 + 1e4*y2*y3;
  dy2 = 0.04*y1 - 1e4*y2*y3 - 3e7*y2^2;

  dX = [dy1 dy2]';
endfunction % ODEs
%----------------------------------------

% AEs
function Y = AEs(t,X)
% Datos
  [y1 y2] = num2cell(X'){1,:};
  y3 = 1 - y1 - y2;

  Y = [y3];
endfunction % AEs

%=============== Programa =================

% Parámetros de simulación
graphsys = "gnuplot";

% Inicialización
X0 = [1; 0];

% Resolución
tpts = [0 4*logspace(-6,6)]';
[tpts,X] = ode15s(@ODEs,tpts,X0);

% Cálculo de las variables dependientes
n = size(tpts,1);
YY = zeros(n,7);

for i =1:n
  YY(i,:) = AEs(tpts(i),X(i,:)');
endfor
```

Método específico

```
% Sistema DAE
function out = robertsdae(t,y)
out = [-0.04*y(1) + 1e4*y(2).*y(3)
   0.04*y(1) - 1e4*y(2).*y(3) - 3e7*y(2).^2
   y(1) + y(2) + y(3) - 1 ];
end

% Resolución
y0 = [1; 0; 0];
tspan = [0 4*logspace(-6,6)];
M = [1 0 0; 0 1 0; 0 0 0];
options = odeset('Mass',M,'RelTol',1e-4,'AbsTol',[1e-6
1e-10 1e-6]);
[t,y] = ode15s(@robertsdae,tspan,y0,options);
```

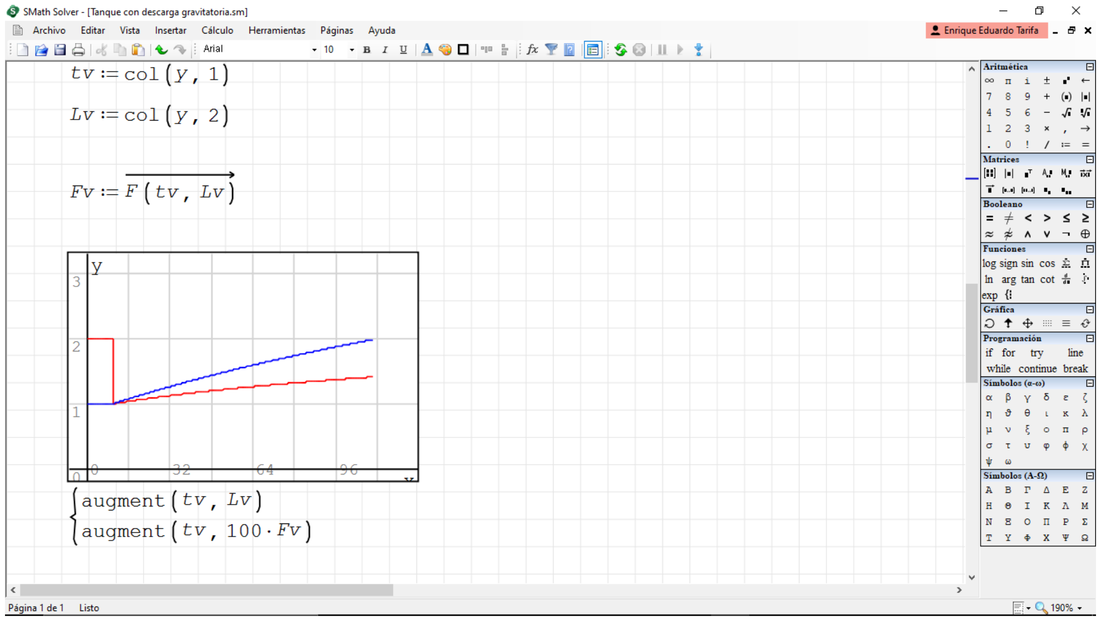
Implementación en GNU Octave

```
% ODEs
function dX = ODEs(t,X)
  [y1 y2] = num2cell(X'){1,:};
  Y = AEs(t,X);
  [y3] = num2cell(Y){1,:};
  dy1 = -0.04*y1 + 1e4*y2*y3;
  dy2 = 0.04*y1 - 1e4*y2*y3 - 3e7*y2^2;
  dX = [dy1 dy2]';
endfunction % ODEs
%----------------------------------------
% AEs
function Y = AEs(t,X)
  [y1 y2] = num2cell(X'){1,:};
  y3 = 1 - y1 - y2;
  Y = [y3];
endfunction % AEs

