

Differential Equations

**Practical methods for ODE and PDE problem
solving with Excel**

Leonardo Volpi

Index

Differential Equations	5
Runge-Kutta formulas	5
ODE Multi-Steps Formulas	9
Multi-step coefficients tables	10
Predictor- Corrector.....	12
PECE algorithm of 2 nd order	12
PECE algorithm of 4 th order	14
Piecewise integration.....	16
Stability	17
A Stiff Problem	17
An "A-Stable" PC Schema	19
The Lotcka-Volterra Model	21
The van der Pol Equation	23
1st Order Linear ODE	24
Example 1. Homogeneous system.....	24
Example 2. Non-homogeneous system	25
Example 3. Piecewise integration	26
Using Finite-Differences	28
Example 4. Equation of motion.....	29
High order linear ODE	31
Vibrating spring-mass system 1	35
Vibrating spring-mass system 2	38
Electric Circuit 1	40
Electric Circuit 2	42
Elastic Beam 1.....	43
Non-Linear ODE	45
The Cardiac Cell Electric model	45
Rigid body equations.....	46
Pendulum	47
Solutions Family	48
Macro ODE Solver	51
Runge-Kutta-Fehlberg.....	51
Variable grid.....	54
Symbolic equations	55
Stiff ODE.	56
Piecewise differential equations	57
Discontinue derivative.....	58
Non Linear System.....	59
Non-Linear Electric Circuit.....	60
The Orbits	62
Macro ODE Slope Grid	64
About FD method	66
Grid, Mesh and Cells.....	66
Partial Differential Equations	68
Parabolic equation	68

Hyperbolic equation	68
Elliptic equation	68
Boundary and Initial Value Problem	69
Axes Scales	70
Macros working rules	71
2D-Laplace.....	72
Heat Diffusion of a 2D Rectangular Workpiece	72
Heat Diffusion of 2D Rectangular Workpiece with isolated edges	74
Laplace equation in irregular domain.....	75
2D-Poisson	76
Electric potential	76
Fluid flow velocity in a rectangular tube.....	77
2D Laplace Polar.....	78
Heat diffusion in a circular domain	79
Heat diffusion in a circular domain with constant temperature.....	80
Laplace polar equation over a full circular domain	81
2D Poisson Polar	82
Poisson polar equation in a circular domain.....	82
1D Heat Diffusion.....	84
Thermal diffusion along a bar	84
Thermal diffusion along a bar with insulated boundary.....	86
Thermal transient of a plate.....	87
1D Heat Convection	89
Linear Convection of a truncated sine wave	89
1D Wave.....	92
Wire oscillations.....	92
The guitar string	93
Linear ODE (BC) with FD	95
2 nd Linear ODE with boundary conditions	95
Linear ODE (IC) with FD	97
Linear, 1st order, differential equation with Initial condition	97
Linear, 2nd order, differential equation with Initial condition	99
Linear, 3rd order, differential equation with Initial condition.....	100
FD in Excel	101
The iterative mode.....	101
FD Formulas.....	102
Boundary conditions	104
PDE solving examples.	106
2D Laplace's equation	106
1D Heat Diffusion	108
Thermal Transient	110
2 nd order ODE - Boundary conditions	113
Credits.....	115
References.....	116

Differential Equations

Xnumbers and FDSolver¹ contain functions for solving 1st order differential problems with initial conditions (Cauchy's problem).

$$y' = f(t, y) \quad , \quad y(t_0) = y_0$$

and for solving 1st order differential systems.

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}) \quad , \quad \mathbf{y}(t_0) = \mathbf{y}_0 \quad \Leftrightarrow \quad \begin{cases} y_1' = f_1(t, y_1, y_2 \dots y_n) \\ y_2' = f_2(t, y_1, y_2 \dots y_n) \\ \dots \\ y_n' = f_n(t, y_1, y_2 \dots y_n) \end{cases} , \quad \begin{cases} y_1(t_0) = y_{10} \\ y_2(t_0) = y_{20} \\ \dots \\ y_n(t_0) = y_{n0} \end{cases}$$

Runge-Kutta formulas

The function ODE_RK4 integrates numerically a 1st order ordinary differential equation or a 1st order differential system, with the Runge-Kutta formula of 4th order

$$\begin{aligned} k_1 &= f(t_i, y_i) \\ k_2 &= f(t_i + 0.5 h, y_i + 0.5 h k_1) \\ k_3 &= f(t_i + 0.5 h, y_i + 0.5 h k_2) \\ k_4 &= f(t_i + h, y_i + h k_3) \\ y_{i+1} &= y_i + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

= ODE_RK4(Equations, VarInit, Step, [Par, ...])

"Equations" is a math expression string containing the equation to solve. For a (n x n) system, it is a vector of m equations. Correct equation definitions are:

$$y' = -2*y*x \quad , \quad v' = 2*x - v^2 + v \quad , \quad y1' = -3*y1 + y2 + \sin(10*t)$$

Each string may contain symbolic functions with variables, operators, parenthesis and other basic functions (for a complete list see the Xnumbers help-on-line).

The parameter "VarInit" is a vector containing the initial values.

For two variables "VarInit" it is a (1 x 2) vector: [t₀, y₀].

For a system with n+1 variables " VarInit " is an (1 x n+1) vector [t₀, y₁₀, y₂₀, ..., y_{n0}].

The parameter "Step" is the integration step.

The optional parameter "Par" contains the values of extra parameters.

Let's see how it works with an example

Solve numerically the following Cauchy's problem for $0 \leq x \leq 3$

$$y' = -2xy \quad , \quad y(0) = 1$$

We know that the exact solution is $y = e^{-x^2}$

We can arrange a worksheet like the following

¹ Xnumbers.xla and FDSolver.xla are freeware Excel add-ins by Foxes Team
<http://digilander.libero.it/foxes>

	A	B	C	D	E	F	G
1							
2	starting values				step	differential equation	
3							
4	x	y			h	diff. equation	
5	0	1			0.1	$y' = -2*x*y$	
6	0.1	0.99005					$\{=ODE_RK4(\$G\$5,A5:B5,$F\$5)\}$
7							
8							

As we can see, we have written in cell G5 the differential equation

$$y' = - 2*x*y$$

In the range A5:B5 we have inserted the starting values of x and y. Note that we have written the labels just above theirs values. Labels are necessary for the correct variables assignment.

Finally, in the range A6:B6 - just below the starting values - we have inserted the ODE_RK4, that returns the value $y(0.2) \cong 0.960789$, with a precision of about $1E-7$ (compare with the exact solution)

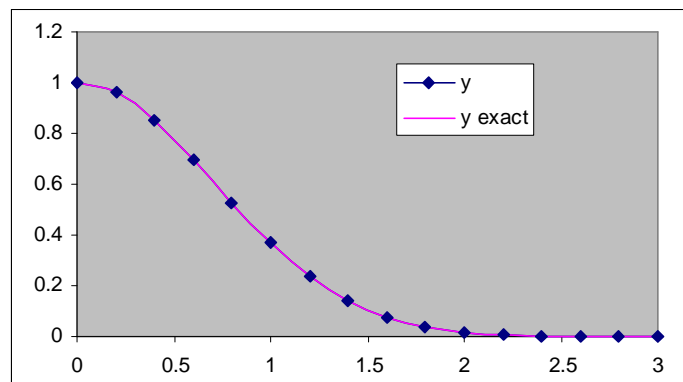
	x	y	y (exact)	error
4				
5	0	1	1	0
6	0.1	0.99005	0.9900498	4.158E-10
7	0.2	0.960789	0.9607894	3.917E-09
8	0.3	0.913931	0.9139312	1.125E-08
9	0.4	0.852144	0.8521438	1.65E-08
10	0.5	0.778801	0.7788008	2.528E-09
11	0.6	0.697676	0.6976763	6.111E-08
12	0.7	0.612627	0.6126264	2.158E-07
13	0.8	0.527293	0.5272924	5.065E-07
14	0.9	0.444859	0.4448581	9.705E-07
15	1	0.367881	0.3678794	1.625E-06
16	1.1	0.2982	0.2981973	2.459E-06
17	1.2	0.236931	0.2369278	3.428E-06
18	1.3	0.184524	0.1845195	4.459E-06

Tip: In order to get all other values, select the range A6:B6 and simply drag it down. The cells below will be filled automatically

Only remember to fix the cells G5 and F5 in the function by the "\$" symbol

$\{=ODE_RK4(\$G\$5,A5:B5,$F\$5)\}$

We have also added the column with the exact values in order to check the approximation error. Both exact and approximated solutions are plotted in the following graph



The approximate solution (dots) fits accurately the exact solution (pink curve).

If we need, we can include parameters inside the differential equation

Example. Solve the following differential problem

$$y' = -k \cdot x^n \cdot y$$

$$y(0) = 1$$

where $k = 2$ and $n = 1$

	A	B	C	D	E
1	diff. equation		h	k	n
2	$y' = -k \cdot x^n \cdot y$		0.1	2	1
3	<code>=ODE_RK4(\$A\$2,A6:B6,\$C\$2,\$D\$2,\$E\$2)</code>				
4					
5	x	y			
6	0	1			
7	0.1	0.9900498			
8	0.2	0.9607894			
9	0.3	0.9139312			
10	0.4	0.8521438			

Note that we have added the labels "k" and "n" above the cells D2 and E2. In this way, the parser will correctly assigns the value 2 to the variable "k" and the value 1 to the variable "n", in the differential equation string

Do not forget the labels "x" and "y" in the cells A5 and B5

Example: Solve the following linear differential equation

$$y' + \frac{1}{x} y = a \cdot x^n, \quad y(1) = 0$$

For $n = 3$ and $a = 1$. Rearranging, we get

$$y' = a \cdot x^n - \frac{y}{x}, \quad y(1) = 0$$

	A	B	C	D	E
1	diff. equation		h	a	n
2	$y' = a \cdot x^n - y/x$		0.1	1	3
3	<code>=ODE_RK4(\$A\$2,A6:B6,\$C\$2,\$D\$2,\$E\$2)</code>				
4					
5	x	y			
6	1	0			
7	1.1	0.1110019			
8	1.2	0.2480535			
9	1.3	0.417374			
10	1.4	0.6254631			

Note that we have added the labels "a" and "n" above the cells D2 and E2. In this way, the parser will correctly assigns the value 1 to the variable "a" and the value 3 to the variable "n", in the differential equation string

Do not forget the labels "x" and "y" in the cells A5 and B5

With step $h = 0.1$, we get a very accurate numerical solution comparing with the exact solution $y = (x^5 - x)/5x$, (average error about $1E-6$)

The ODE_RK4 function can be used to solve ordinary differential systems.

Example: Solve numerically the following differential system, where $v(t)$ and $i(t)$ are the voltage and the current of an electric network

$$\begin{cases} v' = i - 7 \cdot v \\ i' = -5 \cdot i + 15 \cdot v \end{cases} \quad \begin{cases} v(0) = 10 \\ i(0) = 0 \end{cases}$$

	A	B	C	D
1	$v' = i - 7 \cdot v$			h
2	$i' = -5 \cdot i + 15 \cdot v$			0.05
3				
4	$\{=ODE_RK4(\$A\$1:\$A\$2;A7:C7;\$D\$2)\}$			
5				
6	t	v	i	
7	0	10	0	
8	0.05	7.185458333	5.58875	
9	0.1	5.371308656	8.448001073	
10	0.15	4.174289895	9.701682976	
11	0.2	3.360887149	10.02694722	
12	0.25	2.788538799	9.830311635	
13	0.3	2.369953963	9.354496143	

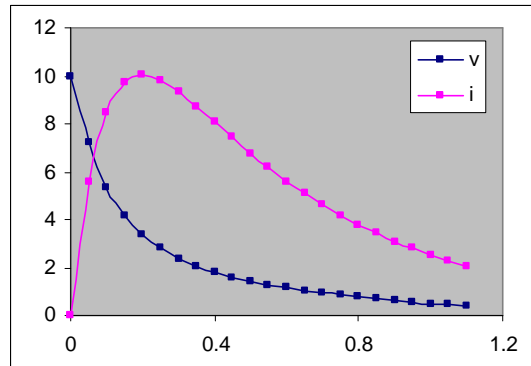
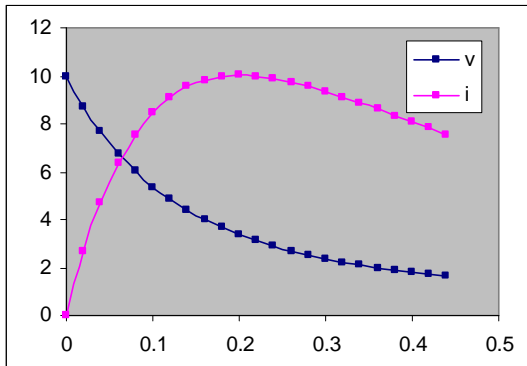
The computation can be arranged as following.

Write the variables labels in row 6. The labels "v" and "i" must be the same that you have written in the equations. Just one row below, insert the starting values in the same order.

Select the range A8:C8 and insert the function ODE_RK4. The first step will be returned.

Now select this row and drag it down for evaluating all the steps that you need

The graph below shows the transient of $v(t)$ and $i(t)$ with good accuracy. Note that you can re-compute the all transient changing the step "h" in a very easy and quick way.



Optional constant parameters can be arranged. For example if you want to add a parameter R, independent from the time "t", write:

	A	B	C	D
1	$v' = i - 7 \cdot v$		R	h
2	$i' = -R \cdot i + 15 \cdot v$		5	0.05
3				
4	$\{=ODE_RK4(\$A\$1:\$A\$2;A7:C7;\$D\$2;\$C\$2)\}$			
5				
6	t	v	i	
7	0	10	0	
8	0.05	7.185458333	5.58875	
9	0.1	5.371308656	8.448001073	

Constant parameters can be written in any part of the worksheet. You need only to add the labels with the same symbols with they appear in the differential equations. In this case, we have added the label "R" in the cell C1, upon its values.

You can add as many optional parameters that you like

ODE Multi-Steps Formulas

Another very popular method for integrating ordinary differential equations adopts the multi-step Adams' formulas. Even if a little formally complicated, they are very fast, and adapted to build a large family of ODE integration methods

The multi-step Adams' formulas can be generally written as:

$$y_{i+1} = y_i + \frac{h}{M} \sum_{k=1}^N b_{N-k} \cdot y'_{i-k+1} = y_i + \frac{h}{M} (b_{N-1} \cdot y'_i + b_{N-2} \cdot y'_{i-1} + \dots + b_0 \cdot y'_{i-N+1})$$

$$y_{i+1} = y_i + \frac{h}{M} \sum_{k=1}^N b_{N-k} \cdot y'_{i-k+2} = y_i + \frac{h}{M} (b_{N-1} \cdot y'_{i+1} + b_{N-2} \cdot y'_i + \dots + b_0 \cdot y'_{i-N+2})$$

where $y'_i = f(t_i, y_i)$ $t_i = t_0 + h \cdot i$

The first formula generates the explicit formulas – also called predictor formulas.

The second formula generates the implicit formulas – also called corrector formulas.

The number N is the order of the formula. A formula of N order requires N starting steps. Of course, formulas with high N are more accurate.

For N = 1 we get the popular Euler integration formulas

$y_{i+1} = y_i + h \cdot y'_i$	Euler's predictor (1 step)
$y_{i+1} = y_i + \frac{h}{2} \cdot (y'_{i+1} + y'_i)$	Trapezoid formula corrector (1 step)

Their errors are given by

$e \approx \frac{1}{2} h^2 y^{(2)}$	Error predictor 1 st order
$e \approx -\frac{1}{12} h^3 y^{(3)}$	Error corrector 2 st order

For N = 4 we get the popular Adams-Bashfort-Moulton predictor-corrector formulas

$y_{i+1} = y_i + \frac{h}{24} \cdot (55y'_i - 59y'_{i-1} + 37y'_{i-2} - 9y'_{i-3})$	Predictor (4 step)
$y_{i+1} = y_i + \frac{h}{24} \cdot (9y'_{i+1} + 19y'_i - 5y'_{i-1} + y'_{i-2})$	Corrector (4 step)

Their errors are given by

$e \approx \frac{251}{720} h^5 y^{(5)}$	Error predictor 4 th order
$e \approx -\frac{19}{720} h^5 y^{(5)}$	Error corrector 4 th order

There are a large set of predictor-corrector formulas

Multi-step coefficients tables

The following tables list the coefficients for the Adams predictor-corrector formulas up to the 9th order and relative errors

Multi-step Predictor coefficients

Ord P	1	2	3	4	5	6	7	8	9	10
M	1	2	12	24	720	1440	60480	120960	3628800	7257600
β_0	1	-1	5	-9	251	-475	19087	-36799	1070017	-2082753
β_1		3	-16	37	-1274	2877	-134472	295767	-9664106	20884811
β_2			23	-59	2616	-7298	407139	-1041723	38833486	-94307320
β_3				55	-2774	9982	-688256	2102243	-91172642	252618224
β_4					1901	-7923	705549	-2664477	137968480	-444772162
β_5						4277	-447288	2183877	-139855262	538363838
β_6							198721	-1152169	95476786	-454661776
β_7								434241	-43125206	265932680
β_8									14097247	-104995189
β_9										30277247

Multi-step Corrector coefficients

Ord P	1	2	3	4	5	6	7	8	9	10
M		2	12	24	720	1440	60480	120960	3628800	7257600
β_0		1	-1	1	-19	27	-863	1375	-33953	57281
β_1		1	8	-5	106	-173	6312	-11351	312874	-583435
β_2			5	19	-264	482	-20211	41499	-1291214	2687864
β_3				9	646	-798	37504	-88547	3146338	-7394032
β_4					251	1427	-46461	123133	-5033120	13510082
β_5						475	65112	-121797	5595358	-17283646
β_6							19087	139849	-4604594	16002320
β_7								36799	4467094	-11271304
β_8									1070017	9449717
β_9										2082753

Error coefficient

The general error is $e \approx -k \cdot h^{\text{Ord}+1} y^{(\text{Ord}+1)}$ where k is given by the following table

Ord P	1	2	3	4	5	6	7	8	9	10
predictor	0.5	0.41667	0.375	0.34861	0.32986	0.31559	0.30422	0.29487	0.28698	0.28019
corrector	-	-0.0833	-0.0417	-0.0264	-0.0188	-0.0143	-0.0114	-0.0094	-0.0079	-0.0068

The predictor-corrector algorithm

Usually the multi-step formulas, implicit and explicit, are used together to build a Predictor-Corrector algorithm. Here is how to build the 2nd order PEC algorithm (Prediction-Evaluation-Correction). It uses the Euler's formula as predictor and the trapezoidal formula as corrector

Prediction	Evaluation	Correction
$y_{p1} = y_0 + h f(t_0, y_0) \Rightarrow$	$f(t_1, y_{p1}) \Rightarrow$	$y_1 = y_0 + h/2 [f(t_0, y_0) + f(t_1, y_{p1})]$
$y_{p2} = y_1 + h f(t_1, y_{p1}) \Rightarrow$	$f(t_2, y_{p2}) \Rightarrow$	$y_2 = y_1 + h/2 [f(t_1, y_1) + f(t_2, y_{p2})]$
$y_{p3} = \dots$	\dots	\dots

The value y_1 can be reused to evaluate again the function $f(t_1, y_1)$ which can be used again in the corrector formula to obtain a more accurate value for y_1 .

If we indicate the first value obtained by the corrector with $y_1^{(1)}$ and the second value with $y_1^{(2)}$ we can arrange a new following schema

Prediction	Evaluation	Correction	Evaluation	Correction
$y_{p1} \Rightarrow$	$f(t_1, y_{p1}) \Rightarrow$	$y_1^{(1)} \Rightarrow$	$f(t_1, y_1^{(1)}) \Rightarrow$	$y_1^{(2)}$

This is the so called PECEC or $P(EC)^2$ schema.

The group EC can also be repeated m-times or even iterated still the convergence. In these cases we have the scheme $P(EC)^m$ and $P(EC)^\infty$ respectively.

Note that, for $m \gg 1$ the final accuracy depends mainly by the corrector.

Let's come back to the PEC schema.

We note that, at the step, we use the value $f(t_1, y_{p1})$ to predict the new value y_{p2}

We could increase the accuracy if we take the better approximation $f(t_1, y_1)$.

The new schema becomes:

Prediction	Evaluation	Correction	Evaluation
$y_{p1} \Rightarrow$	$f(t_1, y_{p1}) \Rightarrow$	$y_1 \Rightarrow$	$f(t_1, y_1) \Rightarrow$

This schema is called PECE and it is used very often being a reasonable compromise between the accuracy and the computation effort.

Using different scheme with different predictor-corrector formulas we can build a wide set of algorithms for the ODE integration. Of course they are not equivalent. Some of them have a high accuracy, others show a better efficiency and others have a better stability. This last characteristic may be very important for long integration intervals. In fact, the most algorithms, especially those with higher order, become unstable when the integration step grows over a limit. Algorithms that are stable for any integration step (so called A-stable algorithms) are much appreciated, but unfortunately they require implicit formulas that, generally, can be solved only by iterative algorithms.

The 2nd order trapezoid formula is one of the most popular A-stable formula.

Predictor- Corrector

= ODE_PRE(y_n, f, h)

= ODE_COR(y_n, fp, f, h)

These functions perform the integration of the ordinary differential equations with the popular multi-step predictor-corrector Adams' formulas

$$y' = f(t, y) \quad , \quad y(t_0) = y_0$$

The first function returns the predictor value $y_{n+1,p}$ while the second function returns the corrector y_{n+1} .

The parameter "yn" is the last point of the function y(t).

The parameter "f" is a vector containing the last N values of the derivative of y(t). That are the last N-1 values of the corrector.

The parameter "fp", only for the corrector, is the best approximation of the derivative of y(t) at the step n+1. Usually it is provided by a predictor formula

The parameter "h" sets the integration step

PECE algorithm of 2nd order

Now we see how arrange a PECE algorithm of 2nd order to solve a the following differential problem.

$$y' = -xy^2 \quad , \quad y(0) = 2$$

Let's set in a cell that we like the integration step "h" and then the heading of the data table. We set separate columns for predictor and corrector values

	A	B	C	D	E
1	Predictor-Corrector (PECE) of 2° order				
2					
3	h =	0.2			
4					
5	x	yp	fp	yc	fc
6	0	2	0	2	0
7					
8					
9					

Build the first row.

Begin to insert the starting values (x_0, y_0) in the cells A6 and B6 respectively, and the formula evaluations of f(x,y) in the cell C6 and E6. The corrector value is set equal to the starting value B6

	A	B	C	D	E
1	Predictor-Corrector (PECE) of 2° order				
2					
3	h =	0.2			
4					
5	x	yp	fp	yc	fc
6	0	2	0	2	0
7	0.2	2	-0.8	2	-0.8
8					

The second row is a bit more complicated. Let's see.

Select the first row A6:E6 and drag it down one row. This will copy the formula for **fp** and **fc**

Insert in the cell A7 the increment formula

$$x_{i+1} = x_i + h$$

Now we have to add the predictor and corrector function

	A	B	C	D	E
1	Predictor-Corrector (PECE) of 2° order				
2					
3	h =	0.2	=ODE_PRE(D6;E6;\$B\$3)		
4					
5	x	yp	fp	yc	fc
6	0	2	0	2	0
7	0.2	=ODE_PRE(D6;E6;\$B\$3)	-0.8	1.92	-0.73728
8					

Insert in the cell B7
 =ODE_PRE(yn, f, h)
 “yn” is the last value of y(x). contained in D6. “f” is the last value of f(x,y) contained in E6. “h” is the step B3.

	A	B	C	D	E
1	Predictor-Corrector (PECE) of 2° order				
2					
3	h =	0.2			
4					
5	x	yp	fp	yc	fc
6	0	2	0	2	0
7	0.2	2	-0.8	=ODE_COR(D6;C7;E6;\$B\$3)	-0.73728
8					
9					

Insert in the cell D7
 =ODE_COR(yn, fp, f, h)
 Where “yn” is the last value of y(x). In that case is D6. “f” is the last value of f(x,y), E6.
 “fp” is the predicted. value of f(x,y), C7 in this case.
 “h” is the constan step.

	A	B	C	D	E
1	Predictor-Corrector (PECE) of 2° order				
2					
3	h =	0.2			
4					
5	x	yp	fp	yc	fc
6	0	2	0	2	0
7	0.2	2	-0.8	1.92	-0.73728
8	0.4	1.772544	-1.2567649	1.72059551	-1.1841796
9	0.6	1.4837596	-1.3209255	1.470085	-1.2966899
10	0.8	1.21074701	-1.1727267	1.22314334	-1.1968637
11	1	0.9837706	-0.9678046	1.00667651	-1.0133976
12	1.2	0.80399699	-0.7756934	0.82776741	-0.8222387

Now the setting of the PECE algorithm of 2nd order is completed. Select the second row A7:E7 and drag it down in order to calculate the steps that you want.

The y_p and y_c values can be compared with the ones of the exact solution.

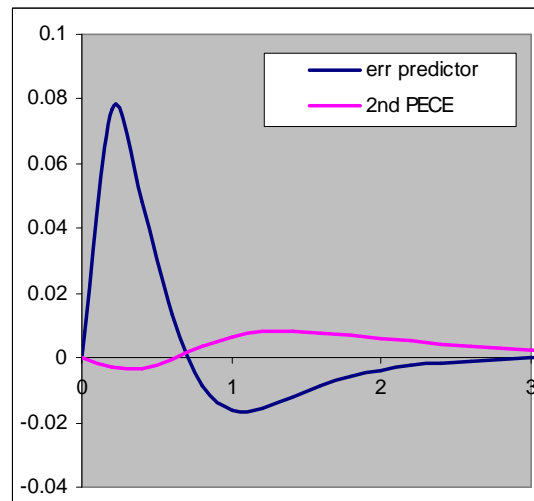
$$y = \frac{2}{1+x^2}$$

The differences:

$$d_{ip} = y_{ip} - y(x_i)$$

$$d_{ic} = y_{ic} - y(x_i)$$

are plotted in the graph at the right
 We note clearly the characteristic behavior of the predictor-corrector algorithm. The second formula refines the approximation of the first one.
 The final accuracy of PECE algorithm is practically the accuracy of the corrector



PECE algorithm of 4th order

Now we solve the differential equation of the previous example with a 4th order PECE algorithm using the 4 steps Adams-Bashfort-Moulton formulas

$$y' = -xy^2, \quad y(0) = 2$$

To start this algorithm needs 4 steps. A good set of starting steps is:

x	y(x)
0	2
0.2	1.9230769231
0.4	1.7241379310
0.6	1.4705882353

We do not investigate here how to get the extra 3 values (they could come by Runge-Kutta method or by Taylor series approximation). The only thing that we have to point out is that these values must be sufficiently accurate in order to preserve the global accuracy of the algorithm

The first 4 rows of the PECE algorithm are built as shown in the previous example.

	A	B	C	D	E
1	Predictor-Corrector (PECE) of 4^o order				
2					
3	h	0.2			
4					
5	x	yp	fp	yc	fc
6	0	2	0	2	0
7	0.2	1.9230769	-0.739645	1.9230769	-0.739645
8	0.4	1.7241379	-1.1890606	1.7241379	-1.1890606
9	0.6	1.4705882	-1.2975779	1.4705882	-1.2975779
10	0.8	=ODE_PRE(E	-1.2151057	1.217386	-1.1856229
11					
12		=ODE_PRE(B9;E6:E9;\$B\$3)			

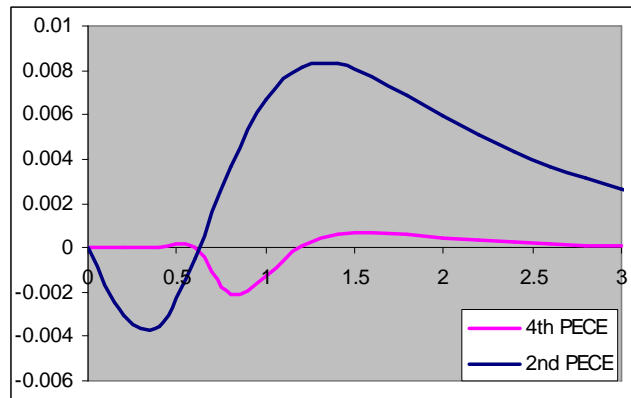
The first 4 values of yp and yc are the same. Now let's insert in the cell B10 =ODE_PRE(yn, f, h) where "yn" is the last value of y(x), D9 in that case. "f" is a vector of the the last four values of f(x,y), E6:E9 in this case. "h" is the step B3.

	A	B	C	D	E
1	Predictor-Corrector (PECE) of 4^o order				
2					
3	h	0.2			
4					
5	x	yp	fp	yc	fc
6	0	2	0	2	0
7	0.2	1.9230769	-0.739645	1.9230769	-0.739645
8	0.4	1.7241379	-1.1890606	1.7241379	-1.1890606
9	0.6	1.4705882	-1.2975779	1.4705882	-1.2975779
10	0.8	1.2324293	-1.2151057	=ODE_COR(D9;C	-1.1856229
11					
12		=ODE_COR(D9;C10;E7:E9;\$B\$3)			

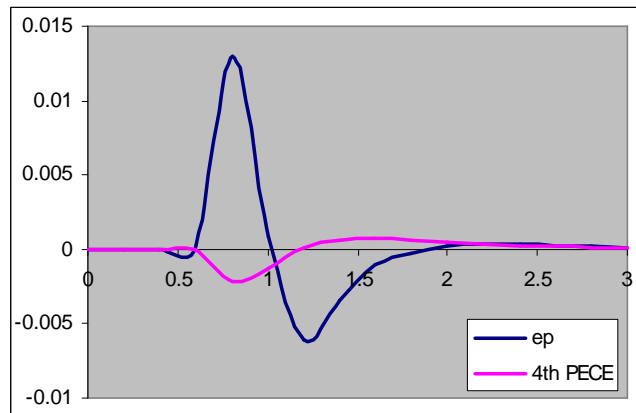
Insert in the cell D10 =ODE_COR(yn, fp, f, h) where "yn" is the last value of y(x). In that case D9. "f" is a vector of the last 3 values of f(x,y), E7:E9. "fp" is the predicted value of f(x,y), C10 in this case. "h" is the step B3

Now the setting of the PECE algorithm of 4th order is completed. Select the 5th row and drag it down in order to calculate the steps you want.

The predictor-corrector error curves are shown in the following graph



In order to compare the accuracy of the solutions of this algorithm with the 2nd order algorithm of the previous example, let's draw both the error curves in a same graph



As we can see, the 4th order algorithm is evidently more accurate than the 2nd order. On the other hand, the first one requires an extra work for providing 3 starting points.

Piecewise integration

A differential equation may have different definitions in different adjacent intervals of x-axis

$$y' = f_i(x, y) \quad \text{for } x_i \leq x < x_{i+1}$$

In that case we can repeat the integration for each intervals providing only that the starting values of each integration interval must be equal to the final values of the previous one.

Note that we may use different algorithms with different steps for each sub-interval

Example. Find the solution of the following IVP problem

$$y' = a(x)y, \quad y(0) = 1 \quad \text{where} \quad a(x) = \begin{cases} 1 & x \leq 1 \\ -2x+3 & 1 < x < 2 \\ -1 & x \geq 2 \end{cases} \Rightarrow \begin{cases} y' = y & x \leq 1 \\ y' = (3-2x)y & 1 < x < 2 \\ y' = -y & x \geq 2 \end{cases}$$

If we call y_1, y_2, y_3 the solutions in each sub-interval, for the continuity, must be $y_1(1) = y_2(1)$, $y_2(2) = y_3(2)$. The numerical solution can be found as the following

	A	B	
1	h		
2	0.2		
3	$y' = y$		
4			
5	x	y	
6	0	1	
7	0.2	1.2214	

Begin with the first equation.

Choose a suitable integration step in the cell A2, and insert the equation " $y' = y$ " in A3

Just below the labels "x", "y" insert the starting point (0, 1). in the cells A6:B6. Then, select the range A7:B7 and insert the following function

=ODE_RK4(\$A\$3,A6:B6,\$A\$2)

5	x	y	
6	0	1	
7	0.2	1.2214	
8	0.4	1.491818	
9	0.6	1.8221065	
10	0.8	2.2255208	
11	1	2.7182511	

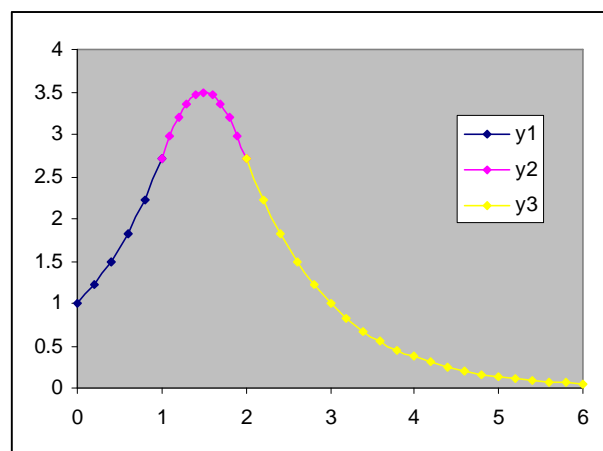
Now drag the range A7:B7 down till you reach the point $x = 1$

Repeat the task for the next equation $y' = (3-2x)y$

filling the next starting cells C6:D6 with the last values of the cells A11:B11

Below the final worksheet with the graph of the three functions y_1, y_2, y_3 . Note that for the central ODE we have used a smaller step then the others. The final accuracy is about $1E-5$ in all the range

	A	B	C	D	E	F
1	h		h		h	
2	0.2		0.1		0.2	
3	$y' = y$		$y' = (3-2x)*y$		$y' = -y$	
4						
5	x	y	x	y	x	y
6	0	1	1	2.71825	2	2.71825
7	0.2	1.2214	1.1	2.97424	2.2	2.22552
8	0.4	1.49182	1.2	3.1899	2.4	1.82211
9	0.6	1.82211	1.3	3.35345	2.6	1.49182
10	0.8	2.22552	1.4	3.45557	2.8	1.2214
11	1	2.71825	1.5	3.4903	3	1
12			1.6	3.45557	3.2	0.81874
13			1.7	3.35345	3.4	0.67033
14			1.8	3.1899	3.6	0.54882
15			1.9	2.97424	3.8	0.44934
16			2	2.71825	4	0.36789
17					4.2	0.3012
18					4.4	0.2466



Stability

The step h is the crucial parameter for the stability of any integration algorithm. If the step increases over a certain step h_s , the numerical solution suddenly blow up to the infinity; practically the algorithm will get an overflow error. The stability limit h_s depends by the method and by the ODE that we want to solve. We have to choose an integration step always less then stability limit: $h < h_s$.

Normally we can choose the integration step much less then the stability limit but sometime, especially in "stiff" problems, the above relation would be easily violated. In that case we see the solution graph "jumping" to unrealistic large values. When this occurs, we have simply to reduce the integration step, and thus, for covering the same integration interval, we have to take more grid points.

About this very important topic there are lots and lots of documentation. Here we want simply to show the problem with a simple, practical example.

A Stiff Problem

Assume to have the following 2nd order differential equation.

$$\frac{d^2 y}{d x^2} + (k+1) \frac{d y}{d x} + k y = 0 \quad y(1) = 1, \quad y'(1) = -0.5$$

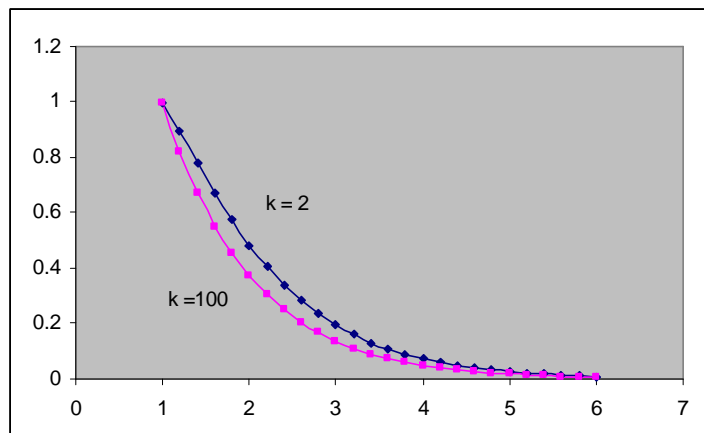
First of all, we transform the problem in a 1st order differential equations system taking $y_1 = y$, $y_2 = y'$. This leads to the following system

$$\begin{cases} y_1' = y_2 \\ y_2' = -k y_1 - (k+1) y_2 \end{cases} \quad \begin{cases} y_1(1) = 1 \\ y_2(1) = -0.5 \end{cases}$$

The exact solution $y(x)$ of this problem is

$$y(x) = \frac{1}{2k-2} [(2k-1)e^{1-x} - e^{k(1-x)}]$$

Now consider two solutions for $k = 2$ and $k = 100$. Their plots, sampled with a step of 0.2, are shown in the following graph



We can see that the behavior of the two functions seem similar. Both curves are smooth and regular. The step $h = 0.2$ appears adequate for a good plot in the range $1 < x < 6$ (about 26 points).

Now let's see what happens if we try to solve the two differential IVP problems for $k = 2$ and $k = 100$, with one method that we like, for example the 4th Runge-Kutta integrator, that usually shows good performance of accuracy and stability. About the integration step we can initially choose $h = 0.2$

$$K = 2$$

$$\begin{cases} y_1' = y_2 \\ y_2' = -2y_1 - 3y_2 \end{cases}$$

$$K = 100$$

$$\begin{cases} y_1' = y_2 \\ y_2' = -100y_1 - 101y_2 \end{cases}$$

The initial values are the same for both systems: $y_1(1) = 1, y_2(1) = -0.5$

We use the ODE_RK4 for solving both problem. Let's arrange a worksheet like the following

	A	B	C	D	E
1	Stiff Differential System				
2	$y_1' = y_2$			k	h
3	$y_2' = -k*y_1 - (k+1)*y_2$			2	0.2
4					
5			={ODE_RK4(A2:A3;A8:C8;E3;D3)}		
6					
7	x	y1	y2		
8	1	1	-0.5		
9	1.2	0.8929	-0.5577		
10	1.4	0.78077	-0.55605		
11	1.6	0.67257	-0.521923		
12	1.8	0.57301	-0.472009		
13	2	0.48412	-0.416412		
14	2.2	0.40641	-0.361017		
15	2.4	0.33947	-0.309043		

We have set the parametric system in the cells A2 and A3, the parameter k is in the cell D3 and the step h in the cell E3

The starting values [0, 1, -0.5] are in the range A8:C8.

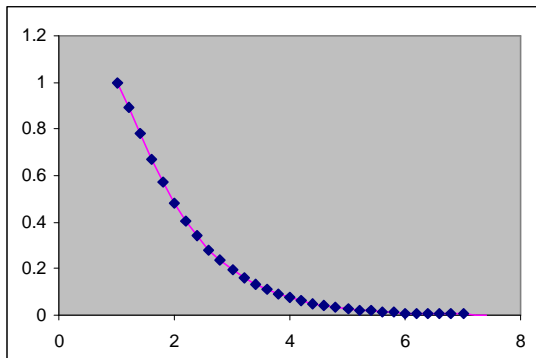
Do not miss the labels "k", "x", "y1", "y2". They are necessary for the correct variable assignment.

Select the range A9:C40 and insert the function ODE_RK4 with the ctrl+shift+enter key sequence

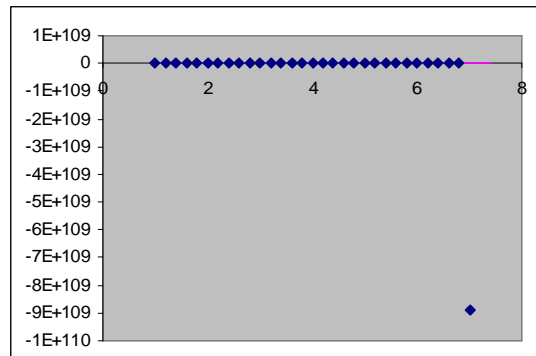
The numerical solution of the first problem appear excellent.

Also the relative error is very low: $err < 5.2E-05$. Now let's try to solve the second problem setting the cell D2 to 100. Saddenly the solution blow up to very large negative values.

