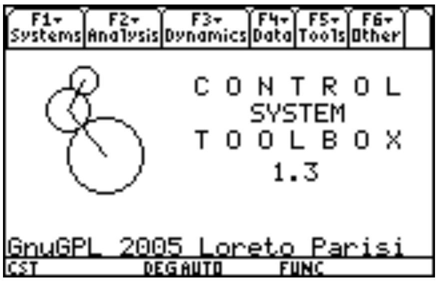
Control SystemToolbox

for TI-89



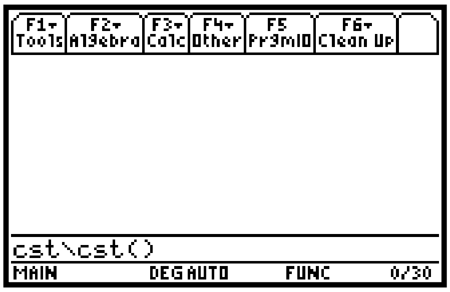
release 1.3

The CST User Guide

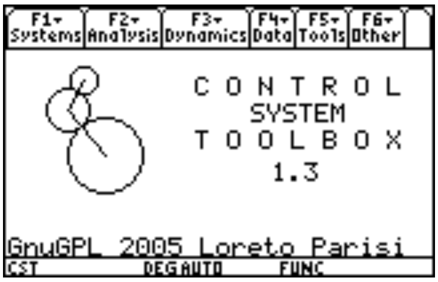
Fifth Edtion October 2005

About Control System Toolbox *for TI-89*

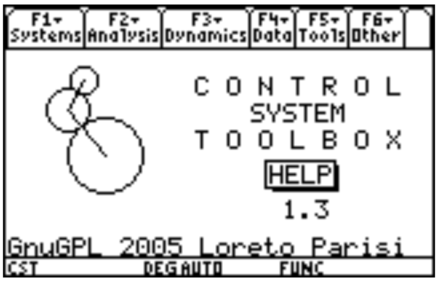
Control System Toolbox (CST) for TI-89 is a suite of specialized functions and programs for Systems Control and Tuning created by **Loreto Parisi** from June 2002 for the TI-89 personal calculator.



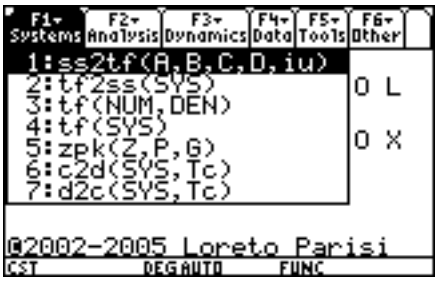
After installing (see *How To Install* on page 10), to run the program on your calculator, types *CST/cst()* from folder *MAIN* and wait few seconds.



This is the main screen of *cst()*. You can see several menus, in which you can find all the function you need to work with state space, linear and non – linear models, etc., grouped in a logical order.



If you have trouble to use any function, you can choose *help()* from *Other* menu (*F6*), to run the useful on- line help tool, which can be used instead of this reference guide to obtain instant help. Note that this is a standalone program so you can recall it typing *CST/help()* from *HOME.*



To recall menus you can use *Function-keys* instead of arrow keys. Then to choose a function, simply select it typing the number or the letter on the left, or use the arrow keys to navigate in the menu.

Disclaimer

The Open Source Philosophy

If you have an apple and I have an apple and we exchange apples then you and I will still each have one apple.
But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas.

This is our way of thinkin'...

How To Get Help

- *Consult the new CST Guides:*
  The **CST Start Guide** will guide you through the installation of CST.
  This guide is bundled with CST r1.3.

  The **CST Reference Guide** will guide you through all CST functions.
  Download this guide separately from CST Home.

  The **CST User Guide** will guide you using CST with complete examples.
  Download this guide separately from CST Home.

  Get the new CST Guides here:
  http://www.webalice.it/loretoparisi/downloads.html

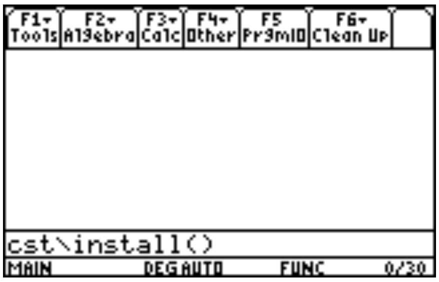
- *Get In Touch:*
  To get more help about CST *for TI-89* and/or to send comments, questions and suggestions, you can contact me at

  Loreto Parisi
  Email: loreto_parisi@yahoo.it
  CST Home: http://www.webalice.it/loretoparisi/

- *Send Feedback:*
  http://www.webalice.it/loretoparisi/feedback.html

How To Install

Use your linking software to send the program *CSTxxx.89G* on the calculator. All the files are automatically placed in the folder CST. Once installation has occurred, do not move, delete, or rename any of the functions and programs or pictures in the folder CST. All files included in folder CST are necessary to *cst()* to work right. For a list of files included in this folder, see *Contents*. For further notice please see **Note**.

Install CST	
	Once sent CST to your device, please run *cst\install()* from HOME.
	CST Install Tool starts. Please confirm pressing Enter now.
	The first step is to executing once all functions to improve performances. Please press Enter.
	Please wait while executing once all functions. This will take few minutes. The progress bar indicates the Install Tool is working. *Please don't break execution during this time.*

	The second step is to archive all executed functions. Please press Enter.
	Please wait while archiving all functions This will take few minutes. The progress bar indicates the Install Tool is working. *Please don't break execution during this time.*
	Now it's time to execute once and archive the programs. Press Enter will run the program. Then simply quit. Choose Enter to run *bodex()*, then press F1→ 1 to exit.
	Choose Enter to run *gstep()*, then press F7 to exit.
	Choose Enter to run *feedback()*, then press F4→ 1 to exit.

<table>
<tr><td></td><td>Choose Enter to run nyquist(), then press F4 to exit.</td></tr>
<tr><td></td><td>Choose Enter to run rlocus(), then press F5 to exit.</td></tr>
<tr><td></td><td>This will execute and install the Error Management System, error().</td></tr>
<tr><td></td><td>The Error Management System was installed.</td></tr>
<tr><td></td><td>Choose Enter to run cst(), then press F6 → 7 to exit.</td></tr>
</table>

	Choose Enter to run *cst()*, then press F6 → 7 to exit.
	Congratulations! Control System Toolbox *for TI-89* installation successed. To run *cst()* just now, choose Yes and press Enter. Enjoy the journey!

Note.

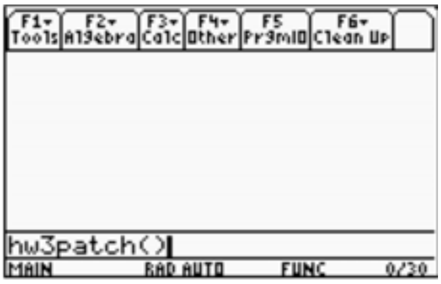
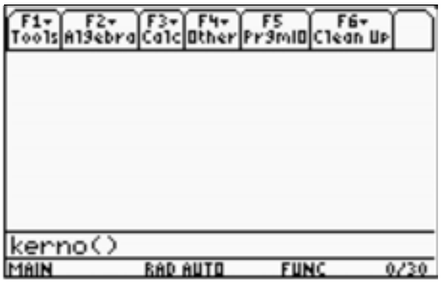
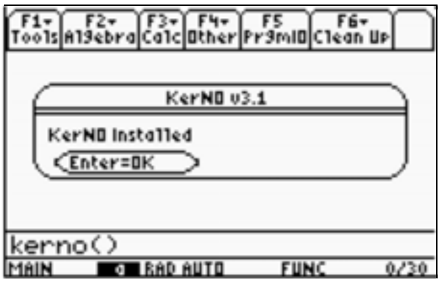
From release 1.3, CST needs the tool LZT to perform symbolic calculations (i.e. Laplace and Zeta transforms). To install LZT please follow instructions we provide in the following section **Install LZT**. We also raccomend to read the LZT readme file.

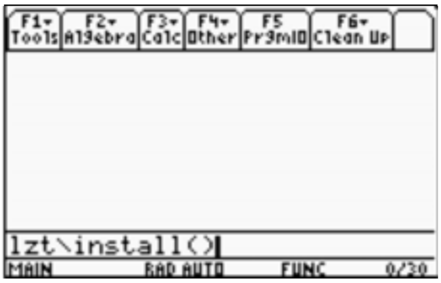
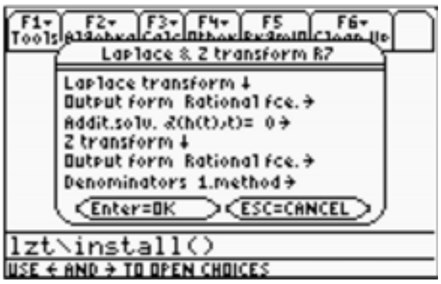
LZT r7
Author: Jiri Bazant
Email: georger@razdva.cz
Home: http://www.razdva.cz/georger/

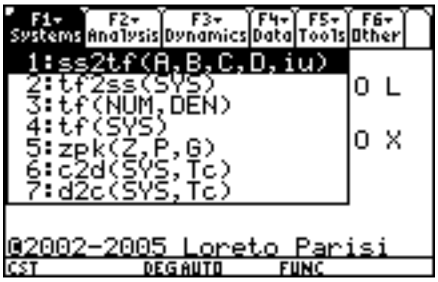
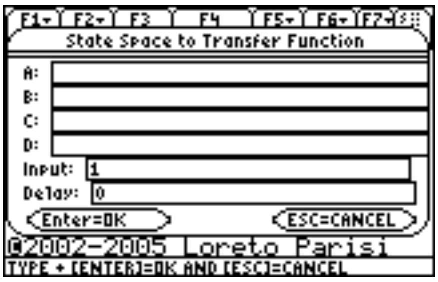
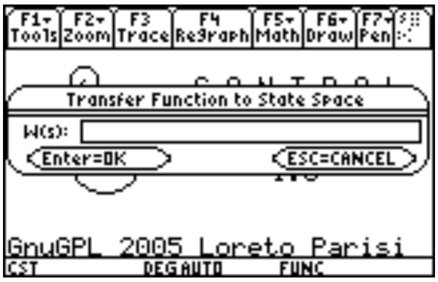
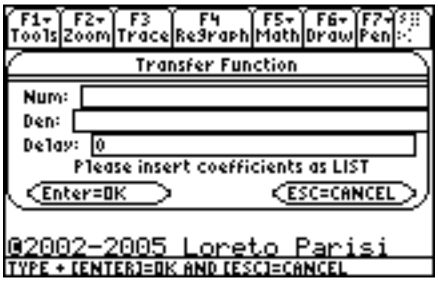
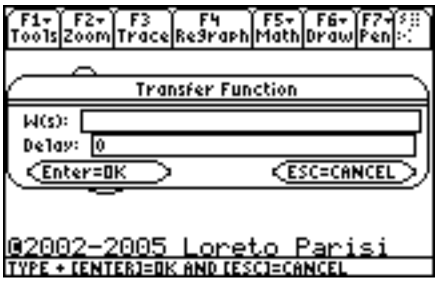
This powerful tool needs any kernel like DoorsOS, UniOS or KerNO. We provide KenNO r3.1 from CST r1.3 as its convenient installation. To install KerNO please follow instructions we provide in the following section **Install KerNO**. We also raccomend to read the KerNo readme file.

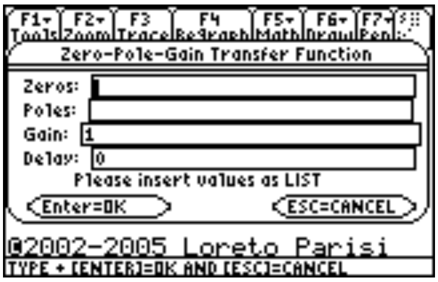
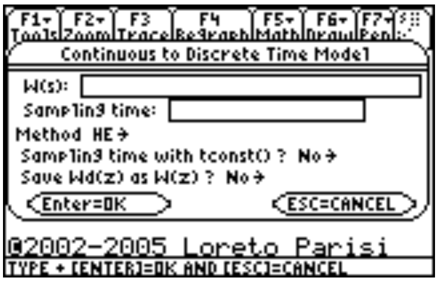
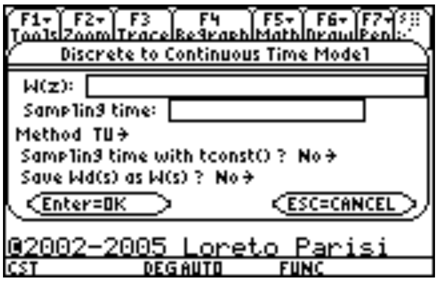
KerNO r3.1
Author: Greg Dietsche
E-Mail: mailto:calc@gregd.org
Home: http://calc.gregd.org/

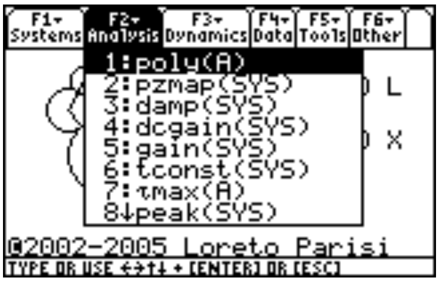
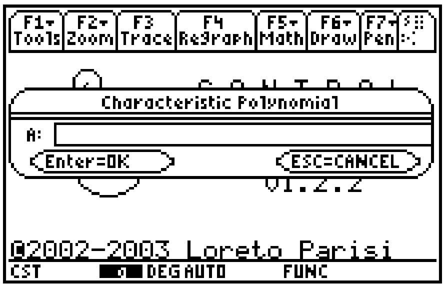
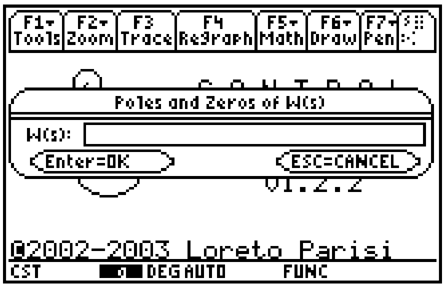
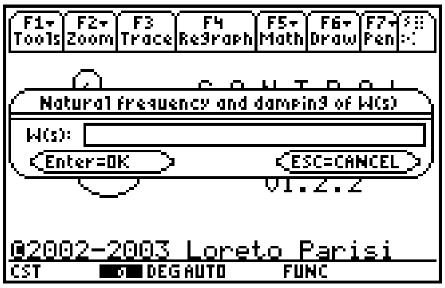
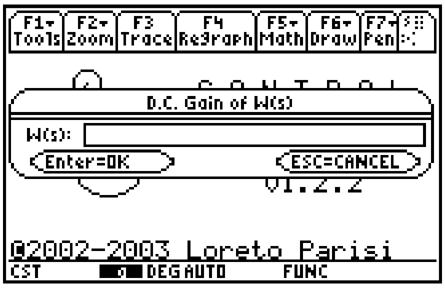
Install KerNO

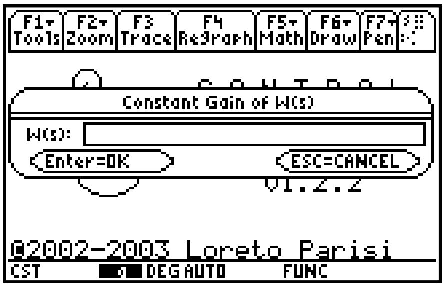
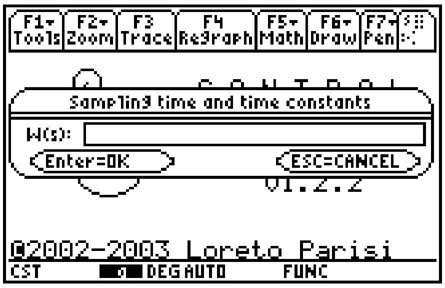
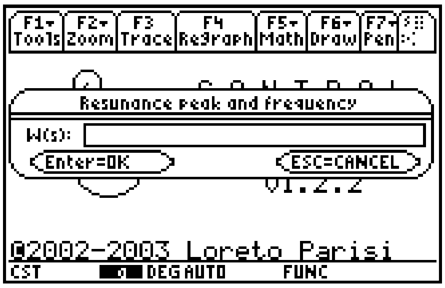
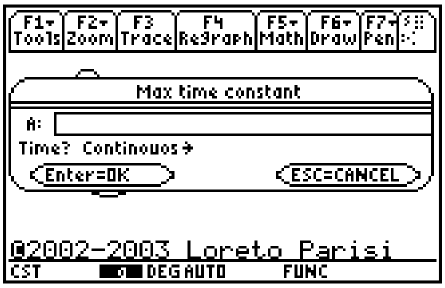
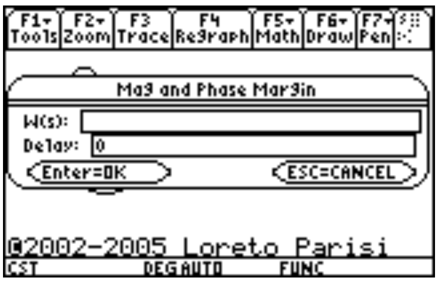
	First, we have to install the hw3patch(), for Hardware Version up to 3. Transfer the patch to TI-89, then run it from *main*.
	HW3Patch 1.00 Author: Kevin Kofler Copyright (C) 2004 Kevin Kofler. All rights reserved. Home: http://tigcc.ticalc.org.
	Now, we can install the kernel. Transfer KerNo to TI-89, then simply run it from *main*.
	KerNO is now installed in TI-89 memory.
	KerNO r3.1 Author: Greg Dietsche E-Mail: mailto:calc@gregd.org Home: http://calc.gregd.org/

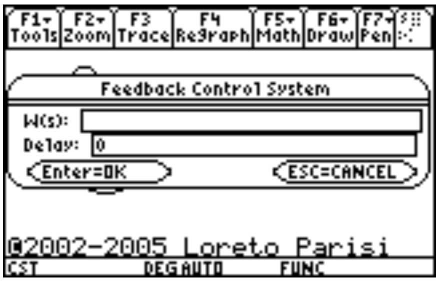
<table>
<tr><th colspan="2">Install LZT</th></tr>
<tr><td></td><td>After installing a kernel, we can install LZT release 7 (current).</td></tr>
<tr><td>
</td><td>Send lztR7.89g to TI-89 and run install from lzt folder.</td></tr>
<tr><td>
</td><td>Choose output options for Laplace and Zeta transforms:
We will use 0 as derivative of the Heavside’s Step and rational fce as output forms.</td></tr>
<tr><td>
</td><td>Now we will choose Archive to improve performances of lzt and to save space in RAM memory, archiving lzt in Flash ROM memory.</td></tr>
<tr><td></td><td>LZT r7
Author: Jiri Bazant
Email: georger@razdva.cz
Home: http://www.razdva.cz/georger/</td></tr>
</table>

Systems	
	The ***Systems*** menu (F1) contains all the functions to build the model, using state-space or transfer function and to perform conversions from one representation to another, even from continuous time to discrete time model.
	***ss2tf(A,B,C,D,iu)*** Gives transfer function W(s)=C(sI-A)$^{-1}$B+D from state-space $\dot{x} = Ax + Bu$, $y = Cx + Du$, relating to input iu (it works on MIMO systems, but only one input at time). Delay τ is the Time Delay $e^{-\tau s}$.
	***tf2ss(SYS)*** Convert transfer function SYS in the state-space representation $\dot{x} = Ax + Bu$, $y = Cx + Du$, using the observability canonical form. Delay τ is the Time Delay $e^{-\tau s}$.
	***tf(NUM,DEN)*** Calculates transfer function, where NUM and DEN are LIST of coefficients of numerator's and denominator's polynomial: NUM={b_0,b_1,…,b_n}, DEN={a_0,a_1,…a_n}, so W(s)=$\dfrac{b_0 s^n + b_1 s^{n-1} + ... + b_n}{a_0 s^n + a_1 s^{n-1} + ... + a_n}$. Delay τ is the Time Delay $e^{-\tau s}$.
	***tf(SYS)*** Calculates transfer function from a rational expression in s Delay τ is the Time Delay $e^{-\tau s}$.

