ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/343472236

Introduction to Xcos : A Scilab Tool for Modeling Dynamical Systems

Book · August 2020

CITATIONS
2

READS
20,348

2 authors, including:

Rachna Verma
JNV University Jodhpur INDIA
24 PUBLICATIONS **52** CITATIONS

SEE PROFILE

All content following this page was uploaded by Rachna Verma on 06 August 2020.

The user has requested enhancement of the downloaded file.

Introduction to Xcos

A Scilab Tool for Modeling Dynamical Systems

First Edition

Arvind Kumar Verma
Rachna Verma

MBM Engineering College
JNV University, Jodhpur, Rajasthan, India

Introduction to Xcos

A Scilab Tool for Modeling Dynamical Systems

Copyright © 2020 Dr. Arvind Kumar Verma and Dr. Rachna Verma

ISBN: 9798670206693

First published on kdp.amazon.com by Dr. Arvind Kumar Verma and Dr. Rachna Verma

All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the authors.

Cover (illustration) by Dr. Rachna Verma

Set by Dr. Rachna Verma

Published on kdp.amazon.com

This book provides a comprehensive, hands-on introduction to Xcos, a powerful, open-source Scilab tool for modeling dynamical systems. Starting with basic concepts, the book lucidly builds advanced Xcos concepts one needs for modeling complex real-life systems. The book contains a large number of illustrative examples. The book is well suited as a textbook for learning Xcos for science and engineering students. It is sold under the express understanding that the information contained in this book is accurate to the best of authors' knowledge. However, the authors will not be held responsible for the consequences of any actions based on the content of the book for any purpose.

Dedicated to Scilab Developers

Contents

Preface xv

Chapter1: Introduction to Xcos 1

1.1 Dynamical Systems 1

1.2 Solutions of dynamical systems 3

1.3 Solving Differential Equations in Xcos 5

1.3.1 Solving a first order ODE in Xcos 5

1.3.2 Solving second order or higher order ODE in Xcos 7

1.3.3 State space approach to solve differential equations 9

1.3.4 Transfer function approach to solve ODEs 11

1.3.5 Solution of non-linear differential equations 13

1.4 Xcos Modeling Environment 14

1.4.1 Running Xcos 14

1.4.2 Palette browser 16

1.4.3 Model editor 19

1.4.3.1 File Menu 19

1.4.3.2 Edit Menu 20

1.4.3.3 View Menu 20

1.4.3.4 Simulation Menu 20

1.4.3.5 Format Menu 33

1.4.3.6 Tools Menu 44

1.4.3.7 ? Menu (Help Menu) 45

1.5 Model Creation in Xcos 45

1.6 Step by step construction of an Xcos model 45
1.7 Inside an Xcos block 49
Chapter 2: Sources Palette 57
2.1 CLOCK_c block 57
2.2 CONST_m block 62
2.3 Counter block 65
2.4 CURV_f block 67
2.5 FROMWSB block 70
2.6 GENSIN_f block 76
2.7 GENSQR_f block 77
2.8 Modulo_Count block 79
2.9 Ramp block 81
2.10 Rand_m block 83
2.11 READC_f block 86
2.12 RFILE_f block 87
2.13 SampleCLK block 94
2.14 SAWTOOTH_f block 96
2.15 Sigbuilder block 98
2.16 STEP_FUNCTION block 104
2.17 PULSE_SC block 106
2.18 TIME_f block 107
2.19 TKSCALE block 108
Chapter 3: Sinks Palette 111
3.1 AFFICH_m block 112
3.2 BARXY block 118
3.3 CANIMXY block 120

3.4 CANIMXY3D block 126
3.5 CFSCOPE block 128
3.6 CMATVIEW block 131
3.7 CMAT3D block 132
3.8 CSCOPE block 134
3.9 CMSCOPE block 134
3.10 CSCOPXY block 137
3.11 CSCOPXY3D block 139
3.12 ENDBLK block 139
3.13 END_c block 141
3.14 TOWS_c block 142
3.15 TRASH_f block 144
3.16 WRITEC_f block 145
Chapter 4: Mathematical Operations Palette 147
4.1 ABS_VALUE block 149
4.2 BIGSOM_f block 152
4.3 COSBLK_f block 152
4.4 EXPBLK_m block 152
4.5 GAINBLK_f block 154
4.6 INVBLK block 159
4.7 LOGBLK_f block 161
4.8 MATMAGPHI block 163
4.9 MATZREIM block 163
4.10 MAXMIN block 164
4.11 MAX_f block 167
4.12 MIN_f block 168

4.13 POWBLK_f block 168
4.14 PRODUCT block 169
4.15 Relational block 171
4.16 SIGNUM block 171
4.17 SINBLK_f block 172
4.18 SQRT block 172
4.19 SUMMATION block 174
4.20 TANBLK_f block 175
4.21 TrigFun block 176
Chapter 5: Matrix Operation Palette 179
5.1 CUMSUM block 179
5.2 EXTRACT block 180
5.3 EXTTRI block 183
5.4 MATBKSL block 184
5.5 MATCATH block 184
5.6 MATCATV block 185
5.7 MATDET block 186
5.8 MATDIAG block 187
5.9 MATDIV block 187
5.10 MATEIG block 188
5.11 MATEXPM block 188
5.12 MATINV block 189
5.13 MATLU block 190
5.14 MATMUL block 191
5.15 MATPINV block 191
5.16 MATRESH block 192

5.17 MATSING block 194
5.18 MATSUM block 196
5.19 MATTRAN block 196
5.20 MATZCONJ block 198
5.21 RICC block 199
5.22 ROOTCOEF block 200
5.23 SUBMAT block 201
Chapter 6: Signal Routing Palette 203
6.1 DEMUX block 205
6.2 EXTRACTOR block 205
6.3 GOTO / FROM and GotoTagVisibility blocks 206
6.4 ISELECT_m block 207
6.5 MUX block 211
6.6 M_SWITCH block 213
6.7 NRMSOM_f block 214
6.8 RELAY_f and SELECT_m blocks 215
6.9 SELF_SWITCH block 217
6.10 SWITCH2_m block 218
6.11 SWITCH_f block 218
Chapter 7: Event Handling Palette 221
7.1 ANDBLK block 223
7.2 ANDLOG_f block 224
7.3 CEVENTSCOPE block 225
7.4 CLKFROM, CLKGOTO & CLKGotoTagVisibility blocks 226
7.5 CLKSOMV_f block 227
7.6 EDGE_TRIGGER block 228

7.7 ESELECT_f block 229
7.8 EVTDLY_c block 231
7.9 EVTGEN_f block 231
7.10 EVTVARDLY block 233
7.11 Extract_Activation block 234
7.12 HALT_f block 234
7.13 IFTHEL_f block 235
7.14 MCLOCK_f block 237
7.15 MFCLCK_f block 238
7.16 freq_div block 239
Chapter 8: Integer Palette 241
8.1 BITCLEAR block 242
8.2 BITSET block 243
8.3 CONVERT block 244
8.4 DFLIPFLOP and DLATCH blocks 245
8.5 EXTRACTBITS block 248
8.6 INTMUL block 251
8.7 JKFLIPFLOP block 251
8.8 LOGIC block 253
8.9 SHIFT block 253
8.10 SRFLIPFLOP block 256
8.11 LOGICAL_OP block 256
Chapter 9: Continuous Time Systems Palette 263
9.1 CLINDUMMY_f block 264
9.2 CLR block 266
9.3 CLSS block 268

9.4 DERIV block 271
9.5 INTEGRAL_f block 272
9.6 INTEGRAL_m block 273
9.7 PID block 274
9.8 TCLSS block 277
9.9 TIME_DELAY block 277
9.10 VARIABLE_DELAY block 278
Chapter 10: Discrete Time Systems Palette 281
10.1 DELAY_f block 282
10.2 DELAYV_f block 284
10.3 DLR block 285
10.4 DLRADAPT_f block 286
10.5 DLSS block 288
10.6 DOLLAR_f block 291
10.7 REGISTER block 292
10.8 SAMPHOLD_m block 293
10.9 QUANT_f block 295
Chapter 11: Discontinuities Palette 299
11.1 BACKLASH block 300
11.2 DEADBAND block 301
11.3 HYSTHERESIS block 301
11.4 RATELIMITER block 303
11.5 SATURATION block 304
Chapter 12: Port and Subsystem Palette 307
12.1 CLKINV_f block 308
12.2 CLKOUTV_f block 308

12.3 IN_f block 309
12.4 OUT_f block 312
12.5 SUPER_f block 312
12.6 INIMPL_f and OUTIMPL_f blocks 321
Chapter 13: User-Defined Functions Palette and Construction of a New Block 323
13.1 Construction of a new block 223
13.2 Expression block 325
13.3 Scifunc block 326
Chapter 14: Solutions of a few Differential Equations using Xcos 337
14.1 Exponential growth and decay of a population 338
14.2 Newton's law of cooling 339
14.3 Free fall with drag 340
14.4 Logistic model of population growth and decay model 341
14.5 Pursuit Curve problem 341
14.6 Damped Simple Harmonic Motion 344
14.7 Forced Simple Harmonic Motion 347
14.8 Projectile motion with drag 347
14.9 Bouncing ball simulation 351
14.10 Non-linear pendulum motion simulation 352
14.11 Pollution spreading simulation 354
14.12 Spring coupled masses simulation 356
14.13 RLC circuit simulation 358
Chapter 15: Modelica based blocks in Xcos 361
15.1 Electrical Palette 365

15.1.1 Capacitor block 365
15.1.2 Constant Voltage block 367
15.1.3 Current Sensor 367
15.1.4 CCS bock 368
15.1.5 CVS block 369
15.1.6 Diode block 370
15.1.7 Ground block 370
15.1.8 Ideal Transformer block 372
15.1.9 Inductor block 372
15.1.10 NMOS block 374
15.1.11 NPN block 375
15.1.12 OpAmp block 376
15.1.13 PMOS block 377
15.1.14 PNP block 379
15.1.15 Potential Sensor block 380
15.1.16 Resistor block 380
15.1.17 Sine Voltage block 381
15.1.18 Switch block 382
15.1.19 VVsourceAC block 382
15.1.20 Variable Resistor block 383
15.1.21 Voltage Sensor block 384
15.1.22 Vsource AC Block 385
15.2 Thermal-hydraulic blocks 387

Preface

Xcos is a very powerful and open source block-based modeling and simulation system for dynamical systems. Its capabilities are comparable to commercially available block-based modeling and simulation tools, including Simulink®, one of the most popular commercial tool. Xcos is useful for modeling continuous and discrete dynamical systems. Further, it provides facilities to seamlessly integrate continuous and discrete components in a single model, making it capable to handle hybrid dynamical systems. Xcos provides a modular approach to model complex dynamical systems using a block diagram editor. Xcos contains a rich library of commonly used blocks, arranged in various palettes for the convenience of searching them, for elementary operations needed to construct models of many dynamical systems. These blocks can be dragged and dropped into the model editor to create a simulation model. For advanced users, Xcos provides facilities to create new blocks and to create their own libraries to further extend the capabilities of Xcos. Since Xcos is available free of cost to everyone across the globe and is continuously upgraded by a strong team of open source developers, it is suitable for all undergraduate students, researchers, professors and professionals in any field of Science and Engineering. Further, many commercial developers are also using it to reduce their project cost and has reported many successful applications.

This book is written following several years of teaching the software to our students in introductory courses in numerical methods and simulations. The basic objective to write this book is to teach Xcos in a friendly, non-intimidating fashion, without any previous modeling experience. Therefore, the book is written in simple language with many sample problems in mathematics, science, and engineering. Starting from the basic concepts, the book gradually builds advanced concepts, making it suitable for freshmen and professionals. The Xcos models of all the examples included in this book are available at **https://github.com/arvindrachna/Introduction_to_Xcos.**

There are a few books available for teaching basic concepts of Xcos, which is the basic motivation to write this book. We hope the book will be helpful to spread the awareness and use of Xcos for modeling dynamical systems.

The book consists of fifteen chapters. The first chapter gives a brief introduction to dynamical systems and Xcos modeling environment. The second chapter describes the Sources Palette, which consists of blocks to generate varieties of signals and data for models. The third chapter describes the Sink Palette, which consists of different signal and data visualization blocks. The fourth chapter describes the Mathematical Operations Palette, which consists of blocks implementing different mathematical operations and functions. The fifth chapter describes the Matrix Operation Palette, which is a collection of blocks for various matrix operations. The sixth chapter describes the Signal Routing Palette, which consists of blocks to route signals among blocks. The seventh chapter describes the Event Handling Palette, which consists of blocks for handing events during simulation. The eighth chapter describes the Integer Palette which consists of blocks to perform various Boolean operations and bit manipulation operations. The ninth chapter describes the Continuous Time Systems Palette, which consists of blocks for modeling continuous dynamical systems and performing various calculus operations. The tenth chapter describes the Discrete Time Systems Palette, which consists of blocks to model discrete dynamical systems and perform discrete calculus operations. The eleventh chapter describes the Discontinuities Palette, which consists of blocks to handle discontinuities in dynamical systems. The twelfth chapter describes the Port and Subsystem Palette, which consists of blocks to create subsystems to model complex systems in a modular fashion. The thirteenth chapter describes the User-Defined Functions Palette, which consists of generic blocks that can be used to extend the capabilities of Xcos and create customized new blocks. The fourteenth chapter presents Xcos models to solve common engineering and scientific problems. The last chapter briefly introduces the Modelica Based blocks in Xcos for modeling a system using components. For easy comprehension, the book includes more than 300 small examples.

We acknowledge with thanks the constructive suggestions, for the improvement of this book, given by many of our colleagues at MBM Engineering College, Jodhpur, Rajasthan, India. We also thank our parents for inculcating passion for learning and encouraging work harder.

We hope that the book will be useful in lucidly building expertise in Xcos to the readers. We sincerely welcome any suggestion to further improve the book.

Arvind Kumar Verma
Rachna Verma

About the Authors

Dr. Arvind Kumar Verma is working as a Professor, Department of Production and Industrial Engineering, MBM Engineering College, JNV University, Jodhpur, Rajasthan, India. He received his BE (Industrial Engineering) and ME (Production and Industrial Systems Engineering) from IIT Roorkee, India and obtained his PhD in CAD/CAM from MBM Engineering College, Jodhpur, Rajasthan, India. He has 25 years of teaching experience. He teaches numerical methods, CAD/CAM, robotics, computer programming. He has research interest in robotic vision, machining feature recognition and numerical computation.

Dr. Rachna Verma is working as an Assistant Professor, Department of Computer Science and Engineering, MBM Engineering College, JNV University, Jodhpur, Rajasthan, India. She received her BSc (Math Honours) form Delhi University, India and MCA and PhD from JNV University, Jodhpur, India. She has seventeen years of teaching experience. She teaches numerical methods, computer programming and computer graphics and has research interest in computer vision and image processing.

Chapter 1
Introduction to Xcos

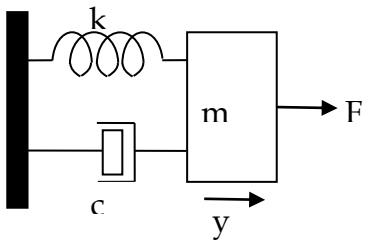
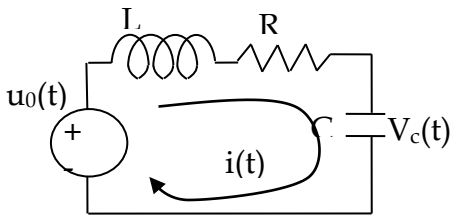
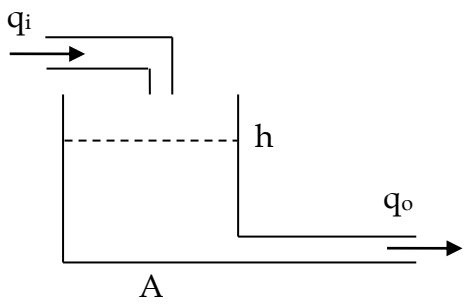
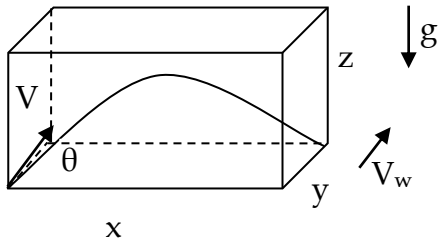
Xcos is a toolbox in Scilab for modeling and simulation of dynamical systems. Similar to Scilab, it is also open source. Xcos is a block-based and very powerful modeling and simulation system, with capabilities comparable to Simulink, the most popular commercial block-based modeling and simulation tool. Xcos is useful for modeling continuous and discrete dynamical systems. Further, it provides facilities to seamlessly integrate continuous and discrete components in a single model, making it capable to handle hybrid dynamical systems. Xcos provides a modular way to model complex dynamical systems using a block diagram editor.

Xcos contains a rich library of commonly used blocks, arranged in various palettes for the convenience of searching them, for elementary operations needed to construct models of many dynamical systems. These blocks can be dragged and dropped into the model editor to create a simulation model. For advanced users, Xcos provides facilities to create new blocks and to create their own libraries to further extend the capabilities of Xcos.

1.1 Dynamical Systems

Dynamics is primarily the study of the time-evolutionary process, such as decay of a radioactive material, oscillatory electrical circuits, oscillation of spring damper systems. The dynamic behaviour of an engineering and scientific system is generally modeled in the form of a system of equations, mostly differential equations. This model is called dynamical system. There are many good books available that deal with formulations and solutions of dynamical systems. However, for the purpose of easy comprehension of the topics discussed in the later part of the book, Table 1.1 lists few dynamic systems, taken from the different fields of engineering, along with the corresponding dynamical systems. For the derivation of the dynamical system equations, readers can refer any standard book of dynamics.

Table 1.1: A few examples of dynamical systems

S. No.	Dynamic system	Corresponding dynamical system
1		$F = m\frac{d^2y}{dt^2} + c\frac{dy}{dt} + ky$
2		$LC\frac{d^2V_c}{dt^2} + RC\frac{dV_c}{dt} + V_c = u_0(t)$
3		$q_i = A\frac{dh}{dt} + q_o$
4		$x = (V\cos\theta)\,t$ $y = V_w t$ $z = (V\sin\theta)\,t - \frac{1}{2}gt^2$

Dynamical systems can be classified as discrete, continuous and hybrid systems. In a discrete system, the state variables (parameters that define the current status of a system, such as the number of customers in a queue) of the system change instantaneously at discrete points in time. For example, the number of customers in a queue at a bank changes only when a new customer arrives or when a customer departs after being served.

In a continuous system, state variables change continuously with respect to time, for example, the trajectory of a projectile in which the state variables such as the position and velocity of the projectile change continuously with respect to time.

In a hybrid system, some state variables change continuously and some at discrete time. For example, during a vehicle driving, the position of the vehicle changes continuously whereas the brake is activated at discrete time intervals. A hybrid system has the benefit of encompassing a larger class of systems within its structure, allowing more flexibility in modeling continuous and discrete dynamic systems.

1.2 Solutions of dynamical systems

As can be seen from Table 1.1, most of the dynamical systems in science and engineering are reduced to differential equations. Hence, their solutions are reduced to the problem of solving differential equations satisfying certain conditions. There are various methods to solve differential equations: analytical methods and numerical methods. An analytical method gives an exact solution whereas a numerical method gives approximate one. Unfortunately, the analytical methods of solution to differential equations are available to only a certain class of equations. However, most of the equations governing real physical systems do not lend themselves to close form analytical solutions. Hence, numerical methods are widely used to solve such equations.

There are a number of numerical methods, such as Euler, Runge-Kutta, Adams-Bashforth, etc., that can be used to solve the following general form of first order differential equations:

$$\frac{dy}{dt} = f(t, y) \qquad 1.1$$

with the initial condition $y(t_0) = y_0$

Further, these methods can be extended to solve a system of first order differential equations of the form:

$$\left.\begin{aligned}\frac{dx}{dt} &= f(t,x,y)\\ \frac{dy}{dt} &= f(t,x,y)\end{aligned}\right\} \qquad 1.2$$

With the initial conditions, $x(t_0) = x_0$ and $y(t_0) = y_0$

For numerically solving higher order differential equations, the higher order equations are first reduced to a system of first order differential equations by introducing extra variables. To illustrate the above process, consider the general form of second order differential equation

$$y'' = f(t, y, y') \qquad 1.3$$

With the initial conditions $y(t_0) = y_0$ and $y'(t_0) = y'_0$

By setting $z = y'$, equation 1.3 can be written as

$$\left.\begin{aligned}y' &= z\\ z' &= f(t,y,z)\end{aligned}\right\} \qquad 1.4$$

With the initial conditions $y(t_0) = y_0$ and $z(t_0) = y'_0$

Similarly, any higher order differential equation can be reduced to a system of first order differential equations; hence it can be solved using numerical methods.

Computer programs for numerical methods can be written in any programming languages. However, Matlab and Scilab are widely used for numerical methods due to their rich library of engineering and scientific functions and very advanced visualization capabilities. Scilab has another advantage of being open source and free for academic and commercial uses.

Another reason for the popularity of Matlab and Scilab for solving dynamical systems is the availability of toolboxes (Simulink in Matlab and Xcos in Scilab) that use the block-diagram approach to create and solve dynamical models. The block-diagram approach is more convenient for modeling complex real-life physical systems. The focus of the book is to introduce Xcos for simulating dynamical systems.

1.3 Solving Differential Equations in Xcos

Xcos has a very powerful and easy-to-use graphical editor that can be used to construct block diagram models of dynamical systems. The editor contains a rich library of blocks for performing common operations, such as signal generation, integration, differentiation, etc. For the convenience of use, these blocks are arranged in palettes. In this section, various approaches to solve differential equations in Xcos are described.

1.3.1 Solving a first order ODE in Xcos

The basic procedure to solve an ODE (Ordinary Differential Equations) in Xcos is illustrated by solving a general first order differential equation shown in equation 1.5:

$$y' = F(t, y) \qquad 1.5$$

with the initial condition $y(0) = c$

Equation 1.5 is solved in Xcos by feeding y' to an integrator block, which outputs y. Mathematically, the concept is equivalent to equation 1.6. A block diagram of this approach of solving a first order differential equation is shown in figure 1.1.

$$y = \int y' \, dt \qquad 1.6$$

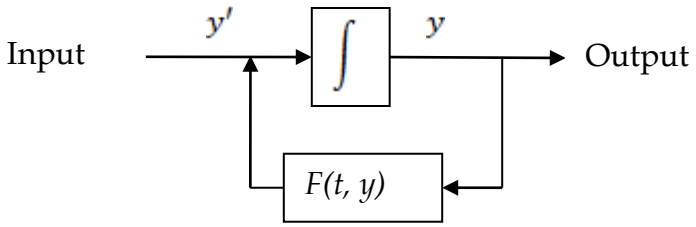


Figure 1.1: Schematic for solving a first order ODE

Let us illustrate the above concepts with the help of well-known Newton's law of cooling, which states that the rate of change of the temperature of a body is proportional to the temperature difference between the body and its surroundings. Mathematically, the law is given in the form of a first order differential equation as given in equation 1.7.

$$\frac{dT}{dt} = -k(T - T_a) \qquad 1.7$$

Where k >0 and is a cooling constant. T_a is the surrounding temperature and T is the current temperature of the body.

An Xcos model of equation 1.7, for a case with $T_0 = 60\ °C$, $T_a = 20\ °C$ and $k = 0.1\ s^{-1}$, is given in figure 1.2. Here, T_o is the initial temperature (initial condition) of the body. The target of the solution is the calculation of temperature of the body as time changes. The output of the model simulation is given in figure 1.3.

