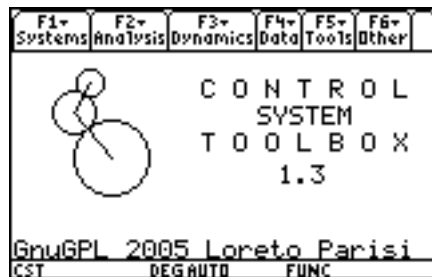


# Control System Toolbox

*for TI-89*



**release 1.3**

## The CST User Guide

*Fifth Edition October 2005*

*Gnu GPL 2002-2005 Loreto Parisi*

<b>Index</b>	<b>Page</b>
About Control System Toolbox for TI-89	7
Disclaimer	8
How to get help	9
How to install	
Install CST	10
Install KerNO	14
Install LZT	15
Fun Index	
<i>Control System Toolbox</i>	
1. System	16
2. Analysis	18
3. Dynamics	21
4. Data	24
5. Tools	26
6. Other	30
7. File	32
<i>Feedback Control Systems</i>	
1. Controller	35
2. Network	40
3. Ouput	43
4. Data	45
5. Tools	45
6. Other	46
Examples	
<i>Control System Toolbox</i>	
1. 1 <sup>st</sup> order LPF	48
2. Linearization	54
3. 3 <sup>rd</sup> order LPF	55
4. 2 <sup>nd</sup> order LPF	57
<i>Feedback Control Systems</i>	
1. Network Design I	60
2. Network Design II	64
3. Network Tuning I	66
4. Network Tuning II	69
Current Release	71
Contents	72
Thanks to...	74

<b>Fun Index (Control System Toolbox)</b>	<b>Page</b>
<b><i>Systems</i></b>	
1. ss2tf(A,B,C,D,iu)	16
2. tf2ss(sys)	16
3. tf(num,den)	16
4. tf(sys)	16
5. zpk(z,p,g)	17
6. c2d(sys,Tc)	17
7. d2c(sys,Tc)	17
<b><i>Analysis</i></b>	
1. poly(A)	18
2. pzmap(sys)	18
3. damp(sys)	18
4. dcgain(sys)	18
5. gain(sys)	19
6. tconst(sys)	19
7. $\tau_{\max}(A)$	19
8. peak(sys)	19
9. margin(sys)	19
10. feedback(sys)	20
<b><i>Dynamics</i></b>	
1. trim(A,B,C,D,u0)	21
2. linmod(f,y,x,u,x0,u0)	21
3. bodex(sys)	21
4. nyquist(sys)	22
5. rlocus(sys)	22
6. step(sys)	22
7. pstep(sys)	22
8. gstep(w_1(t))	23
<b><i>Data</i></b>	
1. $W(\bullet)$	24
2. $W_d(\bullet)$	24
3. $w_1(\bullet)$	24
4. State Space	24
5. $ W(i\omega) $	25
6. $\angle W(i\omega)$	25
7. mag(sys, $\omega_0$ )	25
8. phase(sys, $\omega_0$ )	25
<b><i>Tools</i></b>	

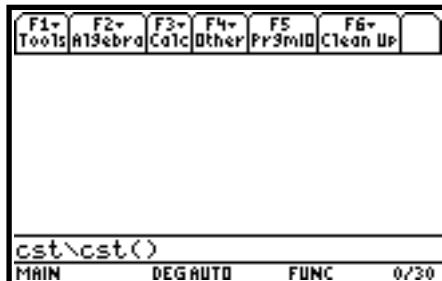
1. cpoles(Cx)	26
2. band(sys)	26
3. polyz2s(Cx)	26
4. routh(Cx)	26
5. routhc(sys,g)	27
6. pade(n, $\tau$ )	27
7. eigenv(A)	27
8. spectre(A)	27
9. sampler(A,B,Tc)	27
10. poly2cof(expr,var)	28
11. rts2poly(roots)	28
12. laplace(f(t))	28
13. ilaplace(F(s))	28
14. zeta(f(k))	28
15. izeta(F(z))	29
<b><i>Other</i></b>	
1. Quick Load	30
2. Quick Save	30
3. File...	30
4. Settings	31
5. Help	31
6. About	31
7. Exit	31
<b><i>File Load</i></b>	
1. Session	32
2. State Space	32
3. Transfer Function	32
4. Step Response	33
5. Bode diagram	33
<b><i>File Save</i></b>	
1. Session	33
2. State Space	33
3. Transfer Function	34
4. Step Response	34

<b>Fun Index (Feedback Control Systems)</b>		<b>Page</b>
<b><i>Controller</i></b>		
1. Design...		35
a. Custom Network		35
b. P Controller		35
c. PI Controller		35
d. PD Controller		36
e. PID Controller		36
<i>f. Direct Design</i>		36
i. Lead Network		36
ii. Lag Network		36
iii. Lead-Lag Network		37
<i>g. Nichols Design</i>		37
i. Lead Network		37
ii. Lag Network		37
iii. Lead-Lag Network		37
2. Tuning...		38
a. Feedback Ziegler-Nichols		38
b. Feedforward Ziegler-Nichols		38
c. Optimal Control		38
d. Smith Predictive Control		38
e. Adaptive Filtering		39
3. Custom		39
4. P		39
5. PI		39
6. PD		39
7. PID		39
8. Lead Network		39
9. Lag Network		39
10. Lead-Lag Network		39
<b><i>Network</i></b>		
1. Design...		40
2. Analysis...		42
3. $G(s)$ ...		41
4. $R(s)$ ...		41
5. $L(s)$		41
6. $F(s)$		41
7. $S(s)$		41
8. $Q(s)$		41
9. $M(s)$		41
<b><i>Data</i></b>		
1. Margins		45

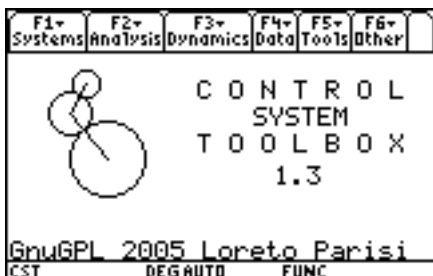
<b><i>Tools</i></b>	
1. Choose SYS...	45
2. pzmap(SYS)	45
3. damp(SYS)	45
4. gain(SYS)	45
5. pade(n, $\tau$ )	45
6. routh(Cx)	45
7. routhc(SYS,v)	45
8. rlocus(SYS)	45
9. nyquist(SYS)	45
10. bodex(SYS)	45
11. mag(SYS, $\omega_0$ )	45
12. phase(SYS, $\omega_0$ )	45
<b><i>Output</i></b>	
1. Input...	43
2. yr(t)	43
3. yd(t)	43
4. yn(t)	43
5. ur(t)	43
6. ud(t)	43
7. un(t)	43
8. er(t)	43
9. ed(t)	43
10.en(t)	43
<b><i>Other</i></b>	
1. Quick Load	46
2. Quick Save	46
3. Help	46
4. About	47
5. Exit	47

## About Control System Toolbox for TI-89

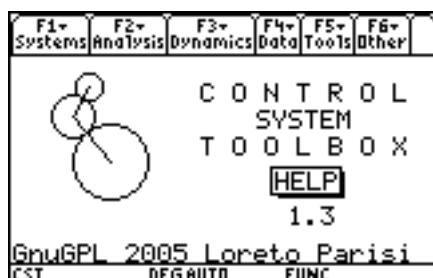
Control System Toolbox (CST) for TI-89 is a suite of specialized functions and programs for Systems Control and Tuning created by **Loreto Parisi** from June 2002 for the TI-89 personal calculator.



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**

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but **WITHOUT ANY WARRANTY**; without even the implied warranty of **MERCHANTABILITY** or **FITNESS FOR A PARTICULAR PURPOSE**. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

### **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](mailto:loreto_parisi@yahoo.it)

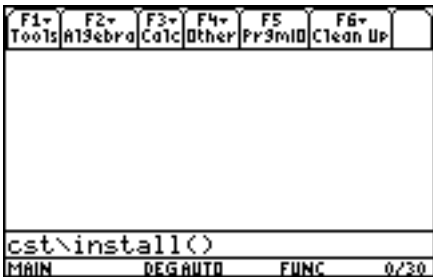



CST Home: <http://www.webalice.it/loretoparisi/>






- *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**.

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

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

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

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

### 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 recommend to read the LZT readme file.

LZT r7

Author: Jiri Bazant

Email: [georger@razdva.cz](mailto: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 recommend to read the KerNo readme file.

KerNO r3.1

Author: Greg Dietsche

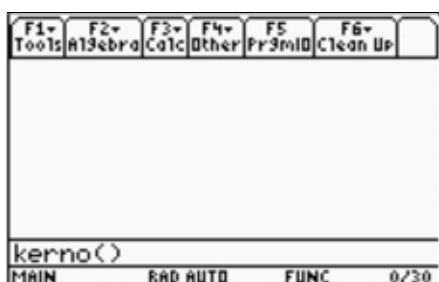
E-Mail: <mailto:calc@gregd.org>

Home: <http://calc.gregd.org/>

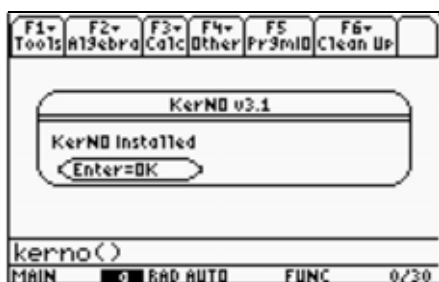
**Install KerNO**

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>.




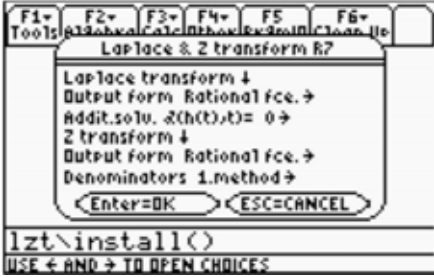

Now, we can install the kernel. Transfer KerNo to TI-89, then simply run it from *main*.



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/>

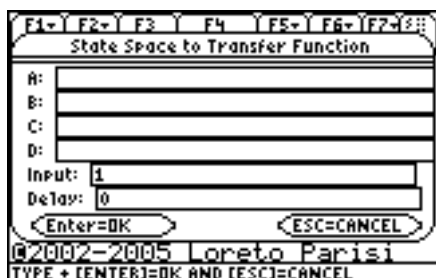
**Install LZT**

	<p>After installing a kernel, we can install LZT release 7 (current).</p>
	<p>Send <i>lztR7.89g</i> to TI-89 and run <i>install</i> from <i>lzt</i> folder.</p>
	<p>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.</p>
	<p>Now we will choose Archive to improve performances of <i>lzt</i> and to save space in RAM memory, archiving <i>lzt</i> in Flash ROM memory.</p>
	<p>LZT r7 Author: Jiri Bazant Email: <a href="mailto:georger@razdva.cz">georger@razdva.cz</a> Home: <a href="http://www.razdva.cz/georger/">http://www.razdva.cz/georger/</a></p>

## Systems

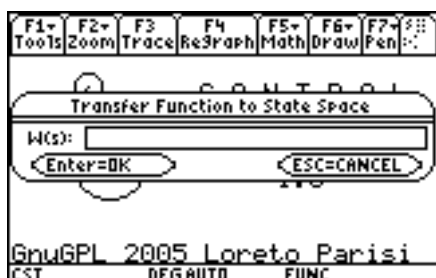


The *Systems* menu (F1) contains all the functions to build the model, using state-space or transfer function and to perform conversions from one representation to another, even from continuous time to discrete time model.

***ss2tf(A,B,C,D,iu)***

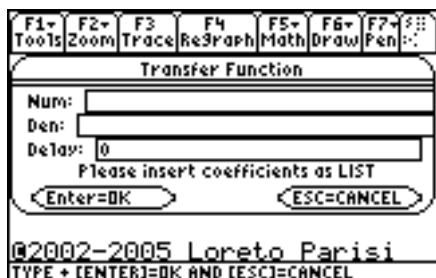
Gives transfer function  $W(s)=C(sI-A)^{-1}B+D$  from state-space  $\dot{x} = Ax + Bu$ ,  $y = Cx + Du$ , relating to input  $iu$  (it works on MIMO systems, but only one input at time).

Delay  $\tau$  is the Time Delay  $e^{-\tau s}$ .

***tf2ss(SYS)***

Convert transfer function  $W(s)$  in the state-space representation  $\dot{x} = Ax + Bu$ ,  $y = Cx + Du$ , using the observability canonical form.