	***zpk(Z,P,G)*** Calculates transfer function W(s) in the zeros-poles-gain representation, where Z, P are LIST of zeros of numerator and denominator (poles), while G is NUM and represents constant gain K. Control's Toolbox v.1.16 author: Francesco Orabona E-mail: bremen79@infinito.it Home:http://web.genie.it/utenti/b/bremen79/
	***c2d(SYS,T$_c$)*** Converts continuous time model SYS to the discrete time model, using sample time Tc and different methods: HE (Linear Hold Equivalence), TU (Bilinear Tustin), BE (Bilinear Backward Eulero), FE (Bilinear Forward Eulero). Can use function tconst(SYS) to determinate sample time T_c. Use function sampler(A,B,T_c) to use ZOH method. Can save the resulting discrete time model Wd(z) as current discrete transfer function W(z).
	***d2c(SYS,T$_c$)*** Converts discrete time model SYS (in z) to the continuous time model, using sample time T_c and different methods: HE (Linear Hold Equivalence), TU (Bilinear Tustin), BE (Bilinear Backward Eulero), FE (Bilinear Forward Eulero).Can use function tconst(SYS) to determinate sample time T_c. Can save the resulting continuous time model Wd(s) as current continuous transfer function W(s).

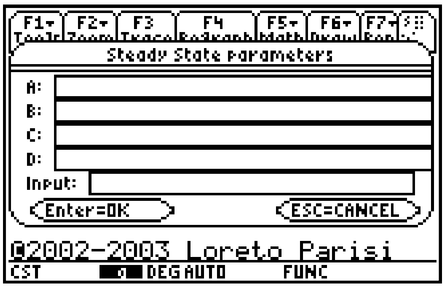
Analysis	
	The ***Analysis*** menu (F2) contains all the tools to analyze the model you have created with Systems' tools. You can also analyze different models, using different SYS at time. This will not change current transfer function.
	***poly(A)*** This function calculates characteristic polynomial of matrix A, as p(s)=|sI-A|, where |•| is determinat of a matrix.
	***pzmap(SYS)*** This function calculates poles and zeros of given transfer function SYS, where poles are zeros of denominator of SYS.
	***damp(SYS)*** Calculate natural frequencies ω_{nh} and damping factors ζ_h for transfer function SYS, where $\omega_{nh}=\sqrt{\alpha_h^2+\omega_h^2}$ and $\zeta_h=\frac{-\alpha_h}{\omega_{nh}}$ for eigenvalue $\lambda_h=\alpha_h+j\omega_h$.
	***dcgain(SYS)*** Calculates d.c. gain G for transfer function SYS, as G=$\lim_{s\to 0} W(s)$.

	***gain(SYS)*** Calculates constant gain K for transfer function SYS, as K=$\lim_{s\to 0} s^{n_0-m_0}W(s)$, where n_0 and m_0 are multiplicity of zero roots for denominator and numerator.
	***tconst(SYS)*** Calculates sample time T_c and time constants τ_i, τ_h, and T_h, where $\tau_i = -\frac{1}{\lambda_i}$, $\tau_h = -\frac{1}{\alpha_h}$ and $T_h = \frac{2\pi}{\omega_h}$, while T_c= 0.1min{τ_i , τ_h , T_h}.
	***peak(SYS)*** This function uses a proprietary numerical algorithm to calculate resonance peak $M_p=\max_\omega M(\omega)$, where $M(\omega)=\lvert W(s)\rvert_{s=j\omega}$ and relating frequency f_r, which is $M(2\pi f_r)=M_P$.
	***tmmax(A)*** Calculates maximum time constant for characteristic polynomial of matrix A, in continuous or discrete time, where $\tau_{max} = \frac{1}{\min(-\Re\lambda_i)}$ (continuous time) and $\tau_{max} = \frac{1}{\min(\ln\lvert\lambda_i\rvert)}$ (discrete time).
	***margin(SYS)*** Calculates Mag and Phase Margins. $K_m= 1/\lvert W(i\omega_m)\rvert$ (Mag Margin) ω_m: $\angle W(i\omega_m)= -180°$ $\phi_m= 180°- \lvert\phi_c\rvert$ (Phase Margin) ω_c: $\lvert W(i\omega_c)\rvert=1$ $\phi_c= \angle W(i\omega_c)$ (Critical Phase) $\tau_c= \phi_m/\omega_c*\pi/180°$ (Critical Time Constant)

	***feedback(sys)*** Performs the analysis and design of the closed loop control system of process SYS. Delay τ is the Time Delay $e^{-\tau s}$. Please see *Feedback Control Systems* section.

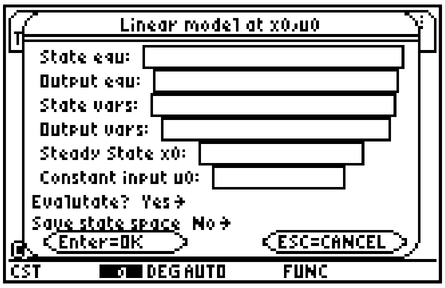
Dynamics

The ***Dynamics*** menu (F3) contains functions concerning dynamics of system for input, output and linearization of a non-linear model, frequency analysis with Bode and Nyquist diagrams and Root Locus yet.



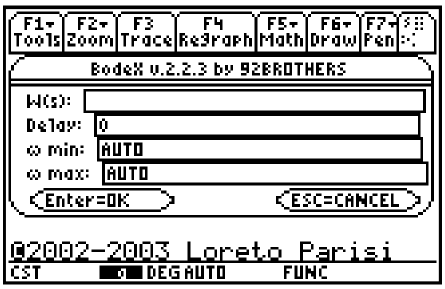
trim(A,B,C,D,u_0)

This function calculates the steady state x_0, relating to input u_0 for state-space $\dot{x} = Ax + Bu$, $y = Cx + Du$.



linmod(f,y,x,u,x_0,u_0)

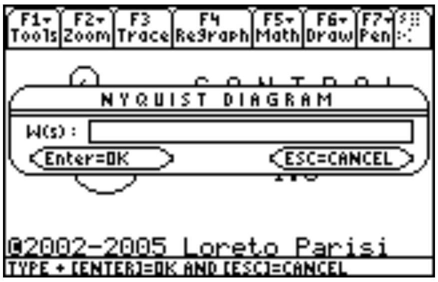
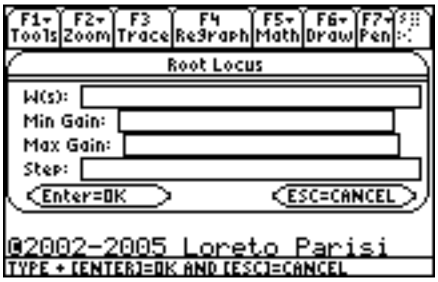
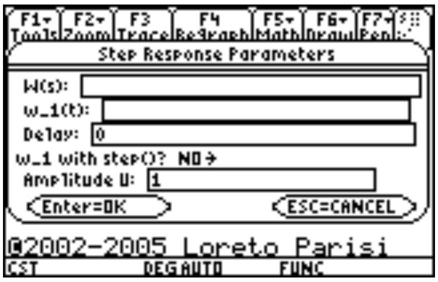
This function calculates linear model for non–linear model assigned in terms of input equations f, such as f={f_1(x,u),…,f_n(x,u)} and output equations y, such as y={y_1(x,u),…y_n(x,u)}, relating to constant input u_0 and steady state x_0. The jacobian matrixes can be evalutated in x_0, u_0 and the state-space can be saved, or can be calculated in a symbolic way, before being evalutated.



bodex(SYS)

This program, made by 92BROTHERS, plots Bode diagrams of phase and magnitude and offers several tools to work with the plottoed diagrams.

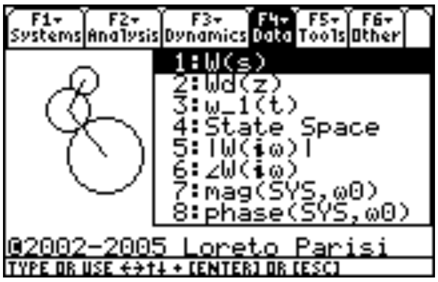
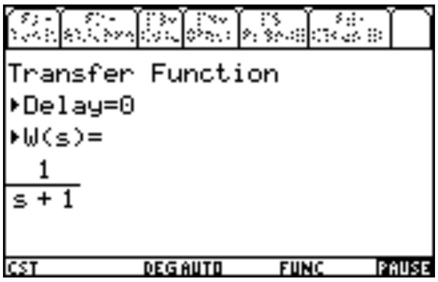
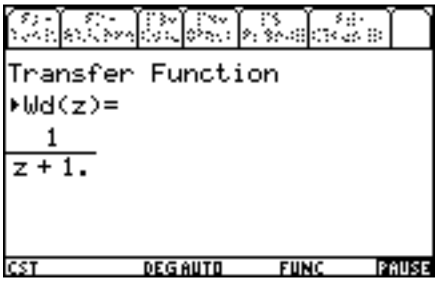
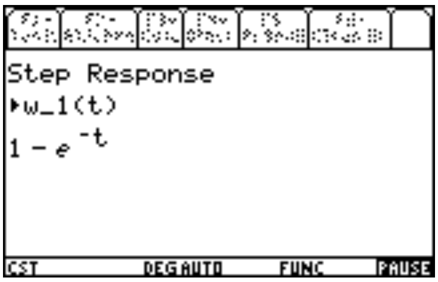
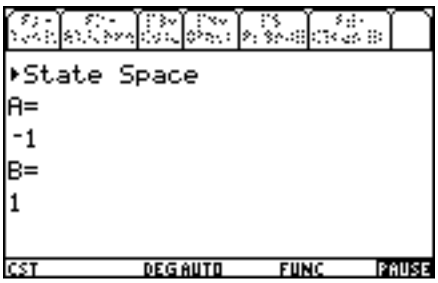
BodeX v.2.2.3
Copyright © 2000 92BROTHERS
Email: 92brothers@infinito.it
Home: http://www.92brothers.net/

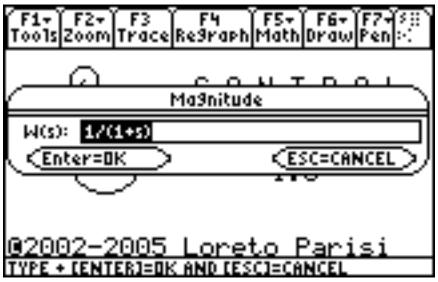
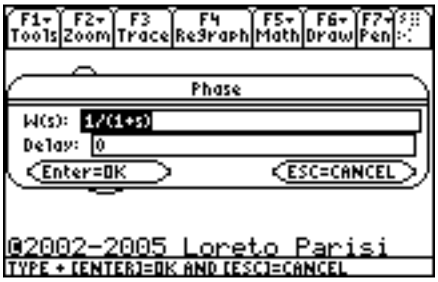
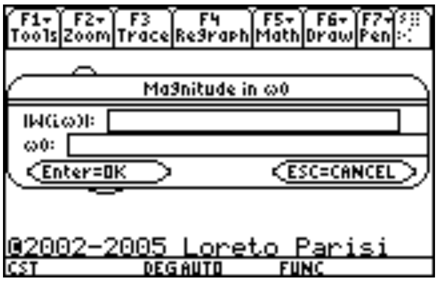
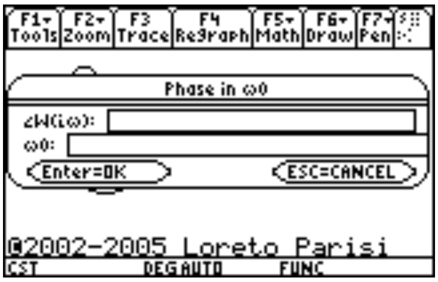
	***nyquist(SYS)*** Plots Nyquist diagram of SYS Control Toolbox v1.16 Author: Francesco Orabona E-mail: bremen79@infinito.it Home:http://web.genie.it/utenti/b/bremen79/
	***rlocus(SYS)*** Plots the Root Locus of SYS Max, min gain are the extremes of the gain list. Porting for CST: Loreto Parisi Control Toolbox v1.16 Author: Francesco Orabona E-mail: bremen79@infinito.it Home:http://web.genie.it/utenti/b/bremen79/
	***step(SYS)*** This tool calculates the step response for SYS, as $U*w_{-1}(t) = L^{-1}(W(s)U/s)$, with amplitude U. Needs the tool LZT to perform symbolic calculation of Laplace direct and inverse transformation. LZT r7 Author: Jiri Bazant Email: georger@razdva.cz Home: http://www.razdva.cz/georger/
	***pstep(SYS)*** Calculates characteristic parameter of step response for transfer function SYS, such as T_e, T_a, T_s, T_p and s. Step response $w_{-1}(t)$ can be specified or calculated with step(SYS). Needs the tool LZT to perform symbolic calculation of Laplace direct and inverse transformation.

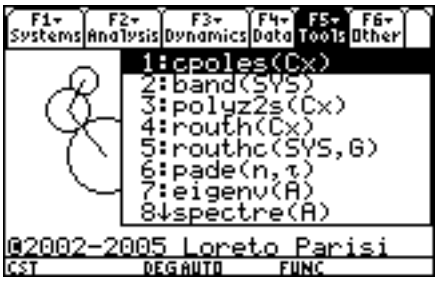
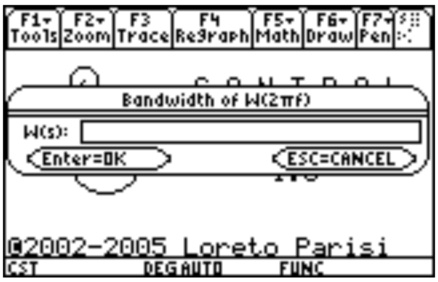
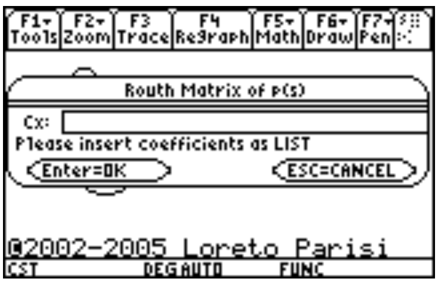
<table>
<tr>
<td>
</td>
<td>gstep(w_1(t))
This tool plots the step response w_1(t) calculated with step(SYS) or specified directly. Can use pstep(SYS) to evalutate w_1(t) around its typical parameters.</td>
</tr>
</table>

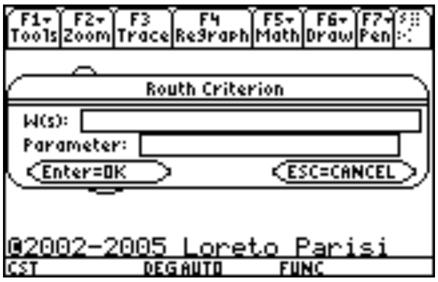
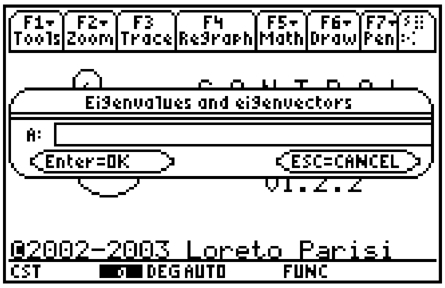
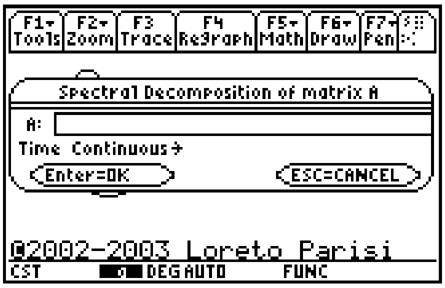
Notes.

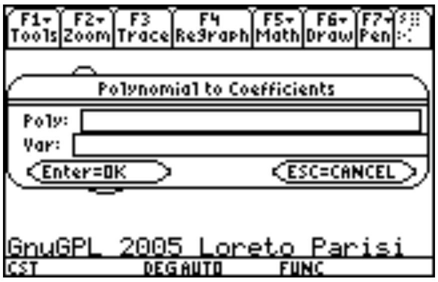
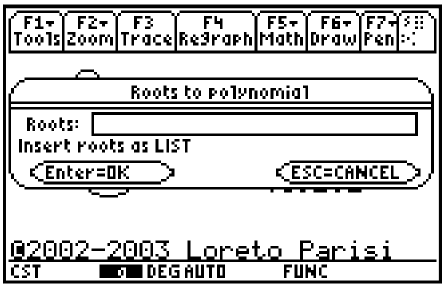
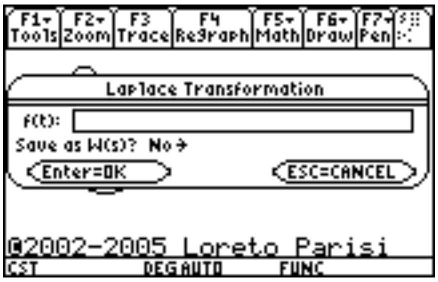
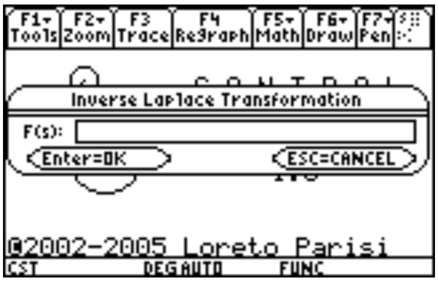
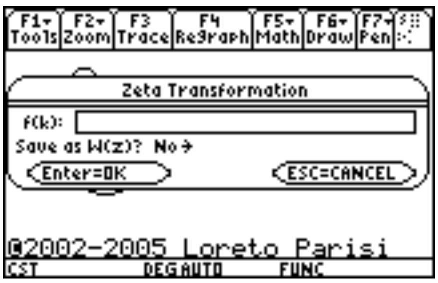
1. About *pstep(SYS)*. Calculates time domain parameters of the step response for transfer function SYS, such as T_e, T_a, T_s, T_p and s, where T_e is the Elongation Time, T_r is the Raise Time, T_s is the Delay Time and s is the elongation.

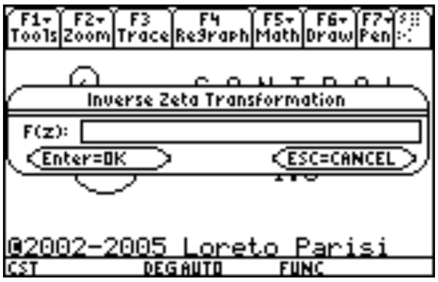
Data	
	The ***Data*** menu (F4) gives access to current transfer function W(•)[1], its discrete model Wd(•), the step response w-1(•), the current state space and magnitude and phase of W(•). [1] According to current Time Domain Settings (see Other menu)
	***W(s) [W(z)]*** Displays the current transfer function. SYS refers to it in all calculations of current session of CST, once you've calculated it with one of the tools of Systems menu.
	***Wd(z) [Wd(s)]*** The discrete transfer function Wd(z) [Wd(s)], obtained by c2d(SYS),Tc) [d2c(SYS,Tc)] using the current transfer function W(s) [W(z)] or by sampler(A,B,Tc) using the current state space.
	***w_1(t) [w_1(k)]*** Shows the current step response obtained with step(SYS).
	***State Space*** It displays the current state space, as defined from one of tools of System menu.