%=============== Programa =================
% Inicialización
X0 = [1; 0];

% Resolución
tpts = [0 4*logspace(-6,6)]';
[tpts,X] = ode15s(@ODEs,tpts,X0);

% Cálculo de las variables dependientes
n = size(tpts,1); YY = zeros(n,7);
for i =1:n
  YY(i,:) = AEs(tpts(i),X(i,:)');
endfor
```

```
% Sistema DAE
function out = robertsdae(t,y)
out = [-0.04*y(1) + 1e4*y(2).*y(3)
   0.04*y(1) - 1e4*y(2).*y(3) - 3e7*y(2).^2
   y(1) + y(2) + y(3) - 1 ];
end

% Resolución
y0 = [1; 0; 0];
tspan = [0 4*logspace(-6,6)];
M = [1 0 0; 0 1 0; 0 0 0];
options = odeset('Mass',M,'RelTol',1e-4,'AbsTol',[1e-6 1e-
10 1e-6]);
[t,y] = ode15s(@robertsdae,tspan,y0,options);
```

SMath Studio

- Notación matemática
- Unidades
- Vínculo con wxMaxima
- Controles
- Exportación de ejecutable
- Lento



Tanque con descarga gravitatoria

Parámetros del sistema

$F0 := 2 \cdot 10^{-3}$

$A := 0.785$

$Cv := 4.039 \cdot 10^{-5}$

$rho := 1000$

$g := 9.81$

Modelo del sistema