Delay  $\tau$  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={ $b_0, b_1, \dots, b_n$ }, DEN={ $a_0, a_1, \dots, a_n$ }, so

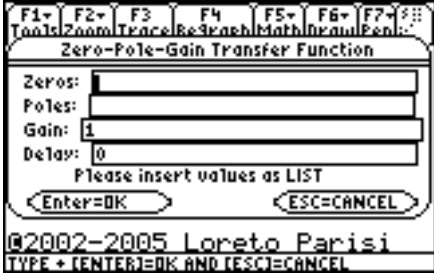
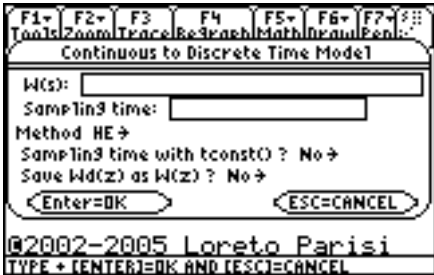

$$W(s) = \frac{b_0 s^n + b_1 s^{n-1} + \dots + b_n}{a_0 s^n + a_1 s^{n-1} + \dots + a_n}$$

Delay  $\tau$  is the Time Delay  $e^{-\tau s}$ .

***tf(SYS)***

Calculates transfer function from a rational expression in  $s$

Delay  $\tau$  is the Time Delay  $e^{-\tau s}$ .

	<p><b><i>zpk(Z,P,G)</i></b> Calculates transfer function <math>W(s)</math> 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.</p> <p>Control's Toolbox v.1.16 author: Francesco Orabona E-mail: <a href="mailto:bremen79@infinito.it">bremen79@infinito.it</a> Home: <a href="http://web.genie.it/utenti/b/bremen79/">http://web.genie.it/utenti/b/bremen79/</a></p>
	<p><b><i>c2d(SYS,T<sub>c</sub>)</i></b> Converts continuous time model SYS to the discrete time model, using sample time <math>T_c</math> and different methods: HE (Linear Hold Equivalence), TU (Bilinear Tustin), BE (Bilinear Backward Eulero), FE (Bilinear Forward Eulero). Can use function <code>tconst(SYS)</code> to determinate sample time <math>T_c</math>. Use function <code>sampler(A,B,T<sub>c</sub>)</code> to use ZOH method. Can save the resulting discrete time model <math>W_d(z)</math> as current discrete transfer function <math>W(z)</math>.</p>
	<p><b><i>d2c(SYS,T<sub>c</sub>)</i></b> Converts discrete time model SYS (in z) to the continuous time model, using sample time <math>T_c</math> and different methods: HE (Linear Hold Equivalence), TU (Bilinear Tustin), BE (Bilinear Backward Eulero), FE (Bilinear Forward Eulero). Can use function <code>tconst(SYS)</code> to determinate sample time <math>T_c</math>. Can save the resulting continuous time model <math>W_d(s)</math> as current continuous transfer function <math>W(s)</math>.</p>

## 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  $|\cdot|$  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.



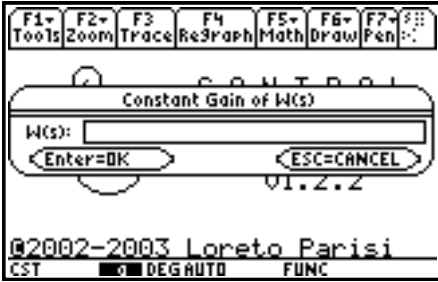




### **damp(SYS)**


Calculate natural frequencies  $\omega_{nh}$  and damping factors  $\zeta_h$  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  $G = \lim_{s \rightarrow 0} W(s)$ .

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

	<p><b><i>feedback(sys)</i></b> Performs the analysis and design of the closed loop control system of process SYS. Delay <math>\tau</math> is the Time Delay <math>e^{-\tau s}</math>. Please see <i>Feedback Control Systems</i> section.</p>
---	---

## 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<sub>0</sub>)*

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,x0,u0)*

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



### *bodex(SYS)*

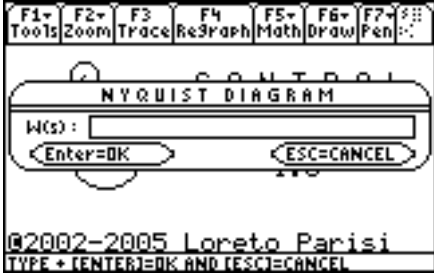
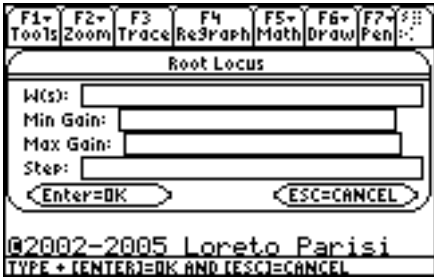


This program, made by 92BROTHERS, plots Bode diagrams of phase and magnitude and offers several tools to work with the plotted diagrams.


BodeX v.2.2.3

Copyright © 2000 92BROTHERS

Email: [92brothers@infinito.it](mailto:92brothers@infinito.it)

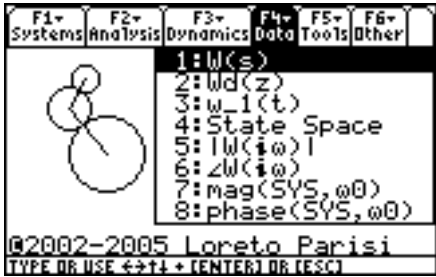
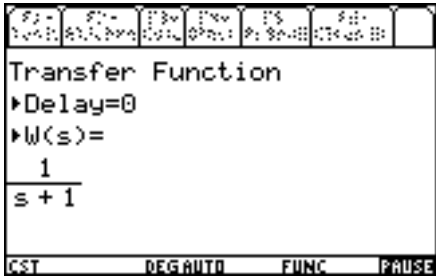
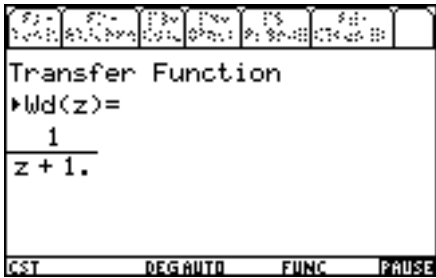
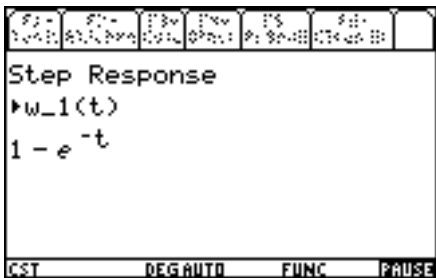
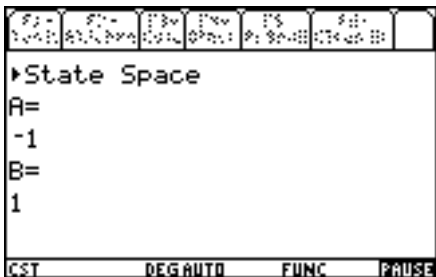
Home: <http://www.92brothers.net/>

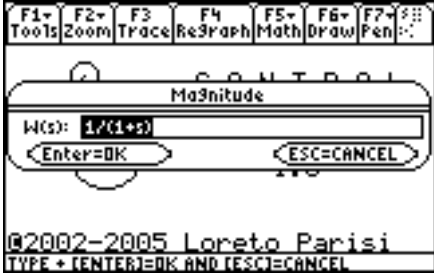
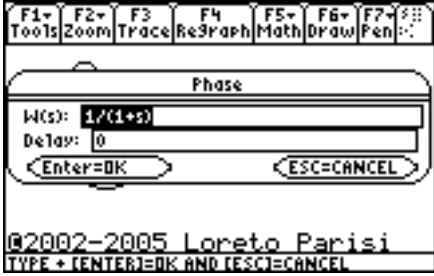


	<p><b><i>nyquist(SYS)</i></b> Plots Nyquist diagram of SYS</p> <p>Control Toolbox v1.16 Author: Francesco Orabona E-mail: <a href="mailto:bremen79@infinito.it">bremen79@infinito.it</a> Home: <a href="http://web.genie.it/utenti/b/bremen79/">http://web.genie.it/utenti/b/bremen79/</a></p>
	<p><b><i>rlocus(SYS)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

	<p><b><i>gstep(w_1(t))</i></b>          This tool plots the step response <math>w_1(t)</math> calculated with <code>step(SYS)</code> or specified directly. Can use <code>pstep(SYS)</code> to evaluate <math>w_1(t)</math> around its typical parameters.</p>
---	--

**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.

<b>Data</b>	
	<p>The <b>Data</b> menu (F4) gives access to current transfer function <math>W(\bullet)</math><sup>1</sup>, its discrete model <math>Wd(\bullet)</math>, the step response <math>w_1(\bullet)</math>, the current state space and magnitude and phase of <math>W(\bullet)</math>.</p> <p><sup>1</sup> According to current Time Domain Settings (see Other menu)</p>
	<p><b><math>W(s)</math> [<math>W(z)</math>]</b> 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.</p>
	<p><b><math>Wd(z)</math> [<math>Wd(s)</math>]</b> The discrete transfer function <math>Wd(z)</math> [<math>Wd(s)</math>], obtained by <math>c2d(SYS, Tc)</math> [<math>d2c(SYS, Tc)</math>] using the current transfer function <math>W(s)</math> [<math>W(z)</math>] or by <math>sampler(A, B, Tc)</math> using the current state space.</p>
	<p><b><math>w_1(t)</math> [<math>w_1(k)</math>]</b> Shows the current step response obtained with <math>step(SYS)</math>.</p>
	<p><b>State Space</b> It displays the current state space, as defined from one of tools of System menu.</p>

	<p><math> W(\omega) </math></p> <p>Displays magnitude of transfer function <math>W(\bullet)</math> in Laplace domain ( for <math>W(s)</math> ) and even in domain Z (for <math>W(z)</math> ).</p>
	<p><math>\angle W(\omega)</math></p> <p>Displays phase of transfer function <math>W(\bullet)</math> in Laplace domain (for <math>W(s)</math> ) and even in domain Z (for <math>W(z)</math> ).</p>
	<p><math>mag(SYS, \omega_0)</math></p> <p>Calculates magnitude of SYS in Laplace domain ( for <math>W(s)</math>) and even in domain Z (for <math>W(z)</math> ), relating to <math>\omega_0</math>.</p>
	<p><math>phase(SYS, \omega_0)</math></p> <p>Calculates phase of SYS in Laplace domain ( for <math>W(s)</math>) and even in domain Z (for <math>W(z)</math> ), relating to <math>\omega_0</math>.</p>

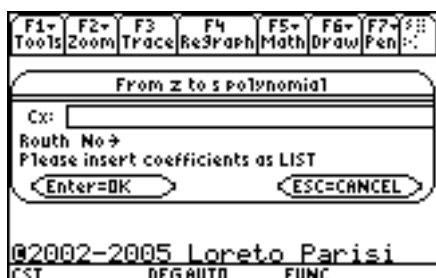
## Tools



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.

***cpoles(Cx)***

It calculates zeros of polinomyal given as LIST of coefficients, Cx.

***polyz2s(Cx)***

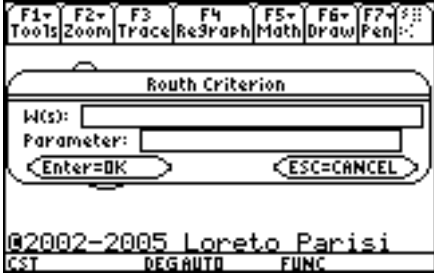
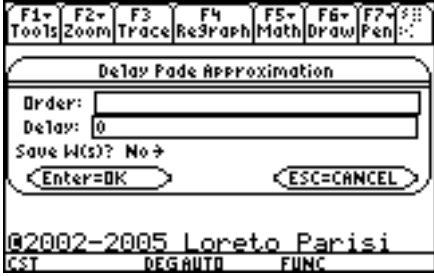

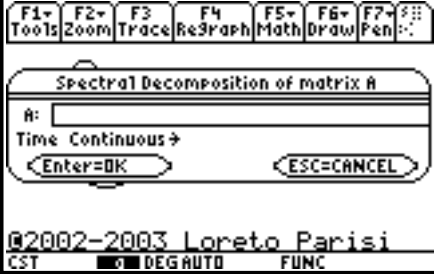

This tool calculates the continuous polynomial  $q(s)$ , relating to discrete polynomial  $p(z)$ , assigned in terms of its coefficients LIST, Cx, using the formula




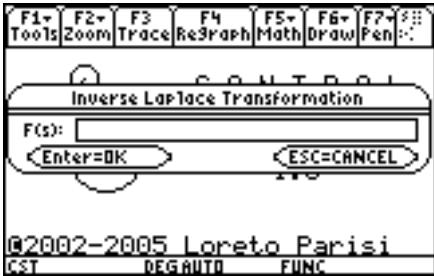

$$q(s) = (s-1)^n p(z) \Big|_{z=\frac{s+1}{s-1}}$$
***band(SYS)***


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).