	***| W(ω) |*** Displays magnitude of transfer function W(•) in Laplace domain (for W(s)) and even in domain Z (for W(z)).
	***∠W(ω)*** Displays phase of transfer function W(•) in Laplace domain (for W(s)) and even in domain Z (for W(z)).
	***mag(SYS,ω₀)*** Calculates magnitude of SYS in Laplace domain (for W(s)) and even in domain Z (for W(z)), relating to ω_0.
	***phase(SYS,ω₀)*** Calculates phase of SYS in Laplace domain (for W(s)) and even in domain Z (for W(z)), relating to ω_0.

<table>
<tr><th colspan="2">Tools</th></tr>
<tr><td>
</td>
<td>The Tools menu (F5) offers several useful functions to complete the analysis of the model you’re working and to give more detailed information about it. Moreover presents different tools for discrete systems and finite state systems.</td></tr>
<tr><td>
</td>
<td>cpoles(Cx)
It calculates zeros of polinomyal given as LIST of coefficients, Cx.</td></tr>
<tr><td>
</td>
<td>polyz2s(Cx)
This tool calculates the continuous polynomial q(s), relating to discrete polynomial p(z), assigned in terms of its coefficients LIST, Cx, using the formula
q(s) = $(s-1)^n p(z)\Big|_{z=\frac{s+1}{s-1}}$</td></tr>
<tr><td>
</td>
<td>band(SYS)
This function uses a numerical algorithm and several preexistent formulas to calculate bandwith of system with transfer function SYS. It calculates f_i, f_s, where B=[f_i,f_s], f_r (resonance frequency) and M_p (resonance peak).</td></tr>
<tr><td>
</td>
<td>routh(Cx)
It calculates the Routh matrix for polynomial assigned with its coefficients LIST, Cx.</td></tr>
</table>

	***routhc(Cx)*** Applies Routh Criterion to parametric W(s) to obtain Routh conditions using the specified parameter.
	***pade(n,τ)*** Calculates the delay Padè approximation. Resulting delay transfer function can be saved as the current continuous transfer function W(s).
	***eingev(A)*** It calculates eigenvalues and eigenvectors of matrix A.
	***spectre(A)*** This tool calculates the spectral decomposition of matrix A, even in the continuous (e^{At}) and in the discrete time (A^k), relating to real eigenvalues and complex eigenvalues.[1]
	***sampler(A,B,T$_c$)*** This function performs the discrete time conversion of continuous time model with state-space $\dot{x} = Ax + Bu$ at sample time T_c, using the ZOH (Zero Order Hold) method. It permits to use sample time T_c calculated with function tconst(SYS) for current transfer function SYS. It can save the resulting discrete transfer function Wd(z).

	***poly2cof(expr,var)*** Gives the LIST of coefficients of the polynomial given in expr in the variable var. Control's Toolbox v.1.16 Author: Francesco Orabona E-mail: bremen79@infinito.it Home:http://web.genie.it/utenti/b/bremen79/
	***rts2poly(roots)*** Builds the polynomial with roots assigned as LIST. Author: Chadd L. Easterday Email: easterday@mindspring.com
	***laplace(f(t))*** Performs Laplace Transformation of f(t) Needs the tool LZT to perform symbolic calculation. LZT r7 Author: Jiri Bazant Email: georger@razdva.cz Home: http://www.razdva.cz/georger/
	***ilaplace(F(s))*** Performs Inverse Laplace Transformation of F(s). Needs the tool LZT to perform symbolic calculation. LZT r7 Author: Jiri Bazant Email: georger@razdva.cz Home: http://www.razdva.cz/georger/
	***zeta(f(k))*** Performs Zeta Transformation of f(k) Needs the tool LZT to perform symbolic calculation. LZT r7 Author: Jiri Bazant Email: georger@razdva.cz Home: http://www.razdva.cz/georger/

	***izeta(F(z))*** Performs Inverse Zeta Transformation of F(z). Needs the tool LZT to perform symbolic calculation. LZT r7 Author: Jiri Bazant Email: georger@razdva.cz Home: http://www.razdva.cz/georger/

Notes.

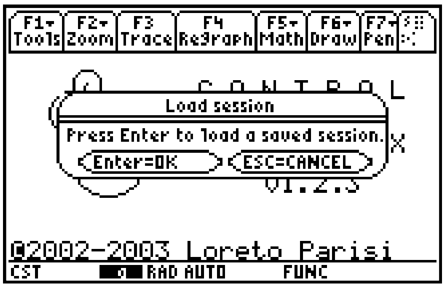
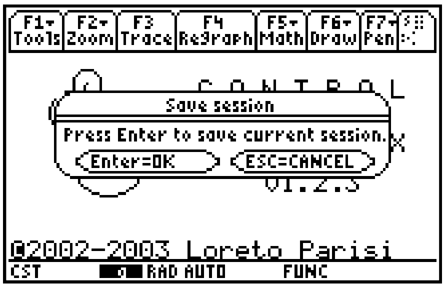
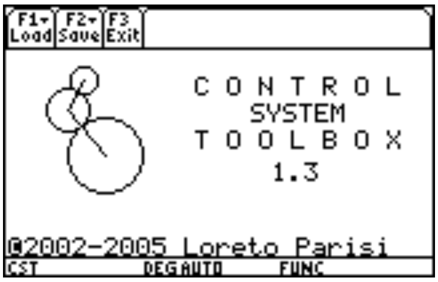
[1] About *spectre(A)*

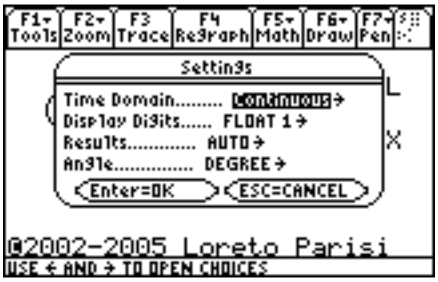
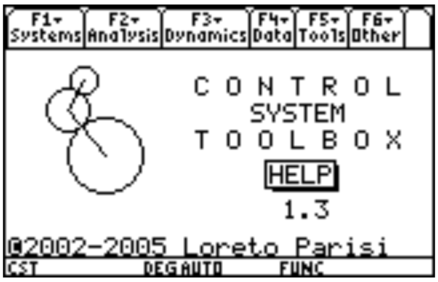
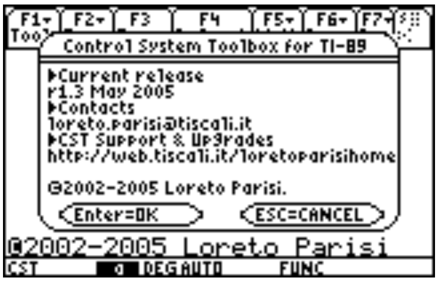
The spectral decomposition of matrix A is

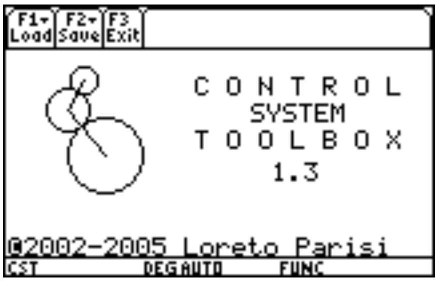
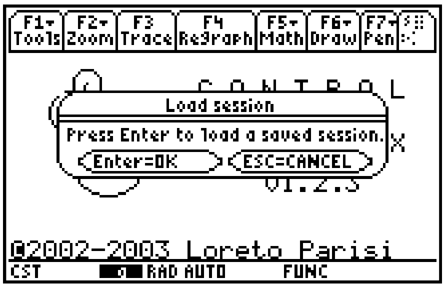
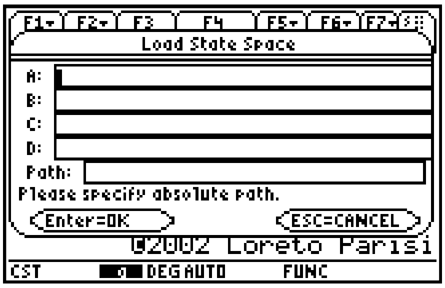
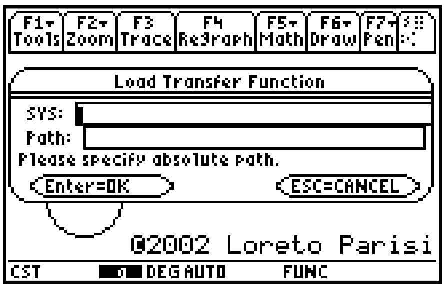
$$e^{At} = \sum_{i=1}^{\mu} u_i e^{\lambda_i t} v_i^T + \sum_{h=1}^{\upsilon} (u_{ha} \quad u_{hb}) e^{\alpha_h t} \begin{pmatrix} \cos\omega_h t & \sin\omega_h t \\ -\sin\omega_h t & \cos\omega_h t \end{pmatrix} \begin{pmatrix} v_{ha}{}^T \\ v_{hb}{}^T \end{pmatrix} \quad \text{(continuous)}$$

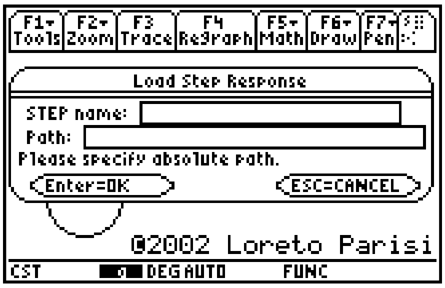
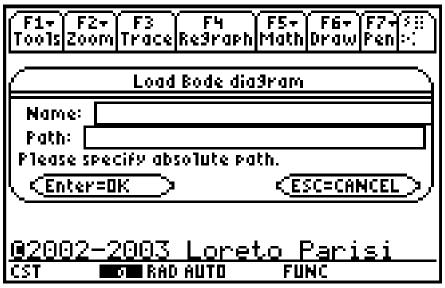
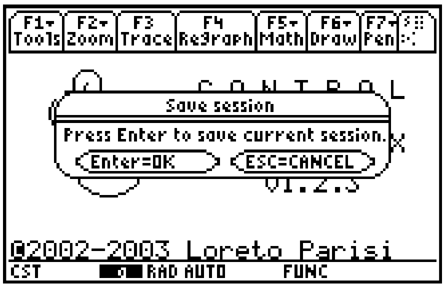
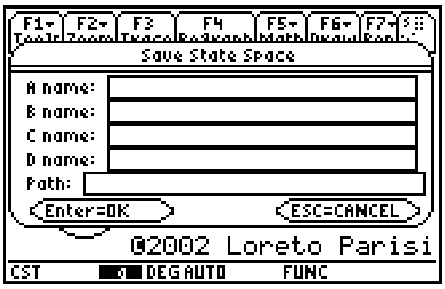
$$A^k = \sum_{i=1}^{\mu} u_i \lambda_i{}^k v_i^T + \sum_{h=1}^{\upsilon} (u_{ha} \quad u_{hb}) \rho_h{}^k \begin{pmatrix} \cos\theta_h k & \sin\theta_h k \\ -\sin\theta_h k & \cos\theta_h k \end{pmatrix} \begin{pmatrix} v_{ha}{}^T \\ v_{hb}{}^T \end{pmatrix} \quad \text{(discrete)}$$

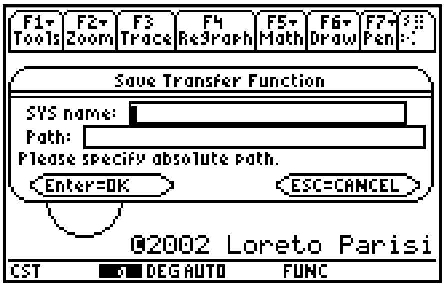
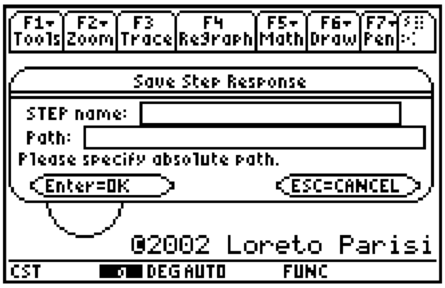
relating to real μ eigenvalues λ_i and 2υ complex eigenvalues $\lambda_h = \alpha_h \pm j\omega_h = \rho_h e^{\pm j\theta_h}$ and the relating eigenvector u_i and $u_h = u_{ha} \pm u_{hb}$.

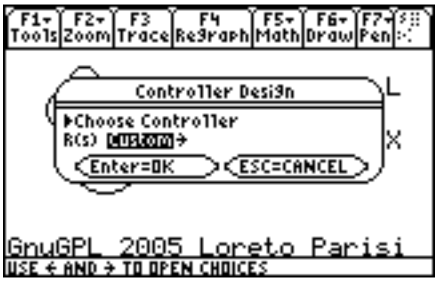
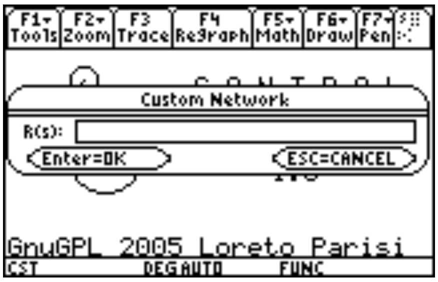
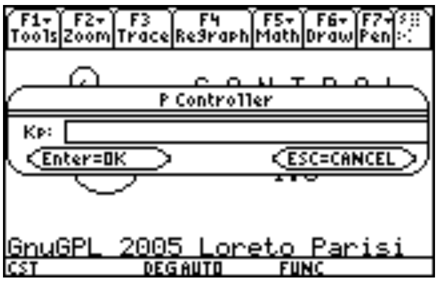
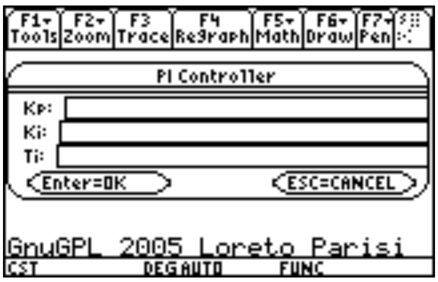
<table>
<tr><th colspan="2">Other</th></tr>
<tr><td>
</td><td>The Other menu (F6) gives tools to manage files, the current working session, the Settings, to access to on-line help tool with help(), some information about CST, and the way to exit CST.</td></tr>
<tr><td>
</td><td>Quick Load
Loads the current working session (i.e. transfer functions W(s) and W(z),State space, w_1(t), Tc, step response parameters,etc.) previously saved. It overwrites all the existing values for the current session. Be careful.</td></tr>
<tr><td>
</td><td>Quick Save
Saves the current working session (i.e. transfer functions W(s) and W(z),State space, w_1(t), Tc, step response parameters,etc.).</td></tr>
<tr><td>
</td><td>File…
The File toolbox gives access to the File & Session Management. Here you can load and save the current working session, the State Space, the Transfer Function, the Step Response and bode diagram obtained with bodex(SYS). There are three menus Load, Save and Exit. Exit menu (F3) brings to the previous toolbox.</td></tr>
</table>

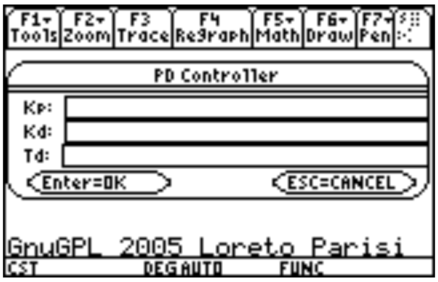
<table>
<tr>
<td>
</td>
<td>Settings
It permits to modify some settings of the calculator, such as the display digits, the angle, the format of results and to switch the current Time Domain: Continuous to work with continuous time model W(s) or Discrete to work with discrete time model W(z) in the same working session.</td>
</tr>
<tr>
<td>
</td>
<td>help()
Starts the help tool. To get help, simply choose a function from one of the menus and you’ll get some information about it.</td>
</tr>
<tr>
<td>
</td>
<td>About
Gives the current version of CST for TI-89, the way to contact the author and to obtain support and upgrades.</td>
</tr>
<tr>
<td>
</td>
<td>Exit
To close Control System Toolbox for TI-89. All previous settings of the calculator will be restored. Prompts for non-saved working session.</td>
</tr>
</table>

File	
	The ***File*** toolbox gives access to the File & Session Management. Here you can load and save the current working session, the State Space, the Transfer Function, the Step Response and bode diagram obtained with bodex(SYS). There are three menus Load, Save and Exit. Exit menu (F3) brings to the previous toolbox.
	***Load*** The Load menu (F1) permits to load the current working session, the State Space, Transfer Function, Step Responde and bode diagrams from the specified path.
	***Load Session*** Loads the current working session (i.e. transfer functions W(s) and W(z),State space, w_1(t), Tc, step response parameters,etc.) previously saved. It overwrites all the existing values for the current session. Be careful.
	***Load State Space*** To load state space matrixes A,B,C,D from specified path. Please use absolute path. For example, if your dynamic matrix A is stored in main as dyn, you have to input dyn in A input field and main as path. All matrixes should be in the same path.
	***Load Transfer Function*** Permits to load Transfer Function from specified path.

<table>
<tr><td>
</td><td>Load Step Response
Permits to load the Step Response from specified path.</td></tr>
<tr><td>
</td><td>Load Bode diagram
This tools permits to load a picture stored in CST folder. It’s aid is in displaying Bode plots, created with bodex() first, and estimating the diagrams in a assigned frequency ω_0.</td></tr>
<tr><td>
</td><td>Save
The Save menu (F2) permits to save the current working session, the State Space, Transfer Function, Step Responde into a specified path.</td></tr>
<tr><td>
</td><td>Save session
Saves the current working session (i.e. transfer functions W(s) and W(z),State space, w_1(t), Tc, step response parameters,etc.).</td></tr>
<tr><td>
</td><td>Save State Space.
Permits to save current State Space into the specified path, using given names.</td></tr>
</table>

	***Save Transfer Function.*** To save current transfer function into the specified path, using given name. The current SYS results from Data menu (F4).
	***Save Step Response.*** To save currrent step response into the specified path, using give name. The current step results from Data menu (F4).

Controller	
	The ***Controller*** menu (F1) is intented to design and tuning the control system.
	The ***Controller Design*** wizard will guide you throught the full design of the network's controller. First step is to choose the *network structure* from the following types: Custom (i.e. user defined), P, PI, PD, PID, Lead, Lag and Lead-Lag networks.
	***Custom Network*** Defines your own custom network's controller R(s).
	***P Controller*** Defines a proportional controller R(s) = Kp.
	***PI Controller*** Defines a PI controller R(s) as you give Kp and Ki or Kp and Ti: $R_{PI}(s)=\frac{K_P s+K_I}{s}=K_P\frac{1+T_I s}{T_I s}$ where $T_I=\frac{K_P}{K_I}$

<table>
<tr>
<td>


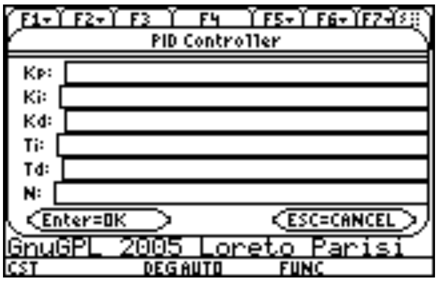
</td>
<td>

PD Controller

Defines a PD controller R(s) as you give Kp and Kd or Kp and Td:

$$R_{PD}(s) = K_P + K_D s = K_P(1 + T_D s)$$

where $T_D = \frac{K_D}{K_P}$

</td>
</tr>
<tr>
<td>


</td>
<td>

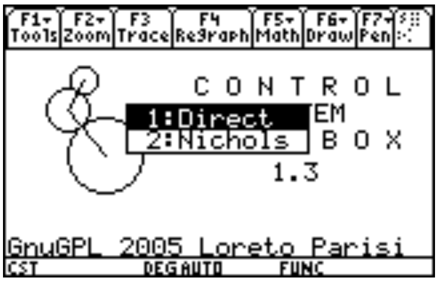
PID Controller

Defines a standard PID controller as you give Kp, Ki, Kd or Kp, Ti, Td:

$$R_{PID}(s) = \frac{K_D s^2 + K_P s + K_I}{s} = K_P \frac{T_I T_D s^2 + T_I s + 1}{T_I s}$$

or a real PID controller specifing N:

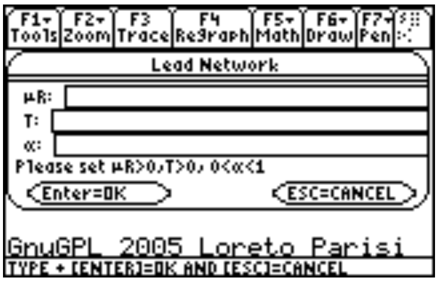
$$R_{PID}(s) = K_P + \frac{K_I}{s} + \frac{K_D s}{1 + \frac{K_D}{K_P N} s} = K_P\left(1 + \frac{1}{T_I s} + \frac{T_D}{1 + \frac{T_D}{N} s} s\right)$$

</td>
</tr>
<tr>
<td>


</td>
<td>

Direct Design

Defines a Lead, Lag or Lead-lag network directly from transfer function's gain μ_R, time constant T and α parameter.

