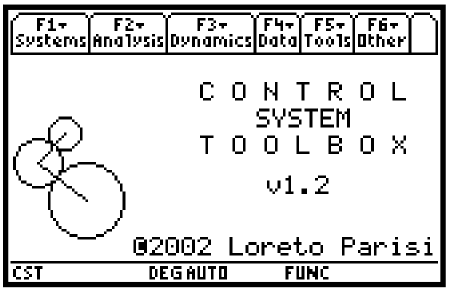
Control System Toolbox

for TI-89



version 1.2

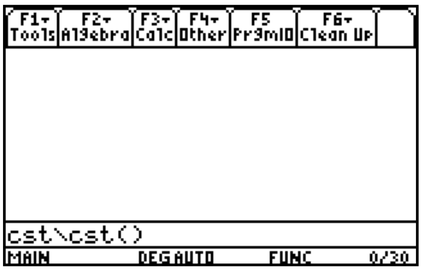
The CST Reference Guide

second release December 2002

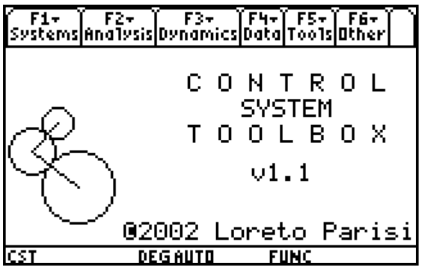
Index **Page**

What's Control System Toolbox *for TI-89*

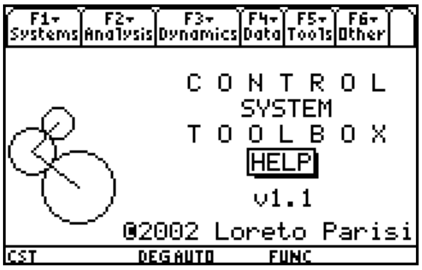
Control System Toolbox for TI-89 is a suite of specialized functions in Systems Control created for the TI-89 portable calculator by *Loreto Parisi* among June and July 2002. This software incorporates most of functions who cames with Control Toolbox of Matlab® by The MathWorks Inc.



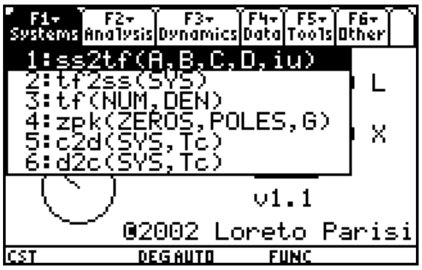
After installing (see *How To Install* on page 8), to run the program on your calculator, types *CST/cst()* from folder *MAIN* and wait few seconds.



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



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



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

Disclaimer

The Open Source Philosophy

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

This is our way of thinkin'...

How To Get Help

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

Loreto Parisi
Via Antonio Gramsci n°13
Alife(CE) 81011
Italy

or you can send me an e-mail to this address

loretoparisi@libero.it

or visit my website at this URL

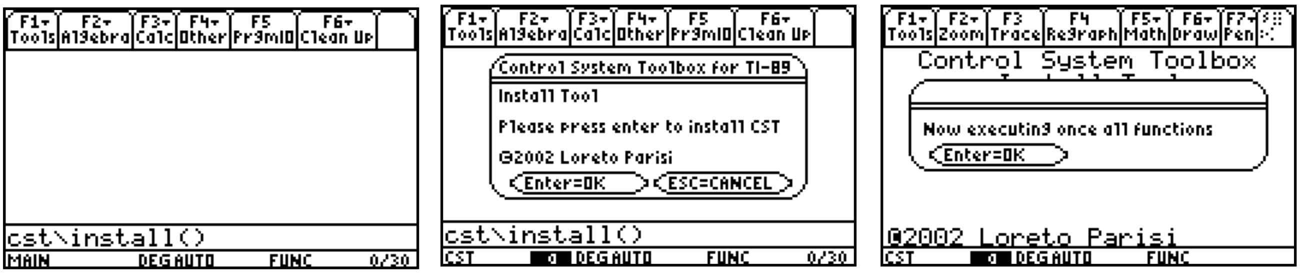
http://web.tiscali.it/loretoparisihome

You can send me feedback from here:

http://web.tiscali.it/loretoparisihome/feedback

How To Install

Use your linking software to send the program *cst.89g* on the calculator. All the files are automatically placed in the *CST* folder. 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*.

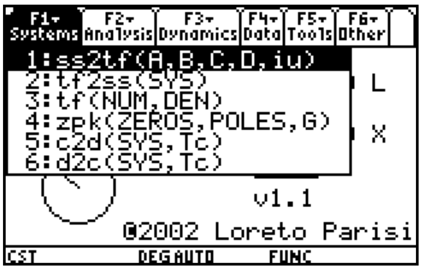


Run *CST\install()* from HOME. First *install*() will execute once all functions before archiving them. This tip will quitely increase execution speed.

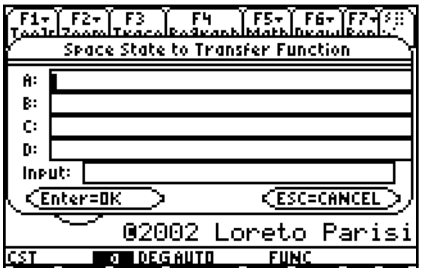
After archiving all functions, *install()* will runs *cst()* once before archiving it, so you have to exit it pressing F6 then 6, or using arrow keys and select *exit* from *Other* menu (*F6*).

To exit *help()* press *F6* then *6* or select *exit help()* from *Other* menu (*F6*). After installation you can delete *install()* from *CST* folder. To run CST type *CST\cst()* from HOME.
This software was tested on a TI-89 with AMS v2.08 and it works fine.

Systems

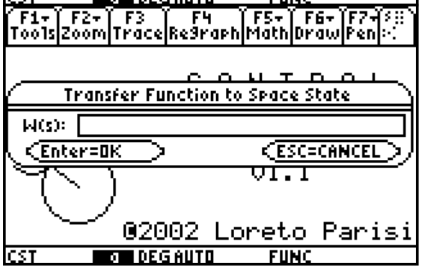


The *Systems* menu (*F1*) contains all the functions to build the model, using state-space or transfer function and to perform conversions from one representation to another, even from continuous time to discrete time model.



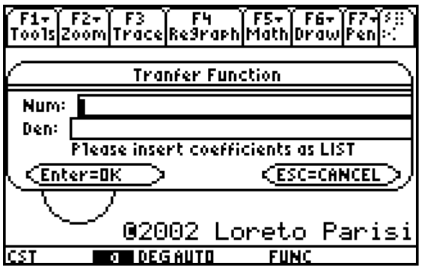
ss2tf(A,B,C,D,iu)

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).



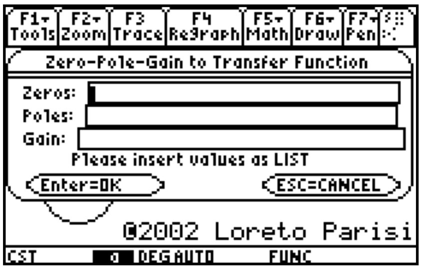
tf2ss(SYS)

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



tf(NUM,DEN)

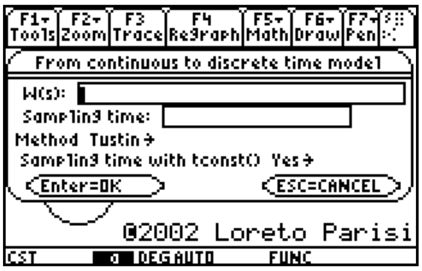
Calculates transfer function, where NUM and DEN are LIST of coefficients of numerator's and denominator's polynomial: NUM={b_0, b_1,…,b_n}, DEN={a_0,a_1,…a_n}, so W(s)=$\frac{b_0 s^n + b_1 s^{n-1} + ... + b_n}{a_0 s^n + a_1 s^{n-1} + ... + a_n}$.



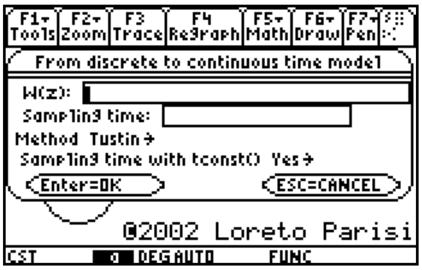
zpk(zeros,poles,gain)

Calculates transfer function W(s) in the zeros-poles-gain representation, where zeros, poles are LIST of zeros of numerator and denominator (poles), while gain is NUM and represents constant gain K.

Original version: *Francesco Orabona*
e-mail: bremen79@infinito.it
Homepage: http://web.genie.it/utenti/b/bremen79/

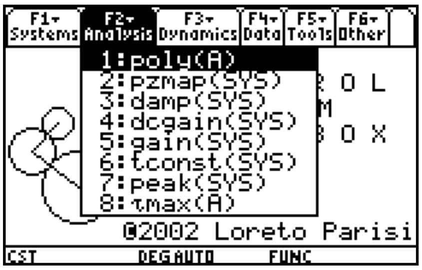


c2d(SYS,T_c)
Converts continuous time model SYS to the discrete time model, using sample time T_c and different methods: linear Euler and bilinear Tustin methods are implemented. Can use function *tconst(SYS)* to determinate sample time T_c. Use function sampler(A,B,T_c) to use ZOH method.

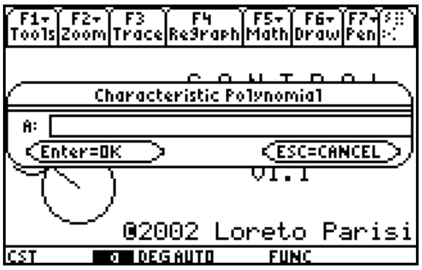


d2c(SYS,T_c)
Converts discrete time model SYS (in z) to the continuous time model, using sample time T_c and different methods: linear Euler and bilinear Tustin methods are implemented. Can use function *tconst(SYS)* to determinate sample time T_c.

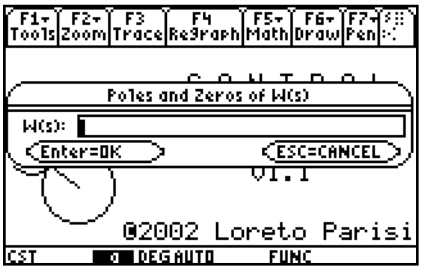
Analysis



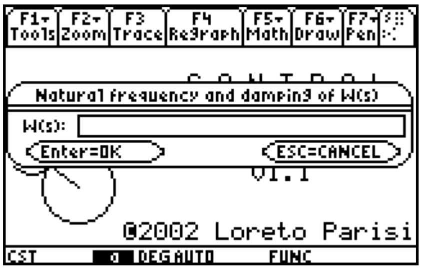
The *Analysis* menu (*F2*) contains all the tools to analyze the model you have created with *Systems'* tools. You can also analyze different models, using different SYS at time. This will not change current transfer function.



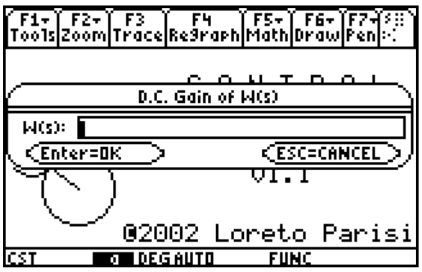
poly(A)
This function calculates characteristic polynomial of matrix A, as p(s)=$|sI-A|$, where $|\bullet|$ is determinat of a matrix.



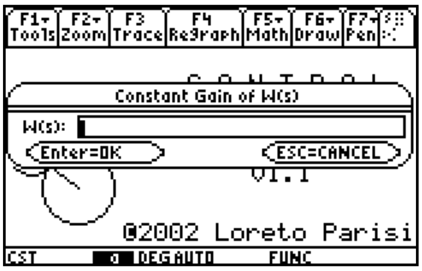
pzmap(SYS)
This function calculates poles and zeros of given transfer function SYS, where poles are zeros of denominator of SYS.



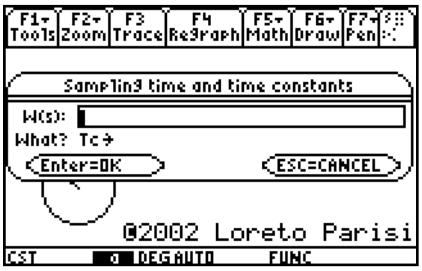
damp(SYS)
Calculate natural frequencies ω_{nh} and damping factors ζ_h for transfer function SYS, where $\omega_{nh}=\sqrt{\alpha_h^{\,2}+\omega_h^{\,2}}$ and $\zeta_h=\frac{-\alpha_h}{\omega_{nh}}$ for eigenvalue $\lambda_h = \alpha_h + j\omega_h$.