***routh(Cx)***

It calculates the Routh matrix for polynomial assigned with its coefficients LIST, Cx.

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

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

	<p><i>izeta(F(z))</i> Performs Inverse Zeta Transformation of F(z). Needs the tool LZT to perform symbolic calculation.</p> <p>LZT r7 Author: Jiri Bazant Email: <a href="mailto:georger@razdva.cz">georger@razdva.cz</a> Home: <a href="http://www.razdva.cz/georger/">http://www.razdva.cz/georger/</a></p>
---	---

**Notes.**<sup>1</sup> About *spectre(A)*

The spectral decomposition of matrix A is

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

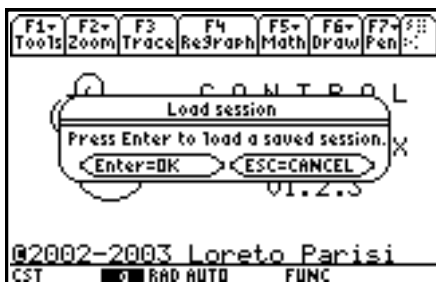
$$A^k = \sum_{i=1}^{\mu} u_i \lambda_i^k v_i^T + \sum_{h=1}^{\nu} (u_{ha} \quad u_{hb}) \rho_h^k \begin{pmatrix} \cos \theta_h k & \sin \theta_h k \\ -\sin \theta_h k & \cos \theta_h k \end{pmatrix} \begin{pmatrix} v_{ha}^T \\ v_{hb}^T \end{pmatrix} \quad (\text{discrete})$$

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

## Other

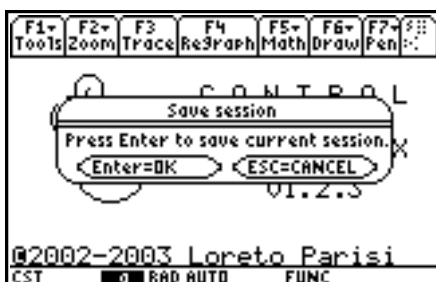


The *Other* menu (F6) gives tools to manage files, the current working session, the Settings, to access to on-line help tool with help(), some information about CST, and the way to exit CST.



### *Quick Load*

Loads the current working session (i.e. transfer functions  $W(s)$  and  $W(z)$ , State space,  $w_1(t)$ ,  $T_c$ , step response parameters, etc.) previously saved. It overwrites all the existing values for the current session. Be careful.



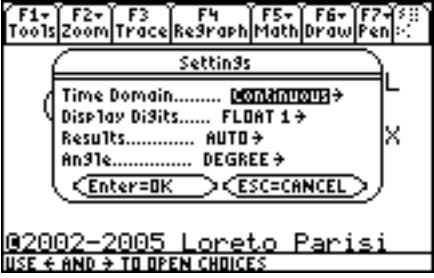
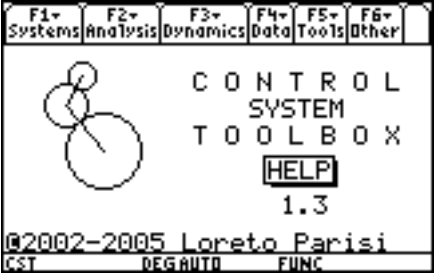


### *Quick Save*

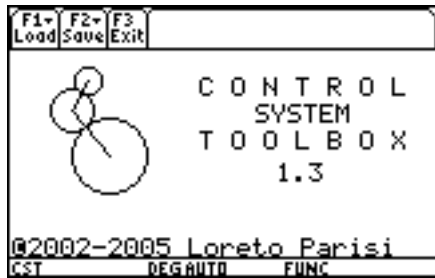
Saves the current working session (i.e. transfer functions  $W(s)$  and  $W(z)$ , State space,  $w_1(t)$ ,  $T_c$ , step response parameters, etc.).



### *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.

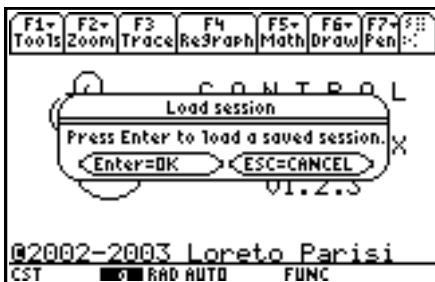
	<p><b>Settings</b></p> <p>It permits to modify some settings of the calculator, such as the display digits, the angle, the format of results and to switch the current Time Domain: Continuous to work with continuous time model <math>W(s)</math> or Discrete to work with discrete time model <math>W(z)</math>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

**File**

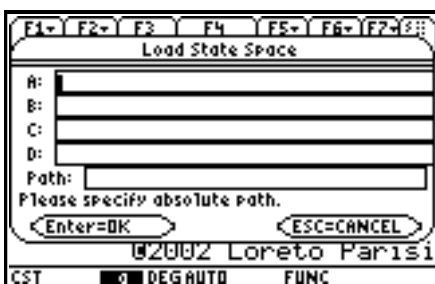
The **File** toolbox gives access to the File & Session Management. Here you can load and save the current working session, the State Space, the Transfer Function, the Step Response and bode diagram obtained with `bodex(SYS)`. There are three menus Load, Save and Exit. Exit menu (F3) brings to the previous toolbox.

**Load**

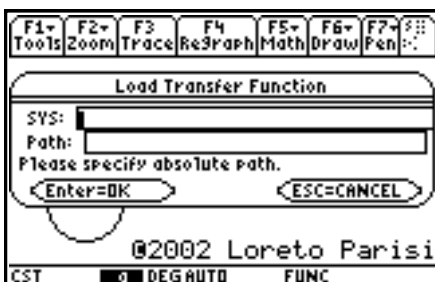
The Load menu (F1) permits to load the current working session, the State Space, Transfer Function, Step Response and bode diagrams from the specified path.

**Load Session**

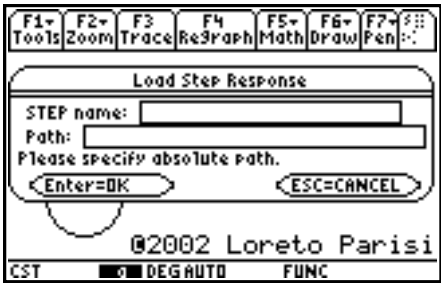


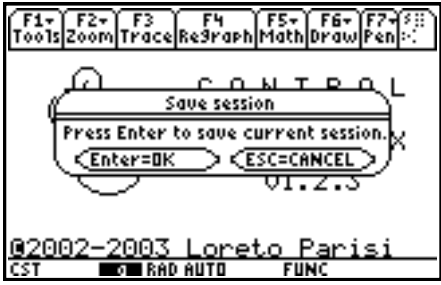
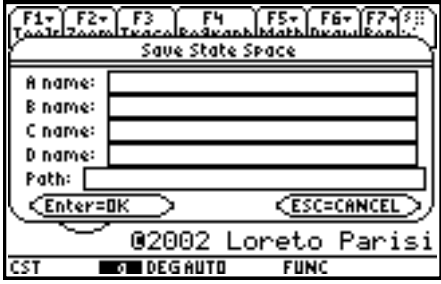
Loads the current working session (i.e. transfer functions  $W(s)$  and  $W(z)$ , State space,  $w_1(t)$ ,  $T_c$ , step response parameters, etc.) previously saved. It overwrites all the existing values for the current session. Be careful.

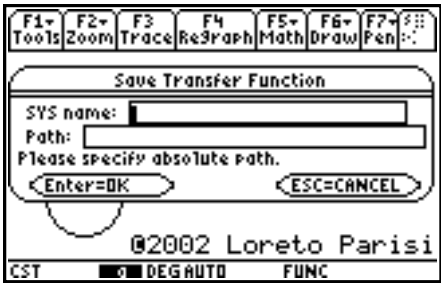

**Load State Space**

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.

	<p><b>Load Step Response</b> Permits to load the Step Response from specified path.</p>
	<p><b>Load Bode diagram</b> 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 <math>\omega_0</math>.</p>
	<p><b>Save</b> The Save menu (F2) permits to save the current working session, the State Space, Transfer Function, Step Response into a specified path.</p>
	<p><b>Save session</b> Saves the current working session (i.e. transfer functions <math>W(s)</math> and <math>W(z)</math>, State space, <math>w_1(t)</math>, <math>T_c</math>, step response parameters, etc.).</p>
	<p><b>Save State Space.</b> Permits to save current State Space into the specified path, using given names.</p>

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

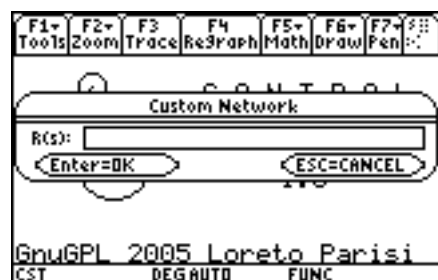
## Controller



The **Controller** menu (F1) is intended to design and tuning the control system.



The **Controller Design** wizard will guide you through 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  $R(s) = K_p$ .





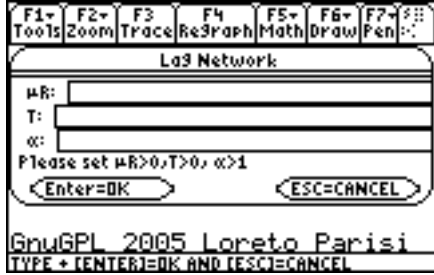


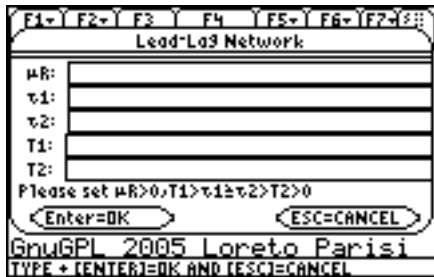
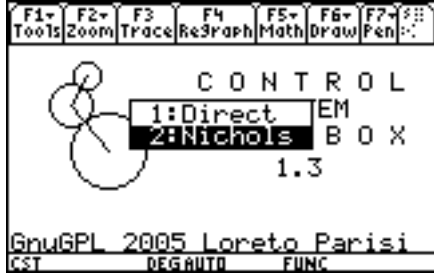
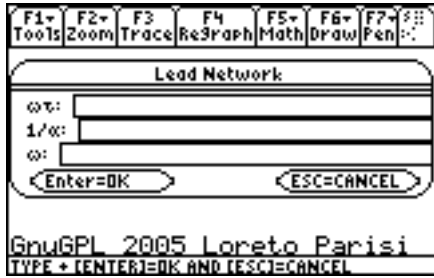
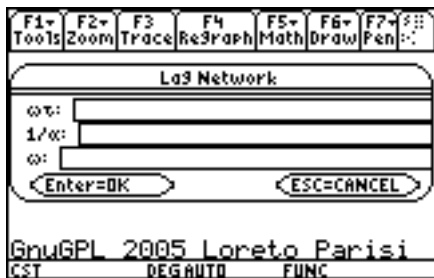
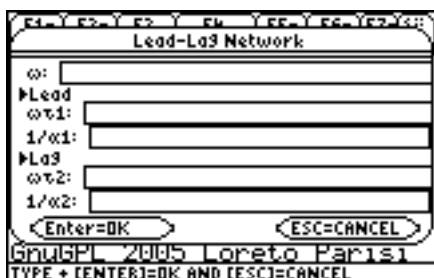
### PI Controller

Defines a PI controller R(s) as you give  $K_p$  and  $K_i$  or  $K_p$  and  $T_i$ :

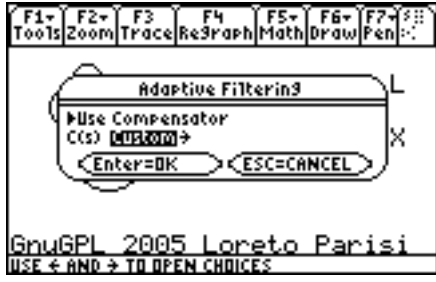
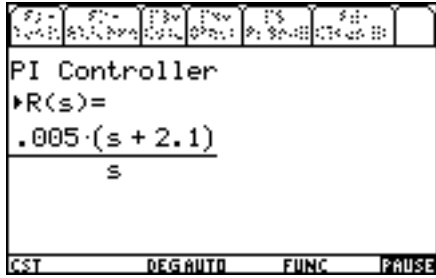
$$R_{PI}(s) = \frac{K_p s + K_i}{s} = K_p \frac{1 + T_i s}{T_i s}$$