</td>
</tr>
<tr>
<td>


</td>
<td>

Lead Network

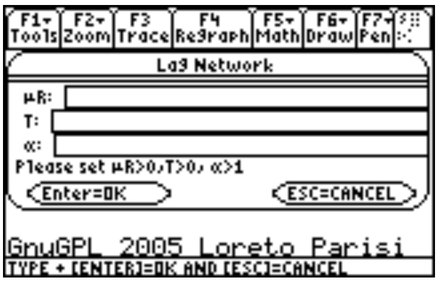
Defines a lead network R(s) as you give the gain μ_R, time constant T and parameter α:

$$R(s) = \mu_R \frac{1 + Ts}{1 + \alpha Ts}$$

Must be:

$\mu_R > 0, T > 0, 0 < \alpha < 1$

Usually, α=0.1 and $T = \frac{1}{\omega_c}$

</td>
</tr>
<tr>
<td>


</td>
<td>

Lag Network

Defines a lag network R(s) as you give the gain μ_R, time constant T and parameter α:

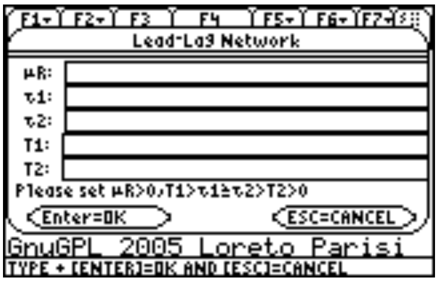
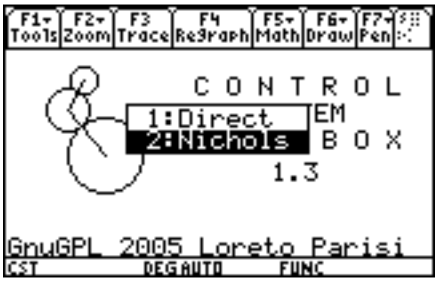
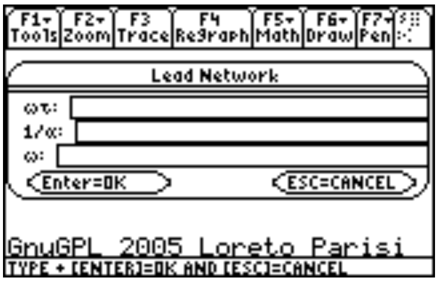
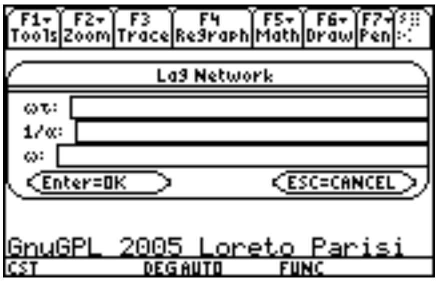
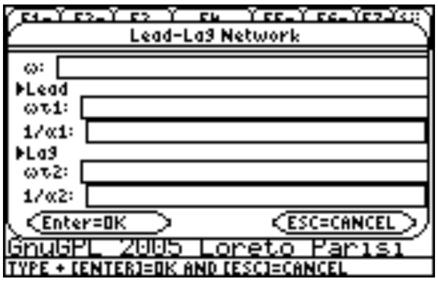
$$R(s) = \mu_R \frac{1 + Ts}{1 + \alpha Ts}$$

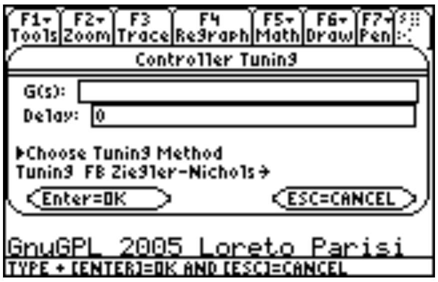
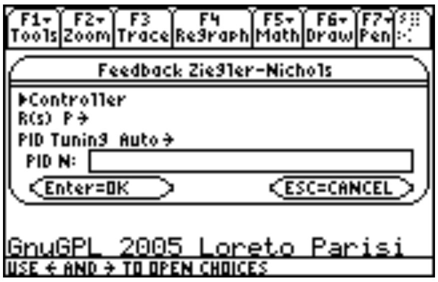
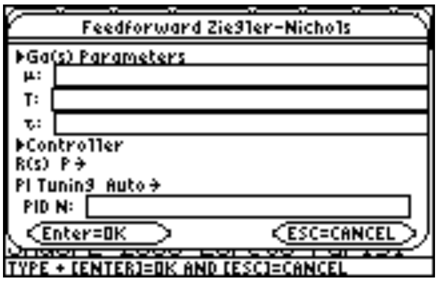
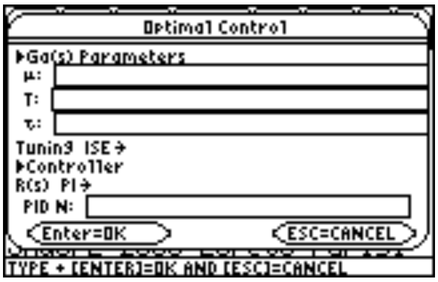
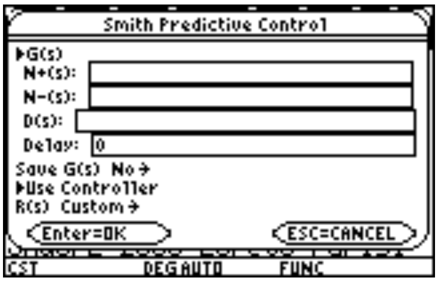
Must be:

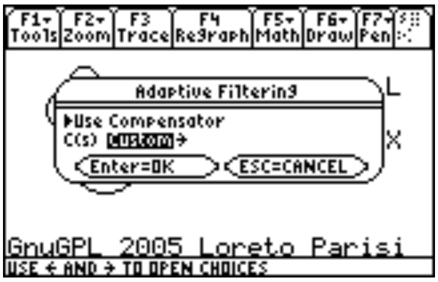
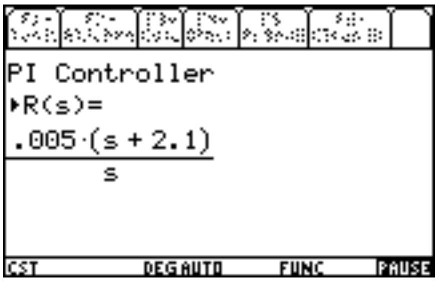
$\mu_R > 0, T > 0, \alpha > 1$

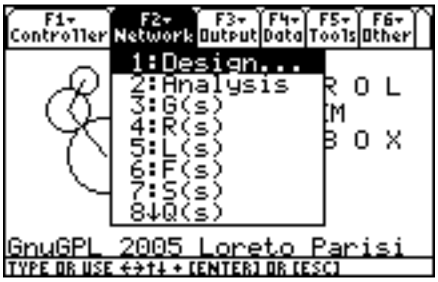
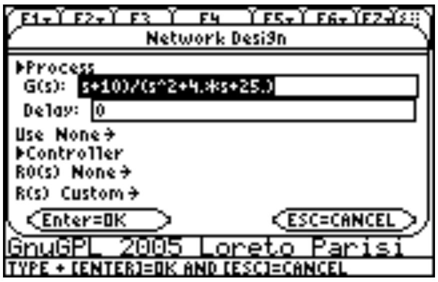
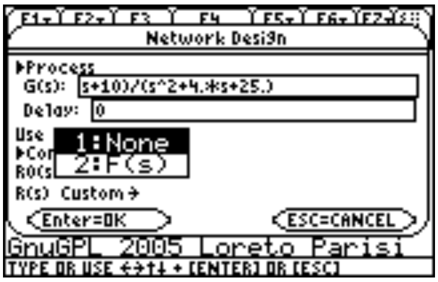
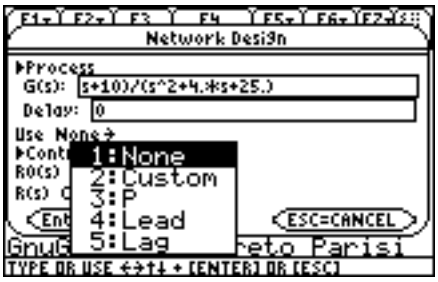
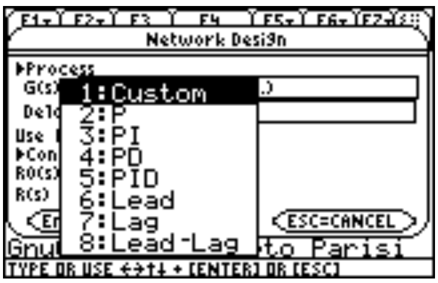
Usually, $T > \frac{1}{\omega_C}$ $(T = \frac{10}{\omega_C})$

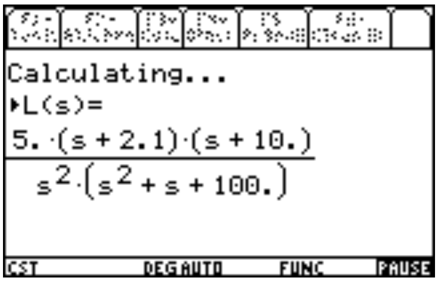
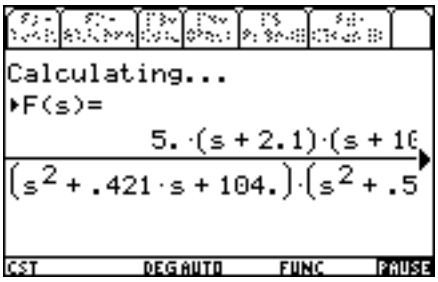
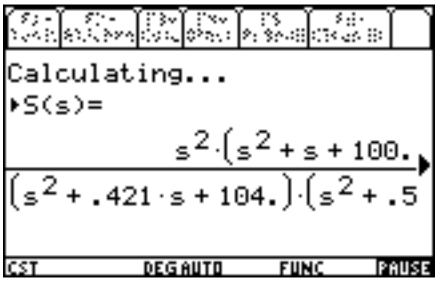
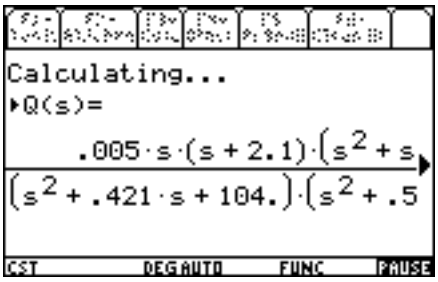
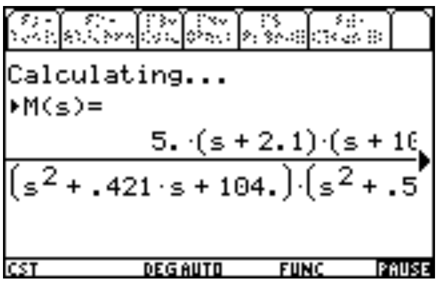
</td>
</tr>
</table>

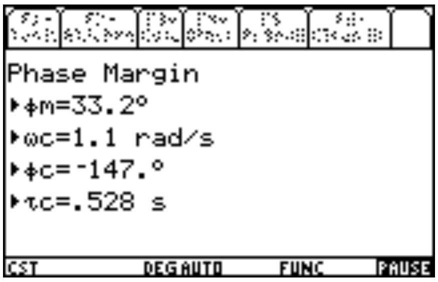
	***Lead-Lag Network*** Defines a lead-lag network R(s) as you give the gain μ_R, and time constants τ_1, τ_2, T_1, T_2: $R(s) = \mu_R \frac{(1+\tau_1 s)(1+\tau_2 s)}{(1+T_1 s)(1+T_2 s)}$ Must be: $\mu_R > 0, T_1 > \tau_1 \geq \tau_2 > T_2 > 0$ Usually, $T_1 T_2 = \tau_1 \tau_2, \tau_2 > \frac{1}{\omega_C} > T_2$
	**Nichols Design** Defines a Lead, Lag or Lead-lag network using Nichols's standard networks parameters $\omega\tau$, $1/\alpha$ at ω_0. In most cases ω_0 will be the critical frequency ω_C.
	**Lead Network** Defines a Lead network using Nichols's standard networks parameters $\omega\tau$, $1/\alpha$ at ω_0. $R(s) = \frac{1+s\tau}{1+s\alpha\tau}$
	**Lag Network** Defines a Lag network using Nichols's standard networks parameters $\omega\tau$, $1/\alpha$ at ω_0. $R(s) = \frac{1+s\alpha\tau}{1+s\tau}$
	**Lead-Lag Network** Defines a Lead-Lag network using Nichols's standard networks parameters $\omega\tau_1$, $1/\alpha_1$ for the lead and $\omega\tau_2$, $1/\alpha_2$ for the lag network at frequency ω_0. $R(s) = \frac{1+s\tau_1}{1+s\alpha_1\tau_1}\frac{1+s\alpha_2\tau_2}{1+s\tau_2}$

	***Controller Tuning*** The Controller Tuning wizard will guide you through the tuning of the network's controller R(s) for the given process G(s) and its delay. First choose the tuning method from ones avaiable: *Feedback Ziegler-Nichols*, *Feedforward Ziegler-Nichols*, *Optimal Control*, *Predictive Control* and *Adaptive Filtering.*
	***Feedback Ziegler-Nichols*** Uses the *Closed Loop Ziegler-Nichols* method to tune the controller for the feedback network. Choose the desidered structure for R(s) – P, PI or PID. Only for PIDs, choose the assignment method for gain and phase margins (Auto, assign Gain Margin or assign Phase Margin) and the parameter N if you wish to use a real PID controller, instead of a standard PID controller.
	***Feedforward Ziegler-Nichols*** Uses the *Open Loop Ziegler-Nichols* method to tune the controller for the approximate process (obtained from the step response using the *areas method*) $G_a(s) = \frac{\mu}{1+Ts}e^{-\tau s}$. Choose the structure (P, PI or PID) for R(s) and for PIs only the assignment method for the phase margin (Auto, assign Phase Margin).
	***Optimal Control*** Uses optimization methods to tune the controller R(s): ISE (*Integral Square Error*), ISTE (*Integral Square Time Error*) and IST^{2}E (*Integral Square Time^2 Error*). Kp, Ti and Td are defined by a table as follows: $K_P = \frac{a_1}{\mu}\theta^{b_1}, T_I = \frac{T}{a_2 + b_2\theta}, T_D = a_3 T\theta^{b_3}$
	***Smith Predictive Control*** Used to tune network's controllers for processes with postive real zeros or time delays. The process G(s) is given as $G(s) = \frac{N^-(s)N^+(s)}{D(s)}e^{-\tau s}$. The predictor P(s) and the network transfer function L'(s) for the given controller R(s) are $P(s) = \left(1 - \frac{N^+(s)}{N^+(-s)}e^{-\tau s}\right)\frac{N^-(s)N^+(-s)}{D(s)}$ and $L'(s) = (G(s)+P(s))R(s)$.

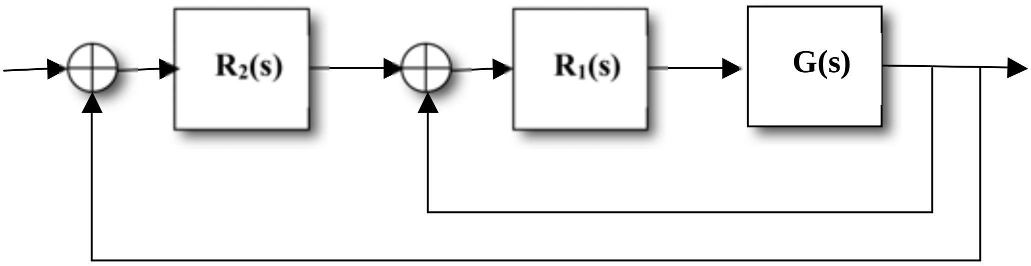
 	***Adaptive Filtering*** Uses a pre-filtering technique (compensation of input signal) to improve static and dynamic behaviour. You have to choose the structure for the compensator C(s). We suppose you have defined it as a controller (custom, lead, lag or lead-lag) yet.
 	***Custom, P, PI, PD, PID, Lead, Lag, Lead-Lag*** Shows the controller defined for that structure. Note that you have to choose the controller first to perform the analysis, but it's possible to define (design or tuning) more controllers, then choose one of them as the current R(s).

Network	
	The ***Network*** *menu (F2)* permits to design and analyse the control system, calculating gain and phase margins, and the network transfer functions.
	***Network Design*** Define the process G(s) and its delay, the controllers R_0(s) and R(s).
	***Network Design*** Use the inner loop transfer function F(s) as G(s) in the unstable processes control systems. Now you can tune the controller against the inner closed loop transfer function. See notes for more informations.
	***Network Design*** Choose the controller R_0(s) between custom, proportional, lead or lag structures. Generally, use it to satisfy static requirements.
	***Network Design*** Choose the controller R(s) between all structures avaiable to satisfy dynamic requirements.