K = 2, h = 0.2



K = 100, h = 0.2



Proceeding by trials, we reduce the integration step until the algorithm return stable. This occurs, for example, with $h \leq 0.025$. Of course, in order to integrate the same interval we need much more points: $n \geq 6/0.025 = 240$

Then, select the range A9:C250 and re-insert the function ODE_RK4, using $h = 0.025$. In this case we see the complete correct curve for $k = 100$. Note that in the previous case we have got a good, accurate solution with no more then 30 points.

This simple example shows the main drawback of a "stiff" ODE: we are obligated to choose a very small integration step to assure the stability even if the solution evolves slowly for a long integration interval. Usually a quick approach to solve a "stiff" problem is to choose a grid with many points (1000 - 9000) using a Runge-Kutta or a Predictor-Corrector schema. The last algorithm is preferable because more efficient for such long calculation

Another approach consist to use the so called "A-stable" integrators. Such algorithms are much more stable having a very large stability region ($h_s \rightarrow \infty$), and they are suitable to integrate "stiff" problem. For example, they could perform the intregation of the problem for $k = 100$ in 30-60 points or less

Clearly there is good reasons to develop dedicated method for stiff ODE.

The main drawbacks of "A-stable" method are the following: they have generally a low order and they are always implicit. This means that the core of the algorithm requires to solve a system of non-linear equations. For each step, an iterative process find the y_{n+1} solution. It is comprehensible, thus, that the efficiency of these methods tend to be quite low.

In Xnumbers the ODE_PC2I function uses the Euler formula as predictor and the implicit trapezoidal formula as corrector to implement an "A-stable" integrator of 2nd order for stiff ODE.

An "A-Stable" PC Schema

An "A-stable" Predictor-Corrector schema for integrate the differential equation $y' = f(t, y)$ uses the following formulas:

$y_{n+1} = y_n + h \cdot f(t_n, y_n)$	Euler's predictor (1 st ord)
$y_{n+1} = y_n + h/2 \cdot (f(t_n, y_n) + f(t_{n+1}, y_{n+1}))$	Trapezoidal formula corrector (2 nd ord.)

In order to preserve the A-stable performance of the trapezoidal formula the unknown y_{n+1} must be solved with the highest accuracy possible. Because in generally the function $f(t, y)$ is non-linear, we need an iterate process converging to the solution y_{n+1}

A naive approach could take to the following iterative process

$$y_{n+1}^{i+1} = y_n + h/2 \cdot (f(t_n, y_n) + f(t_{n+1}, y_{n+1}^i)) \quad (1)$$

where the initial value is given by the Euler predictor $y_{n+1}^0 = y_{n+1}^{(p)}$

We can easily show that this algorithm cannot converge if $h \rightarrow hs$ and thus is useless for solving stiff problem.

The barrier effect

Assume to have this simple test differential problem $y' = -I y$, $y(0) = 1$, $I = 5$

For this problem we know exactly the limit $hs = 2 / \lambda = 0.4$

Using the iterative process (1) we try to find $y_1 = y(h)$ for different value of h approaching 0.4, example $h = \{0.1, 0.2, 0.3, 0.35, 0.38...0.4-\epsilon\}$, where $\epsilon > 0$ and very small

	A	B	C	D	E
4	h =	0.1			
5	f(x0, y0)	-5			
6					
7	i	y1	f(x1, y1)	err	
8	0	0.5	-2.5		
9	1	=B\$2+B\$4*B5	-3.125	0.125	
10	2	0.59375	-2.96875	0.03125	
11	3	0.601563	-3.00781	0.007813	
12	4	0.599609	-2.99805	0.001953	
13	5	0.600098	-3.00049	0.000488	
14	6	0.599976	-2.99988	0.000122	
15	7	0.600006	-3.00003	3.05E-05	
16	8	0.599998	-2.99999	7.63E-06	
17	9	0.6	-3	1.91E-06	
18	10	0.6	-3	4.77E-07	

The iterative process (1) for finding y_1 can be arranged in an Excel worksheet

The starting values $y'_0 = -5$ and, thus, the predicted y_1 is 0.5

Functions contained are:

$$[B8] = 1+B4*B5$$

$$[C8] = -5*B8$$

$$[B9] = B$2+B$4/2*(B$5+C8)$$

$$[C9] = -5*B9$$

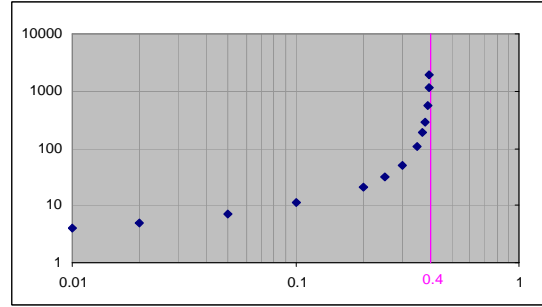
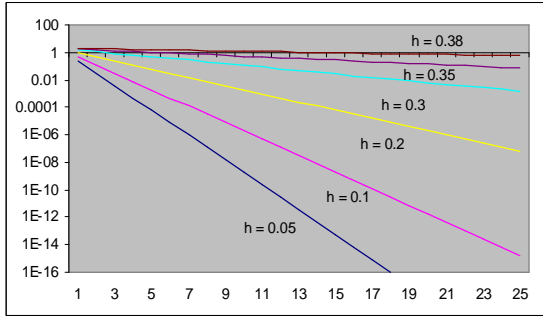
$$[D9] = ASS(B9-B8)$$

The column |err| measures the convergence.

Now select the range A9:D9 and drag it down.

The number of steps to reach a prefixed precision, is related to the convergence speed. In this case the speed seems quite fast. But is it always true for all values of h? Unfortunately not! As h increases the convergence speed becomes lower and lower, growing sharply when $h \rightarrow 0.4$

The following graph report the convergence trajectory for several increasing value of h



As we can see, for value $h > 0.35$ the trajectory approaches the vertical line. This means that the number of steps for reaching the convergence increases sharply as h approaches to 0.4. The graph shows a sort of "barrier" at $h = 0.4$. In addition, it is easy to see that for $h > 0.4$ the method diverges.

This simple example shows that the implicit schema PC with a "fixed point" iterative method is useless to solve "stiff" problems.

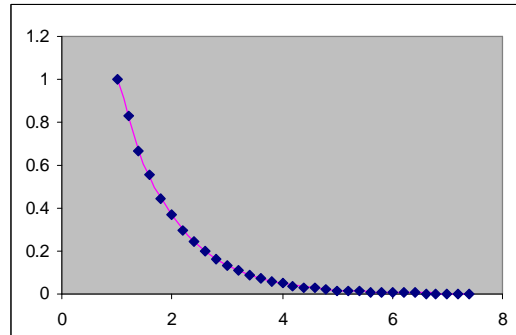
To overcome the barrier effect we have to solve the non-linear equation by a more efficient iterative algorithm such as the Newton method.

$$\Phi(y_{n+1}) = y_n + h/2 \cdot (f(t_n, y_n) + f(t_{n+1}, y_{n+1})) - y_{n+1}$$

For a differential system this method requires to calculate the Jacobian matrix at each step that it would be, thus, quite expensive. A method based on the secants interpolation (Broyden method) might be the better compromise. The function ODE_PC2I adopts this method for improving the convergence and, therefore, for avoiding the barrier effect.

Here is its result with the above "stiff" IVP problem.

	A	B	C	D	E
1	Stiff Differential System				
2	y1' = y2			k	h
3	y2' = -k*y1 - (k+1)*y2			100	0.2
4					
5	={ODE_PC2I(A2:A3;A8:C8;E3;D3)}				
6					
7	x	y1	y2		
8	1	1	-0.5		
9	1.2	0.82645	-1.235537		
10	1.4	0.66942	-0.334711		
11	1.6	0.55324	-0.827095		
12	1.8	0.44813	-0.224063		
13	2	0.37035	-0.553675		
14	2.2	0.29998	-0.149992		
15	2.4	0.24792	-0.370642		
16	2.6	0.20082	-0.100408		
17	2.8	0.16596	-0.248116		
18	3	0.13443	-0.067215		



As we can see, with a step $h = 0.2$ and no more than 40 points, the 2nd order implicit integrator ODE_PC2I returns the correct curve of a the stiff problem that, requires more than 240 points with 4th RK formula. The main advantage of ODE_PC2I is that we can choose the integration step without regarding the stability constraint.

The Lotcka-Volterra Model

The following two-dimensional differential system

$$\begin{cases} \frac{dx}{dt} = a_1 x - a_2 x y \\ \frac{dy}{dt} = -a_3 y + a_4 x y \end{cases}$$

is called the Lotcka-Volterra model or also "prey-predator" model. It is used in biology, chemistry and many other fields.

The function $x(t)$ and $y(t)$ depending from the time variable, may represent different things. In a biological models, for example, they simulate respectively the population of pray (rabbits) and predator (foxes) at time t . The proportionality constants a_1 , a_2 , a_3 , and a_4 are positive. It is known that its solution leads to oscillate steady-state.

The numerical integration of this ODE family requires a stable algorithm otherwise the result are not acceptable. Let's see with a practical example

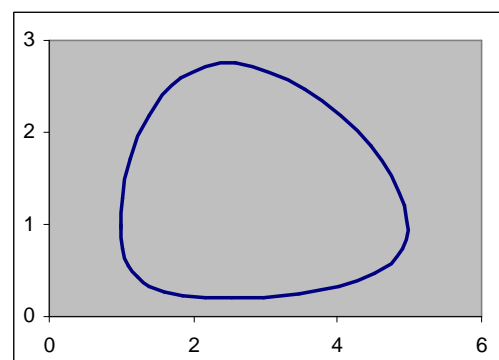
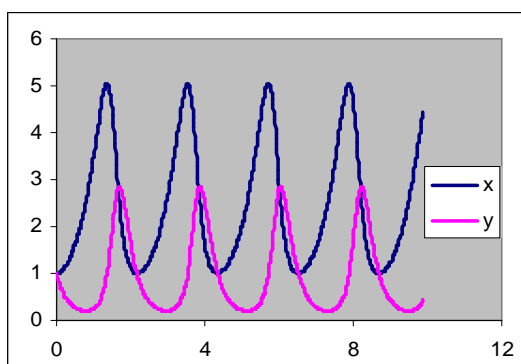
$$\begin{cases} x' = 2x - 2xy \\ y' = -5y + 2xy \end{cases} \quad \begin{cases} x(0) = 1 \\ y(0) = 1 \end{cases}$$

	A	B	C	D
1	$x' = 2*x - 2*x*y$		h	
2	$y' = -5*y + 2*x*y$		0.02	
3	={ODE_PC2I(A1:A2;A6:C6;C2)}			
4				
5	t	x	y	
6	0	1	1	
7	0.02	1.001166	0.941769	
8	0.04	1.004602	0.887009	
9	0.06	1.010194	0.835583	
10	0.08	1.017845	0.787347	

Insert the equation definition in the cells A1 and A2, the step in the cell C2 = 0.02 and the starting values [0, 1, 1] in the range A6:C6

Do not miss the labels "t", "x", "y"
Then, select the range A7:C407 and insert the function ODE_PC2I with the ctrl-shift-enter keys sequence

The following graphs at the left shows the result of the function $x(t)$ and $y(t)$. By performing the scatter plot of x - y variable we have the steady state plot at the right. It shows a closed loop, a limit cycle.



We can easily change the zoom of the graph simply by changing the integration step h
Note that the oscillations are quite large. The above system has two equilibrium points where $dx/dt = 0$ and $dy/dt = 0$ simultaneously.

Therefore for finding the equilibrium points we solve the algebraic system

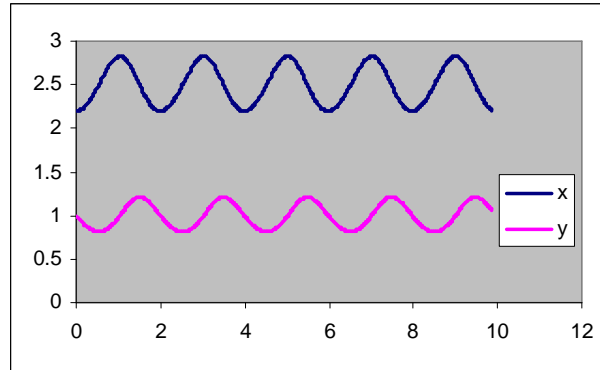
$$\begin{cases} 2x - 2xy = 0 \\ -5y + 2xy = 0 \end{cases} \Rightarrow \begin{cases} x = 0 \\ y = 0 \end{cases} \quad \begin{cases} x = 2.5 \\ y = 1 \end{cases}$$

The first solutions unstable while the second one is stable.

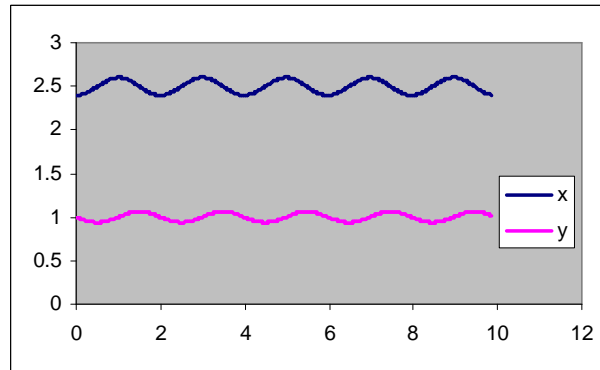
A stable solution means that if the initial populations $x(0) = 2.5$ and $y(0) = 1$ then there will not any oscillation and the population will be constant along the time. Starting near the stable equilibrium point the oscillations become to grow progressively.

Here same examples:

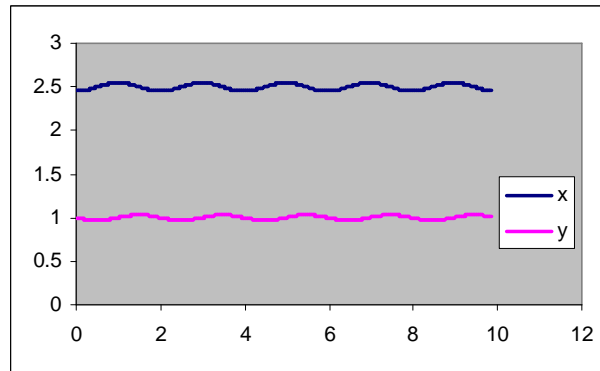
$x(0) = 2.2$ and $y(0) = 1$



$x(0) = 2.4$ and $y(0) = 1$



$x(0) = 2.45$ and $y(0) = 1$



The van der Pol Equation

The following non linear equation, called *van der Pol* equation, is known as a very stiff equation

$$y'' - \mu(1 - y^2)y' + y = 0$$

The oscillating solution shows regions where it changes slowly alternating with regions of very sharp change (quasi-discontinuities). As the parameter μ increases, the problem becomes more stiff.

Example. approximate the solution for $\mu = 8$ and 16 and for the initial value $[y(0) = 2, y'(0) = 0]$
 First of all, we transform the given 2nd order equation into its equivalent 1st order system setting $y_1 = y$ and $y_2 = y'$

$$\begin{cases} y_1' = y_2 \\ y_2' = \mu(1 - y_1^2)y_2 - y_1 \end{cases} \quad \begin{cases} y_1(0) = 2 \\ y_2(0) = 0 \end{cases}$$

	A	B	C	D	E
1					
2	$y_1' = y_2$			m	h
3	$y_2' = \mu(1 - y_1^2)y_2 - y_1$			8	0.05
4					
5	={ODE_PC2I(\$A\$2:\$A\$3;A7:C7;\$E\$3;D3)}				
6	x	y1	y2		
7	0	2	0		
8	0.05	1.998437	-0.062524		
9	0.1	1.994917	-0.078275		
10	0.15	1.9909	-0.08242		
11	0.2	1.986747	-0.083683		

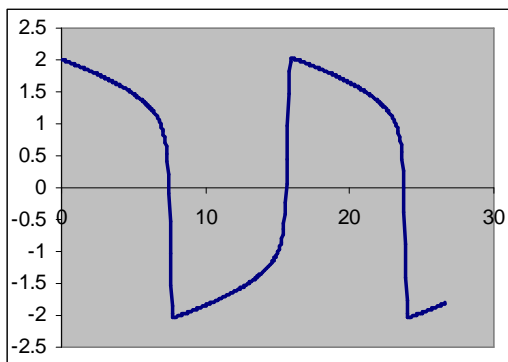
Insert the equations definitions in the cells A2 and A3, the parameter m and the step h in the cell D3 and E3 respectively.

Then insert the starting values in the range "A7:C7". Do not miss the labels "x", "y1", "y2" and "m".

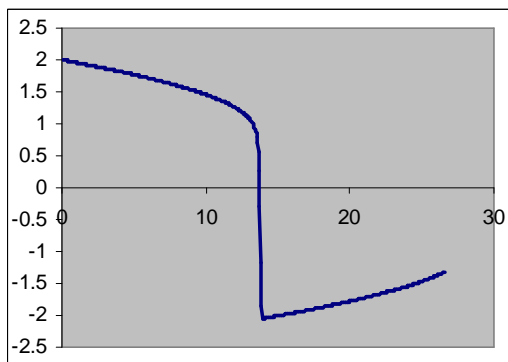
Select the range "A8:C500" and insert the function ODE_PC2I

Tip. In order to have a sufficiently accurate plot, we need many integration points (500 - 1000)

$\mu = 8, h = 0.05$



$\mu = 16, h = 0.05$



As we can see, the oscillating period increases with the parameter μ . For problems with large μ , the integration becomes extremely difficult and expensive. We also observe that for $h > 0.1$ the accuracy degrades fastly.

1st Order Linear ODE

Linear ODE can be numerical solved, of course, with the general methods RK and PC shown in the previous chapter. But their special structure allows to use more efficient dedicated algorithms.

The function ODE_SYSL, integrates numerically a system of ordinary linear differential equations of 1st order with constant coefficients.

For example, the general form of a 3x3 linear differential system is:

$$\begin{cases} y_1' = a_{11}y_1 + a_{12}y_2 + a_{13}y_3 + b_1 \\ y_2' = a_{21}y_1 + a_{22}y_2 + a_{23}y_3 + b_2 \\ y_3' = a_{31}y_1 + a_{32}y_2 + a_{33}y_3 + b_3 \end{cases}$$

where all the coefficients a_{ij} and b_i are constant.

Such system can be put in the following handy matrix form.

$$\bar{y}' = [A] \cdot \bar{y} + \bar{b}$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad \bar{y} = [y_1, y_2, y_3]^T$$
$$\bar{b} = [b_1, b_2, b_3]^T$$

and with the initial condition:

$$\bar{y}_0 = [y_1^{(0)}, y_2^{(0)}, y_3^{(0)}]^T$$

The constant term b is optional. If omitted the system is called "homogeneous"

This function uses the exponential expansion method that, for this kind of differential systems, is both accurate and efficient.

The function returns an $(n \times m)$ array containing all the nodes of the integration: m is the number of equations; n is the numbers of the nodes. The function automatically sets n equal to the rows of the range that you have selected.

Let's see how it works practically

Example 1. Homogeneous system

Solve the following homogeneous differential system with constant coefficients

$$\bar{y}' = [A] \cdot \bar{y}$$

where

$$A = \begin{bmatrix} -20 & -10 & 19 \\ 16 & 6 & -16 \\ -2 & -2 & 1 \end{bmatrix} \quad \bar{y}_0 = [1, 0, 0]^T$$

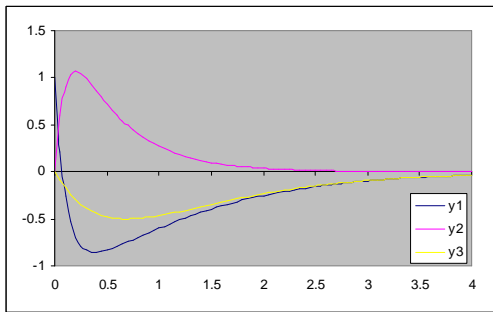
For $0 \leq x \leq 4$ and $h = 0.05$

The numerical solution can be arranged as in the following worksheet

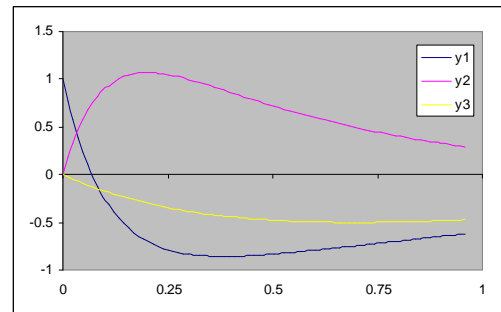
The initial values are in the first row (range B7:D7). The column "x" was added only for clarity but it is not indispensable at all. Select the range B8:D87, then insert the function ODE_SYSL giving the suitable parameters A, y_0 , h. Then press CTRL+SHIFT+ENTER

	A	B	C	D	E	F	G
1					-20	-10	19
2	h	0.05		A =	16	6	-16
3					-2	-2	1
4							
5		={ODE_SYSL(E1:G3;B7:D7;B2)}					
6	x	y1	y2	y3			
7	0	1	0	0			
8	0.05	0.21544	0.59661	-0.0928			
9	0.1	-0.2552	0.9017	-0.1722			
10	0.15	-0.5343	1.03538	-0.2398			
11	0.2	-0.6965	1.06997	-0.2968			
12	0.25	-0.7869	1.04889	-0.3445			
13	0.3	-0.8333	0.99805	-0.384			
14	0.35	-0.8524	0.93278	-0.4162			

All the 240 cells will be filled with the nodal solutions of y_1 , y_2 , y_3 .
The scale can be easily arranged simply by changing the parameter h



$h = 0.05$



$h = 0.012$

Compare with the exact solutions

$$\begin{cases} y_1 = -2e^{-x} + e^{-2x} + 2e^{-10x} \\ y_2 = 2e^{-2x} - 2e^{-10x} \\ y_3 = -2e^{-x} + 2e^{-2x} \end{cases}$$

Example 2. Non-homogeneous system

Solve the following homogeneous differential system with constant coefficients

$$y' = [A]y + b$$

where

$$A = \begin{bmatrix} 1 & 5 \\ -1 & -3 \end{bmatrix} \quad b = \begin{bmatrix} -4 \\ 2 \end{bmatrix} \quad y_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

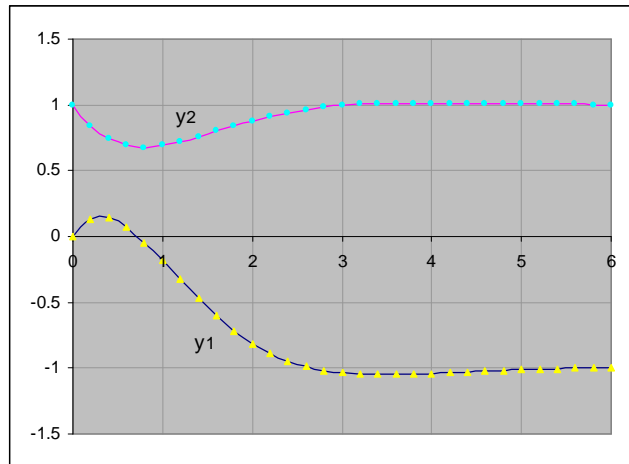
For $0 \leq x \leq 6$ and $h = 0.2$

The numerical solution can be arranged as in the right worksheet

The initial values are in the first row (range B7:C7). The column "x" was added only for clarity but it is not indispensable at all. Select the range B8:C37, then insert the function ODE_SYSL giving the suitable parameters **A**, **y₀**, **h** and **b**. Then press CTRL+SHIFT+ENTER

	A	B	C	D	E	F
1	h			A	b	
2	0.2			1	5	-4
3				-1	-3	2
4						
5		={ODE_SYSL(C2:D3;B7:C7;"A2,E2:E3)}				
6	x	y1	y2			
7	0	0	1			
8	0.2	0.12772	0.83734			
9	0.4	0.13948	0.73897			
10	0.6	0.07272	0.69012			
11	0.8	-0.04229	0.67767			
12	1	-0.18211	0.69044			
13	1.2	-0.32941	0.71928			
14	1.4	-0.47207	0.75699			

The tabulated functions y_1 and y_2 are shown in the following graph together with the exact solutions



The exact solutions are:

$$\begin{cases} y_1 = e^{-x}(\cos x + 2 \sin x) - 1 \\ y_2 = 1 - e^{-x} \sin x \end{cases}$$

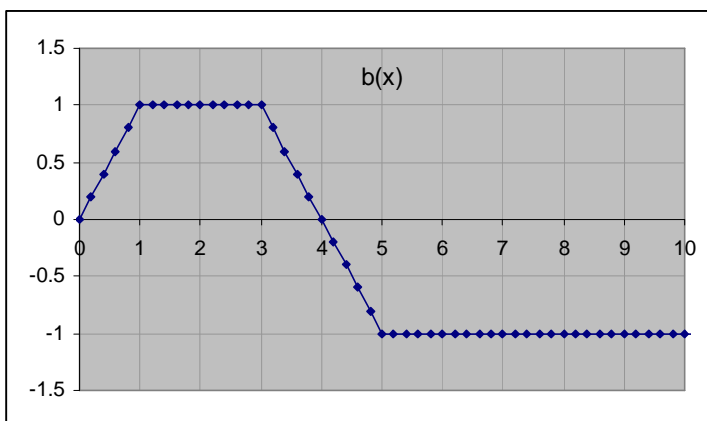
Example 3. Piecewise integration

The general first-order linear equation has the form

$$y' + a(x) \cdot y = b(x)$$

Of course, it can be rearranged in the normal form $y' = f(x, y)$ where $f(x, y) = b(x) - a(x) \cdot y$ and numerically solved with an RK or PC algorithm. Here we show how to get a quick solution when the coefficient $a(x)$ and $b(x)$ are linear inside adjacent sub-intervals: that is $x_i < x < x_{i+1}$

Example. Find a numerical solution of the IVP problem $y' + a y = b(x)$, $y(0) = 0$ where $a = 1$ and $b(x)$ is the function showed in the following graph



In this case it is quite simple to get the explicit equation of each sub-interval

$b(x) = x$	$0 \leq x < 1$
$b(x) = 1$	$1 \leq x < 3$
$b(x) = -x + 4$	$3 \leq x < 5$
$b(x) = -1$	$x \geq 5$

Then, we could substitute each function $b(x)$ in the differential equation and solving for each sub-interval. This method usually works but for many sub-interval becomes quite tedious. Overall we have to consider that the function $b(x)$ may be more complicated when it comes, for example, from experimental measurement. A quick solution may be obtained with the Finite-Differences method (FD). For this scope is useful the Excel add-in FDSolver.xla

In this case it is better to consider $b(x)$ as a $(n \times 1)$ vector, where n is the number of the grid points, that can be easily arranged in a spreadsheet column.

At the right we can see a possible arrangement.

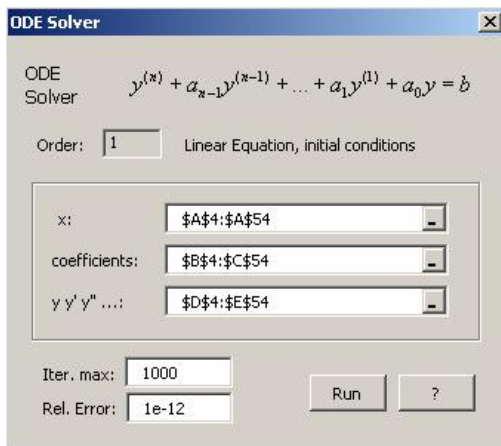
Insert the integration range x from 0 to 10, with step 0.2, in the range A4:A50. In the adjacent column insert the values of $a(x_i)$; in that case they are constant as $a = 1$.

In the next column insert the vector $b(x_i)$; each value may be taken from the given table or computed from the above function $b(x)$, as we prefer.

Leave the next two blank columns for the value of $y(x)$ and $y'(x)$. Remember only to insert the initial value $y(0)$ in the cell **D4** and color it as you like (yellow, for example)

	A	B	C	D	E
1	h	0.2			
2					
3	x	a	b	y	y'
4	0	1	0	0	
5	0.2	1	0.2		
6	0.4	1	0.4		
7	0.6	1	0.6		
8	0.8	1	0.8		
9	1	1	1		
10	1.2	1	1		
11	1.4	1	1		
12	1.6	1	1		
13	1.8	1	1		
14	2	1	1		
15	2.2	1	1		
16	2.4	1	1		
17	2.6	1	1		
18	2.8	1	1		
19	3	1	1		
20	3.2	1	0.8		
21	3.4	1	0.6		
22	3.6	1	0.4		
23	3.8	1	0.2		
24	4	1	0		

Now select the entire range A4:E54 and start the macro "**ODE Linear IC**" from the menu **Tools/FDSolver**.



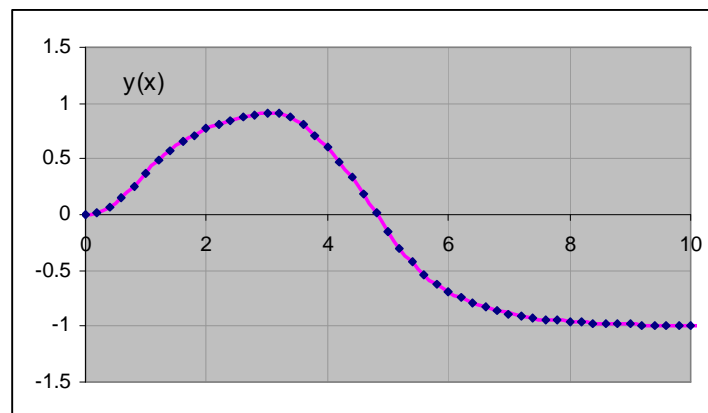
Using this macro is very simple. If you have arranged the worksheet as explained in the previous paragraph and if you have selected the input/output range before starting this macro, all the input boxes will be correctly filled. Normally you have only to press "Run".

The input boxes "Iter. max" and "Rel.Error" - thus the error of two consecutive iterations - are used by the macro only for the stopping criterion.

The macro output its results in the range y and y' (D4:E5) except in the cell that you have colored.

In a first-order IVP problem the colored cell must be the first cell of y .

In the following plot we show the $y(x_i)$ values obtained (dotted line) together with the exact solution (pink line). As we can see, matching seems quite satisfactory.



The exact solution of the give problem is

$$y(x) = \begin{cases} e^{-x} + x - 1 & 0 \leq x < 1 \\ e^{-x}(1-e) + 1 & 1 \leq x < 3 \\ e^{-x}(1-e-e^3) - x + 5 & 3 \leq x < 5 \\ e^{-x}(1-e-e^3+e^5) - 1 & x \geq 5 \end{cases}$$

You can verify that the average precision is better than 1E-3 (0.1%) for $0 < x < 10$. Of course the precision increases if you try with more fine grid.

Using Finite-Differences

We can also find the numerical solution of the previous example using only Excel features without any add-in. Let's see.

First of all we need a forward finite difference formula

$$\frac{y_{i+1} - y_i}{h} = \frac{y'_{i+1} + y'_i}{2} \Rightarrow y_{i+1} = y_i + \frac{h}{2}(y'_{i+1} + y'_i)$$

As we can see, this is the implicit trapezoid formula

Note. Before attempting to follow this example activate the iterative mode in Excel from the menu **tools/options...** (see chapter "FD - The iterative method" for better details)

Prepare a spreadsheet as in the previous example.

Insert the formula `=D4+B1/2*(E4+E5)` in D5 and `=C5-B5*D5` in E5. Note that this insertion forms a circular reference. Excel does not allow circular references except when iteration mode is activated. Now select the range D5:E5 and drag it down filling all the cells to row 54.

	A	B	C	D	E
1	h	0.2			
2					
3	x	a	b	y	y'
4	0	1	0	0	0
5	0.2	1	0.2	0.01818	0.18182
6	0.4	1	0.4		
7	0.6	1	0.6		
8	0.8	1	0.8		
9	1	1	1		
10	1.2	1	1		
11	1.4	1	1		
12	1.6	1	1		

	A	B	C	D	E	F
1	h	0.2				
2						
3	x	a	b	y	y'	
4	0	1	0	0	0	
5	0.2	1	0.2	0.01818	0.18182	
6	0.4	1	0.4	0.06942	0.33058	
7	0.6	1	0.6	0.14771	0.45229	
8	0.8	1	0.8	0.24813	0.55187	
9	1	1	1	0.36665	0.63335	
10	1.2	1	1	0.4818	0.5182	
11	1.4	1	1			
12	1.6	1	1			

Instantaneously the cells begin quickly to change until they stop themselves. Repeat the elaboration several times by pressing the F9 key. The values of the vectors y and y' will quickly approximate the solution of the differential equation.

Example 4. Equation of motion.

The motion of an inertial object under an influence of a variable force satisfies the ODE(Newton's equation second law)

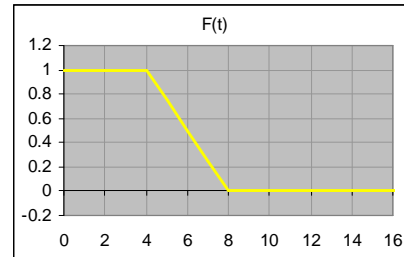
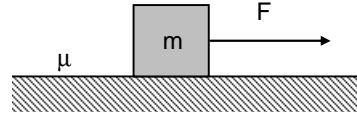
$$m x'' + \mu x' = F(t)$$

where μ is the kinetic friction, m is the mass and $F(t)$ is variable force having the law showed in the right graph.

$x(t)$ represent the position along the plane, $x'(t)$ the velocity and $x''(t)$ the acceleration.

Dividing for m we get the normal linear equation form

$$x'' + \frac{\mu}{m} x' = \frac{F(t)}{m}$$



We want to find the position and the velocity of the object for $0 < t < 50$, with $m = 2$, $\mu = 0.2$, and with the following starting conditions: $x(0) = 0$, $x'(0) = 0$

Here we use the finite-difference method. For that, prepare a worksheet as the following

	A	B	C	D	E	F	G
1	$x'' + (b/m)*x' = (F/m)$						
2	$m =$	2					
3	$\mu =$	0.2					
4	$h =$	1					
5	t	a0	a1	b	x	x'	x''
6	0	0	0.1	0.5	0	0	
7	1	0	0.1	0.5			
8	2	0	0.1	0.5			
9	3	0	0.1	0.5			
10	4	0	0.1	0.5			
11	5	0	0.1	0.375			
12	6	0	0.1	0.25			
13	7	0	0.1	0.125			
14	8	0	0.1	0			
15	9	0	0.1	0			
16	10	0	0.1	0			
17	11	0	0.1	0			

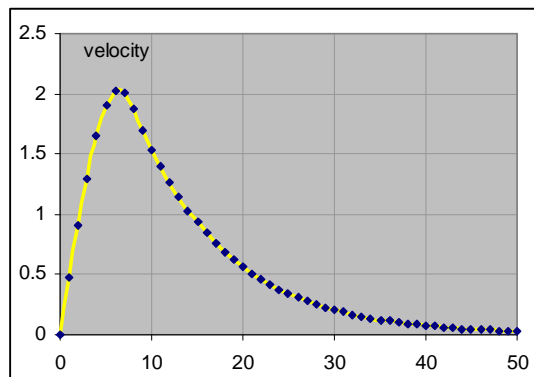
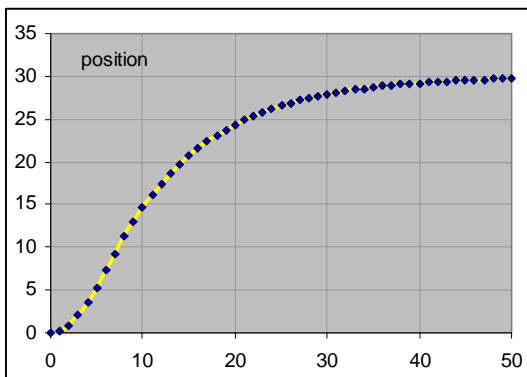
The given equation is equivalent to the 2nd order linear equation $x'' + a_1 x' + a_0 x = b$, where $a_0 = 0$, $a_1 = 0.1$ and $b(t)$ is

$$b(t) = \begin{cases} 0.5 & 0 \leq t < 4 \\ (1-t/8) & 4 \leq t \leq 8 \\ 0 & t > 8 \end{cases}$$

Note that we have chosen a quite large step ($h = 1$) to emphasize the robustness of the method. A suitable step should be 0.2 or lower.

Now select the range A6:G56 and start the macro "**ODE Linear IC**" from the menu **Tools/FDSolver**.

If you have followed the above arrangement the input boxes will be correctly filled and you have only to press "Run" and the macro will fill the columns x , x' , x'' . The following graphs show the first two columns (dotted lines) against the exact solutions.



As we can see, the fitting appears very good. Compare with the exact solutions.

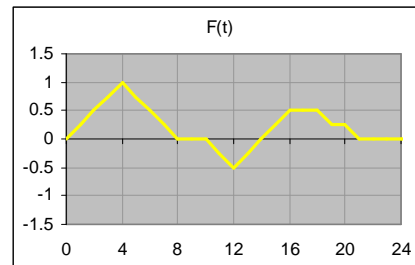
$$x(t) = \begin{cases} 50(e^{-t/10} - 1) + 5t \\ 25e^{-t/10}(5e^{2/5} + 2) - 5(t^2 - 36t - 376)/8 \\ 30 - 25e^{-t/10}(5e^{4/5} - 5e^{2/5} - 2) \end{cases} \quad x'(t) = \begin{cases} 5 - 5e^{-t/10} \\ -5e^{-t/10}(5e^{2/5} + 2)/2 - 5(t-10)/4 \\ 5e^{-t/10}(5e^{4/5} - 5e^{2/5} - 2)/2 \end{cases}$$

The relative error is better than 1% for all range, even with the relevant step used ($h = 1$).

FD versus PC or RK

If we integrate the equation with ODE_RK4 or ODE_PC4 using the piecewise method we can find a very similar graph but with an higher global accuracy thanks to their higher order. On the other hand FD method becomes useful when we have irregular or very complicated function.

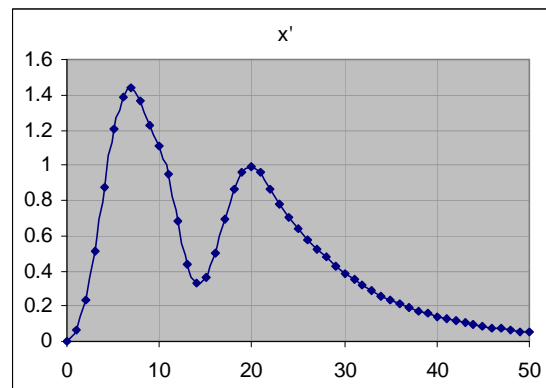
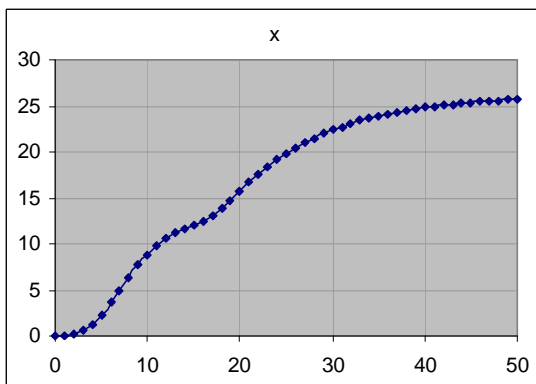
For example, if we have to solve the equation of motion of the previous example with a variable force $F(t)$ like the graph to the right, it would be very complicated and tedious to derive each equation for each sub-interval. It is more convenient to sample the function $F(t)$ and solve the ODE with an FD method. The sampled data of $F(t_i)$ are listed in the following tables. The sampling time is 1 sec.



t	0	1	2	3	4	5	6	7	8	9	10
F(t)	0	0.25	0.5	0.75	1	0.75	0.5	0.25	0	0	0

t	11	12	13	14	15	16	17	18	19	20	> 20
F(t)	-0.25	-0.5	-0.25	0	0.25	0.5	0.5	0.5	0.25	0.25	0

If we assign the values of the vector $b(i) = F(i)/m$ in the worksheet showed in the previous example and we solve the ODE with the macro ODE Linear IC, we find the following nice graphs of the position and velocity.



Note how solving is quick and simple with FD method even in this case.