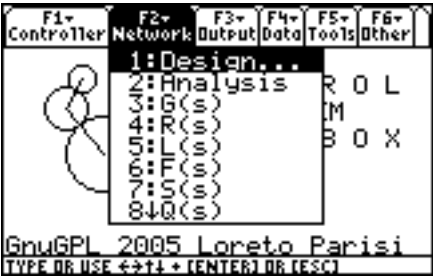
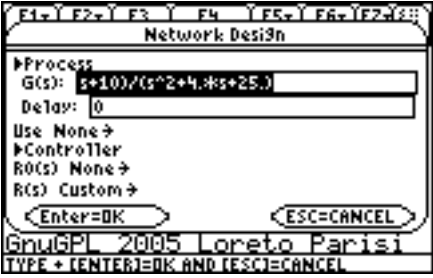



$$\text{where } T_i = \frac{K_p}{K_i}$$

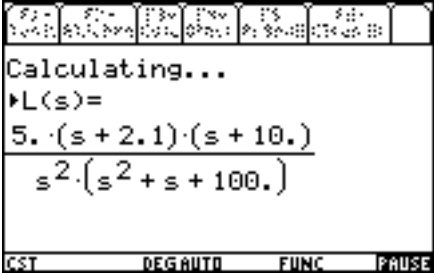
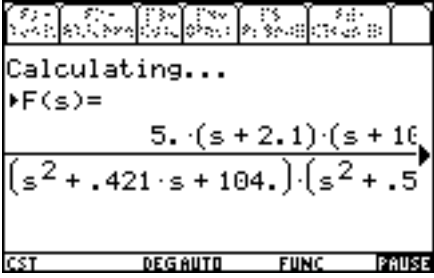
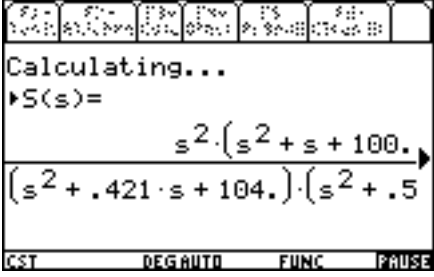
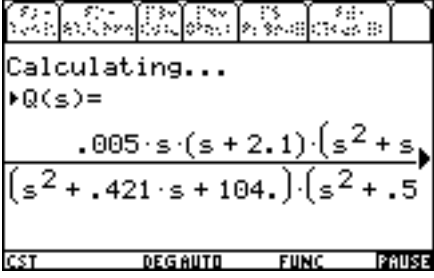
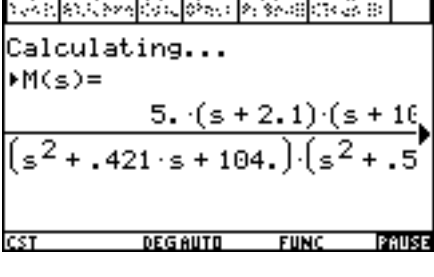
	<p><b>PD Controller</b>                  Defines a PD controller <math>R(s)</math> as you give <math>K_p</math> and <math>K_d</math> or <math>K_p</math> and <math>T_d</math>:</p> $R_{PD}(s) = K_p + K_D s = K_p (1 + T_D s)$ <p>where <math>T_D = \frac{K_D}{K_p}</math></p>
	<p><b>PID Controller</b>                  Defines a standard PID controller as you give <math>K_p</math>, <math>K_i</math>, <math>K_d</math> or <math>K_p</math>, <math>T_i</math>, <math>T_d</math>:</p> $R_{PID}(s) = \frac{K_D s^2 + K_P s + K_I}{s} = K_P \frac{T_I T_D s^2 + T_I s + 1}{T_I s}$ <p>or a real PID controller specifying <math>N</math>:</p> $R_{PID}(s) = K_p + \frac{K_I}{s} + \frac{K_D s}{1 + \frac{K_D}{K_p N} s} = K_p \left( 1 + \frac{1}{T_I s} + \frac{T_D}{1 + \frac{T_D}{N} s} s \right)$
	<p><b>Direct Design</b>                  Defines a Lead, Lag or Lead-lag network directly from transfer function's gain <math>\mu_R</math>, time constant <math>T</math> and <math>\alpha</math> parameter.</p>
	<p><b>Lead Network</b>                  Defines a lead network <math>R(s)</math> as you give the gain <math>\mu_R</math>, time constant <math>T</math> and parameter <math>\alpha</math>:</p> $R(s) = \mu_R \frac{1 + Ts}{1 + \alpha Ts}$ <p>Must be:  <math>\mu_R &gt; 0, T &gt; 0, 0 &lt; \alpha &lt; 1</math></p> <p>Usually, <math>\alpha = 0.1</math> and <math>T = \frac{1}{\omega_c}</math></p>
	<p><b>Lag Network</b>                  Defines a lag network <math>R(s)</math> as you give the gain <math>\mu_R</math>, time constant <math>T</math> and parameter <math>\alpha</math>:</p> $R(s) = \mu_R \frac{1 + Ts}{1 + \alpha Ts}$ <p>Must be:  <math>\mu_R &gt; 0, T &gt; 0, \alpha &gt; 1</math></p> <p>Usually, <math>T &gt; \frac{1}{\omega_c}</math> (<math>T = \frac{10}{\omega_c}</math>)</p>

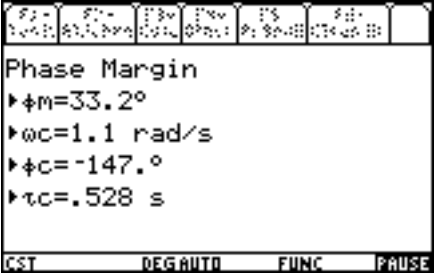
	<p><b>Lead-Lag Network</b>                  Defines a lead-lag network <math>R(s)</math> as you give the gain <math>\mu_R</math>, and time constants <math>\tau_1, \tau_2, T_1, T_2</math>:</p> $R(s) = \mu_R \frac{(1 + \tau_1 s)(1 + \tau_2 s)}{(1 + T_1 s)(1 + T_2 s)}$ <p>Must be:  <math>\mu_R &gt; 0, T_1 &gt; \tau_1 \geq \tau_2 &gt; T_2 &gt; 0</math></p> <p>Usually, <math>T_1 T_2 = \tau_1 \tau_2, \tau_2 &gt; \frac{1}{\omega_C} &gt; T_2</math></p>
	<p><b>Nichols Design</b>                  Defines a Lead, Lag or Lead-lag network using Nichols's standard networks parameters <math>\omega\tau, 1/\alpha</math> at <math>\omega_0</math>. In most cases <math>\omega_0</math> will be the critical frequency <math>\omega_C</math>.</p>
	<p><b>Lead Network</b>                  Defines a Lead network using Nichols's standard networks parameters <math>\omega\tau, 1/\alpha</math> at <math>\omega_0</math>.</p> $R(s) = \frac{1 + s\tau}{1 + s\alpha\tau}$
	<p><b>Lag Network</b>                  Defines a Lag network using Nichols's standard networks parameters <math>\omega\tau, 1/\alpha</math> at <math>\omega_0</math>.</p> $R(s) = \frac{1 + s\alpha\tau}{1 + s\tau}$
	<p><b>Lead-Lag Network</b>                  Defines a Lead-Lag network using Nichols's standard networks parameters <math>\omega\tau_1, 1/\alpha_1</math> for the lead and <math>\omega\tau_2, 1/\alpha_2</math> for the lag network at frequency <math>\omega_0</math>.</p> $R(s) = \frac{1 + s\tau_1}{1 + s\alpha_1\tau_1} \frac{1 + s\alpha_2\tau_2}{1 + s\tau_2}$

	<p><b>Controller Tuning</b></p> <p>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 available: <i>Feedback Ziegler-Nichols</i>, <i>Feedforward Ziegler-Nichols</i>, <i>Optimal Control</i>, <i>Predictive Control</i> and <i>Adaptive Filtering</i>.</p>
	<p><b>Feedback Ziegler-Nichols</b></p> <p>Uses the <i>Closed Loop Ziegler-Nichols</i> method to tune the controller for the feedback network. Choose the desired 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.</p>
	<p><b>Feedforward Ziegler-Nichols</b></p> <p>Uses the <i>Open Loop Ziegler-Nichols</i> method to tune the controller for the approximate process (obtained from the step response using the <i>areas method</i>)</p> $G_a(s) = \frac{\mu}{1 + Ts} e^{-\tau s}$ <p>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).</p>
	<p><b>Optimal Control</b></p> <p>Uses optimization methods to tune the controller R(s): ISE (<i>Integral Square Error</i>), ISTE (<i>Integral Square Time Error</i>) and IST<sup>2</sup>E (<i>Integral Square Time<sup>2</sup> Error</i>). K<sub>p</sub>, T<sub>i</sub> and T<sub>d</sub> are defined by a table as follows:</p> $K_p = \frac{a_1}{\mu} \theta^{b_1}, T_i = \frac{T}{a_2 + b_2 \theta}, T_d = a_3 T \theta^{b_3}$
	<p><b>Smith Predictive Control</b></p> <p>Used to tune network's controllers for processes with positive real zeros or time delays. The process G(s) is given as <math>G(s) = \frac{N^-(s)N^+(s)}{D(s)} e^{-\tau s}</math>. The predictor P(s) and the network transfer function L'(s) for the given controller R(s) are <math>P(s) = \left(1 - \frac{N^+(s)}{N^+(-s)} e^{-\tau s}\right) \frac{N^-(s)N^+(-s)}{D(s)}</math> and <math>L'(s) = (G(s) + P(s))R(s)</math>.</p>

	<p><b>Adaptive Filtering</b></p> <p>Uses a pre-filtering technique (compensation of input signal) to improve static and dynamic behaviour. You have to choose the structure for the compensator <math>C(s)</math>. We suppose you have defined it as a controller (custom, lead, lag or lead-lag) yet.</p>
	<p><b>Custom, P, PI, PD, PID, Lead, Lag, Lead-Lag</b></p> <p>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 <math>R(s)</math>.</p>

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

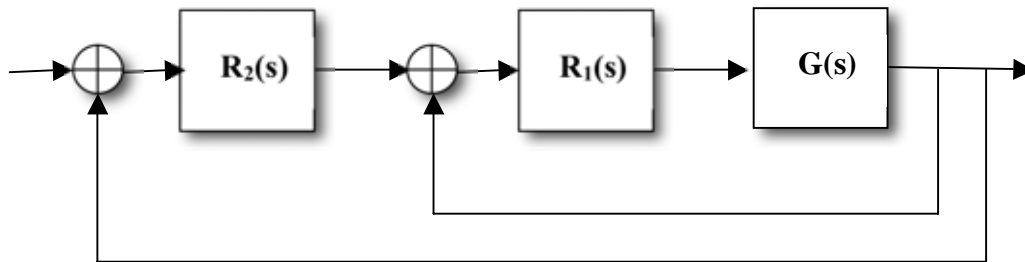
	<p><math>L(s)</math> The network transfer function</p> $L(s) = R(s)G(s)$
	<p><math>F(s)</math> The network transfer function</p> $F(s) = \frac{R(s)G(s)}{1 + R(s)G(s)}$
	<p><math>S(s)</math> The network transfer function</p> $S(s) = \frac{1}{1 + R(s)G(s)}$
	<p><math>Q(s)</math> The network transfer function</p> $Q(s) = \frac{R(s)}{1 + R(s)G(s)} = F(s)G(s)^{-1} = R(s)S(s)$
	<p><math>M(s)</math> The network transfer function</p> $M(s) = G(s)S(s)$

	<p><b>Analysis</b> Performs an analysis of the defined control system, calculating the Gain and the Phase margins.</p>
---	--

### Notes.

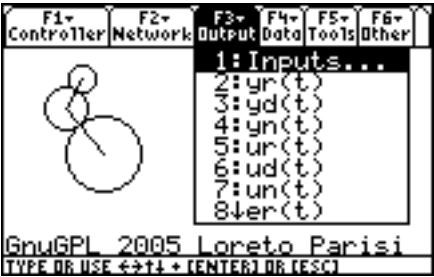
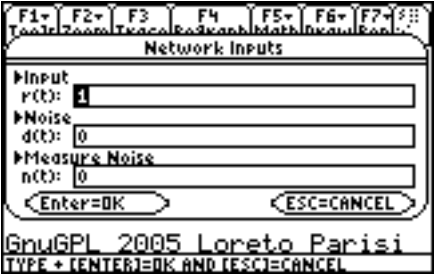
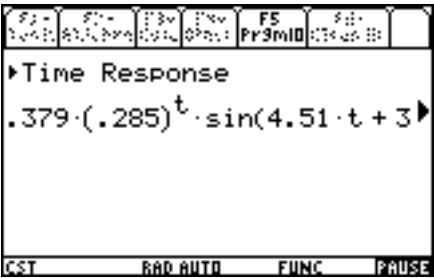
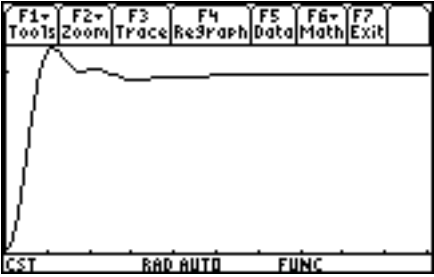
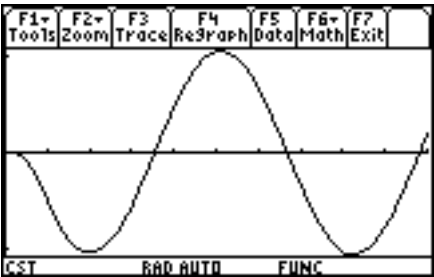
About *Network Design*.