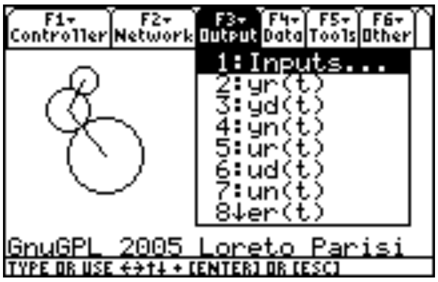
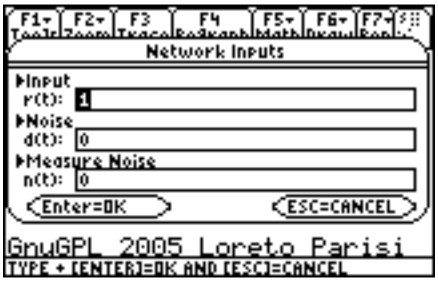
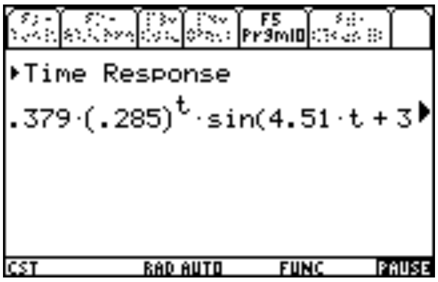
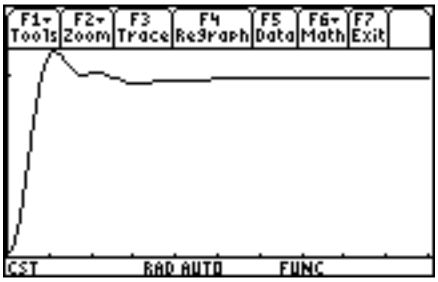
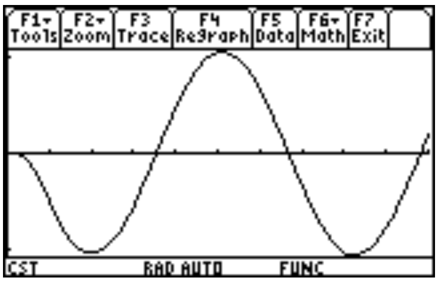
 	***L(s)*** The network tranfer function $L(s) = R(s)G(s)$
 	***F(s)*** The network transfer function $F(s) = \dfrac{R(s)G(s)}{1 + R(s)G(s)}$
 	***S(s)*** The network transfer function $S(s) = \dfrac{1}{1 + R(s)G(s)}$
 	***Q(s)*** The network transfer function $Q(s) = \dfrac{R(s)}{1 + R(s)G(s)} = F(s)G(s)^{-1} = R(s)S(s)$
 	***M(s)*** The network transfer function $M(s) = G(s)S(s)$

	***Analysis*** Performs an analysis of the defined control system, calculating the Gain and the Phase margins.

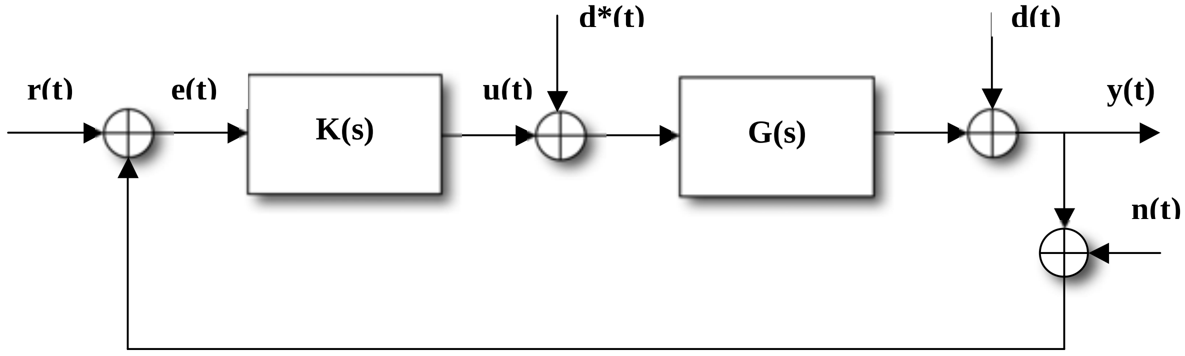
Notes.
About *Network Design*.
1. When G(s) is such an unstable process, we'll use a block model with a inner control loop, like the block diagram below.



And we'll tune $R_1(s)$ to stabilize G(s) and $R_2(s)$ against $F(s) = \frac{R_1(s)G(s)}{1 + R_1(s)G(s)}$ to satisfy given requirements. To do so, first design $R_1(s)$ as usual. When the inner closed loop is stable, you can design $R_2(s)$ choosing in *Network Design* **Use F(s)** as new G(s) from the drop down menu.

Output	
	The **Output** menu (F3) permits to perform a time domain analisys of the closed loop system against inputs, noises and measure noise.
	**Inputs** To set the input r(t), noise inputs d(t) and d*(t), and measure noise input n(t). To set a step input of amplitude K use K for r(t). To set a sinusoidal measure noise, use sin(ωt) as n(t).
	**yr(t), yd(t),yn(t)** The output y time response against input r(t), noise input d(t) and measure noise input n(t).
	**ur(t), ud(t), un(t)** The control variable u time response against input r(t), noise input d(t) and measure noise input n(t).
	**er(t), ed(t), en(t)** The error e=r-y time response against input r(t), noise input d(t) and measure noise input n(t).

Notes.
About *Output* menu.
1. We assume a Closed Loop Control System block model like that below.



2. We assume the following transfer functions.

$$\begin{bmatrix} Y(s) \\ U(s) \\ E(s) \end{bmatrix} = \begin{bmatrix} F(s) & S(s) & -F(s) \\ Q(s) & -Q(s) & -Q(s) \\ S(s) & -S(s) & F(s) \end{bmatrix} \cdot \begin{bmatrix} R(s) \\ D(s) \\ N(s) \end{bmatrix}$$

where

$$F(s) = \frac{K(s)G(s)}{1+K(s)G(s)},$$

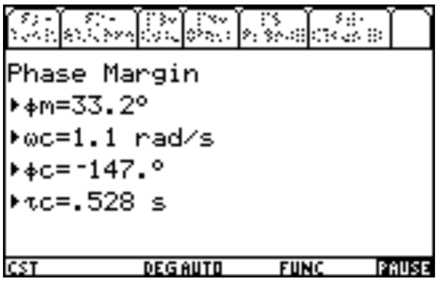
$$S(s) = \frac{1}{1+K(s)G(s)},$$

$$Q(s) = \frac{K(s)}{1+K(s)G(s)},$$

and *Y*(s)=M(s)D*(s)* where *M(s)=G(s) S(s).*

Data

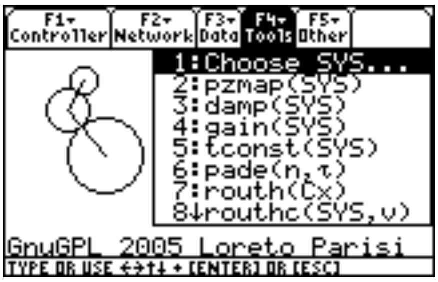
The ***Data*** menu (F3) shows detailed informations about the control system.



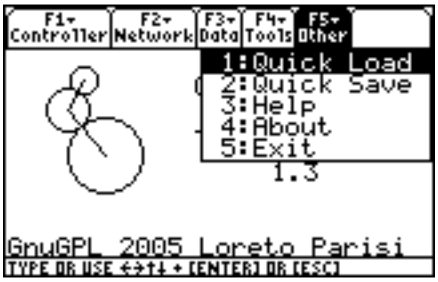
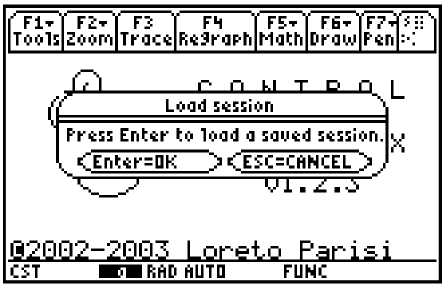
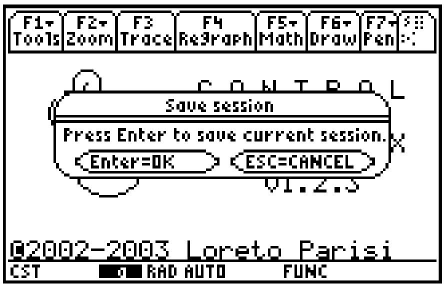
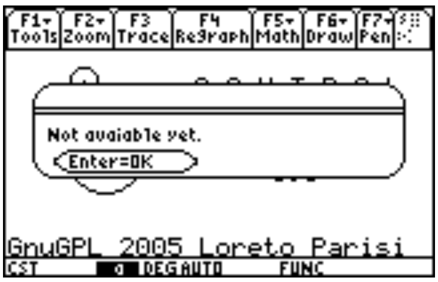
Margins

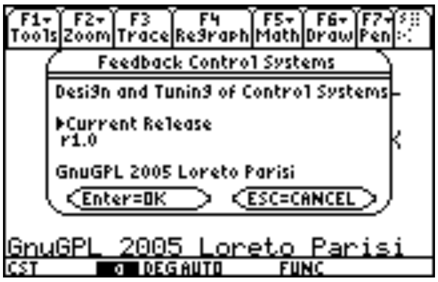
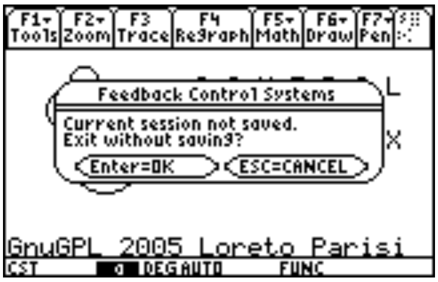
Shows the Gain and Phase margins of the control system.

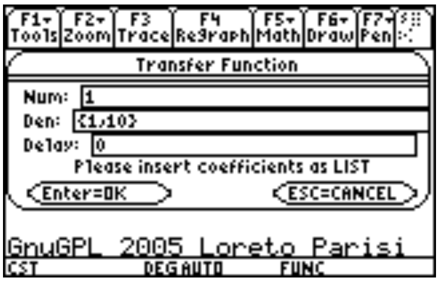
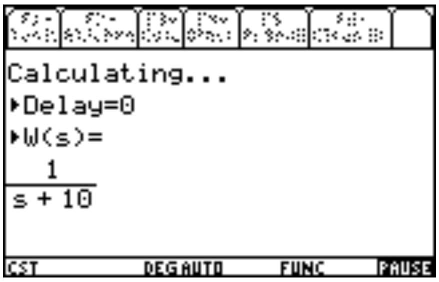
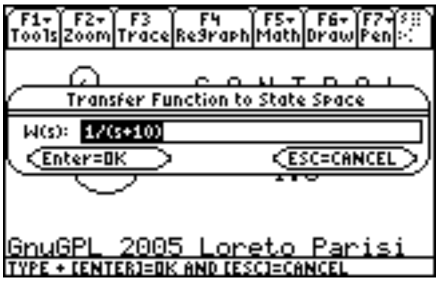
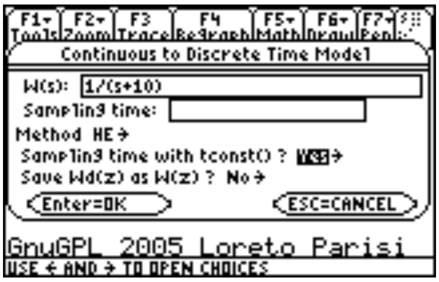
Tools

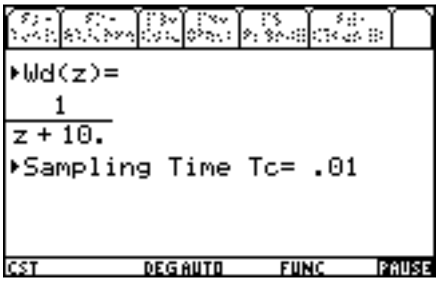
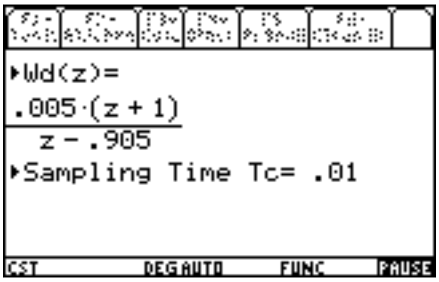
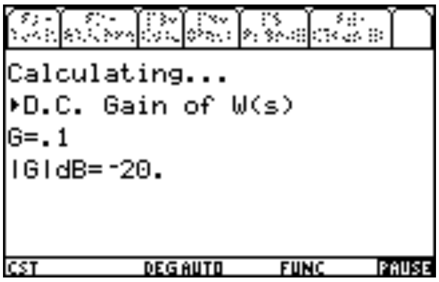
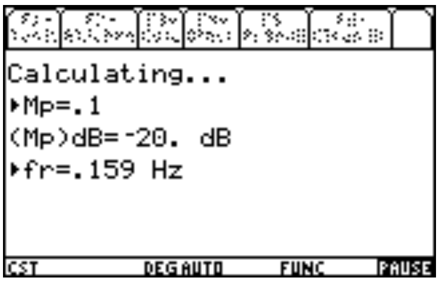


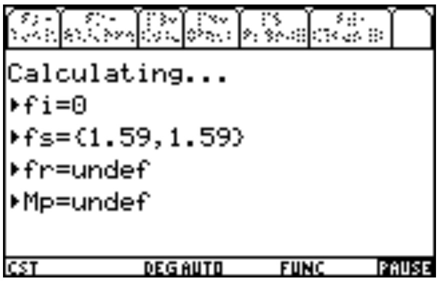
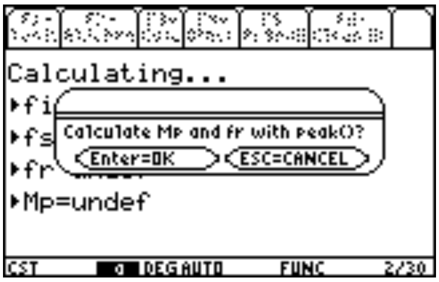
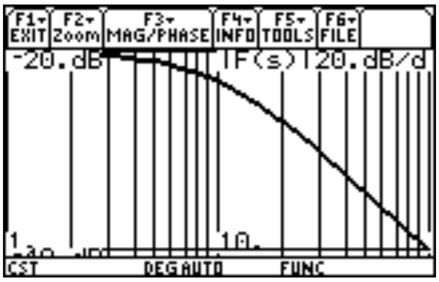
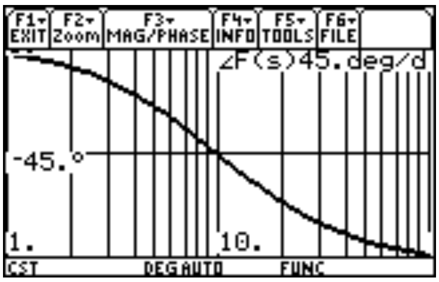
The ***Tools*** menu (F4) permits you to perform a detailed analysis of transfer function SYS. You need to choose the current transfer fuction SYS first.

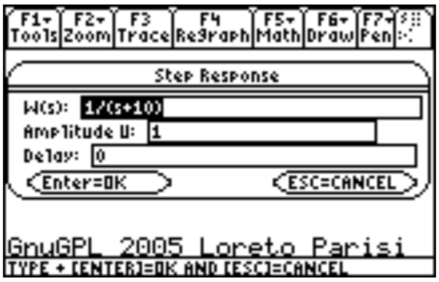
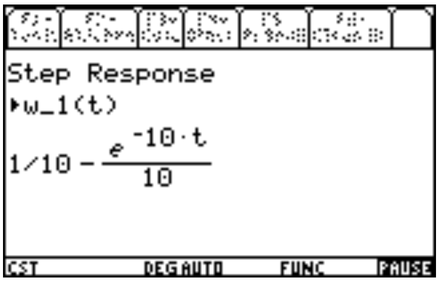
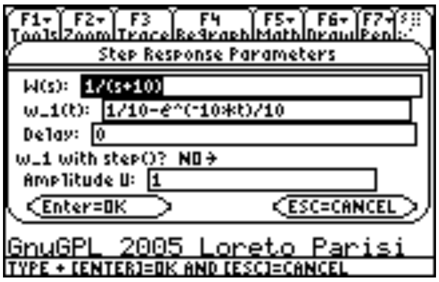
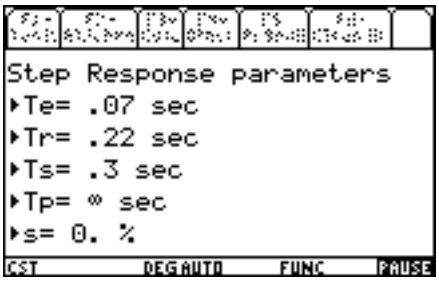
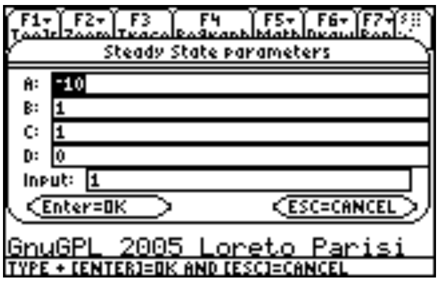
<table>
<tr><th colspan="2">Other</th></tr>
<tr><td>
</td>
<td>The Other menu (F3) shows some help, info and permits to exit the program.</td></tr>
<tr><td>
</td>
<td>Quick Load
Loads the current working session (i.e. transfer functions of process, network and controllers, calculated margins, etc.) previously saved. It overwrites all the existing values for the current session. Be careful.</td></tr>
<tr><td>
</td>
<td>Quick Save
Saves the current working session (i.e. transfer functions of process, network and controllers, calculated margins, etc.).</td></tr>
<tr><td>
</td>
<td>Help
Shows some help.</td></tr>
</table>

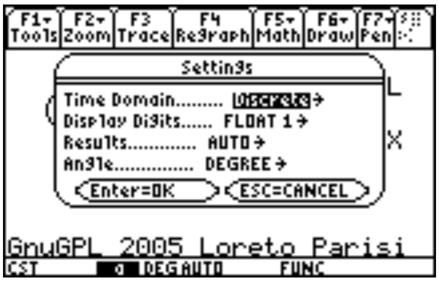
	***About*** Shows some info.
	***Exit*** Exit the program. Unsaved session will be losed.

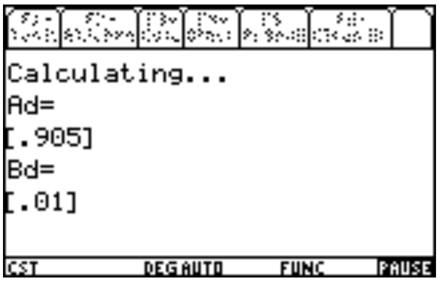
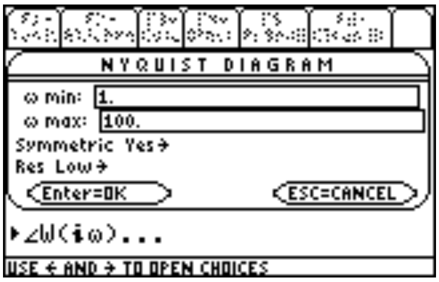
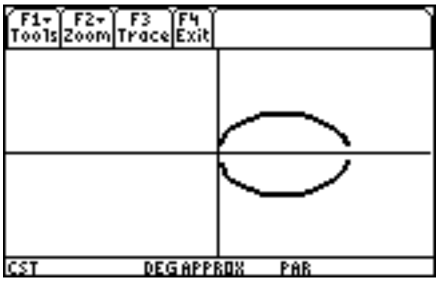
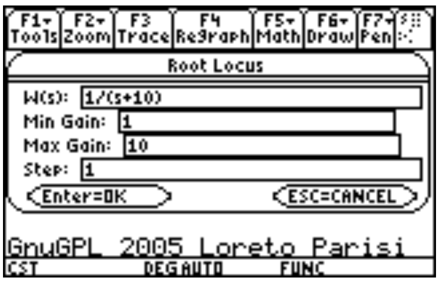
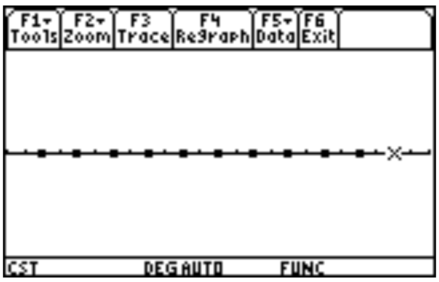
Examples. 1 - First Order LPF.	
	Considers a sample low pass filter, with transfer function $\frac{1}{1+10s}$. First of all, we'll define the current transfer function. From Systems menu (F1) choose function tf(NUM,DEN) (3) where NUM=1 and DEN={1,10}.
	This stores $\frac{1}{1+10s}$ as the current transfer function SYS.
	To get relating state-space, now we'll use function tf2ss(SYS), where SYS is the current transfer function, automatically filled in the input field.
	After calculation of all matrixes, they will be the current state space.
	If you need to obtain the discrete model of SYS, use c2d(SYS,Tc) where Tc is the sample time desidered. We'll use tconst(SYS) to get Tc and 'HE' (Hold Equivalence) transformation as method. We could save resulting Wd(z) as the current discrete transfer function W(z).