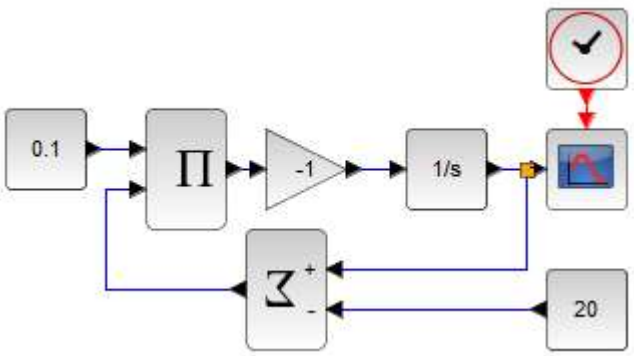


Figure 1.2: Xcos model to solve equation 1.7

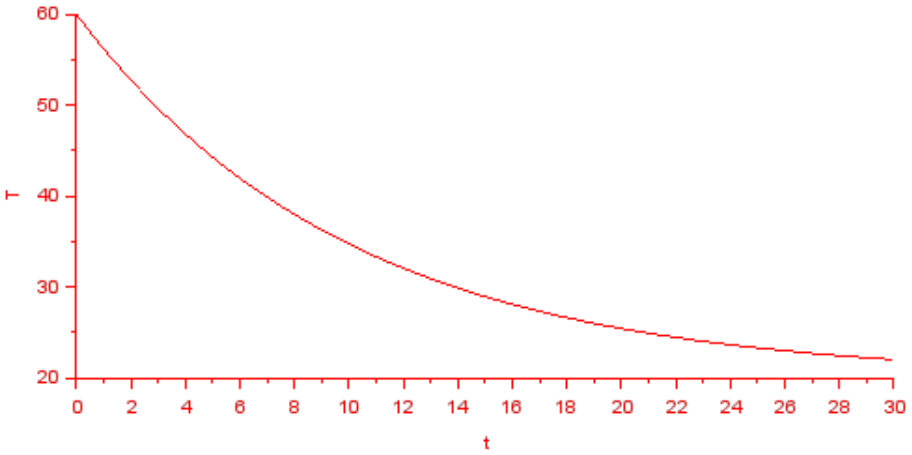


Figure 1.3: Output of the Xcos model of equation 1.7

1.3.2 Solving second order or higher order ODE in Xcos

To solve a second or higher order ODE in Xcos, a series of integrators are used. Consider a general second order general ODE as given in equation 1.8.

$$y''(t) = F(t, y(t), y'(t)) \qquad 1.8$$

Equation 1.8 is solved using two integrators. The first integrator takes y'' as input and gives y' as output. The second integrator takes y' as input and output y as output. This procedure is shown in figure 1.4. This is basically achieved by rewriting the given second order ODE into two linked first order ODEs, using a state variable. For example, equation 1.8 can be rewritten as a system of two linear first order ODEs as given in equation 1.9. To further clarify the rewriting process, consider the second order ODE given in equation 1.10, which model the simple harmonic motion of a spring-mass system sliding on a frictionless horizontal surface. Equation 1.11 shows the system of first order ODEs corresponding to equation 1.10. Xcos model of equation 1.11 (with m=2 and k=5) is shown in figure 1.5. Figure 1.6 shows the simulated solution for few oscillations.

Similarly, without loss of generality, this scheme of solving second order ODEs can be extended to any higher order ODE by rewriting the equation in terms of a corresponding system of first order ODEs.

$$\left.\begin{aligned} y' &= z \\ z' &= F(t, y, z) \end{aligned}\right\} \qquad 1.9$$

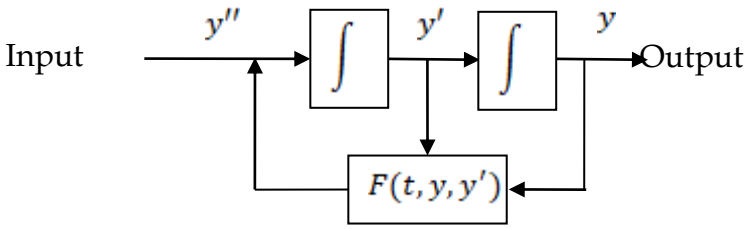


Figure 1.4: Schematic for solving a second order ODE

$$x''(t) = -\frac{1}{m}(kx) \qquad 1.10$$

Where, k is the spring constant, m is the mass of the sliding block, x is the displacement of the spring from its un-stretched equilibrium.

$$\left.\begin{aligned} x' &= z \\ z' &= -\frac{1}{m}(kx) \end{aligned}\right\} \qquad 1.11$$

Where, k is the spring constant, m is the mass of the sliding block, x is the displacement of the spring from its un-stretched equilibrium and z is a state variable.

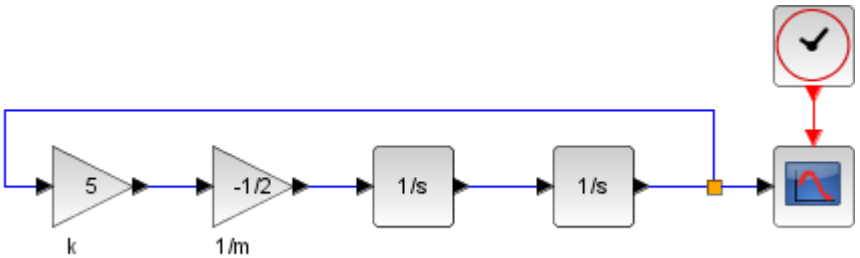


Figure 1.5: Xcos model of equation 1.10

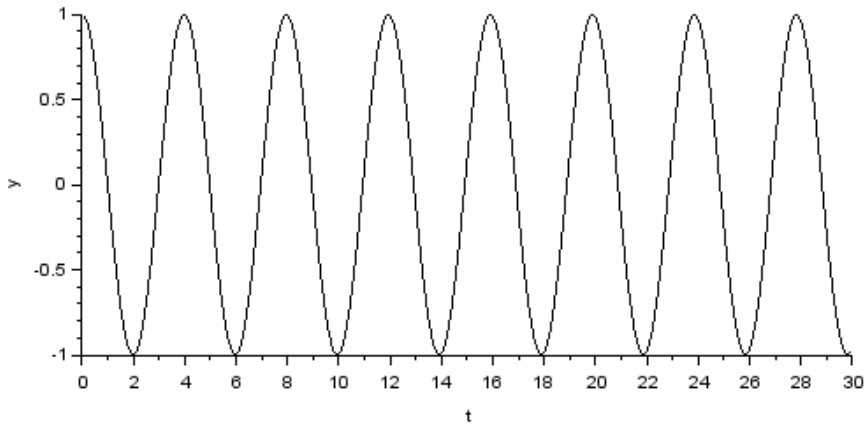


Figure 1.6: Output of Xcos model shown in figure 1.5

1.3.3 State space approach to solve differential equations

Xcos also has a block that uses the state space approach to solve a differential equation. The basic concept of the state space approach is explained with the help of a differential equation (equation 1.12) representing forced, damped oscillations.

$$mx'' + bx' + kx = u(t) \qquad 1.12$$

Defining, $x_1 = x'$ and $x_2 = x$, equation 1.12 can be rewritten as equation 1.13

$$\left.\begin{aligned} x_1{}' &= -\frac{b}{m}x_1 - \frac{k}{m}x_2 + \frac{1}{m}u(t) \\ x_2' &= x1 \end{aligned}\right\} \qquad 1.13$$

Here, two state variables have been introduced so as equation 1.12 can be written in the matrix form as given in equation 1.14.

$$x' = Ax + Bu \qquad 1.14$$

Where

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$A = \begin{bmatrix} -\frac{b}{m} & -\frac{k}{m} \\ 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{m} \\ 0 \end{bmatrix}$$

The state space block of Xcos is designed to efficiently integrate a differential equation written in the form of equation 1.14 in a single block, which makes model compact and neat. The state space block computes values of all the state variables in the form of a vector, *x*. For the purpose of display of the solution, the block returns combined output in the form of a signal generated by equation 1.15.

$$y = Cx + D\ u(t) \qquad 1.15$$

Where, *C* is a row vector (the typical value of *C* is [0,1]) and *D* =0 or 1

By properly setting the values of *C* and *D*, the desired component of the solution vector can be obtained. For example, if *C* is set to [0, 1] and *D* is set to 0, the final integral values of the original differential equation is returned. To return the first derivative, *C*= [1, 0] and *D*=0 are used.

Figure 1.7 shows an Xcos model of equation 1.12 using a state space block. The values of *m*= 4 kg, *b*=0.4 kg/s and *k*=2.0 N/m is used in the model. The excitation function, *u(t)* is a sinusoidal wave with amplitude=1, frequency 1 radian/s and phase angle=0. The output of the model is shown in figure 1.8. The parameters of the state space block (*A, B, C, D* and the initial conditions of various state variables in the form of a vector i.e. [x_1; x_2]) of the model are given as below:

$$A = \begin{bmatrix} -0.1 & -0.5 \\ 1 & 0 \end{bmatrix},\ B = \begin{bmatrix} 0.25 \\ 0 \end{bmatrix},\ C = [0,1],\ D = 0\ ,\ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

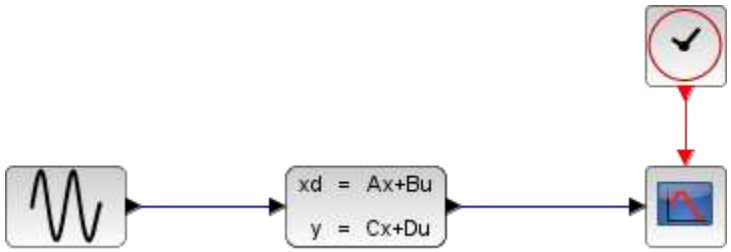


Figure 1.7 : Xcos model of equation 1.12, using state space block

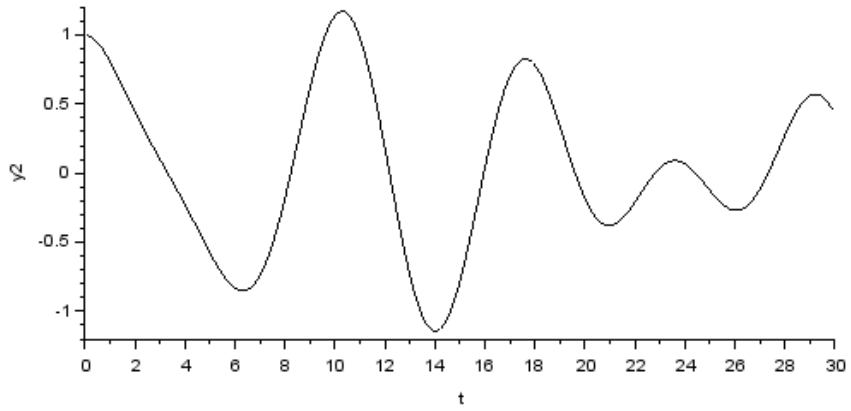


Figure 1.8: Solution of equation 1.12

1.3.4 Transfer function approach to solve ODEs

Xcos provides another elegant and compact approach to solve an ODE using the transfer function block. In this approach, one needs to enter the transfer function's numerator and denominator as the parameters of the transfer function block. This method can only be used for ODEs, which can be written in the transfer function form, with zero initial condition.

Let us again consider equation 1.12 to illustrate the use of the transfer function approach. Equation 1.12 is rewritten to include the independent variable, time and is given in equation 1.16

$$mx''(t) + bx'(t) + kx(t) = u(t) \qquad 1.16$$

Taking the Laplace transform of Equation 1.16, the transfer function of the system may be written as given in equation 1.17.

$$ms^2X(s) + bsX(s) + kX(s) = U(s) \qquad 1.17$$

Equation 1.17 may be rearranged in the form of equation 1.18

$$\frac{X(s)}{U(s)} = \frac{1}{ms^2 + bs + k} \qquad 1.18$$

Once the given ODE is rewritten in an equivalent equation in the transfer function form, an Xcos model of the same can be created by using the transfer function block. Figure 1.9 shows an Xcos model using a transfer function block to solve equation 1.16. The values of m= 4 kg, b=0.4 kg/s and k=2.0 N/m are used in the model. The excitation function, $u(t)$ is a sinusoidal wave with amplitude=1, frequency 1 radian/s and phase angle=0. For the given numerical values, the transfer function form of equation 1.18 is given in equation 1.19. The output of the model shown in figure 1.9 is given in figure 1.10.

$$\frac{X(s)}{U(s)} = \frac{1}{4s^2 + 0.4s + 2} \qquad 1.19$$

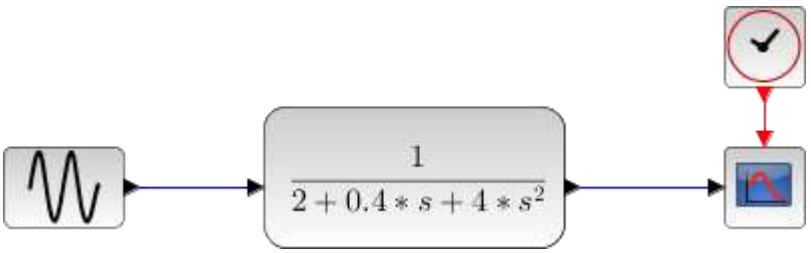


Figure 1.9: Xcos model of equation 1.17

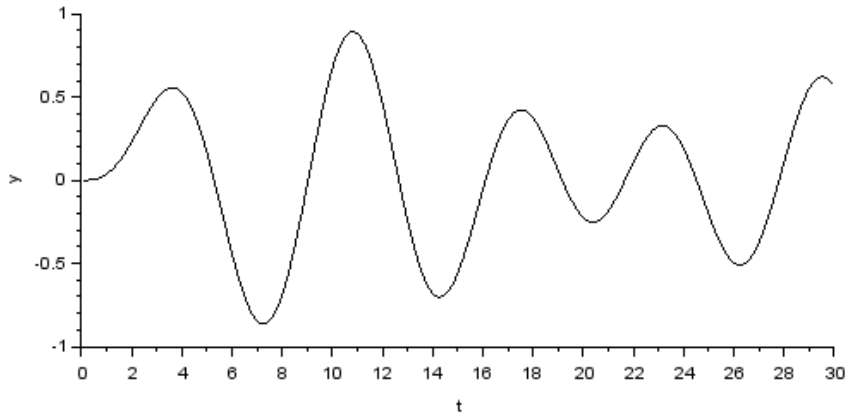


Figure 1.10: Output of the model shown in figure 1.9

1.3.5 Solution of non-linear differential equations

Non-linear differential equations can also be solved in Xcos. Consider a non-linear differential equation given in equation 1.20, which is a dynamical model of a non-linear pendulum.

$$x'' + \frac{g}{L}\sin(x) = 0 \qquad 1.20$$

Where x is the angular displacement of the pendulum, L is the length of the pendulum string and g is the acceleration due to gravity.

The Xcos model of equation 1.20 is created using two integrators, as shown in figure 1.11. The values of the constants used in the model are g=10 m/s^{-2} and L=2.5 m. The initial excitation angle is set to $x_0 = 2$ radian and the initial velocity $x_0' = 0$ radian/s. The output of the model shown in figure 1.11 is given in figure 1.12.

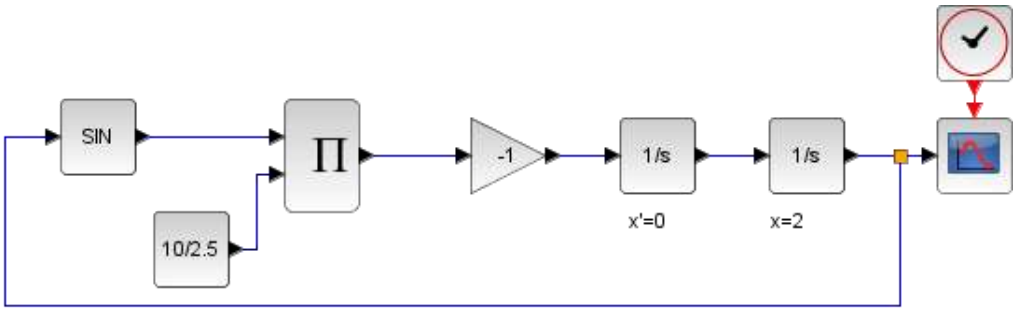


Figure 1.11: Xcos model of a non-linear pendulum

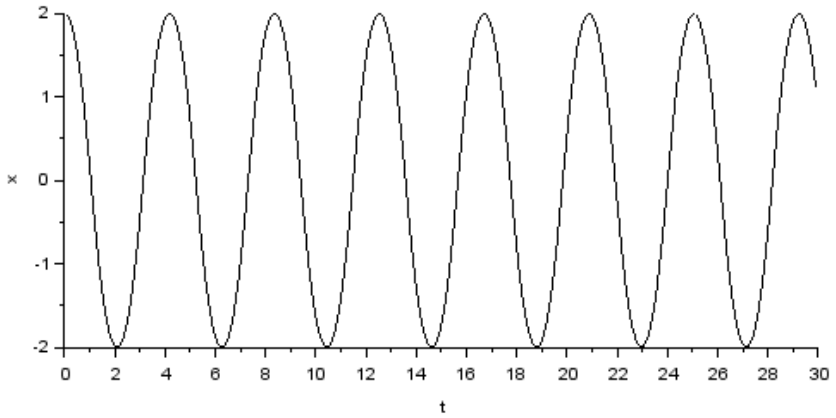


Figure 1.12: Solution of the Xcos model shown in figure 1.11

1.4 Xcos Modeling Environment

In this section, we describe the Xcos modeling environment. Xcos provides an integrated modeling environment where we can create, edit and simulate dynamic models.

1.4.1. Running Xcos

Xcos is an integral application of Scilab. It can be run by issuing the **Xcos** command on the Scilab command prompt, clicking the Xcos icon, selecting the Xcos application from the application drop down menu, or by double clicking an Xcos model. Figure 1.13 shows the default screen of Scilab along with the various options to start Xcos. Figure 1.14 shows the default screen of Xcos. By default, the Xcos command opens a blank model editor and the palette browser. If the Xcos command is followed by an Xcos file name (for example, **Xcos 'abc.zcos'**), the

specified file is loaded into the model editor. The file extensions of an Xcos model are: .zcos, .Xcos and .XMI. The .zcos is the default file extension.

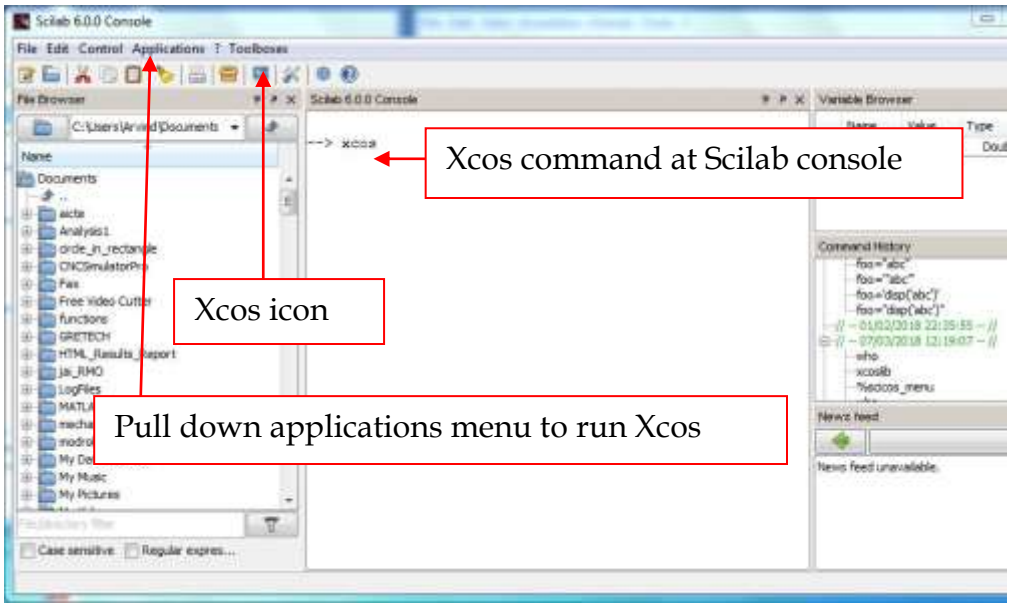


Figure 1.13: Starting Xcos

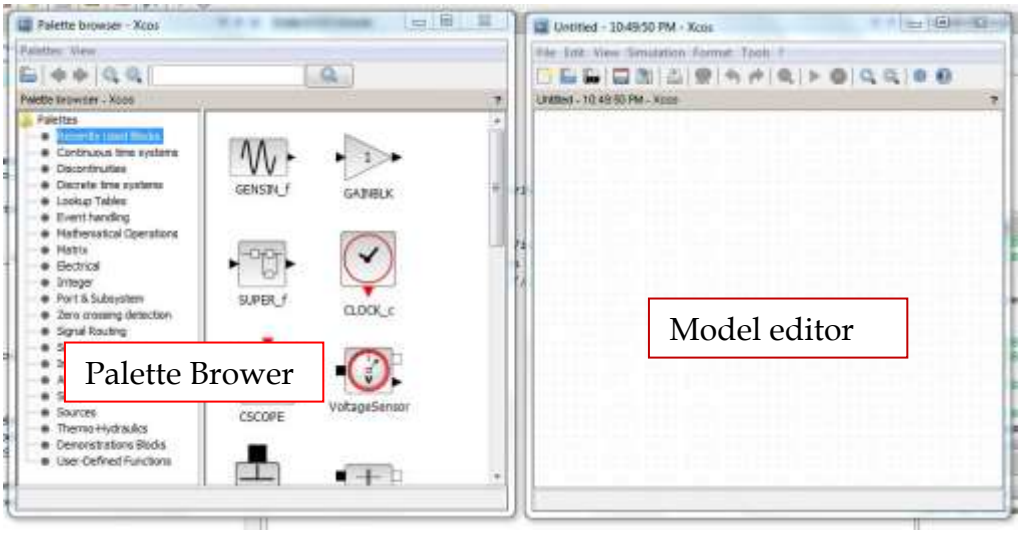


Figure 1.14: The default screen of Xcos

1.4.2. Palette browser

The palette browser window shows the collection of all the block palettes available in Xcos. Xcos has a very large collection of built-in blocks for the most common operations. These blocks are arranged in groups, known as palettes, based on their use categories. For example, mathematical blocks, such as max, sum, product, sin, etc., are included in the mathematical operations palette. For advanced users, Xcos also provides facilities to create new blocks and add them to the palette browser. Any number of user-defined palettes can be created. Figure 1.15 shows the default palette browser. The palette browser window has two panes: palettes library (in the left) and blocks collection (in the right). When we select a palette, the blocks under the selected palette are shown in the right pane. Some block names end with _f, _c or _ m, which indicates the programming languages used to write the computational functions of the blocks. For example, SINGBLK_f block is written in FORTRAN, whereas CLOCK_c is written in C and CONST_m in Scilab. For convenience of viewing, block icons can be zoomed, using zoom in and zoom out buttons on the toolbar or view menu, as needed.