High order linear ODE

The function ODE_SYSL can also be used to solve high order linear ODE with constant coefficients, that in general can be written as

$$y^{(n)} + a_{n-1}y^{(n-1)} + \dots + a_2y'' + a_1y' + a_0y = b$$

with the initial conditions

$$y(x_0) = y_0, y'(x_0) = y_0', y''(x_0) = y_0'', \dots, y^{(n-1)}(x_0) = y_0^{(n-1)}$$

As known, such ODE can be transformed into a linear differential system of 1st order.

$$\begin{bmatrix} y_1' \\ y_2' \\ \mathbf{L} \\ y_{n-1}' \\ y_n' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \mathbf{M} & 0 \\ 0 & 0 & 1 & \mathbf{M} & 0 \\ \mathbf{K} & \mathbf{K} & \mathbf{K} & \mathbf{O} & \mathbf{K} \\ 0 & 0 & 0 & \mathbf{M} & 1 \\ -a_0 & -a_1 & -a_2 & \mathbf{M} & -a_{n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \mathbf{L} \\ y_{n-1} \\ y_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \mathbf{L} \\ 0 \\ b \end{bmatrix}$$

having the initial conditions

$$\bar{y}_0 = [y_1(0), y_2(0), y_3(0), y_n(0)]^T = [y(0), y'(0), y''(0), y^{(n-1)}(0)]^T$$

For example assume that you have to calculate the solution of following IVP problem:

$$y''' + 5y'' + 9y' + 5y = 0, \text{ with } y(0) = 2, y'(0) = -3, y''(0) = 4$$

Introducing the auxiliary variables $y_1 = y$, $y_2 = y'$, $y_3 = y''$, we get the following equivalent differential system

$$\begin{bmatrix} y_1' \\ y_2' \\ y_3' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -5 & -9 & -5 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad \text{with} \quad \begin{cases} y_1(0) = 2 \\ y_2(0) = -3 \\ y_3(0) = 4 \end{cases}$$

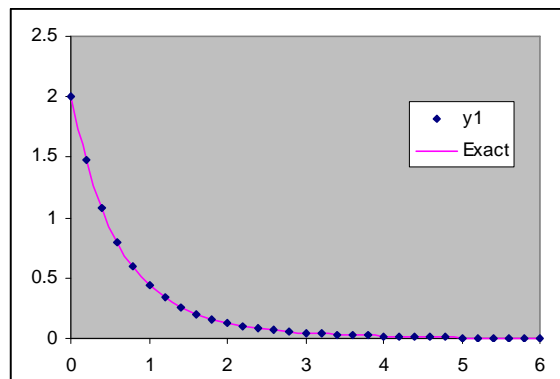
Observe that the last row of the matrix contains the coefficients of the given ODE with the opposite sign; besides of that, it has only all "1" in the upper subdiagonal.

Insert the initial values in the first row (range B10:D10). The column "x" was added only for clarity but it is not indispensable at all. Select the range B11:D40, where you want to output the results and insert the function ODE_SYSL giving the suitable parameters: A, y0, h.

Then press CTRL+SHIFT+ENTER

The selected area will be filled with the numerical solution of the given system

	A	B	C	D	E
1	A =	0	1	0	
2		0	0	1	
3		-5	-9	-5	
4					
5	h =	0.2			
6					
7		={ODE_SYSL(B1:D3;B10:D10;B5)}			
8					
9	x	y1	y2	y3	
10	0	2	-3	4	
11	0.2	1.47569	-2.2658	3.32229	
12	0.4	1.08418	-1.673	2.61181	
13	0.6	0.7974	-1.2161	1.97484	
14	0.8	0.58999	-0.8755	1.45064	
15	1	0.444	-0.628	1.04377	



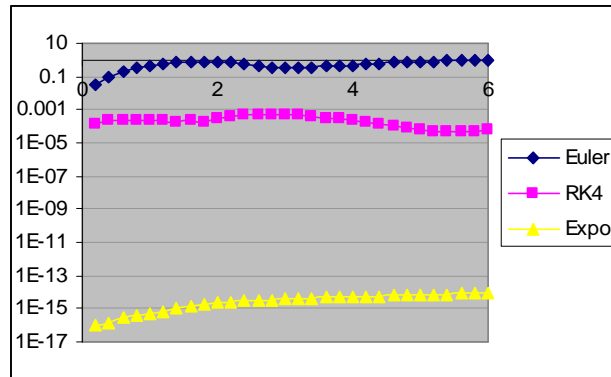
Observe that the solution $y_1(x)$ is also the solution $y(x)$ of the given ODE. Compare with the exact solution

$$y(x) = e^{-x} + e^{-2x} \cos x$$

Of course, the above differential system may also be solved with other methods.

In order to show the accuracy of the exponential method we put in a graph the average relative error $e(x) = |y_i - y(x_i)| / |y(x_i)|$, obtained in the same condition, with 3 different methods: Exponential, Euler, and Runge-Kutta 4

The graph is eloquent. The error of the Exponential method is several orders more accurate than the others



Clearly it takes advantages using dedicated methods for linear differential equations.

Another important feature of the exponential method is its high stability

Let's try the following test stiff system

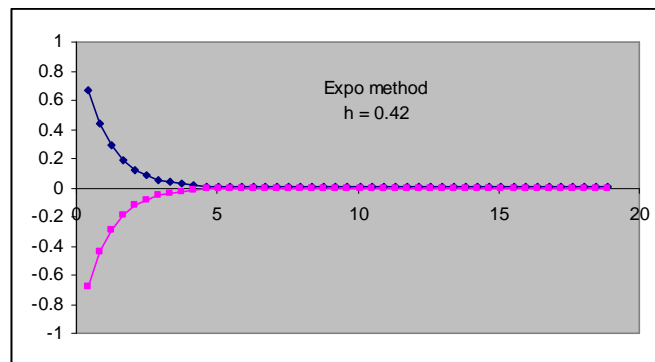
$$\begin{cases} y_1' = y_2 \\ y_2' = -20y_1 - 21y_2 \end{cases} \quad \begin{cases} y_1(0.4) = 0.6706555 \\ y_2(0.4) = -0.6770293 \end{cases}$$

The exact solution is

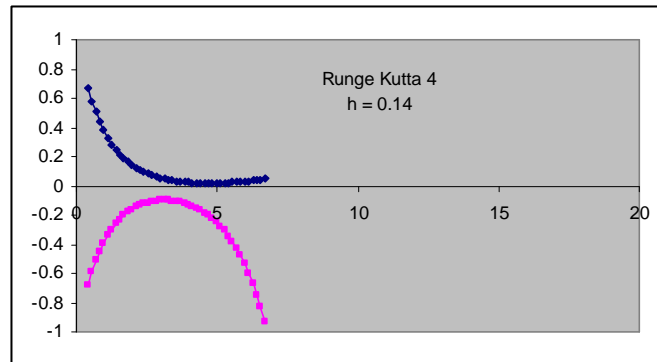
$$\begin{cases} y_1 = e^{-x} + e^{-20x} \\ y_2 = -e^{-x} - 20e^{-20x} \end{cases}$$

In the following graph we shows the numerical result obtained with three different integration methods, Exponential, Runge Kutta of 4th order and Euler, using different integration steps

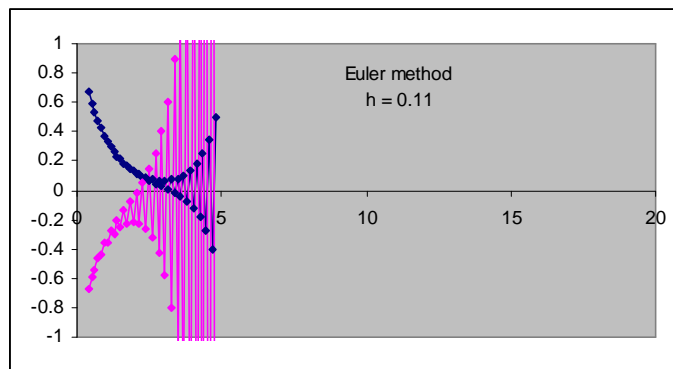
The Expo method reaches the integration length of about $x = 19$ with a step $h = 0.4$



The Runge-Kutta method, using a step $h = 0.14$ gives accurate only for $x < 3$; after that the solution begins to diverge.



The Euler method, with a step $h = 0.11$, begins to oscillate even from the first steps. For this step the method is completely unstable.



As we can see the stability step of the Exponential method is here about 3 times greater than the others methods.

Example. Solve the following linear differential equation of 3rd order.

$$y''' + 5y'' + 33y' + 29y = 0 \quad \text{with} \quad y(0) = 2, \quad y'(0) = -3, \quad y''(0) = -20$$

This linear equation with constant terms can be transformed into the equivalent differential system:

$$\begin{bmatrix} y_1' \\ y_2' \\ y_3' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -29 & -33 & -5 \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad \text{with} \quad \begin{cases} y_1(0) = 2 \\ y_2(0) = -3 \\ y_3(0) = -20 \end{cases}$$

	A	B	C	D	E	F	G
1	y''' + 5y'' + 33y' + 29y = 0			y(0) = 2	y'(0) = -3	y''(0) = -20	
2							
3	h =	0.05	=ODE_SYSL(B5:D7;B10:D10;B3)				
4							
5	A =	0	1	0			
6		0	0	1			
7		-29	-33	-5			
8							
9	x	y1	y2	y3			
10	0	2	-3	-20			
11	0.05	1.8279	-3.8239	-12.982			
12	0.1	1.6233	-4.3044	-6.3333			
13	0.15	1.4028	-4.4697	-0.4229			
14	0.2	1.1809	-4.3634	4.4941			
15	0.25	0.9701	-4.0392	8.2743			
16	0.3	0.7796	-3.5556	10.874			
17	0.35	0.6103	-3.0209	13.222			

Solving numerically this differential system is very rapid.

Simply write the system matrix A in the worksheet, for example, in the range B5:D7. Insert the row starting values, [2,-3,-20] in the range B10:D10, where you want to begin the functions tabulation. Choose a suitable step that you like, for example h = 0.05

Then, select the range B11:D60 and insert the function

=ODE_SYSL(B5:D7;B10:D10;B3)

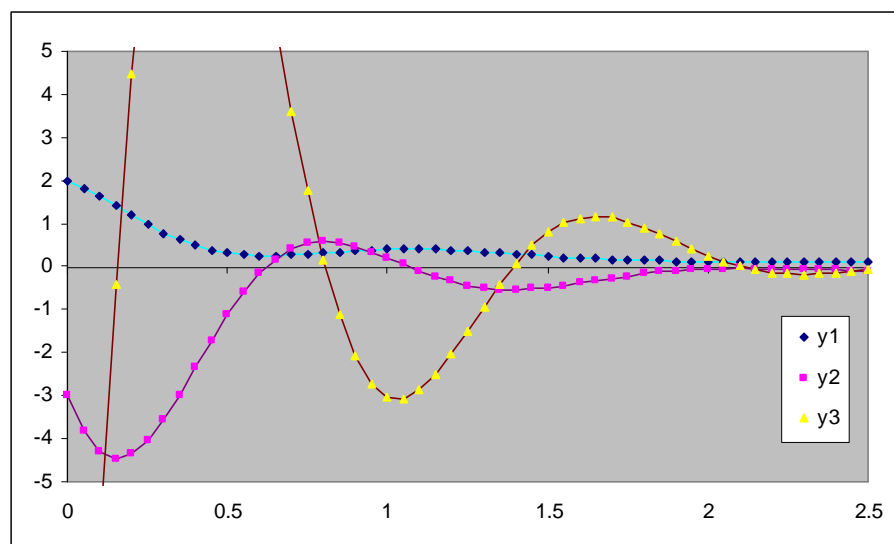
And give the ctrl+shift+enter key sequence.

The selected cells will be automatically filled with the calculated solutions.

Compare with the exact solution

$$y = e^{-x} + e^{-2x} \cos(5x)$$

The following graph shows the exact solutions (continue lines) compared with the calculated solutions (dotted lines). We note a global good fitting.



Vibrating spring-mass system 1

Consider a simple second order differential equation for a vibrating spring mass system.

$$m \frac{d^2 x}{dt^2} + c \frac{dx}{dt} + k x = 0$$

Where x measures the displacement from the equilibrium position, " m " is the mass of the object M , " c " is the damping coefficient and " k " is the spring constant.

Suppose that the object M is pulled down of a the amount x_0 . Starting from the time $t = 0$, the body is released. We want to plot the space position $x(t)$ and its velocity in function of the time.

The initial conditions are $x(0) = x_0$ and $v(0) = 0$. Because is $x'(t) = v(t) \Rightarrow x'(0) = 0$

Therefore the system is solved by the following Cauchy problem

$$\begin{cases} m \frac{d^2 x}{dt^2} + c \frac{dx}{dt} + k x = 0 \\ x(0) = x_0, \quad x'(0) = 0 \end{cases}$$

Rewrite the equation

$$\frac{d^2 x}{dt^2} = -\frac{c}{m} \left(\frac{dx}{dt} \right) - \frac{k}{m} x$$

and being

$$\frac{dx}{dt} = v, \quad \frac{d^2 x}{dt^2} = \frac{dv}{dt}$$

The equation can be written as a system of two first order equations

$$\begin{cases} x' = v \\ v' = -(k/m)x - (c/m)v \end{cases} \quad \text{with} \quad \begin{cases} x(0) = x_0 \\ v(0) = 0 \end{cases}$$

Or, in a compact matrix form, as:

$$\begin{bmatrix} x' \\ v' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(k/m) & -(c/m) \end{bmatrix} \cdot \begin{bmatrix} x \\ v \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} x \\ v \end{bmatrix}_{t=0} = \begin{bmatrix} x_0 \\ 0 \end{bmatrix}$$

Let's see how to solve the problem with $x_0 = 10$ cm , $k = 10$ N/m , $m = 2.5$ Kg and different values of the damping factors: $c = 2, 4, 9$ N·s/m

	A	B	C	D	E	F
1	Parameters			A =		
2	k	10		0	1	
3	c	2		-4	-0.8	
4	m	2.5				
5						
6	h	0.09	={ODE_SYSL(D2:E3;B9:C9;B6)}			
7						
8	t	x	v			
9	0	0.1	0			
10	0.09	0.09842	-0.0345			
11	0.18	0.09389	-0.0656			
12	0.27	0.08674	-0.0925			
13	0.36	0.07738	-0.1146			

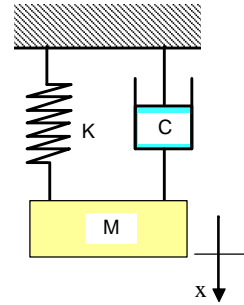
We use the ODE_SYSL function.

To set up the matrix insert the following functions: [D3] = -B2/B4, [E3] = -B3/B4

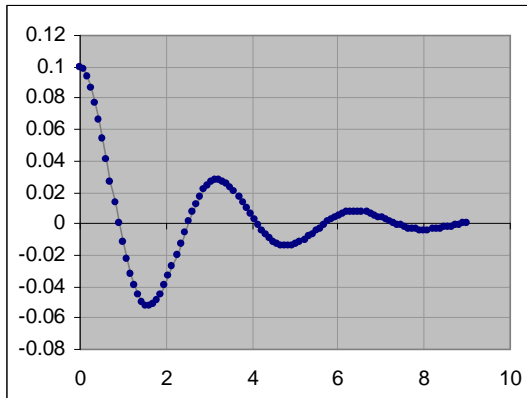
Choose an adapt integration step, for example, $h = 0.09$.

Insert the starting values [0.1 , 0] in the first row B9:C9.

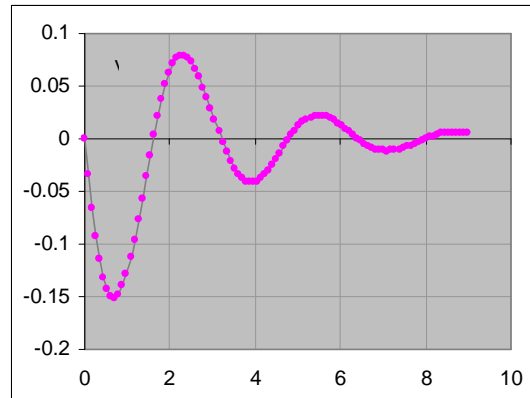
Select the range B10:C109 and insert the function ODE_SYSL with its arguments (see the figure)



Note that ODE_SYSL is an array function and must be inserted with the "ctrl+shift+enter" key sequence.

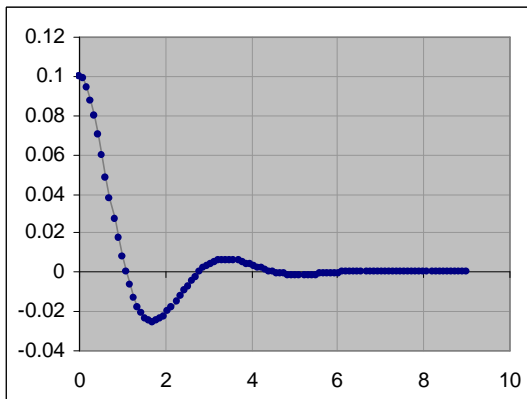


Position $x(t)$ for $c = 2$, $h = 0.09$

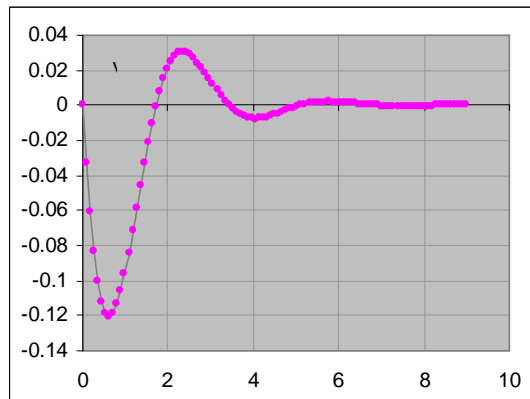


Velocity $v(t)$ for $c = 2$, $h = 0.09$

By changing the parameter "c" in the cell B3 is easy to obtain the following transient functions

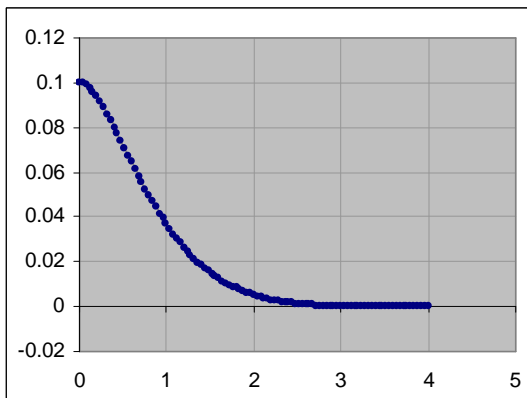


Position $x(t)$ for $c = 4$, $h = 0.09$

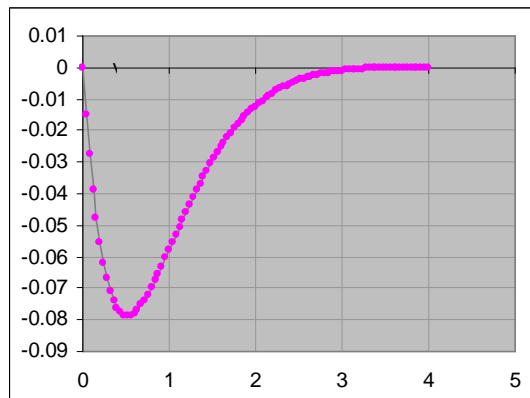


Velocity $v(t)$ for $c = 4$, $h = 0.09$

By changing the parameter "h" in the cell B6 is easy to rescale the transient functions



Position $x(t)$ for $c = 9$, $h = 0.04$



Velocity $v(t)$ for $c = 9$, $h = 0.04$

Now suppose that a constant force F is applied to the object M in equilibrium. We want to plot the space position $x(t)$ and its velocity $v(t)$. The initial condition are $x(0) = 0$ and $v(0) = 0$. The Cauchy problem is now:

$$\begin{cases} x' = v \\ v' = -(k/m)x - (c/m)v + F/m \end{cases} \quad \text{where} \quad \begin{cases} x(0) = 0 \\ v(0) = 0 \end{cases}$$

Or, in its equivalent matrix form:

$$\begin{bmatrix} x' \\ v' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(k/m) & -(c/m) \end{bmatrix} \cdot \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ F/m \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} x \\ v \end{bmatrix}_{t=0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solve the problem with $F = 1 \text{ N}$, $k = 10 \text{ N/m}$, $m = 2.5 \text{ Kg}$ and $c = 3 \text{ N-s/m}$

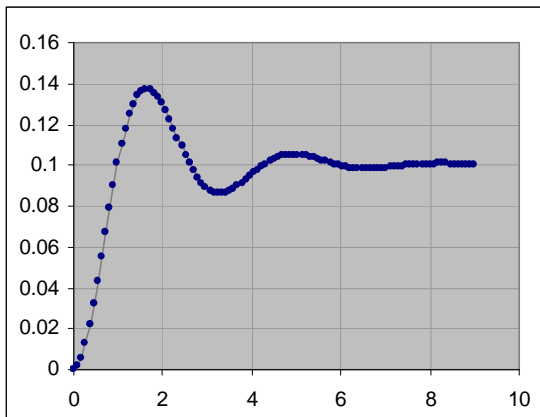
	A	B	C	D	E	F	G
1	Parameters			A			b
2	k	10		0	1		0
3	c	3		-4	-1.2		0.4
4	m	2.5					
5	F	1					
6	h	0.09	{=ODE_SYSL(D2:E3;B9:C9;B6;G2:G3)}				
7							
8	t	x	v				
9	0	0	0				
10	0.09	0.00156	0.03394				
11	0.18	0.00597	0.06337				
12	0.27	0.01282	0.08784				
13	0.36	0.02163	0.10712				

Respect to the previous example we have added the parameter "F" in the cell B5 and the vector "b" in G2:G3, calculated as $[0, B5/B4]$. Choose an adapt integration step, for example, $h = 0.09$.

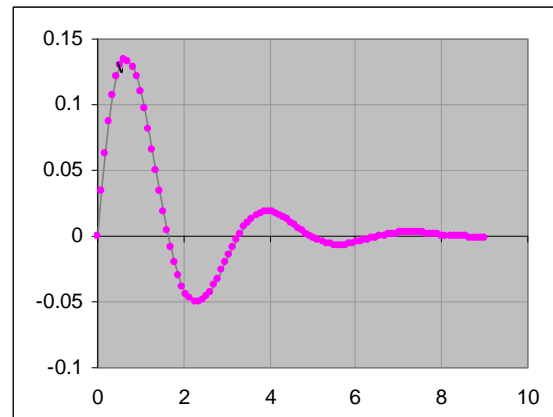
Insert the starting values $[0, 0]$ in the first row B9:C9.

Select the range B10:C109 and insert the function ODE_SYSL with its arguments (see the figure)

The position and velocity transients look like the following.



Position $x(t)$ for $c = 3$, $h = 0.09$



Velocity $v(t)$ for $c = 3$, $h = 0.09$

From the position plot we note that the final equilibrium position is $0.1 \text{ m} = 10 \text{ cm}$ below the "free" unloaded equilibrium position. This can be confirmed writing the static equilibrium equation

$$F = k \cdot x_{\infty} \Rightarrow x_{\infty} = F / k$$

where we the symbol x_{∞} means $x(\infty)$. Substituting we have $x_{\infty} = 0.1 \text{ m}$

The "settling" time, thus the time where the position is set under the 5% of the final position is about 6 sec.

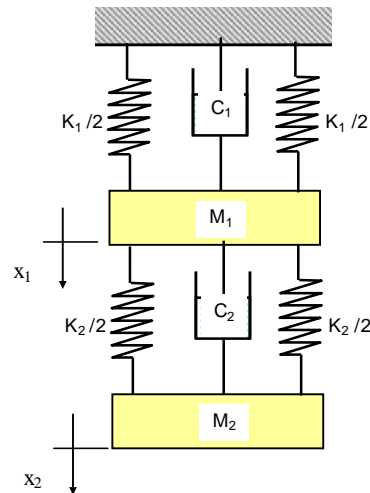
Vibrating spring-mass system 2

Consider a system of second order differential equations for a double vibrating spring mass system.

$$m_1 \frac{d^2 x_1}{dt^2} + c_1 \frac{dx_1}{dt} + k_1 x_1 = 0$$

$$m_2 \frac{d^2 x_2}{dt^2} + c_2 \left(\frac{dx_2}{dt} - \frac{dx_1}{dt} \right) + k_2 (x_2 - x_1) = 0$$

Where x_1 and x_2 measure the displacements from the equilibrium positions; " m_1 " and " m_2 " are the mass of the objects M_1, M_2 ; " c_1 " and " c_2 " are the damping coefficients; " k_1 " and " k_2 " are the springs constants.



The problem is equivalent of 4 first order ODEs with four variables and initial conditions, that can be easily written in matrix form

$$\frac{dx_1}{dt} = v_1$$

$$\frac{dv_1}{dt} = - \left[\frac{c_1}{m_1} v_1 + \frac{k_1}{m_1} x_1 \right]$$

$$\frac{dx_2}{dt} = v_2$$

$$\frac{dv_2}{dt} = - \left[\frac{c_2}{m_2} (v_2 - v_1) + \frac{k_2}{m_2} (x_2 - x_1) \right]$$

$$\begin{bmatrix} x_1' \\ v_1' \\ x_2' \\ v_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_1}{m_1} & -\frac{c_1}{m_1} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_2}{m_2} & \frac{c_2}{m_2} & -\frac{k_2}{m_2} & -\frac{c_2}{m_2} \end{bmatrix} \begin{bmatrix} x_1 \\ v_1 \\ x_2 \\ v_2 \end{bmatrix}$$

Now let's perturb the system pulling down the object M_2 of the amount $x_0 = 0.1$ m. Starting from the time $t = 0$, the system is released and we want to plot the space position $x_1(t)$, $x_2(t)$ and their velocities $v_1(t)$ and $v_2(t)$. The initial conditions are $x_1(0) = x_0$, $v_1(0) = 0$, $x_2(0) = x_0 \cdot k_1 / (k_1 + k_2)$ and $v_2(0) = 0$

	A	B	C	D	E	F	G
1							
2	k1	10		0	1	0	0
3	c1	2		-10	-2	0	0
4	m1	1		0	0	0	1
5	k2	10		4	2.4	-4	-2.4
6	c2	6					
7	m2	2.5					
8							
9	h	0.04	{=ODE_SYSL(D2:G5;B12:E12;B9)}				
10							
11	t	x1	v1	x2	v2		
12	0	0.05	0	0.1	0		
13	0.04	0.04961	-0.0192	0.09983	-0.0085		
14	0.08	0.04849	-0.0366	0.0993	-0.018		
15	0.12	0.04671	-0.0521	0.09838	-0.0283		

The matrix is built with the following functions.

$$[D3] = -B2/B4$$

$$[E3] = -B3/B4$$

$$[D5] = B5/B7$$

$$[E5] = B6/B7$$

$$[F5] = -B5/B7$$

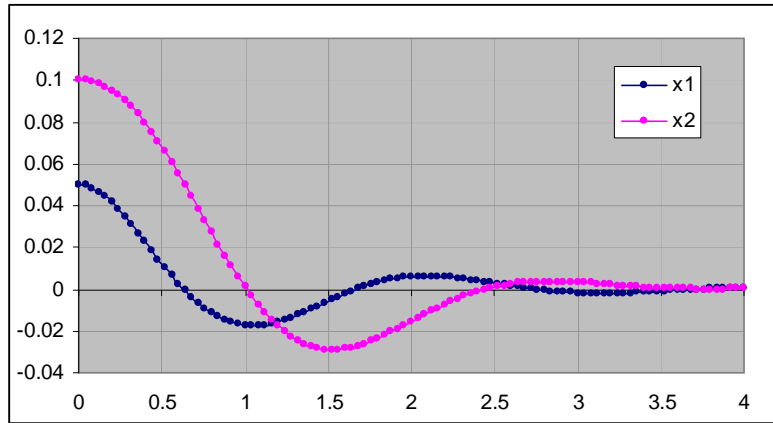
$$[G5] = -B6/B7$$

Calculate the starting value $x_1(0)$ by the formula $=D12*B5/(B2+B5)$

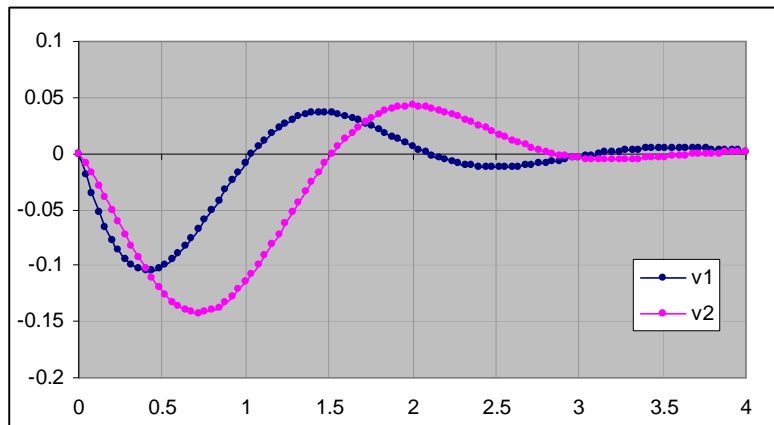
Insert the starting values $[0, 0.1, 0]$ in the range C12:E12.

Choose an adapt integration step, for example, $h = 0.04$. Select the range B13:D112 and insert the function ODE_SYSL with its arguments (see the figure)

The transient of the positions $x_1(t)$ and $x_2(t)$ are shown in the following plot



and their correspondent velocities $v_1(t)$ and $v_2(t)$ are shown in the following plot



Note how quick and easy is the solution of this complicated differential system. We can simulate the complete transient simply changing one or more parameters of the given mechanical system.

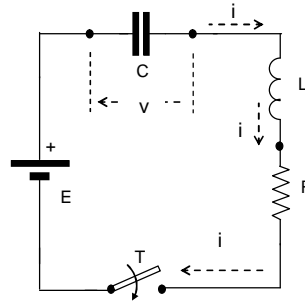
Electric Circuit 1

Consider the one mesh circuit of the figure.
At same arbitrary time, i.e. $t = 0$, the switch is closed.
We want to evaluate numerically the transient of the current $i(t)$ flowing into the coil and the voltage drop $v(t)$ across the capacitor.

Parameters are:

$R1 = 10 \text{ ohm}$, $L1 = 1.5 \text{ H}$, $C = 100 \text{ uF}$

$E = 2 \text{ V}$



By mesh analysis we write the equations of the voltage drops across the circuit

$$\begin{cases} \frac{dv}{dt} = \frac{1}{C}i \\ v + L\frac{di}{dt} + Ri = E \end{cases} \quad \text{That system can be rearranged as} \quad \begin{cases} \frac{dv}{dt} = \frac{1}{C}i \\ \frac{di}{dt} = -\frac{1}{L}v - \frac{R}{L}i + \frac{E}{L} \end{cases}$$

That can be written in matrix form as

$$\begin{bmatrix} v' \\ i' \end{bmatrix} = \begin{bmatrix} 0 & 1/C \\ -1/L & -R/L \end{bmatrix} \begin{bmatrix} v \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ E/L \end{bmatrix}$$

A simple worksheet arrangement may be the following

	A	B	C	D	E
1	R	10	ohm		
2	C	100	uF		
3	L	1.5	H		
4	E	2	V		
5	h	0.002			
6					
7	A=			b	
8		0	10000		0
9		-0.6667	-6.6667		1.3333
10					
11		={ODE_SYSL(A8:B9,C14:D14,B11,D8:D9)}			
12					
13	t		v(t)	i(t)	
14		0	0	0	
15		0.002	0.0265	0.0026	
16		0.004	0.1048	0.0052	
17		0.006	0.2321	0.0075	
18		0.008	0.4045	0.0097	
19		0.01	0.6168	0.0115	
20		0.012	0.8628	0.013	
21		0.014	1.1357	0.0142	

Insert the following functions into the relative cells.

[B8] = 1/(B2*10⁻⁶)

[A9] = - 1/B3

[B9] = -B1/B3

[D9] = B4/B3

Insert the starting value [0, 0] in the range B13:C13, and choose a suitable step (h = 0.002)

Select the range C15:D177 and insert the function ODE_SYSL with its parameters.

The exact solutions are

$$i(t) = \frac{E}{wL} e^{-s t} \sin(w t)$$

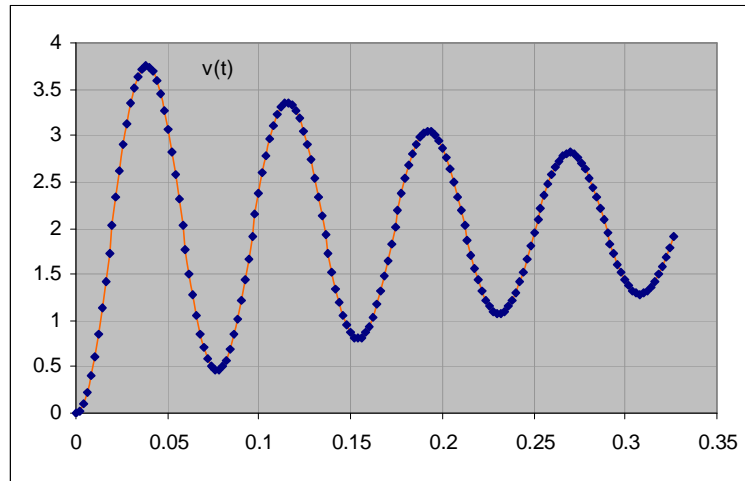
$$v(t) = \frac{E}{LC(w^2 + s^2)} \left[1 - e^{-s t} \left(\cos(w t) + \frac{s}{w} \sin(w t) \right) \right]$$

where

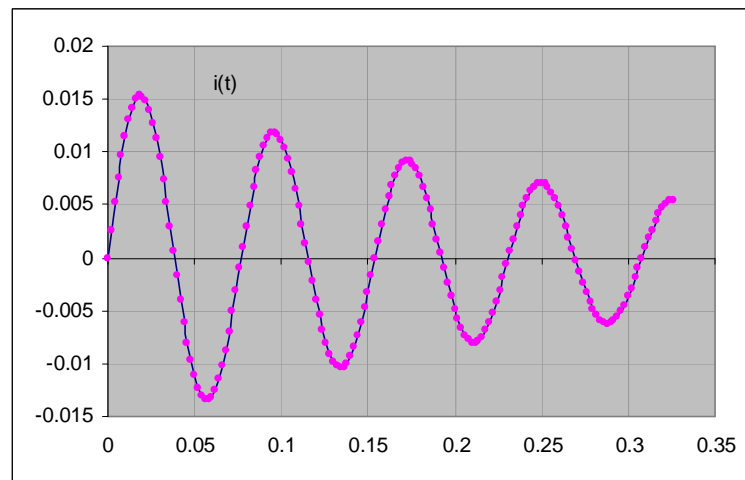
$$w = \frac{1}{2} \sqrt{\frac{4 \cdot 10^6}{LC} - \left(\frac{R}{L}\right)^2} \quad \text{and} \quad s = \frac{R}{2L}$$

The curves obtained (dotted line) are shown in the following graphs together with the exact solutions (continue line)

The voltage $v(t)$



The current $i(t)$



You can verify that the global relative error is better than $1E-10$ in the range $0 \leq t \leq 0.35$ (164 points)

Electric Circuit 2

Consider the coupled coils of the figure.

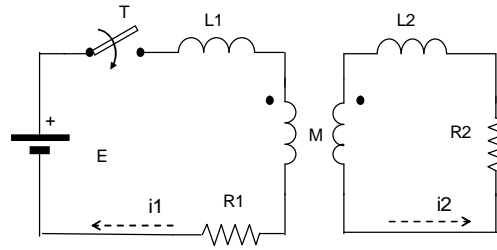
At same arbitrary time, which we shall call $t = 0$, the switch is closed. We want to evaluate numerically the currents transient of the circuit.

Parameters are:

$L_1 = 1 \text{ H}$, $L_2 = 2 \text{ H}$, $M = 1 \text{ H}$,

$R_1 = 4 \text{ ohm}$, $R_2 = 6 \text{ ohm}$,

$E = 1 \text{ V}$



By mesh analysis we write the system of the voltage drops across the coils

$$\begin{cases} L_1 \frac{di_1}{dt} + R_1 i_1 + M \frac{di_2}{dt} = E \\ L_2 \frac{di_2}{dt} + R_2 i_2 + M \frac{di_1}{dt} = 0 \end{cases} \quad \text{That system can be rearranged as} \quad \begin{cases} \frac{di_1}{dt} = \frac{-R_1 L_2 i_1 + R_2 M i_2 + E L_2}{L_1 L_2 - M} \\ \frac{di_2}{dt} = \frac{R_1 M i_1 - R_2 L_1 i_2 - E M}{L_1 L_2 - M} \end{cases}$$

That can be written in matrix form as

$$\begin{bmatrix} \dot{i}_1 \\ \dot{i}_2 \end{bmatrix} = \begin{bmatrix} -R_1 L_2 / \Delta & R_2 M / \Delta \\ R_1 M / \Delta & -R_2 L_1 / \Delta \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} + E \begin{bmatrix} L_2 / \Delta \\ -M / \Delta \end{bmatrix}$$

where $\Delta = L_1 L_2 - M^2$

A simple worksheet arrangement may be the following

	A	B	C	D	E	F	G	H
2	Parameters			$\Delta =$	1			
3	L1	1						
4	L2	2		-8	6		2	
5	M	1		4	-6		-1	
6	R1	4						
7	R2	6						
8	E	1						
9								
10	h	0.02	={ODE_SYSL(D4:E5;B13:C13;B10;G4:G5)}					
11								
12	t	i1	i2					
13	0	0	0					
14	0.02	0.0359	-0.0174					
15	0.04	0.0649	-0.0304					
16	0.06	0.0883	-0.04					
17	0.08	0.1074	-0.0469					

Insert the following functions into the relative cells.

[E2] = -B3*B4-B5^2

[D4] = -B6*B4/E2, [E4] = B7*B5/E2

[D5] = B6*B5/E2, [E5] = -B7*B3/E2

[G4] = B8*B4/E2

[G5] = -B8*B5/E2

Insert the starting value [0, 0] in the range B13:C13, and choose a suitable step (h = 0.02)

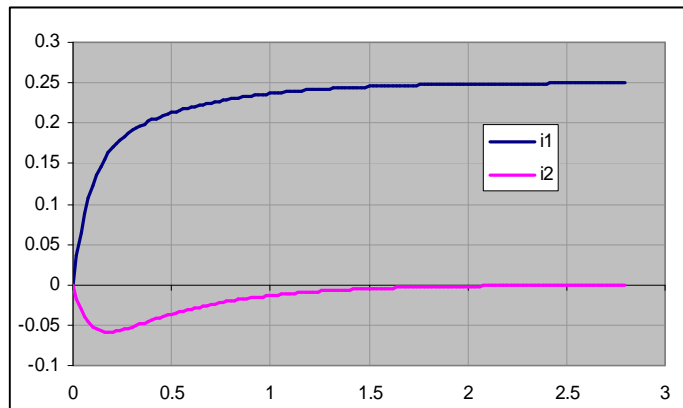
Select the range B14:C153 and insert the function ODE_SYSL with its parameters.

The curves obtained are shown in the following graph.

Compare with the exact solution

$$i_1(t) = 0.25 - 0.1 e^{-2t} - 0.15 e^{-12t}$$

$$i_2(t) = -0.1(e^{-2t} - e^{-12t})$$



Elastic Beam 1

Another linear ODE that has important applications in materials science is that for the deflection of a beam.

$$\frac{d^2}{dx^2} \left(EI \frac{d^2 y}{dx^2} \right) = w(x)$$

where: y is the beam deflection, $w(x)$ is the load density (force per unit length of beam); E is Young's modulus of elasticity for the beam and I is the moment of inertia of the cross section of the beam. If the moment of inertia and the Young's modulus do not depend on the position in the beam (the case for a uniform beam of homogeneous material), then the beam equation becomes:

$$EI \frac{d^4 y}{dx^4} = w(x)$$

Substituting the variables $y_1 = y'$, $y_2 = y''$, $y_3 = y'''$ this 4th order equation can be transformed in a 1st order equivalent system.

$$\begin{cases} y_1' = y_1 \\ y_1' = y_2 \\ y_2' = y_3 \\ y_3' = w/EI \end{cases} \quad \begin{bmatrix} y' \\ y_1' \\ y_2' \\ y_3' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} y' \\ y_1' \\ y_2' \\ y_3' \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ w/EI \end{bmatrix}$$

Unfortunately the system matrix, in this case, has a row and a column both null and, thus, is singular.