<table>
<tr><td>
</td><td>Here is the discrete time model with sample time Tc from tconst(SYS) and ‘HE’ (Hold Equivalence) method.</td></tr>
<tr><td>
</td><td>This is the discrete time model with sample time Tc from tconst(SYS) and ‘Tustin’ method.</td></tr>
<tr><td>
</td><td>Now, we’ll gonna calculate poles and zeros of SYS. From Analysis menu (F2) we’ll choose pzmap(SYS) where SYS is the current transfer function.</td></tr>
<tr><td>
</td><td>With function dcgain(SYS) we have calculated the d.c. gain of SYS, which results –20 dB (0.1 linear).</td></tr>
<tr><td>
</td><td>In the same way, with function peak(SYS) we’ll obtain the resonance peak MP (in this case it matches with previous d.c. gain G) and relating resonance frequency fr that results .159 Hz.</td></tr>
</table>

	We can calculate the Bandwith B_3 choosing band(SYS) from Tools menu (F5).
	As you can see, M_P and fr will results from peak(SYS) when band(SYS) fails.
	From Dynamics menu (F3) we'll going to plot Bode diagrams with function bodex(SYS), courtesy of 92BROTHERS. We'll use w and 0.01 and 100 as transfer function, ωmin and ωmax. Current SYS is showned in the W(s) input field.
	The Mag Bode diagram. Use Get Point from Info (F4) to trace the plot and get a specific value for magnitude. You can even save this plot from File (F6).
	The Phase Bode diagram. Use Get Point from Info (F4) to trace the plot and get a specific value for phase. You can even save this plot from File (F6).

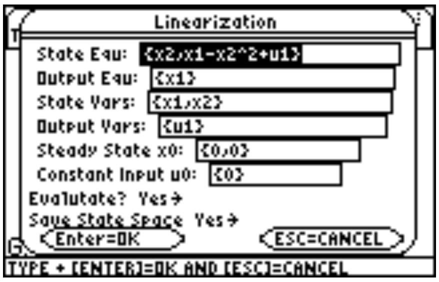
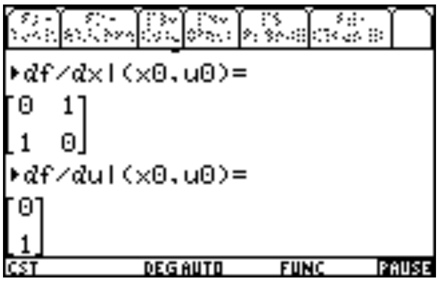
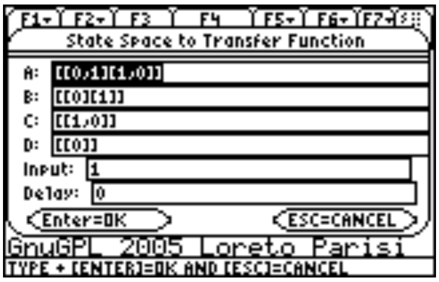
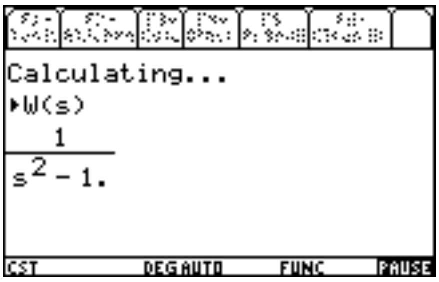
	We can calculate the step response using step(SYS). Set the amplitude (1 by default) and the time delay.
	Here is the step response.
	Then we could calculate the time domain step response parameters using function pstep(SYS). We can decide to use current step response or define a new one with step(SYS), for wich we will set the amplitude and time delay.
	Here are the time domain step response parameters, Te, Tr, Ts, Tp and s.
	To get the steady state we'll use function trim(A,B,C, D,u0).

	Note that x0 and y0 matches with w-1(t).
	With spectre(A) we'll obtain e^{At} in the Continuous Time Domain. Because of real pole, we have not complex expansion.
	Now we will set the Time Domain to Discrete from Settings in Other (F6) menu.
	With spectre(A) we'll obtain A^K in the Discrete Time Domain. Because of real pole, we have not complex expansion.
	Here we have obtained a discrete model of SYS, using ZOH method. We can save the resulting discrete transfer function as the current discretized transfer function.

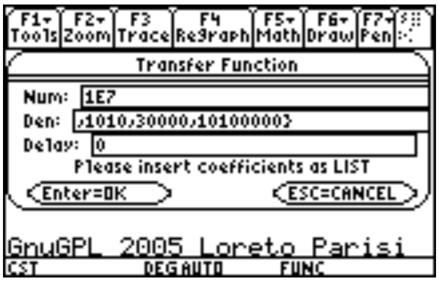
 	Here are the Discrete time model of SYS.
 	Now we will plot the Nyquist diagram to study the asimphtotic stability of SYS. Choose the ω range, the resolution and if the polar diagram must be symmetric against the real axe. nyquist(SYS) Control Toolbox v1.16 Author: Francesco Orabona E-mail: bremen79@infinito.it
 	Here is the Nyquist diagram. We can zoom, save and trace it.
 	We can now plot the Root Locus for SYS. We must set the range of gains and the plot's step. rlocus(SYS) Control Toolbox v1.16 Author: Francesco Orabona E-mail: bremen79@infinito.it
 	Here is the Root Locus of SYS. We can zoom, trace and get significant plot info from Data menu (F5).

<table>
<tr><th colspan="2">Examples. 2 – Linearization.</th></tr>
<tr><td>Now consider a non linear model:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_1 - x_2 \\ y = x_1 \end{cases}$$

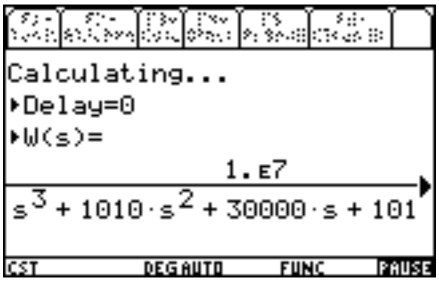
To work with it, we need to linearize around a steady state x0 relating to constant input u0.</td><td>First, we have to calculate the steady state x0. It results $x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ for input u0=0.</td></tr>
<tr><td>
</td><td>From Dynamics menu (F3) we choose linmod(f,y,x,u,x0,u0) where we have f={x2,x1-x2 2 +u1}, y={x1}, x={x1,x2}, u={u1}, x0={0,0} and u0={0}. We have decided to evalutate jacobian matrixex in x0 and u0 to save the state-space of obtained linearized model.</td></tr>
<tr><td>
</td><td>Here the jacobian matrixes evalutated in (x0,u0).</td></tr>
<tr><td>
</td><td>Now, we can get transfer function of this new model, that is a approximation of non-linear model above around x0 and u0.</td></tr>
<tr><td>
</td><td>Here is the linearized model transfer function. We can now procede to study this system in the usual way with our powerful tools.</td></tr>
</table>

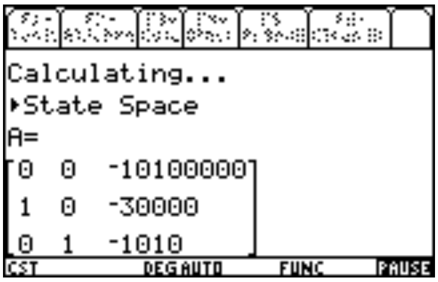
Examples. 3 – 3rd Order LPF.

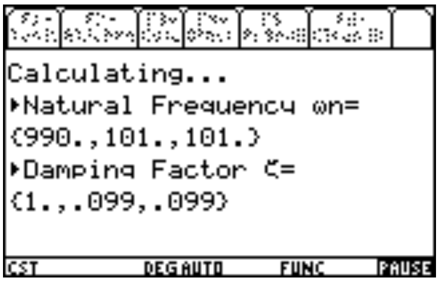


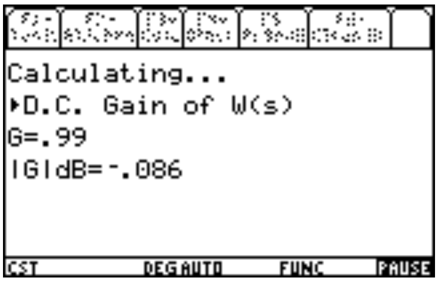
Consider a 3rd order low pass filter

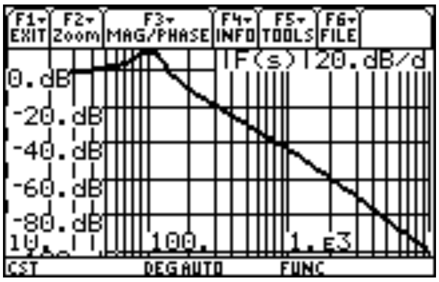
$$W(s) = \frac{10000000}{s^3 + 1010s^2 + 30000s + 10100000}$$

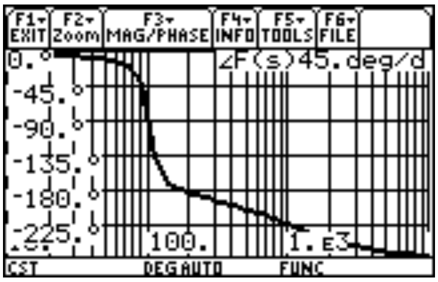


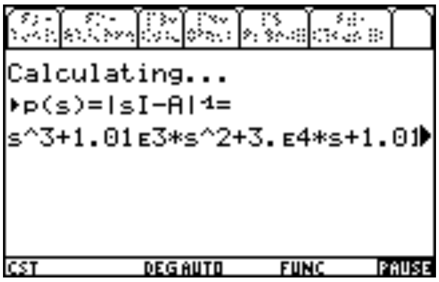


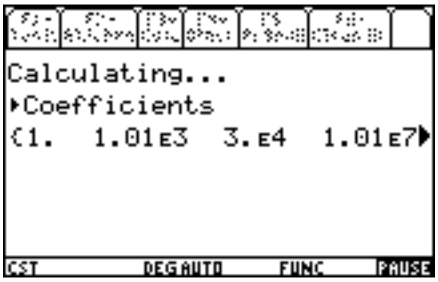


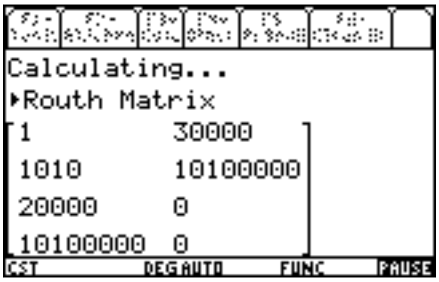


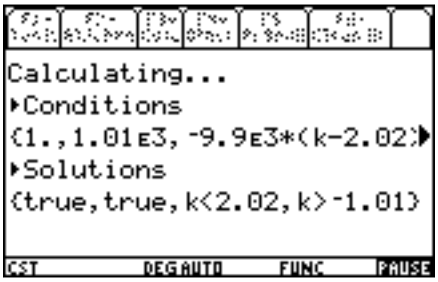


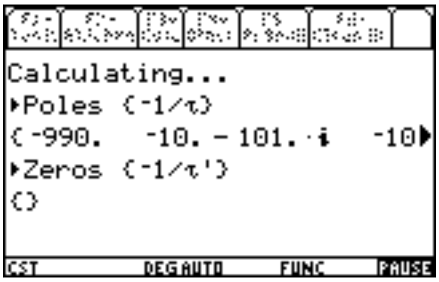


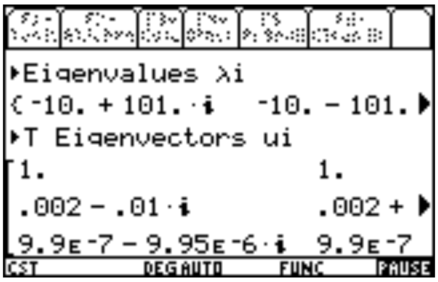


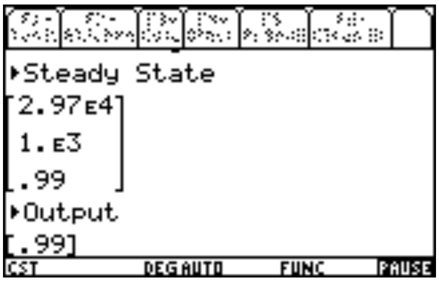


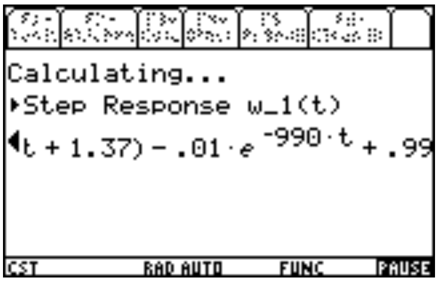

Calculating...
▸Routh Matrix
1 30000
1010 10100000
20000 0
10100000 0
CST DEG AUTO FUNC PAUSE

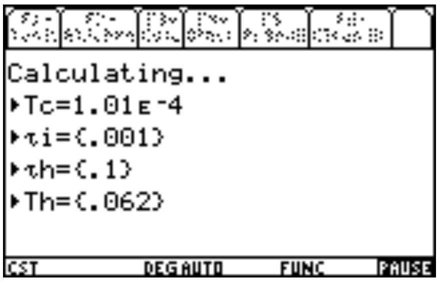

Calculating...
▸Conditions
{1.,1.01E3,-9.9E3*(k-2.02)
▸Solutions
{true,true,k<2.02,k>-1.01}
CST DEG AUTO FUNC PAUSE

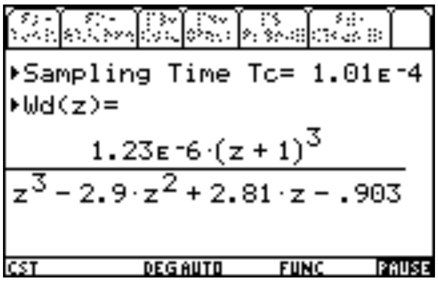

Calculating...
▸Poles {-1/τ}
{-990. -10. - 101.·i -10
▸Zeros {-1/τ'}
{}
CST DEG AUTO FUNC PAUSE

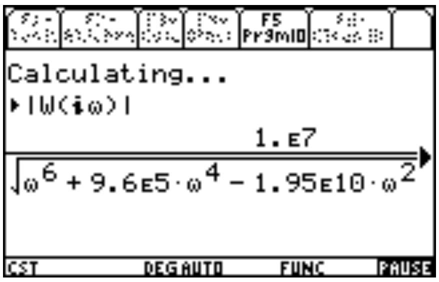

▸Eigenvalues λi
{-10. + 101.·i -10. - 101.
▸T Eigenvectors ui
1. 1.
.002 - .01·i .002 +
9.9E-7 - 9.95E-6·i 9.9E-7
CST DEG AUTO FUNC PAUSE

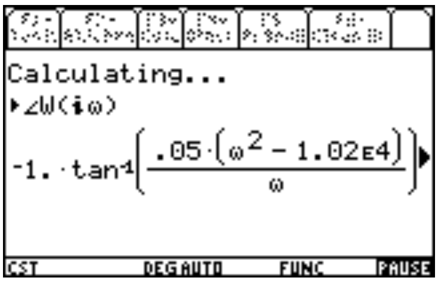

▸Steady State
2.97E4
1.E3
.99
▸Output
[.99]
CST DEG AUTO FUNC PAUSE

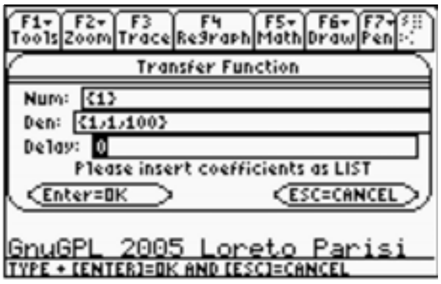
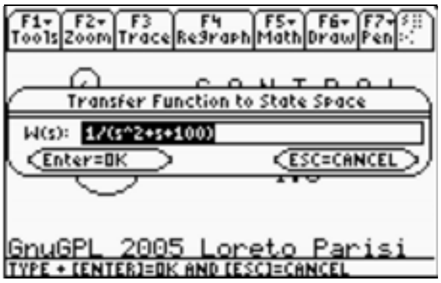
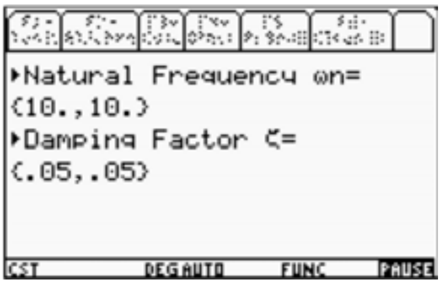
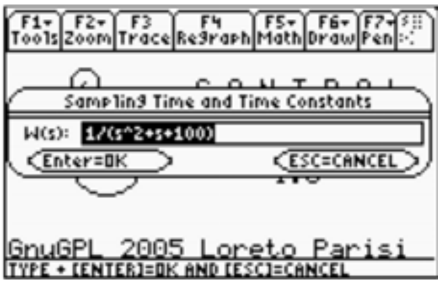
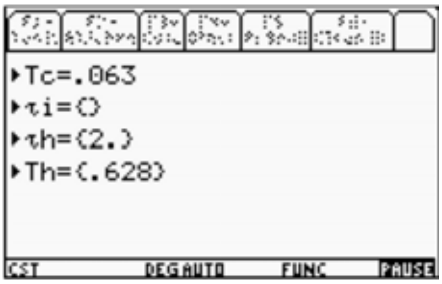
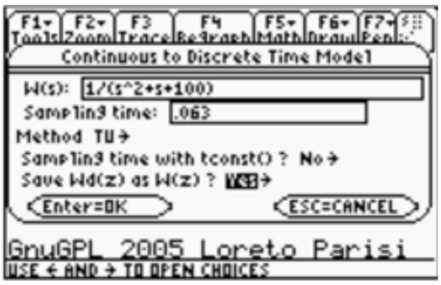
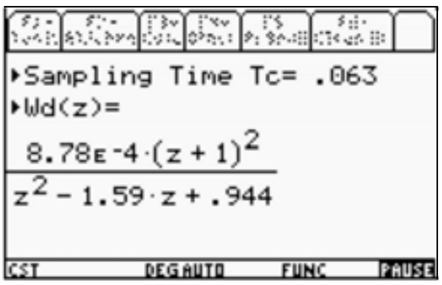

Calculating...
▸Step Response w_1(t)
t + 1.37) - .01·e^-990·t + .99
CST RAD AUTO FUNC PAUSE