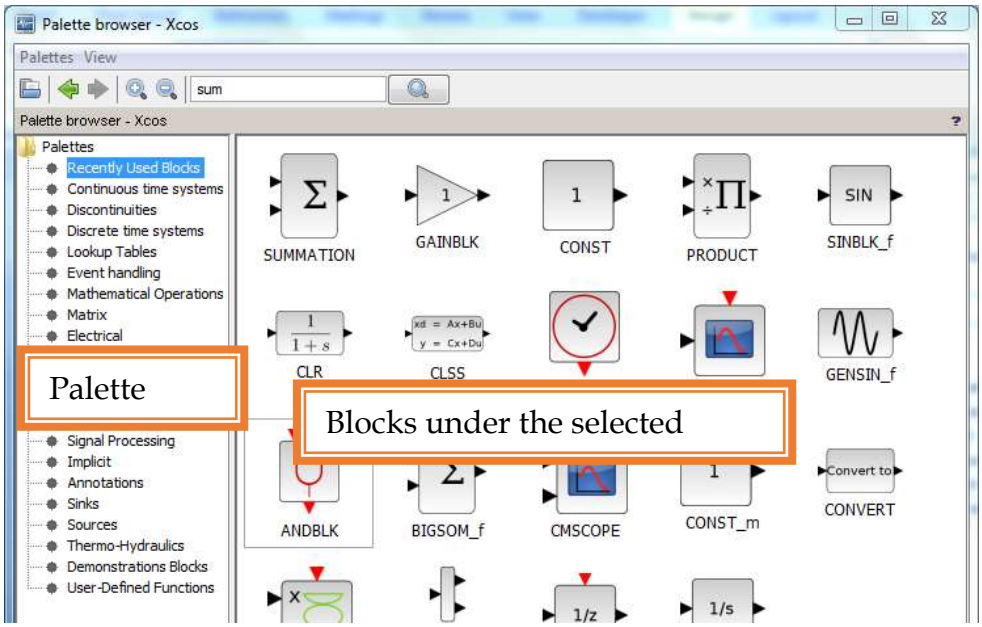


Figure 1.15: Palette browser window

The palette browser window has a number of facilities to search a block. If a user is not sure which palette contains a particular block, the search facility can be very helpful in locating the block based on its partial name. For example, suppose we want to list all the blocks that contain the word sum in their names. For this, we type sum in the search box and press search. The palette browser lists all the blocks related with sum, as shown in figure 1.16.

Xcos provides a very basic description, and sometimes an illustrative use, of a block. This help can be found by right-clicking the block's icon in the palette browser, which pops up a menu with the help button and the add to model button. Clicking the help button, the help page of the concerned block is displayed. Figure 1.17 shows how to get help of a block and figure 1.18 shows a part of the help page. The add to model button will send a copy of the selected block to the currently active model editor.

If the palette browser is closed, it can be reopened from the view menu of the model editor window.

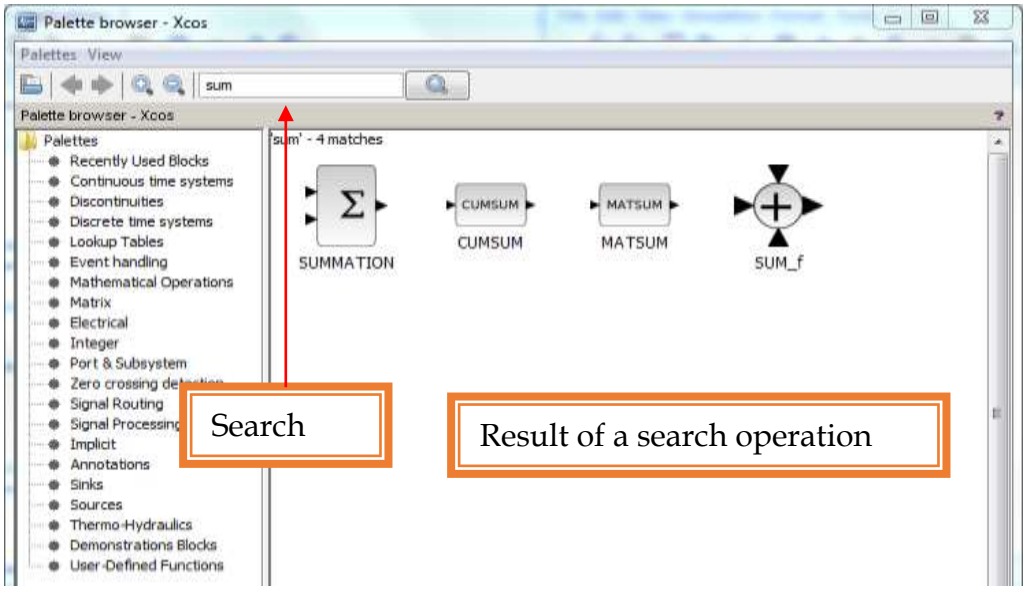


Figure 1.16: Finding blocks with the help of search command

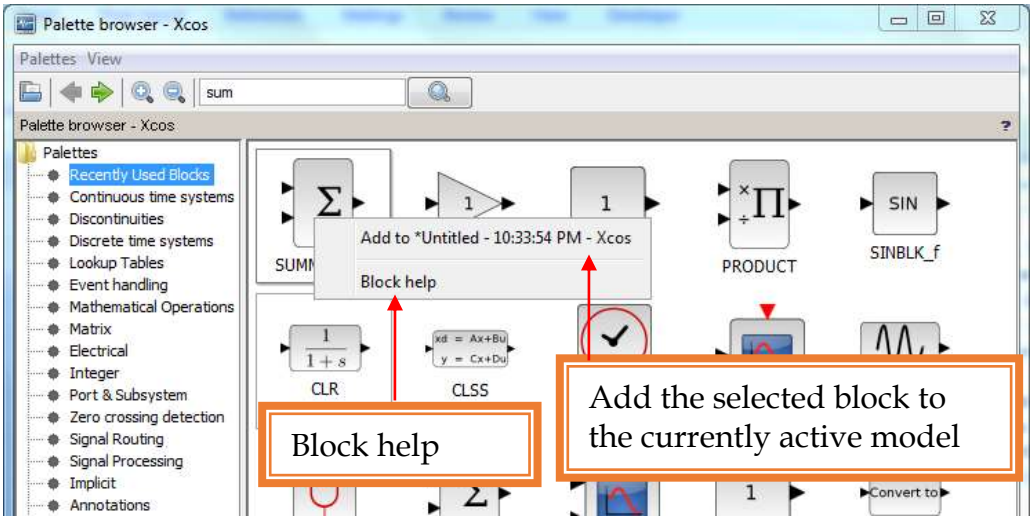


Figure 1.17: Getting the help page of a block

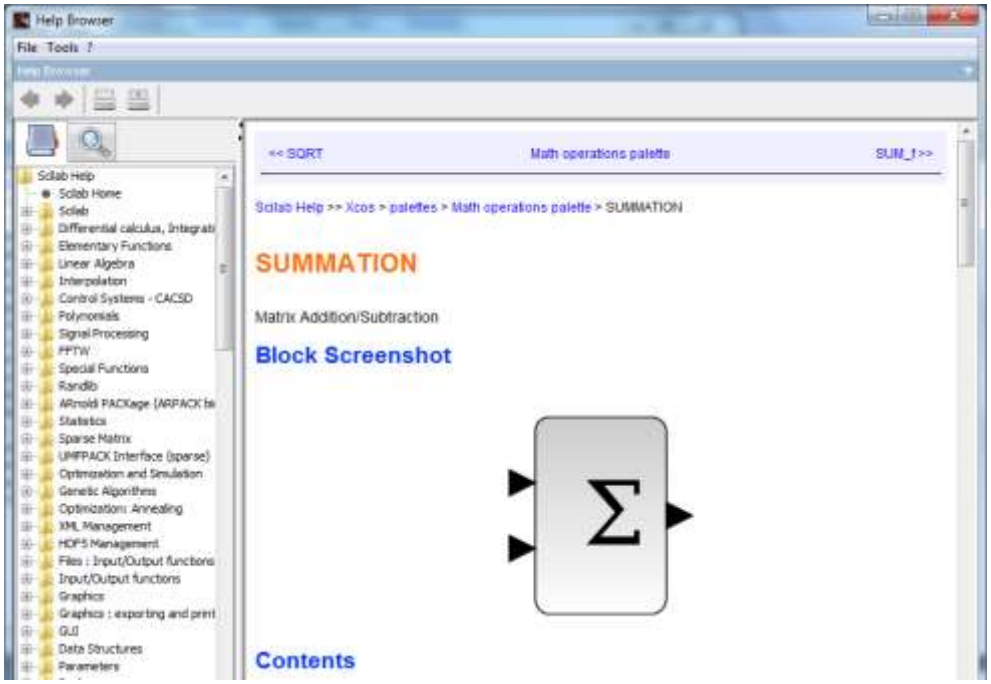


Figure 1.18: The help page of the summation block

1.4.3. Model editor

Xcos has a very powerful and easy to use graphical editor for creating models using blocks. In this section, we briefly describe various menu options for creating, managing and simulating models. All menu entries are available from the menu bar. Some commonly used menus are available on the tool bar. Further, for rapid access, the editor gives context specific pop-up menu on right mouse click. Figure 1.19 shows various options to access a menu.

All menu entries of Xcos are grouped in seven categories according to their uses: File, Edit, View, Simulation, Format, Tools and help (?).

1.4.3.1 File Menu

Figure 1.20 shows the editor with all the file group menu entries, along with their shortcuts. Since the File menu is very similar to other common windows software, it is very briefly described in Table 1.2.

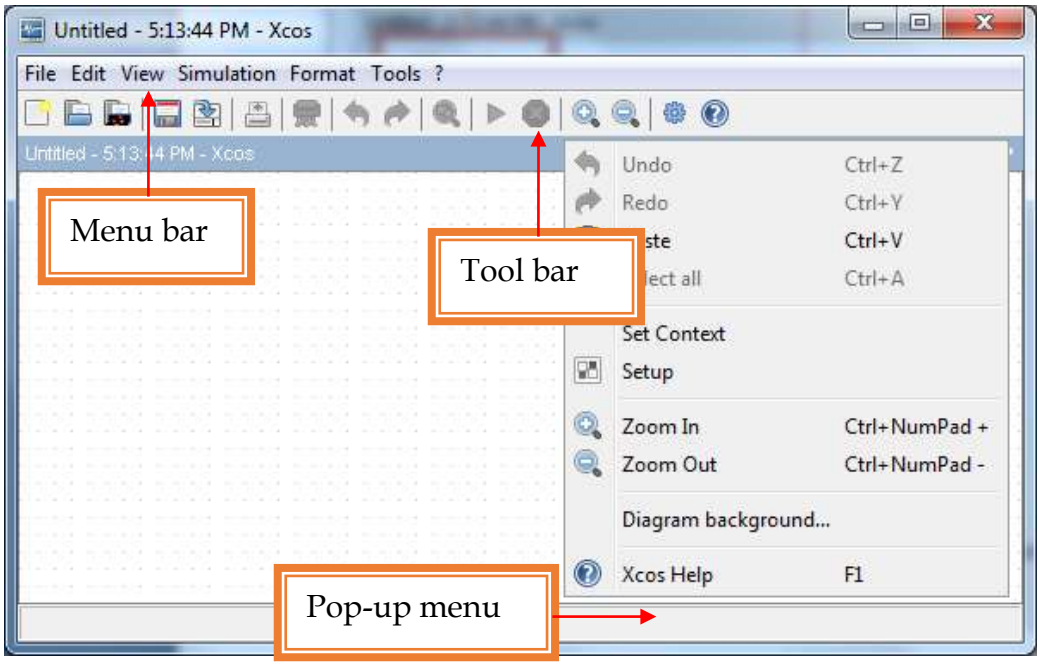


Figure 1.19: Various ways to access a command in the model editor

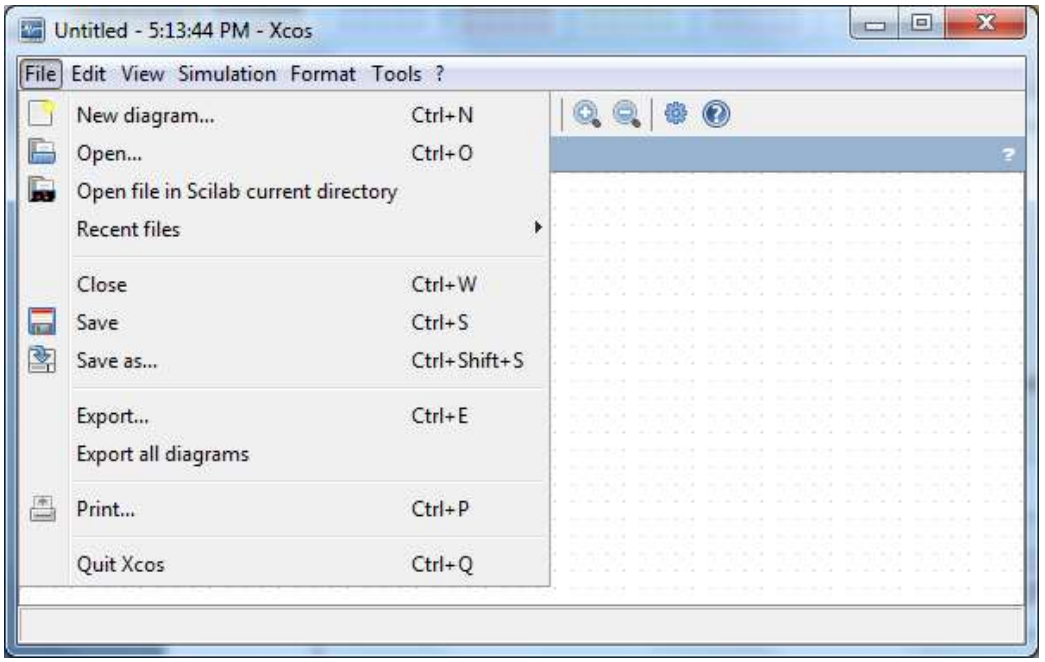


Figure 1.20: File menu entries

1.4.3.2 Edit Menu

Figure 1.21 shows the Edit menu. Again, most of the entries in the Edit menu are similar to common applications. Hence, this menu is briefly described in table 1.3.

1.4.3.3 View Menu

The View menu has commands for showing the models at different scales and other model management commands. Figure 1.23 shows all the View menu commands along with their shortcuts. Table 1.4 describes these commands.

1.4.3.4 Simulation Menu

Simulation menu commands are used to set the simulation environment (such as, simulation duration, computation accuracy, step size, etc.) and conduct the simulation of the created model. Table 1.5 lists and describes various Simulation menu entries.

Table 1.2: File menu entries and their descriptions

File menu	Description
New	The New command opens a new empty model editor window.
Open	The Open command opens a file browser window to locate and load previously saved Xcos models. The command opens a dialog box that allows you to choose the folder and the file.
Save	The Save command saves the model on the disk. If the Save command is used for the first time for a new model, a save dialog box is opened, which allows you to choose a directory and specify a file name for the model.
Save As	The Save As command opens a dialog box, which allows you to save the model with a new name in a specified folder.
Export	The Export command is used to export the current Xcos model as an image. The export can be done in the WBMP, GIF, HTML, JPEG, JPG, PNG, SVG or VML formats.
Recent Files	This command provides a quick access to the recently opened files.
Print	The Print command prints the current model.
Close	The Close command closes the current model editor window. If there is only one open model editor, the command also closes Xcos along with the model editor and the auxiliary windows, such as viewport, palettes, etc.
Quit	The Quit command ends Xcos after giving chance to save all modified models and closing all the auxiliary windows.

Table 1.3: Edit menu

Edit menu	Description
Undo	The Undo command is used to undo the last operation.
Redo	The Redo command is used to redo the last undo operation.
Cut	The Cut command is used to remove the selected objects from the model and copy them in the clipboard. When you cut a block all links connected to it are deleted as well.
Copy	The Copy command is used to place a copy of the selected objects in the clipboard.
Paste	The Paste command is used to place the current content of the clipboard in the current model.
Delete	The Delete command is used to delete the selected object form the model. When you delete a block, all the links connected to it are also deleted.
Select all	The Select all command selects all the blocks and links in the current model.
Invert selection	The Invert selection command inverts the current selection i.e. all the selected objects are unselected and previously unselected objects are selected.
Block Parameters	The Block Parameters command opens the block configuration dialog box for the currently selected block. The block configuration dialog box of a block can also be opened by double clicking the block in the model editor, or through the right mouse button click pop-up menu. Most blocks in Xcos require some parameters for their configuration, for example, the sine wave generator block requires three parameters:

	amplitude, frequency and phase angle. Figure 1.22 shows the configuration dialog box of the sine wave generator block.
Selection to superblock	The Selection to superblock command converts the currently selected blocks into a superblock.

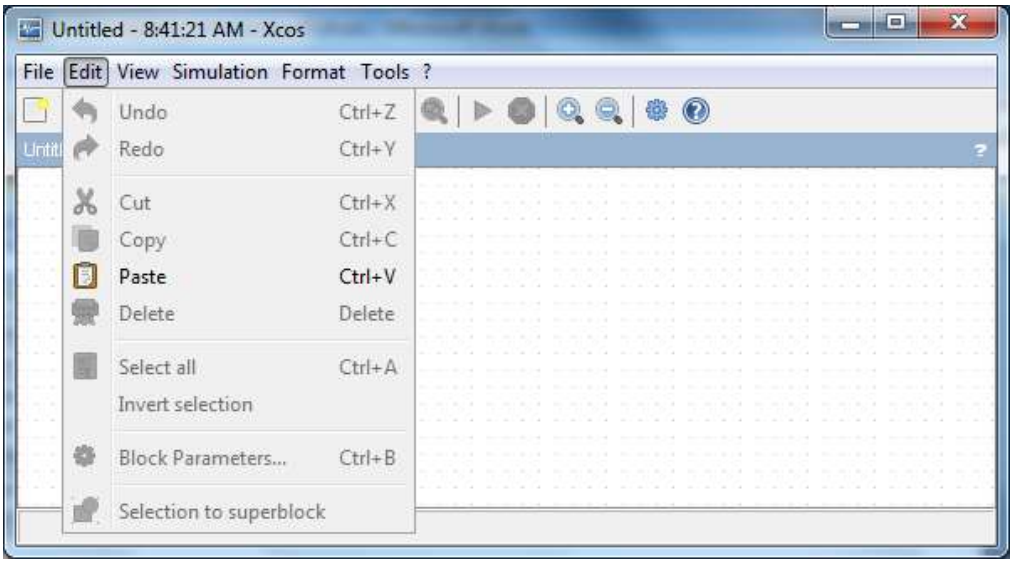


Figure 1.21: Edit menu

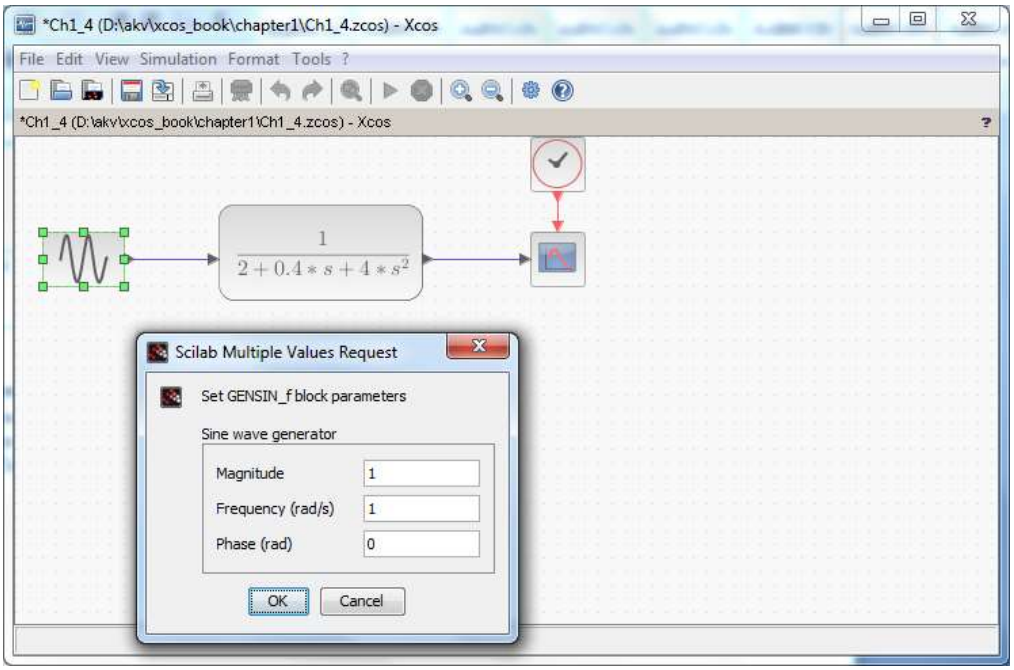


Figure 1.22: The configuration dialog box of the sine wave generator block

Table 1.4 : View menu commands

View menu	Description
Zoom in	The Zoom in command enlarges the current model view by a factor of 10%. This command is also available on the toolbar.
Zoom out	The Zoom out command reduces the current model view by a factor of 10%. This command is also available on the toolbar.
Fit diagram to view or block to view	The Fit diagram or block to view fits the selected part of the model to the size of the current window. If nothing is selected, the whole model is fitted to the current model window.

Normal 100%	The Normal 100% command resizes the model components at their normal displaying dimensions.
Palette browser	The Palette browser command toggles the palette browser. If the palette browser is open, the command closes it. If the palette browser is closed, the command opens it.
Tree_show(scs_m)	The Tree_show(scs_m) command shows the tree overview of the current model. The tree overview contains all details of a model in the form of a tree. Figure 1.24 shows a model and its tree overview.
Diagram browser	The Diagram browser command displays a window which lists the global properties of a diagram and all its objects (blocks and links). Figure 1.25 shows a model and its diagram browser window.
Viewport	The Viewport command opens a new window with the complete image of the current model and a sliding rectangular viewport. The original window shows the content of the viewport and the content is zoomed to fit in the original window. The viewport window is especially very useful for large models. With the Viewport, the working area of the model can be zoomed for proper viewing. The viewport can be resized and moved around the model image. Figure 1.26 shows use of the viewport to zoom sine block. Similarly, other portion of the model can be zoomed by moving the viewport around the model image.
Details	The Details command displays tree overview of the currently selected block.