```
x(t) := if  t < 100
            0.5
         else
            0.25
```

AEs

$F(t, L) := Cv \cdot x(t) \cdot \sqrt{rho \cdot g \cdot L}$

ODEs

$dL(t, L) := \frac{F0 - F(t, L)}{A}$

Simulación del tanque

$D(t, X) := \left\| \begin{array}{l} [L] := X \\ [dL(t, L)] \end{array} \right.$

$X0 := [1]$

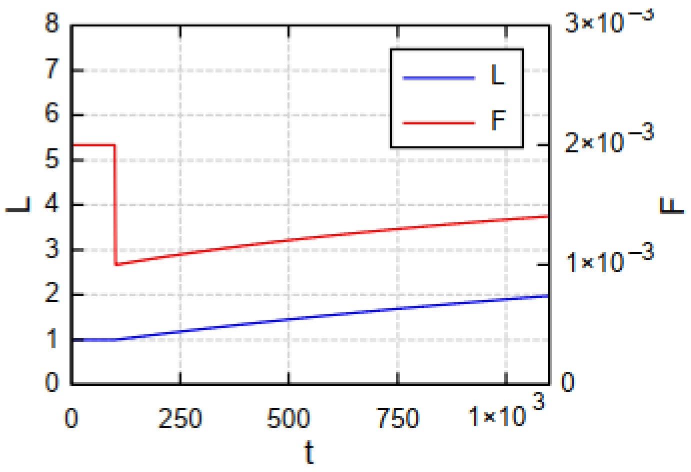
$t_0 := 0 \quad t_f := 1100 \quad N := 500$

$y := \mathrm{Rkadapt}(X0, t_0, t_f, N, D)$