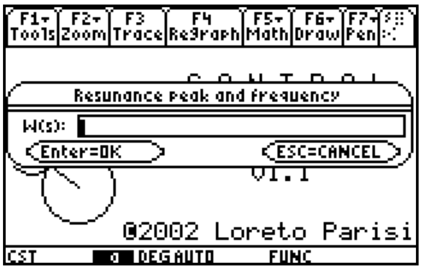
dcgain(SYS)
Calculates d.c. gain G for transfer function SYS, as G=$\lim_{s\to 0} W(s)$.



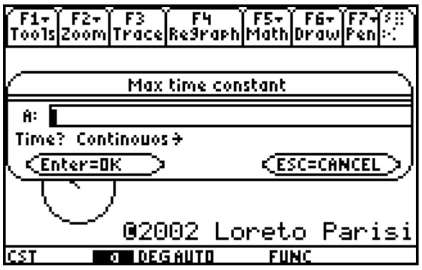
gain(SYS)
Calculates constant gain K for transfer function SYS, as $K=\lim_{s\to 0} s^{n_0-m_0} W(s)$, where n_0 and m_0 are multiplicity of zero roots for denominator and numerator.



tconst(SYS)
Calculates sample time T_c and time constants τ_i, τ_h, and T_h, where $\tau_i = -\frac{1}{\lambda_i}$, $\tau_h = -\frac{1}{\alpha_h}$ and $T_h = \frac{2\pi}{\omega_h}$, while $T_c = 0.1\min\{\tau_i, \tau_h, T_h\}$.



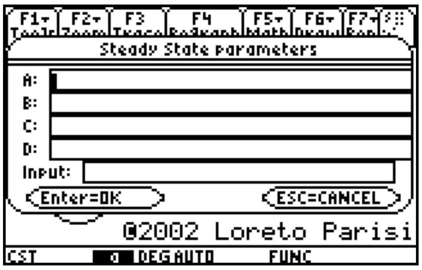
peak(SYS)
This function uses a proprietary numerical algorithm to calculate resonance peak $M_p = \max_\omega M(\omega)$, where $M(\omega) = |W(s)|_{s=j\omega}$ and relating frequency f_r, which is $M(2\pi f_r) = M_P$.



tmmax(A)
Calculates maximum time constant for characteristic polynomial of matrix A, in continuous or discrete time, where $\tau_{max} = \frac{1}{\min(-\Re\lambda_i)}$ (continuous time) and $\tau_{max} = \frac{1}{\min(\ln|\lambda_i|)}$ (discrete time).

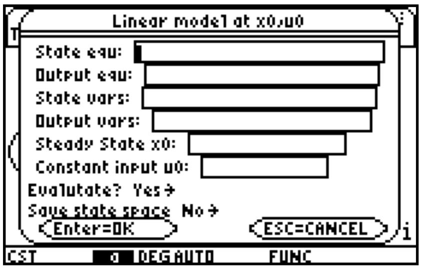
Dynamics

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



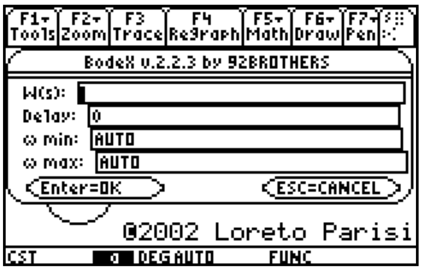
trim(A,B,C,D,u_0)

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



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

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



bodex(SYS)

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

Included in the suite CST *for TI-89* with permission of 92BROTHER.

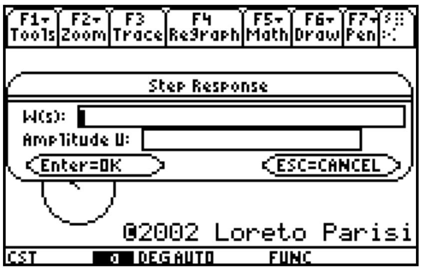
BodeX v.2.2.3

Copyright © 2000 92BROTHERS

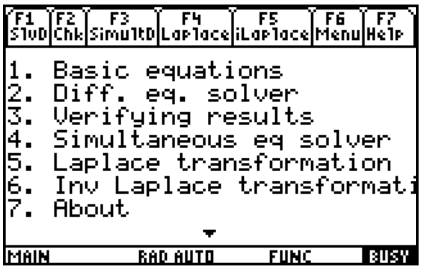
For contacts please refers to:

e-mail: 92brothers@infinito.it

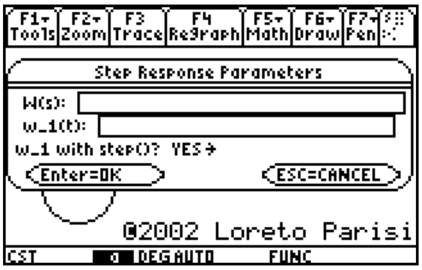
URL: http://www.92brothers.net/



step(SYS)
This tool calculates the step response for SYS, as $U^*w_{-1}(t) = L^{-1}(W(s)U/s)$, with amplitude U. It needs the tool DiffEq v. 2.04 or next by *Lars Frederiksen* to perform symbolic calculation of Laplace direct and inverse transformation.

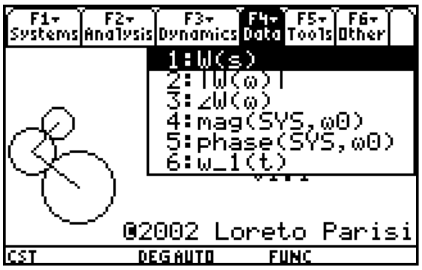


DiffEq v. 2.04
Copyright © *Lars Frederiksen*
To contact please refers to
e-mail: ltf@post8.tele.dk

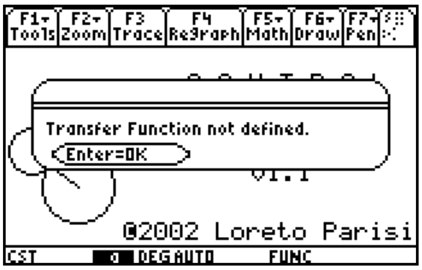


pstep(SYS)
This tool calculates characteristic parameter of step response for transfer function SYS, such as T_e, T_a, T_s, T_p and s. Step response $w_{-1}(t)$ can be specified or calculated with *step(SYS)*. It needs DiffEq v. 2.04 or next by *Lars Frederiksen* (as above).

Data

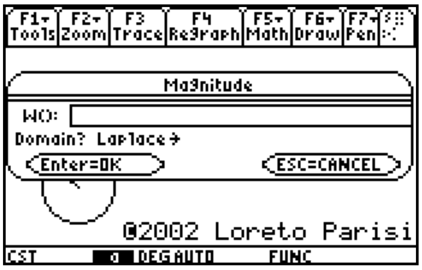


The *Data* menu (*F4*) gives access to current transfer function W(s), to its magnitude and phase, and to current step response $w_{-1}(t)$.



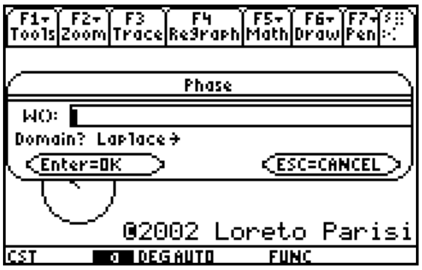
W(s)

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.



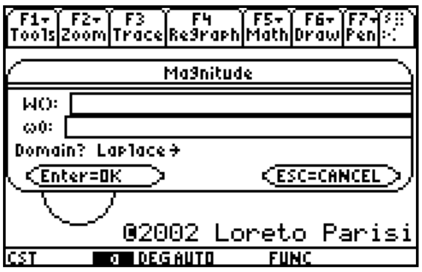
$|W(\omega)|$

Displays magnitude of transfer function W(•) in Laplace domain (for W(s)) and even in domain Z (for W(z)).



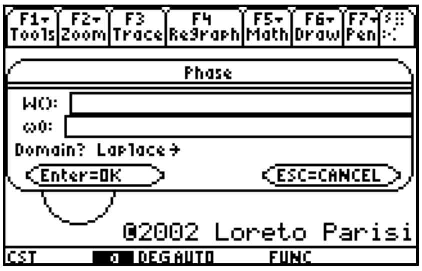
$\angle W(\omega)$

Displays phase of transfer function W(•) in Laplace domain (for W(s)) and even in domain Z (for W(z)).



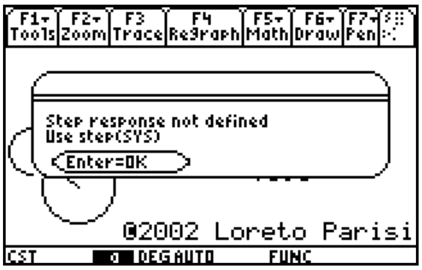
mag(SYS,ω_0*)*

Calculates magnitude of SYS in Laplace domain (for W(s)) and even in domain Z (for W(z)), relating to ω_0.



phase(SYS,ω_0)

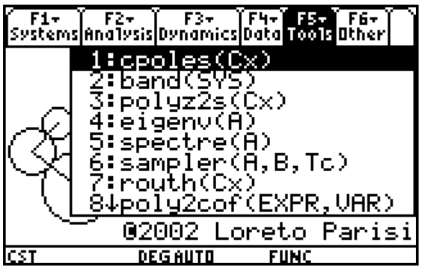
Calculates phase of SYS in Laplace domain (for W(s)) and even in domain Z (for W(z)), relating to ω_0.



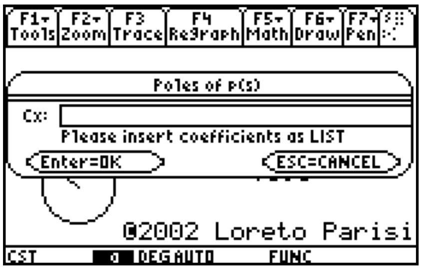
$w_{-1}(t)$

Displays the step response of current transfer function W(s), or the last calculated step response of SYS, obtained with *step(SYS).*

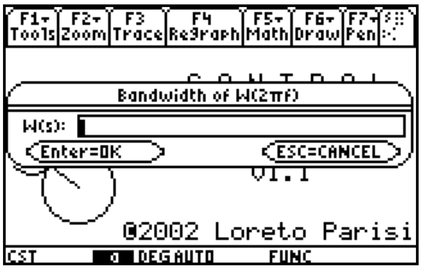
Tools



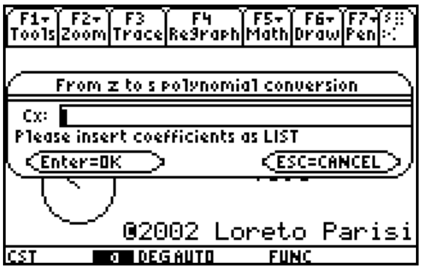
The *Tools* menu (*F5*) offers several useful functions to complete the analysis of the model you're working and to give more detailed information about it. Moreover presents different tools for discrete systems and finite state systems.



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

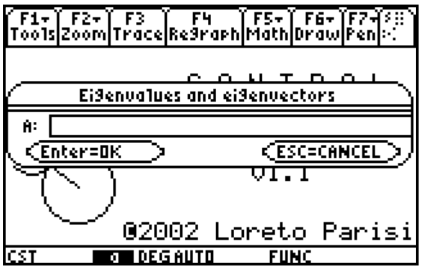


band(SYS)
This function uses a numerical algorithm and several preexistent formulas to calculate bandwith of system with transfer function SYS. It calculates f_i, f_s, where B=[f_i,f_s], f_r (resonance frequency) and M_p (resonance peak).

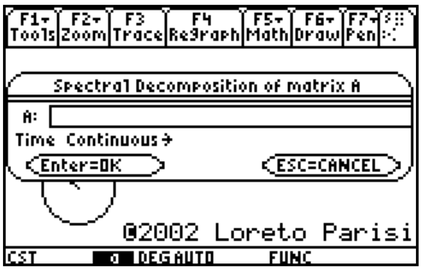


polyz2s(Cx)
This tool calculates the continuous polynomial q(s), relating to discrete polynomial p(z), assigned in terms of its coefficients LIST, Cx, using the formula

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



eingev(A)
It calculates eigenvalues and eigenvectors of matrix A.