We cannot use it with the ODE_SYSL function. We should use ODE_RK4 or ODE_PC4.

Before attempting to solve the ODE is convenient to transform further the given system in a better useful form using the know relations:

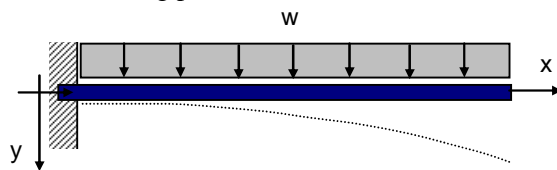
$$\begin{array}{ll} s = y' & \text{Slope} \\ M = EI y'' & \text{Bending Moment}^2 \\ T = M' = EI y''' & \text{Shear Force} \end{array}$$

Substituting, we have a new differential system in the variable $y(x)$, $s(x)$, $M(x)$, $T(x)$

$$\begin{cases} y' = s \\ s' = M/EI \\ M' = T \\ T' = w \end{cases} \quad \text{with the initial conditions} \quad \begin{cases} y(0) = y_0 \\ s(0) = s_0 \\ M(0) = M_0 \\ T(0) = T_0 \end{cases}$$

The shape of a loaded beam is determined by the loads applied over its length and its boundary conditions. The boundary conditions can be determined because each derivative of y has a specific meaning: the beam curvature is related to the local moment M ; shear forces T are related to the rate of change of moment along the beam. Usually they can be calculated using static equilibrium conditions.

Let's see how it works with the following problem



² Many authors conventionally define $M = -EI y''$ but the concept is the same. Of course the graphs M and T have the sign inverted

The beam has length $L = 1$; it is loaded with a constant load density $w = 1$, and its product $EI = 10$. First of all we calculate the initial conditions. They can be easily derived from the following equilibrium equations: $T_0 + wL = 0 \Rightarrow T_0 = -wL$ and $M_0 - wL^2/2 = 0 \Rightarrow M_0 = wL^2/2$

A possible worksheet arrangement is:

	A	B	C	D	E	F
1	h	w	L	EI		$y' = s$
2	0.05	1	1	10		$s' = M / EI$
3						$M' = T$
4	={ODE_PC4(F1:F4,A7:E7,A2,B2,D2)}					$T' = w$
5						
6	x	y	s	M	T	
7	0	0	0	0.5	-1	
8	0.05	6E-05	0.00238	0.45125	-0.95	
9	0.1	0.00023	0.00452	0.405	-0.9	
10	0.15	0.00051	0.00643	0.36125	-0.85	
11	0.2	0.00087	0.00813	0.32	-0.8	
12	0.25	0.00132	0.00964	0.28125	-0.75	

In the cell D7 and E7 insert the following formulas:

[D7] = $B2 * C2^2 / 2$

[E7] = $-B2 * C2$

Insert the ODE equations in the cells F1:F4

$y' = s$

$s' = M / EI$

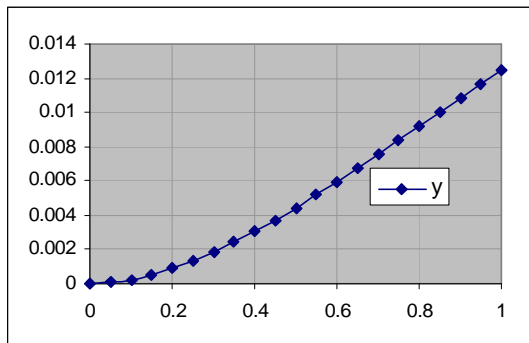
$M' = T$

$T' = w$

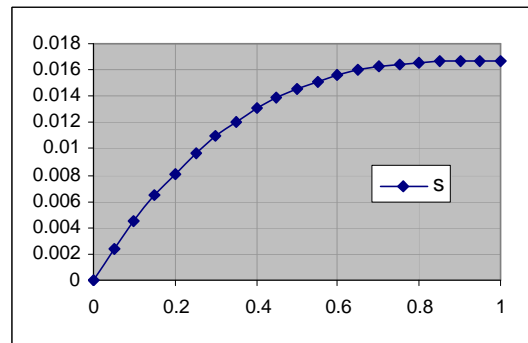
Select the range A8:E27, insert the function ODE_PC4 (or ODE_RK4) giving its parameters and press ctrl+shift+enter

Remark. The labels "w", "EI", "x", "y", "s", "M", "T" inserted in the cells B1:D1 and A7:E7 must exactly match with the variables inserted in the ODE equations.

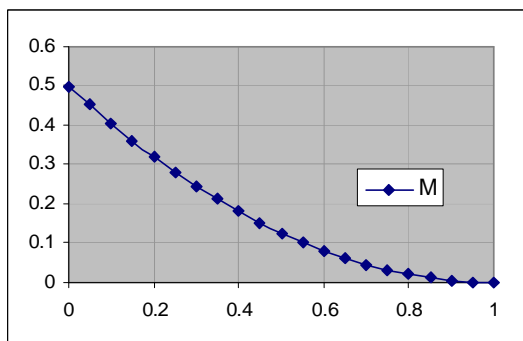
As we can see, all the most important functions are returned directly from the ODE integration, and their plots are the following



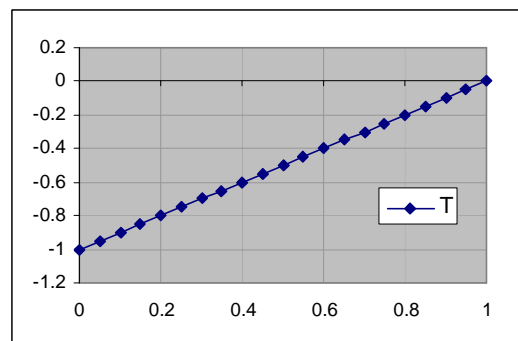
Beam deflection



Slope of the beam deflection



Bending Moment



Shear Force

Non-Linear ODE

The Cardiac Cell Electric model

The predictor-corrector schema can be applied to small non linear smooth system. A suitable such system is the *FitzHugh-Nagumo* equations, a very simple model for the electrical activity in cardiac cells. For simplicity we consider a normalized system, given by:

$$\begin{cases} v' = v(1-v)(v-1/2) - w \\ w' = v - w \end{cases}$$

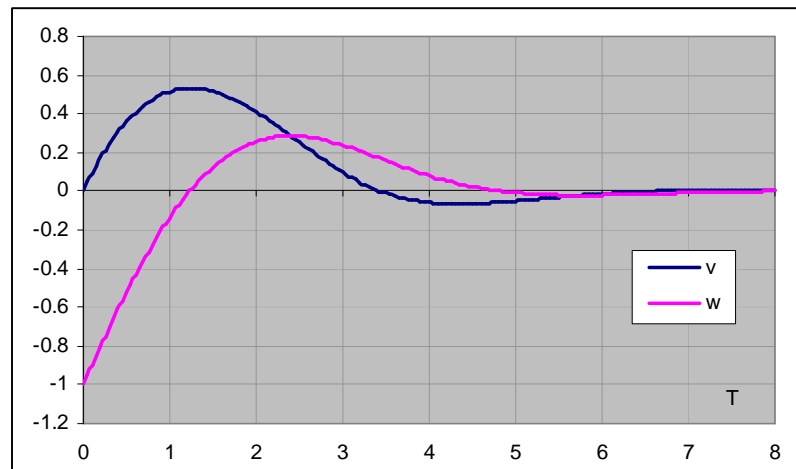
with initial conditions $v(0) = 0$ and $w(0) = -1$

	A	B	C	D
1				
2	$v' = v*(1-v)*(v-1/2) - w$			h
3	$w' = v - w$			0.05
4				
5	{=ODE_PC4(A2:A3;A7:C7;D3)}			
6	t	v	w	
7	0	0	-1	
8	0.05	0.048198	-0.95003	
9	0.1	0.093051	-0.90023	
10	0.15	0.134881	-0.85075	
11	0.2	0.17394	-0.80171	
12	0.25	0.210426	-0.75322	

Insert the equations definitions in the cells A2 and A3; the step h in the cell D3 and the initial values in the range A7:C7. Do not miss the labels "t", "v", "w" just above the cells A7:C7

Select the range A8:C160 and insert the ODE_PC4 (the 4th order Adams-Multon-Bashforth integrator) by the ctrl-shift-enter keys sequence.

The plot of the solution $v(t)$ and $w(t)$ for $0 \leq t \leq 8$ is shown in the following figure



Rigid body equations

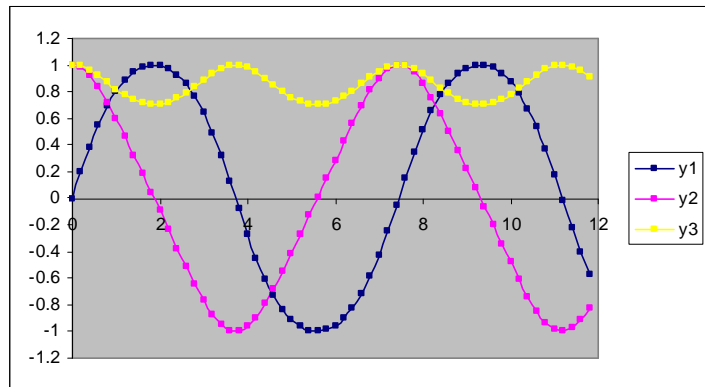
This example illustrates the solution of a standard test problem proposed by Krogh for solvers intended for nonstiff problems.

The ODEs are the Euler equations of a rigid body without external forces

$$\begin{cases} y_1' = y_2 y_3 \\ y_2' = -y_1 y_3 \\ y_3' = -0.51 y_1 y_2 \end{cases} \quad \begin{cases} y_1(0) = 0 \\ y_2(0) = 1 \\ y_3(0) = 1 \end{cases}$$

The solution in Excel is very quick.

	A	B	C	D
1	$y_1' = y_2 y_3$			h
2	$y_2' = -y_1 y_3$			0.2
3	$y_3' = -0.51 y_1 y_2$			
4	={ODE_PC4(A1:A3,A7:D7,D2)}			
5				
6	x	y1	y2	y3
7	0	0	1	1
8	0.2	0.198	0.9802	0.98995
9	0.4	0.38457	0.92309	0.96155
10	0.6	0.55054	0.8348	0.91946
11	0.8	0.69038	0.72338	0.86997
12	1	0.80212	0.59698	0.81958
13	1.2	0.88658	0.46223	0.7739
14	1.4	0.94597	0.32368	0.73715



we may also use the ODE_RK4 for solving this ODE system. The differences between the values returned from the two functions are less than 0.1%

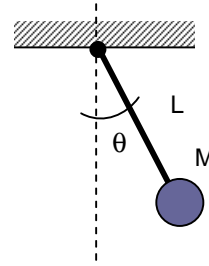
Note that we have obtained all the 59 points in a single step selecting the range A8:D66 and inserting ODE_PC4 with the CTRL+SHIFT+ENTER sequence. This trick improves the efficiency but on the other hand it obligates to choose in advance all the integration points.

More flexible (but less efficient) is the following method step-by-step

Select the range A8:D8 and insert ODE_PC4 giving the correct parameters. Remember to fix the range A1:A3 and the cell D2 by the character \$					Drag the selection down in order to get all the integration points that you like.						
1	A	B	C	D	E	1	A	B	C	D	E
2	$y_1' = y_2 y_3$			h		2	$y_1' = y_2 y_3$			h	
3	$y_2' = -y_1 y_3$			0.2		3	$y_2' = -y_1 y_3$			0.2	
4	$y_3' = -0.51 y_1 y_2$					4	$y_3' = -0.51 y_1 y_2$				
5	={ODE_PC4(\$A\$1:\$A\$3,A7:D7,\$D\$2)}					5	={ODE_PC4(\$A\$1:\$A\$3,A7:D7,\$D\$2)}				
6	x	y1	y2	y3		6	x	y1	y2	y3	
7	0	0	1	1		7	0	0	1	1	
8	0.2	0.198	0.9802	0.98995		8	0.2	0.198	0.9802	0.98995	
9						9	0.4	0.38457	0.92309	0.96155	
10						10	0.6	0.55054	0.8348	0.91946	
11						11	0.8	0.69039	0.72343	0.87	
12						12					

Pendulum

The pendulum shown consists of a ball with concentrated mass, M attached to a rod of length L whose mass is assumed to be negligible with respect to that of the ball. The governing equation for this dynamical system is given by the following 2nd order, non-linear ODE for the displacement angle θ .



$$q'' + \frac{g}{L} \sin q = 0$$

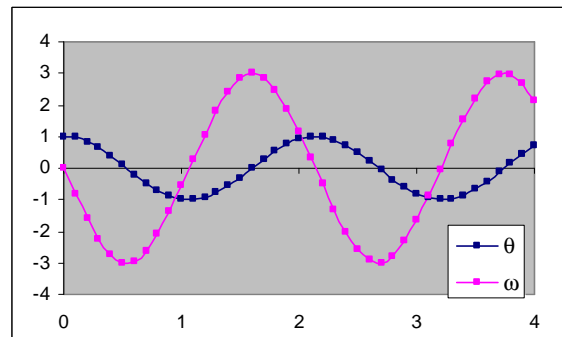
where g is the acceleration due to gravity ($g \cong 9.81 \text{ m/s}^2$ near the ground)
This equation can be transformed in the following equivalent system

$$\begin{cases} q' = w \\ w' = -\frac{g}{L} \sin q \end{cases}$$

where ω is the angular acceleration.

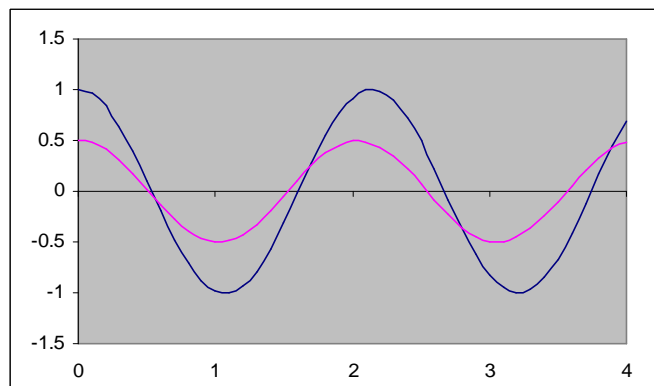
We can solve the problem setting the parameter $L = 1 \text{ m}$ and the starting position $\theta_0 = 0.5$
A simple arrangement is

	A	B	C	D	E
1	$\theta' = \omega$		g	L	h
2	$\omega' = -g/L * \sin(\theta)$		9.81	1	0.1
3					
4	={ODE_PC4(A1:A2,A6:C6,E2,C2,D2)}				
5	t	θ	ω		
6	0	0.5	0		
7	0.1	0.476653	-0.463536		
8	0.2	0.408638	-0.886742		
9	0.3	0.301966	-1.2304513		
10	0.4	0.18635	-1.4598976		



Now we solve the problem with a different starting position $\theta_0 = 1$. It can simply be done by changing the value in the cell B6

Plotting the two functions $\theta(t)$ in the same graph, we observe that the same pendulum oscillates with different periods for different amplitudes. This illustrates an important property inherent to non-linear differential equations: the free response of a linear equation has the same period for any initial conditions, while in contrast, the form of the free response of a non-linear ODE often depends on the particular values of the initial conditions.



Solutions Family

In the study of a differential problem, for example a Cauchy problem, we shall encounter the following important sub-problems: the existence and the uniqueness of the problem itself; the continuation of the solution, the study of the behavior of the solutions as x approaches to infinity, etc.

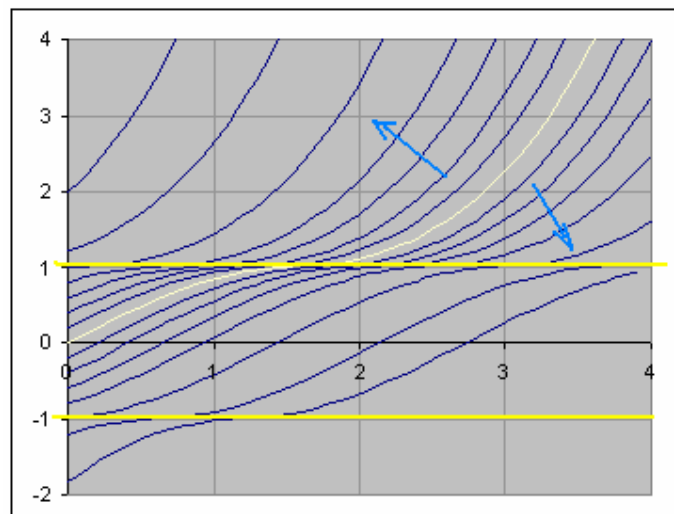
All these problems needs of functional analysis instead of numerical computing. But, often, the numerical approach gives us good feedback and valuable hits for the problem comprehension³

Example 1: study the following differential equation $y' = \sqrt{|y^2 - 1|}$

	A	B	C	D
1	$y' = \text{sqr}(y^2-1)$		x	y
2			0	-2
3	h	0.1	0.1	-1.836515
4			0.2	-1.69141
5	={ODE_RK4(A1,C2:D2,B3)}		0.3	-1.563234
6			0.4	-1.450702

Insert the equation in A1 and choose a reasonable step, for example 0.1. Add the labels "x", "y" at the cell C1 and D1 just above the starting values. Select the range A3:D40 for 40 integration points and insert ODE_RK4. The solution [xi, yi] appears in the selected cells.

The solution can be quickly changed by changing the cell D2. Choosing a set of values from -2 to 2 with step 0.2 we can draw the following family solution



The blank curve acrosses the zero, i.e., $y(0) = 0$. the curves above are obtained for $y(0) > 0$ while the curve below for $y(0) < 0$. Note that that for $y(0) = \pm 1$ the solutions became constant, $y = \pm 1$

Compare with the exact solution for $y(0) = 0$

$$y = \begin{cases} -\cosh(x+p/2) & x < -p/2 \\ \sin x & -p/2 \leq x \leq p/2 \\ \cosh(x-p/2) & x > p/2 \end{cases}$$

Example 2. study the differential equation $y' = y - x^2 y^2$ for $x > 0$ and $0 \leq y(0) \leq 1$

This is a Bernoulli's equation.

For $y(0) = 0$ the solution is $y = 0$, as we can easily verify by substituting.

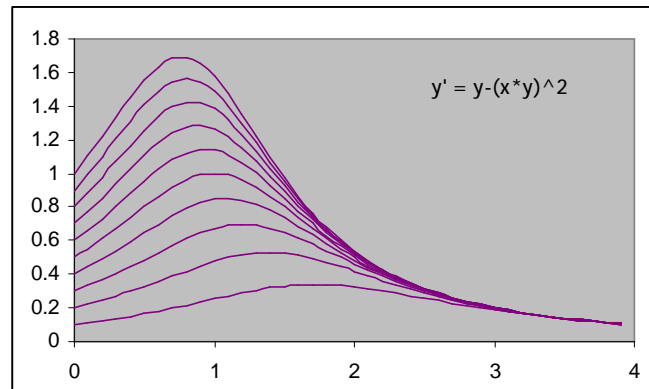
³ see also the Macro Slope Grid

For $y(0) > 0$ we can numerically solve the equation choosing the initial values $y(0) = 0.1, 0.2, 0.3 \dots 1$.

	A	B
1	$y' = y - x^2 * y^2$	
2		
3	h	0.1
4		
5	x	y
6	0	1
7	0.1	1.104774
8	0.2	1.217627
9	0.3	1.334789
10	0.4	1.449947
11	0.5	1.554074
12	0.6	1.636155
13		

Insert the equation $y' = y - x^2 * y^2$ in A1 and choose a suitable step, for example $h = 0.1$.
 Now insert the label "x", "y" in A5 and B5 and insert the starting values in the cells A6 and B6 just below, for example [0, 1]
 Select the range A7:B7 and insert the formula `=ODE_RK4(A1,A6:B6,B3)` with ctrl+shift+enter key.
 Drag the selection A:B7 down until you reach the wanted range (for example $0 \leq x \leq 4$).
 Now change the starting cell B2 from 1 to 0.9, 0.8 till 0.1, in order to have a representative set of the solutions family.

Plotting the solutions set, we have the following graph



Compare with the exact solution.

$$y(x) = \frac{1}{x^2 - 2x + 2 + (1/y_0 - 2)e^{-x}}$$

Now try to extend the solutions set for $x < 0$. Re-insert the value $y = 1$ in the cell B6. and insert the step $h = -0.1$ in the cell B3.

This simple trick inverts back the direction of integration: from 0 to -0.1, -0.2 ... -4 .

Overflow. This example shows an interesting thing. Many time the solution grows so large that can easily overcome the limit of the standard arithmetic. In this example, with $y_0 = 1$, probably you will get an overflow error "?" at the step $x = -2.8$ and consequently the errors #VALUE! in the cells below.

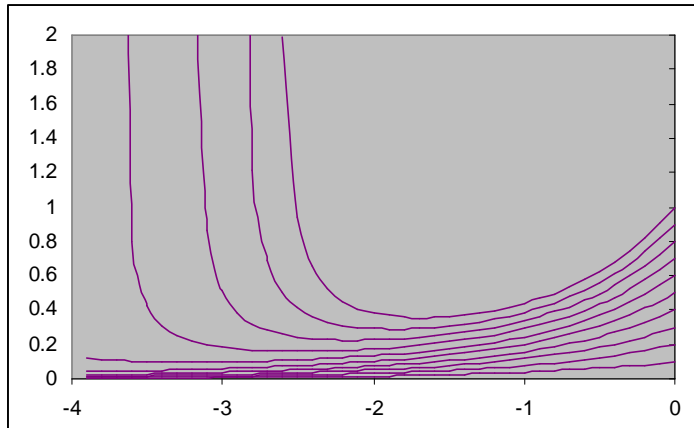
32	-2.6	1.988426
33	-2.7	72.868837
34	-2.8	3.984E+23
35	?	?
36	#VALUE!	#VALUE!
37	#VALUE!	#VALUE!

Note that just before the overflow interruption, the value of y is more than $1E+23$.

This is very common for mathematical and didactical ODE. For particular value of x the solution can blow-up to infinity.

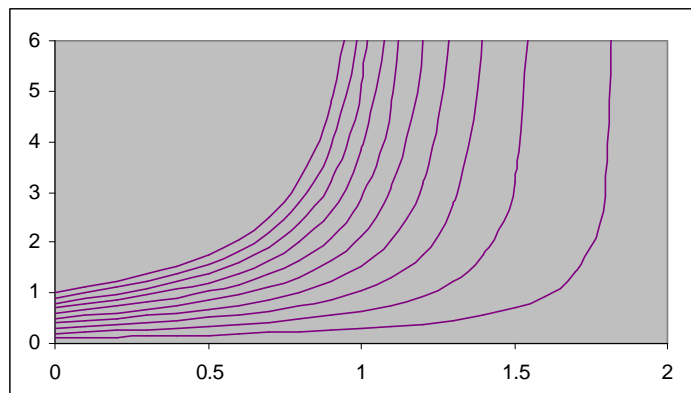
Anyway the steps calculated before the overflow are substantially correct and we can use them for plotting. Note that, instead of inserting ODE_RK4 in the entire range A7:B45, we have inserted the integration function step-by step. This mode is less efficient but it allows to us to discharge the steps that are in overflow error.

Putting together, we have the final plot also for the negative x -axis

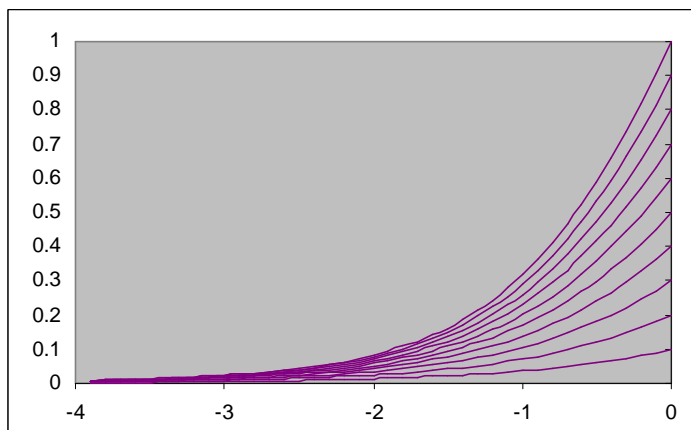


Example. study the differential equation $y' = y + x^2 y^2$ for $-4 < x < 2$ and $0 \leq y(0) \leq 1$
 This is still a Bernoulli's equation very similar to the previous one.
 Using the same method we can build the following graphs

Solution set for $x > 0$ and $y(0) = 0.1, 0.2, 0.3, \dots, 1$



Solution set for $x < 0$ and $y(0) = 0.1, 0.2, 0.3, \dots, 1$



Macro ODE Solver

This macro performs the numerical integration of ordinary differential equations systems (ODE) using the variable step *Runge-Kutta-Fehlberg* method.

Runge-Kutta-Fehlberg

The Runge-Kutta-Fehlberg method is an adaptive procedure for approximating the solution of the differential equation $y'(x) = f(x,y)$ with initial condition $y(x_0) = c$. This implementation evaluates $f(x,y)$ six times per step using embedded fourth order and fifth order Runge-Kutta estimates to estimate the not only the solution but also the error. .

$$y[i+1] = y[i] + h * (25 / 216 * k1 + 1408 / 2565 * k3 + 2197 / 4104 * k4 - 1 / 5 * k5)$$

where

$$k1 = f(x[i],y[i]),$$

$$k2 = f(x[i]+h/4, y[i] + h*k1/4),$$

$$k3 = f(x[i]+3h/8, y[i]+h*(3/32 k1 + 9/32 k2)),$$

$$k4 = f(x[i]+12h/13, y[i]+h*(1932/2197 k1 - 7200/2197 k2 + 7296/2197 k3)),$$

$$k5 = f(x[i]+h, y[i]+h*(439/216 k1 - 8 k2 + 3680/513 k3 - 845/4104 k4))$$

$$k6 = f(x[i]+h/2, y[i]+h*(-8/27 k1 + 2 k2 - 3544/2565 k3 + 1859/4104 k4 - 11/40 k5))$$

$$x[i+1] = x[i] + h.$$

The error is estimated to be

$$err = h*(k1 / 360 - 128 k3 / 4275 - 2197 k4 / 75240 + k5 / 50 + 2 k6 / 55)$$

The step size h is then scaled by the scale factor

$$scale = 0.8 * | epsilon * y[i] / [err * (xmax - x[0])] | ^{(1/4)}$$

The scale factor is further constrained $0.125 < scale < 4.0$.

The next step size is then calculated using the preassigned tolerance and error estimate

$$h := scale * h.$$

The macro ODE solver implements this algorithm for solving ODE with initial values (Cauchy problem)

$$y' = f(x, y) \quad , \quad y(x_0) = y_0 \quad x_0 \leq x \leq x_{\max}$$

as well an ODE system

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \quad , \quad \mathbf{y}(x_0) = \mathbf{y}_0 \quad x_0 \leq x \leq x_{\max}$$

where $\mathbf{y} = [y_1, y_2, \dots, y_n]^T$ is the vector of depend variables

The macro can work with equations inserted directly in the worksheet as any other Excel function or, alternatively, with equations defined by symbolic strings.

The macro implements a control of the local error and changes the integration step consequently Therefore the precision can be set independently from the tabulating step. The independence of the integration step from the tabulation step is the main feature of this macro.

In order to save time, the macro implement a predefined simple input schema

Let's see how to use it with a simple example

Example 1. We want to solve the problem $y' = -6x^5 y^2$, $y(0) = 1$, for $0 \leq x \leq 2.5$ and we want to tabulate the solution $y(x_i)$ with 50 equispaced nodes (tabulating step = 0.05). In addition we would to obtain each nodes with high precision of about 1E-6
Let's prepare the following simple worksheet arrangement

	A	B	C	D
4	x	y	y'	
5	0	1	0	
6	0.05			
7	0.1			
8	0.15			
9	0.2			
10	0.25			
11	0.3			
12	0.35			

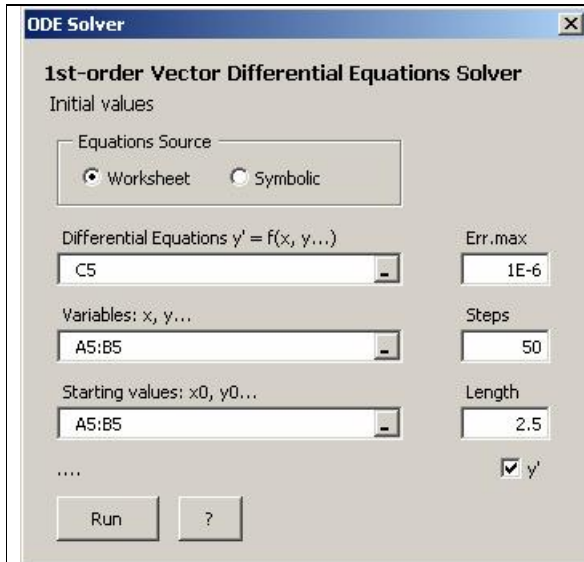
=-6*A5^5*B5^2

In A column, range A5:A55 we have set the tabulation grid with step 0.05.

In B column we put the solution $y(x)$ and in column C the derivative $y'(x)$. Just simple and straight.

In the range A5:B5 we insert the starting values (0, 1) and, finally, the function $=-6*A5^5*B5^2$ in the cell C5

Now select the all range A4:C55 and start the macro



Differential Equations links to the cell where the equations $Y' = f(x, y)$ is inserted.

Variables are the cells (x, y) referenced by the equation formula of the cell C5

Starting values provides the starting values of the differential problem and must point to the cells at the beginning of the table (they may or not may coincide with the variables cells).

If we have followed the above schema, the input fields are correctly filled.

We have only to set the precision

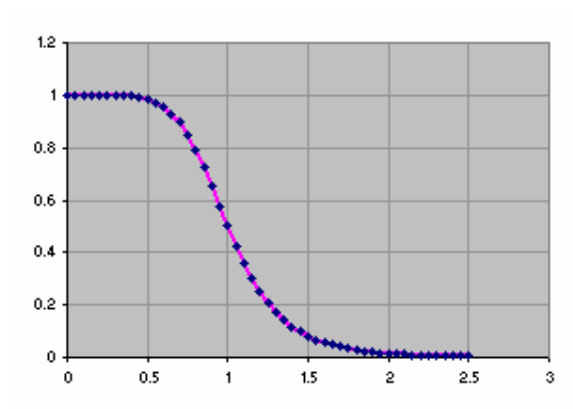
Err. Max = 1E-6

Optionally, if we want the macro to fill also the derivative column, activate the check box



The macro output the result y and, optionally, the derivative y' just below the starting values
Press "Run" for starting the ODE solver

	A	B	C
4	x	y	y'
5	0	1	0
6	0.05	1	-1.875E-06
7	0.1	0.999999	-6E-05
8	0.15	0.9999886	-0.00045561
9	0.2	0.999936	-0.00191975
10	0.25	0.9997559	-0.00585652
11	0.3	0.9992715	-0.01455877
12	0.35	0.9981651	-0.03139759
13	0.4	0.9959207	-0.06093976
14	0.45	0.9917646	-0.10890079
15	0.5	0.9846154	-0.18177515
16	0.55	0.9730649	-0.28592251
17	0.6	0.9554238	-0.42589209



The graph at the right shows the tabulated solution (dotted line) compared with the exact solution $y(x) = 1/(x^6+1)$ (pink line). We note a global good fitting; the relative error is less the 1E-6 for each node in the whole range. Note that at the end of the range the values become quite small ($y < 0.005$) but the RKF algorithm still maintains a very high relative precision ($\sigma_r < 1E-6$).

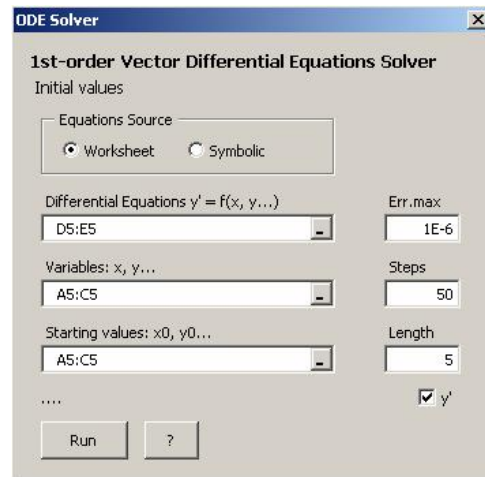
The macro can solve also ODE systems.

Example 2. Find a numerical solution of this simple differential system

$$\begin{cases} y_1' = y_2 \\ y_2' = -4y_1 - 5y_2 \end{cases} \quad \text{with} \quad \begin{cases} y_1(0) = 0 \\ y_2(0) = 2 \end{cases} \quad \text{and with} \quad 0 \leq x \leq 5$$

Arrange the worksheet reserving two columns for $[y_1, y_2]$ and the two adjacent columns for the derivatives $[y_1', y_2']$. In the first column set a x-grid with 50 step of increment $\bullet x = 0.1$. Then select the area A4:E55 and start the macro ODE Solver.

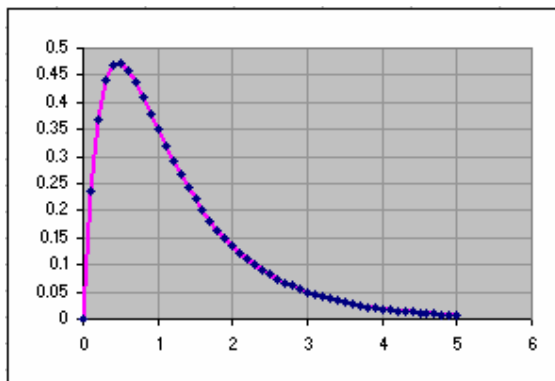
	A	B	C	D	E
4	x	y ₁	y ₂	y ₁ '	y ₂ '
5	0	0	3	3	-15
6	0.1				
7	0.2		=C5		
8	0.3			=(4*B5+5*C5)	
9	0.4				
10	0.5				
11	0.6				
12	0.7				
13	0.8				
14	0.9				
15	1				
16	1.1				
17	1.2				
18	1.3				
19	1.4				



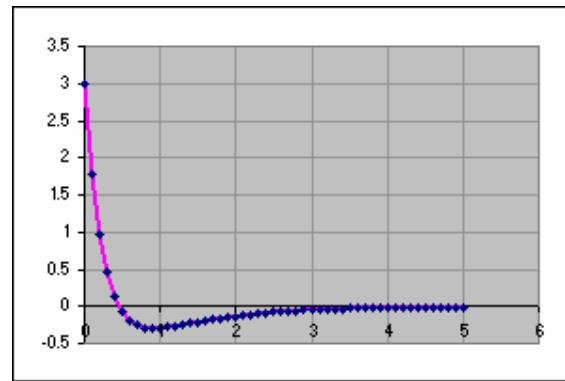
Press "Run" to start the integration algorithm. After few seconds and about 1400 function evaluations, the macro output its result. Each solution node has a relative error less then 1E-6. Compare it with the exact solution

$$y_1 = e^{-x} - e^{-4x} \quad y_2 = -e^{-x} + 4e^{-4x}$$

The following graphs shows the exact solution (pink line) and the calculated solution (dotted line)



Solution $y_1(x)$



Solution $y_2(x)$

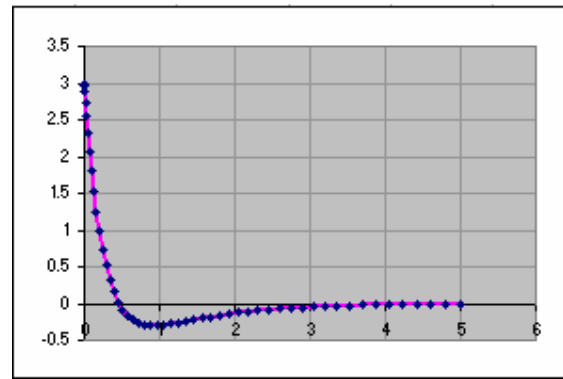
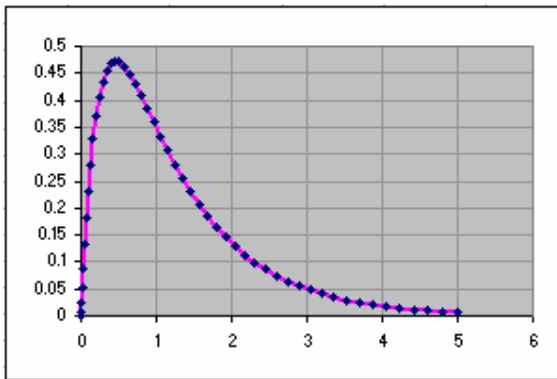
Note the general good fitting. It is a remarkable result overall if we consider the relative few points in the initial range where both functions have high swinging. This is a common effect due to the fixed grid: only 4 nodes cover the high oscillating range.

Variable grid. Sometime we would take more nodes in the range where the functions change faster, using a variable step grid. In the above example we need closer nodes at the beginning of the origin and more sparse in the remaining part. The macro can also work with variable grid.

For example taking the nodes: $x_n = 0.002 * n^2$ for $n = 0, 1, 2, \dots, 50$ we get the following result

n	x_n
0	0
1	0.002
2	0.008
3	0.018
4	0.032
5	0.05
...

	A	B	C	D	E	F
4	n	x	y_1	y_2	y_1'	y_2'
5	0	0	0	3	3	-15
6	1	0.002	0.00597	2.970126	2.9701257	-14.87451
7	2	0.008	0.023525	2.881994	2.8819944	-14.50407
8	3	0.018	0.05163	2.739963	2.7399626	-13.90633
9	4	0.032	0.088653	2.550907	2.5509069	-13.10915
10	5	0.05	0.132499	2.323694	2.3236936	-12.14846
11	6	0.072	0.180769	2.068515	2.0685154	-11.06565
12	7	0.098	0.230945	1.796168	1.7961675	-9.904617
13	8	0.128	0.280558	1.51733	1.5173296	-8.708879



We note more dense points where the functions change quickly. Observe that the grid choice (variable or fixed) does not affect the general precision of the solution. In both cases the nodes have a precision better than $1E-6$

Stopping long elaboration

Sometime the elaboration becomes too slow or the algorithm cannot overcome a too difficult step. To escape from this hanging status we can click the "Stop" button. The macro automatically ends its elaboration and output the results obtained at this moment (if any).

The macro can automatically abort the elaboration when some critical situation occurs.

- IERR 2: Too many evaluation loops. For each step the max number of loops is 400.
- IERR 3: Max relative error too small. The minimum precision is $3E-13$.
- IERR 4: the solution values grow too large (overflow) and/or the optimal increment step cannot be achieved.