$tv := \mathrm{col}(y, 1)$

$Lv := \mathrm{col}(y, 2)$

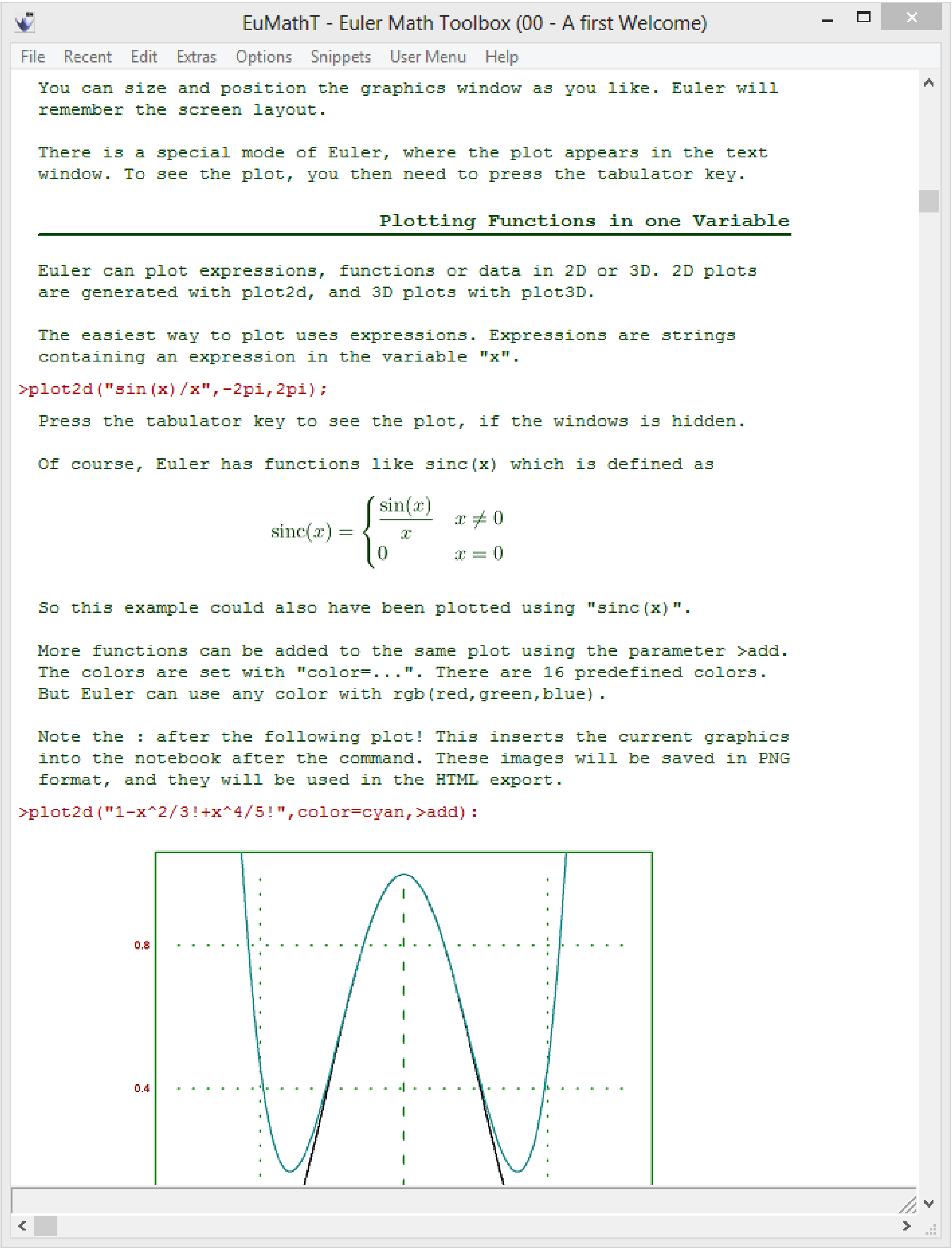
$Fv := \overrightarrow{F(tv, Lv)}$



augment(tv, Lv)
augment(tv, Fv)

Euler Math Toolbox

- Interfaz con celdas
- Vínculo con wxMaxima
- Comentarios con LaTeX
- Sintaxis similar a MATLAB
- Unidades



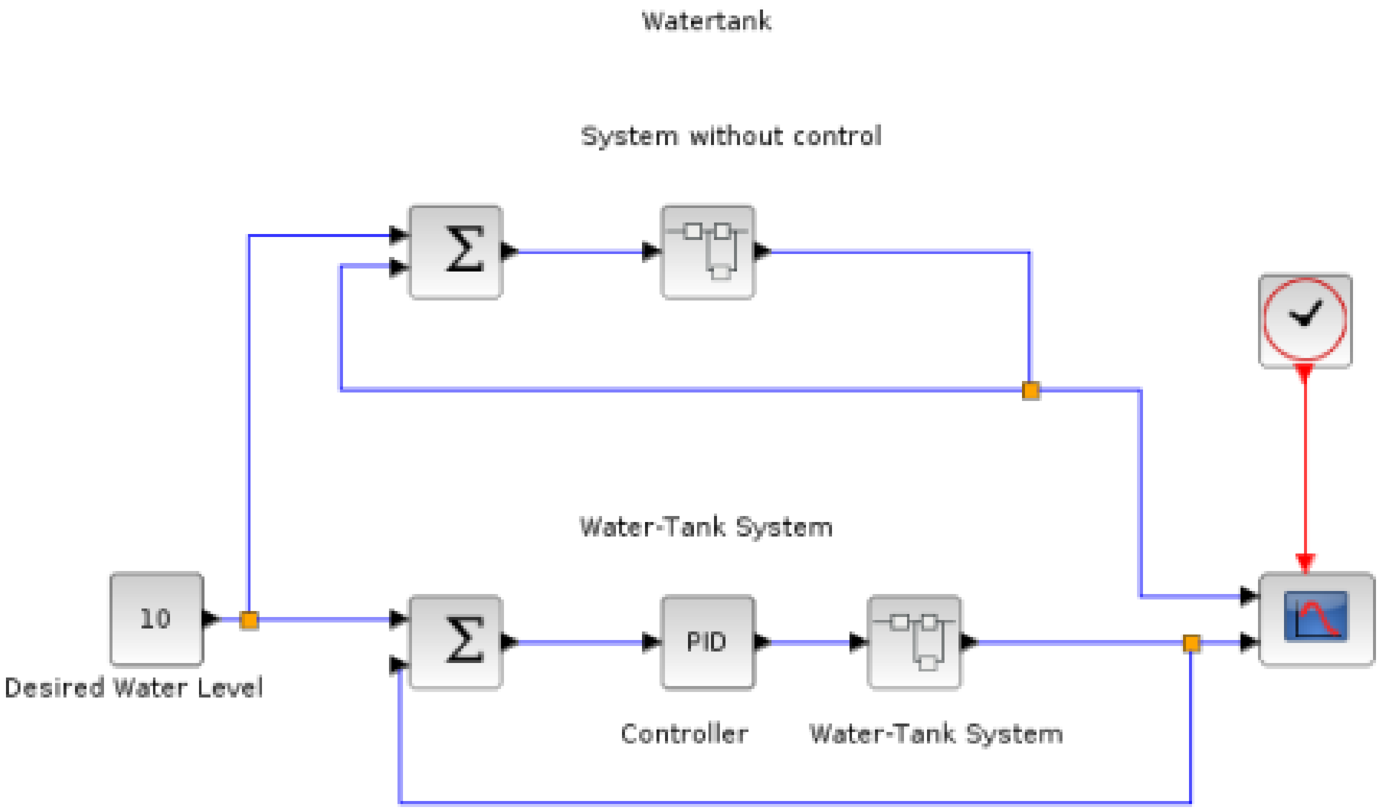
Euler Math Toolbox

Tanque con descarga gravitatoria