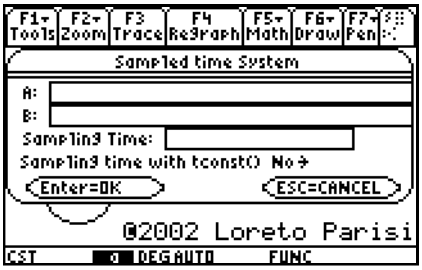
spectre(A)
This tool calculates the spectral decomposition of matrix A, even in the continuous (e^{At}) and in the discrete time (A^k), relating to real eigenvalues and complex eigenvalues.

The spectral decomposition of matrix A is

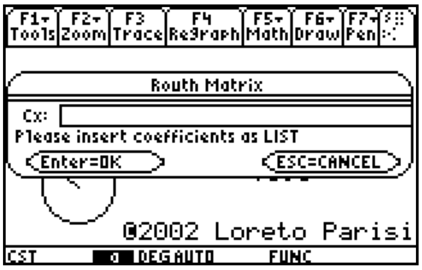
$$e^{At} = \sum_{i=1}^{\mu} u_i e^{\lambda_i t} v_i^T + \sum_{h=1}^{\upsilon} (u_{ha} \quad u_{hb}) e^{\alpha_h t} \begin{pmatrix} \cos\omega_h t & \sin\omega_h t \\ -\sin\omega_h t & \cos\omega_h t \end{pmatrix} \begin{pmatrix} v_{ha}{}^T \\ v_{hb}{}^T \end{pmatrix} \text{ (continuous time)}$$

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

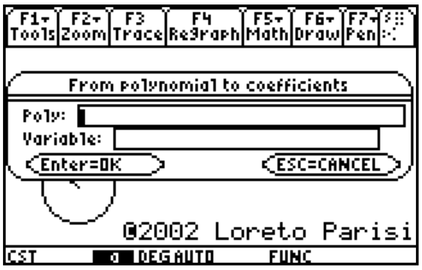
relating to real μ eigenvalues λ_i and 2υ complex eigenvalues $\lambda_h = \alpha_h \pm j\omega_h = \rho_h e^{\pm j\theta_h}$ and the relating eigenvector u_i and $u_h = u_{ha} \pm u_{hb}$.



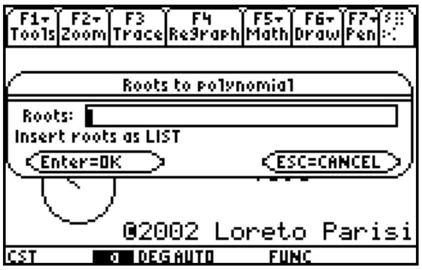
sampler(A,B,Tc)
This function performs the discrete time conversion of continuous time model with state-space $\dot{x} = Ax + Bu$ at sample time T_c, using the ZOH (Zero Order Hold) method. It permits to use sample time T_c calculated with function *tconst(SYS)* for current transfer function SYS, stored in w. Saving state-space is leaved as option.



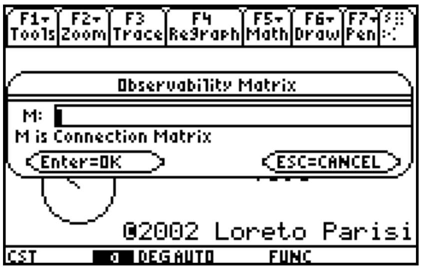
routh(Cx)
It calculates the Routh matrix for polynomial assigned with its coefficients LIST, Cx.



poly2cof(expr,var)
Gives the LIST of coefficients of the polynomial given in *expr* in the variable *var*.
Function *poly2cof(expr,var)* included in *CST for TI-89* with permission of *Francesco Orabona*. For contacts:
E-mail: bremen79@infinito.it
Homepage: http://web.genie.it/utenti/b/bremen79/

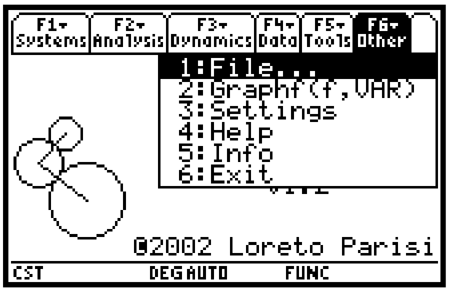


rts2poly(roots)
Builds the polynomial with *roots* assigned as LIST.
Function *rts2poly(roots)* included in *CST for TI-89* with permission of *Chadd L. Easterday*. For contacts:
Email: easterday@mindspring.com

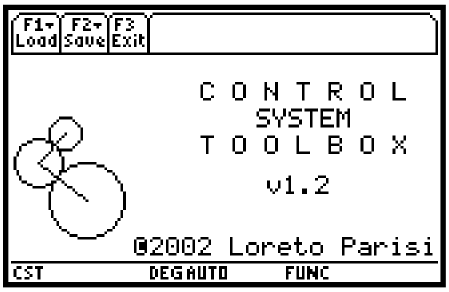


reach(M)
Builds the observability matrix M_c^T from the assigned connection matrix M.

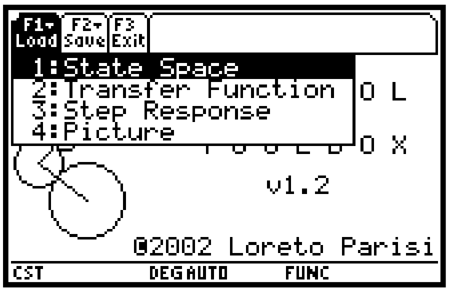
Other



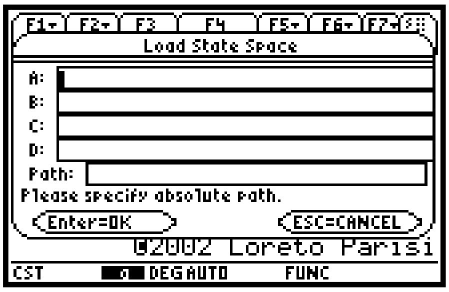
The *Other* menu (*F6*) gives tools to manage files, the *Settings*, the access to on-line help tool with *help*(), some *Info*, and the way to *exit* CST.



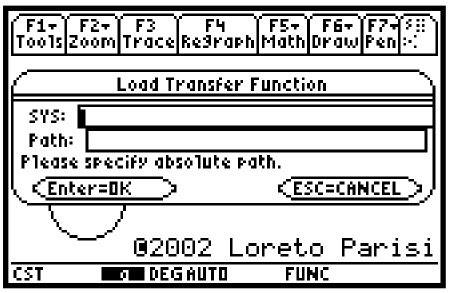
The *File* toolbox gives access to the File Management. Here you can load and save the State Space, the Transfer Function, the Step Response and pictures created by *bodex(SYS)*.There are three menus *Load*, *Save* and *Exit*. *Exit* menu (F3) brings to previous toolbox.



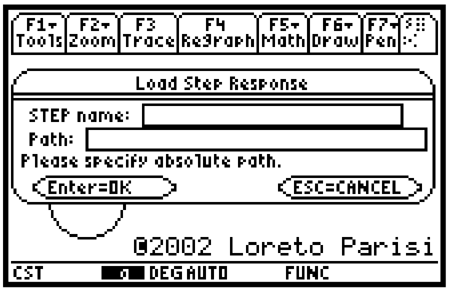
The *Load* menu (F1) permits to load State Space, Transfer Function, Step Responde and Picture from a specified path.



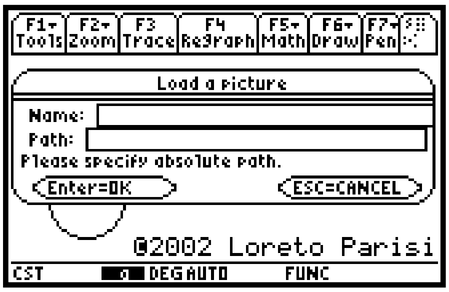
Load State Space. To load state space matrixes A,B,C,D from specified path. Please use absolute path. For example, if your dynamic matrix A is stored in *main* as *dyn*, you have to input *dyn* in A input field and *main* as path. All matrixes should be in the same path.



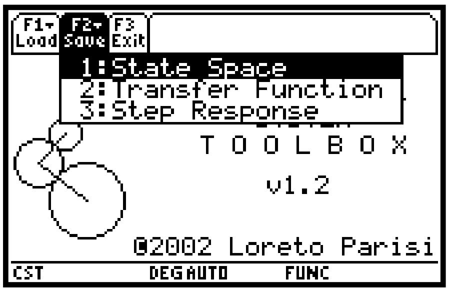
Load Transfer Function. Permits to load Transfer Function from specified path.



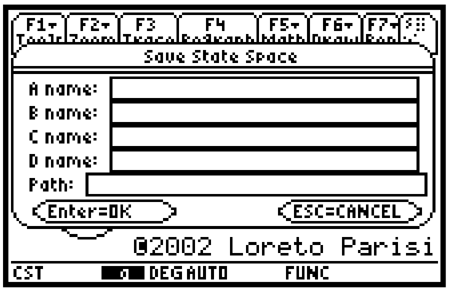
Load Step Response. Permits to load the Step Response from specified path.



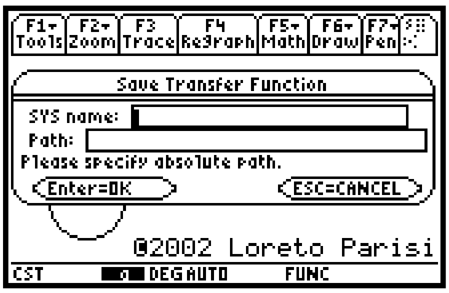
Load a Picture.This tools permits to load a picture stored in *CST* folder. It's aid is in displaying Bode plots, created with *bodex()* first, and estimating the diagrams in a assigned frequency ω_0.



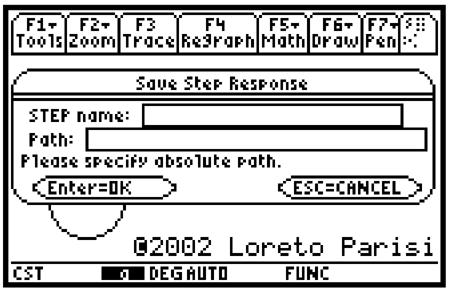
The *Save* menu (F2) permits to save State Space, Transfer Function, Step Responde into a specified path.



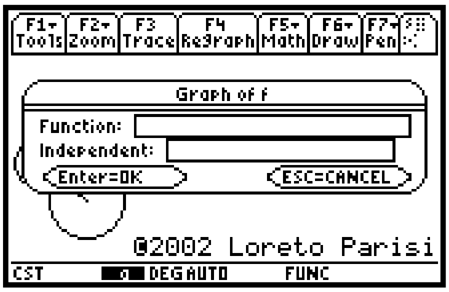
Save State Space. Permits to save current State Space into the specified path, using given names.



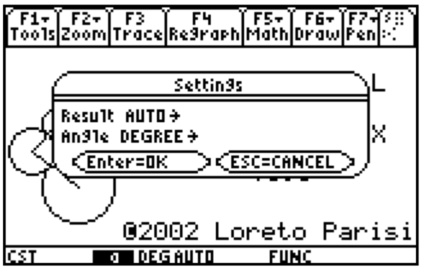
Save Transfer Function. To save current transfer function into the specified path, using given name. The current SYS results from *Data* menu (F4).



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

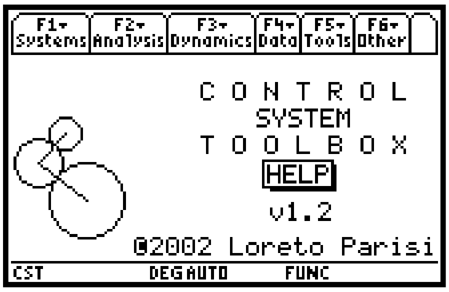


graphf(f,var) Simply graphs a function with independent variable *var*.