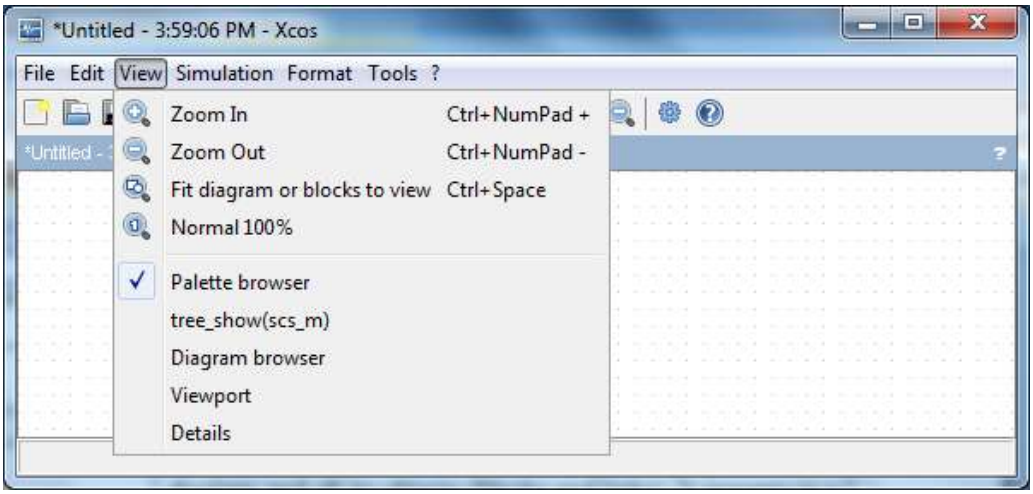


Figure 1.23: View menu entries

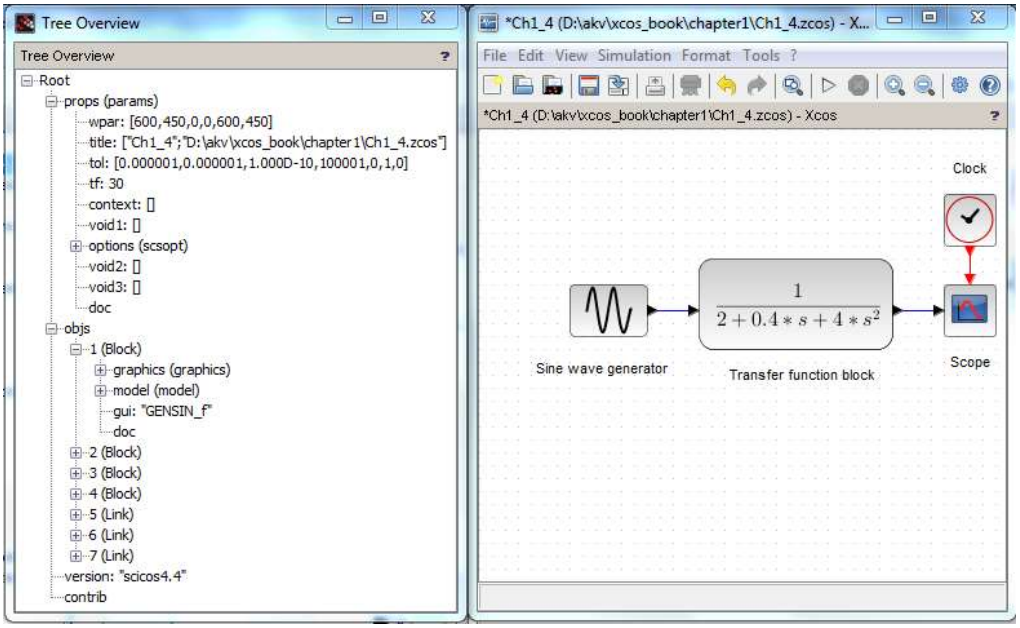


Figure 1.24: The tree overview of a model

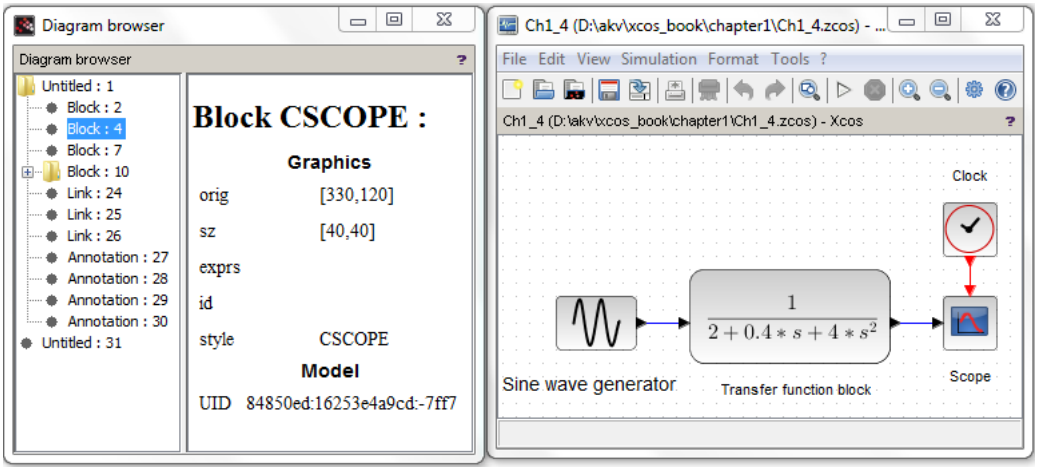


Figure 1.25: Diagram browser window

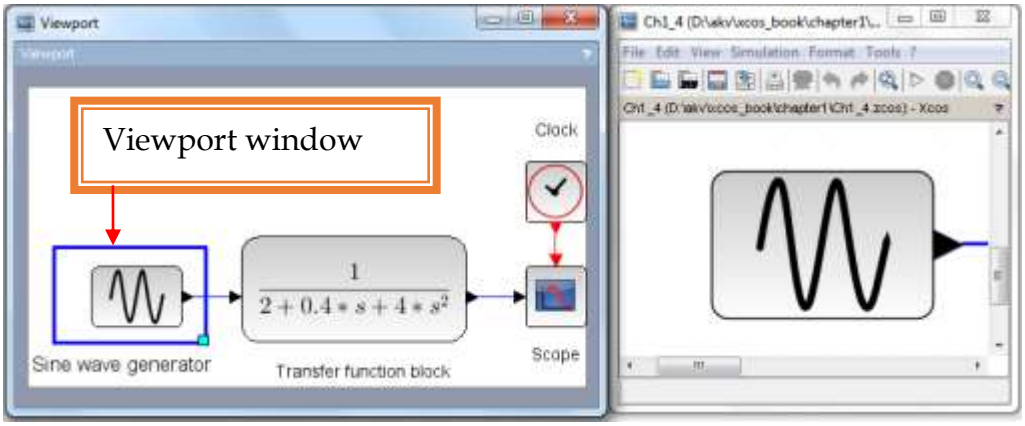


Figure 1.26: Use of viewport command

Table 1.5: Simulation menu entries

Simulation Menu	Description
Setup	The Setup command pops up the simulation environment setting dialog window as shown in figure 1.28. Through this window, the following simulation environment parameters can be modified: • Final integration time is used to set the duration of simulation run. Simulation always starts from 0. • Real time scaling forces real time simulation by setting Xcos simulation in unit of time to 1 second. • Absolute and relative error tolerances set the calculation accuracy of the numerical equation solver. • Time tolerance specifies the smallest time interval at which the ODE solver will update continuous states of the system. • Max integration time interval specifies the maximum time interval for each call to solver. It must be reduced if the error message "Too many calls" is encountered • Solver kind dropdown menu lists all the solvers available in Xcos. Based on the problem requirement, users can select an appropriate numerical solver. Users can choose either an ordinary differential equations (ODE) solver or an algebraic differential equation solver (IDA). However, if Xcos detects that the model requires an IDA solver, Xcos displays an information box on the automatic switching to the IDA solver.

	• Max step size sets the maximum time step that can be used by the solver. Xcos tries to reduce the calculation time by increasing the time step size if it observes a monotonic system response. This parameter is useful to search localized singularities in a monotonic system response. For this type of response, this parameter forces the solver to take smaller step to handle localized singularities more accurately. • The Set context button is used to create symbolic constants. Explained in more details below. • The Default button resets the simulation parameters to their factory values. • The OK button is used to accept the modifications in simulation setting and the Cancel button rejects the changes.
Set Context	The Set Context command is used to define symbolic constants and expressions which can be used in models. This command opens a dialog box where you can enter Scilab instructions to define symbolic Xcos parameters. These parameters are used in block definitions or other purposes. These instructions will be evaluated each time the model is loaded. If a parameter is modified in the context, all the blocks that contains this symbolic parameter are updated when you click on OK. For example, if we want to generate three different sine wave signals of amplitudes in the ratio of 1:2:3 for a sine wave generator, we can create one independent constant (say *A*) and two related symbolic constants (say *B* and *C*) in the Set Context dialog box and write Scilab instructions as shown in figure 1.29. During modeling, these symbolic constants (*A*, *B* and *C*) can be used in various blocks. Figure 1.29 also shows a model which uses

	these symbolic constants for setting the three gain blocks. The output of the model shown in figure 1.29 is shown in figure 1.30, which shows three sine waves with amplitudes in the ratio of 1:2:3.
Compile	The Compile command is used to compile models. However, this command is rarely used as models are compiled automatically when we run models.
Modelica initialize	Dynamical models in Xcos are normally created based on the governing equations of physical systems. However, Xcos includes Modelica blocks that can be used to create models in terms of components of physical systems. In this case, the governing equations are internally generated by Xcos and Modelica compilers. The Modelica initialize command is used to initialize Modelica compiler. A brief introduction to the Modelica modeling is given in chapter 15.
Start	The Start command, also available on the toolbar, is used to start the simulation.
Stop	The Stop command, also available on the toolbar, stops the currently running model. After stopping, you can change any of the block parameters and restart the simulation with the new values.

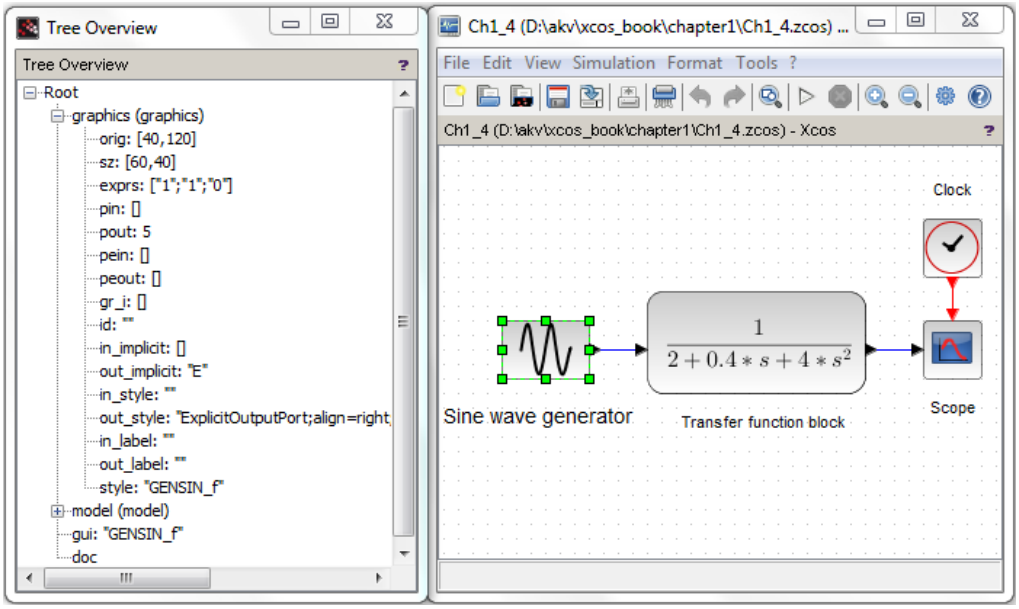


Figure 1.27: Details of a block in the tree overview form

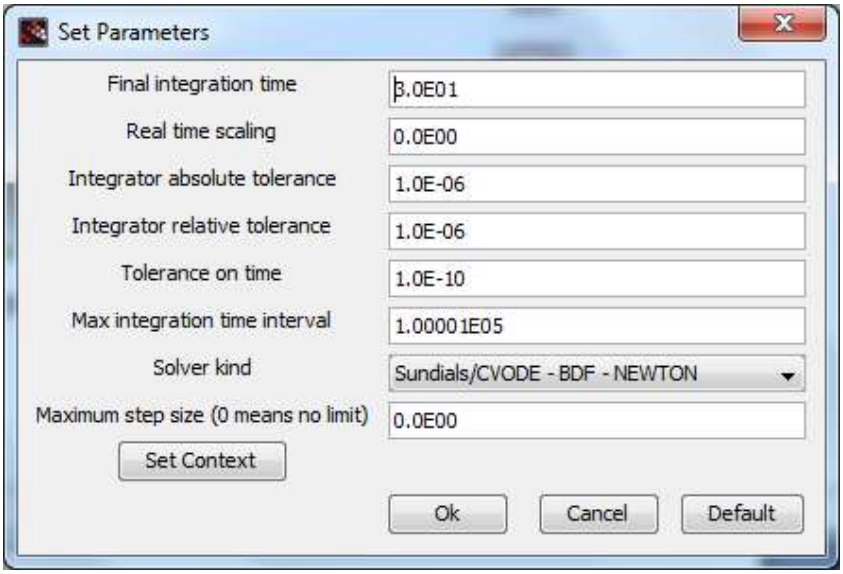


Figure 1.28: Simulation environment setup dialog box

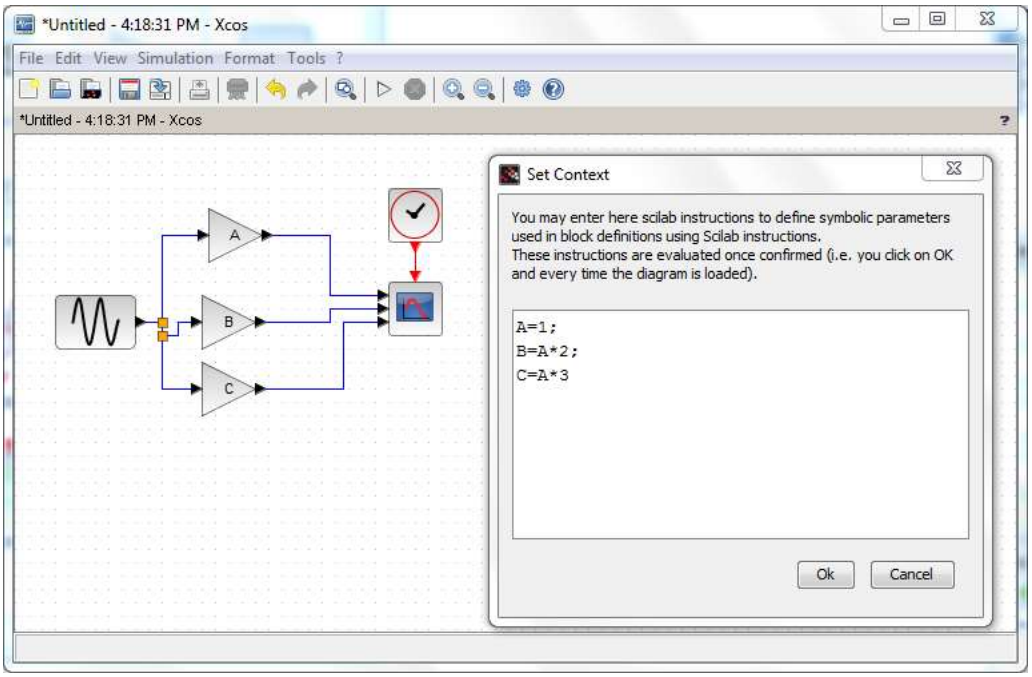


Figure 1.29: Creating symbolic constants

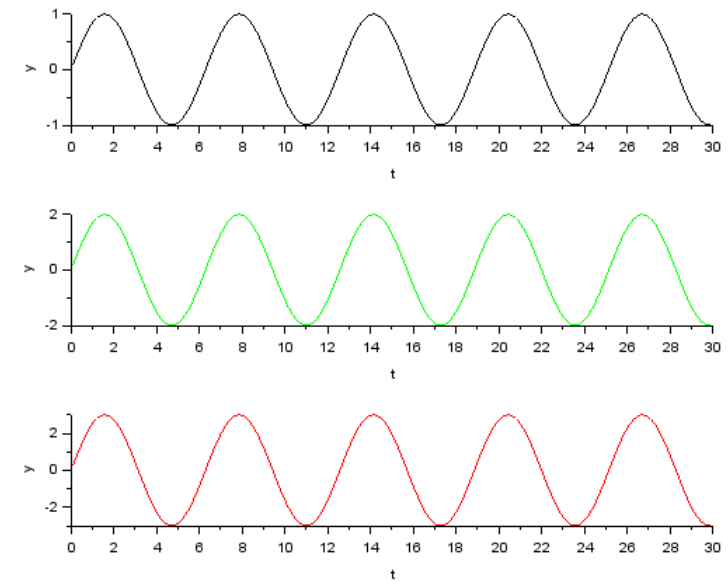


Figure 1.30: Output of the model shown in figure 1.29

1.4.3.5 Format Menu

The Format menu contains commands which are used for formatting the appearance of blocks. For better visualization of the logic flow of a model, we require to rotate, flip or mirror blocks. By doing this, we show blocks and their connections in more logical ways. For better presentation and look, we sometimes align blocks and give different colors to blocks. However, these formattings have no influence on the logic and the output of a model.

A block in Xcos is a graphical representation (say icon) of an operation as shown in figure 1.31. A block consists of a few regular input and output ports (data ports) and activation input and output ports (control signal ports). It is not necessary for each block to have all types of ports. The number of ports in a block is dependent on the operation performed by the block. For some blocks, the number of ports can be customized by changing block parameters. For example, in a summation block, the number of input ports is set by its parameter named input ports sign/gain. Ports are used to connect blocks for transferring data and control signals from one block to other. Blocks are placed in a model by dragging them from the palette browser to the model editor. Alternatively, a block can be send to the model editor by a right click on the block icon in the palette browser and then choosing Add to the model option in the pop-up dialog box. Once placed in a model, the size of a block can be changed by dragging resize handles, which are shown in green-color small squares on the boundary of the block.

Data and control signals are transmitted from one block to another through links. A link is represented by a series of line segments with an arrow head at one end. The direction of the signal flow is in the direction of the arrow head. A link can originate from an output port or from a previously defined link and can terminate at an input port or an existing link.

To start a link from a port, take the mouse pointer near the port, when a small green square appears and encloses the port, press the left mouse button and drag-out the mouse pointer. To terminate a link at a port, take the mouse pointer to the port and when a small green square appears and encloses the port, click the left mouse button. For starting a link from an existing link, take mouse over the existing link and when it turns green, press left mouse and drag out. To terminate a link at an existing link, take the mouse pointer over the link and when it turns green, click left mouse button. Any number of line segments can be added to a link by a series of left clicks before the link terminates. A link can start or end at an existing link, but one end of the link must be a port, i.e. a link cannot start from a link and ends at another link.

By default, data links (also called regular link) are in the blue color and activation links are in the red color. The color of a data link can be customized, whereas the color of activation links is fixed. Figure 1.32 shows a model with different types of links.

By default, regular input ports of a block are placed on the left-side edge of the block and output ports are placed on the right-side edge. Regular ports are black-colored filled triangles. The input ports of a block point towards the block, whereas the output ports away from the body of the block. The activation input ports of a block are placed on the top edge and the activation output ports are on the bottom edge of the block. Activation ports are red-colored filled triangles. Table 1.6 lists and describes various formatting commands available in Xcos.

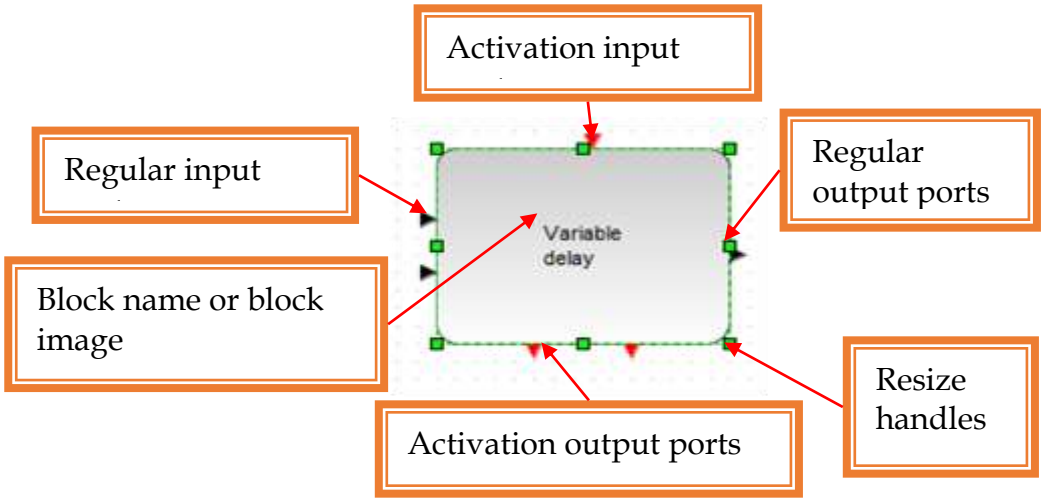


Figure 1.31: Features of a block

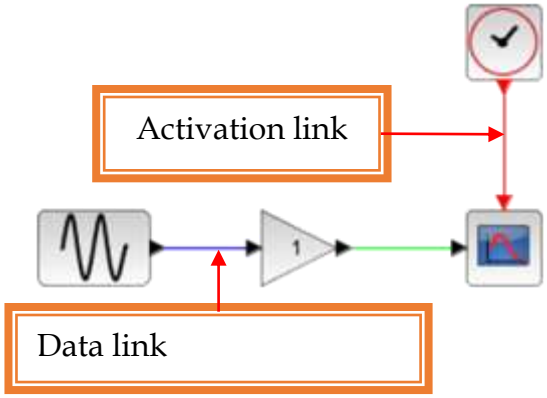


Figure 1.32: Different types of links

Table 1.6: Format menu entries

Format Command	Description
Rotate	The Rotate command rotates the selected blocks, around their centers, by 90° in counter clock wise. The links of the model are adjusted to maintain the original connections. Figure 1.33 shows a block in its default orientation and after rotation. The text block name maintains its original orientation during rotation, whereas the block image rotates along with the block.
Flip	The Flip command swaps the locations of the activation inputs and outputs, i.e. in a default block orientation, after flip command, the input activation ports goes to the bottom edge and the output ports to the top edge. In a default orientation, the activation ports of a block are numbered from left to right. Flipping does not affect the original numbering sequence of ports. Figure 1.34 shows block flipping.
Mirror	The Mirror command swaps the positions of the regular input and output ports of the selected blocks. In the default orientation of a block, the regular ports of the block are numbered from top to bottom. Mirroring does not affect the original numbering sequence of ports. Figure 1.35 shows block mirroring.
Show/Hide shadow	Currently, this command has no effect on the appearances of the selected blocks.
Align Blocks	The Align Blocks command gives options to align selected blocks in different ways. There are three ways to align selected blocks on a horizontal axis: Top, Bottom and Middle and three ways for a vertical axis: Left, Right and Center. Figure 1.36 shows all the six sub-commands of the Align Blocks command and figure 1.37 shows their illustrative uses. In the horizontal top alignment, the top edges of the

	smallest enclosing rectangles of the selected blocks are aligned along a common horizontal axis. In the horizontal center alignment, the horizontal centerlines of the smallest enclosing rectangles of the selected blocks are aligned on a common horizontal axis. In the horizontal bottom alignment, the bottom edges of the smallest enclosing rectangles of the selected blocks are aligned along a common horizontal axis. In the vertical left alignment, the left edges of the smallest enclosing rectangles of the selected blocks are aligned along a common vertical axis. In the vertical center alignment, the vertical centerlines of the smallest enclosing rectangles of the selected blocks are aligned along a common vertical axis. In the vertical right alignment, the right edges of the smallest enclosing rectangles of the selected blocks are aligned along a common vertical axis.
Border Color	The Border Color command is used to change the border color of the selected blocks. Figure 1.38 shows the effect of border color change.
Fill Color	The Fill Color command is used to change the fill color of the selected blocks. Figure 1.38 also shows the effect of fill color change. Fill color and border color change facilities can be used to represent hot and cold entities in the model to make the presentation more impressive.
Auto-Position	In Xcos, to send a data signal to more than one block, the signal link is split into the desired number of links. However, during this process, the original link and new links become zigzag. The Auto-Position command is used to align them properly as

	shown in figure 1.39.
Link Style	A link between two blocks can be of different styles: Horizontal, Straight, Vertical and Optimal. Different link styles are shown in figure 1.40. In a horizontal link, the link horizontally starts from the source block and terminates horizontally to the target block. The link may bend at 90° if required. In a vertical link, the link vertically starts from the source block and also terminates vertically to the target block. The link may bend at 90° if required. In a straight link, the link is a single straight line between the source block and the target block. In an optimal link, the link may start/terminate vertically or horizontally from the block depending on the block orientation. The link may bend at 90° if required. Users can draw a link in any shape, using any number of line segments. The Link style command can be used to automatically set the selected links in a desired style. Figure 1.41 shows various link style commands. These commands are also available on right mouse click pop-up window if links are selected.
Diagram background	By default, Xcos sets the background of a new model to white color. The Diagram background command can be used to change the background color of a model. Figure 1.42 shows a model with light green color.
Grid	The Grid command activates / deactivates the background grid of a model. With the grid, the block and link placement in the model is easier and results in a more readable model.