```
>F0 = 20E-3; A = 0.785; Cv = 4.039E-4; rho = 1000; g = 9.81;
>function x(t) ...

    if t < 10 then return 0.5;
       else return 0.25;
    endif
  endfunction

>function F(t,L) &= Cv*x(t)*sqrt(rho*g*L);
>function dL(t,L) &= (F0-F(t,L))/A;
>t = 0:1:110; L0 = 1;
>y = runge("dL",t,L0);
>plot2d(t,y,title="Nivel",xl="t",yl="L"):
>plot2d(t,map(F,t,y),title="Descarga",xl="t",yl="F"):
>
```

Scilab

- Cálculo numérico
- Orientado a matrices
- Sintaxis similar a MATLAB
- XCOS
- ODEs
- AEs

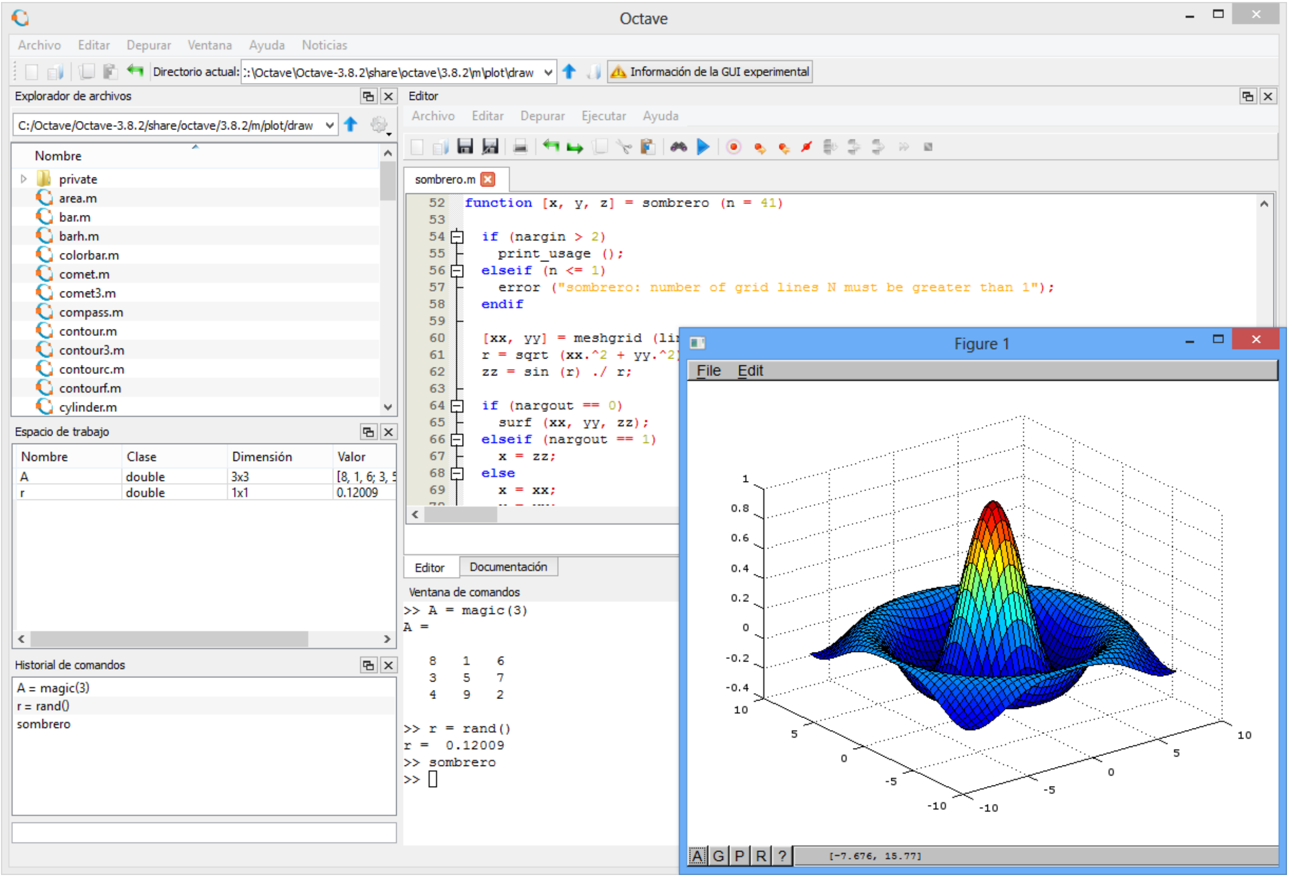


```
// Tanque con descarga gravitatoria
clear all;
close(winsid());
clc;

// ODEs
function dX = ODEs(t,X)
  Y = AEs(t,X);
  A = Y(1); Cv = Y(2); rho = Y(3); g = Y(4);
  x = Y(5); F0 = Y(6); F = Y(7);

  dL =(F0-F)/A;

  dX = [dL];
endfunction //ODEs
//----------------------------------------

//AEs
function Y = AEs(t,X)
// Datos
F0 = 20E-3; A = 0.785; Cv = 4.039E-4;
rho = 1000; g = 9.81; x = 0.5;

if t < 10
  x = 0.5;
  else
  x = 0.25;
end

L = X(1);

F = Cv*x*sqrt(rho*g*L);

Y = [A Cv rho g x F0 F];
endfunction //AEs
```

```
//================ Programa =================
// Parámetros de simulación
tfin = 110; nts = 100;

// Inicialización
L0 = 1; X0 = [L0];
tpts = linspace(0, tfin, nts)';

// Resolución
X = ode(X0,0,tpts,ODEs)';

YY = zeros(nts,7);

// Cálculo de las variables dependientes
for i = 1:nts
    YY(i,:) = AEs(tpts(i),X(i));
end