Settings

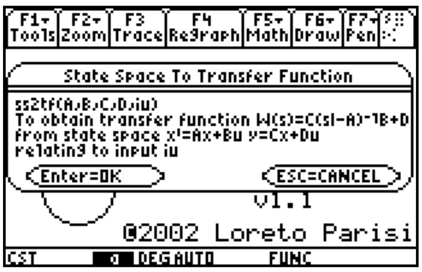
It permits to modify some settings of the calculator, such as the *angle* and the *result* format.



help()

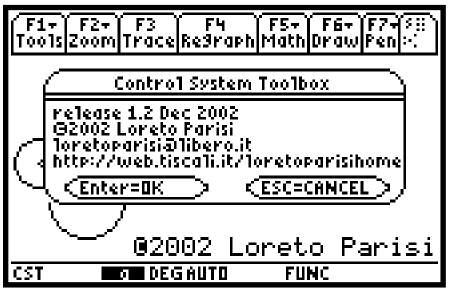
The program *help()* loads the on-line help tool, which gives an instant help on all the functions of CST suite.

Note that *help*() is a standalone program also, so you can recall it from HOME, typing *CST/help().*

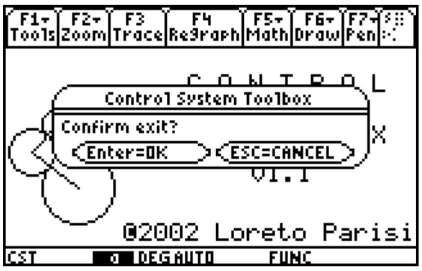


help()

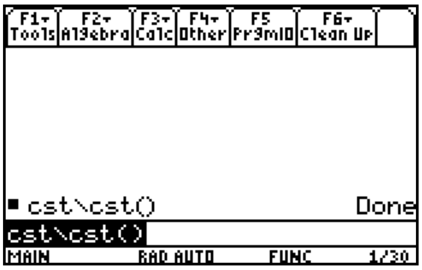
To get help, simply choose a function from one of the menus and you’ll get some information about it.



Info
Gives the current version of CST *for TI-89* and several ways to contact the author.



Exit
Exit Control System Toolbox *for TI-89.* All previous settings and folder will be restored.



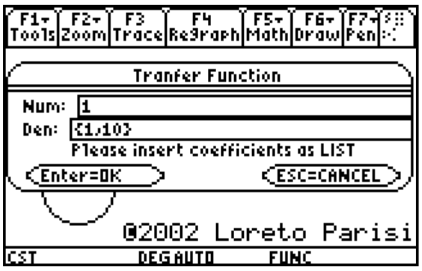
cst()
Enjoyed the journey ?
We’ll be pleased.

Examples

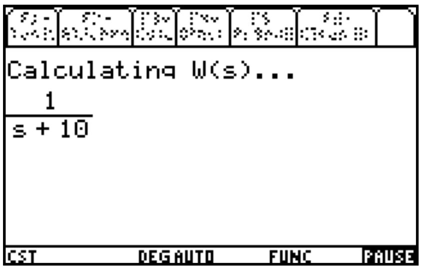
Example 1. *First order LPF*

Considers a sample low pass filter, with transfer function W(s)=$\frac{1}{s+10}$.

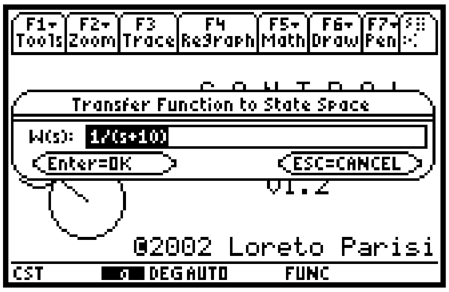
See what can be done in *CST.*



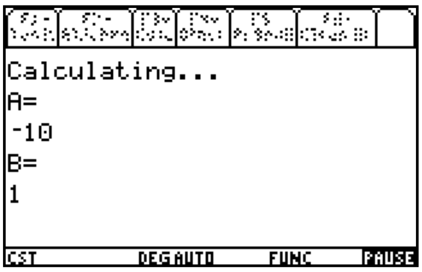
First of all, we'll define the current transfer function. From *Systems* menu (*F1*) choose *CST* function *tf(NUM,DEN)* where NUM=1 and DEN={1,10}.



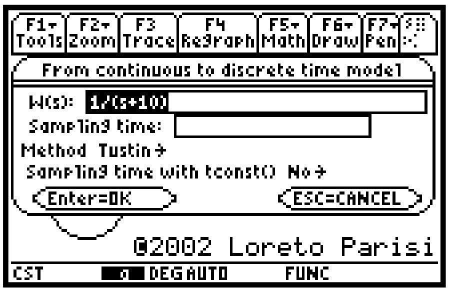
This stores $\frac{1}{s+10}$ as the current transfer function SYS.



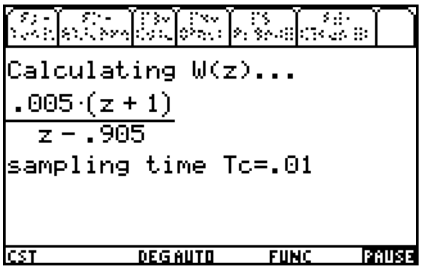
To get relating state-space, now we'll use function *tf2ss(SYS)*, where SYS is the current transfer function, automatically showned in the input field.



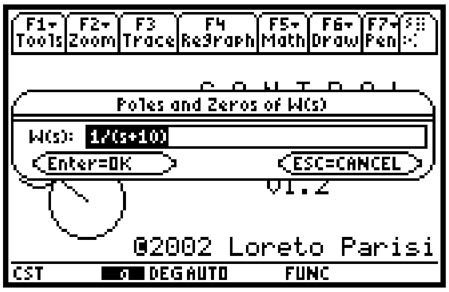
After calculation of all matrixes, they will be the current state space SS.



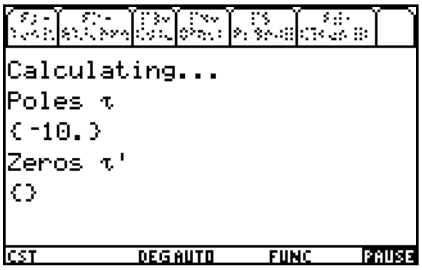
If you need to obtain the discrete model of $\frac{1}{s+10}$, use *c2d(SYS,T_c)* where SYS=w and T_c the sample time desidered. We'll use *tconst(SYS)* to get T_c and 'Tustin' bilinear transformation as method.



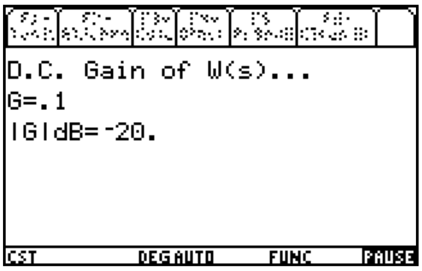
Here is the discrete time model with sample time T_c from *tconst(SYS)* and 'Tustin' method.



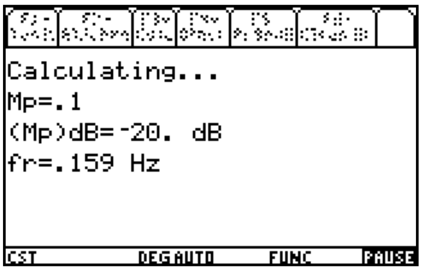
Now, we'll gonna calculate poles and zeros of $\frac{1}{s+10}$, considering that it is stored in w. From *Analysis* menu (*F2*) we'll choose *pzmap(SYS)* where SYS is the current transfer function.



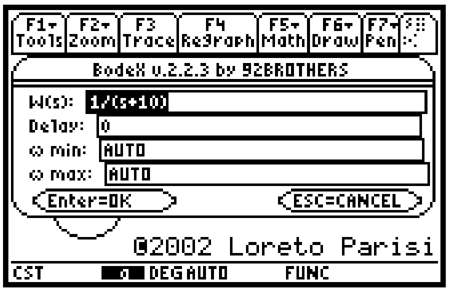
We have one pole at 10 rad/sec and no zeros.



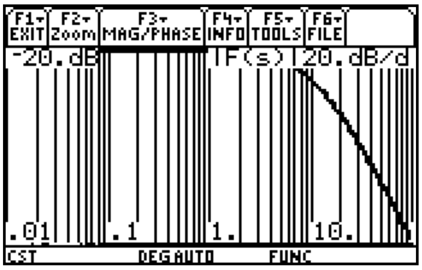
With function *dcgain(SYS)* we have calculated the d.c. gain of $\frac{1}{s+10}$, that results –20 dB.

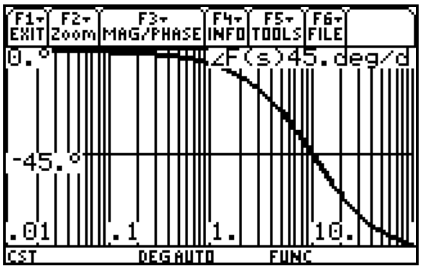


In the same way, with function *peak(SYS)* we'll obtain the resonance peak M_P (in this case it matches with previous d.c. gain G) and relating resonance frequency f_r that results .159 Hz.

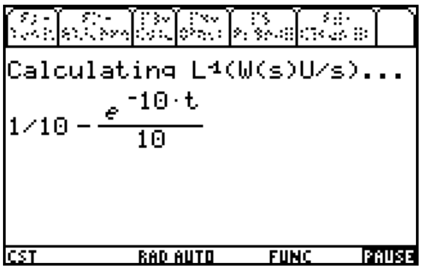


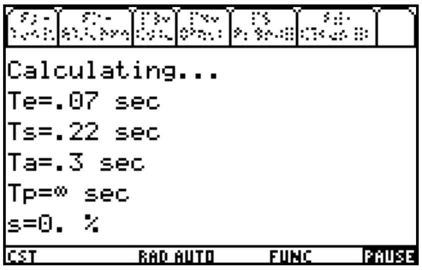
From *Dynamics* menu (*F3)* we'll going to plot Bode diagrams with function *bodex(SYS)*, courtesy of 92BROTHERS. We'll use w and 0.01 and 100 as transfer function, ω_{min} and ω_{max}. Current SYS is automatically showned in the W(s) input field.



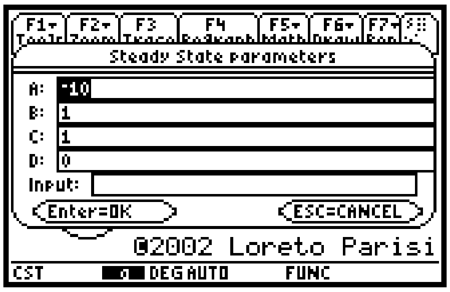


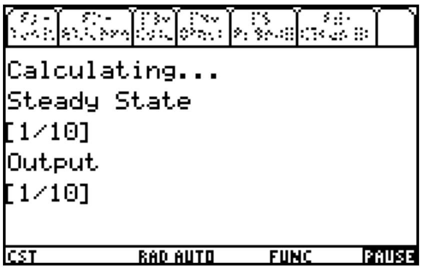
Now we'll save those plots and exit *bodex()*.



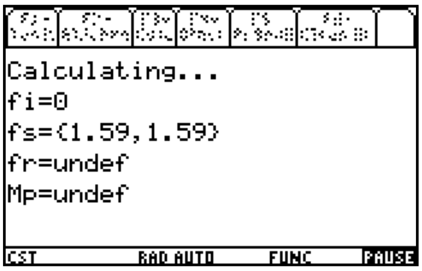


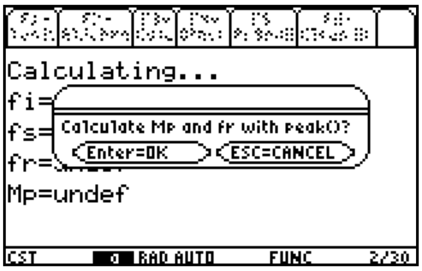
We can calculate the step response using *step(SYS)*, then we could calculate the relating parameters using function *pstep(SYS)*.

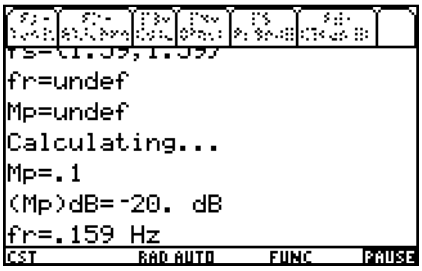




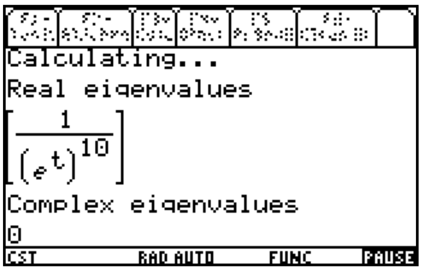
To get steady state we'll use function *trim(A,B,C, D,u_0)*. Notice that x0 and y0 matches with $w_{-1}(t)$.