Blocks in their default

Blocks after rotation

Figure 1.33: Block rotation

Original Flipped block Rotated and flipped

Figure 1.34: Block flipping

Original block Mirrored block Mirrored and flipped Rotated and Mirrored Rotated, mirrored and flipped

Figure 1.35: Block mirroring

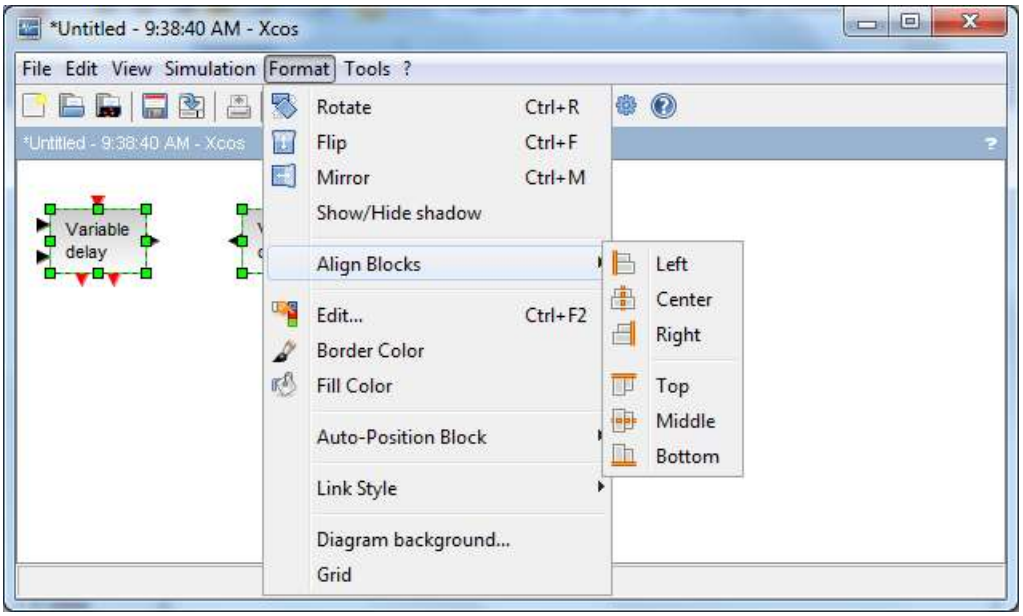


Figure 1.36: Block alignment commands

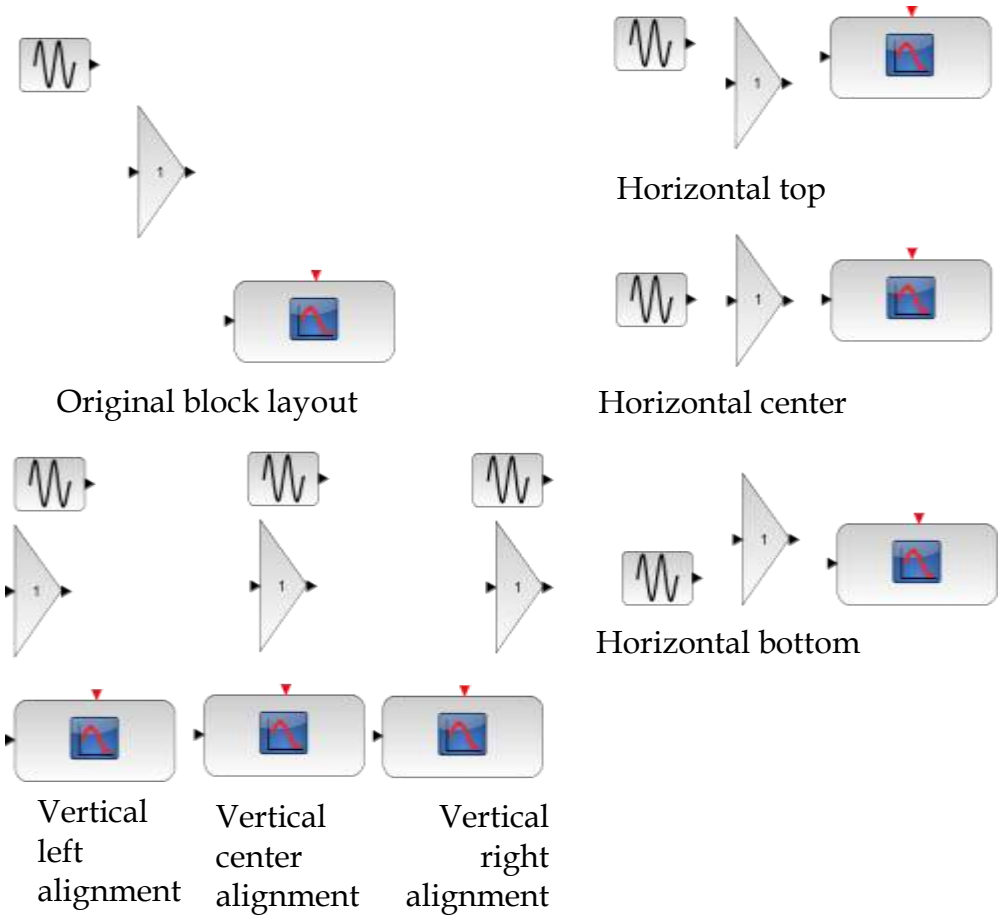


Figure 1.37: Various block alignments

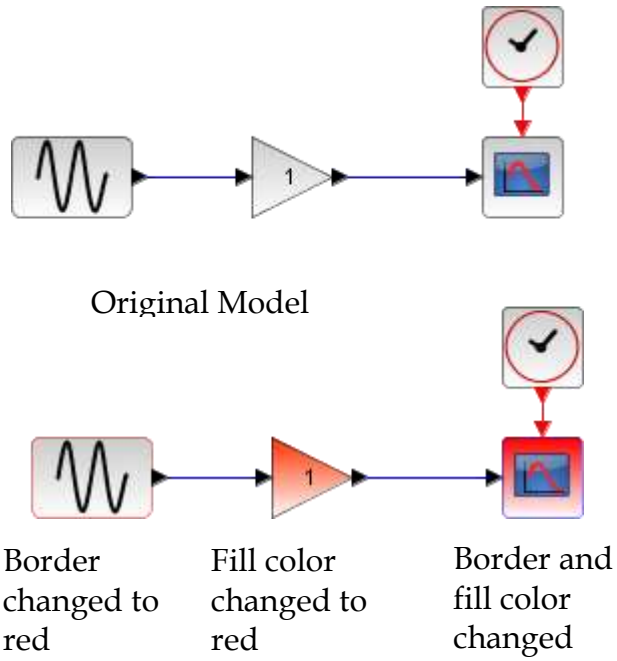


Figure 1.38: Changing colors of blocks

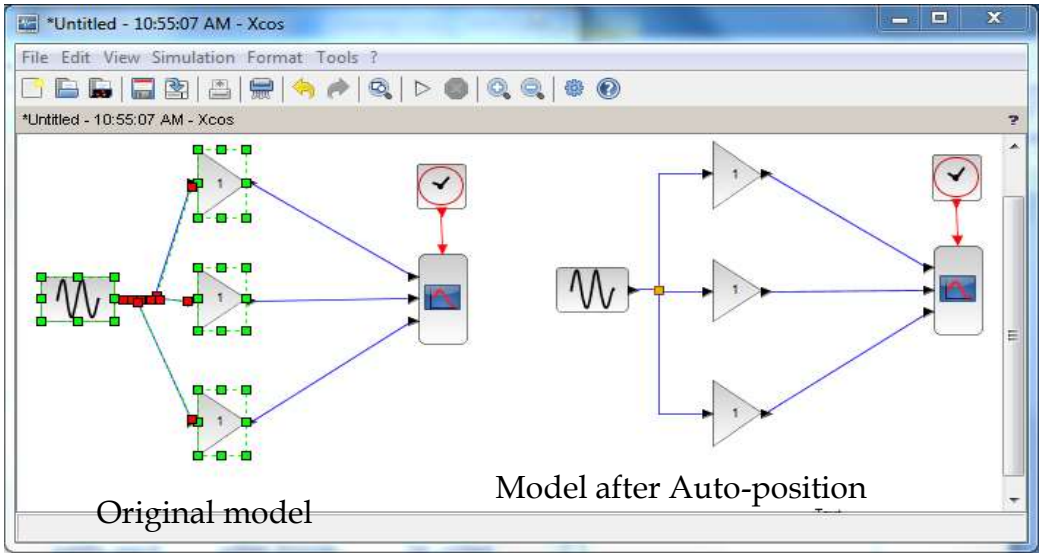


Figure 1.39: Auto-position command

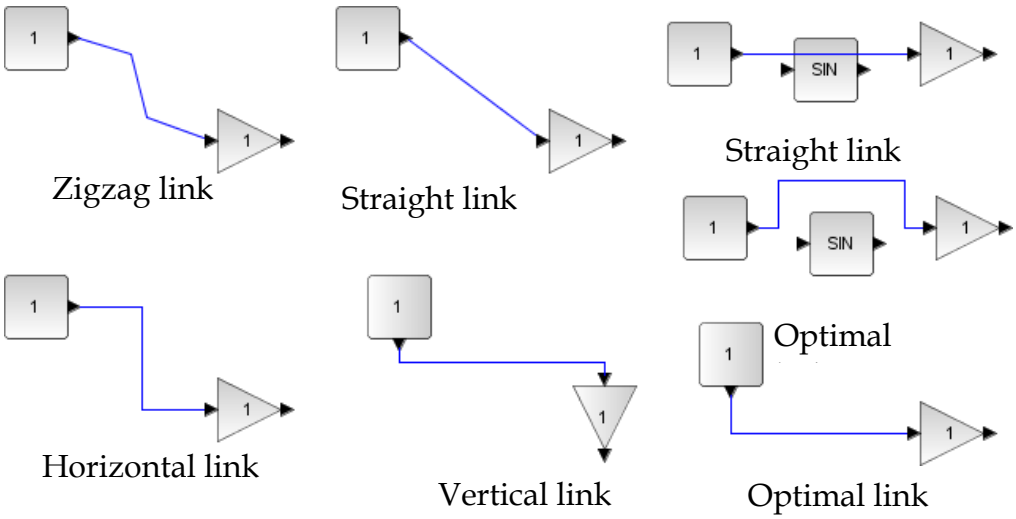


Figure 1.40: Various link styles

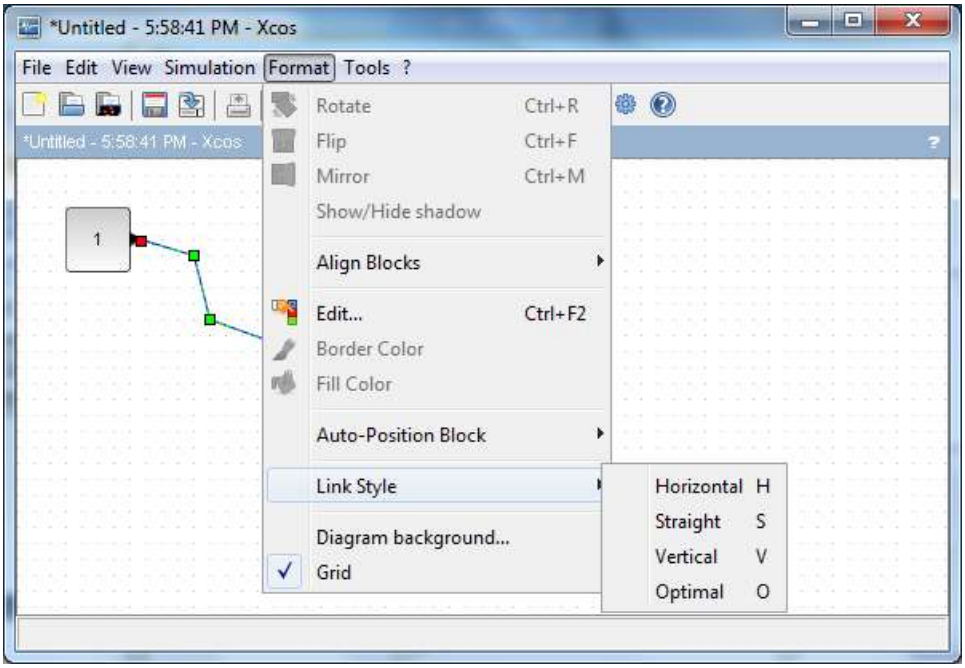


Figure 1.41: Various link style commands

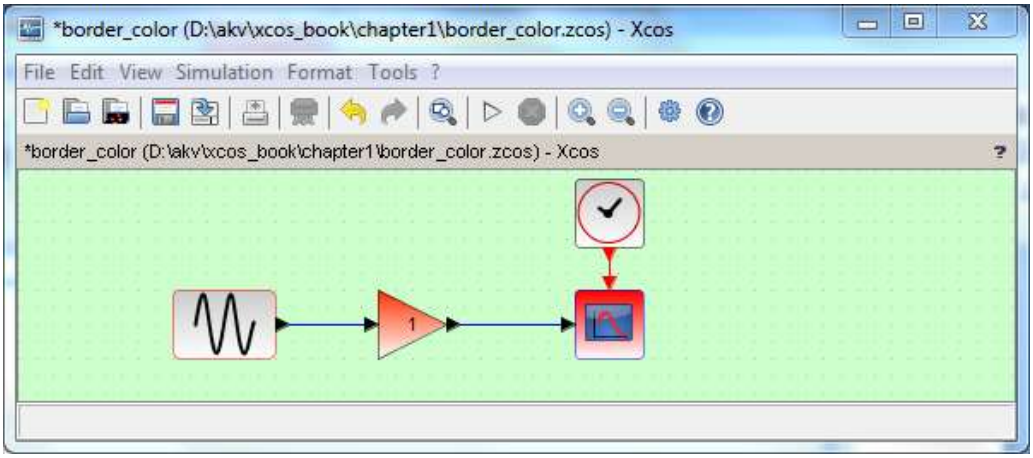


Figure 1.42: A model with changed background

1.4.3.6 Tools Menu

The Code generation command, as shown in figure 1.44, is used to create a new customized block that is equivalent to a discrete time super block. A super block is basically a collection of a number of blocks and links to implement an operation. The content of a super block can be viewed by double clicking on it.

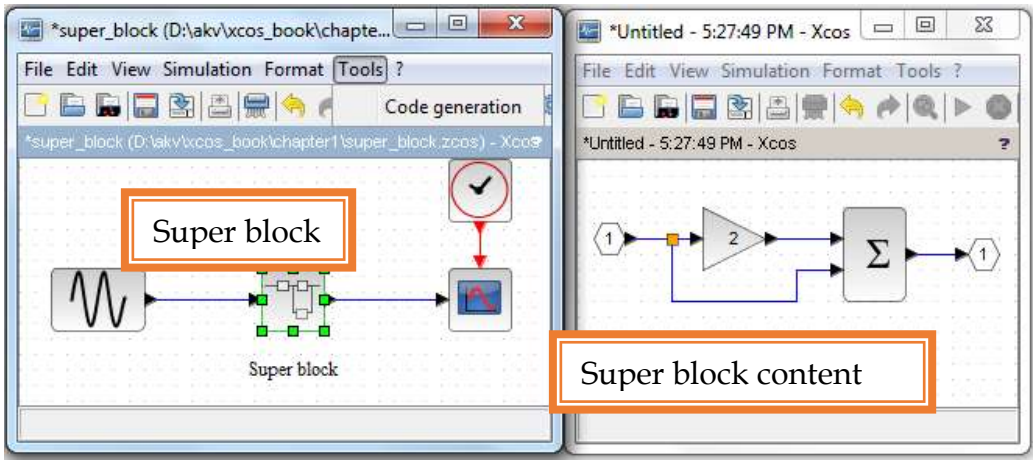


Figure 1.43: Code generation menu

1.4.3.7 ? Menu (Help Menu)

The help menu provides links for various help pages related with Xcos. It also provides a link to various demo models included in Scilab. Readers are encouraged to experiment with these demo models and explore help pages.

1.5 Model Creation in Xcos

Model creation in the Xcos environment involves the following activities:

a. Open a new model editor window
b. Add required blocks from the palette browser into the model editor
c. Arrange the blocks according to the flow of signal/data and connect the blocks
d. Set the block parameters
e. Set the simulation parameters and run the model.

1.6 Step by step construction of an Xcos model

In this section, we, step-by-step, illustrate the construction of an Xcos model to find out the voltage variation with time across the capacitor in response to a step voltage supply for a simple electrical circuit as shown in figure 1.44. Based on the Kirchhoff's voltage law, the governing equation for the voltage variation across the capacitor is given in equation 1.21. For the purpose of an Xcos model of equation 1.21, it is rewritten as given in equation 1.22. Following steps are used to create the model:

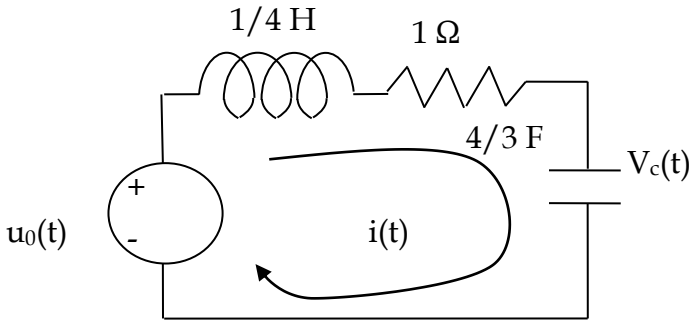


Figure 1.44: A simple electrical circuit

$$V_c''(t) + 4V_c'(t) + 3V_c(t) = 3u(t) \qquad 1.21$$

$$V_c''(t) = 3u(t) - 4V_c'(t) - 3V_c(t) \qquad 1.22$$

Since Equation 1.22 is a second ODE, we require two integrator blocks to solve it (Refer section 1.3.2). The first integrator block will take $V_c''(t)$ as input and gives $V_c'(t)$ as output. The second integrator block takes $V_c'(t)$ as input and gives $V_c(t)$ as output. Further we require a step signal block to generate activation input signal, $u(t)$, a gain block and a summation block to combine various terms. And Finally we require a scope block to display the solution., i.e. $V_c(t)$ and a clock pulse generator to activate blocks.

Step 1. Open a new model editor. Rename the new model with a relevant name, here it is renamed as equation_1_22.zcos, by saving it for the first time. It is always recommended to give a meaningful name to a model. This helps searching models later.

Step 2. Drag and drop a step signal generator block from the sources palette.

Step 3. Similarly, add a gain block from the mathematical operations palette.

Step 4. Align these two blocks along a horizontal line and connect the output port of the signal generator and input port of the gain block. For connecting blocks, bring the mouse pointer near the port you want to start the connection and when a small green square appears enclosing the port, click left-mouse button and drag the mouse pointer to the other port. A connection can be started either from the output port of the source block or the input port of the target block. Figure 1.44(a) shows the model after step 4.

Figure 1.44(a): The model after step 4

Step 5. Set the gain parameter of the gain block to 3. For this, double click the gain block and enter 3 in the gain text box. Leave the other parameters unchanged. Use the default values of the parameters of the step block.

Step 6. Similarly, bring a summation block from the mathematical operations palette, two integrator blocks from the continuous palette, one scope block from the sink palette and one clock block from the source palette and arrange them and connect them as shown in figure 1.44(b). By a double click in any blank area of the editor, a text block is added to the model. Use text blocks to label various blocks. Change the size of the summation block and change the number of port parameter to [1; 1; 1]. By changing this parameter, three input ports are shown. Change the initial state parameter of Integrator1 to 0 and Integrator2 to 0.5. Keep the other blocks with their default parameter values.

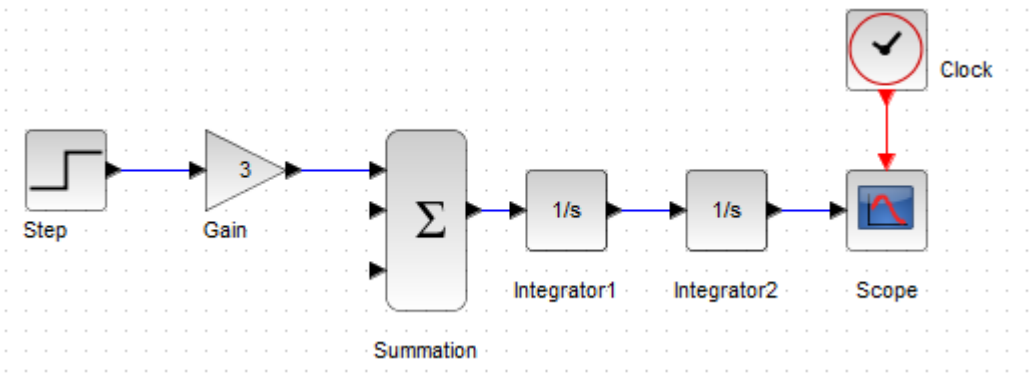


Figure 1.44(b): The model after step 6

Step 7. Now, add two gain blocks. Set the gain of Gain1 to -4 and of Gain2 to -3. Mirror the Gain1 and Gain2 blocks. Connect input port of Gain1 to the output link of Integrator1 and input port of Gain2 to the output link of Integrator2. Finally, connect the output ports of Gain1 and Gain2 to the two un-connected input ports of the Summation block. Arrange the model as shown in figure 1.44(c).

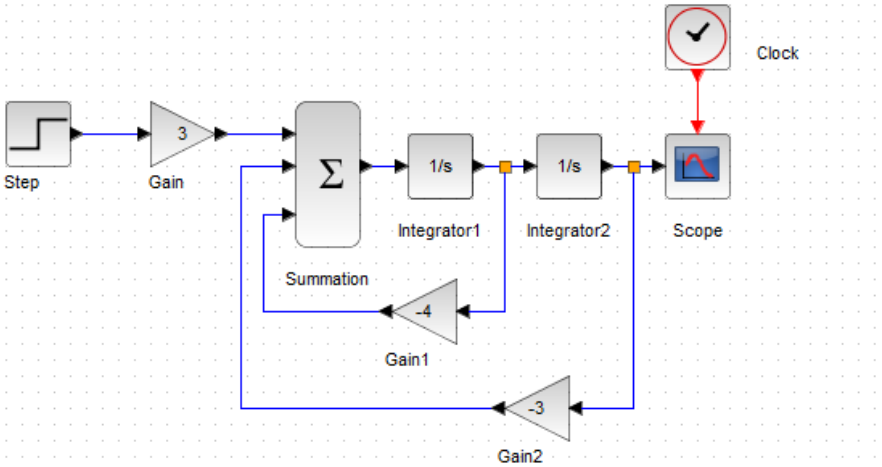


Figure 1.44(c): The final model

Step 8. Set the Ymin and Ymax parameters of the scope block to 0 and 1, respectively. Set the simulation time to 30 seconds and run the simulation. The output of the simulation is shown in figure 1.45.