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

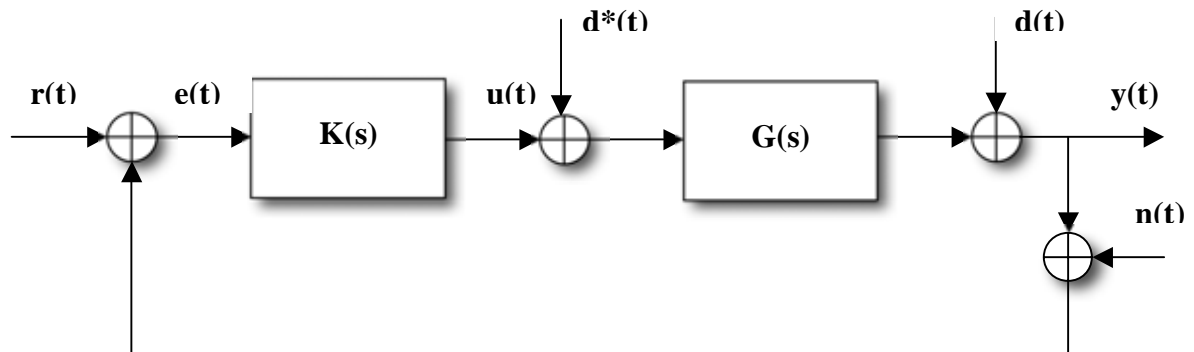
given requirements. To do so, first design  $R_1(s)$  as usual. When the inner closed loop is stable, you can design  $R_2(s)$  choosing in *Network Design Use F(s)* as new  $G(s)$  from the drop down menu.

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

**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

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

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

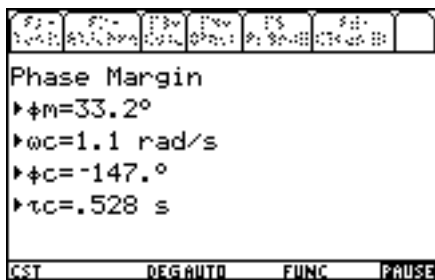
$$Q(s) = \frac{K(s)}{1 + K(s)G(s)},$$

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.

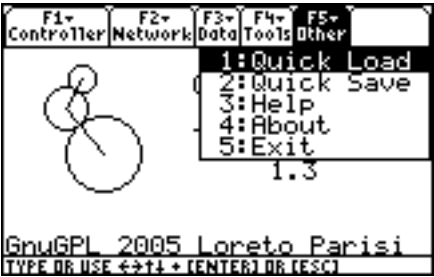
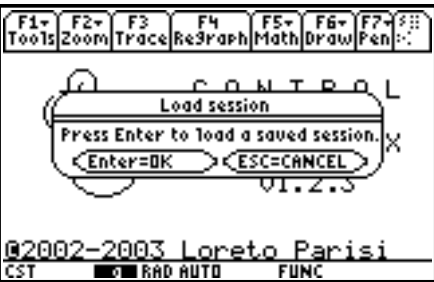
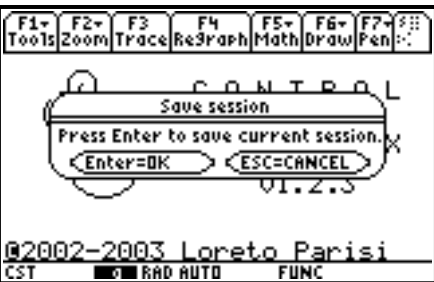



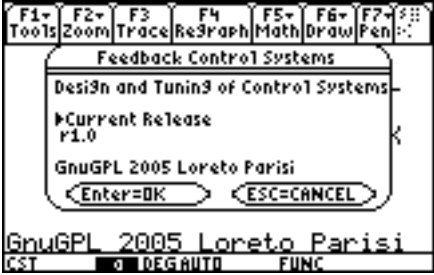
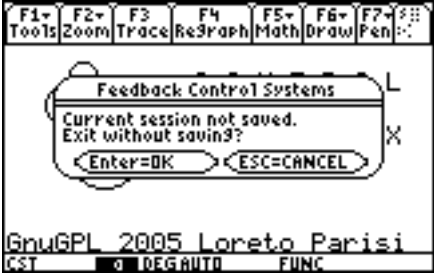
*Margins*  
Shows the Gain and Phase margins of the control system.

**Tools**



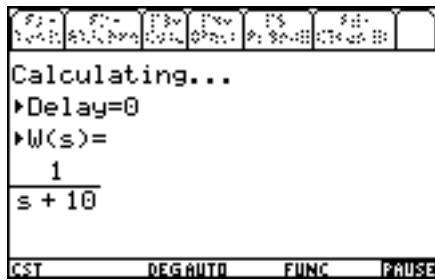
The *Tools* menu (F4) permits you to perform a detailed analysis of transfer function SYS. You need to choose the current transfer function SYS first.

<b>Other</b>	
	<p>The <i>Other</i> menu (F3) shows some help, info and permits to exit the program.</p>
	<p><b>Quick Load</b> 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.</p>
	<p><b>Quick Save</b> Saves the current working session (i.e. transfer functions of process, network and controllers, calculated margins, etc.).</p>
	<p><b>Help</b> Shows some help.</p>

	<p><b>About</b> Shows some info.</p>
	<p><b>Exit</b> Exit the program. Unsaved session will be lost.</p>

**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`.



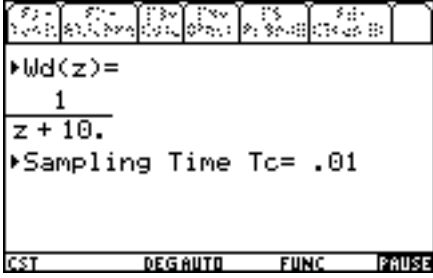
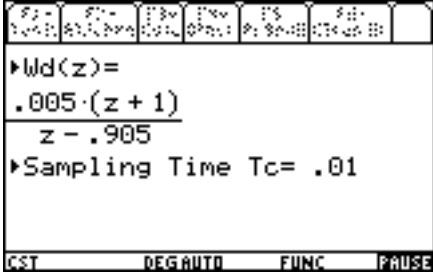

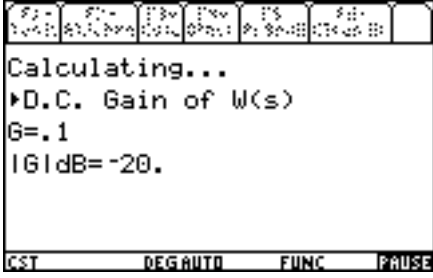

To get relating state-space, now we'll use function `tf2ss(SYS)`, where `SYS` is the current transfer function, automatically filled in the input field.

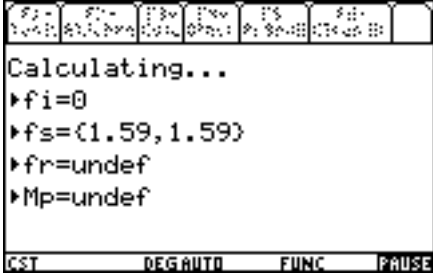


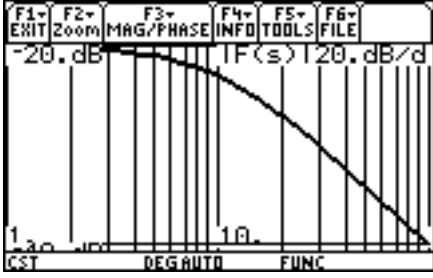
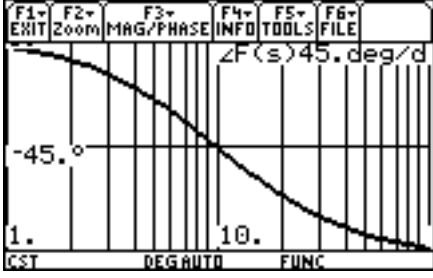



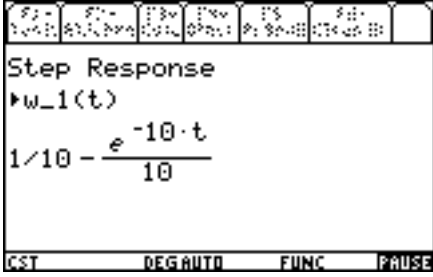

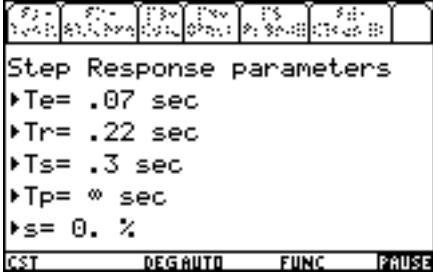
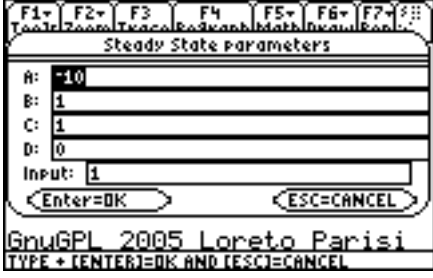
After calculation of all matrixes, they will be the current state space.









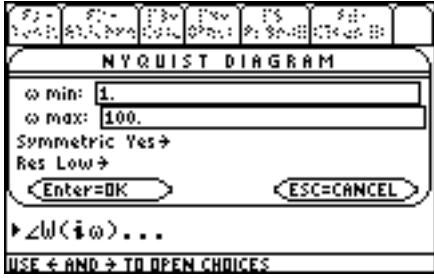
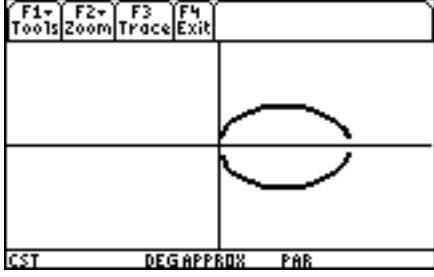

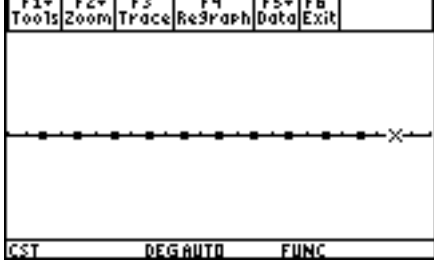
If you need to obtain the discrete model of `SYS`, use `c2d(SYS,Tc)` where `Tc` is the sample time desired. We'll use `tconst(SYS)` to get `Tc` and 'HE' (Hold Equivalence) transformation as method. We could save resulting `Wd(z)` as the current discrete transfer function `W(z)`.