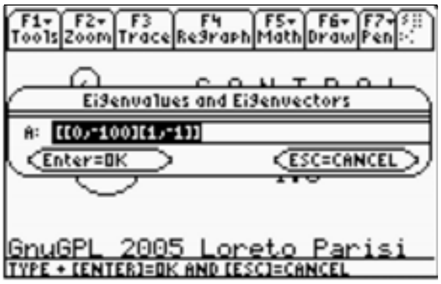
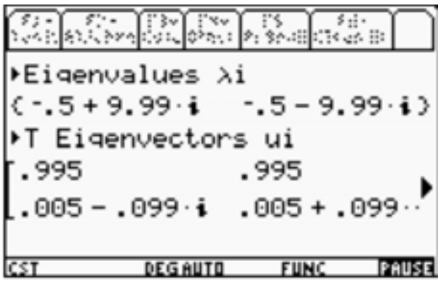
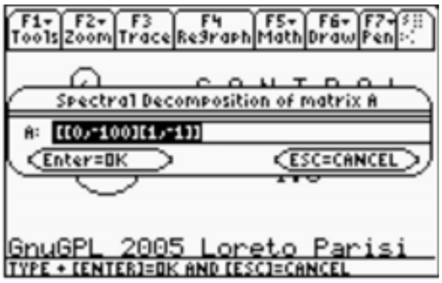
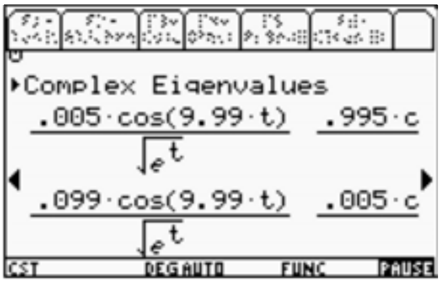
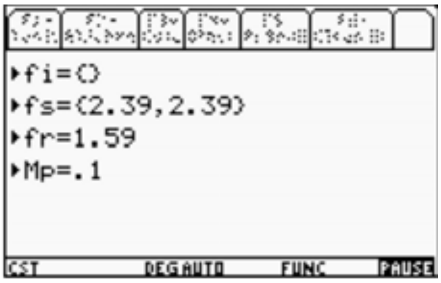
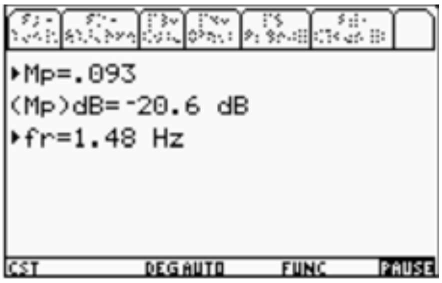
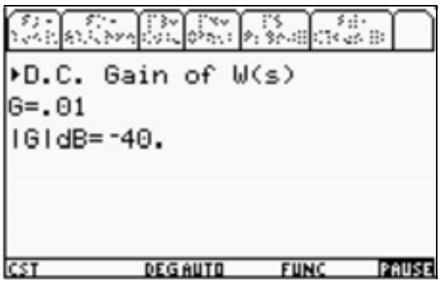

Calculating...
▸Tc=1.01E-4
▸τi={.001}
▸τh={.1}
▸Th={.062}
CST DEG AUTO FUNC PAUSE

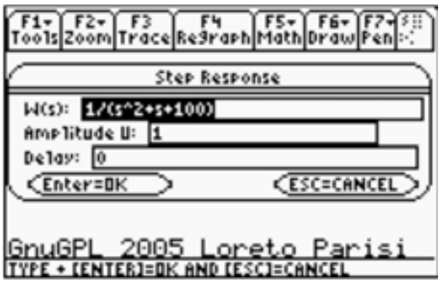

▸Sampling Time Tc= 1.01E-4
▸Wd(z)=
1.23E-6·(z + 1)^3
z^3 - 2.9·z^2 + 2.81·z - .903
CST DEG AUTO FUNC PAUSE


F5 PrgmIO
Calculating...
▸|W(iω)|
1.E7
√(ω^6 + 9.6E5·ω^4 - 1.95E10·ω^2
CST DEG AUTO FUNC PAUSE

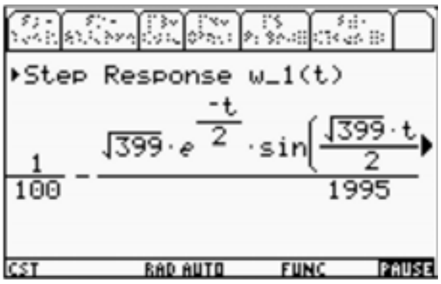

Calculating...
▸∠W(iω)
-1.·tan⁻¹(.05·(ω^2 - 1.02E4) / ω)
CST DEG AUTO FUNC PAUSE

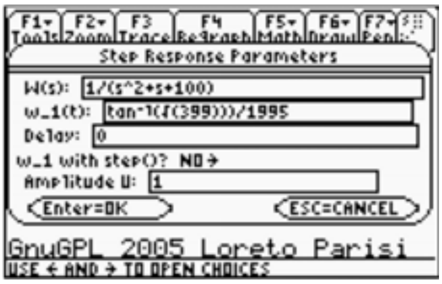
Examples. 4 – 2nd Order LPF.	
	Consider a 2nd order low pass filter $W(s) = \dfrac{1}{s^2 + s + 100}$. First, we will define the Transfer Function and obtain the State Space Representation.
	Now we'll get the characteristic polynomial, the natural frequency ω_n and the damping factor ξ.
	Then we'll calculate the maximum Time Constant, Sample Time and other time constants, and the Discrete Time model relating to this Sample Time with Tustin method. We will save this transfer function Wd(z) as the current discrete transfer function W(z).
	
	

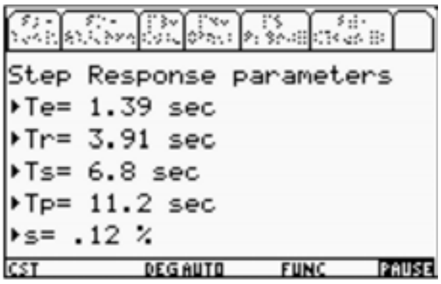
	Now we will calculate Eigenvalues and eigenvectors, and the Spectral Decomposition of dynamics matrix A.
	
	Now let's calculate the Bandwidth, the Resonance Peak and the D.C. Gain.
	
	As we can see, *band()* and *peak()* sometimes differs in resulting values of Mp and fr because of numerical nature of *peak()* algorithm, instead of formulas in *band()*.



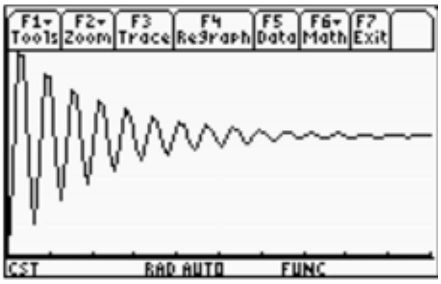
Let's now calculate and graph the Step Response. Finally we will get the Step Response parameters. This will take some minutes. Please be patient!
If *pstep()* fails, the Error Management System prevents *cst* to abort, so don't break execution manually (you will loose all unsaved session's data). The EMS will do it for you, safing the current session.

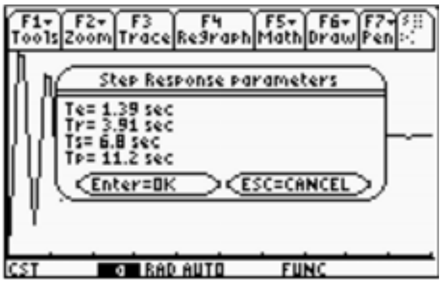


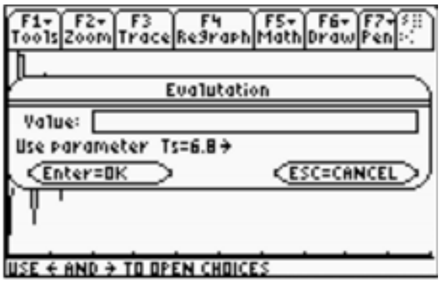


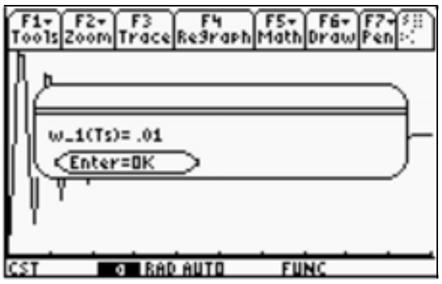


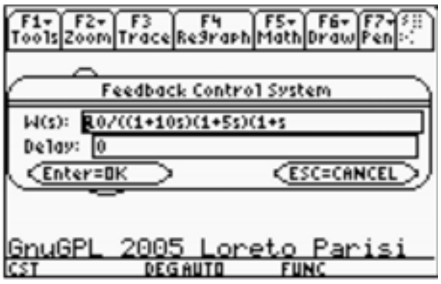
Now we can analyse in the Time Domain the Step Response with *gstep()*. We can see the Step Response parameters (Data – F5), zoom and trace the graph, get a value (F6) such as the Settlement Time, Ts.



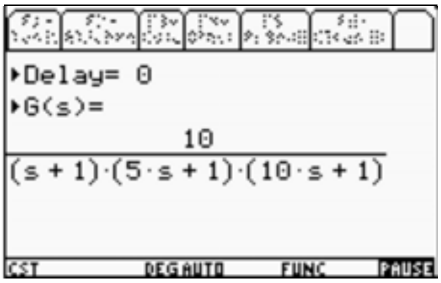


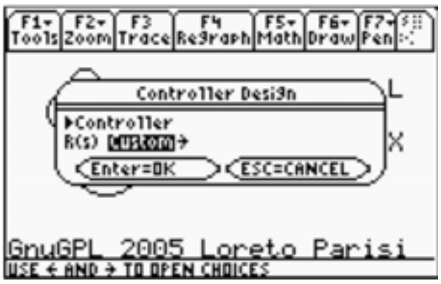


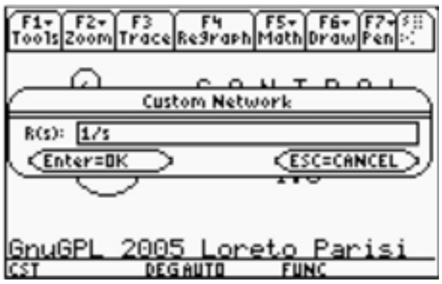


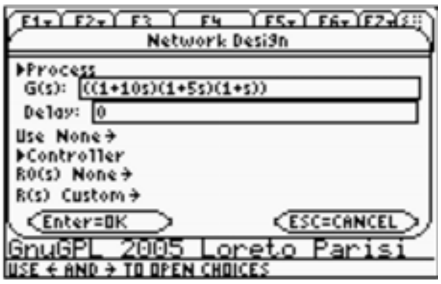
<table>
<tr><th colspan="2">Examples. 5 – Network Design I</th></tr>
<tr><td>


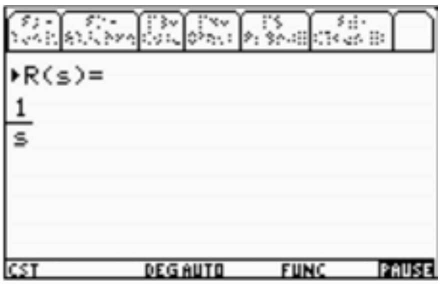
</td><td>
We want to design a closed-loop control system for transfer function

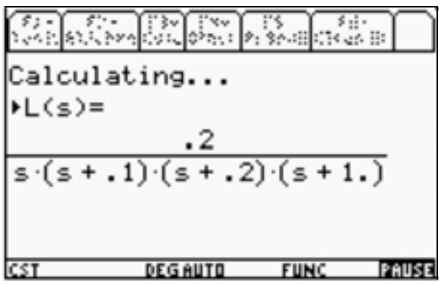
$$W(s) = \frac{10}{(1+10s)(1+5s)(1+s)}$$
</td></tr>
<tr><td>


</td><td>
This will be the current process transfer function, G(s), as we can see in **Network** menu (F2). Now we will design the controller. Choose *Design* from **Controller** menu (F1) and select the desidered structure. We will choose Custom.
</td></tr>
<tr><td>


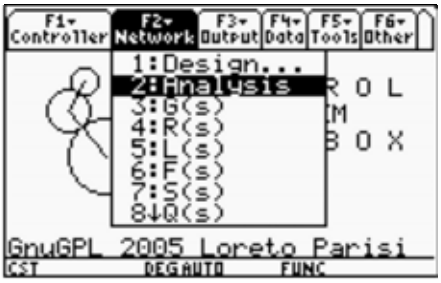
</td><td>


</td></tr>
<tr><td>


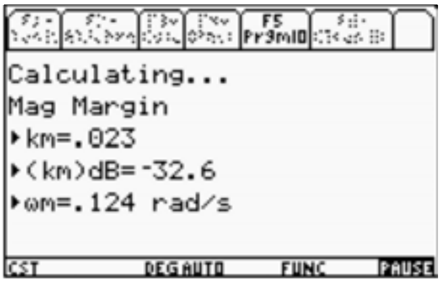
</td><td>
From **Network** menu (F2) choose *Design* to design the network. As you can see, you can specify more than one controller. We will use the previously defined Custom controller as R(s) and, by now we don't specify R0(s). Then we will get the Closed Loop Transfer Function, L(s).
</td></tr>
<tr><td>


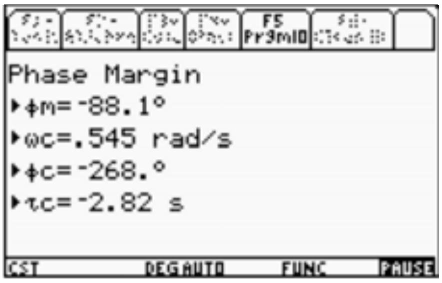
</td><td>


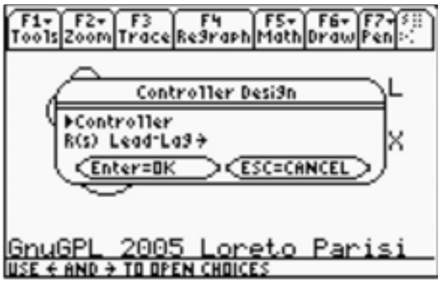
</td></tr>
</table>



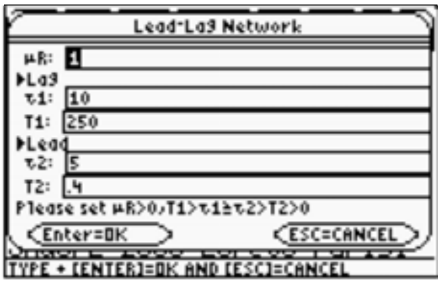
Now we will analyse the network using *Analisys* from **Network** menu (F2), obtaining the Magnitude (from now Mag) and Phase margins. As we can see, they are both negative. We need a lead network to get a phase advancing and a lag network to get a good mag margin.

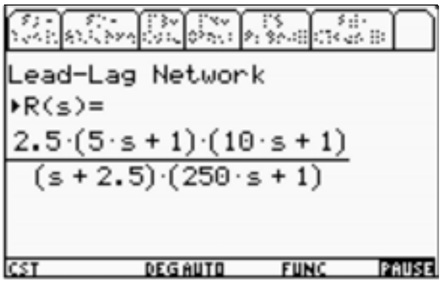


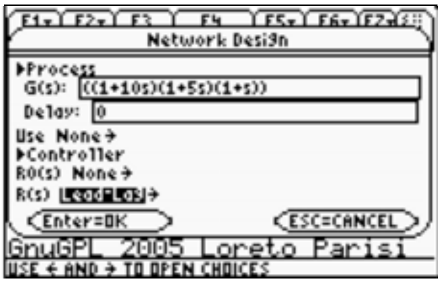


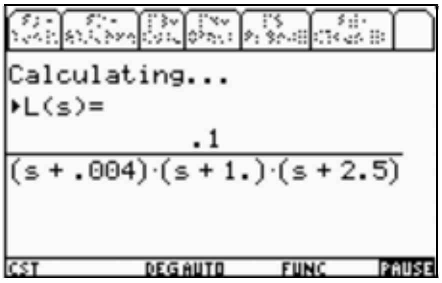


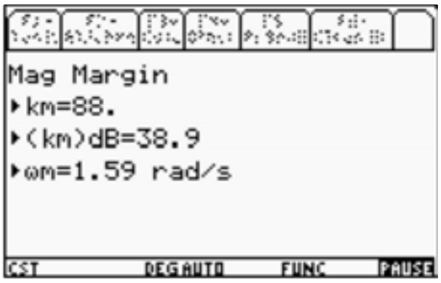
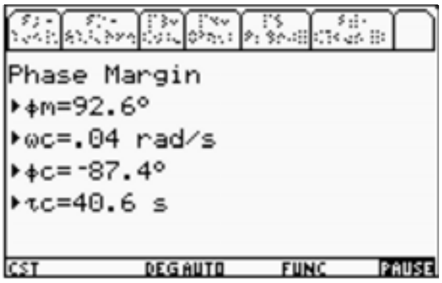
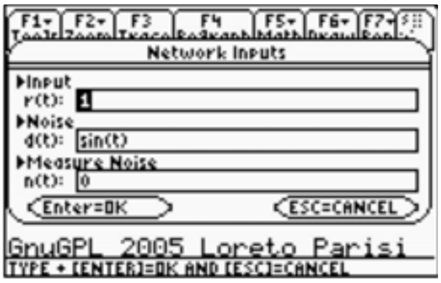
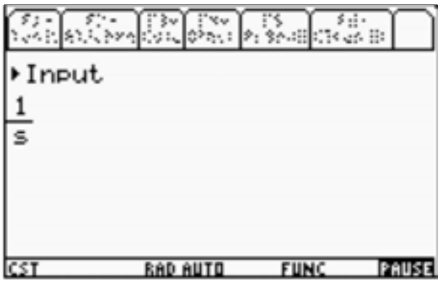
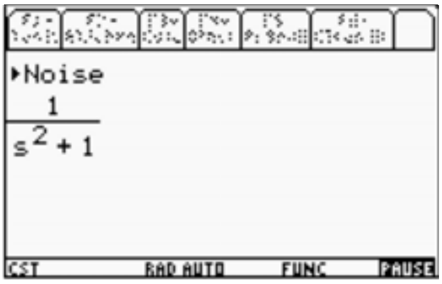
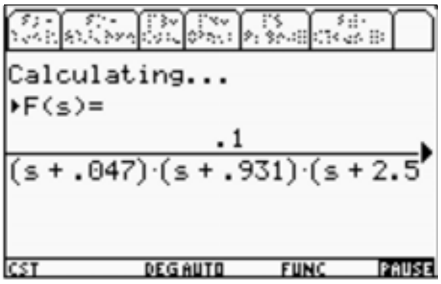
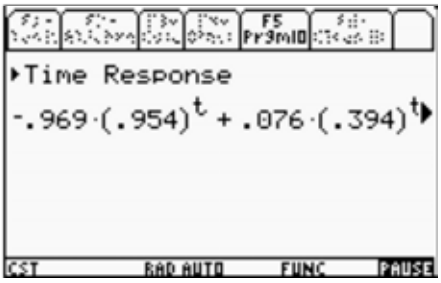
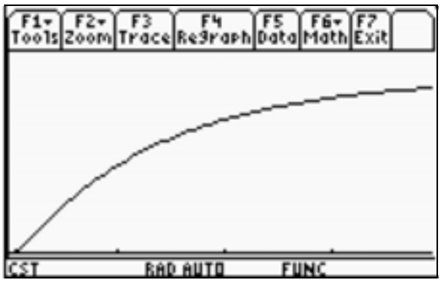
So we will define a Lead-Lag network from **Controller** menu (F1), and do the analysis again. To do so, we need to define again the network structure from **Network** menu(F2).

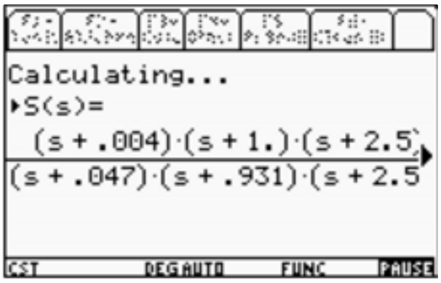








	
	We have obtained robust margins for ωc=0.04 rad/sec. We can now define inputs from **Output** menu (F3) to analyse network time behaviour. Apply a Heavside step as input and a simply sin wave as noise. Suppose we have no measure noise. We now can see these inputs in Laplace domain.
	
	Now we need to calculate the Network Transfer Function $F(s)=\dfrac{R(s)G(s)}{1+R(s)G(s)}$ to obtain the Time Response against input from **Network** menu(F2). Then we will choose yr(t) from **Output** menu (F3).
	