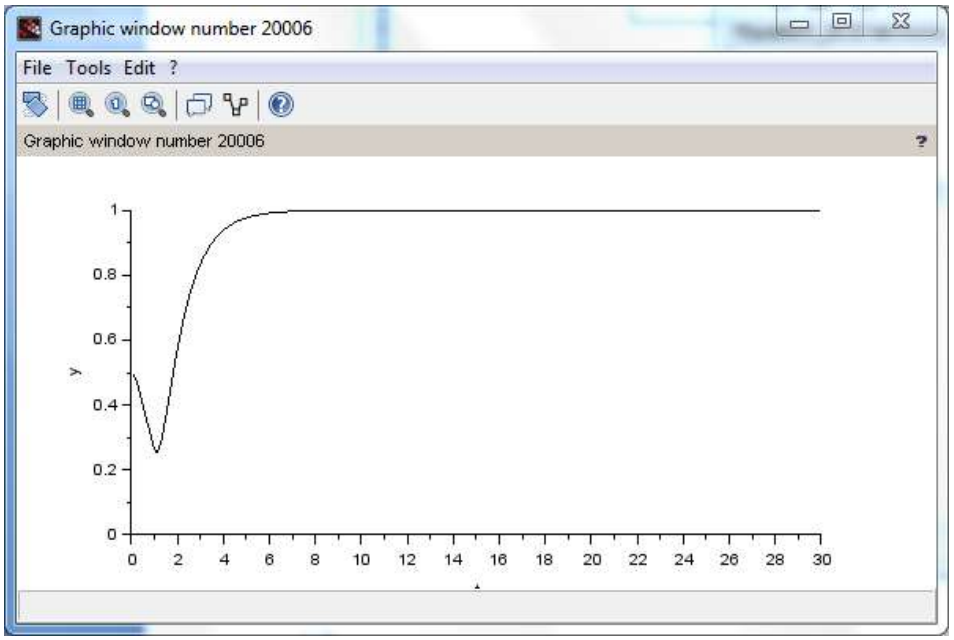


Figure 1.45: The output of the model

1.7 Inside an Xcos block

An Xcos block is a complex object, with multiple inputs and outputs, and can handle continuous-time vector states, a list of discrete-time matrix states, zero-crossing surfaces, etc. Figure 1.46 shows an overview of the internal modules, using a block diagram, of a general Xcos block. The continuous time module is used to simulate continuous time dynamical systems, whereas, the discrete time model is used for discrete time systems. The derivative module is used to calculate derivatives and the mode and zero crossing modules ensure accurate processing of inputs that contains abrupt changes making them non-differentiable at some points. The zero crossing module processes them in piece-wise manner to get required accuracy. However, every block does not require all the modules. Every Xcos block is associated with a simulation function, also called computational function, which is responsible for the block behaviors during simulation. The simulation function is called automatically when a block is activated during simulation. Further, the behavior of a block is dependent upon the way it is activated. The block parameter module manages the block parameters.

There are primarily four ways in which a block can be activated: activation signal, inheritance, always active and zero-crossing. In the activation signal mode, a block is activated by arrival of external activation signals at its activation input ports. If the same source is used to generate activation signals for two or more blocks, their activation is synchronized. If two different sources are used to generate activation signals for different blocks, even if the time of the generation of the signals is the same, their activation in not guaranteed to be synchronized by the simulator. When an Xcos model contains more than one activation sources, the simulation compiler computes activation sequences separately for each source. During the simulation, it is the timing of the activations that determines the order in which the events are fired, and consequently the blocks are activated. In case, two activation sequences have identical timing, then the order of firing of the blocks is arbitrary. If synchronization is essential for the logic, the same source must be used to activate the concerned blocks. Example 1.1(a) shows synchronized (right model) and asyncronized activation (left model) of blocks. As can be seen from the outputs, the output in asynchronous activation is not as per the actual output expected.

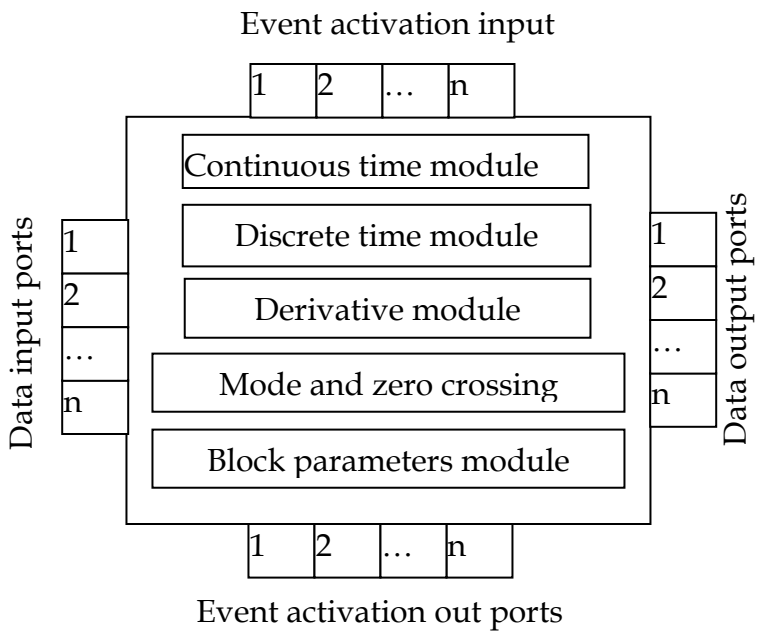
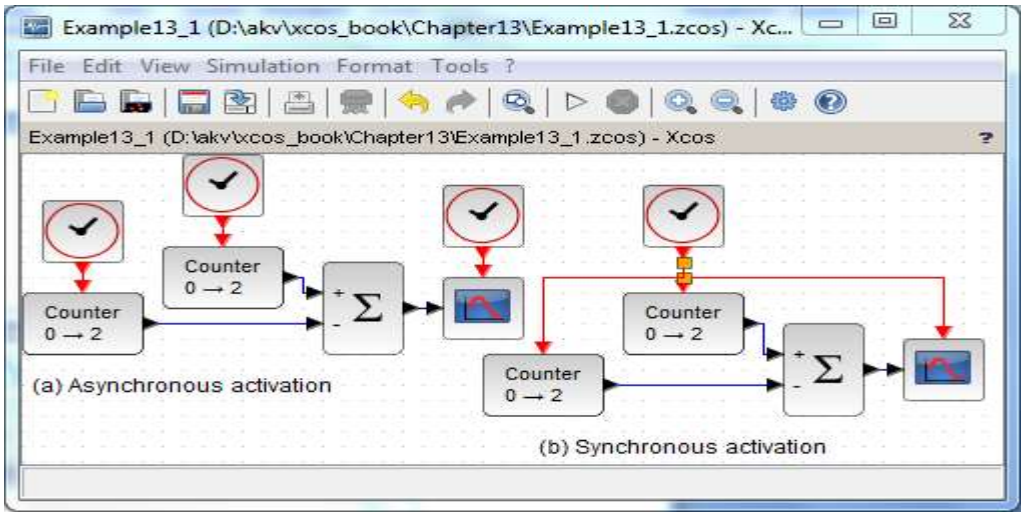


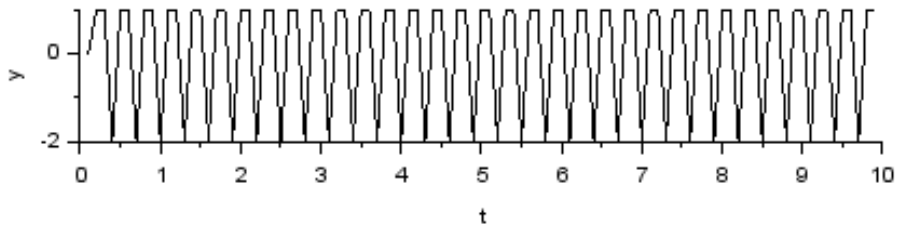
Figure 1.46: Inside of an Xcos block

When a block has more than one activation input ports, the block is activated at the union of the all the activation times of the various input activation ports. Example 1.1(b) illustrates the above concepts in which, the logical AND block generates outputs whenever any one or both of the activation signal generators fire.

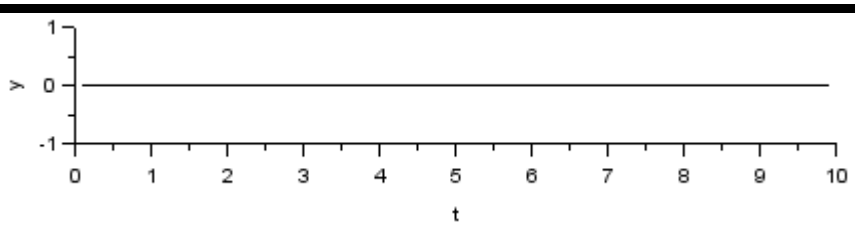
Example 1.1(a): Synchronized and asyncronized activation of blocks



Output

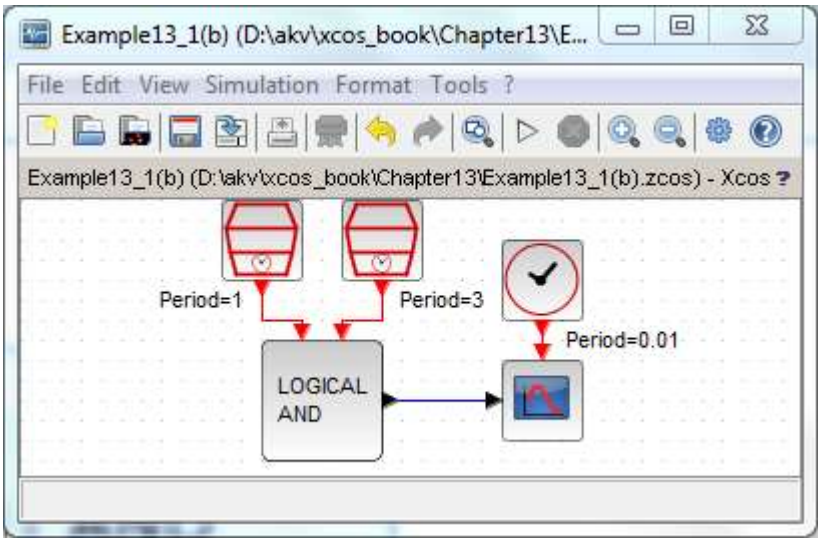


(A) Asyncronized activation

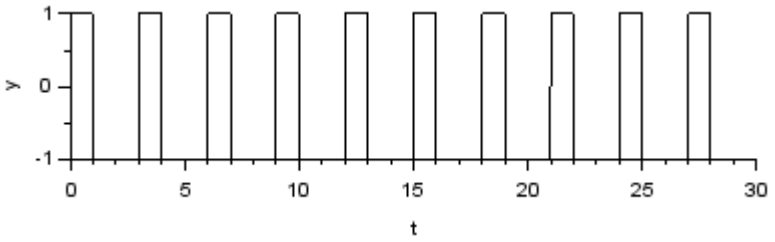


(B) Syncronized activation

Example 1.1(b): Activation of a block with more than one activation port



Output



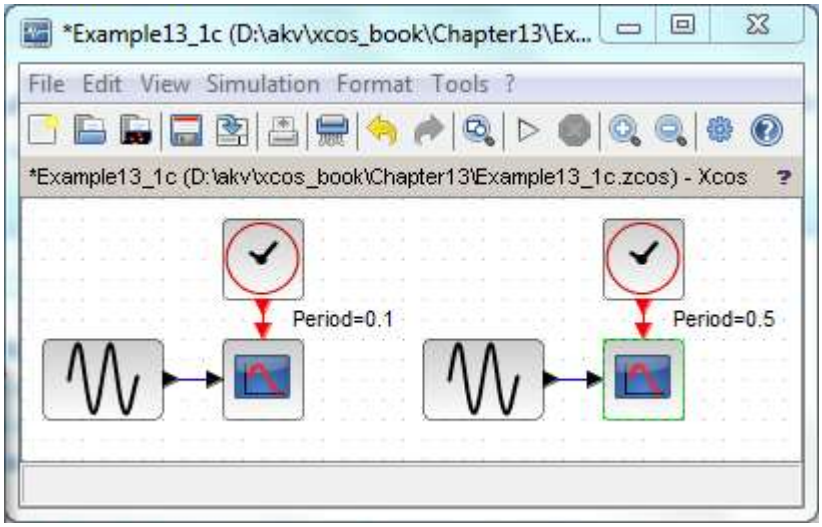
In the inheritance activation mode, there is no activation input port in the block and the block is activated at every change in the regular input signals. If the block has more than one regular input, the inheritance mechanism activates the block at every change in any of the input signals. In example 1.1(a), the summation block is activated through inheritance mechanism.

In the always active mode, a block is declared always active and generates output as per the requirement. For example, neither the sine wave generator has any activation input port nor it inherits from any other block, but it generates output based on the down line blocks requirements. In these blocks, the simulation

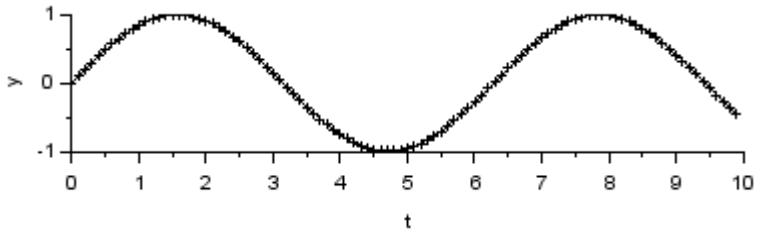
functions are called depending on the down line requirements. In example 1.1(c), the sine wave generator generates output based on the period of the scopes. As can be seen from the outputs, the sine wave generators are activated according to the periods of the scopes.

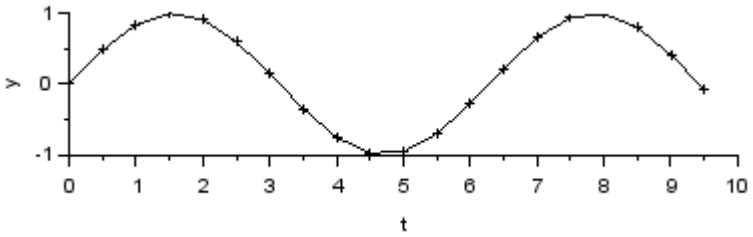
The constant block is an exception to the always active block which calls the simulation function only once at the start of the simulation and maintains that output throughout the simulation. Blocks inherited only to constant blocks also call their simulation functions only once and maintain the outputs throughout the simulation. Example 1.1(d) illustrates the above concept, in which two constant blocks and one summation block behaves like a constant activation block.

Example 1.1(c): Always active blocks

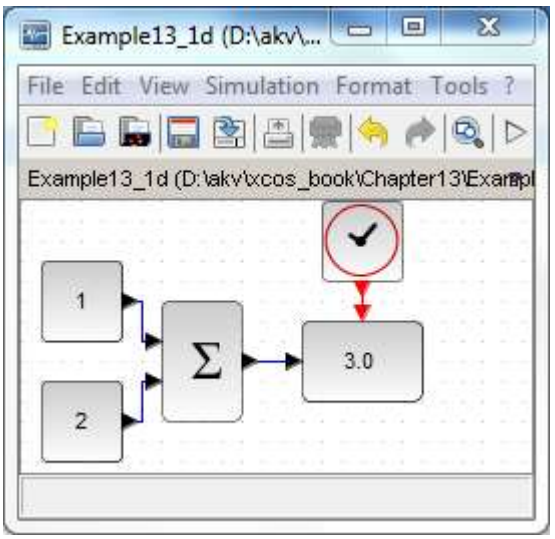


Output





Example 1.1(d): Constant blocks



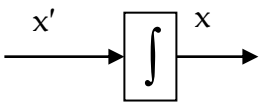
The zero-crossing activation mode is activated in a block which monitors internal zero-crossing. If a block is designed to handle internal zero-crossing, the block automatically calls the simulation function on the occurrence of such situations. This type of activation is typically used where the input is not continuously differentiable for the whole range of the input data, but it can be differentiable in piece wise manner. For example, the bouncing balls demo block of Demonstration blocks palette, in which balls abruptly changes their directions.

Most blocks in Xcos have parameters that are specified by the user specific to his requirements. On double clicks of a block icon, the block parameter setting window pops up (can also be opened using right click context menu or edit menu or Ctrl+B key). Block parameters can be either numeric constants or valid Scilab expressions using variables that are defined in the context of the model. The workspace variables cannot be directly used in the expressions. Block parameters are managed by the interfacing function, discussed in chapter 13.

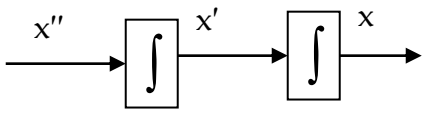
Chapter 14

Solutions of a few Differential Equations using Xcos

As stated in chapter 1, Xcos is a very powerful and open source toolbox in Scilab for modeling and simulation of dynamical systems. In this chapter, Xcos is used to create graphical models for solving differential equations to solve few practical problems of engineering and science. The basic strategy to solve a first order differential equation in Xcos is to rearrange the given equation in the form $x' = f(x,t)$ and feed the x' to an integrator block and get the value of x. The value of t is generated independently by a timer block and the value of x is taken from the previous output of the integrator block. The scheme is shown in figure 14.1(a). Similarly, for a second order differential equation, rearrange the equation in the form of $x'' = f(x,x',t)$ and use two integrator blocks to solve the equation. The first block takes x'' as input and gives x' as output and the second integrator block takes x' as input and gives x as output. The scheme is shown in figure 14.1(b).



(a) First order equations



(b) Second order

Figure 14.1: General schemes to solve differential equations in Xcos

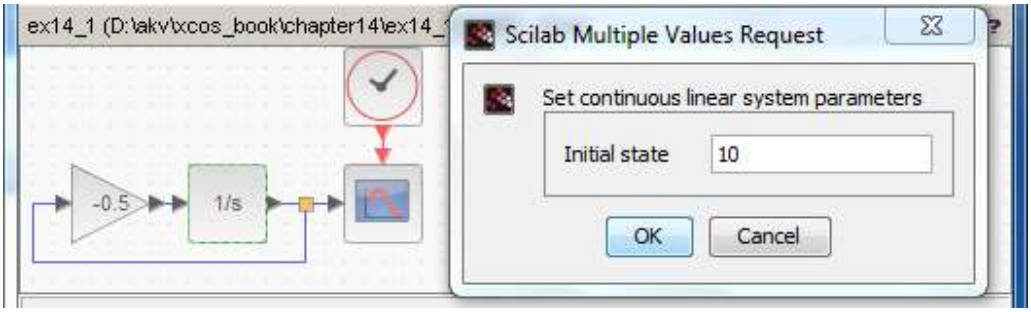
14.1 Exponential growth and decay of a population

The growth and decay of a population with time can be modeled using a differential equation of the form given in equation 14.1(a), where *P(t)* is the population at time *t* and $\frac{dP(t)}{dt}$ is the rate of change of the population under the assumption that there is no migration of the population. Further, *b* is the birth rate and *m* is the mortality rate. The analytical solution of equation 14.1(a), with initial population P_0, is given in equation 14.1(b), as the equation is separable. The Xcos model to solve equation 14.1(a) (with P_0 =10 and *k*= -0.5) is given in example 14.1 along with the output.

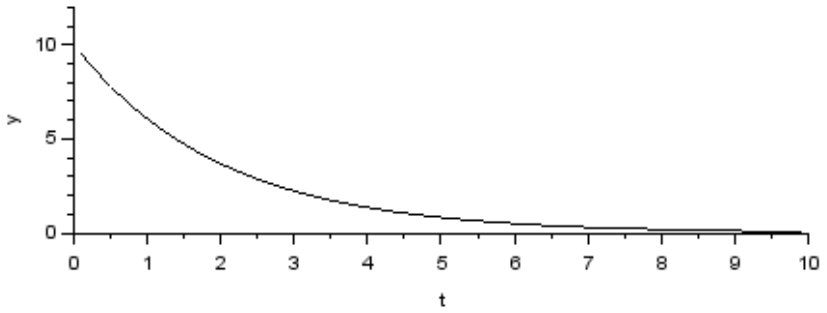
$$\frac{dP(t)}{dt} = bP(t) - mP(t) = (b-m)P(t) = kP(t) \qquad 14.1(a)$$

$$P(t) = P_0 e^{-kt} \qquad 14.1(b)$$

Example 14.1: Exponential Growth and Decay of population



Output

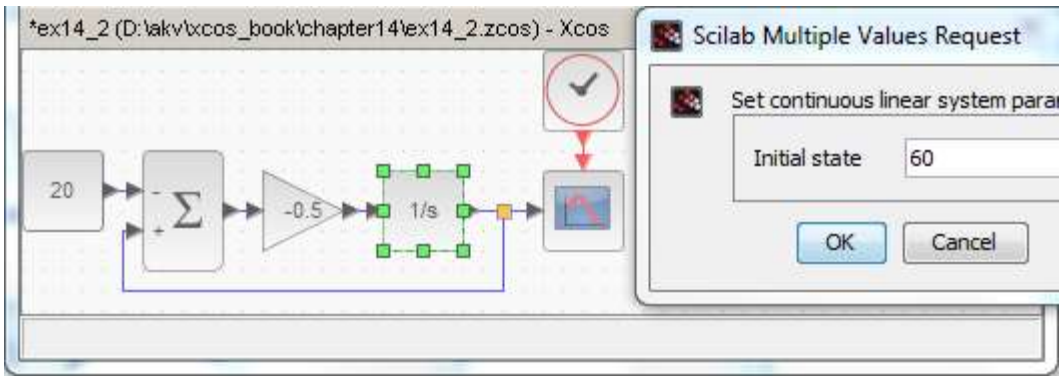


14.2 Newton's law of cooling

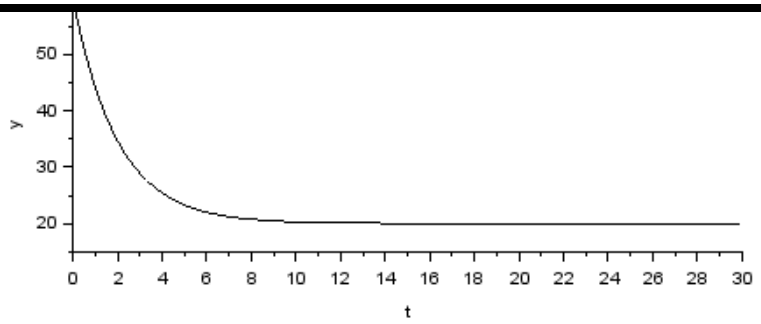
The Newton's law of cooling states that the rate of change in the temperature of a body is proportional to the instantaneous temperature difference of the body and its surroundings and can be mathematically represented as given in equation 14.2, where T_a is the ambient temperature, k is is the cooling constant and T is the temperature at time t. Example 14.2 shows an Xcos model of equation 14.2, with the initial temperature of the body 60°, ambient temperature 20° and the cooling constant is k=0.5 per second.

$$\frac{dT}{dt} = -k(T - T_a) \qquad 14.2$$

Example 14.2: Newton's Law of cooling



Output



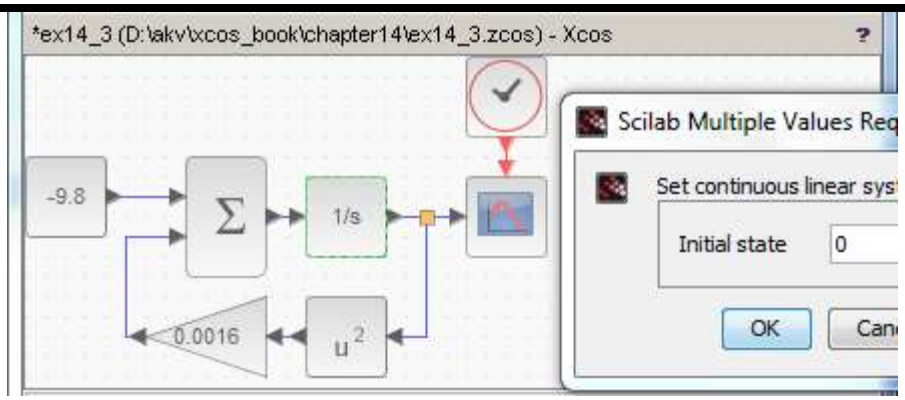
14.3 Free fall with drag

Free fall of a body near the Earth surface with air resistance can be modeled with equation 14.3(a), where m is the mass of the body, g is the acceleration due to gravity, v is the instantaneous velocity and $f(v)$ is the air resistance (which is a function of velocity). Typically, the air resistance is modeled as quadratic in the velocity, $f(v)=bv^2$. Hence, equation can be rewritten as given in equation 14.3(b), where $k = b/m$. Example 14.3 shows an Xcos model of a free falling body in air with $v_0 = 0$ m/s, $g = 9.8$ m/s^2 and $k = 0.0016$ m^{-1}.