 <pre> ▶Wd(z)=   1   z + 10. ▶Sampling Time Tc= .01 </pre>	<p>Here is the discrete time model with sample time Tc from tconst(SYS) and 'HE' (Hold Equivalence) method.</p>
 <pre> ▶Wd(z)= .005 * (z + 1) z - .905 ▶Sampling Time Tc= .01 </pre>	<p>This is the discrete time model with sample time Tc from tconst(SYS) and 'Tustin' method.</p>
 <pre> Calculating... ▶Poles (-1/τ) (-10.) ▶Zeros (-1/τ') () </pre>	<p>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.</p>
 <pre> Calculating... ▶D.C. Gain of W(s) G=.1  G dB=-20. </pre>	<p>With function dcgain(SYS) we have calculated the d.c. gain of SYS, which results -20 dB (0.1 linear).</p>
 <pre> Calculating... ▶Mp=.1 (Mp)dB=-20. dB ▶fr=.159 Hz </pre>	<p>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.</p>

	<p>We can calculate the Bandwith <math>B_3</math> choosing band(SYS) from Tools menu (F5).</p>
	<p>As you can see, <math>M_p</math> and <math>f_r</math> will results from peak(SYS) when band(SYS) fails.</p>
	<p>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, <math>\omega_{min}</math> and <math>\omega_{max}</math>. Current SYS is showned in the W(s) input field.</p>
	<p>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).</p>
	<p>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).</p>

	<p>We can calculate the step response using step(SYS). Set the amplitude (1 by default) and the time delay.</p>
	<p>Here is the step response.</p>
	<p>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 which we will set the amplitude and time delay.</p>
	<p>Here are the time domain step response parameters, Te, Tr, Ts, Tp and s.</p>
	<p>To get the steady state we'll use function trim(A,B,C, D,u0).</p>

 <p>Calculating...          ▶Steady State          [1/10]          ▶Output          [1/10]</p> <p>CST    DEG AUTO    FUNC    PAUSE</p>	<p>Note that x0 and y0 matches with w-1(t).</p>
 <p>Calculating...          ▶Real Eigenvalues  <math>[e^{-10 \cdot t}]</math>          ▶Complex Eigenvalues          0</p> <p>CST    DEG AUTO    FUNC    PAUSE</p>	<p>With spectre(A) we'll obtain <math>e^{At}</math> in the Continuous Time Domain. Because of real pole, we have not complex expansion.</p>
 <p>Settings          Time Domain..... 10547343 →          Display Digits..... FLOAT 1 →          Results..... AUTO →          Angle..... DEGREE →          Enter=OK    ESC=CANCEL</p> <p>GnuGPL 2005 Loreto Parisi          CST    DEG AUTO    FUNC</p>	<p>Now we will set the Time Domain to Discrete from Settings in Other (F6) menu.</p>
 <p>Calculating...          ▶Real Eigenvalues  <math>[(10.)^k \cdot \cos(2. \epsilon 2 \cdot k) + (10.)^k]</math>          ▶Complex Eigenvalues          0</p> <p>CST    DEG AUTO    FUNC    PAUSE</p>	<p>With spectre(A) we'll obtain <math>A^k</math> in the Discrete Time Domain. Because of real pole, we have not complex expansion.</p>
 <p>Hold Equivalence          A: -10          B: 1          Sampling Time: .01          Sampling Time with tconst() ? No →          Save h(z)? No →          Enter=OK    ESC=CANCEL</p> <p>GnuGPL 2005 Loreto Parisi          TYPE + (ENTER)=OK AND (ESC)=CANCEL</p>	<p>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.</p>

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

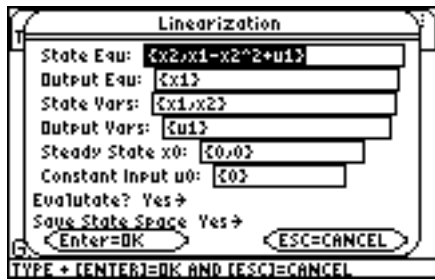
**Examples. 2 – Linearization.**

Now consider a non linear model:

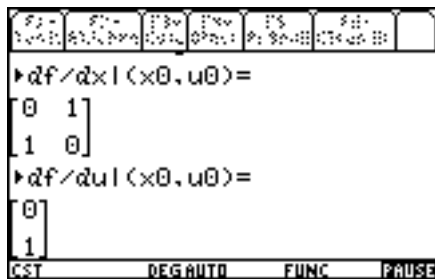
$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_1 - x_2 \\ y = x_1 \end{cases}$$

To work with it, we need to linearize around a steady state  $x_0$  relating to constant input  $u_0$ .

First, we have to calculate the steady state  $x_0$ . It results  $x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$  for input  $u_0=0$ .



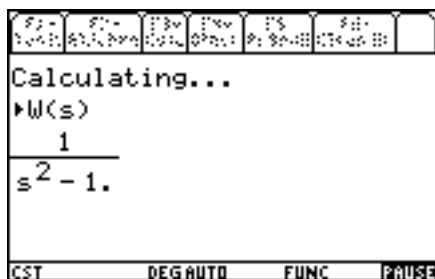
From Dynamics menu (F3) we choose `linmod(f,y,x,u,x0,u0)` where we have  $f=\{x_2,x_1-x_2+u_1\}$ ,  $y=\{x_1\}$ ,  $x=\{x_1,x_2\}$ ,  $u=\{u_1\}$ ,  $x_0=\{0,0\}$  and  $u_0=\{0\}$ . We have decided to evaluate jacobian matrix  $x$  in  $x_0$  and  $u_0$  to save the state-space of obtained linearized model.



Here the jacobian matrices evaluated in  $(x_0,u_0)$ .



Now, we can get transfer function of this new model, that is a approximation of non-linear model above around  $x_0$  and  $u_0$ .



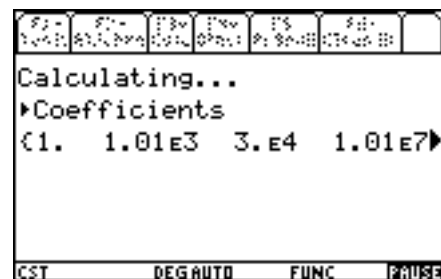
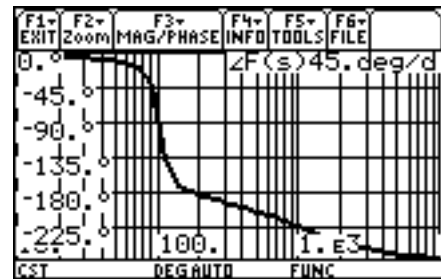
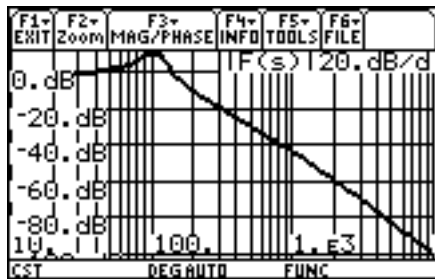
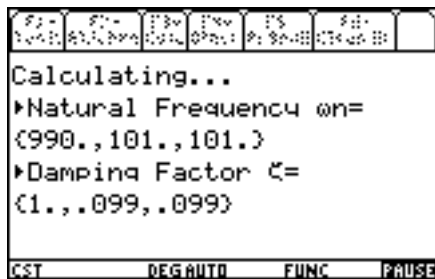
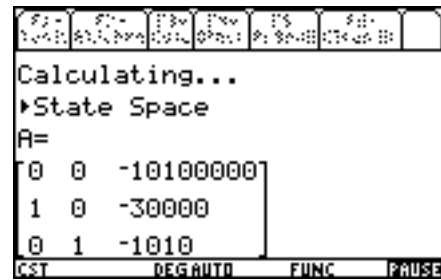
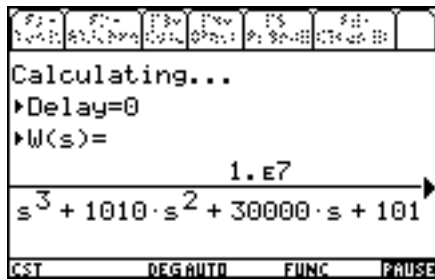
Here is the linearized model transfer function. We can now proceed to study this system in the usual way with our powerful tools.

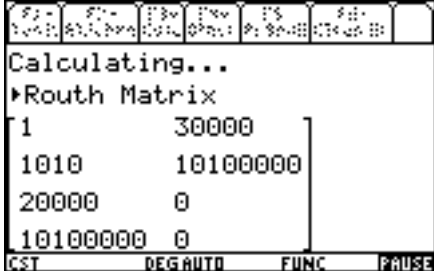


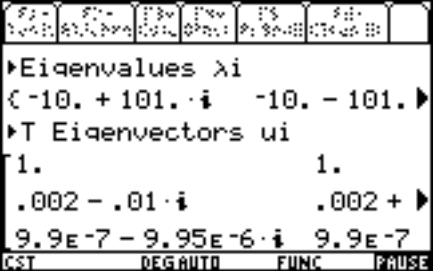



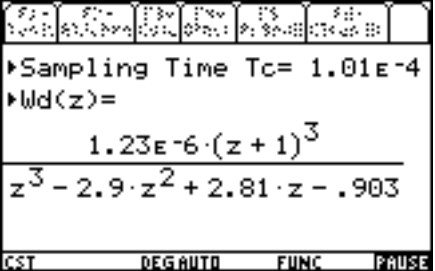
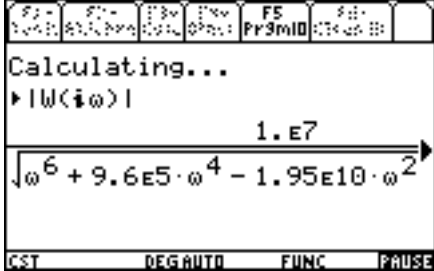
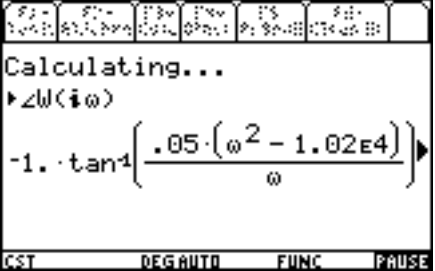
**Examples. 3 – 3<sup>rd</sup> Order LPF.**



Consider a 3<sup>rd</sup> order low pass filter  

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



 <p>Calculating...          ▶Routh Matrix  <math display="block">\begin{bmatrix} 1 &amp; 30000 \\ 1010 &amp; 10100000 \\ 20000 &amp; 0 \\ 10100000 &amp; 0 \end{bmatrix}</math></p>	 <p>Calculating...          ▶Conditions  <math>(1., 1.01E3, -9.9E3*(k-2.02))</math>          ▶Solutions  <math>(true, true, k &lt; 2.02, k &gt; -1.01)</math></p>
 <p>Calculating...          ▶Poles (-1/τ)  <math>(-990., -10., -101.·i, -10)</math>          ▶Zeros (-1/τ')</p>	 <p>Calculating...          ▶Eigenvalues λi  <math>(-10. + 101.·i, -10. - 101.·i)</math>          ▶T Eigenvectors ui  <math>\begin{bmatrix} 1. &amp; 1. \\ .002 - .01·i &amp; .002 + .01·i \\ 9.9E-7 - 9.95E-6·i &amp; 9.9E-7 + 9.95E-6·i \end{bmatrix}</math></p>
 <p>Calculating...          ▶Steady State  <math>\begin{bmatrix} 2.97E4 \\ 1.·E3 \\ .99 \end{bmatrix}</math>          ▶Output  <math>[.99]</math></p>	 <p>Calculating...          ▶Step Response w<sub>1</sub>(t)  <math>(t + 1.37) - .01·e^{-990·t} + .99</math></p>
 <p>Calculating...          ▶Tc=1.01E-4          ▶τi=(.001)          ▶τh=(.1)          ▶Th=(.062)</p>	 <p>Calculating...          ▶Sampling Time Tc= 1.01E-4          ▶Wd(z)=  <math display="block">\frac{1.23E-6·(z+1)^3}{z^3 - 2.9·z^2 + 2.81·z - .903}</math></p>
 <p>Calculating...          ▶ W(iω)   <math display="block">\frac{1.·E7}{\sqrt{\omega^6 + 9.6E5·\omega^4 - 1.95E10·\omega^2}}</math></p>	 <p>Calculating...          ▶zW(iω)  <math display="block">-1.·\tan^{-1}\left(\frac{.05·(\omega^2 - 1.02E4)}{\omega}\right)</math></p>

**Examples. 4 – 2<sup>nd</sup> Order LPF.**

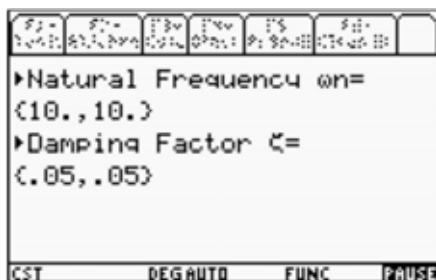
Consider a 2<sup>nd</sup> order low pass filter

$$W(s) = \frac{1}{s^2 + s + 100}$$

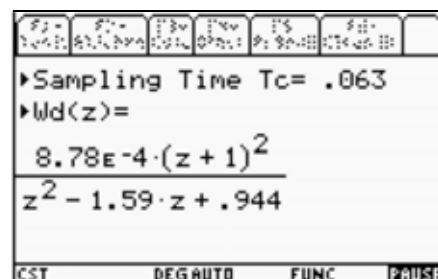
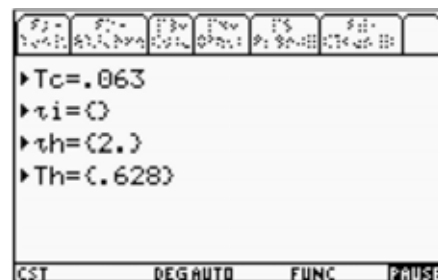
First, we will define the Transfer Function and obtain the State Space Representation.