Automatic Grid. In order to save time, the macro memorizes the parameters (length, steps, err.max) of the last elaboration. Therefore if you want to solve another problem with the same fixed grid it is sufficient that you provide only the first definition row. The macro will automatically generate the grid (only fixed grid). See the following example

$$\begin{cases} y_1' = 2y_1 - y_1y_2 \\ y_2' = 3y_1 + 0.5y_1y_2 \end{cases} \quad \begin{cases} y_1(0) = 1 \\ y_2(0) = 0 \end{cases} \quad 0 \leq x \leq 5$$

	A	B	C	D	E
4	x	y ₁	y ₂	y ₁ '	y ₂ '
5	0	1	0	2	3
6				=2*B5-B5*C5	
7					=3*B5+0.5*B5*C5
8					
9					

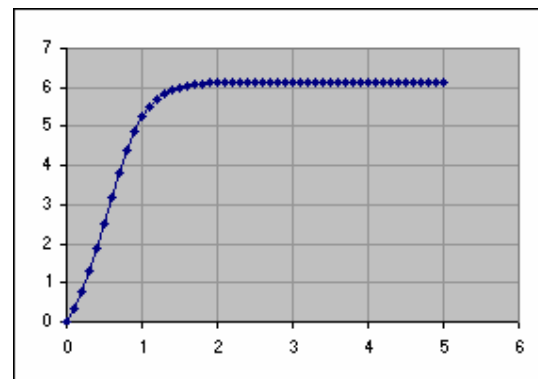
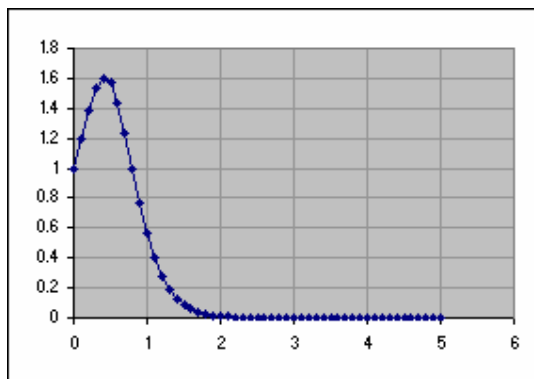
Insert in D2 the function: = 2*B5-B5*C5

Insert in E2 the function: = 3*B5-0.5*B5*C5

Select the range A5:E5 (or A4:E5 indifferently) and start the macro

	A	B	C	D	E
4	x	y ₁	y ₂	y ₁ '	y ₂ '
5	0	1	0	2	3
6	0.1	1.201654	0.339514	1.9953302	3.808951
7	0.2	1.389907	0.764623	1.7170591	4.7010986
8	0.3	1.533833	1.279449	1.1052045	5.5827311
9	0.4	1.600968	1.875876	0.1987185	6.3045124
10	0.5	1.56938	2.529186	-0.830494	6.6927687
11	0.6	1.439383	3.198935	-1.725727	6.6203969
12	0.7	1.236075	3.837503	-2.271291	6.079945
13	0.8	0.999156	4.403437	-2.401409	5.1973291
14	0.9	0.766837	4.872122	-2.202449	4.1685709

As we can see the x-grid is generated by the macro itself



Symbolic equations

We can also define a differential equation by a symbolic string. For example

$$y' = -6y^2 \sin^5(\pi x) \quad , \quad y(0) = 1$$

	A	B	C
3			
4	y' = -6*sin(pi*x)^5*y^2		
5			
6	x	y	y'
7	0	1	
8	0.1		
9	0.2		
10	0.3		
11	0.4		

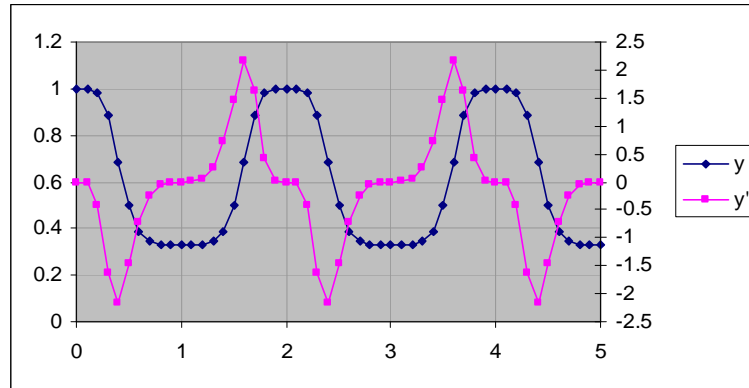
Put the string $y' = -6*\sin(\pi*x)^5*y^2$ somewhere above the table defined in A6:C57.

The y' column is optional. If present, do not add any formula in it. Select the range A6:C57 and start the



macro

Because the macro do not find any Excel formula in column y' (if any), search for a differential equation string above the header table. Make sure that the equation contains the same symbolic variables "x" and "y" that we have inserted in the header labels.



The **parameters** entry field is optional; it may be used for passing values to same parameters contained in the equation

	A	B	C	D
3				ω
4	$y' = -6 * \sin(\omega * x) * y$			2.2
5				
6	x	y	y'	
7	0	1	0	
8	0.08	0.958744	-1.00721	

When the equation contains one or more parameters we must also insert the cells containing their current values: D4 in this case



Note that, also in this case, the label symbol " ω " above the cell D4 must coincide with those contained in the symbolic equation. This permits to the internal math parser the correct association values-variables

Stiff ODE.

The RKF algorithm can be used for solving moderate stiff equation.

$$y'' + 101y' + 100y = 0, \quad y(0) = 0, \quad y'(0) = 99, \quad 0 \leq x \leq 5$$

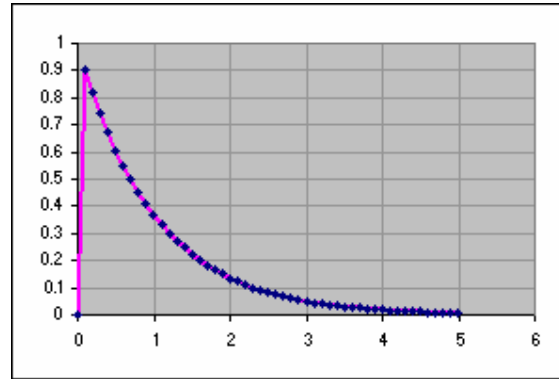
has the exact solution $y = e^{-x} - e^{-100x}$. The stiff-ratio is 1:100

We choose a tabulating step of 0.1, in order to obtain a table of 50 nodes with a rel. error less than $1E-5$, without regarding of the constraint induced by the integration step that should be considerable smaller. As known, the 2nd order differential equation can be transformed into the following 1st order differential system

$$\begin{cases} y_1' = y_2 \\ y_2' = -100y_1 - 101y_2 \end{cases} \quad \begin{cases} y_1(0) = 0 \\ y_2(0) = 99 \end{cases} \quad 0 \leq x \leq 5$$

	A	B	C	D	E
4	x	y ₁	y ₂	y ₁ '	y ₂ '
5	0	0	99	99	-9999
6	0.1	0.90479	-0.9003	-0.9003	0.45084
7	0.2	0.81873	-0.8187	-0.8187	0.81871
8	0.3	0.74082	-0.7408	-0.7408	0.74081
9	0.4	0.67032	-0.6703	-0.6703	0.67032

Insert in D1 = -C5
and insert in E1 = -100*B5 -101*E5
Then select A5:E5 and start the macro using
Err Max = 1E-5, Steps = 50, Length = 5



As we can see, all the nodes are correctly tabulated with the require precision even if the initial transient is quite invisible. But, as told, the output setting does not influence the final precision.

Of course, for reaching this result the algorithm has performed much more inner steps than 50 used for plotting. In this example the total evaluations was about 4000, taking about a few ten of seconds. Higher stiff-ratio equations could be solved but the elaboration time increases sharply.

In that case we can try to reduce the precision: a relative error of 1E-3 (0.1%) is usually a good compromise for plotting. For example, the solution of the following Van der pool stiff equation

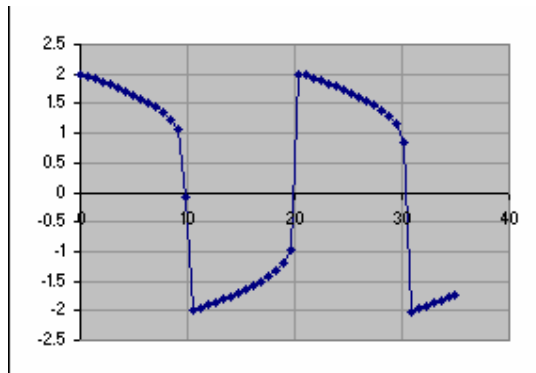
$$y'' - 11(1 - y^2)y' + y = 0, \quad y(0) = 2, \quad y'(0) = 0, \quad 0 \leq x \leq 35$$

is equivalent to

$$\begin{cases} y_1' = y_2 \\ y_2' = 11(1 - y_1^2)y_2 - y_1 \end{cases} \quad \begin{cases} y_1(0) = 2 \\ y_2(0) = 0 \end{cases} \quad 0 \leq x \leq 35$$

The ODE Solver, with Err. Max = 1E-3, steps = 50, length = 35, requires no more then 4500 evaluation points and about 10 seconds for producing the plot at the right.

Observing the steps counter, you will note that the stronger consumption of elaboration time happens at the transition points (about 10, 20 and 30).



Piecewise differential equations

It is very common in practical situations have differential problems where the functions have different definitions in sub interval of the integration range (piecewise definition)

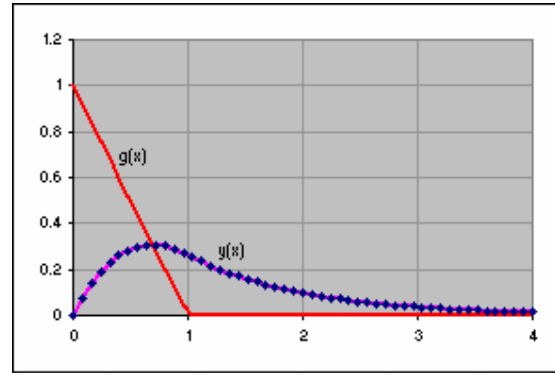
Example

$$y' + y = g(x), \quad y(0) = 0, \quad \text{where } g(x) = \begin{cases} 1-x & x < 1 \\ 0 & x \geq 1 \end{cases} \quad \text{for } 0 \leq x \leq 4$$

The usual method for solving this problem is to divide it into 2 parts: we solve before the left part $y_1(x)$ from 0 to 1 and then, we solve the second part $y_2(x)$ from 1 to 4 using the value $y_1(1)$ as starting value of the second integration task. This method becomes quite tedious when there are many parts. RKF algorithm can easily integrate this differential equation in one single task

	A	B	C	D
4	x	y	y'	
5	0	0		1
6	=IF(A5<1,1-A5-B5,-B5)			0.2327
7				0.2876
8	0.24	0.186744	0.5732557	
9	0.32	0.227702	0.452298	
10	0.4	0.25936	0.3406401	
11	0.48	0.282433	0.2375667	

Insert in C5 the function = IF(A5<1,1-A5-B5, -B5)



Compare with the exact solution

$$y(x) = \begin{cases} 2 - x - e^{-x} & x < 1 \\ (e-2)e^{-x} & x \geq 1 \end{cases}$$

Observe that the function $g(x)$, in that case, has no discontinuity and therefore also the derivative y' has no discontinuity.

Discontinue derivative.

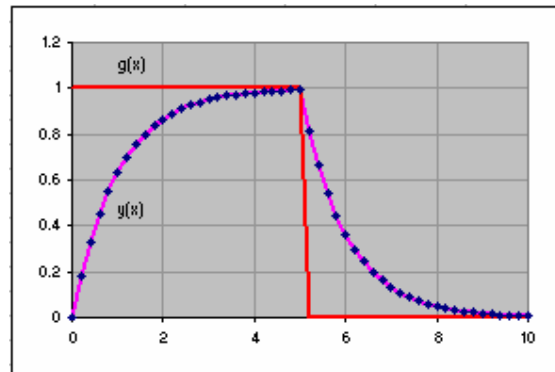
Sometime the function $g(x)$ may have finite discontinuities (jumps) in isolated points. This means that the direction of $y(x)$ may changes sharply (corner points). This situation, very difficult for many algorithms, may be successfully solved by RKF algorithm. Discontinuity points of the derivative $y'(x)$ obligate the algorithm to heavy calculation like a stiff problem. Let's see the following example

$$y' + y = g(x), \quad y(0) = 0, \quad \text{where } g(x) = \begin{cases} 1 & x < 5 \\ 0 & x \geq 5 \end{cases} \quad \text{for } 0 \leq x \leq 10$$

	A	B	C	D
4	x	y	y'	
5	0	0		1
6	=IF(A5<5,1-B5,-B5)			0.7306
7				0.3199
8	0.6	0.451189	0.5488112	
9	0.8	0.550672	0.4493283	
10	1	0.632121	0.3678787	

Insert in C5 the function = IF(A5<5, 1-B5, -B5)

Note that the max computation effort of the algorithm happens during the transition of the point $x = 5$

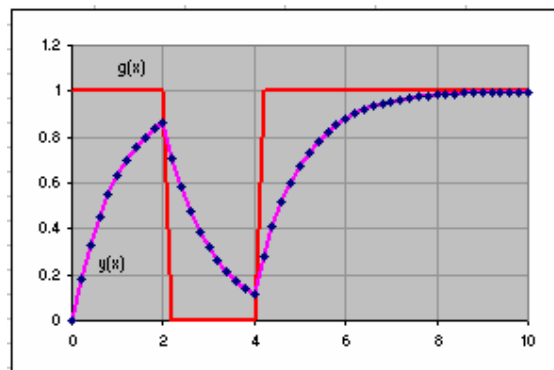


Repeating with the function $g(x)$

$$g(x) = \begin{cases} 1 & x \leq 2 \vee x > 4 \\ 0 & 2 < x \leq 4 \end{cases}$$

we get the following graph

Note that, in spite of the relative few points around the transition instants $x = 2$ and $x = 4$, the solution curve stays acceptable for all integration range.



Non Linear System

Example. Solve the following non linear ODE system for $0 \leq t \leq 5$

$$\begin{cases} x' = -xz - 1 \\ y' = x - yz \\ z' = x^2 + y^2 \end{cases} \quad \text{with} \quad \begin{cases} x(0) = 1 \\ y(0) = 0 \\ z(0) = 0 \end{cases}$$

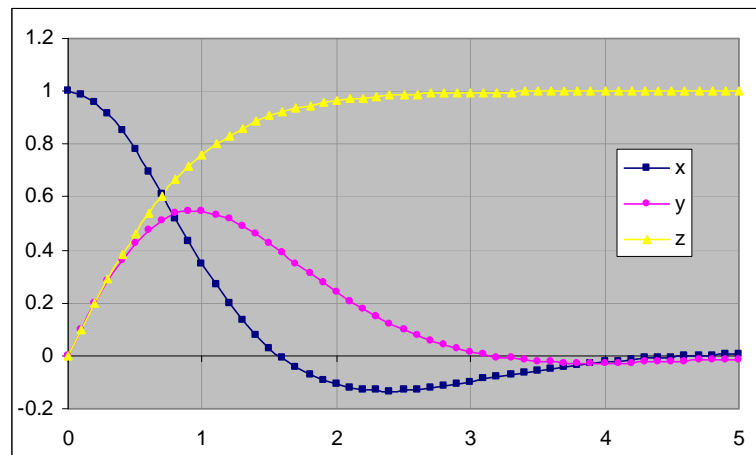
Choose a step grid of $\Delta t = 0.1$ and arrange a worksheet as the following. Now select the range A4:G55 and start the macro ODE solver.

Set on the check box y' and click "Run". After few seconds, the table will be filled with the solution found

	A	B	C	D	E	F	G
4	t	x	y	z	x'	y'	z'
5	0	1	0	0	0	1	1
6	0.1						
7	0.2		=B5*D5-C5		=B5-C5*D5		
8	0.3						
9	0.4					=B5^2+C5^2	
10	0.5						

	A	B	C	D	E	F	G
4	t	x	y	z	x'	y'	z'
5	0	1	0	0	0	1	1
6	0.1	0.99	0.09934	0.09967	-0.198	0.98015	0.99007
7	0.2	0.9608	0.19476	0.19738	-0.3844	0.92235	0.96104
8	0.3	0.9139	0.2827	0.29131	-0.5489	0.83155	0.91514
9	0.4	0.852	0.36021	0.37995	-0.6839	0.71512	0.85564
10	0.5	0.7783	0.42516	0.46212	-0.7848	0.58178	0.78645

The plot of the solution $x(t)$, $y(t)$, $z(t)$ are shown in the following graph

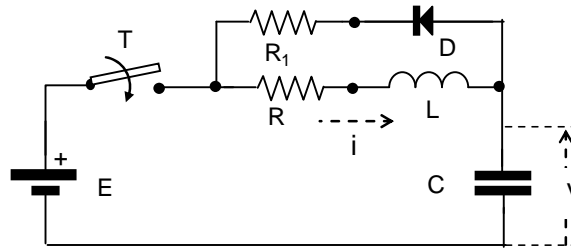


Compare with the exact solution

$$x(t) = \frac{x_0 \cos(t)}{\cosh(x_0 t)}, \quad y(t) = \frac{x_0 \sin(t)}{\sinh(x_0 t)}, \quad z(t) = x_0 \tanh(x_0 t), \quad \text{where } x_0 = 1$$

Non-Linear Electric Circuit

Sometime the differential functions may be quite complicate. We can use the worksheet for storage intermediate results or set complicate auxiliary functions. Let's see the following electric problem



$$\begin{aligned} R &= 30 \Omega \\ C &= 100 \cdot \text{F} \\ L &= 2 \text{ H} \\ R_1 &= 1000 \Omega / 10 \Omega \\ E &= 1 \text{ V} \end{aligned}$$

We study the electric transient up to 45 ms when $R_1 = 1000$ and 10Ω

The diode D is assumed ideal

The non-linear differential system of the circuit can be written as functions of the current intensity "i" flowing in the inductor and the voltage drop "v" of the capacitor.

$$\begin{bmatrix} i' \\ v' \end{bmatrix} = \begin{bmatrix} -R/L & -1/L \\ 1/C & a(v) \end{bmatrix} \cdot \begin{bmatrix} i \\ v \end{bmatrix} + \begin{bmatrix} E/L \\ b(v) \end{bmatrix}, \quad \begin{bmatrix} i \\ v \end{bmatrix}_{t=0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

where the auxiliary functions $a(v)$ and $b(v)$, depending from v , are

$$a(v) = \begin{cases} 0 & v < E \\ (R_1 C)^{-1} & v \geq E \end{cases} \quad b(v) = \begin{cases} 0 & v < E \\ E(R_1 C)^{-1} & v \geq E \end{cases}$$

A possible worksheet arrangement may be the following

	A	B	C	D	E
1	R1	R	C	L	E
2	1000	30	0.0001	2	1
3					
4		A		b	
5		-15	-0.5	0.5	
6		10000	0	0	
7					
8	t	i	v	i'	v'
9	0	0	0	0.5	0
10	0.0045	=B5*B9+C5*C9+D5			
11	0.009		=B6*B9+C6*C9+D6		
12	0.0135				
13	0.018				

The formulas for the matrix **A** are:

$$B5 = -B2/D2,$$

$$C5 = -1/D2$$

$$B6 = 1/C2,$$

$$C6 = \text{IF}(C9 > E2, -1/(A2 * C2), 0)$$

The formulas for the vector **b** are:

$$D5 = E2/D2$$

$$D6 = \text{IF}(C9 > E2, E2/(A2 * C2), 0)$$

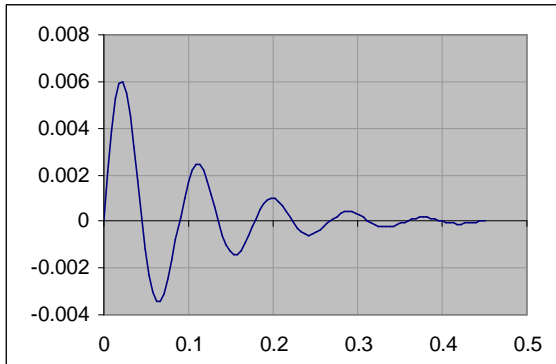
The tabulating step is set to $\Delta t = 0.0045$ (100 steps)

Select the range A8:E109 and start the macro.

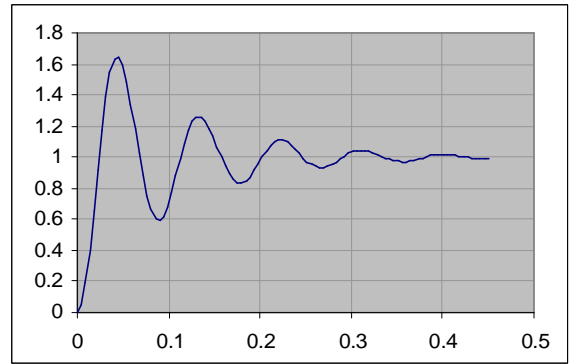
Observe that the macro works only in the cells of row 9: precisely it writes in A9 and C9 and reads from D9 and E9, no matter how complicate the formulas are for obtaining these values. This is one of the main advantage of working directly "on-worksheet".

Differential Equations $y' = f(x, y)$		Err.max
D9:E9		0.0001
Variables: x, y	A9:C9	Steps
		100
Starting values: x0, y0	A9:C9	Length
		0.45
....		<input checked="" type="checkbox"/> y'

Clicking "Run", we obtain the following plot

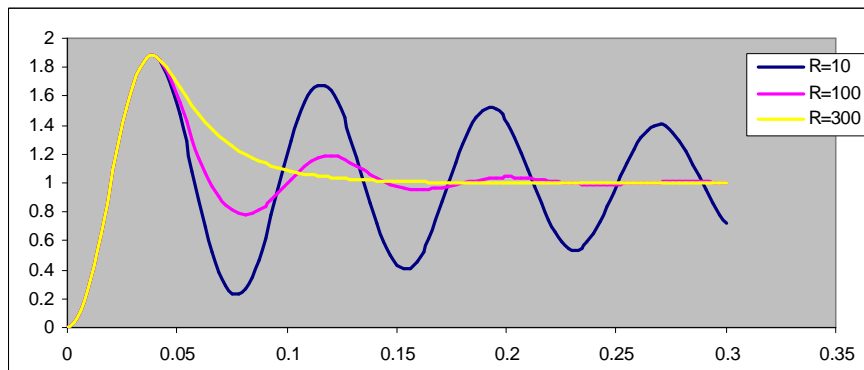


Inductor current $i(t)$ ($R_1 = 1000$ ohm)

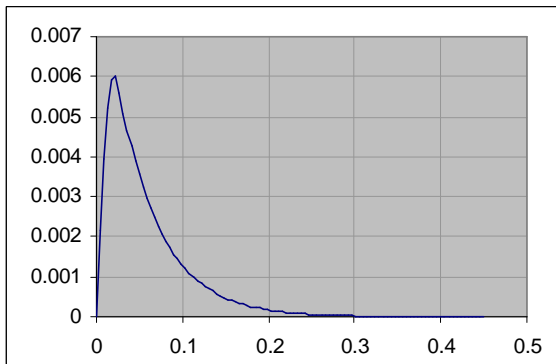


Capacitor voltage $v(t)$ ($R_1 = 1000$ ohm)

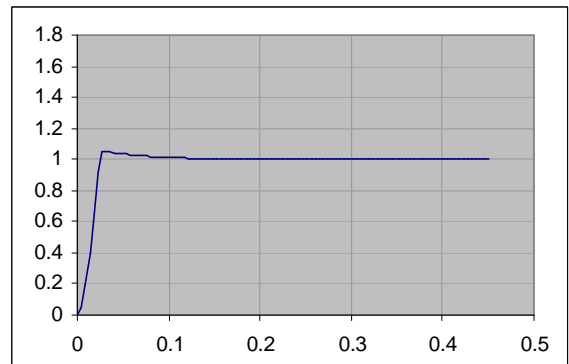
As we can see this is a classical underdamped circuit where the capacitor voltage oscillates largely around its equilibrium final value $v(\infty) = 1$



Now let's repeat the analysis after setting $R_1 = 10$. The curves for this case is drawn in the following graphs



Inductor current $i(t)$ ($R_1 = 10$ ohm)



Capacitor voltage $v(t)$ ($R_1 = 10$ ohm)

The voltage capacitor reaches its final value without unwanted oscillations and in shorter time (settling time $\cong 0.025$ s). The transient evolves as in damped case.

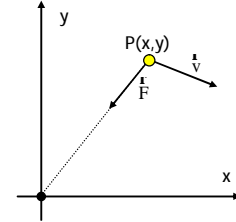
The Orbits

Another interesting problem is the integration of the motion equations in a given strength field
 For simplicity we restrict our examination to a radial plane field $\mathbf{F} = (F_x, F_y)$ and a body of infinitesimal dimension having mass $m = 1$ kg.

Calling the coordinates of the position $\mathbf{P} = (x, y)$ and the velocity $\mathbf{v} = (v_x, v_y)$, the motion equations are:

$$x' = v_x \quad , \quad y' = v_y \quad , \quad v_x' = F_x / m \quad , \quad v_y' = F_y / m \quad (1)$$

A radial field can be written as $\mathbf{F} = f(r)[x/r \quad , \quad y/r]$



where the "r" is the distance from the origin $r = \sqrt{x^2 + y^2}$ and $f(r)$ is a suitable scalar function characterizing the field itself. For example $f(r) = -k \cdot r$ (elastic strength), $f(r) = -k / r$ (distance inverse), $f(r) = -k / r^2$ square distance inverse.

Let's try to solve the differential motion in the three cases using the same initial conditions $P_0=(0,1)$, $v_0=(0.4, 0)$ and setting the same constant $k = 0.4$

First of all, prepare a worksheet as the following

	A	B	C	D	E	F	G	H	I
4	k	r	u _x	u _y	f(r)	F _x	F _y		
5	0.4	1	0	1	-0.4	0	-0.4		
6									
7	t	x	y	v _x	v _y	x'	y'	v _x '	v _y '
8	0	0	1	0.4	0	0.4	0	0	-0.4
9	0.2								
10	0.4								
11	0.6								

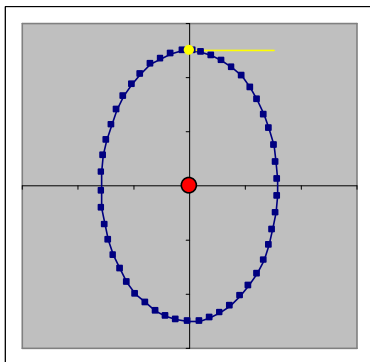
where the cells contain the formulas:

$B5 = \text{SQRT}(B8^2 + C8^2)$, $C5 = B8/B5$, $D5 = C8/B5$, $E5 = -A5*B5$, $F5 = E5*C5$, $G5 = E5*D5$, $F8 = D8$, $G8 = E8$, $H8 = F5$, $I8 = G5$

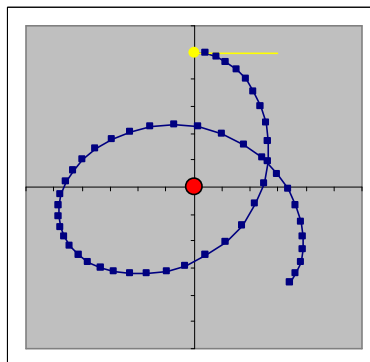
Set a grid of 50 steps with increment $\Delta t = 0.2$.

Select the range A7:I58 and start the macro ODE solver. The solutions (x, y) appear in the ranges B8:B58 and C8:C58. Plotting one against the other gives the orbit of the body

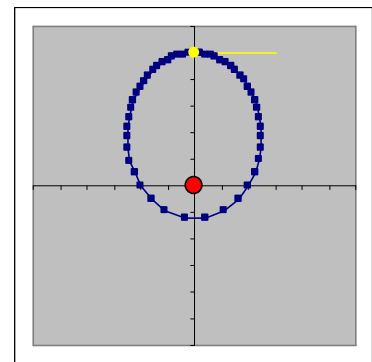
$F(r) = -k \cdot r$, $0 \leq t \leq 10$



$F(r) = -k/r$, $0 \leq t \leq 10$

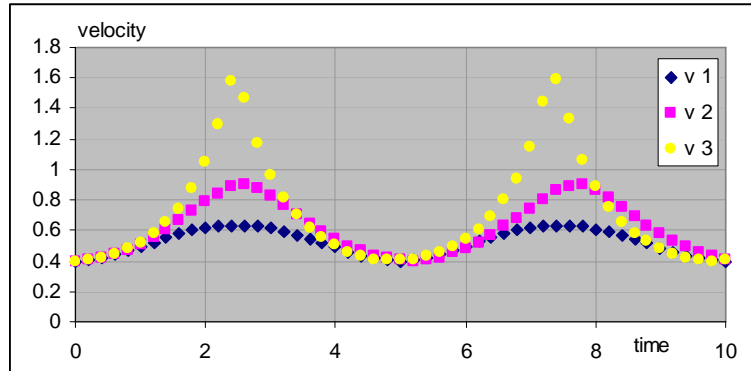


$F(r) = -k/r^2$, $0 \leq t \leq 5$

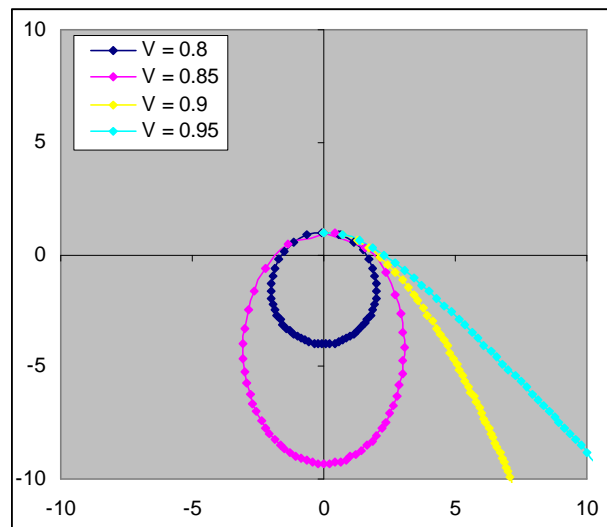


Note that in the last case we have taken a time of 5 instead of 10 because the motion is faster than the previous cases (the rotation is about twice than the first case). Note also that the points near the origin (0,0) are quite sparse while, on the contrary, the points distant from origin are very close. This means that, in the last case, the velocity increases sharply near the origin and decreases far from it. This is just what happens in the orbits of the planets.

If we compute the scalar velocity $|v| = \sqrt{v_x^2 + v_y^2}$ using the data of the columns D and E, we can compare the scalar velocity of each case. As we can see, the graph evidences the "slingshot effect" of the third case.



The initial scalar velocity, as known, determines the type of the orbit. When the velocity increases, the cyclic elliptic orbit becomes progressively more eccentric until it opens itself in a parabolic trajectory ($V = 0.9$). For higher velocity the trajectory becomes an open hyperbolic curve. All these very interesting results can be easily simulated with the aid of the macro ODE solver



Macro ODE Slope Grid

This macro generate and visualize the slope $y'(x)$ of a 1st ODE solution over a rectangular domain. It is didactically useful for studying the 1st order differential equation $y' = f(x, y)$

For example we have to study the following equation

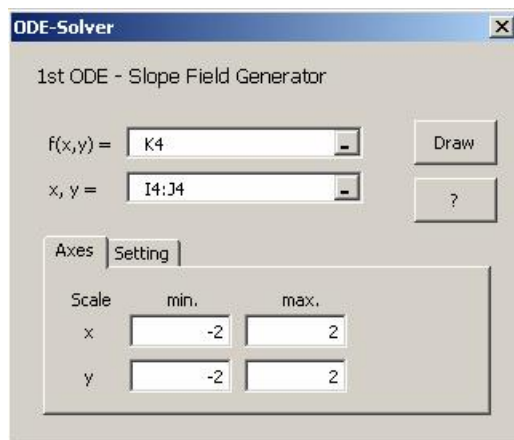
$$y' + y = e^{-x} \quad \Rightarrow \quad y' = xe^{-x} - y$$

Therefore we have to plot the slope given by the bivariate function $f(x, y) = e^{-x} - y$ in a suitable rectangular domain, for example in $D = \{-2 \leq x \leq 2, \{-2 \leq y \leq 2\}$

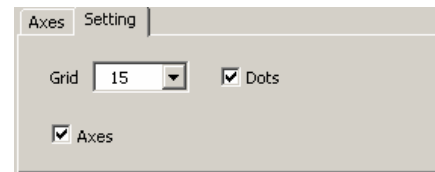
For using this macro, set the variables x and y in adjacent cells, for example I4 and J4
Then, in the adjacent right cell K4, insert the formula defining y' . thus:
= EXP(-I4)-J4

	H	I	J	K
1				
2				=EXP(-I4)-J4
3		x	y	f(x,y)
4		1.75	1.75	-1.576226
5				

Select the range I4:K4 and start the macro ODE Slope Field



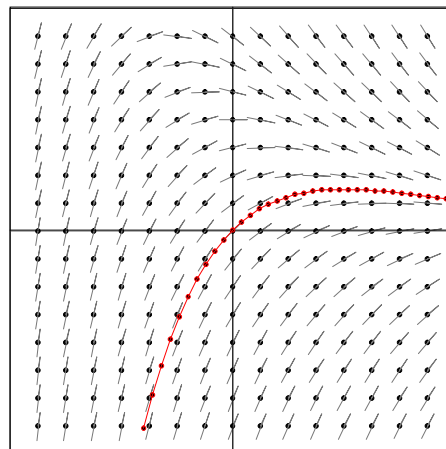
Option Tab



The Grid number, from 4 to 24, set the density of the grid. The dots check box adds the grid points to the plot. The axes check box adds the x-y axes

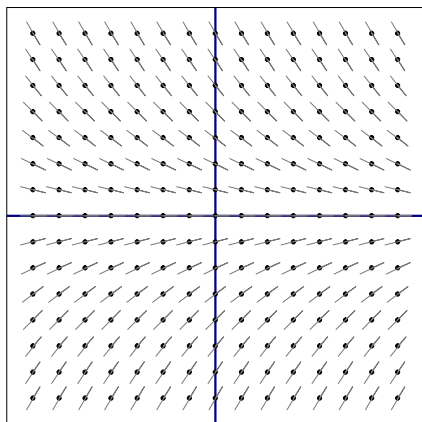
In the Axes tab we can set the domain D and after that click the "Draw" button

The macro generates a graphical object showing the slope y' of each point of the grid.
Every solution $y(x)$ of the differential equation follows the direction of the slope field. Therefore the field gives a global view of the solutions crossing the given domain.
For example, the solution crossing the point (0,0) is $y = e^{-x}x$ (red line)
We can see that this solution evolves following the slope directions. It never crosses the them any point.

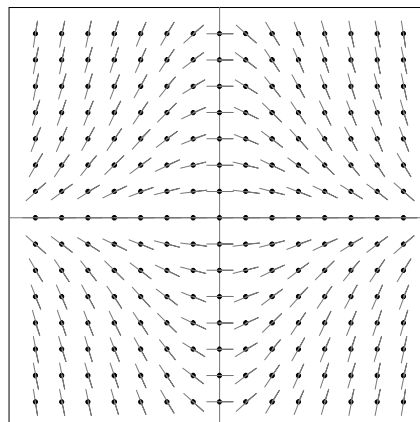


Other examples of slope fields

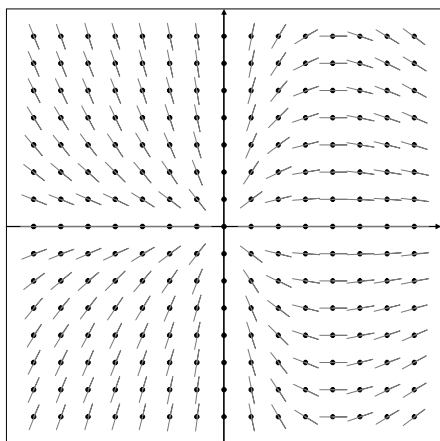
$$y' = -y \quad D \equiv \{-2 \leq x \leq 2, -2 \leq y \leq 2\}$$



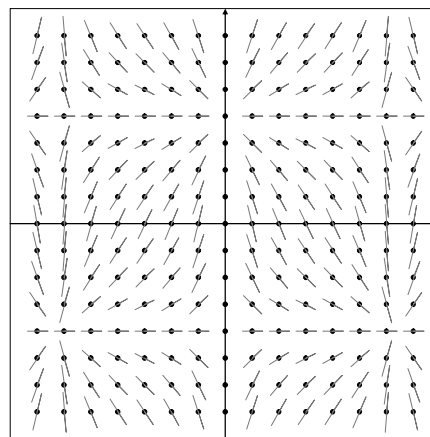
$$y' = -2xy \quad D \equiv \{-2 \leq x \leq 2, -2 \leq y \leq 2\}$$



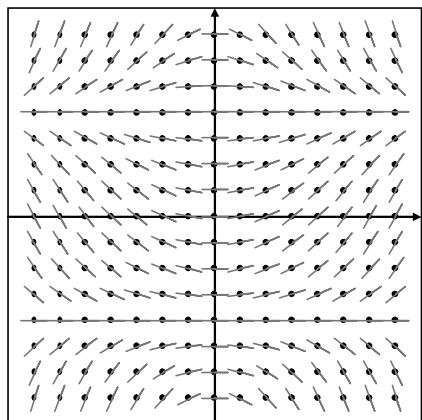
$$y' = y \frac{1-x}{x} \quad D \equiv \{-2 \leq x \leq 2, -2 \leq y \leq 2\}$$



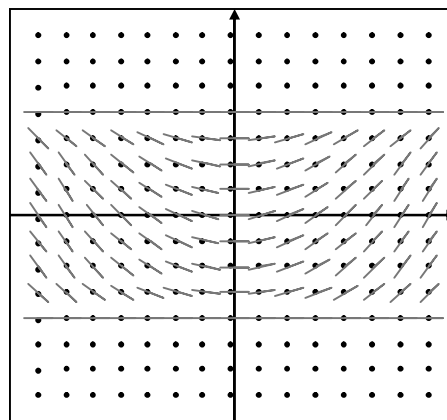
$$y' = \frac{|y|-1}{\sin x} \quad D \equiv \{-4 \leq x \leq 4, -2 \leq y \leq 2\}$$



$$y' = x(1-y^2) \quad D \equiv \{-2 \leq x \leq 2, -2 \leq y \leq 2\}$$



$$y' = x\sqrt{1-y^2} \quad D \equiv \{-2 \leq x \leq 2, -2 \leq y \leq 2\}$$



About FD method

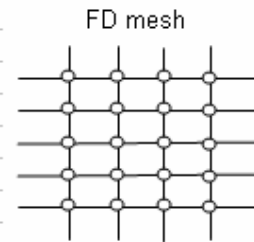
Many problems in science and engineering can be described mathematically by partial differential equations with boundary (BC) and initial conditions (IC). The goal of the finite-differences method is to replace continuous derivatives with difference formulas that involve only the discrete values associated with the position on the mesh.

Grid, Mesh and Cells

In FD method the continuous differential equation is replaced with discrete formulas. So the solution is known only at the cross of a rectangular grid (mesh). Each intersection point (knot) takes a numeric value.

Translating in the language of the spreadsheet we can say that each cell represents a knot of the mesh and the entire range corresponds to the mesh itself

Spreadsheet range			
0.58	0.18	0.11	0.84
0.18	0.19	0.26	0.55
0.98	0.37	0.63	0.73
0.21	0.75	0.69	0.01
0.14	0.65	0.36	0.59



Each cell contains the discrete solution of the correspondent knot. As we can see, for rectangular mesh, the similarity is very tight, so it is natural to think to apply the power of the spreadsheet tool to the FD method.

In this context the word “discrete” means that the numerical solution is known only at a finite number of points in the physical domain. The number of those points can be chosen by the user. In general, increasing the number of points not only increases the resolution (i.e., detail), but also the accuracy of the numerical solution.

The **mesh** is the set of locations points where the discrete solution is computed. These points are called **nodes** or **knots**. The key parameters of the mesh are Δx and Δy - the local distance between adjacent points in space - and Δt - the local distance between adjacent time steps. The steps of both axes are uniform throughout the mesh.

$$x_j = (j-1) \Delta x, \quad j = 1, 2, \dots, N \quad y_i = (i-1) \Delta y, \quad i = 1, 2, \dots, M \quad (\text{for } x\text{-}y \text{ mesh})$$

$$x_j = (j-1) \Delta x, \quad j = 1, 2, \dots, N \quad t_i = (i-1) \Delta t, \quad i = 1, 2, \dots, M \quad (\text{for } x\text{-}t \text{ mesh})$$

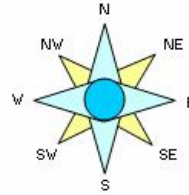
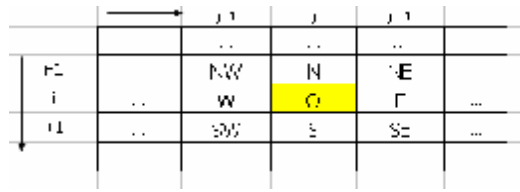
We assume to write the continuous solution evaluated at the mesh point as

$$T(x_j, y_i) = T(i, j) \quad \text{or} \quad T(t_i, x_j) = T(i, j)$$

There is a relationship between the discrete solution and the values of the spreadsheet cells, as shown in the following figure

		j-1	j	j+1	
		
i-1	...	$T(i-1, j-1)$	$T(i-1, j)$	$T(i-1, j+1)$...
i	...	$T(i, j-1)$	$T(i, j)$	$T(i, j+1)$...
i+1	...	$T(i+1, j-1)$	$T(i+1, j)$	$T(i+1, j+1)$...
		