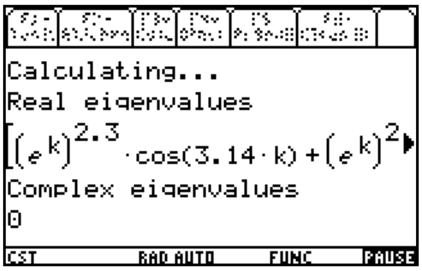




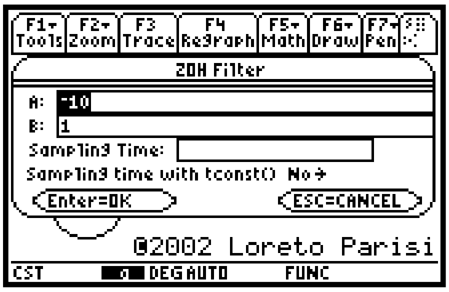


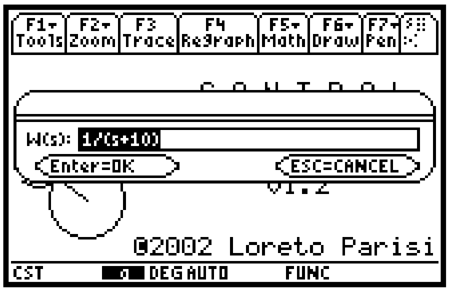
We have calculated the bandwith choosing *band(SYS)* from *Tools* menu (*F5)*. Notice that M_P and f_r results from *peak(SYS)* when *band(SYS)* fails.

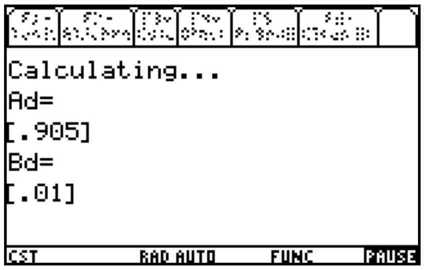




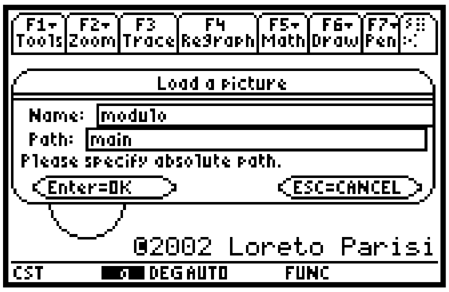
With *spectre*(A) we'll obtain e^{At} and A^K. Because of real pole, we have not complex expansion.

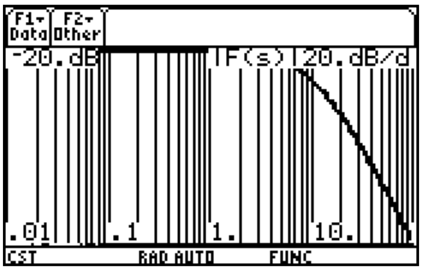


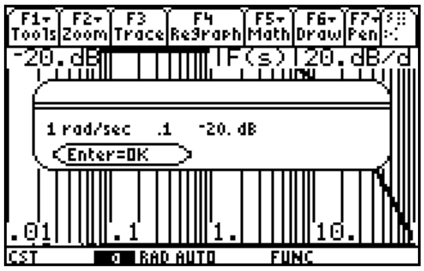




Here we have obtained discrete model of $\frac{1}{s+10}$, using ZOH method. Notice that only if we use *tconst(SYS)* we need to specify transfer function (this because we would like to work with a transfer function different from the current one).



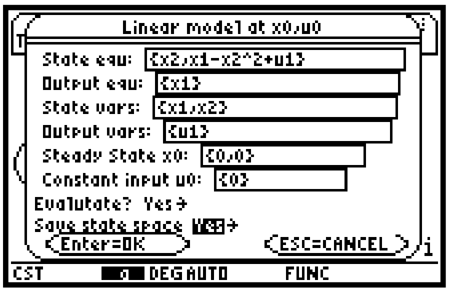


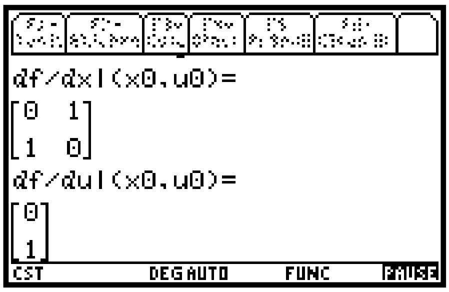


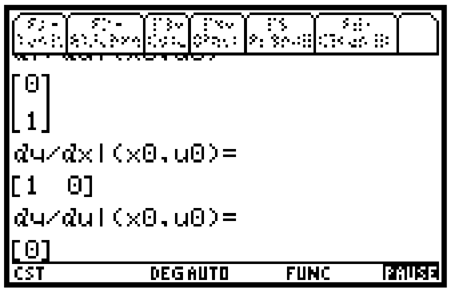
From *Load* menu (*F1)* we'll choose *Picture* to load previous magnitude Bode plot. Here we can evalutate magnitude (or phase) at ω_0 choosing from *Data* menu (*F1)* function *mag(SYS,ω_0)*.

Example 2. *Linearization*

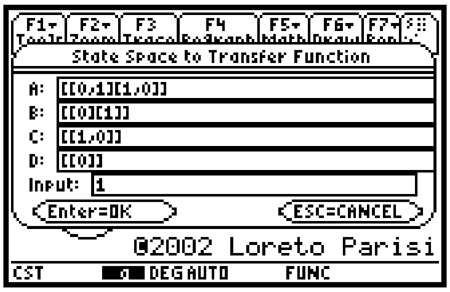
Now consider a non linear model: $\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_1 - x_2^{\,2} + u \\ y = x_1 \end{cases}$. To work with it, we need to linearize around a steady state x_0 relating to constant input u_0. First, we have to calculate the steady state x_0. It results: $x_0=\begin{pmatrix}0\\0\end{pmatrix}$ for input u_0=0. Now in *CST* we have:

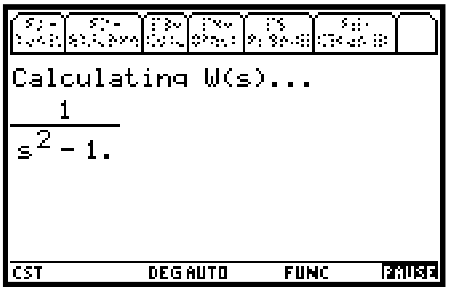


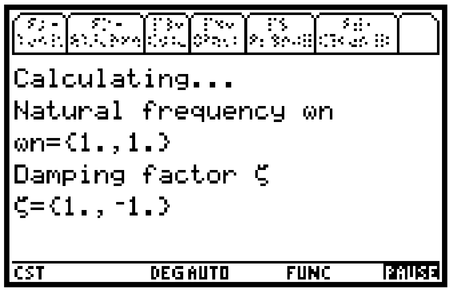




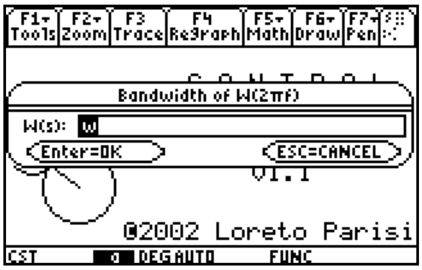
From *Dynamics* menu (*F3)* we choose *linmod*(f,y,x,u,x_0,u_0) where we have f={x_2,x_1-$x_2^{\,2}$+u_1}, y={x_1}, x={x_1,x_2}, u={u_1}, x_0={0,0} and u_0={0}. We have decided to evalutate jacobian matrixex in x_0 and u_0 to save the state-space of obtained linearized model. Now, we can get transfer function of this new model, that is a approximation of non-linear model above around x_0 and u_0:

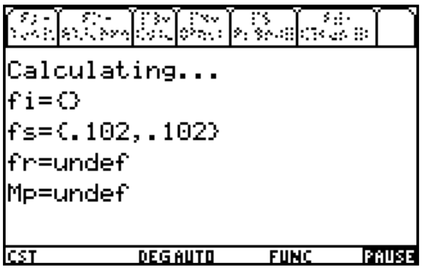


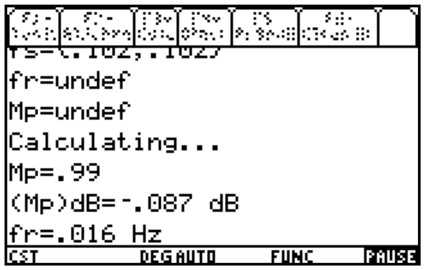




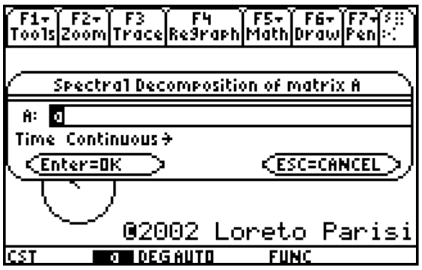
For this linear model, we'll get natural frequencies ω_{nh} and damping factors ζ_{nh} using function *damp(SYS)* from *Analysis* menu (*F2*). Now we'll use *band(SYS)*:

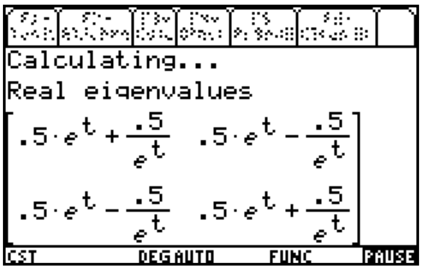


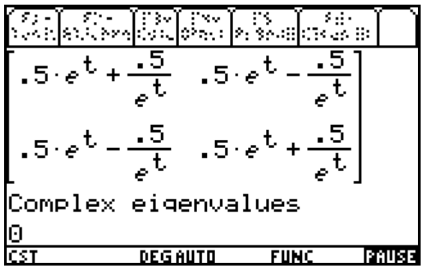




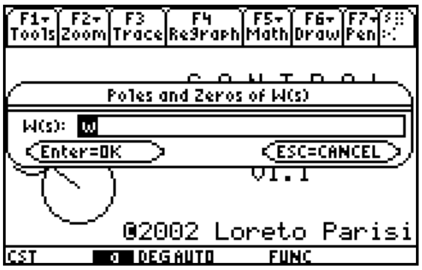
Now we'll use *spectre(SYS)* from *Tools* menu (*F5*) to calculate the spectral decomposition of matrix A:

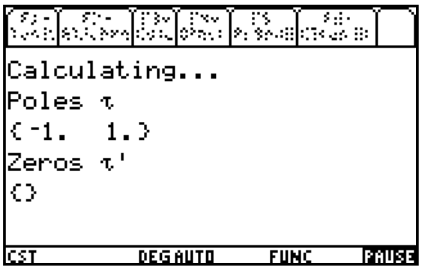






To verify if the model is stable we can simply calculate poles of transfer function with *pzmap(SYS)*:

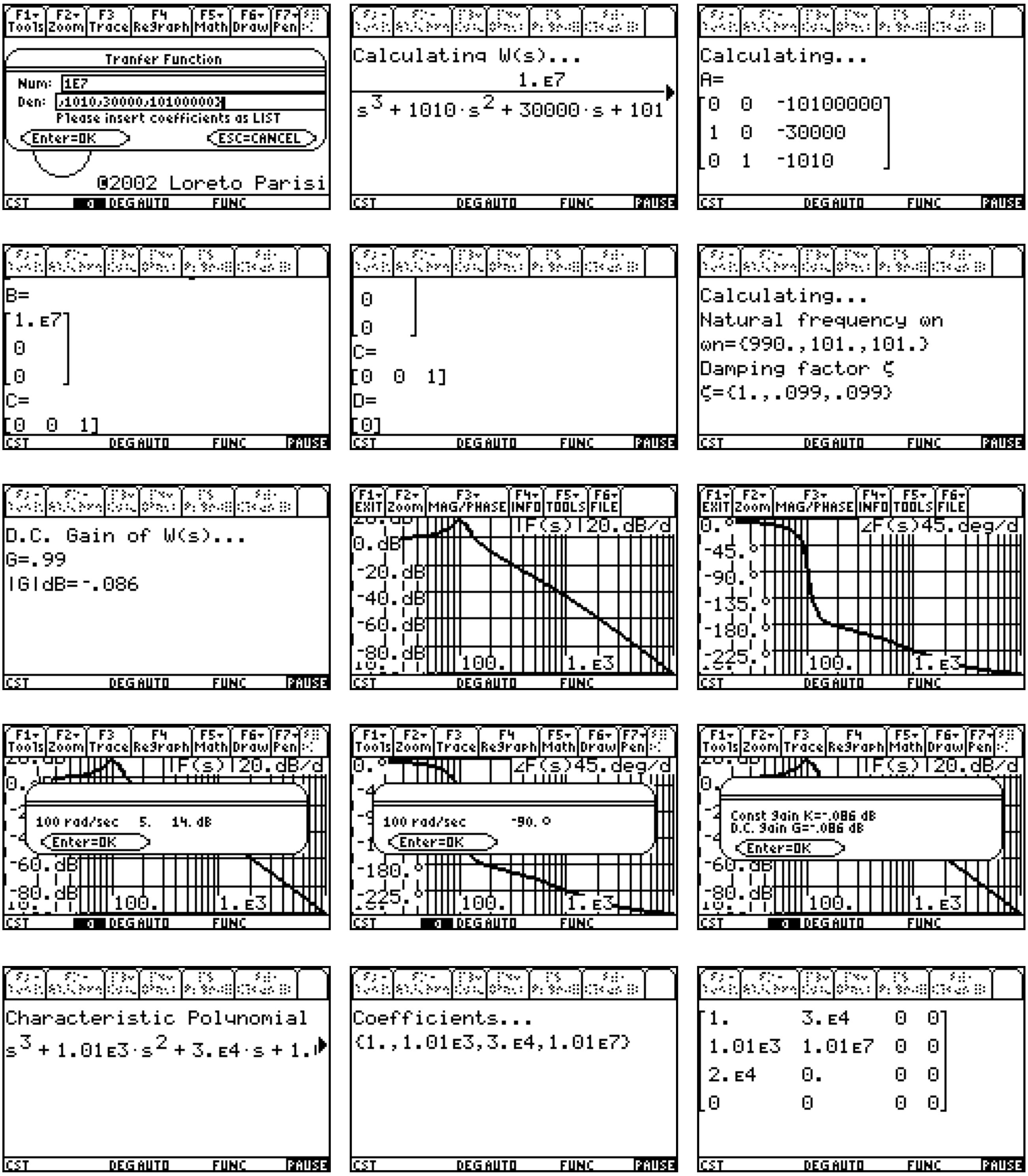


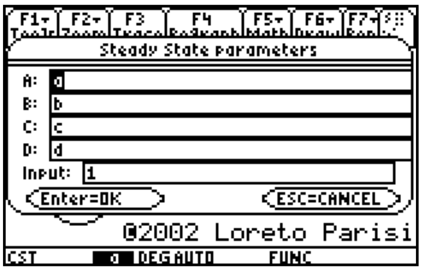


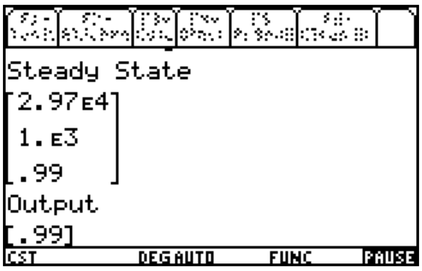
As we can see, there is a positive pole who causes instability of the model.

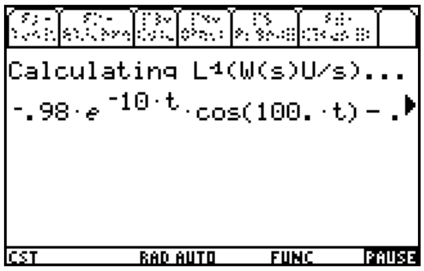
Example 3. *3rd order LPF*

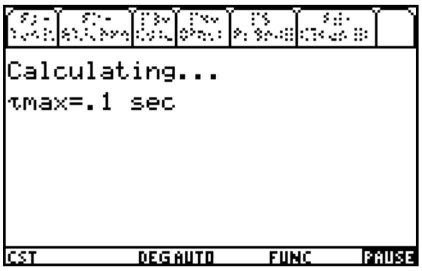
Consider a 3rd order low pass filter $W(s)=\dfrac{10000000}{s^3+1010s^2+30000s+10100000}$

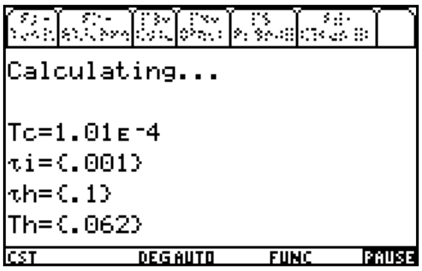


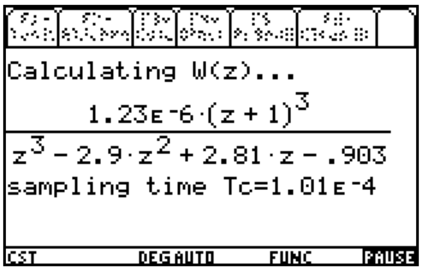

Steady State parameters
A: a
B: b
C: c
D: d
Input: 1
Enter=OK
ESC=CANCEL
©2002 Loreto Parisi
CST DEG AUTO FUNC

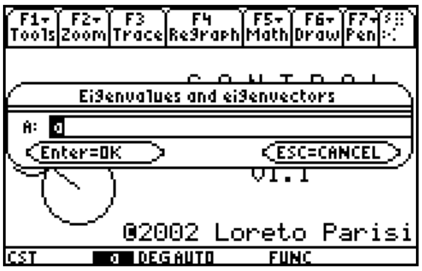

Steady State
2.97E4
1.E3
.99
Output
[.99]
CST DEG AUTO FUNC PAUSE

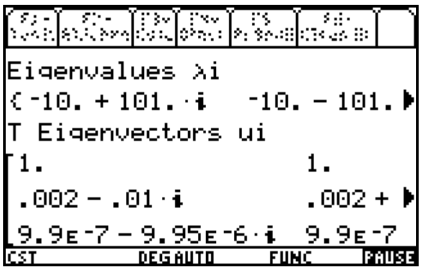

Calculating L⁻¹(W(s)U/s)...
-.98·e^-10·t·cos(100.·t)-.
CST RAD AUTO FUNC PAUSE

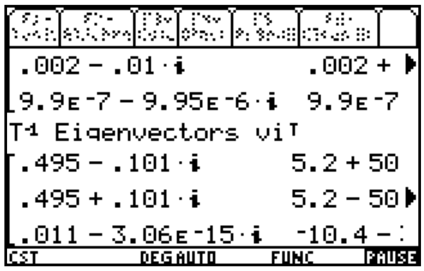

Calculating...
τmax=.1 sec
CST DEG AUTO FUNC PAUSE


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


Calculating W(z)...
1.23E-6·(z+1)^3
z^3-2.9·z^2+2.81·z-.903
sampling time Tc=1.01E-4
CST DEG AUTO FUNC PAUSE

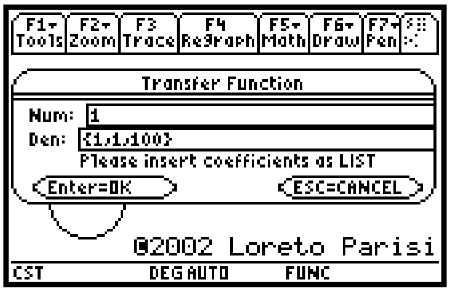

F1 Tools F2 Zoom F3 Trace F4 ReGraph F5 Math F6 Draw F7 Pen
Eigenvalues and eigenvectors
A: a
Enter=OK
ESC=CANCEL
V1.1
©2002 Loreto Parisi
CST DEG AUTO FUNC

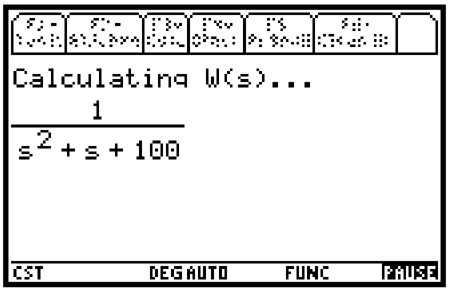

Eigenvalues λi
{-10.+101.·i -10.-101.
T Eigenvectors ui
1. 1.
.002-.01·i .002+
9.9E-7-9.95E-6·i 9.9E-7
CST DEG AUTO FUNC PAUSE

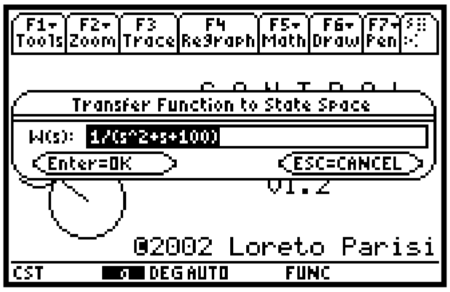

.002-.01·i .002+
9.9E-7-9.95E-6·i 9.9E-7
T⁻¹ Eigenvectors viᵀ
.495-.101·i 5.2+50
.495+.101·i 5.2-50
.011-3.06E-15·i -10.4-
CST DEG AUTO FUNC PAUSE

Example 4. *2^{nd} order LPF*

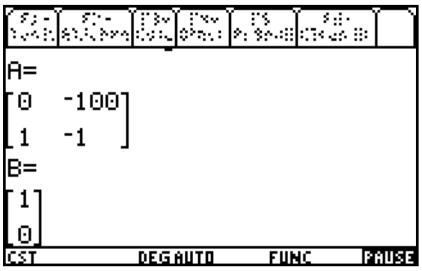
Consider a 2^{nd} order low pass filter $W(s)=\dfrac{1}{s^2+s+100}$;

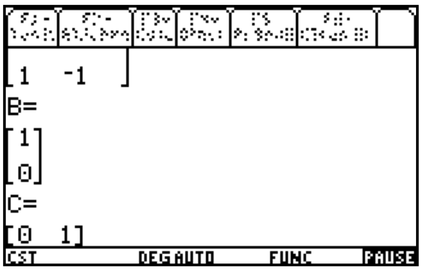


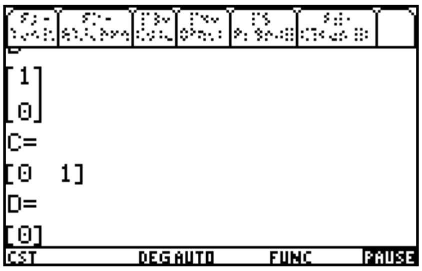




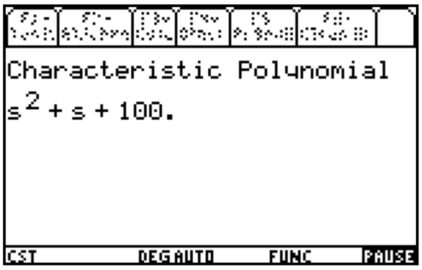
First, we'll define transfer function and calculates state-space representation:

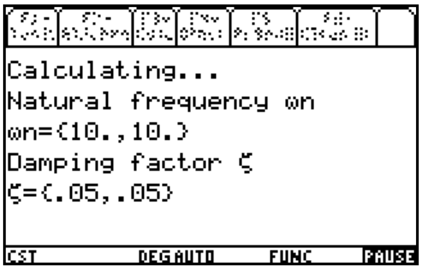




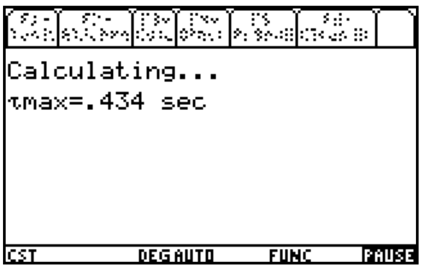


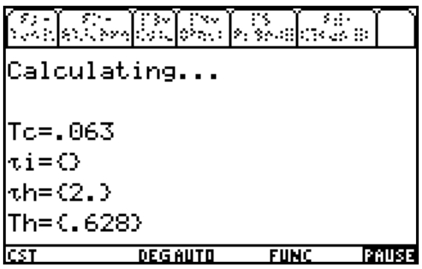
Now we'll get the characteristic polynomial, natural frequency and damping factor:

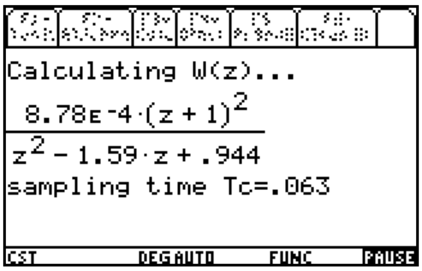




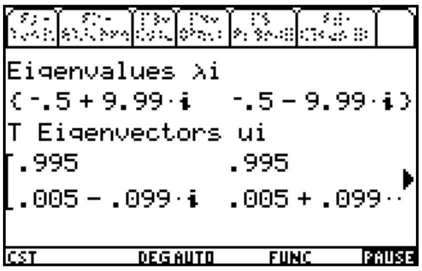
Then we'll calculate maximum time constant, sample time and other time constants, and the discrete model relating to previous sample time with Tustin method:

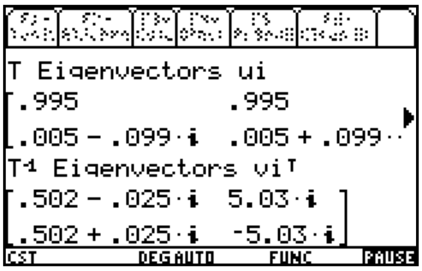




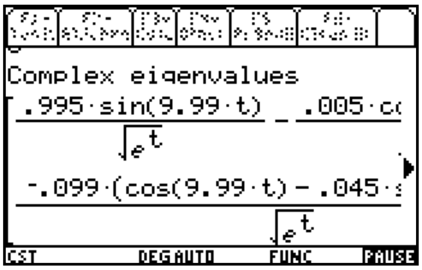


Eigenvalues and eigenvector :

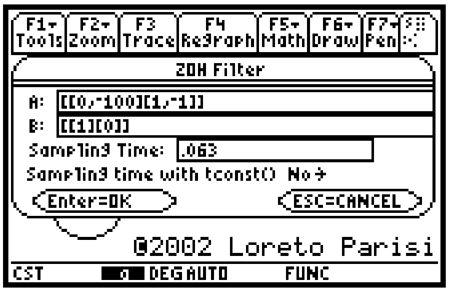


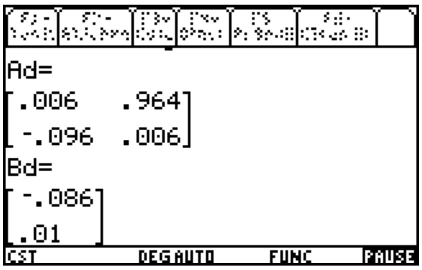


Spectral decomposition of dynamics matrix A:

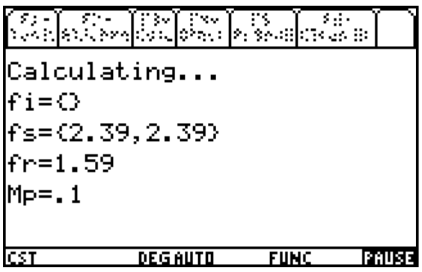


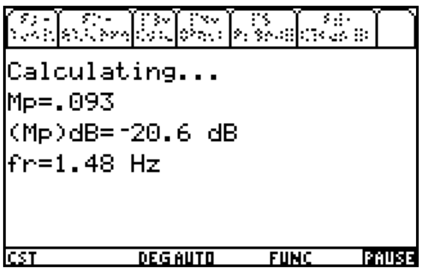
Discrete model with ZOH method:

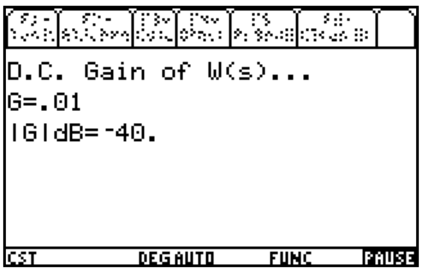




Bandwidth, resonance peak and d.c. gain:

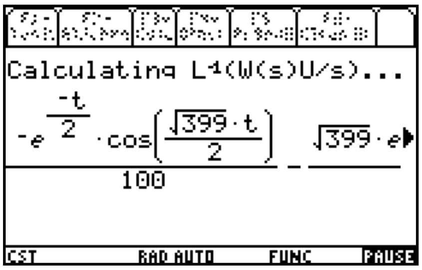




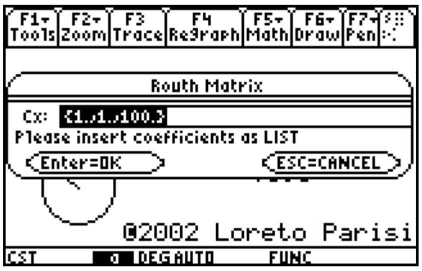


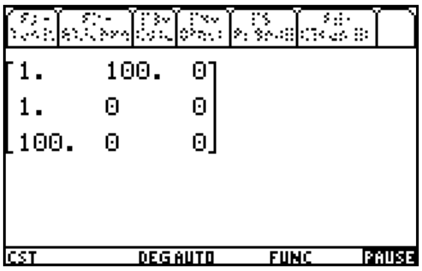
Notice that *band(SYS)* and *peak(SYS)* sometimes differs in values of M_p and f_r , when the first function uses formulas instead of numerical algorithm of *peak(SYS)*.

Step response:

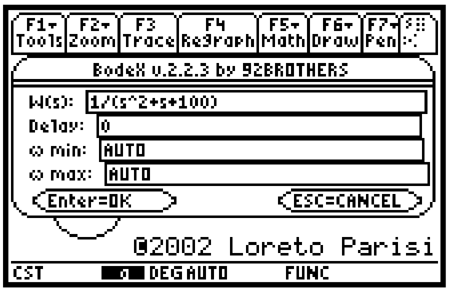


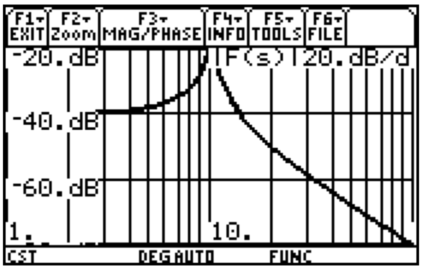
Routh matrix:

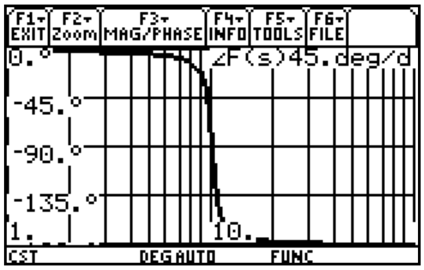


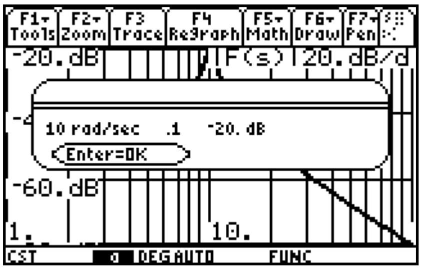


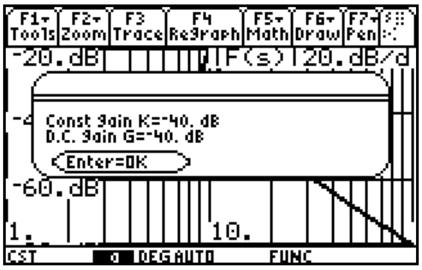
Bode plots:

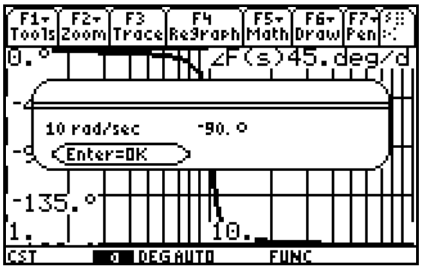






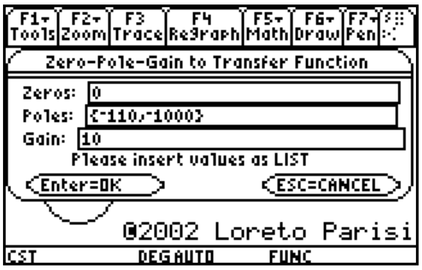


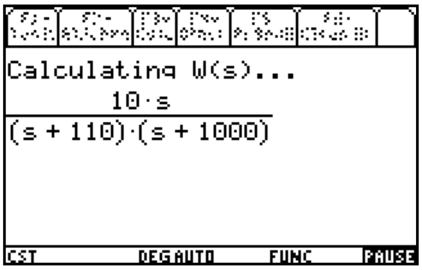




Example 5. 2nd order BPF

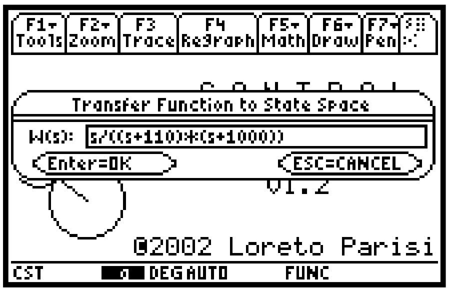
Consider a 2nd order band pass filter W(s)=$\frac{10s}{(s+110)(s+1000)}$. We can assign this transfer function, using function *zpk(zeros,poles,gain)* in this way:

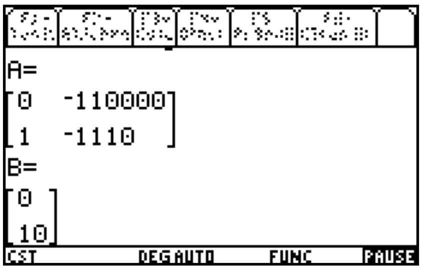


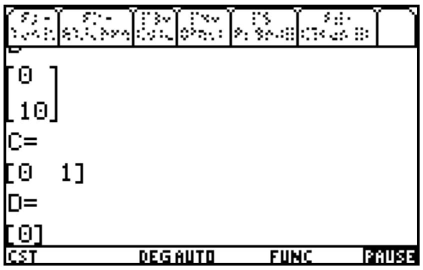


Zeros=0
Poles={-110,-1000}
Gain=10

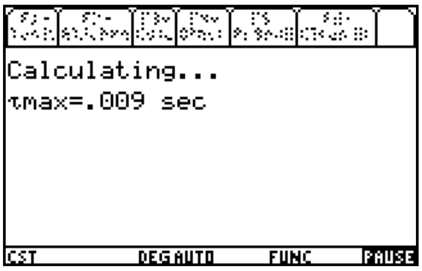
Now we need the state-space representation, so we use *tf2ss(SYS)*:

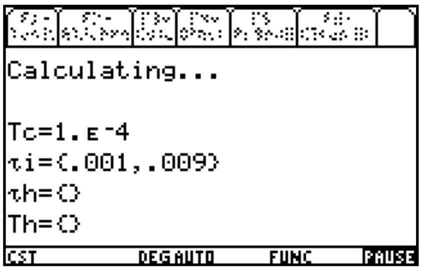


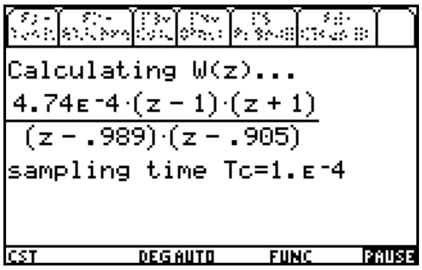




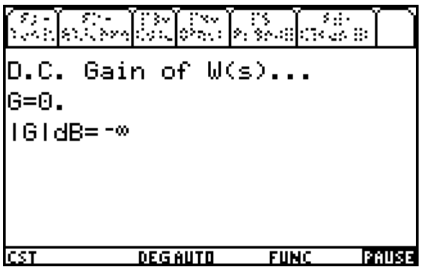
We can get the relating discrete model with *c2d(SYS,T_c)*, but we could need to obtain time constants too, so we use *tconst(SYS)* and *τmax(A)*:

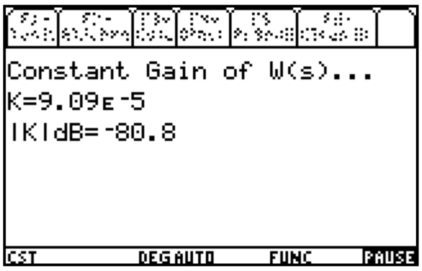




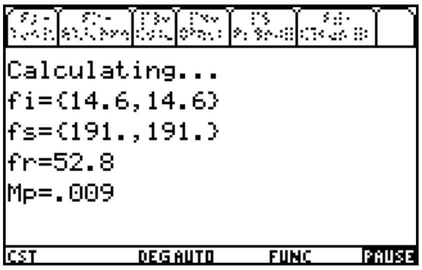


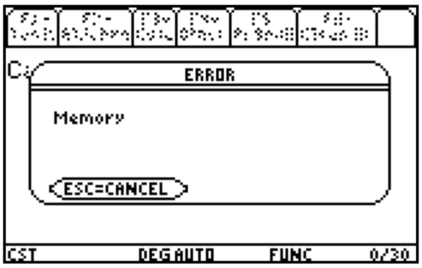
Before plotting Bode diagrams, we could need G and bandwidth of this bpf, so we use *dcgain(SYS)* and *band(SYS)*. We use *peak(SYS)* too, because of numerical algorithm, we could find some differences between values of resonance peak and frequency calculated with those two functions:





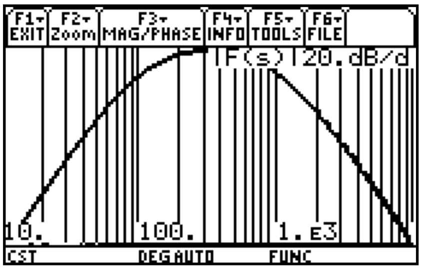
Notice that G is 0, so we have to use *gain(SYS)* to calculate constant gain K which results of –80dB.