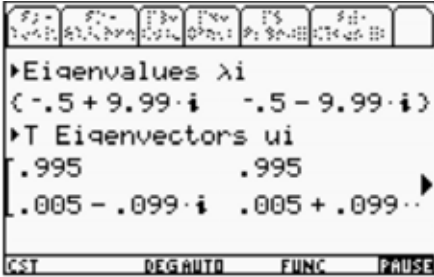
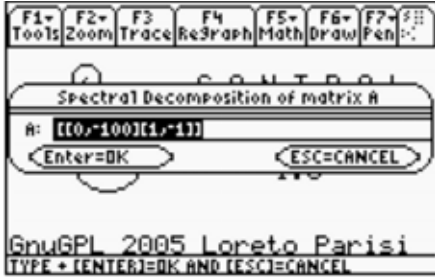
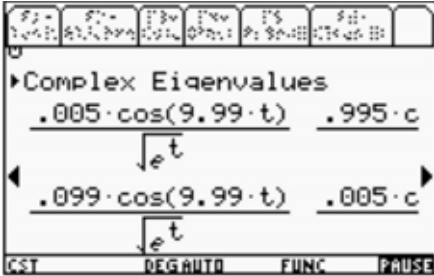
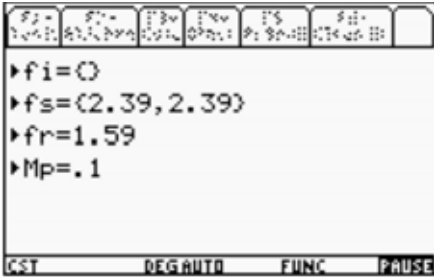
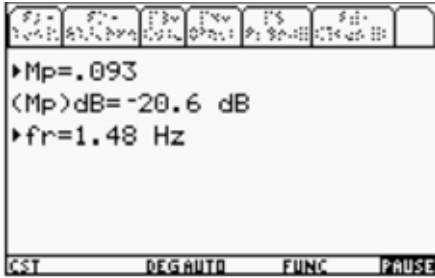
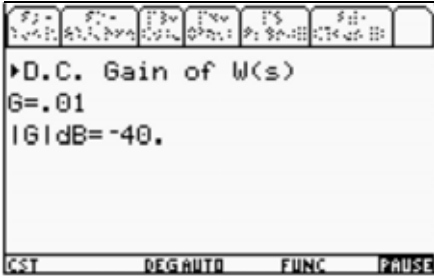


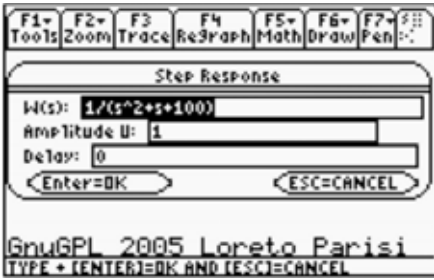
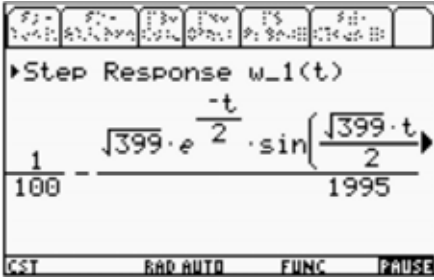


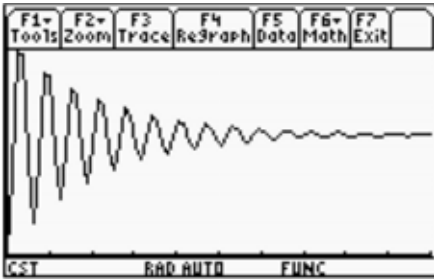

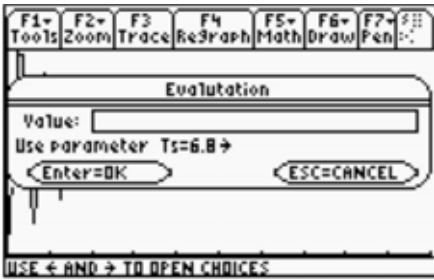
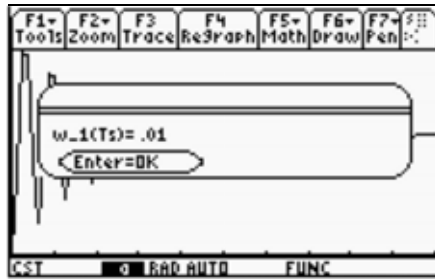
Now we'll get the characteristic polynomial, the natural frequency  $\omega_n$  and the damping factor  $\xi$ .



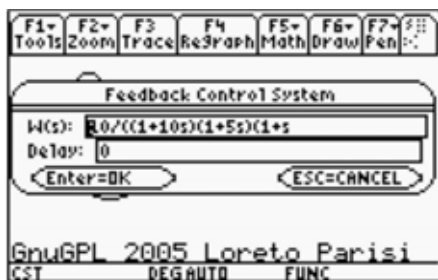
Then we'll calculate the maximum Time Constant, Sample Time and other time constants, and the Discrete Time model relating to this Sample Time with Tustin method. We will save this transfer function  $W_d(z)$  as the current discrete transfer function  $W(z)$ .



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

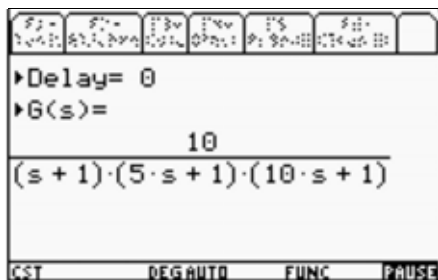
	<p>Let's now calculate and graph the Step Response. Finally we will get the Step Response parameters. This will take some minutes. Please be patient! If <i>pstep()</i> fails, the Error Management System prevents <i>cst</i> to abort, so don't break execution manually (you will lose all unsaved session's data). The EMS will do it for you, saving the current session.</p>
	
	<p>Now we can analyse in the Time Domain the Step Response with <i>gstep()</i>. We can see the Step Response parameters (Data – F5), zoom and trace the graph, get a value (F6) such as the Settlement Time, Ts.</p>
	
	

**Examples. 5 – Network Design I**

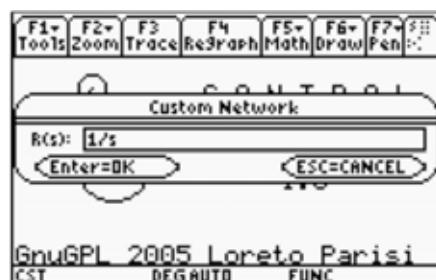


We want to design a closed-loop control system for transfer function

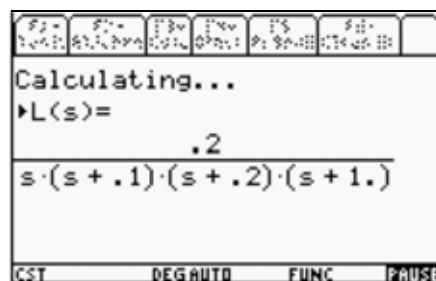
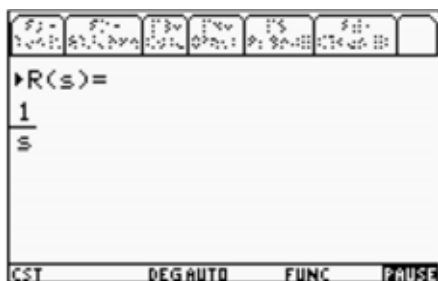
$$W(s) = \frac{10}{(1 + 10s)(1 + 5s)(1 + s)}$$

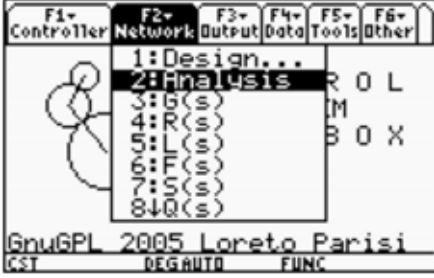

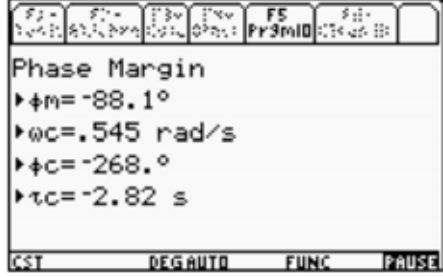
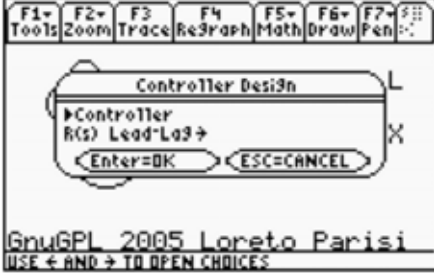
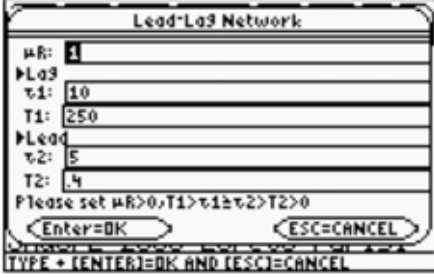
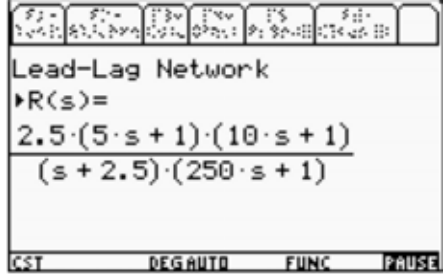
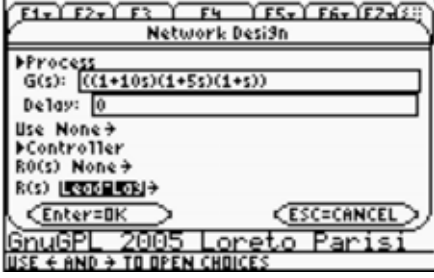
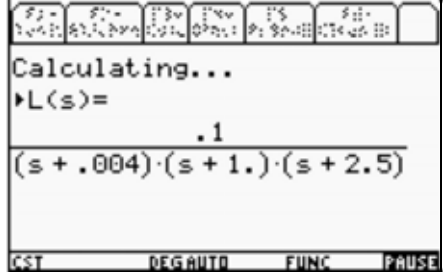



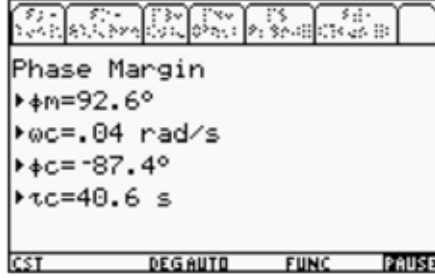
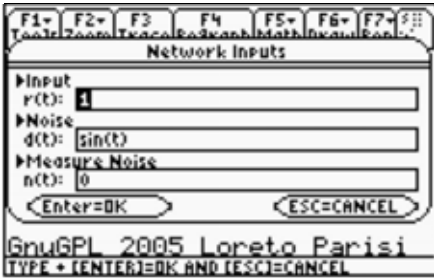
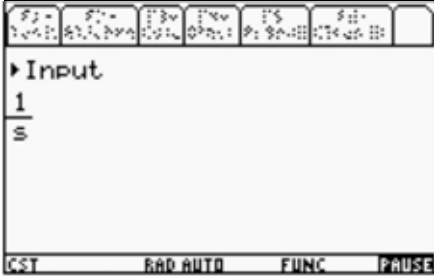
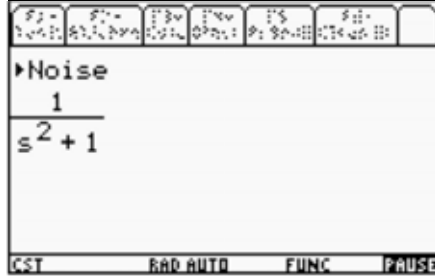

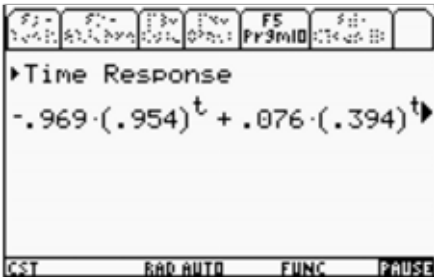
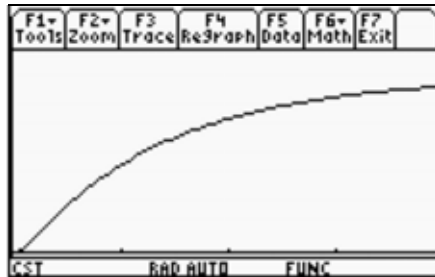
This will be the current process transfer function,  $G(s)$ , as we can see in **Network** menu (F2). Now we will design the controller. Choose *Design* from **Controller** menu (F1) and select the desired structure. We will choose Custom.