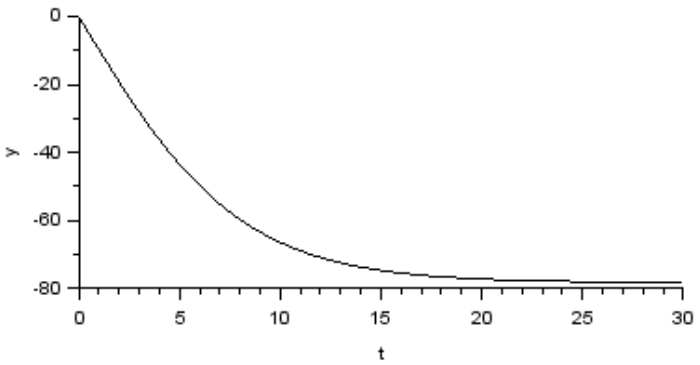
$$m\frac{dv}{dt} = -mg + f(v) \qquad 14.3(a)$$

$$\frac{dv}{dt} = -g + kv^2 \qquad 14.3(b)$$

Example 14.3: Free falling body with air resistance



Output



14.4 Logistic model of population growth and decay model

In section 14.1, a simple population growth and decay model is simulated. However, in reality, when population gets large enough, there is competition for resources, such as space and food, which can lead to a higher mortality rate. Thus, the mortality rate may be a function of the population size, $m = m(y)$. This resource constraint model of population growth is known as the logistic model. The simplest model would be a linear dependence, $m = cy + a$. With this consideration, equation 14.1(a) can take the form of equation 14.4, where $k=b-a$. The Xcos model of equation 14.4 is given in example 14.4, with $k=1$, $c=1$ and $P_0=0.1$.

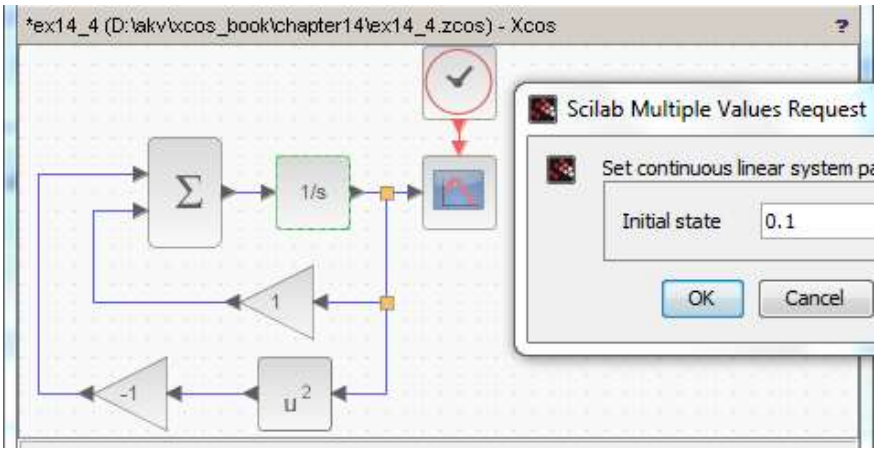
$$\frac{dP(t)}{dt} = kP(t) - cP^2(t) \qquad 14.4$$

14.5 Pursuit curve problem

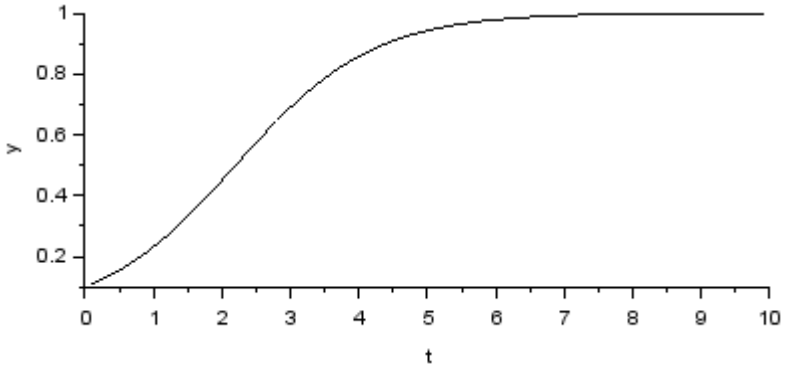
A pursuit curve is the path that a body traces when the body moves towards another moving or static body. The pursuit curve problem can be used to model an aircraft or a submarine following a target, or a predator following a prey.

Consider a classic pursuit curve problem that involves a fast pirate ship which pursues a treasure ship which tracks along a straight line. Also consider, the ratio of the speeds of the pirate and treasure ships is $r > 1$ (which is fixed) and the pirate captain sees the treasure ship initially at a distance d km away and start to follow the treasure ship, always moving toward the treasure ship as shown if figure 14.2.

Example 14.4: Logistic population grown model



Output



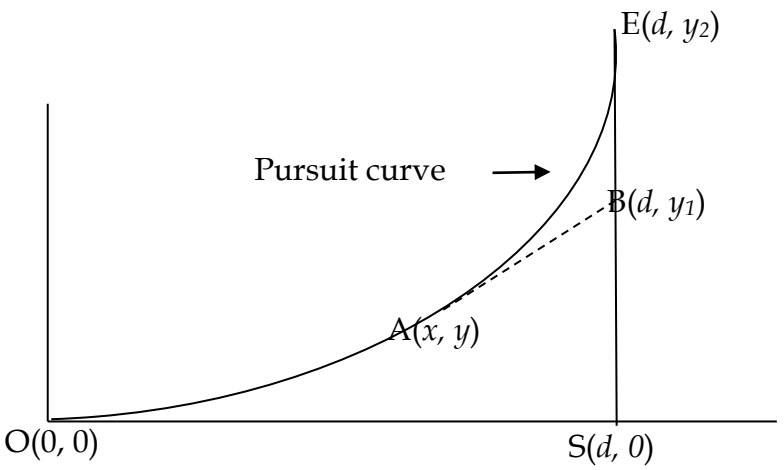


Figure 14.2: Pursuit curve problem

Consider a simple case where the treasure ship sails along the line $x = d$. At any time, let the pirate ship be at point A(x, y) and moves in the line of sight toward the treasure ship is at point B(d,y_1), which results into the basic relationship given in equation 14.5(a), based on the slope of the line of sight. By rearranging the terms, Equation 14.5(a) can be written as equation 14.5(b). Differentiating equation 14.5(b) with respect to x, equation 14.5(c) is derived.

$$\frac{dy}{dx} = \frac{y_1 - y}{d - x} \qquad 14.5(a)$$

$$y_1 = (d - x)\frac{dy}{dx} + y \qquad 14.5(b)$$

$$\frac{dy_1}{dx} = (d - x)\frac{d^2y}{dx^2} - \frac{dy}{dx} + \frac{dy}{dx} = (d - x)\frac{d^2y}{dx^2} \qquad 14.5(c)$$

The arc length along the pursuit curve, for a small segment, can be approximated to $ds = \sqrt{(dx)^2 + (dy)^2}$. The relation between the pirate speed and treasure ship speed is given in equation 14.5(d). Equation 14.5(d) can be rewritten as equation 14.5(e) by eliminating *dt* and dividing both side by *dx*.

$$\frac{ds}{dt} = \frac{(\sqrt{(dx)^2 + (dy)^2}}{dt} = r\frac{dy_1}{dt} \qquad 14.5(d)$$

$$r\frac{dy_1}{dx} = \sqrt{1 + (\frac{dy}{dx})^2} \qquad 14.5(e)$$

Using equations 14.5(c) and 14.5(e), equation of the pursuit curve can be written as given in equation 14.5(f), which can be used to simulate the pursuit path. Example 14.5 shows an Xcos model of a pursuit curve problem of equation 14.5(g) with *d*=15 and *r*=2. The simulation time is set to 15 (equal to *d*), as timer values are used as x-coordinates of the pirate ship and the simulation ends when *x* equals *d*. In the model, the constant block used for *d* is set to a value slightly larger (here, 15.00001) than *d* to avoid division by zero.

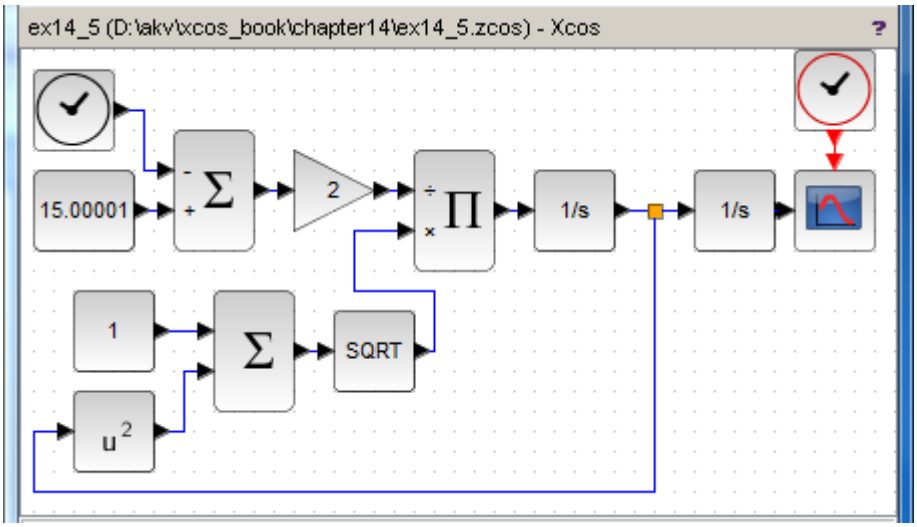
$$r(d - x)\frac{d^2y}{dx^2} = \sqrt{1 + (\frac{dy}{dx})^2} \qquad 14.5(f)$$

$$\frac{d^2y}{dx^2} = \frac{\sqrt{1 + (\frac{dy}{dx})^2}}{r(d - x)} \qquad 14.5(g)$$

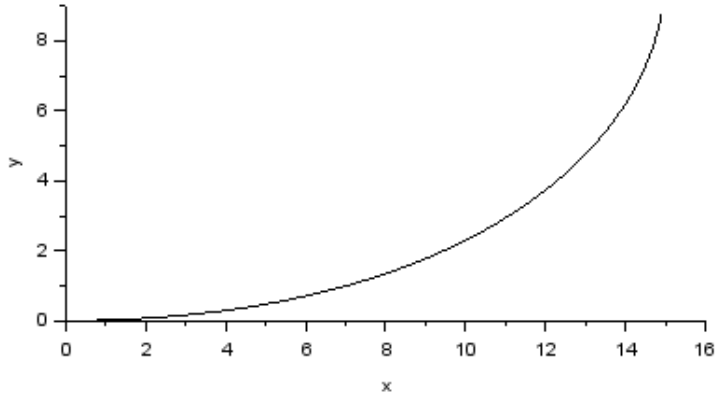
14.6 Damped simple harmonic motion

A simple spring mass damper system is shown in figure 14.3. The oscillating motion of the mass is governed by equation 14.6(a), where m is the mass of the oscillating object, b is the damping coefficient and k is the spring constant. Example 14.6 shows an Xcos model to solve equation 14.6(b), with m=2, b=0.3 and k=5 (all in the SI unit system) and the initial value of x_0=1 and $x_0' = 0$.

Example 14.5: Pursuit curve simulation



Output



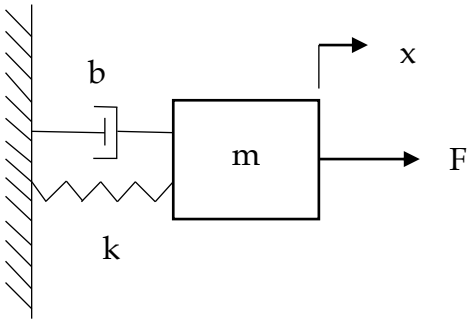
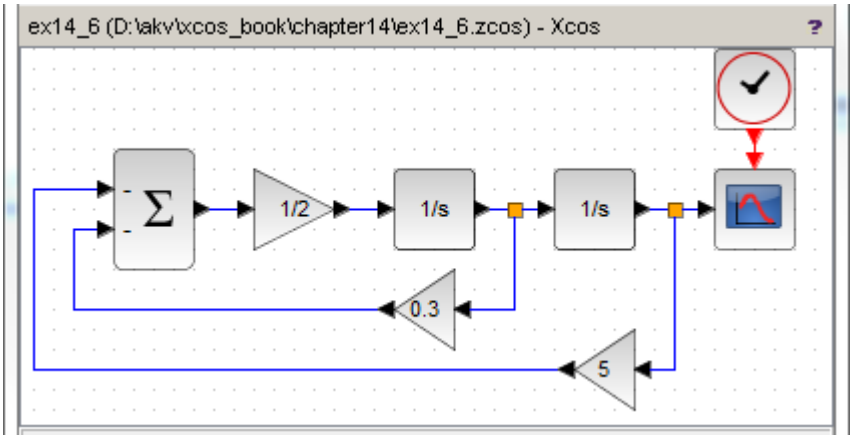


Figure 14.3: A simple spring mass damper system

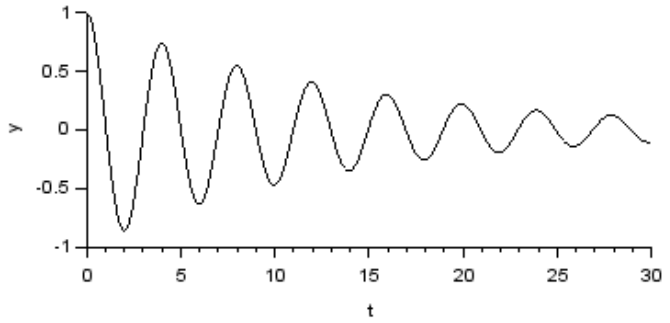
$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = 0 \qquad 14.6(a)$$

$$\frac{d^2x}{dt^2} = \frac{-b\frac{dx}{dt} - kx}{m} \qquad 14.6(b)$$

Example 14.6: Damped oscillation simulation



Output



14.7 Forced simple harmonic motion

The governing equation of a general forced simple harmonic motion of a spring-mass-damper system is given in equation 14.7(a), here dot notation is used for derivatives. Other notations are similar to equation 14.6(a). To simulate such systems, one needs only to add *f(t)* to the sum that is sent into the first integrator block of the model shown in example 14.6. Example 14.7 shows an Xcos model of a simple forced and damped harmonic oscillation with *f(t)*(here a step function as given in equation 14.7(b)), m=2, b=0.3 and k=5 (all in the SI unit system) and the initial values of x_0=1 and $\dot{\mathbf{x}}_0 = \mathbf{0}$.

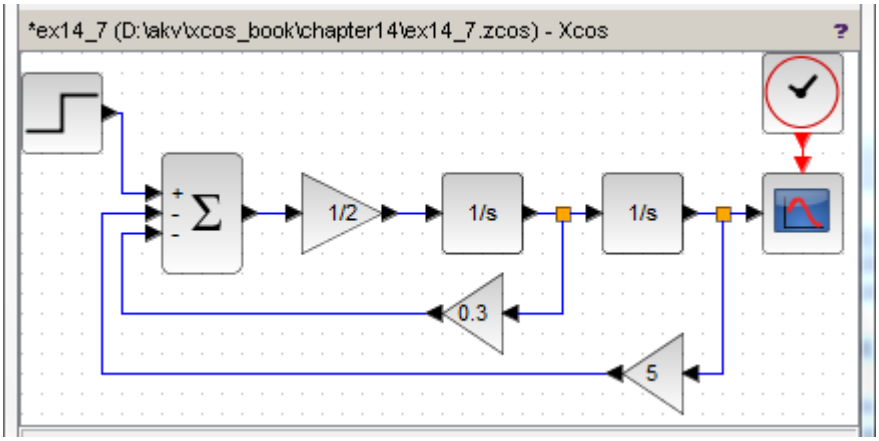
$$m\ddot{x} + b\dot{x} + kx = f(t) \qquad 14.7(a)$$

$$f(t) = \begin{cases} 0, & t < 1 \\ 2, & t \geq 1 \end{cases} \qquad 14.7(b)$$

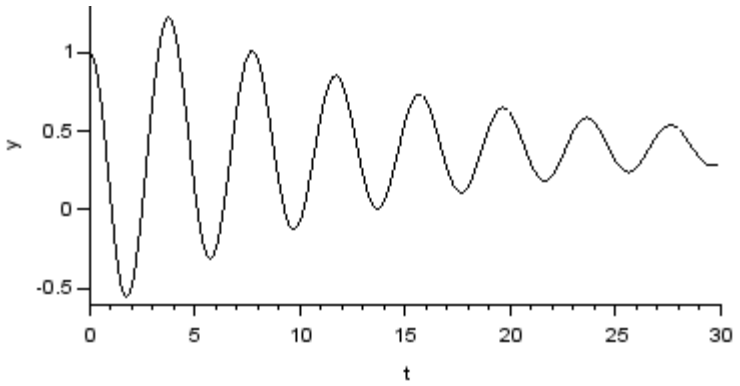
14.8 Projectile motion with drag

Projectile motion of an object under gravity and air drag can be defined by a set of equations given in equation 14.8, where (x, y) is the location of the projectile at any moment of time, g is acceleration due to gravity, k is drag coefficient and (v_y, v_x) are velocity components in x (horizontal) and y (vertical) directions, respectively. The drag force is always in opposite direction to the velocity and here assumed to be proportional to the square of the velocity. Example 14.8 shows and an Xcos model to simulate equation 14.8, with initial values y=150, x=0, v_x=25, v_y=25, k=0.1, g=9.8 (all values are in SI units). The model plots (x, t), (y, t), (v_x, t), (v_y, t) and (x, y) plots.

Example 14.7:Forced and damped oscillation simulation



Output

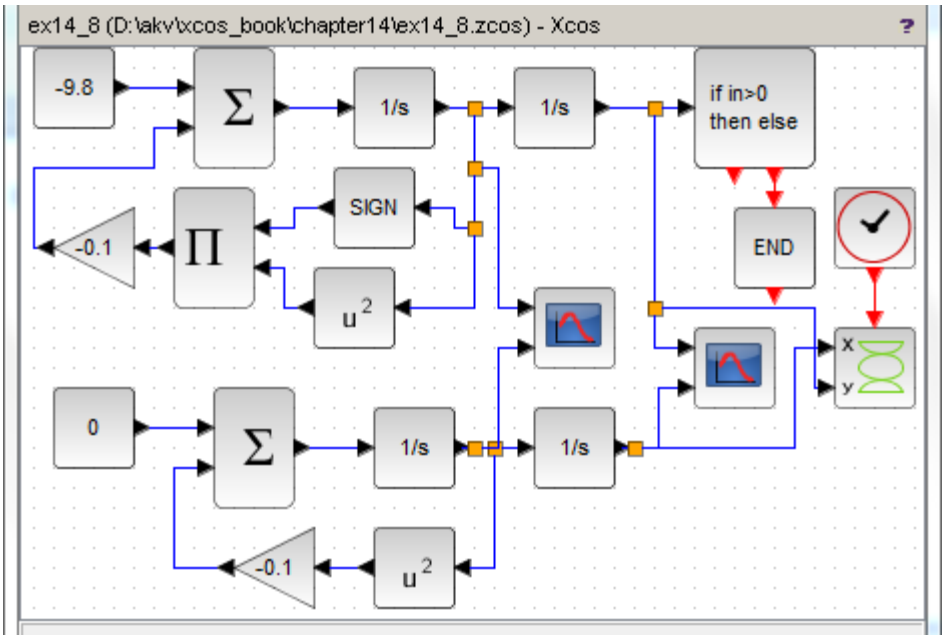


$$\ddot{y} = -g \quad - kv_y^2$$

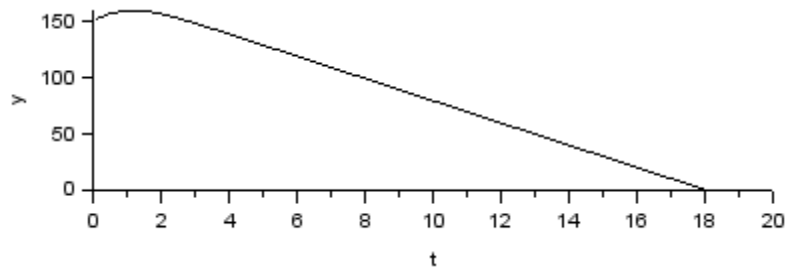
$$\ddot{x} = -kv_x^2$$

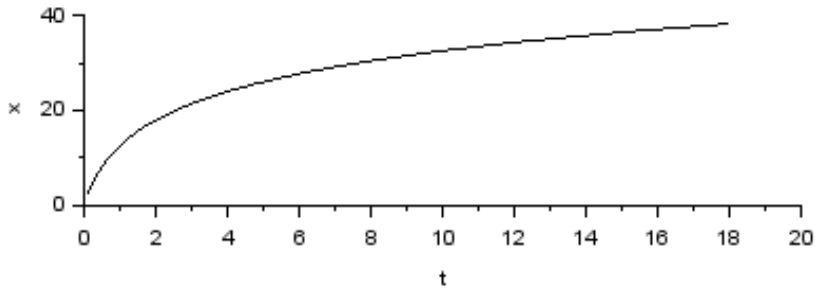
14.8

Example 14.8: Projectile motion simulation

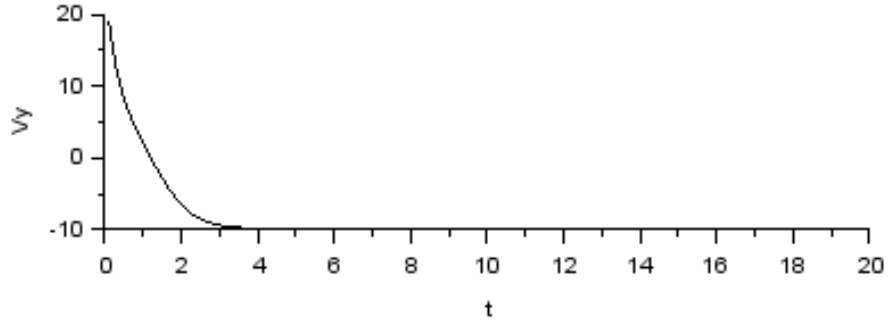


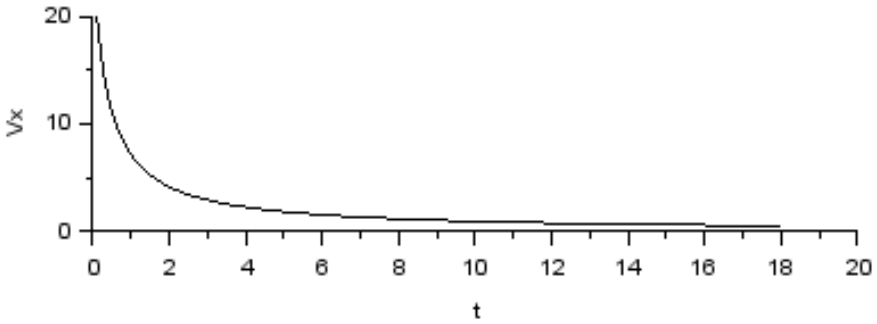
Outputs ((*x*, *t*) and (*y*, *t*) plots)