// Gráfica
// Y = [A Cv rho g x F0 F];
figure(1);
plot(tpts,X(:,1),"-o");
figure(2);
plot(tpts,YY(:,7),"-o");
figure(3);
plot(tpts,YY(:,5),"-o");
```

GNU Octave

- Cálculo numérico
- Orientado a matrices
- Sintaxis similar a MATLAB
- Muy compatible con MATLAB
- ODEs
- AEs

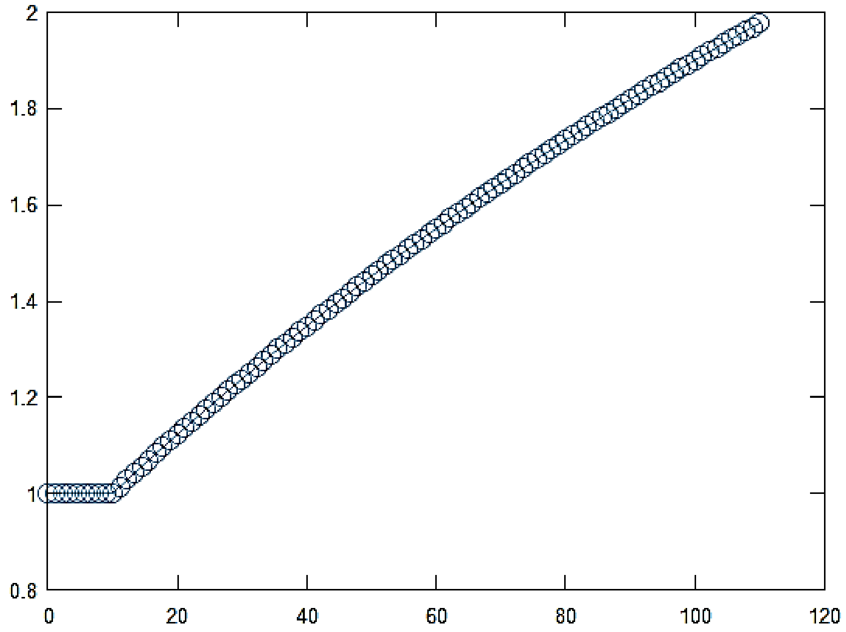


```
% Tanque con descarga gravitatoria
clear all; close all; clc;

% ODEs
function dX = ODEs(t,X)
Y = AEs(t,X);
[A Cv rho g x F0 F] = num2cell(Y){1,:};

dL =(F0-F)/A;

dX = [dL];
endfunction % ODEs
%----------------------------------------

% AEs
function Y = AEs(t,X)
% Parámetros
F0 = 20E-3; A = 0.785; Cv = 4.039E-4;
rho = 1000; g = 9.81;

if t < 10
  x = 0.5;
  else
  x = 0.25;
endif

L = X(1);

F = Cv*x*sqrt(rho*g*L);

Y = [A Cv rho g x F0 F];
endfunction % AEs
```

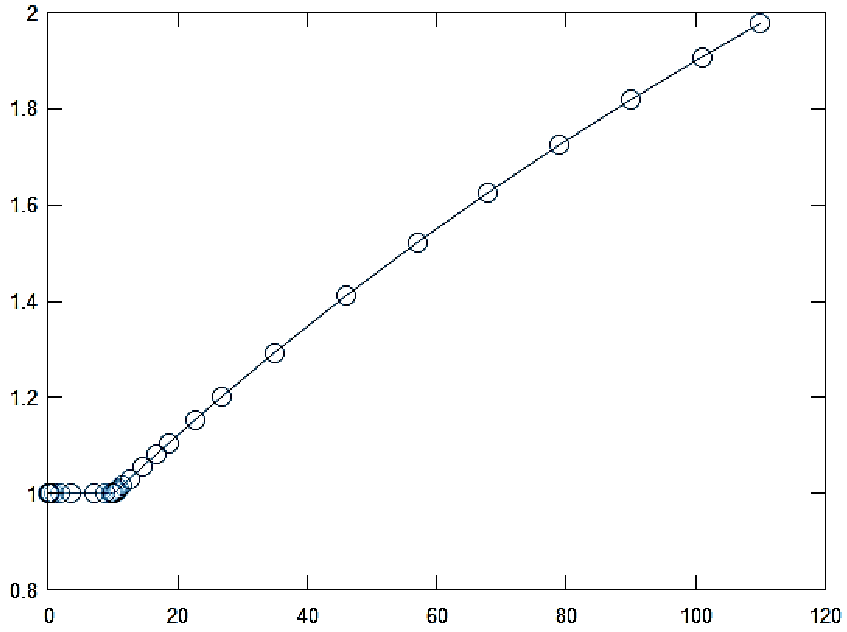
```
%================ Programa =================
% Parámetros de simulación
graphsys = "gnuplot";
tfin = 110; nts = 100;
adaptativo = false; % paso constante o variable