It is common use to rename the cells around the central cell (i, j) following the "Cardinal Points" rule. That is: T_{NW} , T_W , T_{SW} , T_N , T_S , T_{NE} , T_E , T_{SE} . The central point is called T_O



The basic idea of the finite-difference method is to replace continuous derivatives with so-called **Finite Difference** that involve only the discrete values associated with positions on the mesh.

Applying the finite difference method to a differential equation involves replacing all derivatives with difference formulas. In the heat equation there are derivatives with respect to time, and derivatives with respect to space. Using different combinations of mesh points in the difference formulas results in different schemes (FTCS, Lax-Wendrof, Upwind, Crank-Nicolson, Euler etc.)

The reason of so many schemes is that each partial differential equation problem has different mathematical and physical behaviours and requires a dedicated schema. Different classes of equations have different requirements for their boundary and initial conditions. Hyperbolic and parabolic equations have open boundary in the time domain. Conditions specified at $t = 0$ are called initial conditions.

- Parabolic equations require one initial condition and two boundary conditions.
- Hyperbolic equations require two initial conditions and two boundary conditions
- Elliptic equations require boundary conditions at a continuous closed boundary.

The discrete approximation results in a set of algebraic equations that are evaluated (or solved) for the values of the discrete unknown variables. Thus the original continue problem is transformed into a linear algebraic system, often tridiagonal that can be solved with many popular efficient algorithms.

For simplicity, we will only consider second order partial differential equations with two variables of which there are three different types: parabolic (diffusion), hyperbolic (wave), and elliptic (potential). For each of one it is possible to solve the initial or boundary problem, with Dirichlet or Neuman condition.

The **FDSolver** is an add-in for Excel containing macros for solving numerically partial differential (PDE) equations and ordinary differential equations (ODE) with the Finite Differences Method (FD).

The most part of the following documentation is extracted from the help on-line of this add-in

Partial Differential Equations

Basically there are three different types of second order partial differential equations with two variables: parabolic (diffusion), hyperbolic (wave), and elliptic (potential). For each of one it is possible to solve the initial or boundary problem, with Dirichlet or Neuman conditions.

Parabolic equation

This equation, known as the "heat equation" or "diffusion equation", describes how the temperature along a metal bar evolves with time.

This problem is solved by the function $T(x, t)$ which satisfies:

$$\frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2}$$

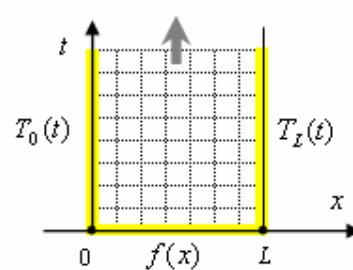
between $0 \leq x \leq L$

With the following boundary condition (BC)

$$T(0, t) = T_0(t) \quad , \quad T(L, t) = T_L(t)$$

And the following initial condition (IC)

$$T(x, 0) = f(x)$$



Hyperbolic equation

This equation is known as the "wave equation" and describes, for example, the propagation of electromagnetic waves.

This problem is solved by the function $W(x, t)$ which satisfies:

$$\frac{\partial^2 W}{\partial t^2} = v^2 \frac{\partial^2 W}{\partial x^2}$$

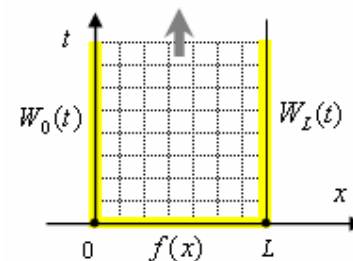
between $0 \leq x \leq L$

With the following boundary condition (BC)

$$W(0, t) = W_0(t) \quad , \quad W(L, t) = W_L(t)$$

And the following initial condition (IC)

$$W(x, 0) = f(x)$$



Elliptic equation

This equation is known as the potential equation and describes, for example, the electric potential due to a charge distribution.

This problem is solved by the function $U(x, y)$ which satisfies:

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = q(x, y) \quad \text{Poisson equation}$$

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0 \quad \text{Laplace equation}$$

Where the point (x, y) belong to a region of the plane

For a rectangular region we have

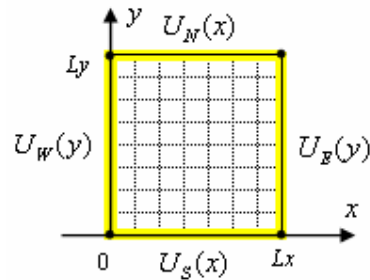
$$0 \leq x \leq L_x \quad , \quad 0 \leq y \leq L_y$$

For this problem we give only boundary conditions (BC)

. For a rectangular domain we have

$$U(0, y) = U_W(y) \quad , \quad U(L_x, y) = U_E(y)$$

$$U(x, 0) = U_S(x) \quad , \quad U(x, L_y) = U_N(x)$$



In polar coordinates these equations are transformed into the following

$$\frac{\partial^2 U}{\partial r^2} + \frac{1}{r} \frac{\partial U}{\partial r} + \frac{1}{r^2} \frac{\partial^2 U}{\partial \theta^2} = q(r, \theta) \quad \text{Poisson polar equation}$$

$$\frac{\partial^2 U}{\partial r^2} + \frac{1}{r} \frac{\partial U}{\partial r} + \frac{1}{r^2} \frac{\partial^2 U}{\partial \theta^2} = 0 \quad \text{Laplace polar equation}$$

Boundary and Initial Value Problem

In problem involving the time domain as in parabolic and hyperbolic equations we have to specify one initial condition, generally speaking a function $f(x)$, to get a unique solution.

This kind of problem is called **Initial Value Problem (IVP)** or also Dirichlet's problem

In problem involving the space finite domain as in elliptic equation we have to specify additional conditions, generally speaking one or more functions $f(x, y)=0$, at the border of the domain itself. Such a problem is called **Initial Boundary Value Problem (IBVP)**.

- The condition imposed at the solution functions $U(x, y)$, $T(x, t)$ or $W(x, t)$, are called **Dirichlet Condition**
- The condition imposed at the partial derivatives of the solution functions are called **Neuman Condition**

Dirichlet Condition: i.e. $U(x, 0) = \text{const}$

Neuman Condition: i.e. $\partial U(x, 0) / \partial x = f(x, y)$

Macros working rules

All the FDSolver macros follows simple rules for setting domain and boundary conditions.

We can specify the domain shape and the boundary conditions by coloured cells

We may set also the scale dx and dy (if not specified the macro assumes dx =1 and dy = 1)

The rules are:

- The macro works only in the region that we have selected
- Coloured cells are not changed by the macro and are used to specify the boundary conditions
- Uncoloured cells are modified during the process
- Uncoloured cell at the border of the domain are constrained with 1st derivative $\partial U/\partial x = 0$ or $\partial U/\partial y = 0$ (Neuman condition)

2D-Laplace

This macro solves the Laplace equation for a plane domain

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0$$

The algorithm uses the central finite differences formula

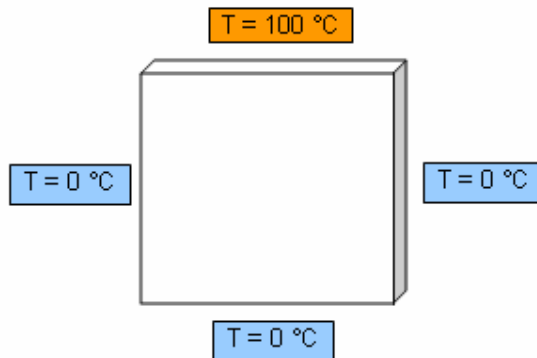
$$U = \frac{h_x^2(U_N + U_S) + h_y^2(U_E + U_W)}{2 \cdot (h_x^2 + h_y^2)}$$

	UN	
UW	U	UE
	US	

Where scale factors h_x and h_y are: $h_x = dx$, $h_y = dy$

Heat Diffusion of a 2D Rectangular Workpiece

Assume a square bidimensional plate having the temperature at each edge as indicated in the following picture



Boundary Conditions

Dirichlet

$$T(x, 0) = 0$$

$$T(x, 1) = 100$$

$$T(0, y) = 0$$

$$T(1, y) = 0$$

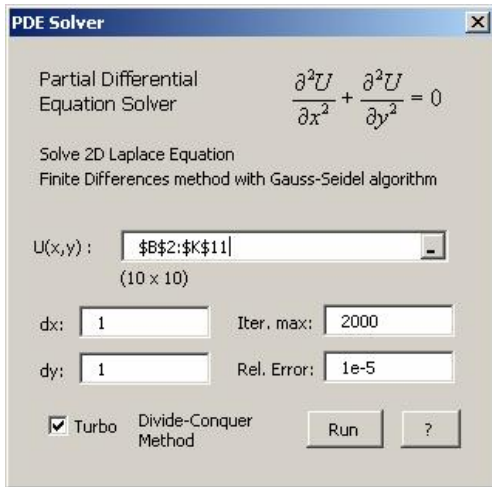
Let's begin to build the worksheet model. We divide the area in little cells. Each cells will be an own temperature. Assume, for clarity, to divide the plane in 8 x 8 cells. More fine is the grid, more accurate will be the model. If we like we can also to choose a rectangular grid like (8 x 16) or (16 x 32), or (20 x 30). If we choose a power of 2 for both axes the macro activates the Divide-and-Conquer acceleration (Turbo option). Usually this option increase the efficiency of more then 3-4 times

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2		100	100	100	100	100	100	100	100	100			
3		0									0		
4		0									0		
5		0									0		
6		0									0		
7		0									0		
8		0									0		
9		0									0		
10		0									0		
11			0	0	0	0	0	0	0	0			
12													
13													

We add the condition values into the cells at the border of the range and we colour them (as we like). These cells will be unchanged.

(8 x 8) grid

Now select the entire range B2:K11 (coloured and uncoloured cells) and star the macro **2D-Laplace**



Field **dx**, **dy** are the scale of the correspondent axes. In that case the macro assumes the default 1 because it has no information about it (we see in the next example how to set different scales)

The box field **Iter. max** sets the iteration limit
The box field **Rel.Error** sets the relative error limit
The macro stops itself when one of these limit is reached

The macro has automatically detected the condition for activating the Turbo option (the grid dimensions are power of 2)

Note. Relative errors and iterations max are parameters used only for stopping criterion. They do not increase the global accuracy. For increasing the global accuracy you have to choose a more fine grid.

Press the "Run" button and after few time all the cells will be filled with the corresponding temperature values $T(i, j)$.

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2			100	100	100	100	100	100	100	100		
3		0	48.6	66.91	74.5	77.3	77.3	74.5	66.9	48.6	0	
4		0	27.6	44.55	53.6	57.6	57.6	53.6	44.6	27.6	0	
5		0	17.2	30.04	38	41.7	41.7	38	30	17.2	0	
6		0	11.3	20.41	26.5	29.6	29.6	26.5	20.4	11.3	0	
7		0	7.48	13.78	18.2	20.4	20.4	18.2	13.8	7.48	0	
8		0	4.86	9.026	12	13.6	13.6	12	9.03	4.86	0	
9		0	2.92	5.45	7.29	8.25	8.25	7.29	5.45	2.92	0	
10		0	1.37	2.565	3.44	3.9	3.9	3.44	2.56	1.37	0	
11			0	0	0	0	0	0	0	0		
12												
13			104	0.01	2E-06							

The macro fills all uncoloured cells.

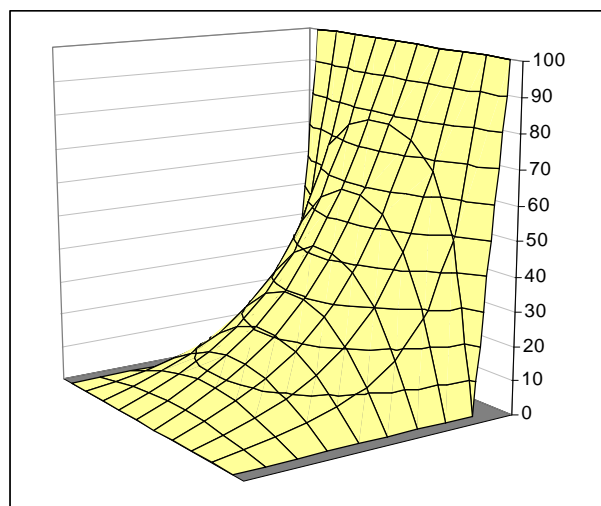
At the bottom also the macro writes:

The total iterations: 104

The elaboration time: 0.01 sec

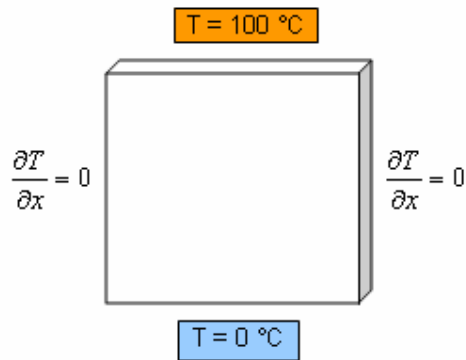
The average relative error: 2E-6

If we plot the surface of this matrix we obtain the 2D plot of the temperature function $T(x,y)$



Heat Diffusion of 2D Rectangular Workpiece with isolated edges

Assume to have the square bidimensional piece having the upper and lower edges at indicated temperature. The two other edges are isolated



Boundary Conditions

Dirichlet

$$T(x, 0) = 0$$

$$T(x, 1) = 100$$

Neuman

$$T_x(0, y) = 0$$

$$T_x(1, y) = 0$$

Where $T_x = \partial T / \partial x$

Let's begin to build the worksheet model. We add the condition values in the cells at the top and at the bottom of the range and we colour them (as we like). These cells will be unchanged.

	B	C	D	E	F	G	H	I	J	K	L
17											
18		100	100	100	100	100	100	100	100		
19											
20											
21											
22											
23											
24											
25											
26											
27		0	0	0	0	0	0	0	0	0	
28											

	B	C	D	E	F	G	H	I	J	K
17										
18		100	100	100	100	100	100	100	100	
19		88.9	88.89	88.89	88.9	88.9	88.9	88.9	88.9	
20		77.8	77.78	77.78	77.8	77.8	77.8	77.8	77.8	
21		66.7	66.66	66.66	66.7	66.7	66.7	66.7	66.7	
22		55.6	55.55	55.55	55.6	55.6	55.6	55.6	55.6	
23		44.4	44.44	44.44	44.4	44.4	44.4	44.4	44.4	
24		33.3	33.33	33.33	33.3	33.3	33.3	33.3	33.3	
25		22.2	22.22	22.22	22.2	22.2	22.2	22.2	22.2	
26		11.1	11.11	11.11	11.1	11.1	11.1	11.1	11.1	
27		0	0	0	0	0	0	0	0	
28										
29		184	0.071	2E-06						

Now select the entire range C18:J27 and star the macro **2D-Laplace**

Press the "Run" button and after few time all the cells will be filled with the corresponding temperature values $T(i, j)$

2D-Poisson

This macro solve the Poisson equation over a plane domain

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = q(x, y)$$

The algorithm uses the central finite differences formula

$$U = \frac{h_x^2(U_N + U_S) + h_y^2(U_E + U_W)}{2 \cdot (h_x^2 + h_y^2)}$$

	UN	
UW	U	UE
	US	

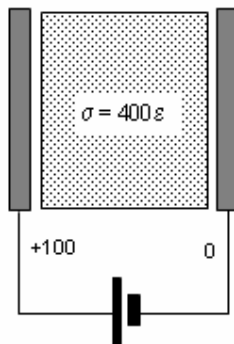
where the scale factors h_x and h_y are: $h_x = dx$, $h_y = dy$

The range $q(x, y)$ must have the same dimension of the integration range $U(x,y)$. Practically: if the solution range has $(n \times m)$ cells also the range $q(x, y)$ must have $(n \times m)$ cells.

Let's see how it works with the following examples

Electric potential

Assume to have a plane capacitor inserted in a region having a constant charge distribution $s = 400 \cdot e$. Find the potential distribution into the capacitance domain



Dimensions: $h = 0.6$ m, $w = 0.7$ m
 Battery: 100 V
 Charge density : $s = 400 \cdot e$ C/m²

The electric potential equation is

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = -\frac{s(x, y)}{e} \qquad \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = -400$$

This problem can be approximated with the following model.

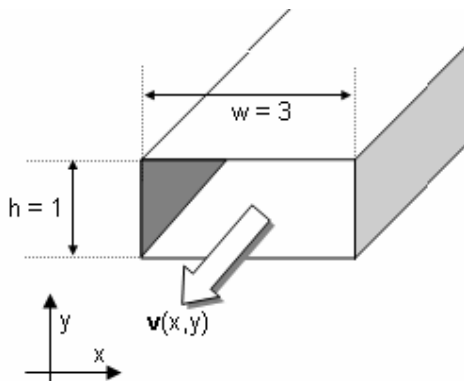
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	
16					U(x, y)										q(x, y)						
17																					
18		0.6	100							0		0.6	-400	-400	-400	-400	-400	-400	-400	-400	
19		0.5	100							0		0.5	-400	-400	-400	-400	-400	-400	-400	-400	
20		0.4	100							0		0.4	-400	-400	-400	-400	-400	-400	-400	-400	
21		0.3	100							0		0.3	-400	-400	-400	-400	-400	-400	-400	-400	
22		0.2	100							0		0.2	-400	-400	-400	-400	-400	-400	-400	-400	
23		0.1	100							0		0.1	-400	-400	-400	-400	-400	-400	-400	-400	
24		0	100							0		0	-400	-400	-400	-400	-400	-400	-400	-400	
25			0	0.1	0.2	0.3	0.4	0.5	0.6	0.7			0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	

The rectangular range at the right contains the value of the function $q(x, y)$, that are constant in this case. Now select the entire range C18:J24 and start the macro **2D-Poisson** You have to fill the input field $q(x,y)$ with the range M18:T24 (ranges of $U(x,y)$ and $q(x, y)$ must have the same dimension). Press the "Run" button and after few time all the cells will be filled with the corresponding values of the electric potential $U(x, y)$

Fluid flow velocity in a rectangular tube

This problem find the velocity of a viscous incompressible liquid flowing in a long tube with rectangular section. The Poisson equation is:

$$\Delta^2 v = s$$



v is the velocity of fluid flow, and s is the pressure drop per unit length divided by the viscosity of the fluid. (This is called Poiseuille flow.) x and y represent different points in the cross-section of the tube.

In this case $s = -10$ is constant in the section.

The section dimensions are: $h = 1$ m, $w = 3$ m

Assume the velocity at the boundary = 0

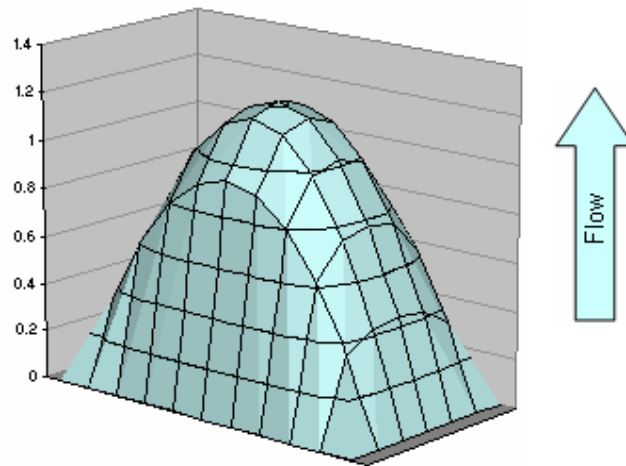
The numerical solution is obtained using the following range domain (at left) representing the tube section. At the right is the range representing the $s(x,y)$ distribution (constant in this case).

	A	B	C	D	E	F	G	H
61								
62								
63	1	0	0	0	0	0	0	0
64	0.9	0	0.353	0.426	0.4392	0.4261	0.353	0
65	0.8	0	0.6172	0.7545	0.7794	0.7545	0.6173	0
66	0.7	0	0.8007	0.9875	1.0217	0.9875	0.8007	0
67	0.6	0	0.9087	1.1266	1.1668	1.1267	0.9088	0
68	0.5	0	0.9444	1.1729	1.2151	1.1729	0.9444	0
69	0.4	0	0.9087	1.1267	1.1668	1.1267	0.9088	0
70	0.3	0	0.8007	0.9876	1.0218	0.9876	0.8008	0
71	0.2	0	0.6173	0.7546	0.7795	0.7546	0.6173	0
72	0.1	0	0.353	0.4261	0.4392	0.4261	0.353	0
73	0	0	0	0	0	0	0	0
74		0	0.5	1	1.5	2	2.5	3
75								

	J	K	L	M	N	O	P	Q
61								
62								
63	1	-10	-10	-10	-10	-10	-10	-10
64	0.9	-10	-10	-10	-10	-10	-10	-10
65	0.8	-10	-10	-10	-10	-10	-10	-10
66	0.7	-10	-10	-10	-10	-10	-10	-10
67	0.6	-10	-10	-10	-10	-10	-10	-10
68	0.5	-10	-10	-10	-10	-10	-10	-10
69	0.4	-10	-10	-10	-10	-10	-10	-10
70	0.3	-10	-10	-10	-10	-10	-10	-10
71	0.2	-10	-10	-10	-10	-10	-10	-10
72	0.1	-10	-10	-10	-10	-10	-10	-10
73	0	-10	-10	-10	-10	-10	-10	-10
74		0	0.5	1	1.5	2	2.5	3
75								

Note that, externally at the integration region, we have added both the scales x, y .

If we perform a surface plot of the region B63:H73, we get the following graph showing the velocity of the fluid into the section

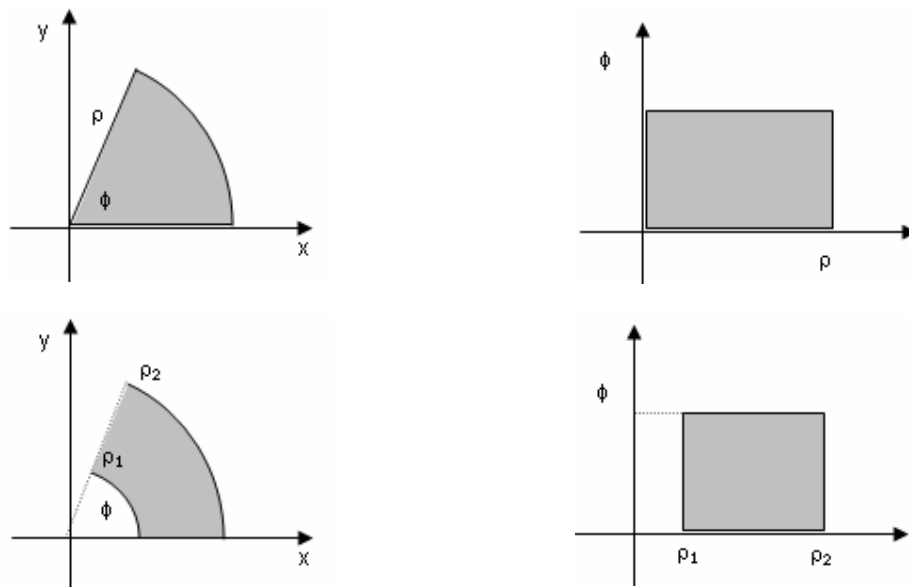


2D Laplace Polar

This macro solves the Laplace equation in polar coordinates over a rectangular plane domain

$$\frac{\partial^2 U}{\partial r^2} + \frac{1}{r} \frac{\partial U}{\partial r} + \frac{1}{r^2} \frac{\partial^2 U}{\partial \phi^2} = 0$$

This is useful when the original domain is circular or semi-circular. In those cases the transformed domain will be rectangular



We can specify the rectangular domain and the boundary conditions by coloured cells

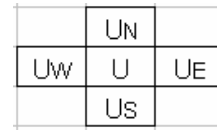
We can set also the scale dr and dq (if not specified the macro assumes $dr = 1$ and $dq = 1$)

Rules are:

- The macro works only in the selected region
- Coloured cells are not changed by the macro and are used to specify the boundary conditions
- Uncoloured cells are modified during the process
- Uncoloured cell at the border of the domain are constrained with 1st derivative equal to zero

The algorithm uses the central finite differences formula

$$U_{i,j} = \frac{1}{M} (a \cdot U_{i,j+1} + b \cdot U_{i,j-1} + c \cdot U_{i+1,j} + d \cdot U_{i-1,j})$$



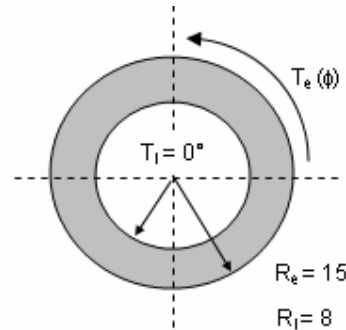
where:

$$a = h_q^2 r \left(r + \frac{h_r}{2} \right) \quad b = h_q^2 r \left(r - \frac{h_r}{2} \right) \quad M = 2(h_q^2 r^2 + h_r^2) \quad c = h_r^2 \quad d = h_r^2$$

Let's see how it works with same examples

Heat diffusion in a circular domain

We have the circular shape of the figure. The internal border has the radius of 8 m and the temperature of 0 C°, while the external border has the radius of 15 m and a temperature profile $T(q)$ given by the following table



θ	0	45	90	135	180	225	270	315	deg
T	100	97	95	90	85	85	90	95	° C

In polar coordinates the give problem can be easily modelled with the following rectangular domain

	A	B	C	D	E	F	G	H	I	J
33										
34	360	6.283	100							0
35	315	5.498	95							0
36	270	4.712	90							0
37	225	3.927	85							0
38	180	3.142	85							0
39	135	2.356	90							0
40	90	1.571	95							0
41	45	0.785	97							0
42	0	0	100							0
43			8	9	10	11	12	13	14	15
44	deg	radian		radius						

Note that we have added the angle scales both in deg and rad for clarity. The macros use only one scale: the nearest one to the left border. In this case the scale in radiant

You can insert or changes axes parameters also in the macro panel but this way is more simple and clear

Select the range C34:J42 and start the macro **2D Laplace polar**.

$\delta\rho$:	<input type="text" value="1"/>
$\delta\theta$:	<input type="text" value="0.78539816"/>
ρ :	<input type="text" value="8"/>
θ :	<input type="text" value="0"/>

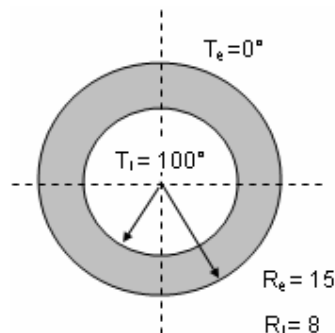
If we have arranged the worksheet as above, all the entry-field will be filled correctly. In particular the increment intervals dr and dq and the starting point (r, q)

Press "run" and all the cells will be automatically filled. The final result is shown below

	A	B	C	D	E	F	G	H	I	J
33										
34	360	6.2832	100	81.059	64.227	49.07	35.276	22.611	10.899	0
35	315	5.4978	95	77.195	61.269	46.864	33.715	21.621	10.424	0
36	270	4.7124	90	73.148	58.068	44.423	31.963	20.499	9.8835	0
37	225	3.927	85	69.216	55.02	42.13	30.332	19.46	9.3847	0
38	180	3.1416	85	69.216	55.02	42.13	30.331	19.46	9.3846	0
39	135	2.3562	90	73.144	58.063	44.417	31.958	20.496	9.8819	0
40	90	1.5708	95	77.127	61.179	46.777	33.644	21.572	10.4	0
41	45	0.7854	97	78.843	62.588	47.879	34.448	22.092	10.651	0
42	0	0	100	81.059	64.227	49.07	35.276	22.611	10.899	0
43			8	9	10	11	12	13	14	15
44	deg	radian								

Heat diffusion in a circular domain with constant temperature

This problem, a particular case of the previous example, is represented in the following schema



This problem has a radial symmetry. Therefore the temperature is independent from the angle. The temperature function can be expressed by the following close formula

$$T(r) = \ln\left(\frac{R_e}{R_i}\right) \cdot \left[T_i \cdot \ln\left(\frac{r}{R_e}\right) - T_e \cdot \ln\left(\frac{r}{R_i}\right) \right]$$

It can be used for result comparing

The spreadsheet model with the complete result is

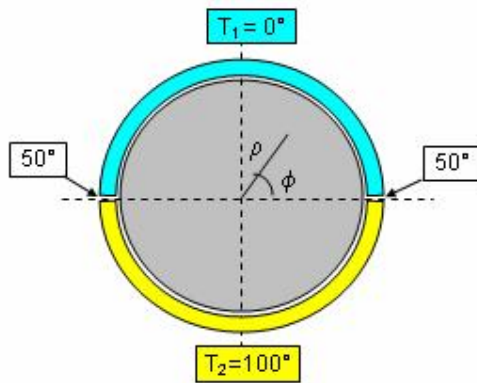
	A	B	C	D	E	F	G	H	I	J	K
23											
24	360	6.283	100	83.012	67.814	54.063	41.509	29.961	19.268	9.3127	0
25	330	5.76	100	83.012	67.813	54.063	41.509	29.961	19.268	9.3128	0
26	300	5.236	100	83.012	67.813	54.063	41.509	29.961	19.268	9.3128	0
27	270	4.712	100	83.012	67.813	54.063	41.509	29.961	19.268	9.3128	0
28	240	4.189	100	83.012	67.813	54.063	41.509	29.961	19.268	9.3128	0
29	210	3.665	100	83.012	67.813	54.063	41.509	29.961	19.268	9.3128	0
30	180	3.142	100	83.012	67.813	54.063	41.509	29.961	19.268	9.3128	0
31	150	2.618	100	83.012	67.813	54.063	41.509	29.961	19.268	9.3128	0
32	120	2.094	100	83.012	67.813	54.063	41.509	29.961	19.268	9.3128	0
33	90	1.571	100	83.012	67.813	54.063	41.509	29.961	19.268	9.3128	0
34	60	1.047	100	83.012	67.813	54.063	41.509	29.961	19.268	9.3128	0
35	30	0.524	100	83.012	67.814	54.063	41.509	29.961	19.268	9.3128	0
36	0	0	100	83.012	67.814	54.063	41.51	29.961	19.268	9.3129	0
37			8	9	10	11	12	13	14	15	16

As we can see the internal thermal distribution is constant along the vertical axis (the angle), and changes only along the horizontal axis (the radius)

Laplace polar equation over a full circular domain

The Laplace equation is not defined for $r = 0$, therefore we could not integrate it over a full circle because it contains the origin. As this domain is very common, the macro can manage also domains containing the origin: the value at the origin point is calculated taking the average of the nearest points. Let's see how it works

Assume to have to solve the following heat problem over a full plane disk.



Radius = 1 m

The distribution of the temperature is

$$T(1, q) = \begin{cases} 0 & 0 < q < p \\ 100 & p < q < 2p \end{cases}$$

At the points 0 and p the temperature is assumed the average between 0 and 100. Therefore:

$$T(1, 0) = 50, \quad T(1, p) = 50$$

The solution will look like the following

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
22														
23	360	6.283	50	50	50	50	50	50	50	50	50	50	50	50
24	330	5.76	50	53.23	56.56	60.14	64.06	68.42	73.31	78.82	85.05	92.08	100	
25	300	5.236	50	55.53	60.96	66.34	71.64	76.84	81.92	86.82	91.5	95.91	100	
26	270	4.712	50	56.35	62.46	68.32	73.89	79.13	84.02	88.56	92.73	96.54	100	
27	240	4.189	50	55.53	60.96	66.34	71.64	76.84	81.92	86.82	91.5	95.91	100	
28	210	3.665	50	53.23	56.56	60.14	64.06	68.42	73.31	78.82	85.05	92.08	100	
29	180	3.142	50	50	50	50	50	50	50	50	50	50	50	
30	150	2.618	50	46.77	43.44	39.86	35.94	31.58	26.69	21.18	14.95	7.92	0	
31	120	2.094	50	44.47	39.04	33.66	28.36	23.16	18.08	13.18	8.498	4.085	0	
32	90	1.571	50	43.65	37.54	31.68	26.11	20.87	15.98	11.44	7.274	3.463	0	
33	60	1.047	50	44.47	39.04	33.66	28.36	23.16	18.08	13.18	8.498	4.085	0	
34	30	0.524	50	46.77	43.44	39.86	35.94	31.58	26.69	21.18	14.95	7.92	0	
35	0	0	50	50	50	50	50	50	50	50	50	50	50	
36			0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	
37	deg	rad												

Select the range C23:M35 and start the macro. The values in the cells C23:L35 has been calculated by the macro. Note that the first column C23:C35 contains the same value because, really, it represents the origin point $\rho = 0$

2D Poisson Polar

This macro solves the Poisson equation in polar coordinates over a rectangular plane domain. It is similar to the Macro 2D Laplace except for the function $q(r, q)$ at the right side of the equation

$$\frac{\partial^2 U}{\partial r^2} + \frac{1}{r} \frac{\partial U}{\partial r} + \frac{1}{r^2} \frac{\partial^2 U}{\partial q^2} = q(r, q)$$

The algorithm uses the central finite differences formula

$$U_{ij} = \frac{1}{M} (a \cdot U_{ij+1} + b \cdot U_{ij-1} + c \cdot U_{i+1j} + d \cdot U_{i-1j}) - r_i \frac{h_q^2 h_r^2}{M} q_{ij}$$

	UN	
UW	U	UE
	US	

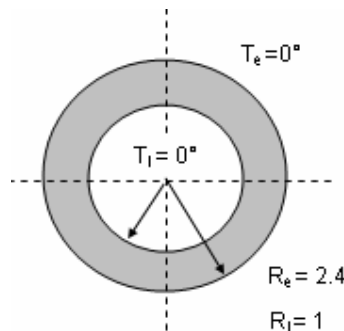
where

$$a = h_q^2 r \left(r + \frac{h_r}{2} \right) \quad b = h_q^2 r \left(r - \frac{h_r}{2} \right) \quad c = h_r^2 \quad d = h_r^2$$

$$M = 2(h_q^2 r^2 + h_r^2) \quad r_i = r_0 + (i-1)h_q$$

Poisson polar equation in a circular domain

A circular plate is uniformly heated while the internal and external boundaries are maintained at temperature of 0 °C degrees. Find the temperature distribution



$$T_e = 0 \text{ } ^\circ\text{C}$$

$$T_i = 0 \text{ } ^\circ\text{C}$$

$$R_e = 2.4 \text{ m}$$

$$R_i = 1 \text{ m}$$

Heat source: $Q = -100 \text{ k}$
where k is the conductivity

The Poisson equation is

$$\Delta^2 T = \frac{Q}{k} \Rightarrow \Delta^2 T = -100$$

Using the polar coordinates, the system can be modelled as the following spreadsheet.

At the left is shown the integration domain; at the right is shown the heat source function $Q(\rho, \theta)$ (constant in that case).

	A	B	C	D	E	F	G	H	I	J	
19											
20	360	6.283	0							0	
21	315	5.498	0							0	
22	270	4.712	0							0	
23	225	3.927	0							0	
24	180	3.142	0							0	
25	135	2.356	0							0	
26	90	1.571	0							0	
27	45	0.785	0							0	
28	0	0	0							0	
29			1	1.2	1.4	1.6	1.8	2	2.2	2.4	
30	deg	rad		radius							

	M	N	O	P	Q	R	S	T	U	V	
	360	6.283	-100	-100	-100	-100	-100	-100	-100	-100	
	315	5.498	-100	-100	-100	-100	-100	-100	-100	-100	
	270	4.712	-100	-100	-100	-100	-100	-100	-100	-100	
	225	3.927	-100	-100	-100	-100	-100	-100	-100	-100	
	180	3.142	-100	-100	-100	-100	-100	-100	-100	-100	
	135	2.356	-100	-100	-100	-100	-100	-100	-100	-100	
	90	1.571	-100	-100	-100	-100	-100	-100	-100	-100	
	45	0.785	-100	-100	-100	-100	-100	-100	-100	-100	
	0	0	-100	-100	-100	-100	-100	-100	-100	-100	
			1	1.2	1.4	1.6	1.8	2	2.2	2.4	
	deg	rad		radius							

Select the range C20:J28 and start the macro **2D-Poisson polar**. Fill the empty field $Q(\rho, \theta)$ with the range O20:V28 and press "Run". The result will look like the following

	A	B	C	D	E	F	G	H	I	J	
19											
20	360	6.283	0	13.75	21.7	24.85	23.87	19.2	11.16	0	
21	315	5.498	0	13.75	21.7	24.85	23.87	19.2	11.16	0	
22	270	4.712	0	13.75	21.7	24.85	23.87	19.2	11.16	0	
23	225	3.927	0	13.75	21.7	24.85	23.87	19.2	11.16	0	
24	180	3.142	0	13.75	21.7	24.85	23.87	19.2	11.16	0	
25	135	2.356	0	13.75	21.7	24.85	23.87	19.2	11.16	0	
26	90	1.571	0	13.75	21.7	24.85	23.87	19.2	11.16	0	
27	45	0.785	0	13.75	21.7	24.85	23.87	19.2	11.16	0	
28	0	0	0	13.75	21.7	24.85	23.87	19.2	11.16	0	
29			1	1.2	1.4	1.6	1.8	2	2.2	2.4	
30	deg	rad		radius							

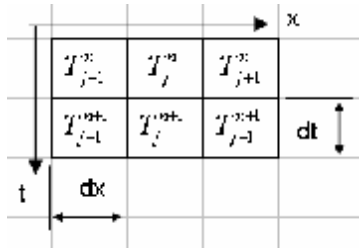
Note that, because of the radial symmetry, the temperature distribution is independent from the angle.