<table>
<tr>
<td>


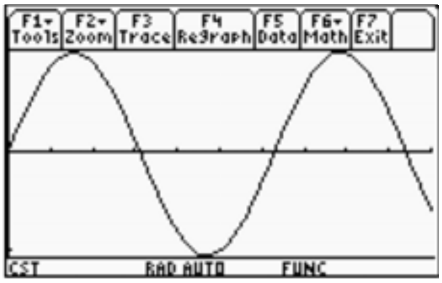
</td>
<td>

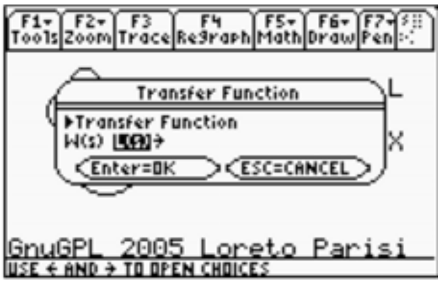
To obtain the Time Response against noise we need to calculate the transfer function

$$S(s) = \frac{1}{1 + R(s)G(s)}$$

from **Network** menu(F2), otherwise we'll have a *Not Defined* error. Then we will choose yd(t) from the **Output** menu(F3).

</td>
</tr>
<tr>
<td>


</td>
<td>


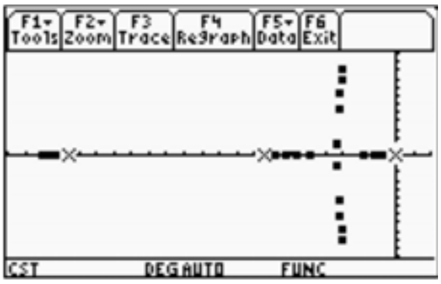
</td>
</tr>
<tr>
<td>


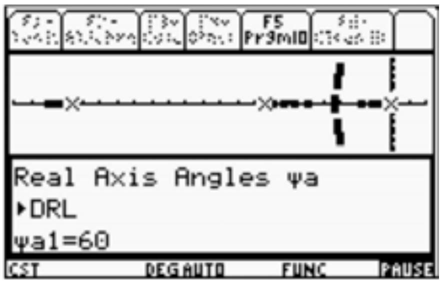
</td>
<td>

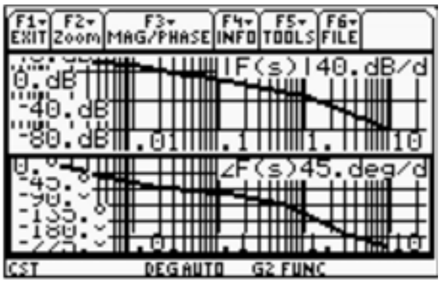
We can even plot the Root Locus and Bode diagrams of the Closed-Loop transfer function

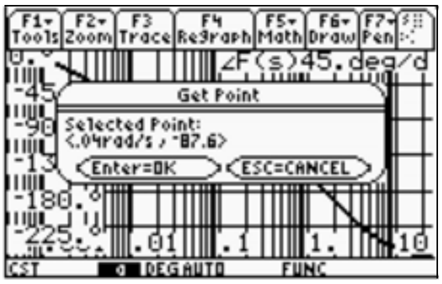
$L(s) = R(s)G(s)$

In **Tools** menu (F5) select *Choose SYS* and choose L(s). This will be the current SYS to apply all Tools menu functions. The Root Locus confirms closed-loop stability.

</td>
</tr>
<tr>
<td>


</td>
<td>


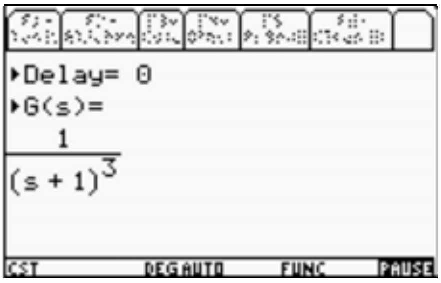
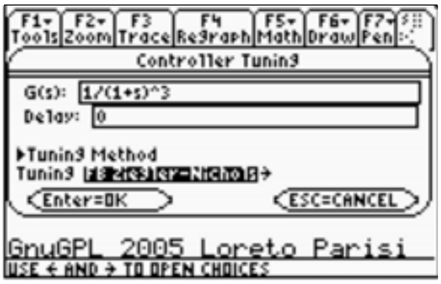
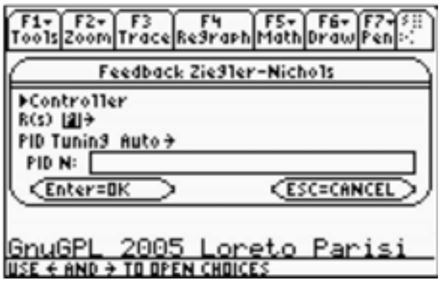
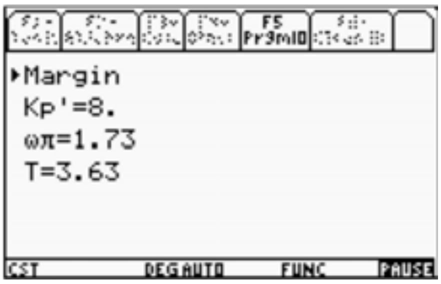
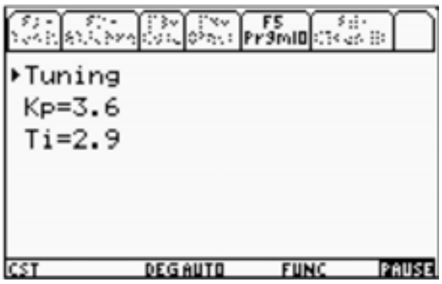
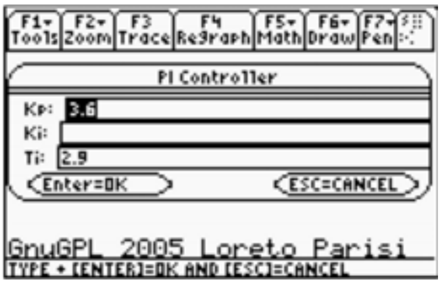
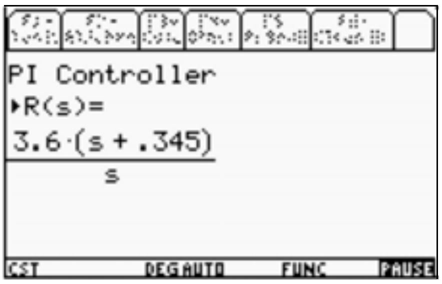
</td>
</tr>
<tr>
<td>


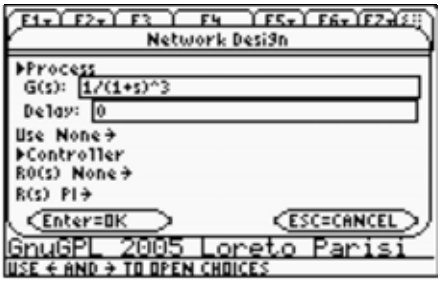
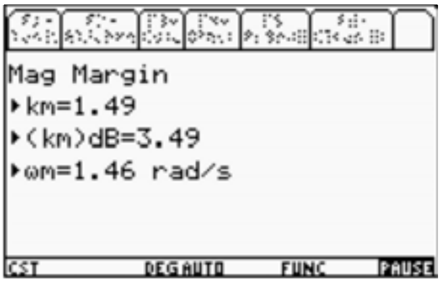
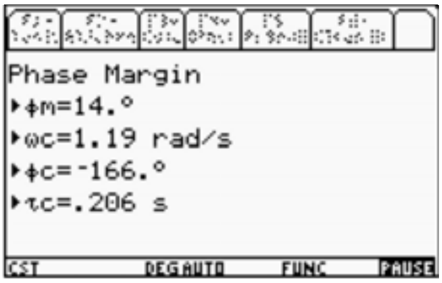
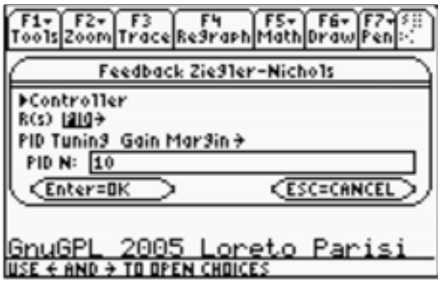
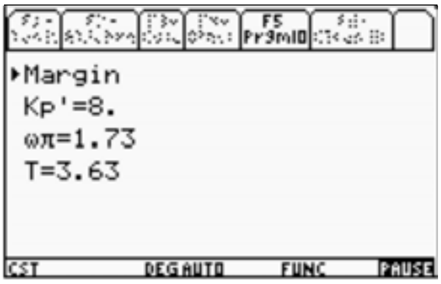
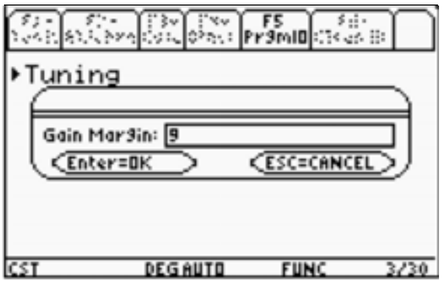
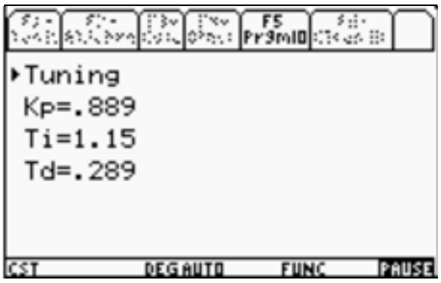
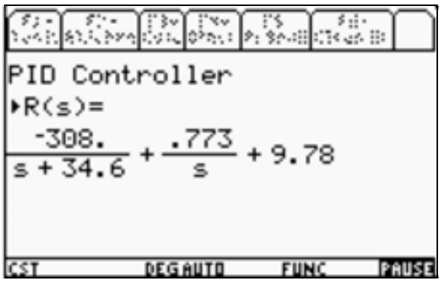
</td>
<td>


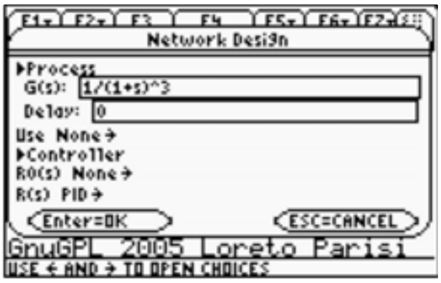
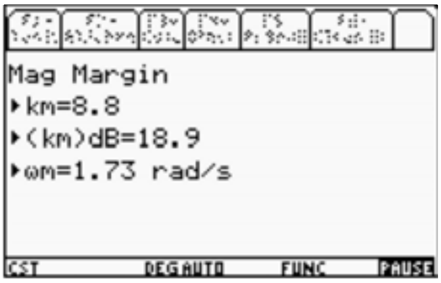
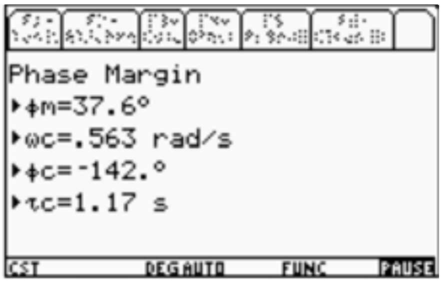
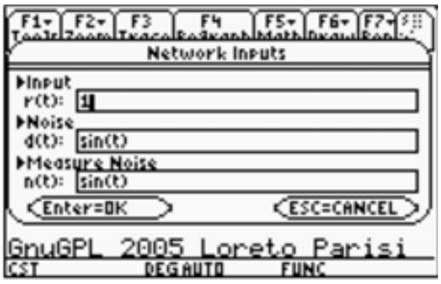
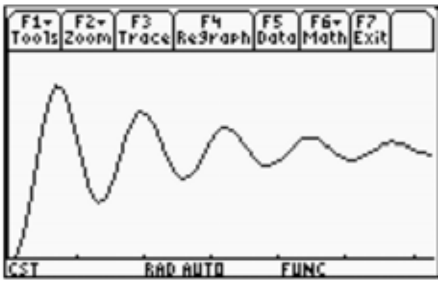
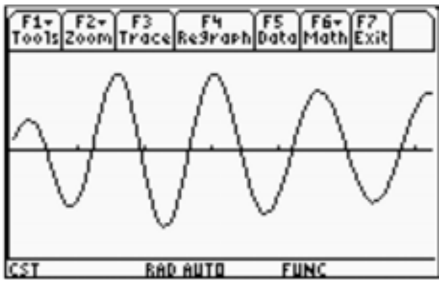
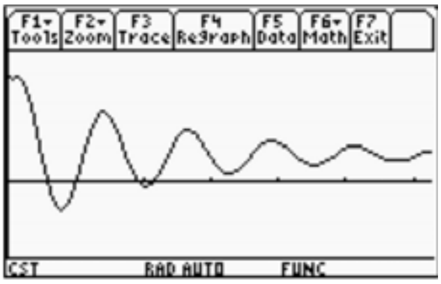
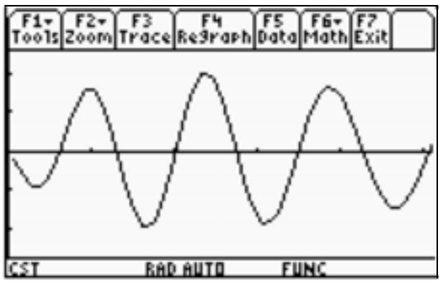
</td>
</tr>
</table>

Examples. 6 – Network Design II

Examples. 7 – Network Tuning I

	Consider a 3rd order transfer function $G(s) = \frac{1}{(1+s)^3}$ We will tune the closed-loop control system using the closed-loop Ziegler-Nichols methodology. Choose *Tuning* from **Controller** menu (F1).
	Select FB (FeedBack) Ziegler-Nichols as *Tuning Method.* Then choose PI (Proportional-Integral) as controller R(s).
	You will obtain the *Critical Mag* Kp' and relating frequency ωπ and period T first. From Zieglier-Nichols tuning rules, then you will get PI parameters, Kp and Ti. Now select *Design* from **Controller** menu (F1) and choose PI. You will see those values filling relating input fields by a kind of magic.
	
	

<table>
<tr>
<td>
</td>
<td>Now we can design the network. Select Design from Network menu(F2). Choose PI as controller R(s). Finally perform a network analysis choosing Analysis from the same menu. We’ll see the closed-loop stability for G(s) by means of values of Mag and Phase margins.</td>
</tr>
<tr>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td>
</td>
<td>In the same way, we can tune a PID controller to assign a custom Gain margin. Choose PID as R(s) and Gain Margin as PID Tuning. Specify a PID’s N parameter for real PIDs (range 5÷20). When asked, insert the requested gain margin. Then procede with PID controller design in Controller menu(F1).</td>
</tr>
<tr>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td>
</td>
<td>
</td>
</tr>
</table>

	Now we will re-define the network with this PID controller and perform a network analysis from **Network** menu (F2). In this case, we have obtained a more robust control network as results from Mag and Phase margin values.
	
	We will analyse the output behaviour of network (with PI controller), defining inputs from **Output** menu (F3). First we will calculate F(s) , S(s) and Q(s) needed by yr(t), yd(t) and ur(t), ud(t) respectly. Please refer to Section *Network Design* for further explanations about inputs.
	
	

Examples. 8 – Network Tuning II

Current Release

- *Control System Toolbox for TI-89*
  Current release: 1.3 October 2005
  Supported Calculator: TI-89 Hardware Version >2.00
  Supported OS: AMS >2.09
  New Features:
  - Symultaneous Continuous and Discrete Time Domain Analysis
  - Time Delay
  - Time Delay's Padè Approximation
  - Phase and Magnitude Margins
  - Routh Criterion and Conditions
  - Backward Eulero, Forward Eulero, Hold Equivalence Discretization
  - Nyquist Diagrams
  - Root Locus
  - Direct and Inverse Laplace Transformations
  - Direct and Inverse Zeta Transformations
  - Feedback Control Systems featuring
  - Design
    - P, PI, PD, PID Controllers
    - Lead, Lag, Lead-Lag Networks
    - Inputs and Noises
  - Analisys
    - Phase and Magnitude Margins
    - Network Transfer Functions
    - Time Domain Outputs
  - Tuning
    - Automatic Tuning featuring
      - Closed Loop Ziegler-Nichols
      - Open Loop Ziegler-Nichols
      - Optimal Control
    - Adaptive Filtering
    - Smith's Predictive Control

- *The CST Start Guide*
  Current version: 1st edition, October 2005
  Distribuition: Portable Document Format

- *The CST Reference Guide*
  Current version: 1st edition, October 2005
  Distribuition: Portable Document Format

- *The CST User Guide*
  Current version: 5th edition, October 2005
  Distribuition: Portable Document Format

Contents

Here are all functions, programs and other objects contained in *cst* folder.

Name	Description	Type
azeros()		Func
band()		Func
Bandn()		Func
Bandsub()		Func
Bodex()		Prgm
c2d()		Func
Check()		Func
Cpoles()		Func
Cst()		Prgm
Cstpi_		Mat
Cstpid_		Mat
Cstver_		Expr
D2c()		Func
Damp()		Func
Db()		Func
Dcgain()		Func
Degroot()		Func
Degzero()		Func
Eigenv()		Func
Error()		Prgm
Feedback()		Prgm
Gain()		Func
Gettd()		Func
Gstep()		Prgm
Help()		Prgm
Install()		Prgm
Linmod()		Func
Linspace()		Func
Logspace()		Func
Mag()		Func
Mag1()		Func
Magz()		Func
Margin()		Func
Nyquist()		Prgm
Pade()		Func
Peak()		Func
Phase()		Func
Phase1()		Func
Phasez()		Func
Poly()		Func
Poly2cof()		Func
Polydeg()		Func
Polyz2s()		Func
Pstep()		Func

Name	**Description**	**Type**
Pzmap()		Func
Rlocdata()		Func
Rloceval()		Func
Rlocus()		Prgm
Roots()		Func
Routh()		Func
Routhc()		Func
Rts2poly()		Func
Sampler()		Func
Spectre()		Func
Splash		Pic
Splhlp		Pic
Ss2tf()		Func
Step()		Func
Tconst()		Func
Tf()		Func
Tf2nd()		Func
Tf2ss()		func
Tmmax()		Func
Trim()		Func
Zoomfit2()		Prgm
Zpk()		Func
Zpkdata()		Func

Removing or modifying one of the objects above could bring *cst* to don't work. Remember that *cst*, and all its contents are released under Gnu Public Licence.

Thanks to…

Many thanks to all those programmers which directly or indirectly gave a hand in making *CST for TI-89*.

The programmers

- *92BROTHERS*
  Contribute: *bodex()*
  E-mail: 92brothers@infinito.it
  Home: http://www.92brothers.net/
- *Francesco Orabona*
  Contribute: *logspace(), poly2cof(), zpk(), nyquist(), rlocus()*
  E-mail: bremen79@infinito.it
  Homepage: http://web.genie.it/utenti/b/bremen79/
- *Lars Frederiksen*
  Contribute: *DiffEq()*
  E-mail: ltf@post8.tele.dk
- *Greg Dietsche*
  Contribute: kerno()
  E-Mail: calc@gregd.org
  Home: http://calc.gregd.org/
- *Kevin Kofler*
  Contribute: hw3patch()
  Home: http://tigcc.ticalc.org.
- *Jiri Bazant*
  Contribute: *lzt()*
  E-mail: georger@razdva.cz
  Home: http://www.razdva.cz/georger/

The Beta Testers

- *Emidio Giordano*, Rome, Italy.

The Users

- *Miroslav Mihalj*
- *John Franklin*
- *Owen Fredericks*
- *Ricardo Vargas*
- *Edgar Salinas*
- *Scott Rogers*
- *James Chizen*
- *Matteo Melotti*
- *Many others…*

And to all those ones who help CST to grow up better and faster!