% Inicialización
L0 = 1;
X0 = [L0];

% Resolución
if ~adaptativo
  tpts = linspace(0, tfin, nts)';
  [tpts,X] = ode15s(@ODEs,tpts,X0);
  else % adaptativo
  [tpts,X] = ode15s(@ODEs,[0 tfin],X0);
endif

% Cálculo de las variables dependientes
n = size(tpts,1);
Y = zeros(n,7);
for i = 1:n
  Y(i,:) = AEs(tpts(i),X(i));
endfor

% Gráfica
% Y = [A Cv rho g x F0 F];
graphics_toolkit(graphsys);
figure(1);
plot(tpts,X(:,1),"-o");
```

Método adaptativo para el nivel

Sin adaptación

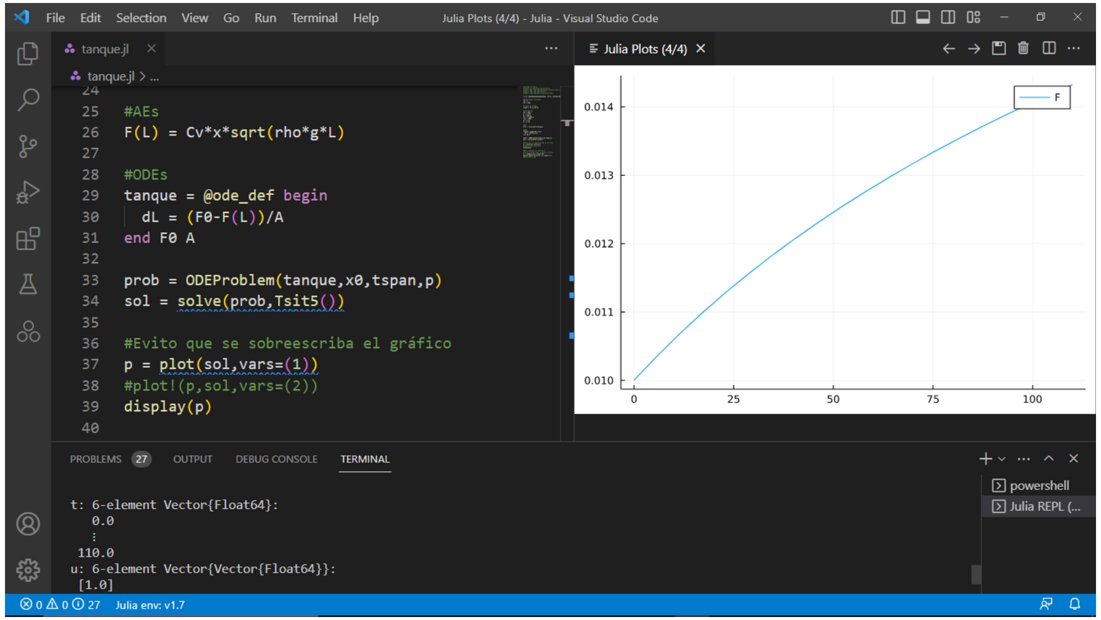


Con adaptación



Julia

- Lenguaje de propósito general
- Maneja matrices
- Sintaxis similar a MATLAB
- Compilación especial JIT
- ODEs
- AEs



```julia
#Listado en Julia
#Tanque con descarga gravitatoria

using ParameterizedFunctions, Plots,
OrdinaryDiffEq

#Condiciones iniciales
L0 = 1
x0 = [L0]

#Rango de tiempo
tspan = (0.0,110.0)

#Parámetros
F0 = 20E-3
A = 0.785
p = [F0 A]
Cv = 4.039E-4
rho = 1000
g = 9.81
x = 0.25
```

```julia
#AEs
F(L) = Cv*x*sqrt(rho*g*L)

#ODEs
tanque = @ode_def begin
  dL = (F0-F(L))/A
end F0 A

prob = ODEProblem(tanque,x0,tspan,p)
sol = solve(prob,Tsit5())

#Evito que se sobreescriba el gráfico
p = plot(sol,vars=(1))
#plot!(p,sol,vars=(2))
display(p)

#Para suavizar el plot de F
#Si no se hace esto, grafica 6 puntos.
t = range(0,sol.t[end],20)
plot(t,map(F,sol(t)[1,:]),label="F")
```