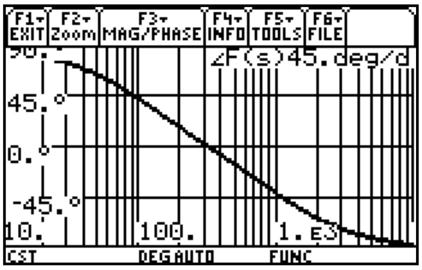


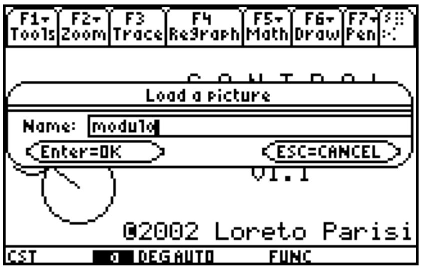


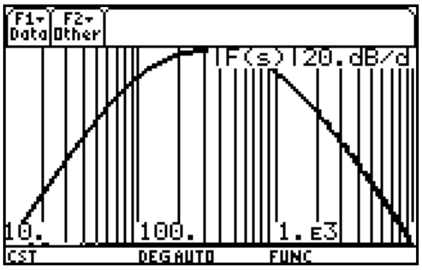
Notice that f_i and f_s have two values, relating to defect and excess approximations. In this case *band(SYS)* has used formulas.

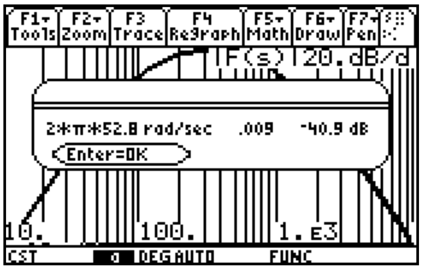
Peak(SYS) went to error because of lack of memory. This is due to too far away poles (one decade). It's our fault! We are working to this trouble and encouraging suggestions. Anyhow, we have obtained the bandwidth and peak. Now let's calculate Bode plots:





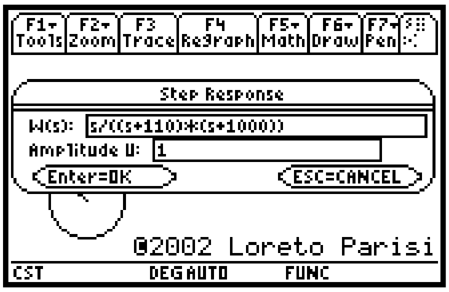


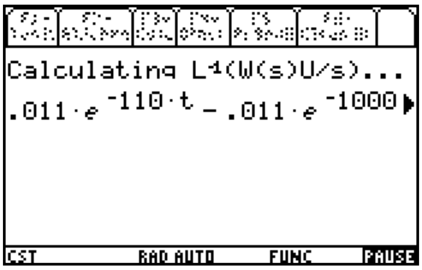




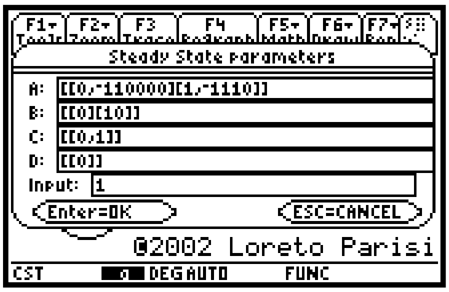
As we can see the magnitude at $2\pi f_r$ is M_p.

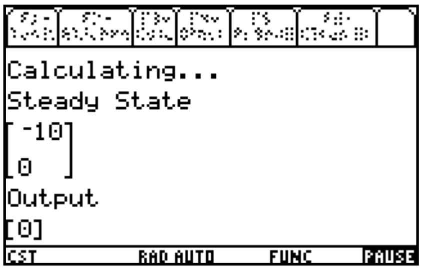
Now we'll calculate the step response with *step(SYS)*:



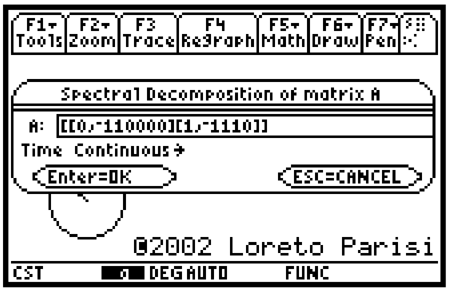


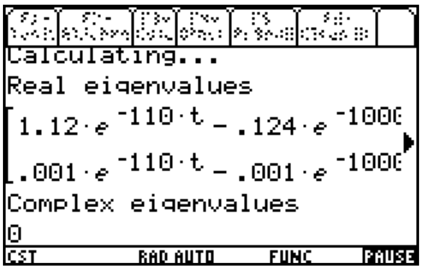
and the steady state with *trim(A,B,C,D,u_0)*:



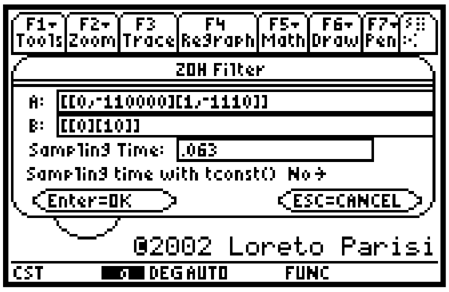


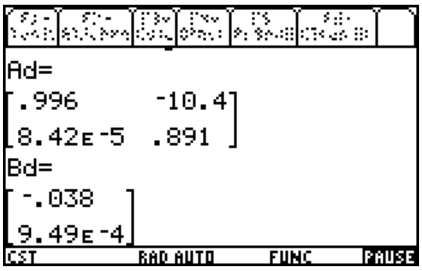
With *spectre(A)* we can obtain e^{At}:





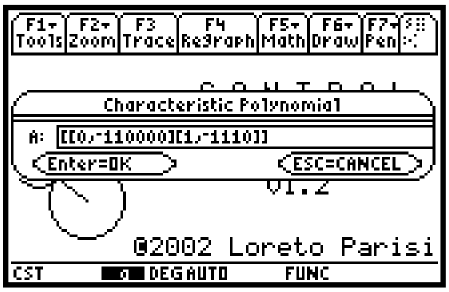
Now we uses *sampler(A,B,T_c)* to obtain the discretization of our filter as a digital filter using ZOH method:

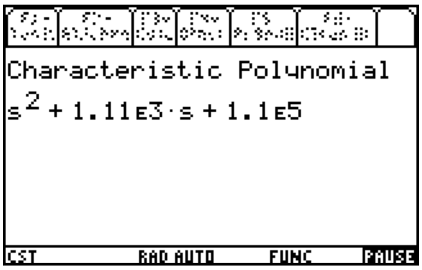




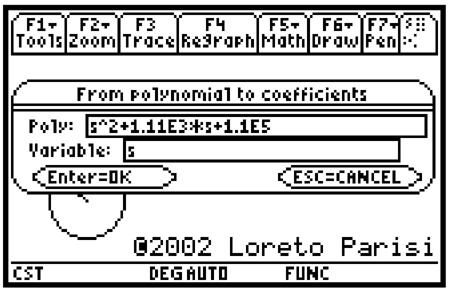
Notice that we have let *tconst(SYS)* to calculate the sample time T_c used in discretization.

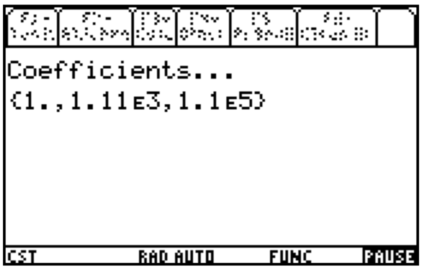
Finally, let's verify stability of this linear system. We'll use function *routh(Cx)*, but we need characteristic polynomial coefficients Cx, so we follow this way:



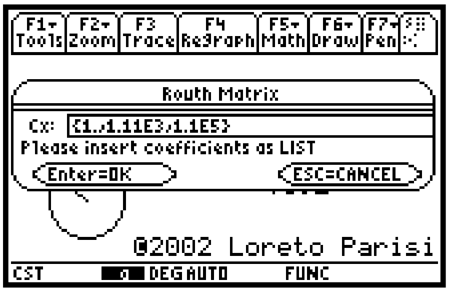


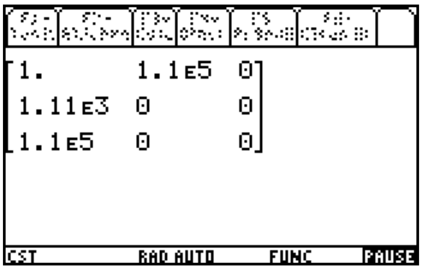
First, get p(s) with *poly(A)* from *Analysis* menu (*F2*).





Second, get coefficients using function *poly2cof(expr, var)* from *Tools* menu (*F5*). These will appear in the next dialog window.





Here is Routh matrix. All elements of first column are positive, so stabily is assured.

Contents

After installation of *CST*, in folder CST, you will find the following files:

File name	*File type*
azeros	FUNC
band	FUNC
bandn	FUNC
bandsub	FUNC
bodex	PRGM
c2d	FUNC
cpoles	FUNC
cst	PRGM
d2c	FUNC
damp	FUNC
db	FUNC
dcgain	FUNC
degzero	FUNC
eigenv	FUNC
gain	FUNC
graphf	PRGM
help	PRGM
install	PRGM
linmod	FUNC
linspace	FUNC
logspace	FUNC
mag	FUNC
magz	FUNC
mreach	FUNC
peak	FUNC

File name	*File type*
phase	FUNC
phasez	FUNC
poly	FUNC
poly2cof	FUNC
polydeg	FUNC
polyz2s	FUNC
pstep	FUNC
pzmap	FUNC
reach	FUNC
routh	FUNC
rts2poly	FUNC
sampler	FUNC
spectre	FUNC
splash	PIC
splhlp	PIC
ss2tf	FUNC
step	FUNC
tconst	FUNC
tf	FUNC
tf2ss	FUNC
trim	FUNC
zpk	FUNC
tmmax	FUNC

All these files are necessary to *cst()* to work properly. Please do not rename,move or delete any of the files above, otherwise the correct working of *cst() is* not guaranteed. Notice that *cst()* creates some variables in *CST* folder. These variable are *a*,*b*,*c*,*d* and contains state-space representation. We let them as global variables for your convenience.

Thanks to…

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

The programmers:

- *92BROTHERS*
Contribute: bodex()
E-mail: 92brothers@infinito.it
Homepage: http://www.92brothers.net/

- *Francesco Orabona*
Contribute: logspace(), poly2cof(), zpk()
E-mail: bremen79@infinito.it
Homepage: http://web.genie.it/utenti/b/bremen79/

- *Lars Frederiksen*
Contribute: DiffEq()
E-mail: ltf@post8.tele.dk

- *Chadd L. Easterday*
Contribute: rts2poly()
Email: easterday@mindspring.com

Thanks all folks!

What next ?

Next versions of Control System Toolbox *for TI-89* will improve global performances and will introduce some news, such as

- New proprietary function to plot Bode diagrams, *bode(SYS)*
- Improved function for calculating bandwidth, *band(SYS)*
- Improved function for calculating peak, *peak(SYS)*
- New functions for system interconnections, *series(),parallel(),feedback()*
- New function to calculate time delay, *delay(SYS)*
- More to come…

If you have any question and suggestion or you would like to help me, please let me know, sending e-mail at

loretoparisi@libero.it