1D Heat Diffusion

This macro solves the mono-dimensional *diffusion equation* over a plane domain
 This PDE is also known as *heat diffusion* equation or also *Fourier* equation

$$\frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2}$$

We can specify the rectangular "x-t" domain and boundary conditions by coloured cells.
 The macro assumes the following schema



The time axis lies in vertical while the x axis lies in horizontal

The grid steps are the scale factors dx and dt
 The macro calculates the steps from the scales set at the border of the domain. If omitted the macro assumes dx = 1 and dt = 1

Notation:

$$T_j^n = T(j \cdot (dx), n \cdot (dt))$$

The macro can use two different implicit formulas.

$$\frac{T_j^{n+1} - T_j^n}{dt} = k \left[(1-q) \frac{T_{j+1}^n - 2T_j^n + T_{j-1}^n}{(dx)^2} + q \frac{T_{j+1}^{n+1} - 2T_j^{n+1} + T_{j-1}^{n+1}}{(dx)^2} \right]$$

where:

- $q = 1/2$ for Crank-Nicholson formula
- $q = 1$ for Euler implicit formula

The solution is found by the iterative Gauss-Seidel algorithm

Thermal diffusion along a bar

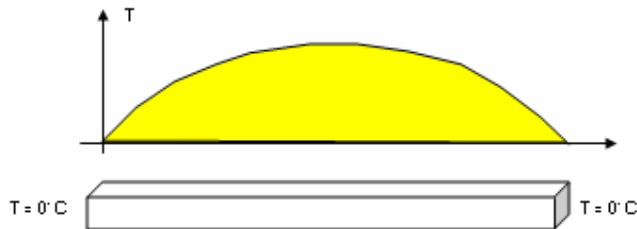
The extremes of a metallic bar, having a length $L = 1$, are maintained at 0°C temperature. The temperature profile along the bar at the initial time is given. Find the thermal evolution for $0 < t < 0.2$ assuming a diffusivity constant $k = 2$

Boundary conditions

$$T(x,0) = \sin\left(\frac{\pi}{L}x\right)$$

$$T(0,t) = 0$$

$$T(L,t) = 0$$



The problem can be modelled as shown in the following spreadsheet

Thermal diffusion along a bar with insulated boundary

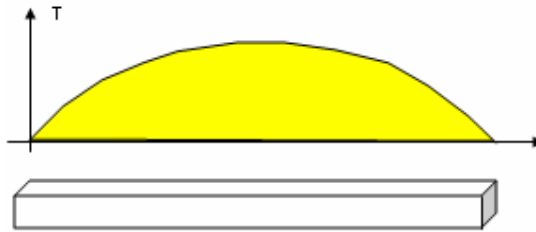
A metallic bar, having a length $L = 1$, has its extremes totally isolated. The temperature profile along the bar at the initial time is given. Find the thermal evolution for $0 < t < 0.2$ sec assuming a diffusivity constant $k = 1$

Boundary conditions

$$T(x,0) = \sin\left(\frac{\pi}{L}x\right)$$

$$T_x(0,t) = 0$$

$$T_x(L,t) = 0$$



The boundary conditions mean that there is no heat flux at $x = 0$ and $x = L$

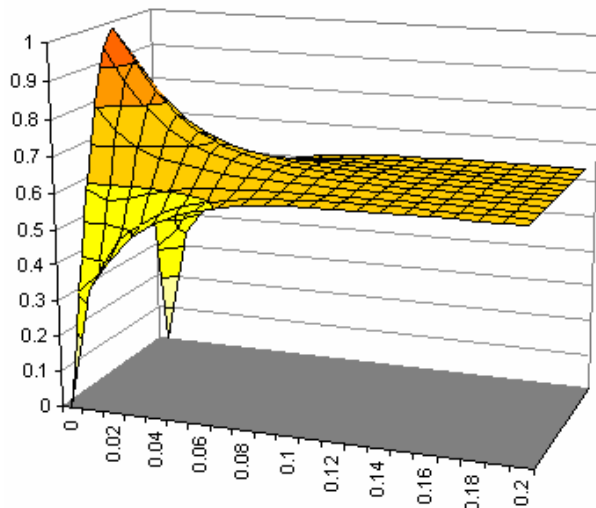
The problem can be modelled as shown in the following spreadsheet

	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	z
13													
14		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	
15	0	0	0.309	0.588	0.809	0.951	1	0.951	0.809	0.588	0.309	0	
16	0.01	0.34	0.371	0.557	0.74	0.864	0.908	0.864	0.74	0.557	0.371	0.34	
17	0.02	0.422	0.472	0.57	0.696	0.793	0.829	0.793	0.696	0.57	0.472	0.422	
18	0.03	0.496	0.52	0.59	0.674	0.741	0.767	0.741	0.674	0.59	0.52	0.496	
19	0.04	0.538	0.557	0.603	0.66	0.706	0.724	0.706	0.66	0.603	0.557	0.538	
20	0.05	0.569	0.58	0.612	0.651	0.682	0.694	0.682	0.651	0.612	0.58	0.569	
21	0.06	0.589	0.597	0.618	0.645	0.666	0.674	0.666	0.645	0.618	0.597	0.589	
22	0.07	0.602	0.608	0.622	0.64	0.655	0.66	0.655	0.64	0.622	0.608	0.602	
23	0.08	0.612	0.615	0.625	0.637	0.647	0.651	0.647	0.637	0.625	0.615	0.612	
24	0.09	0.618	0.621	0.627	0.636	0.642	0.645	0.642	0.636	0.627	0.621	0.618	
25	0.1	0.622	0.624	0.629	0.634	0.639	0.64	0.639	0.634	0.629	0.624	0.622	
26	0.11	0.625	0.626	0.629	0.633	0.636	0.638	0.636	0.633	0.629	0.626	0.625	
27	0.12	0.627	0.628	0.63	0.633	0.635	0.636	0.635	0.633	0.63	0.628	0.627	
28	0.13	0.629	0.629	0.63	0.632	0.634	0.634	0.634	0.632	0.631	0.629	0.629	
29	0.14	0.629	0.63	0.631	0.632	0.633	0.633	0.633	0.632	0.631	0.63	0.629	
30	0.15	0.63	0.63	0.631	0.632	0.632	0.633	0.632	0.632	0.631	0.63	0.63	
31	0.16	0.63	0.631	0.631	0.632	0.632	0.632	0.632	0.632	0.631	0.631	0.63	
32	0.17	0.631	0.631	0.631	0.632	0.632	0.632	0.632	0.632	0.631	0.631	0.631	
33	0.18	0.631	0.631	0.631	0.632	0.632	0.632	0.632	0.632	0.631	0.631	0.631	
34	0.19	0.631	0.631	0.631	0.631	0.632	0.632	0.632	0.631	0.631	0.631	0.631	
35	0.2	0.631	0.631	0.631	0.631	0.632	0.632	0.632	0.631	0.631	0.631	0.631	
36													

The model is similar to the previous example except for the coloured cells.

In that case we have left uncoloured the border cells AF16:AF35 and AP16:AP35.

In this way the macro calculate these values assuming $dT/dx = 0$ at the left and right borders.



The 3D-dimensional graph is quite different from the previous example. In that case, after a short transient, the temperature stabilizes itself around the value of 0.63.

Thermal transient of a plate

A large metallic plate, having a thickness $L = 0.12$ m is immersed into oil at 260°C . The initial temperature of the plate is 38°C . Find the thermal evolution for $0 < t < 60$ sec assuming a diffusivity constant $k = 4\text{E-}5$ (m^2/sec)

Because the thickness of the plate is much lower respect to the other dimensions we can assume that all thermal flux crosses the upper and lower surface. With this assumption the problem can be turned in a mono-dimensional system having the x-axis along the thickness of the plate

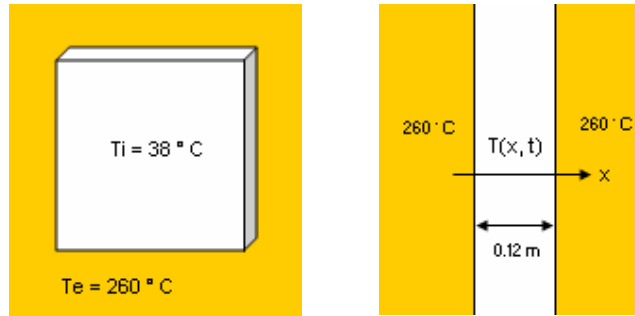
$$\frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2}$$

Boundary conditions

$$T(x,0) = 38$$

$$T(0,t) = 260$$

$$T(L,t) = 260$$



The boundary conditions mean that the surfaces temperature remains at 260°C for all the transient. The initial internal temperature of the plate is 38°C . Of course for $t > 0$ the temperature of the plate evolves until it matches the oil temperature.

Let's see how modelling this problem. We discretize the x-axis with a step of 0.015 m and the t-axis with a step 5 sec. This means that we will get a thermal profile sampled every 5 sec.

	A	B	C	D	E	F	G	H	I	J	K
5											
6		0	0.015	0.03	0.045	0.06	0.075	0.09	0.105	0.12	
7	0	38	38	38	38	38	38	38	38	38	
8	5	260									260
9	10	260									260
10	15	260									260
11	20	260									260
12	25	260									260
13	30	260									260
14	35	260									260
15	40	260									260
16	45	260									260
17	50	260									260
18	55	260									260
19	60	260									260
20											

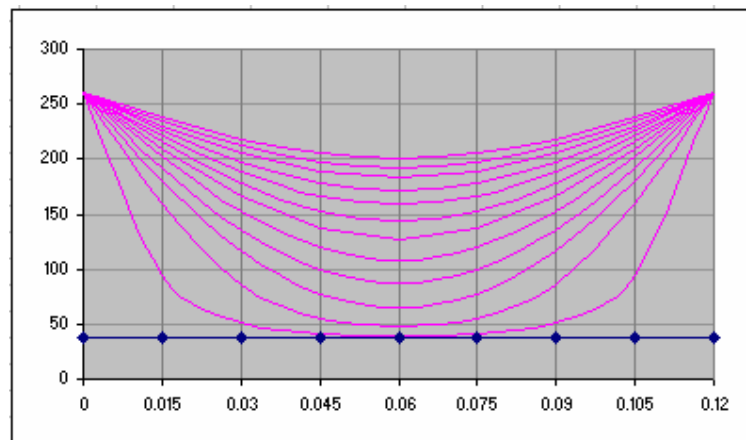
Now select the range B7:J19 and start the macro **1D Heat Diffusion**

Insert the $K = 4\text{E-}5$ parameter and press "Run"

If you have chosen the Crank-Nicolson algorithm, probably the result will be like the following

	A	B	C	D	E	F	G	H	I	J
5										
6		0	0.015	0.03	0.045	0.06	0.075	0.09	0.105	0.12
7	0	38	38	38	38	38	38	38	38	38
8	5	260	93.513	51.929	41.685	39.734	41.6855	51.929	93.5127	260
9	10	260	160.22	85.617	55.473	48.059	55.473	85.617	160.217	260
10	15	260	179.26	116.19	77.436	65.372	77.4358	116.19	179.26	260
11	20	260	192.22	135.94	99.789	87.245	99.7894	135.94	192.222	260
12	25	260	201.48	152.28	119.73	108.44	119.73	152.28	201.478	260
13	30	260	209.11	166.08	137.44	127.4	137.443	166.08	209.113	260
14	35	260	215.62	178.03	152.95	144.15	152.946	178.03	215.621	260
15	40	260	221.26	188.44	166.51	158.81	166.509	188.44	221.265	260
16	45	260	226.18	197.51	178.36	171.63	178.357	197.51	226.179	260
17	50	260	230.47	205.43	188.7	182.83	188.704	205.43	230.468	260
18	55	260	234.21	212.35	197.74	192.61	197.741	212.35	234.211	260
19	60	260	237.48	218.39	205.63	201.15	205.632	218.39	237.48	260
20										

Each row contains the function values $T(x_i)$ at each step time $t_i = (5, 10, 15... 60 \text{ sec})$. Plotting them, we have the following thermal transient graph



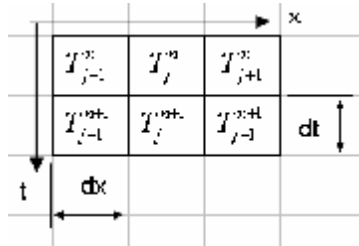
Note that the curves are obtained by parabolic interpolation. Actually we have calculated the values by FD only at the grid points: $x_i = (0, 0.015, 0.03, 0.045, 0.06...)$

1D Heat Convection

This macro solves the mono-dimensional *convection* equation over a plane domain
 This pde is also known as *heat convection* or *linear wave* equation

$$\frac{\partial T}{\partial t} = -a \frac{\partial T}{\partial x}$$

We can specify the rectangular "x-t" domain and boundary conditions by coloured cells.
 The macro assumes the following scheme



The time axis lies in vertical while the x-axis lies in horizontal

The grid steps are the scale factors dx and dt
 The macro calculates the steps from the scales at the border of the domain. If omitted, the macro assumes dx = 1 and dt = 1

Notation:

$$T_j^n = T(j \cdot (dx), n \cdot (dt))$$

The macro uses three different formulas.

$$\frac{T_j^{n+1} - T_j^n}{dt} = -a \left[\frac{T_{j+1}^{n+1} - T_{j-1}^{n+1}}{dx} + \frac{T_{j+1}^n - T_{j-1}^n}{dx} \right]$$

Crank-Nicolson unconditional stable

$$\frac{T_j^{n+1} - T_j^n}{dt} = -a \left[\frac{T_{j+1}^n - T_j^n}{dx} \right]$$

Upwind stable for C < 1

$$\frac{T_j^{n+1} - T_j^n}{dt} = -a \left[\frac{T_{j+1}^n - T_{j-1}^n}{dx} \right] - \frac{a^2 (dt)}{2} \left[\frac{T_{j+1}^n - 2T_j^n + T_{j-1}^n}{(dx)^2} \right]$$

Lax-Wendroff stable for C < 1

where C is the so called Courant- number $C = a \frac{dt}{dx}$ (Courant-Friedrichs-Lewy criterion: C < 1)

The solution is found by the iterative Gauss-Seidel algorithm

Linear Convection of a truncated sine wave

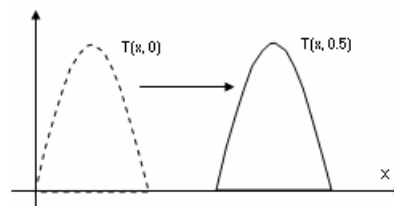
Solve the following wave problem for $t \leq 0.8$ sec

$$\frac{\partial T}{\partial t} = -a \frac{\partial T}{\partial x}$$

$$a = 1$$

$$T(x,0) = \sin(5\pi / 2 \cdot x) \quad 0 \leq x \leq 0.4$$

$$T(0,t) = 0$$



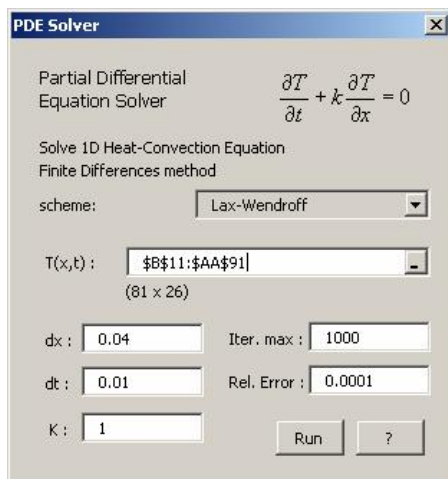
At the initial time $t = 0$, the wave is $T(x, 0)$. For $t > 0$ the shape shifts along the x-axis

The problem can be solved with FD method on the worksheet but it requires a quite larger domain in order to appreciate the result.

In the following we see only a little portion of the entire domain that is B11:AA91. It is divided in a grid with $dx = 0.04$ and $dt = 0.01$.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
10		0	0.04	0.08	0.12	0.16	0.2	0.24	0.28	0.32	0.36	0.4	0.44	0.48	0.52
11	0	0	0.303	0.588	0.809	0.951	1	0.951	0.809	0.588	0.303	0	0	0	0
12	0.01	0													
13	0.02	0													
14	0.03	0													
15	0.04	0													
16	0.05	0													
17	0.06	0													
18	0.07	0													
19	0.08	0													
20	0.09	0													
21	0.1	0													
22	0.11	0													
23	0.12	0													
24	0.13	0													
25	0.14	0													
26	0.15	0													
27	0.16	0													
28	0.17	0													
29	0.18	0													

In row 10, at the top of the domain, insert the x-scale and, just below, in the first row of the domain, insert the function $T(x, 0) = \text{If}(x < 0.4, \sin(2.5 \cdot \pi \cdot x), 0)$
 Fill the first column of the domain, B11:B91 with 0. Then colour the first row and the first column. This obligates the macro do not alter these cells. Now select the range B11:AA91 and start the macro **1D- Heat convection**.



Note that, with this grid, the Couran number $C = 1 \cdot 0.01/0.04 = 0.25 < 1$ satisfies the stability criterion for every schema.

Insert the coefficient $k = 1$ and chose the schema that you want, for example the Lax-Wendroff

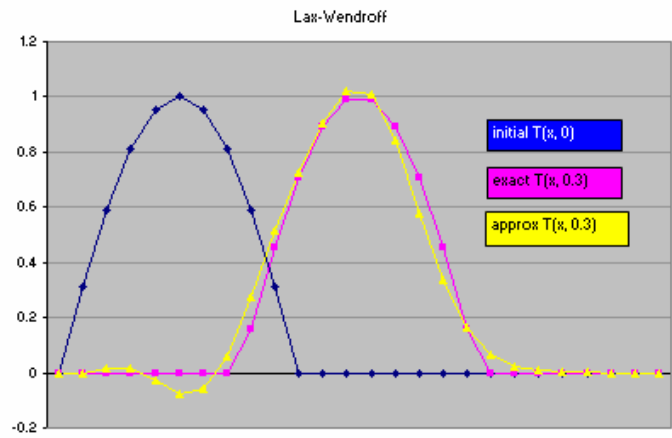
Press "Run"

Try with different schema

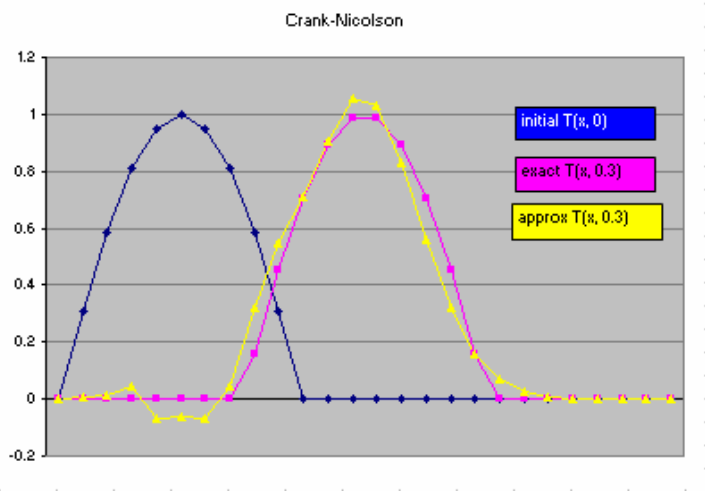
Every row of the domain is a function at the time. We can plot these functions, at different instants ($t = 0, t = 0.3$) in order to see how the wave propagates itself and comparing the calculated values with the exact solution

$$T(x, t) = \sin(5\pi / 2 \cdot (x - t)) \quad 0 \leq (x - t) \leq 0.4$$

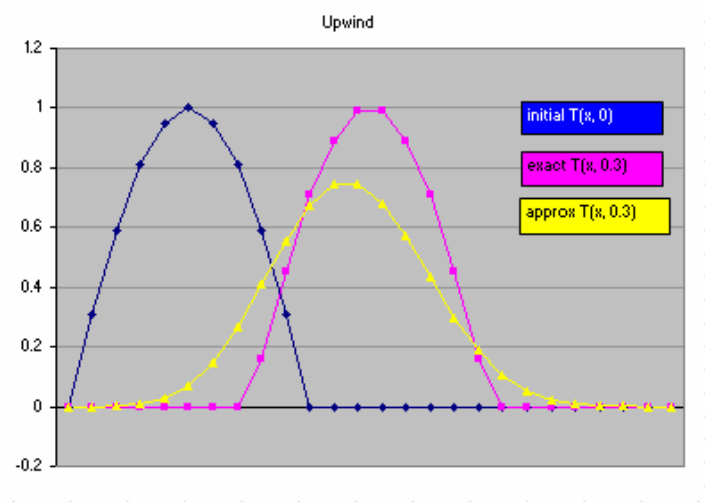
The better schema? There is not. Each algorithm has advantages and disadvantages. The Upwind seems less accurate but it avoids the unwanted oscillations that could be raised by the two other algorithms. Therefore you have to examine the result and decide what method is more adapted for your problem and your scope.



We can solve the problem with a different algorithm in order to compare the result
Here the result with Crank-Nicolson scheme



And here the solution obtained with the Upwind schema



1D Wave

This macro solves the mono-dimensional *wave* equation over a plane domain

$$\frac{\partial^2 j}{\partial t^2} = v^2 \frac{\partial^2 j}{\partial x^2}$$

We can specify the rectangular "x-t" domain and boundary conditions by coloured cells.
With the usual notation

$$j_j^n = j(j \cdot (dx), n \cdot (dt))$$

The macro uses the following formula.

$$j_j^{n+1} = v^2 \left(\frac{dt}{dx} \right)^2 (j_{j+1}^n - 2j_j^n + j_{j-1}^n) + 2j_j^n - j_j^{n-1} \quad \text{explicit}$$

The solution is found by the iterative Gauss-Seidel algorithm
Let's see how it works with the following examples

Wire oscillations

A wire of length $L = 1$, is fixed to its extremes. Its initial profile is $f(x)$. Find the transient for $t \leq 0.5$

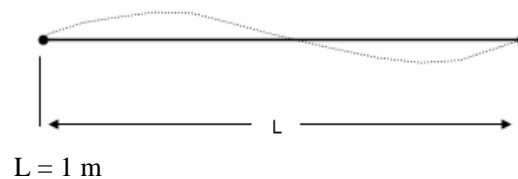
$$\frac{\partial^2 j}{\partial t^2} = v^2 \frac{\partial^2 j}{\partial x^2}$$

$$v = 4$$

$$j(x, 0) = f(x) = 6 \sin(\pi \cdot x) - 3 \sin(4\pi \cdot x)$$

$$j(0, t) = 0$$

$$j(L, t) = 0$$



The problem can be solved with FD method on the worksheet

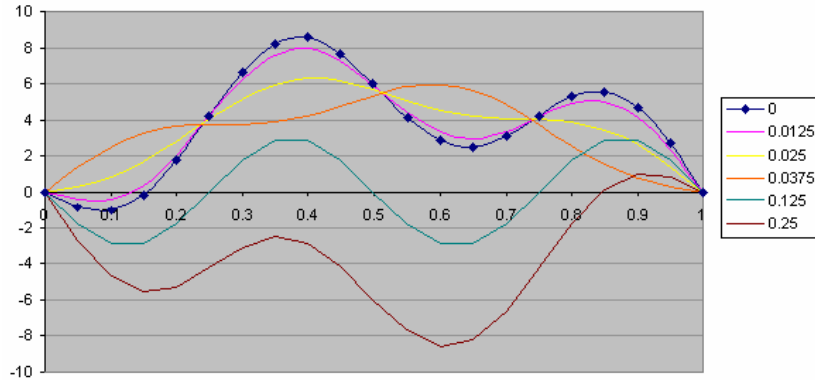
In the following figure we see a little portion of the entire domain, divided in grid having $dx = 0.05$ and $dt = 0.0125$. The entire range is B11:V51. Select it and start the macro **1D - wave**

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
9																							
10			0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
11	0	0	-0.82	-1	-0.13	1.76	4.24	6.62	8.2	8.56	7.69	6	4.16	2.85	2.49	3.09	4.24	5.29	5.58	4.71	2.7	0	
12	0.0125	0																					0
13	0.025	0																					0
14	0.0375	0																					0
15	0.05	0																					0
16	0.0625	0																					0
17	0.075	0																					0
18	0.0875	0																					0

Insert only the coefficients $v^2 = 16$ and click "Run"

After few time the problem is solved. Each row represents the wire configuration at the given time

The following plot shows some of these configurations at different instants



Compare these results with the exact solution:

$$j(x,t) = 6 \sin(p \cdot x) \cos(4p \cdot t) - 3 \sin(4p \cdot x) \cos(16p \cdot t)$$

The guitar string

A string of length $L = 1$, is fixed to its extremes. The string is pulled at $x = 0.25$ for $d = 0.1$ and then leaved free for $t = 0$. Find the successive positions of the string for $0 \leq t \leq 0.25$

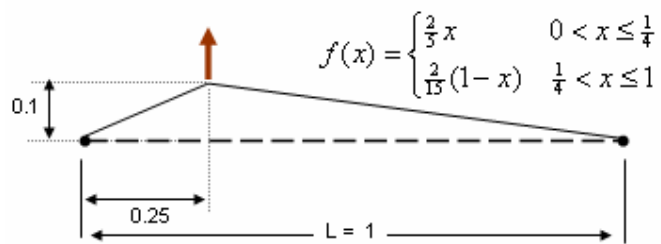
$$\frac{\partial^2 j}{\partial t^2} = v^2 \frac{\partial^2 j}{\partial x^2}$$

$$v = 4$$

$$j(x,0) = f(x)$$

$$j(0,t) = 0$$

$$j(L,t) = 0$$

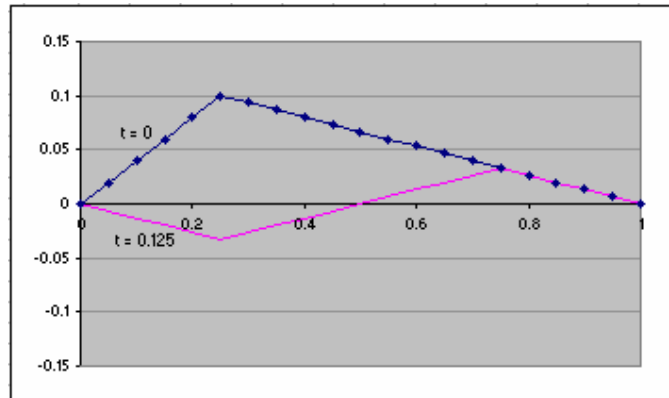


In the following figure we see a little portion of the entire range B11:V51. It is divided in grid having $dx = 0.05$ and $dt = 0.0125$. Select it and start the macro **1D - wave**

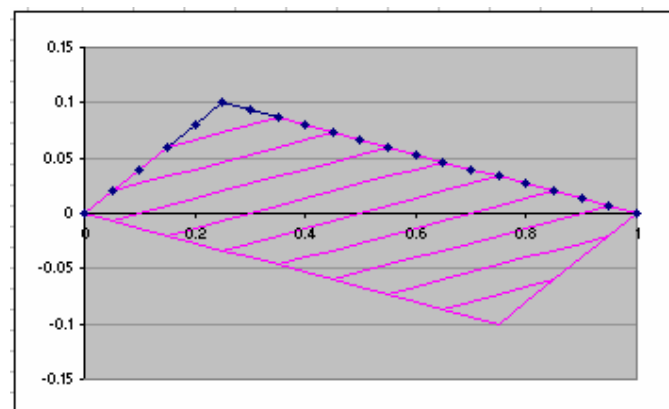
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
9																							
10		0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1	
11	0	0	0.02	0.04	0.06	0.08	0.1	0.09	0.09	0.08	0.07	0.07	0.06	0.05	0.05	0.04	0.03	0.03	0.02	0.01	0.01	-0	
12	0.0125	0																					0
13	0.025	0																					0
14	0.0375	0																					0
15	0.05	0																					0
16	0.0625	0																					0
17	0.075	0																					0

Insert only the coefficients $v^2 = 16$ and click "run"

After few time the problem is solved. Each row represents the string configuration at the given time
The following plot shows the string shapes at $t = 0$ and $t = 0.125$



The following plot shows the string shapes at $0 \leq t \leq 0.25$ with step of $\Delta t = 0.025$



As we can see, after a time of 0.25 s, the shape of the string is symmetric respect to the middle point $x = 0.5$

Compare these results with the aid of the following exact formula

$$j(x,t) = \sum_{n=1}^{\infty} \frac{4}{p^2 n^2} \sin\left(\frac{1}{4} np\right) \cdot \sin(np \cdot x) \cdot \cos(nvp \cdot t)$$

Note that the above formula is expressed as an infinite serie. For computing numerically the function you have necessarily to take finite terms. We suggest to take 10 terms at least In order to have a good approximation.

Linear ODE (BC) with FD

The Finite Differences method can be used also for solving ordinary differential equations with boundary conditions. This macro solves the following 2nd order Linear ODE problem.

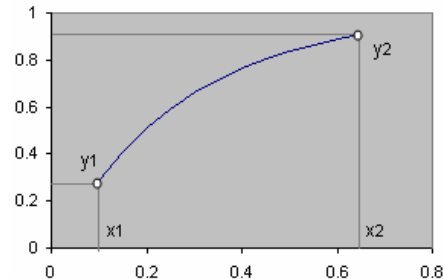
$$y'' + a_1 y' + a_0 y = b$$

$$x_1 \leq x \leq x_2$$

$$y(x_1) = y_1$$

$$y(x_2) = y_2$$

where the coefficients a_0 , a_1 , b can be functions of the variable x



To get how this macro works see these examples

2nd Linear ODE with boundary conditions

Solve the following differential equation with boundary conditions.

$$y'' - \frac{2}{x+1} y' + \frac{8}{x+2} y = -9 \cdot \frac{4x^4 + 7x - 2}{4x^2 + 12x + 8}$$

with:

$$y(0) = 0$$

$$y(1) = 0$$

The differential equation is linear of the 2nd order, having the following coefficients

$$a_0 = \frac{8}{x+2} \quad a_1 = -\frac{2}{x+1} \quad b = -9 \cdot \frac{4x^4 + 7x - 2}{4x^2 + 12x + 8}$$

In a spreadsheet prepare the following schema

	A	B	C	D	E	F	G
7	x	a0(x)	a1(x)	b(x)	y	y'	y''
8	0	4	-2	2.25	0		
9	0.05	3.9024	-1.9048	1.7247			
10	0.1	3.8095	-1.8182	1.2658			
11	0.15	3.7209	-1.7391	0.8627			
12	0.2	3.6364	-1.6667	0.5059			
13	0.25	3.5556	-1.6	0.1875			
14	0.3	3.4783	-1.5385	-0.0996			
15	0.35	3.4043	-1.4815	-0.3617			
16	0.4	3.3333	-1.4286	-0.6043			
17	0.45	3.2653	-1.3793	-0.8322			
18	0.5	3.2	-1.3333	-1.05			
19	0.55	3.1373	-1.2903	-1.2615			
20	0.6	3.0789	-1.25	-1.4703			
21	0.65	3.0189	-1.2121	-1.6796			
22	0.7	2.963	-1.1765	-1.8924			
23	0.75	2.9091	-1.1429	-2.1112			
24	0.8	2.8571	-1.1111	-2.3386			
25	0.85	2.807	-1.0811	-2.5767			
26	0.9	2.7586	-1.0526	-2.8276			
27	0.95	2.7119	-1.0256	-3.0931			
28	1	2.6667	-1	-3.375	0		

The first column contains the x values; the column 2, 3, and 4 contain respectively the coefficients: $a_0(x)$, $a_1(x)$ and $b(x)$

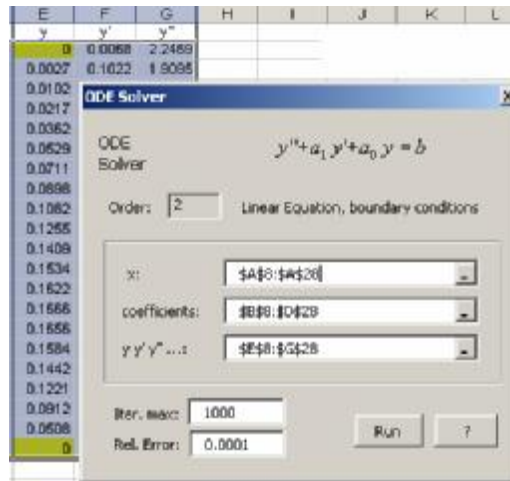
The last three columns will contain respectively the values of the unknown function $y(x)$, the first derivatives $y'(x)$ and the 2nd derivative $y''(x)$

The area E8:G28 will be filled by the macro except the boundary cells (coloured).

Put the boundary conditions $y(0) = 0$ $y(1) = 0$ into the first and last cell E8 and E28

Now select the entire range A8:G28 and start the macro **ODE-2 Linear BC**

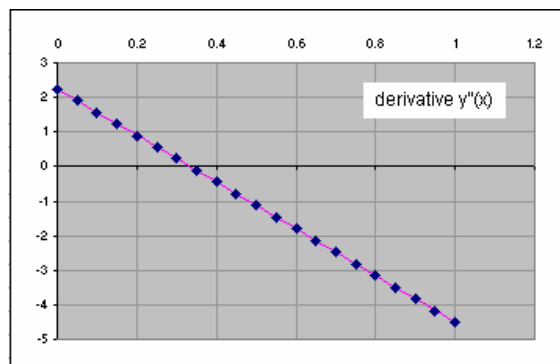
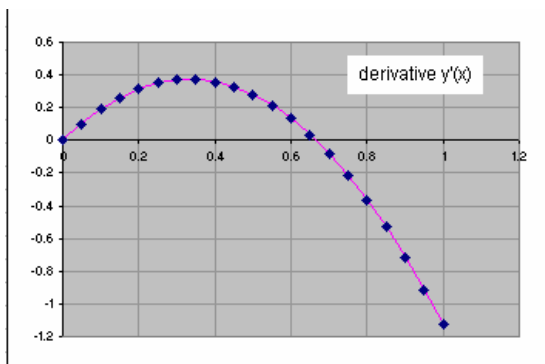
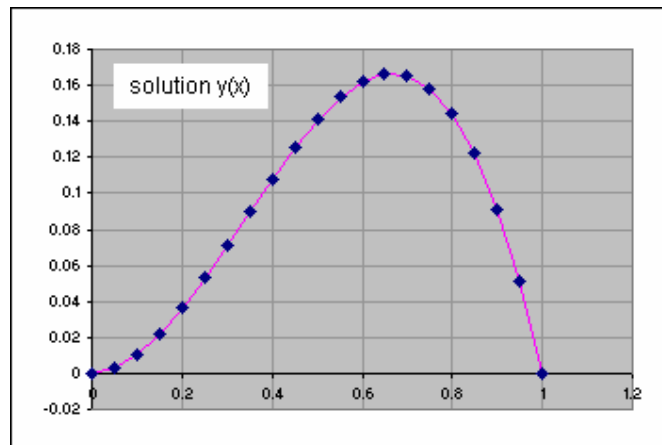
If you have arranged the worksheet as the above one, all the input fields will be correctly filled.
 You only have to press "Run"



The plot of the solution functions y , y' , y'' are shown in the following images
 Compare them with the exact solution

$$y = \frac{9}{8}(x^2 - x^3)$$

The dot points are the calculated solution; the exact function is the pink line



Linear ODE (IC) with FD

The Finite Differences method can be used also for solving ordinary differential equations with initial conditions. This macro solves the Linear ODE problem of 1st, 2nd and 3rd order.

$$y'''+a_2 y''+a_1 y'+a_0 y = b$$

$$y(x_0) = y_0$$

$$y'(x_0) = y_{10}$$

$$y''(x_0) = y_{20}$$

Where the coefficients a_0, a_1, a_2, b can be functions of the variable x

To get how this macro works see these examples

Linear, 1st order, differential equation with Initial condition

Solve the following differential equation.

$$y'+\frac{1}{x+1}y = \frac{p \cos(p x)}{x+1}$$

with

$$y(0) = 0 \quad 0 \leq x \leq 4$$

The differential equation is linear of the 1st order, having the following coefficients

$$a_0 = \frac{1}{x+1} \quad b = \frac{p \cos(p x)}{x+1}$$

In a spreadsheet prepare the following schema

	A	B	C	D	E
4	x	a0	b	y	y'
5	0	1	3.1416	0	
6	0.2	0.8333	2.118		
7	0.4	0.7143	0.6934		
8	0.6	0.625	-0.6068		
9	0.8	0.5556	-1.412		
10	1	0.5	-1.5708		
11	1.2	0.4545	-1.1553		
12	1.4	0.4167	-0.4045		
13	1.6	0.3846	0.3734		
14	1.8	0.3571	0.9077		
15	2	0.3333	1.0472		
16	2.2	0.3125	0.7943		
17	2.4	0.2941	0.2855		
18	2.6	0.2778	-0.2697		
19	2.8	0.2632	-0.6688		
20	3	0.25	-0.7854		
21	3.2	0.2381	-0.6051		
22	3.4	0.2273	-0.2206		
23	3.6	0.2174	0.211		
24	3.8	0.2083	0.5295		
25	4	0.2	0.6283		

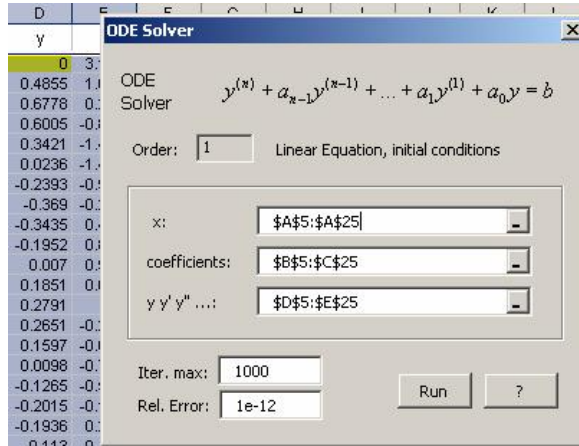
The first column contains the x values;
The columns 2, and 3 contain respectively the coefficients: $a_0(x)$ and $b(x)$

The last two columns will contain respectively the values of the unknown functions: $y(x)$, $y'(x)$

The area D5:E25 will be filled by the macro except the initial cell (coloured).
Put the initial condition $y(0) = 0$ in the first cell D5.

Now select the entire range A5:E25 and start the macro **ODE- Linear IC**

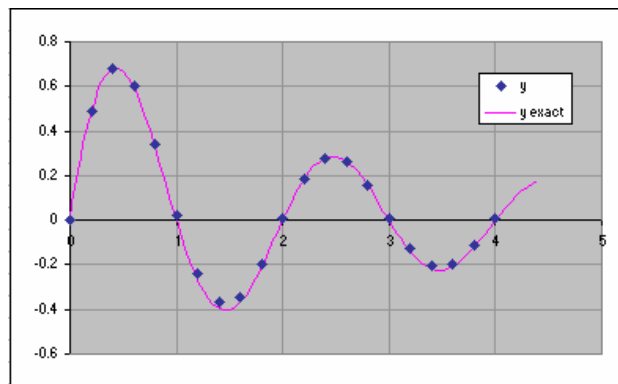
If you have arranged the worksheet as the above, all the input fields will be correctly filled.
You only have to press "Run"



The plot of the solution function y and y' are shown in the following images
Compare them with the exact solution

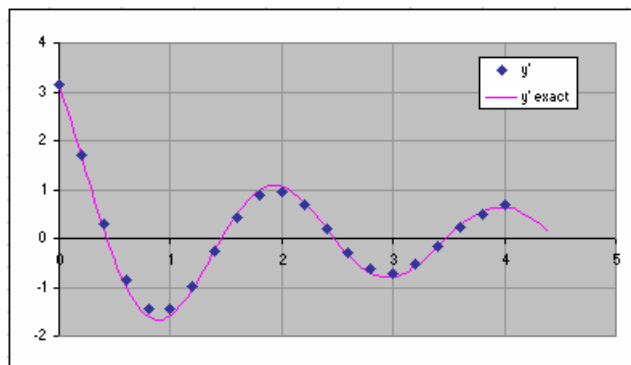
$$y = \frac{\sin(p \cdot x)}{x+1}$$

The dot points are the calculated solution; the exact function is the pink line



Note that we have intentionally adopted a large step for putting in evidence a little difference. If we reduce the step grid the difference will be sharply reduced and the two curves will match perfectly.

The derivative function y' is shown in the following graph



Linear, 2nd order, differential equation with Initial condition

Solve the following differential equation with initial conditions.

$$y'' + (x+1)y' + 2y = \frac{6}{(x+1)^4}$$

$$y(0) = 1$$

$$y'(0) = -2$$

$$0 \leq x \leq 4$$

The differential equation is linear of the 2nd order, having the following coefficients

$$a_0 = 2 \quad a_1 = x+1 \quad b = \frac{6}{(x+1)^4}$$

In a spreadsheet prepare the following schema

	A	B	C	D	E	F	G
4	x	a0	a1	b	y	y'	y''
5	0	2	1	6	1	-2	
6	0.1	2	1.1	4.098			
7	0.2	2	1.2	2.894			
8	0.3	2	1.3	2.101			
9	0.4	2	1.4	1.562			
10	0.5	2	1.5	1.185			
11	0.6	2	1.6	0.916			
12	0.7	2	1.7	0.718			
13	0.8	2	1.8	0.572			
14	0.9	2	1.9	0.46			
15	1	2	2	0.375			
16	1.1	2	2.1	0.309			
17	1.2	2	2.2	0.256			
18	1.3	2	2.3	0.214			
19	1.4	2	2.4	0.181			
20	1.5	2	2.5	0.154			
21	1.6	2	2.6	0.131			
22	1.7	2	2.7	0.113			
23	1.8	2	2.8	0.098			
24	1.9	2	2.9	0.085			

The first column contains the x values; the columns 2, 3 and 4 contain respectively the coefficients: $a_0(x)$, $a_1(x)$ and $b(x)$

The last three columns will contain respectively the values of the unknown functions: $y(x)$, $y'(x)$, $y''(x)$

The area E5:G45 will be filled by the macro except the initial cells (coloured).

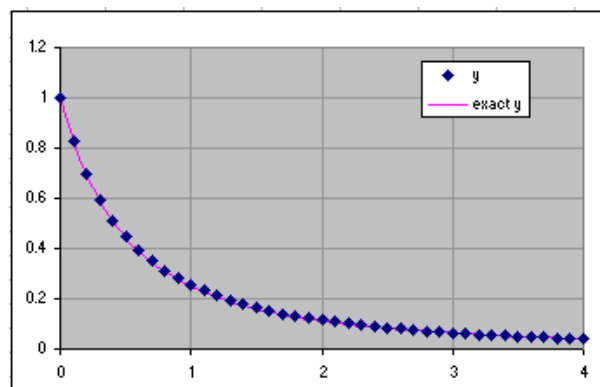
Put the initial conditions values into the first cells E5 and F5.