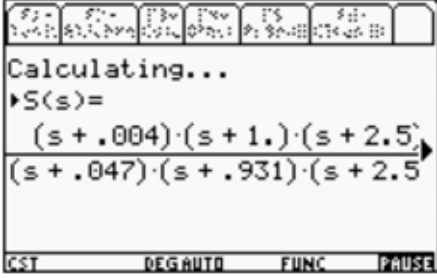
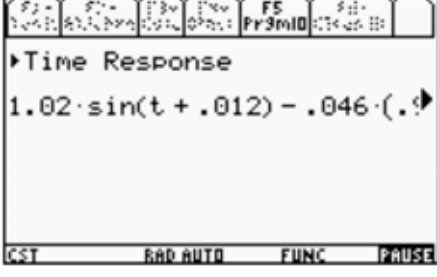
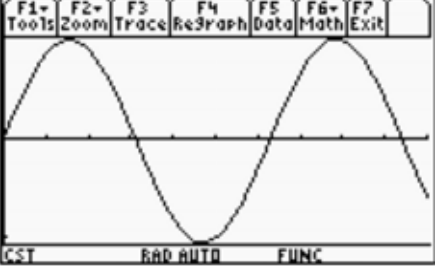

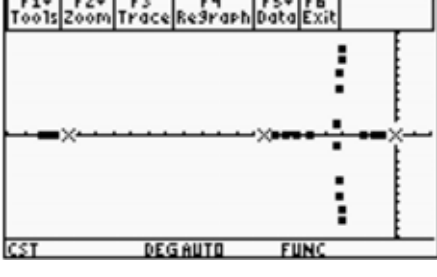
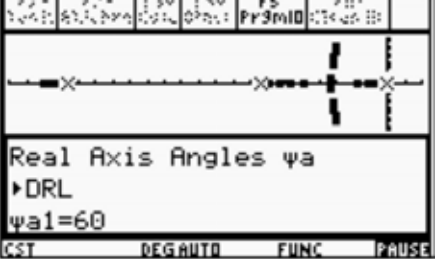
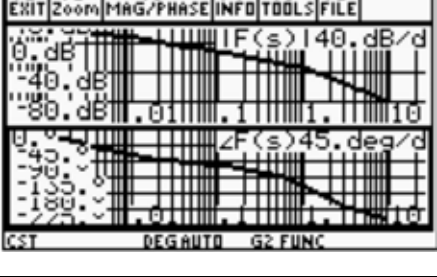
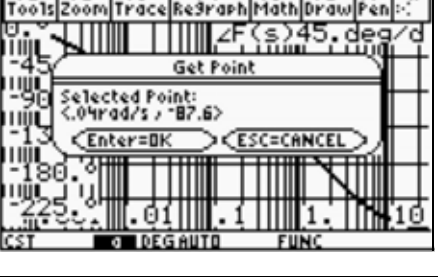


From **Network** menu (F2) choose *Design* to design the network. As you can see, you can specify more than one controller. We will use the previously defined Custom controller as  $R(s)$  and, by now we don't specify  $R_0(s)$ . Then we will get the Closed Loop Transfer Function,  $L(s)$ .



 <p>Now we will analyse the network using <i>Analysis</i> from <b>Network</b> 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.</p>	
	
 <p>So we will define a Lead-Lag network from <b>Controller</b> menu (F1), and do the analysis again. To do so, we need to define again the network structure from <b>Network</b> menu (F2).</p>	
	
	

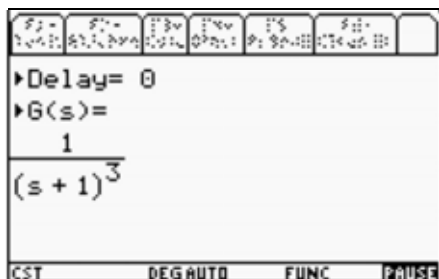
 <pre> Mag Margin ▶km=88. ▶(km)dB=38.9 ▶ωm=1.59 rad/s     </pre>	 <pre> Phase Margin ▶φm=92.6° ▶ωc=.04 rad/s ▶φc=-87.4° ▶τc=40.6 s     </pre>
 <pre> Network Inputs ▶Input r(t): 1 ▶Noise d(t): sin(t) ▶Measure Noise n(t): 0 &lt;Enter&gt;=OK &lt;ESC&gt;=CANCEL GnuGPL 2005 Loreto Parisi TYPE + (ENTER)=OK AND (ESC)=CANCEL     </pre>	<p>We have obtained robust margins for <math>\omega_c=0.04</math> rad/sec. We can now define inputs from <b>Output</b> 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.</p>
 <pre> ▶Input 1 s     </pre>	 <pre> ▶Noise 1 s^2 + 1     </pre>
 <pre> Calculating... ▶F(s)= .1 ----- (s + .047) * (s + .931) * (s + 2.5)     </pre>	<p>Now we need to calculate the Network Transfer Function</p> $F(s) = \frac{R(s)G(s)}{1 + R(s)G(s)}$ <p>to obtain the Time Response against input from <b>Network</b> menu(F2). Then we will choose yr(t) from <b>Output</b> menu (F3).</p>
 <pre> ▶Time Response -.969 * (.954)^t + .076 * (.394)^t     </pre>	

	<p>To obtain the Time Response against noise we need to calculate the transfer function</p> $S(s) = \frac{1}{1 + R(s)G(s)}$ <p>from <b>Network</b> menu(F2), otherwise we'll have a <i>Not Defined</i> error. Then we will choose yd(t) from the <b>Output</b> menu(F3).</p>
	
	<p>We can even plot the Root Locus and Bode diagrams of the Closed-Loop transfer function</p> $L(s) = R(s)G(s)$ <p>In <b>Tools</b> menu (F5) select <i>Choose SYS</i> and choose L(s). This will be the current SYS to apply all Tools menu functions. The Root Locus confirms closed-loop stability.</p>
	
	

**Examples. 6 – Network Design II**



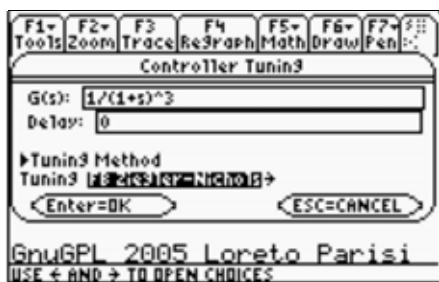
**Examples. 7 – Network Tuning I**



Consider a 3<sup>rd</sup> 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).

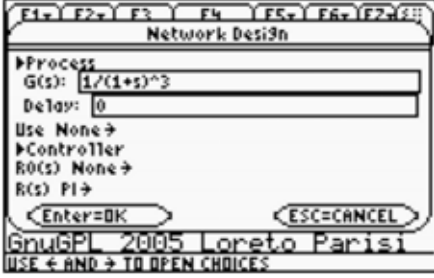


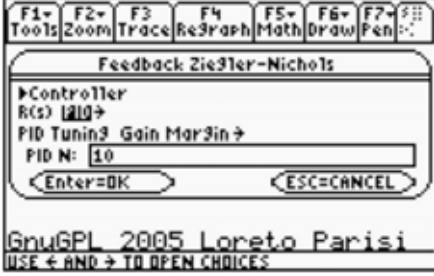

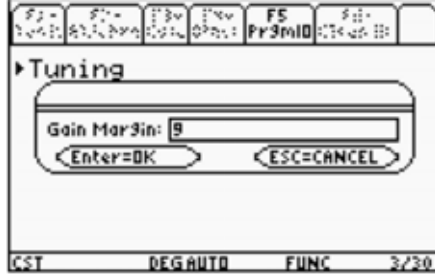
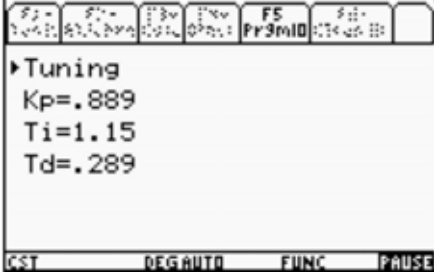
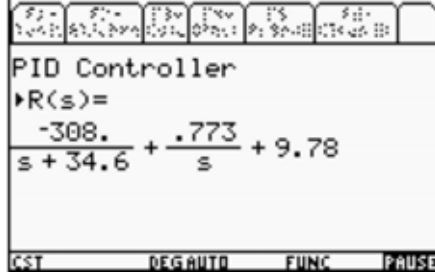


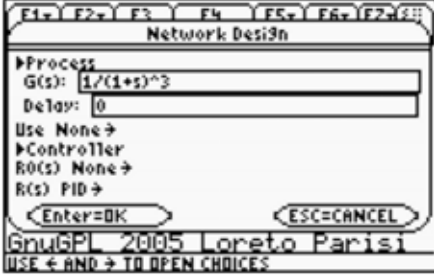
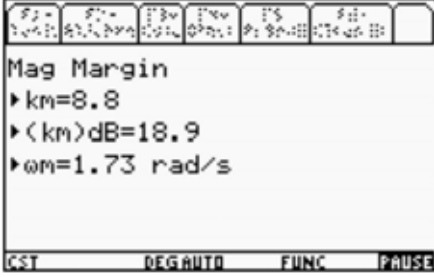
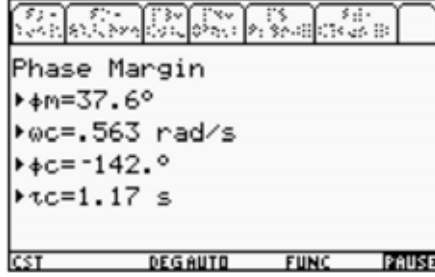

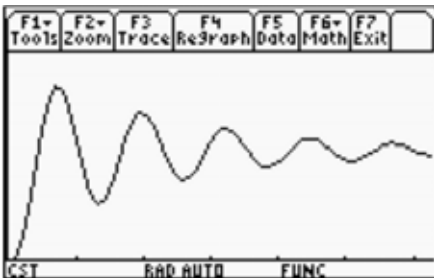
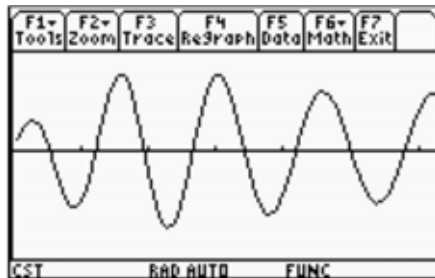
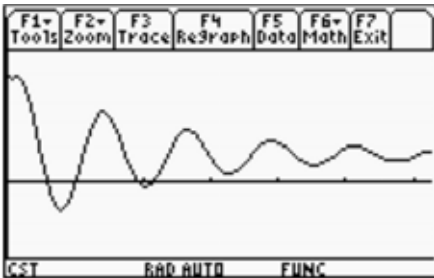
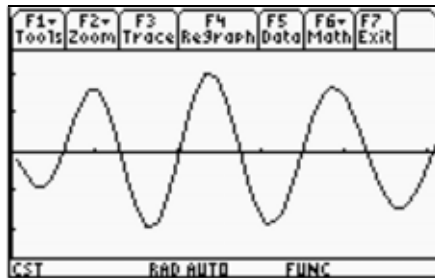
Select FB (FeedBack) Ziegler-Nichols as *Tuning Method*. Then choose PI (Proportional-Integral) as controller R(s).



You will obtain the *Critical Mag Kp'* and relating frequency  $\omega\pi$  and period T first. From Ziegler-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.



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

	<p>Now we will re-define the network with this PID controller and perform a network analysis from <b>Network</b> menu (F2). In this case, we have obtained a more robust control network as results from Mag and Phase margin values.</p>
	
	<p>We will analyse the output behaviour of network (with PI controller), defining inputs from <b>Output</b> menu (F3). First we will calculate <math>F(s)</math>, <math>S(s)</math> and <math>Q(s)</math> needed by <math>y_r(t)</math>, <math>y_d(t)</math> and <math>u_r(t)</math>, <math>u_d(t)</math> respectively.</p> <p>Please refer to Section <i>Network Design</i> for further explanations about inputs.</p>
	
	

**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:
  - Simultaneous 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
    - Analisis
      - 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: 1<sup>st</sup> edition, October 2005  
Distribution: Portable Document Format
- *The CST Reference Guide*  
Current version: 1<sup>st</sup> edition, October 2005  
Distribution: Portable Document Format
- *The CST User Guide*  
Current version: 5<sup>th</sup> edition, October 2005  
Distribution: Portable Document Format

## Contents

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

<b>Name</b>	<b>Description</b>	<b>Type</b>
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



## 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](mailto:92brothers@infinito.it)  
Home: <http://www.92brothers.net/>
- *Francesco Orabona*  
Contribute: *logspace()*, *poly2cof()*, *zpk()*, *nyquist()*, *rlocus()*  
E-mail: [bremen79@infinito.it](mailto:bremen79@infinito.it)  
Homepage: <http://web.genie.it/utenti/b/bremen79/>
- *Lars Frederiksen*  
Contribute: *DiffEq()*  
E-mail: [ltf@post8.tele.dk](mailto:ltf@post8.tele.dk)
- *Greg Dietsche*  
Contribute: *kerno()*  
E-Mail: [calc@gregd.org](mailto: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](mailto: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!*