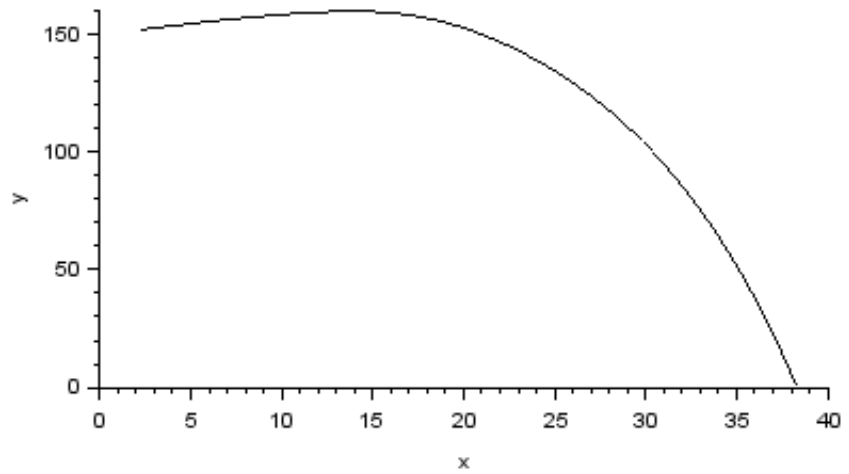


(v_x, t) and (v_y, t) plots


Vy
20
10
0
-10
0
2
4
6
8
10
12
14
16
18
20
t


Vx
20
10
0
0
2
4
6
8
10
12
14
16
18
20
t

(x, y) plot

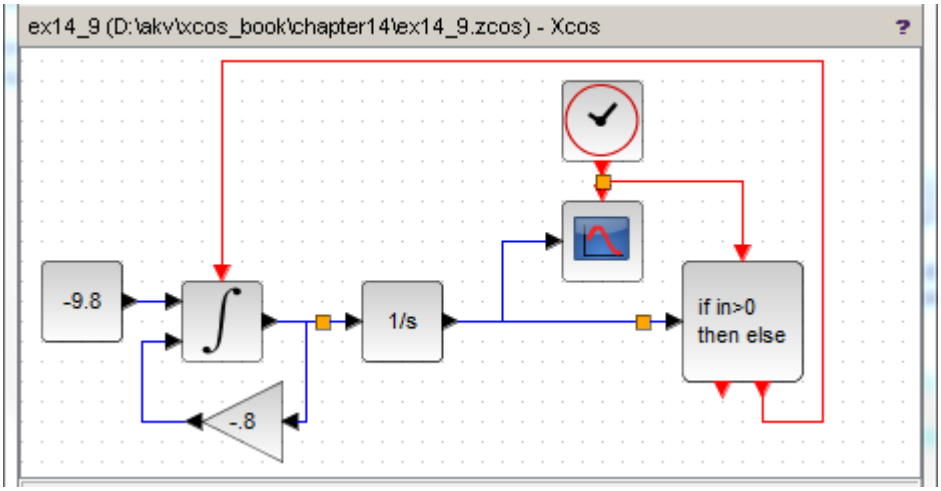

y
150
100
50
0
0
5
10
15
20
25
30
35
40
x

14.9 Bouncing ball simulation

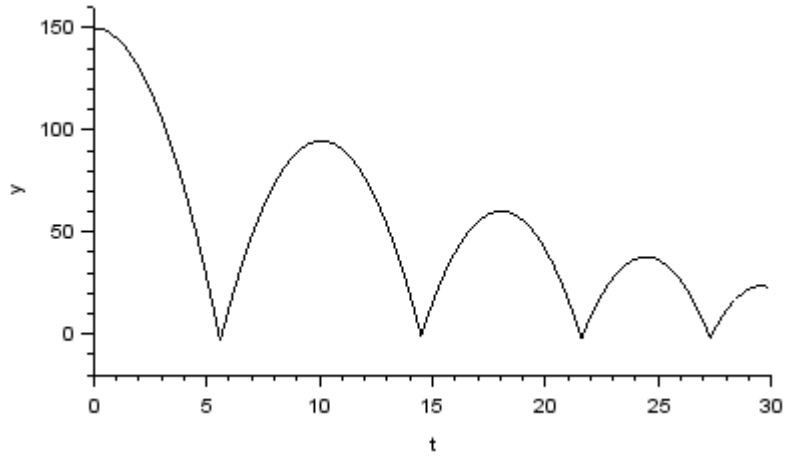
The governing differential equation to simulate a bouncing ball, neglecting the air drag, is given in equation 14.9, where g is acceleration due to gravity, y is the height of the ball from the ground and k is the restitution coefficient. The first part of the equation governs the projectile of the object and the second part changes the velocity direction when the object hits the ground. Example 14.9 is an Xcos model of the bouncing ball simulation. In this model, an integrator with resetting the state of the integrator is used. The first integrator is reset to reverse the velocity (along with the reduction in the velocity due to collision) of the ball when it hits the ground. The initial state of the first integrator is set to zero, the second integrator is set to 150 and k=0.8 (all values are in the SI units).

$$\begin{cases} \ddot{y} = -g & for\ y > 0 \\ \ddot{y} = -k\dot{y} & for\ y = 0 \end{cases} \qquad 14.9$$

Example 14.9:Bouncing ball simulation



Output



14.10 Non-linear pendulum motion simulation

A simple pendulum with a point mass *m* attached with a massless string of length *L* is shown in figure 14.3.

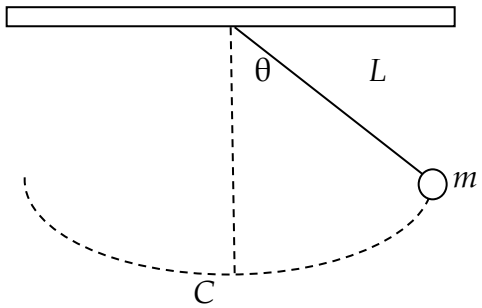


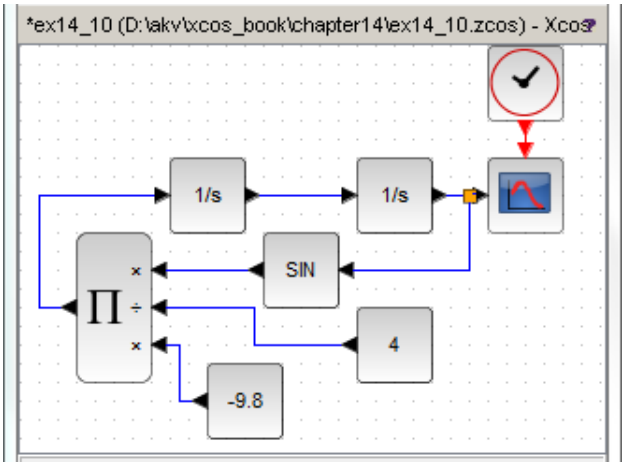
Figure 14.3: A simple pendulum

When the mass at an initial angle θ_0 is released, the pendulum oscillates around its center point C. The pendulum motion is governed by the equation given in equation 14.10(a), where *g* is acceleration due to gravity and *x* is the arc length from the center C. For θ in radian, $x=\theta L$. With this relation, equation 14.10(a) can be written as given in equation 14.10(b). Example 14.10 shows an Xcos model of a pendulum with L= 4 m and $\theta_0=\pi/4$.

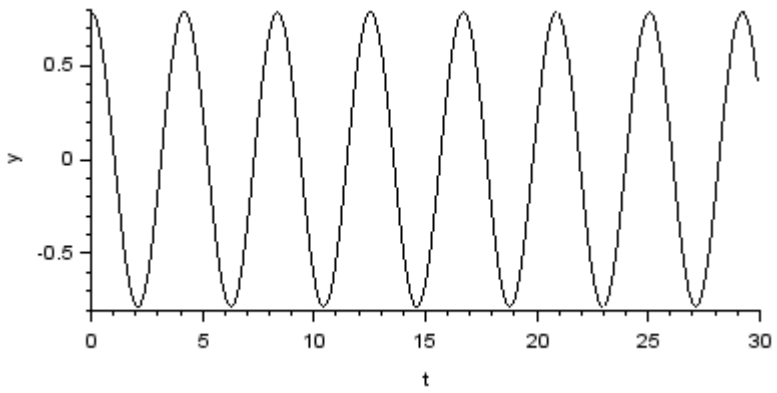
$$m\ddot{x} = -mg\sin(\theta) \qquad 14.10(a)$$

$$\ddot{\theta} = -\frac{g}{L}\sin(\theta) \qquad 14.10(b)$$

Example 14.10:Pendulum simulation



Output



14.11 Pollution spreading simulation

Figure 14.4 shows three ponds (A, B and C) connected by streams. The first pond, A, has a pollution source active for some time, with pollution rate *f(t)*. The pollutants spread to other ponds via the connecting streams. It is desired to estimate the amount of pollutants in each pond as time progresses.

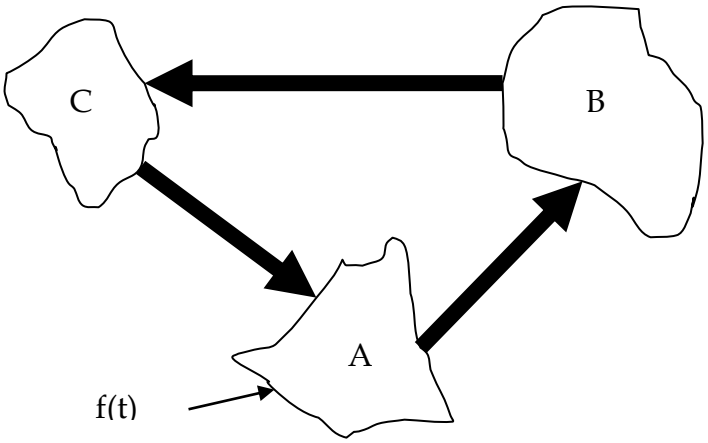


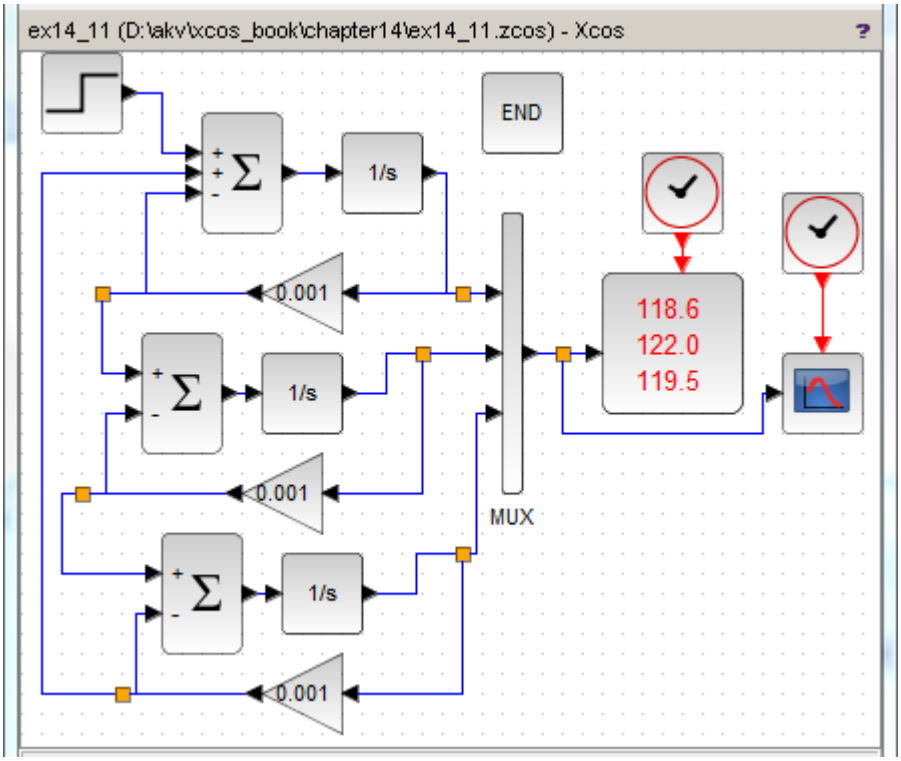
Figure 14.4: Pond system

The pollutant flux for a pond is the flow rate times the pollutant concentration in the pond. The rate of change of the pollutant in a pond is equal to influx from the upstream pond minus outflux from pond plus addition rate of new pollutants. Based on the above, the pollution spread for the pond system shown in figure 4 can be modeled by the system of differential equations given in equation 14.11, where f_1, f_2, f_3 are out flow rates, V_1, V_2, V_3 are volumes and x_1, x_2, x_3 are amount of pollutants at any time in ponds A, B and C, respectively., *f(t)* is the pollutant addition rate for pond A.

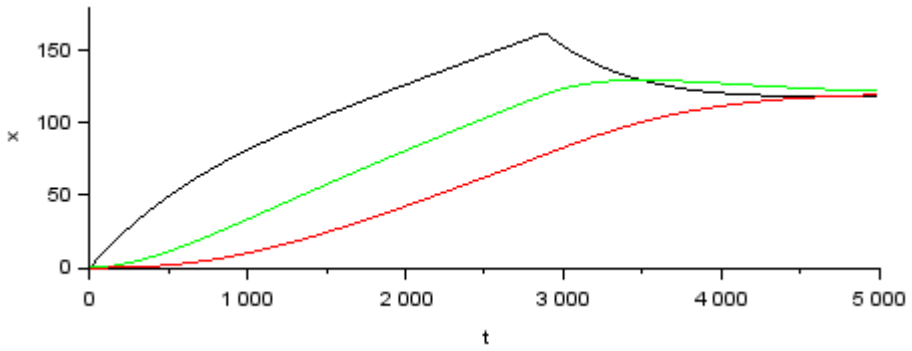
$$\begin{aligned}\dot{x}_1 &= \frac{f_3}{V_3}x_3 - \frac{f_1}{V_1}x_1 + f(t)\\ \dot{x}_2 &= \frac{f_1}{V_1}x_1 - \frac{f_2}{V_2}x_2\\ \dot{x}_3 &= \frac{f_2}{V_2}x_3 - \frac{f_3}{V_3}x_3\end{aligned} \qquad 14.11$$

Example 14.11 shows an Xcos model of a specific pond system with $f_i/V_i=0.001$ for each pond and $f(t)=0.125$ kg/min for the first 2880 minutes, thereafter $f(t)=0$.

Example 14.11:Pollution spread simulation



Output



14.12 Spring coupled masses simulation

Figure 14.5 shows a mechanical system consisting of two masses (m_1 and m_2) coupled with three massless springs (with spring constant k_1, k_2 and k_3) between two immovable walls. The masses are free to slide over a frictionless horizontal surface. The equilibrium positions of the masses are represented with x_1=0 and x_2=0 when there is no compression or extension in any spring. The system of equations of motion of the two masses at any instantaneous displacements (x_1 and x_2) of the masses m_1 and m_2 is given in equation 14.12. Example 14.12 shows an Xcos model of equation 14.12 with k_1=k_2=k_3=1, m_1=m_2=1 and the initial value of x_1=5 and x_2=0.

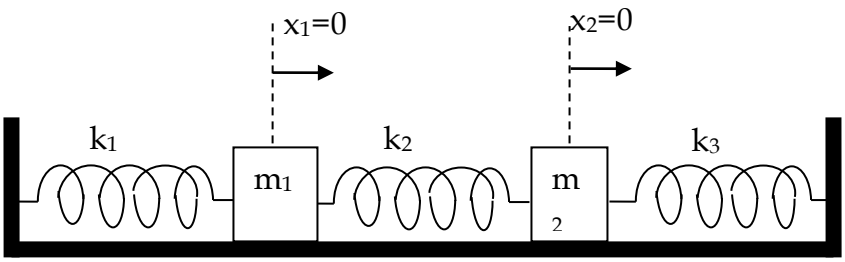
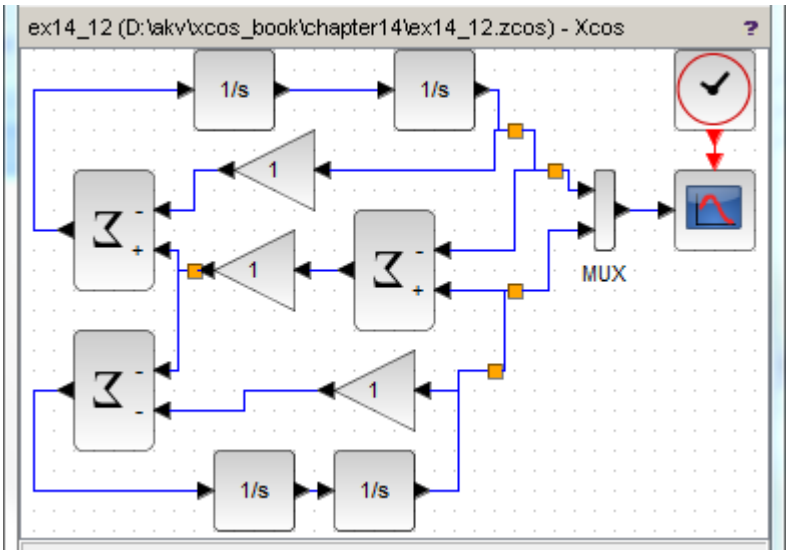


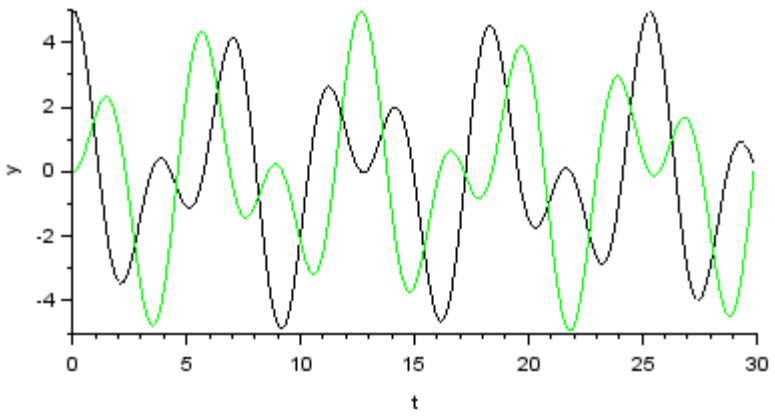
Figure 14.5: Spring coupled mass system

$$\ddot{x_1} = -\frac{k_1}{m_1}x_1 + \frac{k_2}{m_1}(x_2 - x_1)$$
$$\ddot{x_2} = -\frac{k_3}{m_2}x_2 - \frac{k_2}{m_2}(x_2 - x_1) \qquad 14.12$$

Example 14.12:Spring coupled mass system simulation



Output



14.13 RLC circuit simulation

Figure 14.6 shows a typical series RLC circuit, with a constant driving electro-motive force, E. On closing the switch, the current equation for the circuit is given in equation 14.13(a). Differentiating equation 14.13(a), the governing differential equation for the circuit is given in equation 14.13(b). Example 14.13 shows an Xcos model of the RLC circuit shown in figure 14.6 using equation 14.13(b). The outputs are shown for various combinations of RLC values with i_0=2 and $(di/dt)_{t=0}$ =4.

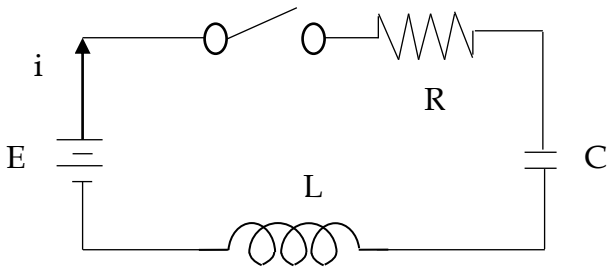
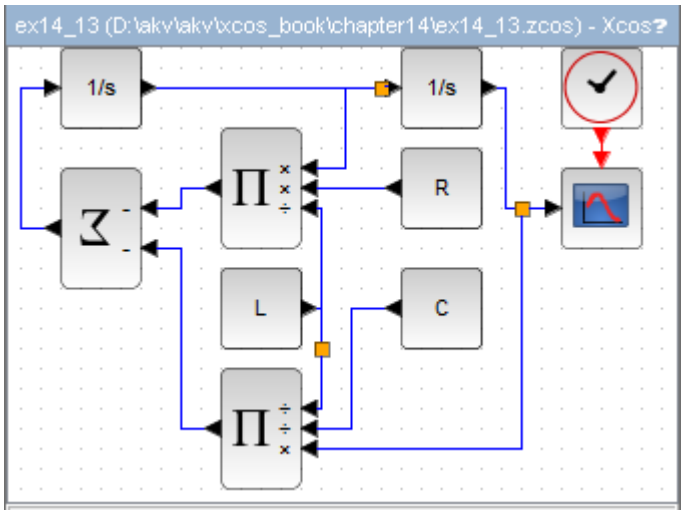


Figure 14.6: A typical RLC circuits

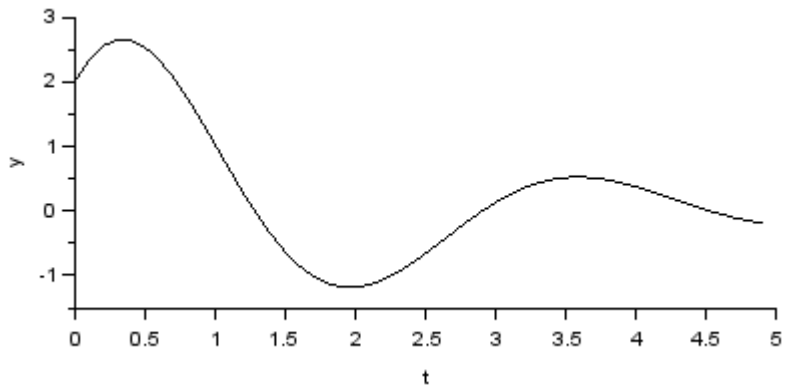
$$L\frac{di}{dt} + Ri + \frac{1}{C}\int i\,dt = E \qquad 14.13(a)$$

$$\frac{d^2i}{dt^2} = -\frac{R}{L}\frac{di}{dt} - \frac{1}{LC}i \qquad 14.13(b)$$

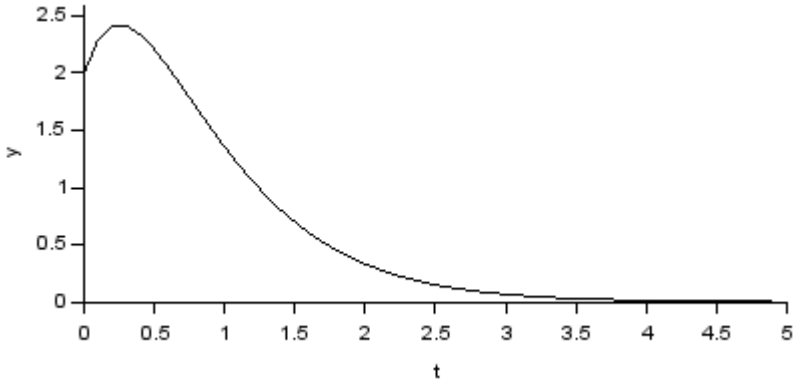
Example 14.13:RLC circuit simulation



Output: R=1,L=1,C=0.25 (Under damped)



Output: R=4,L=1,C=0.25 (Critically damped)



View publication stats