Now select the entire range A5:G45 and start the macro **ODE- Linear IC**

The plot of the solution function y and y' are shown in the following images
Compare them with the exact solution

$$y = \frac{1}{(x+1)^2}$$

The dot points are the calculated solution; the exact function is the pink line



Linear, 3rd order, differential equation with Initial condition

Solve the following differential equation with initial conditions.

$$y''' + y'' + y' + y = 4(x+1) \cdot e^{-x}$$

$$y(0) = 0$$

$$y'(0) = 0$$

$$y''(0) = 2$$

$$0 \leq x \leq 8$$

The differential equation is linear of the 3rd order, having the following coefficients

$$a_0 = 1 \quad a_1 = 1 \quad a_2 = 1 \quad b = 4(x+1) \cdot e^{-x}$$

In a spreadsheet prepare the following schema

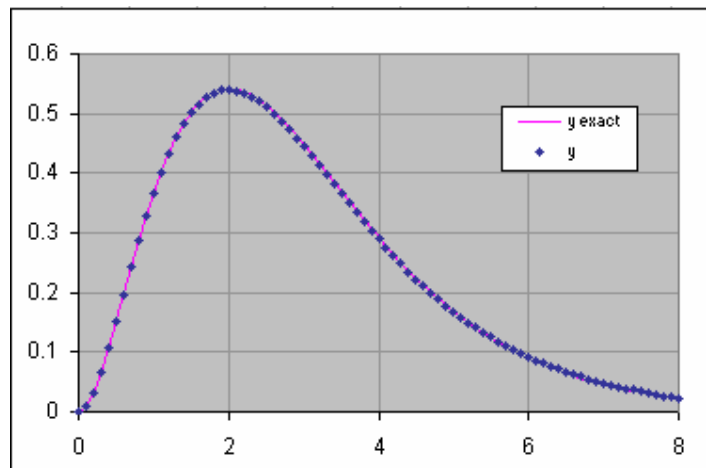
	A	B	C	D	E	F	G	H	I
7	x	a0	a1	a2	b	y	y'	y''	y'''
8	0	1	1	1	-4	0	0	2	
9	0.1	1	1	1	-3.257				
10	0.2	1	1	1	-2.62				
11	0.3	1	1	1	-2.074				
12	0.4	1	1	1	-1.609				
13	0.5	1	1	1	-1.213				
14	0.6	1	1	1	-0.878				
15	0.7	1	1	1	-0.596				
16	0.8	1	1	1	-0.359				
17	0.9	1	1	1	-0.163				
18	1	1	1	1	-2E-16				
19	1.1	1	1	1	0.133				
20	1.2	1	1	1	0.241				
21	1.3	1	1	1	0.327				
22	1.4	1	1	1	0.395				
23	1.5	1	1	1	0.446				

The first column contains the x values;
 The columns 2, 3, 4 and 5 contain respectively the coefficients: $a_0(x)$, $a_1(x)$, $a_2(x)$ and $b(x)$
 The last three columns will contain respectively the values of the unknown functions: $y(x)$, $y'(x)$, $y''(x)$, $y'''(x)$

The area F8:I88 will be filled by the macro except the initial cells (coloured).

Put the initial conditions values into the first cells E8, G8 and H8.

Now select the entire range A5:G45 and start the macro **ODE- Linear IC**
 The plot of the solution function y and y' are shown in the following images
 The dot points are the calculated solution; the exact function is the pink line



Compare them with the exact solution

$$y = x^2 \cdot e^{-x}$$

FD in Excel

The goal of the finite-differences method is to replace continuous derivatives with difference formulas that involve only the discrete values associated with the position on the mesh.

The discrete approximation results in a set of algebraic equations that can be solved with iterative approximate methods like Jordan or Gauss-Seidel or also with direct decomposition methods like Gauss, LU, Thomas, etc..

Both classes of method have advantages and disadvantages. Direct methods are usually faster, but iterative methods are simpler and can be performed directly "on-site".

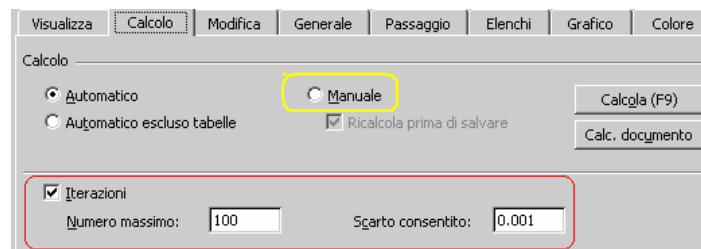
The iterative mode

The spreadsheet can perform iterative methods, but before that, we have to switch on the iterative mode parameter. Activate the iterative mode in Excel from the menu Tools/Options...

Stopping criterion. Usually the spreadsheet has two criteria for stopping the iterative process.

The max iterations limit (usually 100): the spreadsheet breaks the process when the number of iterations has reached this limit. We can restart the process simply pressing the F9 key.

The min changes limit (usually 0.001): the spreadsheet ends the process when the changes of the cells are less than this limit.

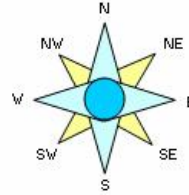


Manual Computing. Iterative mode is quite time consuming. Sometime, especially for large range of cells it is preferable to switch off the automatic computing. In that case we will start the calculus by the F9 key.

FD Formulas

Assuming the "Cardinal Points" rule, the cells around the central cell $T(i, j) = T_O$ are called $T_{NW}, T_W, T_{SW}, T_N, T_S, T_{NE}, T_E, T_{SE}$.

		←	j-1	j	j+1	→
			
↑	i-1	...	NW	N	NE	...
	i	...	W	O	E	...
↓	i+1	...	SW	S	SE	...
			



Derivatives approximation. Following this rule we can write the simplest discrete approximation formulas of the first partial derivatives. There are many formulas. Let's see.

First derivative of $T(x, t)$, respect to x , evaluated at the central point

$$T'_x(i, j) \cong [T(i, j+1) - T(i, j-1)] / 2\Delta x = [T_E - T_W] / 2\Delta x \quad (\text{Central Difference})$$

$$T'_x(i, j) \cong [T(i, j+1) - T(i, j)] / \Delta x = [T_E - T_O] / \Delta x \quad (\text{Forward Difference})$$

$$T'_x(i, j) \cong [T(i, j) - T(i, j-1)] / \Delta x = [T_O - T_W] / \Delta x \quad (\text{Backward Difference})$$

Why so many formulas? They have different uses. The central difference is used inside the grid, when both left and right cells are available. At the left of the grid the "W" cells does not exist, so we use the forward difference. At the right, on the contrary, the "E" cell is missing. In that case we use the backward difference.

		0	1	2	3	4	5
0							
1							
2							
	forward difference		central difference		backward difference		
4							
5							
6							

First derivative of $T(x, t)$, respect to t , evaluated at the central point

$$T'_t(i, j) \cong [T(i+1, j) - T(i-1, j)] / 2\Delta t = [T_S - T_N] / 2\Delta t \quad (\text{Central Difference})$$

$$T'_t(i, j) \cong [T(i+1, j) - T(i, j)] / \Delta t = [T_S - T_O] / \Delta t \quad (\text{Forward Difference})$$

$$T'_t(i, j) \cong [T(i, j) - T(i-1, j)] / \Delta t = [T_O - T_N] / \Delta t \quad (\text{Backward Difference})$$

Second derivative of $T(x, t)$, respect to t , evaluated at the central point

$$T''_{tt}(i, j) \cong \{ [T(i+1, j) - T(i, j)] / \Delta t - [T(i, j) - T(i-1, j)] / \Delta t \} / \Delta t = \\ = \{ (T_S - T_O) / \Delta t - (T_O - T_N) / \Delta t \} / \Delta t = (T_S - 2T_O + T_N) / \Delta t^2$$

Therefore

$$T''_{tt}(i, j) \cong (T_S - 2T_O + T_N) / \Delta t^2$$

Second derivative of $T(x, t)$, respect to x , evaluated at the central point

$$T''_{xx}(i, j) \cong \{ [T(i, j+1) - T(i, j)] / \Delta x - [T(i, j) - T(i, j-1)] / \Delta x \} / \Delta x = \\ = \{ (T_E - T_O) / \Delta x - (T_O - T_W) / \Delta x \} / \Delta x = (T_E - 2T_O + T_W) / \Delta x^2$$

Therefore

$$T''_{xx}(i, j) \cong (T_E - 2T_O + T_W) / \Delta x^2$$

Second derivative of $T(x, t)$, mixed, evaluated at the central point

$$\begin{aligned} T''_{xt}(i, j) &= \partial T'_t(i, j) / \partial x \cong \partial \{ [T(i+1, j) - T(i-1, j)] / 2\Delta t \} / \partial x \\ &\cong \{ [T(i+1, j+1) - T(i-1, j+1)] / 2\Delta t - [T(i+1, j-1) - T(i-1, j-1)] / 2\Delta t \} / 2\Delta x \\ &= (T_{SE} - T_{NE} - T_{SW} + T_{NW}) / (4\Delta t \Delta x) \end{aligned}$$

Therefore

$$T''_{xt}(i, j) \cong (T_{SE} - T_{NE} - T_{SW} + T_{NW}) / (4\Delta t \Delta x)$$

With those formulas we can substitute every partial differential equation with its discrete difference equation involving only the discrete values of the grid

There are formulas for computing the partial derivatives more accurate than the previous ones but they involve values external to the 9x9 grid previously shown. These formulas are quite complicated and are beyond the scope of this document.

Boundary conditions

There are three kind of boundary conditions

- Dirichlet Condition: i.e. $U(x, 0) = \text{const}$ or $U(x, 0) = f(x)$
- Neuman Condition: i.e. $\partial U/\partial x = \text{const}$ or $\partial U/\partial x = g(x, y)$
- Robbins Condition: Mixed BC

Using the spreadsheet FD method we can provide all the above BC. Let's see how.

Dirichlet Condition. They are immediate to set because we have only to fill with the BC value in the appropriate cells

Example. Assume to take a (6 x 8) grid with step $\Delta x = 0.5$, $\Delta y = 0.5$ and we need to impose the following boundary constant conditions:

			T = 100									
			100	100	100	100	100	100	100			
			0	47.21	63.98	69.68	69.68	63.98	47.21	0		
T = 0			0	24.86	39.02	45.06	45.06	39.02	24.86	0		
			0	13.22	22.19	26.47	26.47	22.19	13.22	0		
	y		0	5.815	10.04	12.17	12.17	10.04	5.815	0		
			0	0	0	0	0	0	0	0		
						T = 0						

Dirichlet conditions
constant BC

$$\begin{aligned} T(x, 5) &= 100 \\ T(x, 0) &= 0 \\ T(0, y) &= 0 \\ T(7, y) &= 0 \end{aligned}$$

BC can be also change with x or y or both. Here an example with both variable and constant BC

			T = 100 - 3x(x - 7)									
			100	82	70	64	64	70	82	100		
			0	36.92	46.11	47.88	47.88	46.11	36.92	0		
T = 0			0	19.56	29.65	33.51	33.51	29.65	19.56	0		
			0	11.68	19.42	23.01	23.01	19.42	11.68	0		
	y		0	7.751	13.32	16.11	16.11	13.32	7.751	0		
			0	6	10	12	12	10	6	0		
						T = x(7 - x)						

Dirichlet conditions
variable and constant BC

$$\begin{aligned} T(x, 5) &= 100 - 3x(x - 7) \\ T(x, 0) &= x(7 - x) \\ T(0, y) &= 0 \\ T(7, y) &= 0 \end{aligned}$$

Neuman Condition. This occurs when we impose the condition on the 1st derivative.

It can be constant or variable. Let's see.

Assume to impose the BC $\partial T/\partial x = 0$ to the right border of the grid. It means that

$$\partial T/\partial x \cong [T(i, 0) - T(i, 1)] / \Delta x = 0 \quad \Rightarrow \quad T(i, 0) = T(i, 1)$$

Therefore the first column must be equal to the adjacent one in order to satisfy the 1st derivative boundary condition

PDE solving examples.

2D Laplace's equation

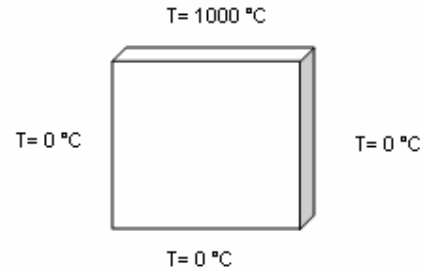
A square plate of dimension $L = 1$ m has the top border at the temperature of 1000°C . The other borders have a temperature of 0° . Find the stationary plate temperature $T(x, y)$. The differential equation is

$$T_{xx} + T_{yy} = 0$$

with $0 \leq x \leq 1$ and $0 \leq y \leq 1$

and with the following boundary conditions

$$T(x, 1) = 1000, \quad T(x, 0) = 0, \quad T(0, y) = 0, \quad T(1, y) = 0$$



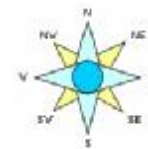
Let's model this problem building a (16×16) grid with $\Delta x = 1/16$ and $\Delta y = 1/16$

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1																			
2		1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000		
3	0																	0	
4	0																	0	
5	0																	0	
6	0																	0	
7	0																	0	
8	0																	0	
9	0																	0	
10	0																	0	
11	0																	0	
12	0																	0	
13	0																	0	
14	0																	0	
15	0																	0	
16	0																	0	
17	0																	0	
18	0																	0	
19		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
20																			

The coloured cells will be filled with the boundary conditions. They will not be altered by the iterative process. On the contrary the internal white cells will be used for approximating the solution of the PDE problem. Note that in this case we have not added any axis scales. They are necessary only when they are different each other.

For this problem we chose the 2nd center difference formula. Adopting the naming convention of the "cardinal points", any 3×3 range of cells can be represented as

		T_N	
T_W		T_O	T_E
		T_S	



$$T''_{xx}(i, j) \cong (T_E - 2T_O + T_W) / \Delta x^2$$

$$T''_{yy}(i, j) \cong (T_N - 2T_O + T_S) / \Delta y^2$$

$$T''_{xx}(i, j) + T''_{yy}(i, j) = 0 \quad \Rightarrow \quad T_O = (T_W + T_E + T_N + T_S) / 4$$

Or in an other simple way: $T_O = \text{AVERAGE}(T_W, T_E, T_N, T_S)$

Now let's begin to insert the iterative formula in the left-top empty cell (seed).

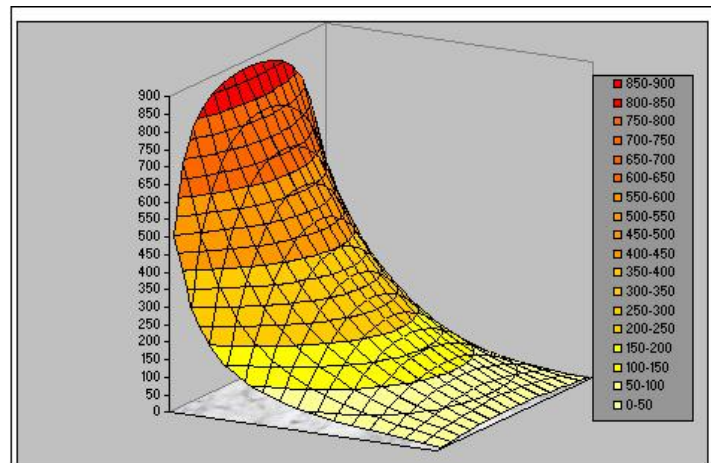
Copy the seed in the entire range of the white grid. From this instant the spreadsheet begins to recalculate all the cells. For large grid it is convenient to switch off the calculation mode and press F9 again and again until the values look stationary.

	A	B	C	D	E
1					
2		1000	1000	1000	1000
3	C	← 250			
4	C				
5	C				
6	C				

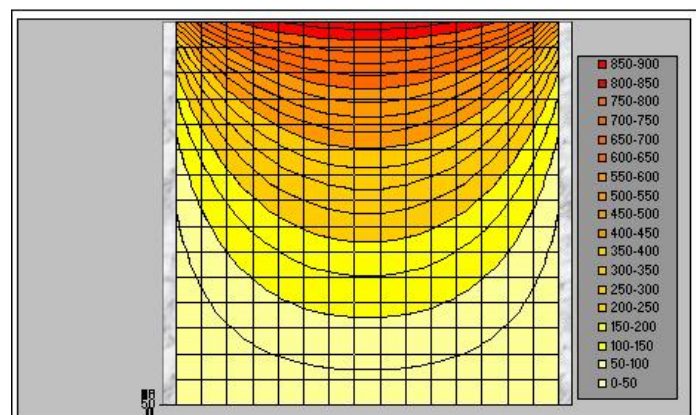
	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
0	496.2	690	778.9	825.4	852.1	867.8	876.9	881	881	876.9	867.8	852.1	825.4	778.9	690	496.2
0	294.8	484.8	600.1	670.8	715	742.5	758.6	766.1	766.1	758.6	742.5	715	670.8	600.1	484.8	294.8
0	198.3	354.3	465.8	542.7	594.5	628.5	649	658.7	658.7	649	628.5	594.5	542.7	465.8	354.3	198.3
0	144.1	268.4	366.3	439.5	492	527.9	550.2	560.9	560.9	550.2	527.9	492	439.5	366.3	268.4	144.1
0	109.6	208.7	291.6	357	406.2	441	463.1	473.8	473.8	463.1	441	406.2	357	291.6	208.7	109.6
0	85.76	165.2	234.2	290.8	334.7	366.6	387.4	397.5	397.5	387.4	366.6	334.7	290.8	234.2	165.2	85.76
0	68.18	132.3	189.2	237.2	275.3	303.6	322.2	331.4	331.4	322.2	303.6	275.3	237.2	189.2	132.3	68.18
0	54.66	106.5	153.3	193.3	225.7	250.1	266.3	274.4	274.4	266.3	250.1	225.7	193.3	153.3	106.5	54.66
0	43.94	85.86	124	157.1	184.2	204.8	218.6	225.6	225.6	218.6	204.8	184.2	157.1	124	85.86	43.94
0	35.22	68.96	99.88	126.9	149.2	166.3	177.8	183.7	183.7	177.8	166.3	149.2	126.9	99.88	68.96	35.22
0	28	54.88	79.62	101.4	119.4	133.4	142.8	147.5	147.5	142.8	133.4	119.4	101.4	79.62	54.88	28
0	21.89	42.94	62.37	79.51	93.82	104.9	112.4	116.2	116.2	112.4	104.9	93.82	79.51	62.37	42.94	21.89
0	16.61	32.61	47.41	60.49	71.45	79.93	85.71	88.64	88.64	85.71	79.93	71.45	60.49	47.41	32.61	16.61
0	11.96	23.48	34.16	43.61	51.54	57.69	61.89	64.02	64.02	61.89	57.69	51.54	43.61	34.16	23.48	11.96
0	7.741	15.2	22.12	28.26	33.41	37.41	40.14	41.53	41.53	40.14	37.41	33.41	28.26	22.12	15.2	7.741
0	3.802	7.469	10.87	13.89	16.42	18.39	19.74	20.42	20.42	19.74	18.39	16.42	13.89	10.87	7.469	3.802
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Example : Heat Diffusion of 2D Rectangular Workpiece

Each cell represents a solution point $T(x_i, y_i)$. We can perform the surface plot of the above data table getting the following characteristic 3D graph



Changing the 3D view of the above graph (Elevation = 90, Rotation = 180, Prospective = 0) we obtain the following contour plot



1D Heat Diffusion

Assume the following didactical problem to solve

$$\nabla^2 \phi = 1/\alpha^2 (\partial \phi / \partial t) \longrightarrow \cdot U_t = k U_{xx}$$

with $k = 1$, $0 \leq t \leq t_{\max}$, $0 \leq x \leq 1$ and with the following boundary conditions

$$\phi(x, 0) = \sin(\pi x) , \phi(0, t) = 0 , \phi(1, t) = 0$$

We model this problem with a grid of $\Delta x = 0.1$ and $\Delta t = 0.01$

dx	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	time
												0
												0.01
												0.02
												0.03
												0.04
												0.05
												0.06

The cells coloured will be filled with the boundary conditions. The x-axis is at the top of the grid while the t-axis is at the right, in descend order.

Let's insert the function $\phi = \sin(\pi x)$ in the second row and fill the first and the last column with 0. After that our spreadsheet looks like the following

dx	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	time
	0	0.3090	0.5878	0.8090	0.9511	1.0000	0.9511	0.8090	0.5878	0.3090	0	0
	0										0	0.01
	0										0	0.02
	0										0	0.03
	0										0	0.04
	0										0	0.05
	0										0	0.06

For this problem we adopt the Crank-Nicolson formula with $\alpha = 1$

The parameter α depends on the coefficient k , and on the grid steps Δx , Δt , $\alpha = k \cdot \Delta t / \Delta x^2$

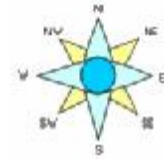
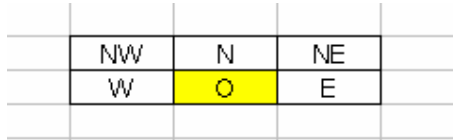
Substituting the values we have $\alpha = 0.01 / (0.1)^2 = 1$ This parameter is necessary for building the schema-matrix as shown above

Crank Nicolson Method : $\alpha = 1$		
1	0	1
1	4	1

α	$2[1 - \alpha]$	α
α	$2[1 + \alpha]$	α

How can we use the schema-matrix?. Let's see.

Adopting the naming convention of the "cardinal points", any 2 x 3 range of cells can be represented as the following



Therefore the Crank-Nicolson formula can be written as

$$U_O = [\alpha U_W + \alpha U_E + \alpha U_{NW} + \alpha U_{NE}] + 2(1-\alpha) U_N / [2(1+\alpha)] = (U_W + U_E + U_{NW} + U_{NE}) / 4$$

Or in an other simple way: $U_O = \text{AVERAGE}(U_W, U_E, U_{NW}, U_{NE})$

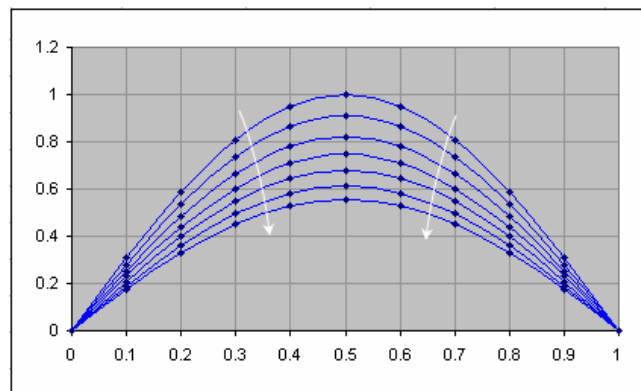
Now let's begin to insert the Crank-Nicolson formula in the left-top empty cell (seed).

Copy the seed in the entire range of the white grid. From this instant the spreadsheet begins to recalculate all the cells. For large grid it is convenient to switch off the calculation mode and press F9 again and again until the values look stationary.



dx	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	time
	0	0.3090	0.5878	0.8090	0.9511	1.0000	0.9511	0.8090	0.5878	0.3090	0	0
	0	0.2802	0.5329	0.7335	0.8623	0.9067	0.8623	0.7335	0.5329	0.2802	0	0.01
	0	0.2540	0.4832	0.6651	0.7818	0.8221	0.7818	0.6651	0.4832	0.2540	0	0.02
	0	0.2303	0.4381	0.6030	0.7089	0.7454	0.7089	0.6030	0.4381	0.2303	0	0.03
	0	0.2088	0.3972	0.5467	0.6427	0.6758	0.6427	0.5467	0.3972	0.2088	0	0.04
	0	0.1893	0.3602	0.4957	0.5827	0.6127	0.5827	0.4957	0.3602	0.1893	0	0.05
	0	0.1717	0.3265	0.4495	0.5284	0.5556	0.5284	0.4495	0.3265	0.1717	0	0.06

Each row represents a solution function $U(x_i)$ for each time step t_i . If we plot these functions we have the following pattern where each curve evolves every 0.01 sec

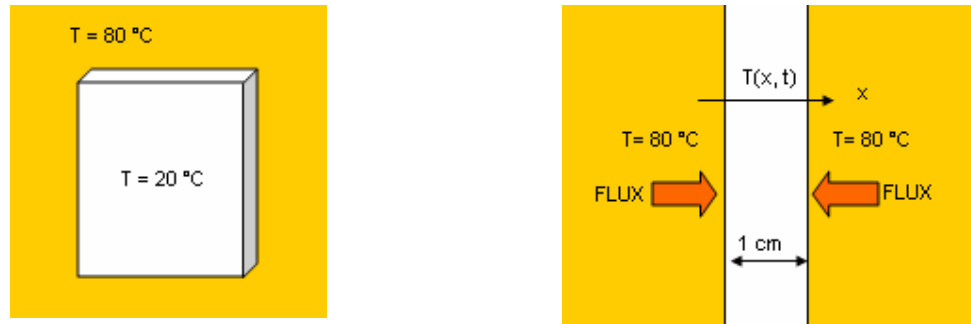


Note that these curves are obtained by parabolic interpolation.

By the FD method we have really calculated only the values at the grid steps. (dot points).

Thermal Transient

A large metallic plate, having a thickness of 1 cm is immersed into oil at 80 °C. The initial temperature of the plate is 20 °C. Find the thermal evolution for $0 < t < 8$ sec assuming a diffusivity constant $k = 0.05$ (cm²/sec). Because the thickness of the plate is tiny respect to the other dimensions we can assume that the entire thermal flux crosses the upper and lower surface. With this assumption the problem can be turned in a mono-dimensional system having the x-axis along the thickness of the plate



The modeling differential equations is

$$\frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2}$$

with $k = 0.05$, $0 \leq t \leq 8$, $0 \leq x \leq 1$ and with the following boundary conditions

$$T(x, 0) = 20 , \quad T(0, t) = 80 , \quad T(1, t) = 80$$

The boundary conditions mean that there the temperature of the surfaces remains at 80 °C for all transient. The initial internal temperature of the plate is 20 °C. Of course, for $t > 0$ the temperature of the plate evolves until it matches the oil temperature

We can model this problem with a grid of $\Delta x = 0.1$ and $\Delta t = 0.4$

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0	20	20	20	20	20	20	20	20	20	20	20
0.4	80										80
0.8	80										80
1.2	80										80
1.6	80										80
2	80										80
2.4	80										80
2.8	80										80
3.2	80										80
3.6	80										80
4	80										80
4.4	80										80
4.8	80										80
5.2	80										80
5.6	80										80
6	80										80
6.4	80										80
6.8	80										80
7.2	80										80
7.6	80										80
8	80										80

For this problem we chose the Crank-Nicolson formula with $\alpha = 2$

The parameter α depends on the coefficient k , and on the grid steps Δx , Δt , $\alpha = k \cdot \Delta t / \Delta x^2$

Substituting the values we have $\alpha = 0.05 \cdot 0.4 / (0.1)^2 = 2$

This parameter is necessary for building the schema-matrix

α $2(1-\alpha)$ α α $2(1+\alpha)$ α			Crank-Nicolson $\alpha = 2$			U_{NW}	U_N	U_{NE}
			2	-2	2	U_W	U_O	U_E

From the schema-matrix we can derive the Crank-Nicolson formula

$$U_O = [(2U_W + 2U_E + 2U_{NW} + 2U_{NE}) - 2U_N] / 6 = (U_W + U_E + U_{NW} + U_{NE} - U_N) / 3$$

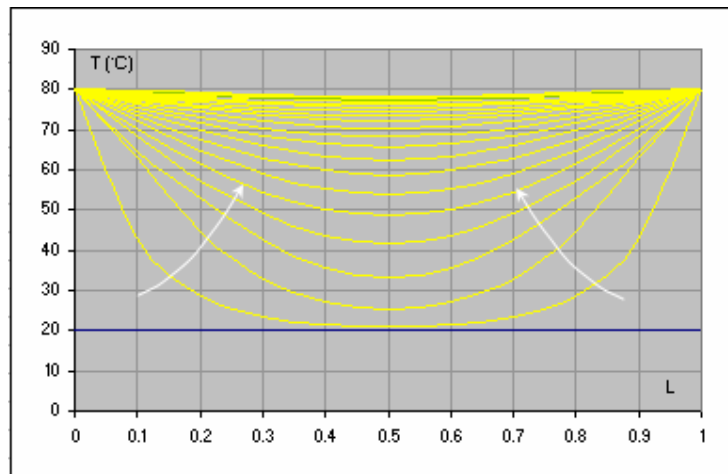
Now let's begin to insert the Crank-Nicolson formula in the left-top empty cell (seed).

Copy the seed in the entire range of the white grid. From this instant the spreadsheet begins to recalculate all the cells. For large grid it is convenient to switch off the calculation mode and press F9 again and again until the values look stationary.

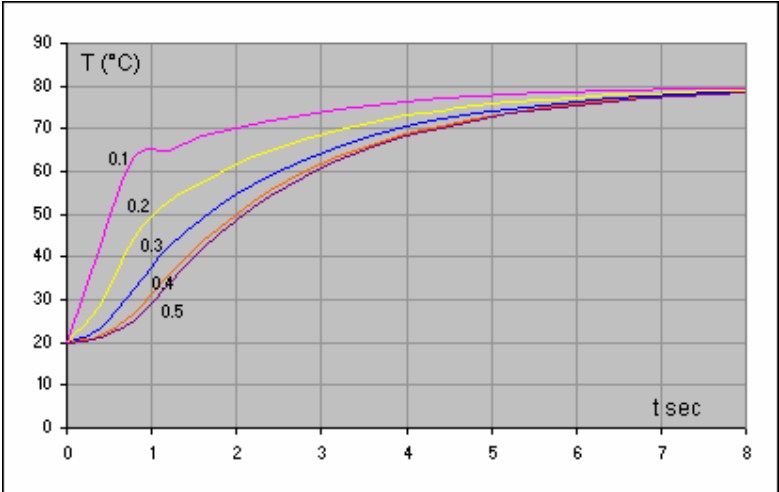
	A	B	C	D	E	F
1		0	0.1	0.2	0.3	
2	0	20	20	20	20	20
3	0.4	80	33.33			
4	0.8	80				
5	1.2	80				
6	1.6	80				
7						

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0	20	20	20	20	20	20	20	20	20	20	20
0.4	80	42.93	28.78	23.41	21.46	20.98	21.46	23.41	28.78	42.93	80
0.8	80	63.49	44.63	32.83	27.03	25.34	27.03	32.83	44.63	63.49	80
1.2	80	64.7	52.96	42.48	35.65	33.34	35.65	42.48	52.96	64.7	80
1.6	80	68.5	57.24	49	43.64	41.75	43.64	49	57.24	68.5	80
2	80	70.13	61.64	54.53	50.06	48.55	50.06	54.53	61.64	70.13	80
2.4	80	72.08	64.74	59.11	55.42	54.14	55.42	59.11	64.74	72.08	80
2.8	80	73.4	67.55	62.8	59.8	58.77	59.8	62.8	67.55	73.4	80
3.2	80	74.62	69.72	65.89	63.4	62.54	63.4	65.89	69.72	74.62	80
3.6	80	75.56	71.58	68.39	66.36	65.66	66.36	68.39	71.58	75.56	80
4	80	76.36	73.07	70.47	68.79	68.22	68.79	70.47	73.07	76.36	80

Each row represents the solution function $T(x_i)$ for each time step t_i . If we plot these functions we have the following pattern where each curve evolves every 0.4 sec



We see that after 8 sec the thermal transient is practically terminated
 Each column represents the solution function $T(t_i)$ at each internal layer.
 The following graph shows the functions $T(t)$ for $x = 0.1, 0.2, 0.3, 0.4, 0.5$ cm



2nd order ODE - Boundary conditions

The Finite Difference method can also be used to solve a 2nd ODE problem with boundary conditions of the following general form

$$y'' = f(x, y, y') \quad y(x_0) = y_0 \quad y(x_1) = y_1$$

Assume to have the following problem

$$x^2 y'' + 2x y' - 2y = x^2, \quad y(1) = 1, \quad y(4) = 1127 / 1376$$

The equation can be rewritten as

$$y'' = f(x, y, y') \quad \text{where } f(x, y, y') = (1 + 2y/x^2 - 2y'/x)$$

The formula. First of all we get the finite difference formulas that approximates the derivatives

$$y'' \cong (y_{i+1} - 2y_i + y_{i-1}) / \Delta x^2$$

$$y' \cong (y_{i+1} - y_{i-1}) / (2\Delta x)$$

Substituting in the given equation we have

$$y_i = (y_{i+1} + y_{i-1}) / 2 - h^2 f(x, y, y') / 2$$

For modelling this problem we choose a step of $\Delta x = 0.2$

Insert the boundary values into the first cell B8 and last cell B23 (yellow cell)

Insert the formulas for y , y' and $f(x, y, y')$ into the first uncoloured cells B9, C9, D9.

Note that the cells C8 and D8 remain blank.

Select the range B9:D9 and drag it down until row 22

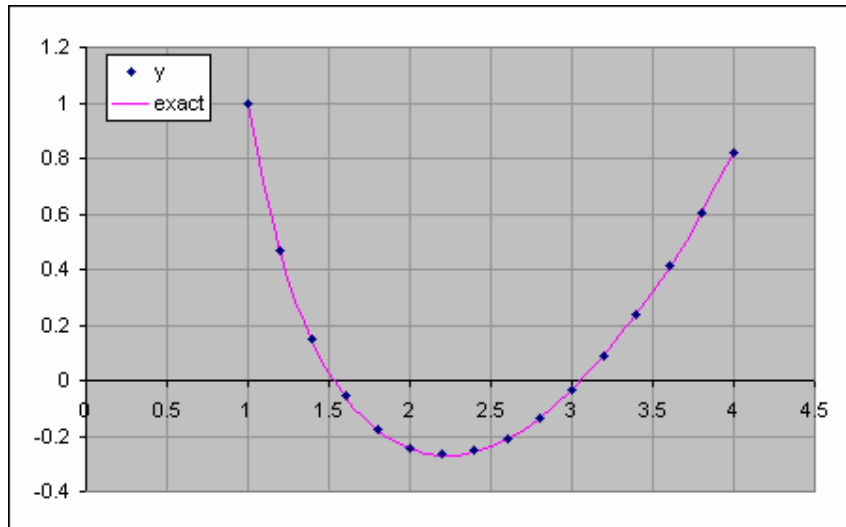
	A	B	C	D	E	F
1	Example of ODE solving with BC conditions					
2	$x^2 y'' + 2x y' - 2y = x^2, \quad y(1)=1, \quad y(4)=1127/1376$					
3						
4	$y'' = f(x, y, y') = f(x, y, y') = 1 + 2y/x^2 - 2y'/x$					
5	$y_i = 0.5(y_{i+1} + y_{i-1}) - 0.5h^2 f(x, y, y')$					
6						
7	x	y	y'	f(x, y, y')		
8	1	1.0000				
9	1.2	0.3859	-2.5000	5.7027		
10	1.4					
11	1.6					
12	1.8					
13	2					
14	2.2					
15	2.4					
16	2.6					
17	2.8					
18	3					
19	3.2					
20	3.4					
21	3.6					
22	3.8					
23	4	0.8190				

Instantaneously the computation begins.
Press F9 again and again till the values stay unchanged.

	x	y	y'	f(x, y, y')
8	1	1.0000		
9	1.2	0.4693	-2.1328	5.2065
10	1.4	0.1468	-1.3104	3.0218
11	1.6	-0.0549	-0.8115	1.9714
12	1.8	-0.1779	-0.4731	1.4159
13	2	-0.2443	-0.2219	1.0998
14	2.2	-0.2667	-0.0213	0.9092
15	2.4	-0.2528	0.1482	0.7887
16	2.6	-0.2074	0.2979	0.7095
17	2.8	-0.1337	0.4343	0.6557
18	3	-0.0338	0.5615	0.6181
19	3.2	0.0909	0.6824	0.5912
20	3.4	0.2392	0.7986	0.5716
21	3.6	0.4103	0.9115	0.5570
22	3.8	0.6038	1.0217	0.5459
23	4	0.8190		

In the following plot we compare the approximate solution (dotted line), calculated by the FD method, with the exact solution (pink continue line)

$$y(x) = 135/(86x^2) - 141x/172 + x^2/4$$



Credits

FDsolver.xla was born in the summer 2004 from the collaboration between our team and [Chatchawan Vongmahadlek](#), Thai engineer, that had developed a set of smart iterative methods to apply Excel to the solution of PDE and ODE with boundary and initial conditions. The application was so brilliant that we have dedicated many of our time to this very interesting topic. We have organized and integrated the material kindly gives us by Chatchawan and we have developed a set of VBA macros for solving FD problem, hoping all this effort will delight same other people.

References

"*Fourier Analysis*", Murray R. Spiegel, Mac Graw Hill, New York, 1976

"*Introduzione alla integrazione numerica di problemi di equazioni differenziali lineari ordinarie*", Gaetano Continillo, Istituto di Ricerche sulla Combustione CNR, Napoli

"*Elementi di integrazione numerica di alcuni problemi di equazioni alle derivate parziali*", Gaetano Continillo, Istituto di Ricerche sulla Combustione CNR, Napoli

"*Equazioni differenziali alle derivate parziali: un'introduzione*", Gino Tironi, Dipartimento di Matematica e Informatica, Univeristà degli Studi di Trieste.

"*Numerical Recipes in FORTRAN - The Art of Scientific Computing*", W.H. Press, et al.; Cambridge U. Press, 1992

"*Finite-Difference Approximations to the Heat Equation*", Gerald W. Recktenwald, Mechanical Engineering Department Portland State University, Portland, Oregon, 2004

"*Numerical methods*", by R. W. Riddaway and revised by M. Hortal, 2001

"*Numerical Methods for Ordinary Differential Equation (ODE)*", Tiziano Zito, Max Plank Institute of Colloids and Interfaces, 2002

"*Finite Differences Methods fo Convection*", Larry Caretto, Mechanical Engineering Department, California Sate University Northridge, 2002

"*Finite Difference and Spectral Method for Ordinary and Partial Differential Equations* ", Lloyd N. Trefethen, Cornel University, 1996
<http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/pdetext.html>

"*Engineering Computatio* ", Lecture 17-20, University College Dublin.
<http://www.ucd.ie/math-phy/Courses/EngComp>

"*Numerical Methods for 1-D Convection Equation Transient Flow Problem*", Parshant Dhand, Computational Fluid Dynamics, 2002

"*Numerical Methods for Hyperbolic Partial Differential Equations*", Ori Weitz, 2001

"*A finite difference Poisson solver for irregular geometries*", Z. Jomaa C. Macaskill, School Maths & Statistics- University of Sydney, 2004

"*LaPlace's and Poisson's Equations*"
<http://hyperphysics.phy-astr.gsu.edu>

"*Introduction to the Finite Element Method*", J.N. Reddy, McGraw Hill Publishers

"*A note on finite difference discretizations for Poisson equation on a disk*", Ming Chih Lai Chung Cheng University, Taiwan

"*The Poisson Equation with Local Nonregular Similarities*", Alexander Yakhot, Zohar Yosibash, Pearlstone Center for Aeronautical Engineering Studies, Department of Mechanical Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel, 2000

"Laplace's and Poisson's equations", Bo E. Sernelius, lecture of the Department of Physics and Measurement Technology, Biology and Chemistry
<http://www.ifm.liu.se/~boser/elma/>

"Course of Computational Fluid Dynamics", Hamn-Ching Chen, 2004
<http://ceprofs.tamu.edu/hchen/>

"Numerical solution of the wave equation", Richard Fitzpatrick, Lectures, University of Texas, Austin, USA
<http://farside.ph.utexas.edu/teaching/329/lectures/node102.html>

"Transitorio Termico", Giovanna Chizzoni, Lezioni di Fisica Tecnica, Università degli Studi di Pavia, 2000

"Numerical Integration of ordinary differential equation - Advanced Excel for scientific data analysis", Robert de Levie, Oxford University Press, 2004



© 2007, by Foxes Team
ITALY
Jan 2007