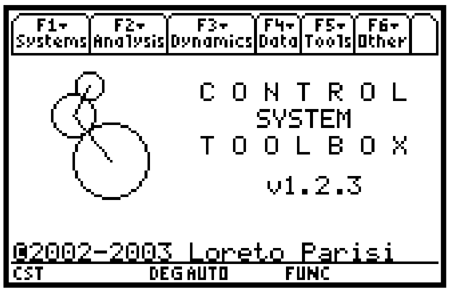
Control System Toolbox

for TI-89



release 1.2.3

©2002-2003 Loreto Parisi

The CST Reference Guide

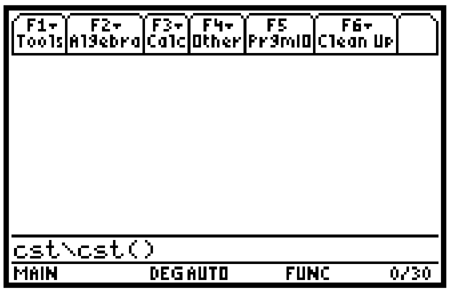
Fourth edtion October 2003

Index **Page**

Fun Index

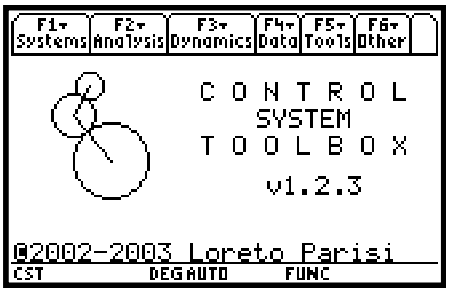
What's Control System Toolbox *for TI-89*

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.



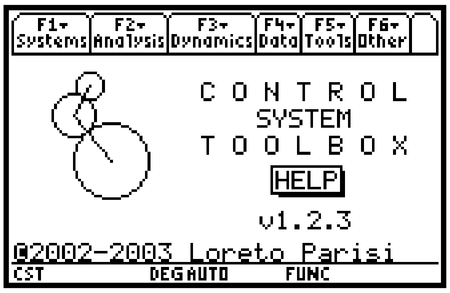
```
cst\cst()
```

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.



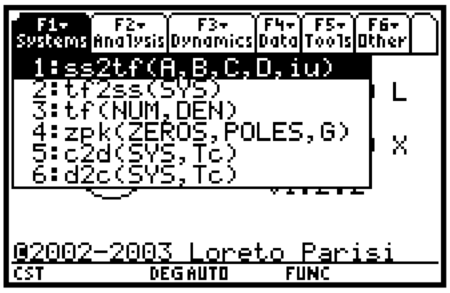
```
C O N T R O L
SYSTEM
T O O L B O X
v1.2.3
©2002-2003 Loreto Parisi
```

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.



```
C O N T R O L
SYSTEM
T O O L B O X
v1.2.3
©2002-2003 Loreto Parisi
```

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

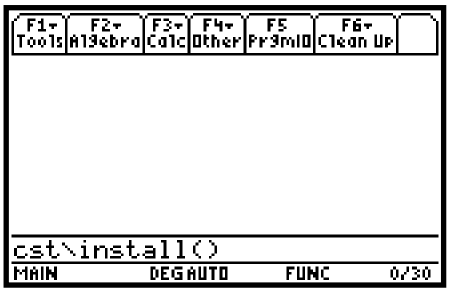
http://web.tiscali.it/loretoparisihome

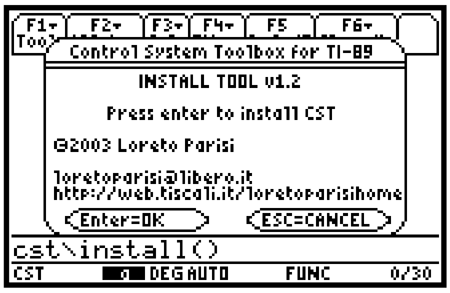
You can send me feedback from here:

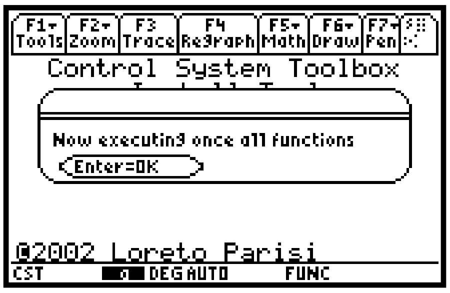
http://web.tiscali.it/loretoparisihome/feedback

How To Install

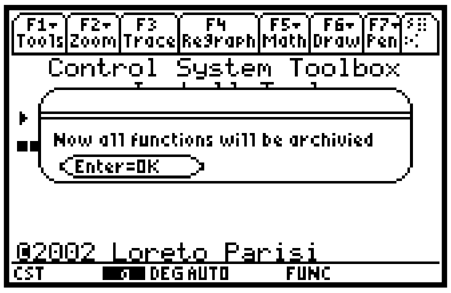
Use your linking software to send the program *CSTxxx.89G* on the calculator. All the files are automatically placed in the folder CST. Once installation has occurred, do not move, delete, or rename any of the functions and programs or pictures in the folder CST. All files included in folder CST are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

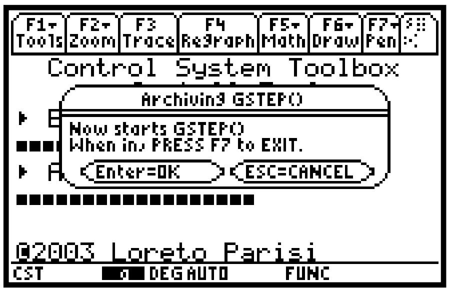

```
cst\install()
MAIN    DEG AUTO    FUNC    0/30
```

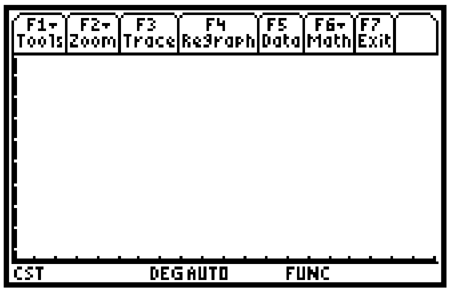

```
Control System Toolbox for TI-89
INSTALL TOOL v1.2
Press enter to install CST
©2003 Loreto Parisi
loretoparisi@libero.it
http://web.tiscali.it/loretoparisihome
cst\install()
CST    DEG AUTO    FUNC    0/30
```


```
Control System Toolbox
Now executing once all functions
©2002 Loreto Parisi
CST    DEG AUTO    FUNC
```

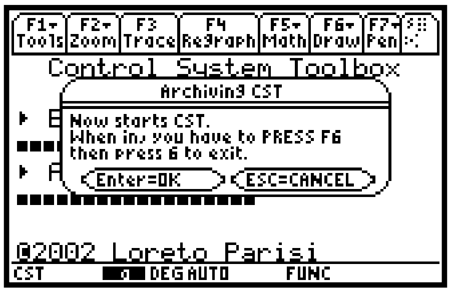
Run *CST\install()* from HOME. First *install*() will execute once all functions before archiving them. This tip will quitely increase execution speed.

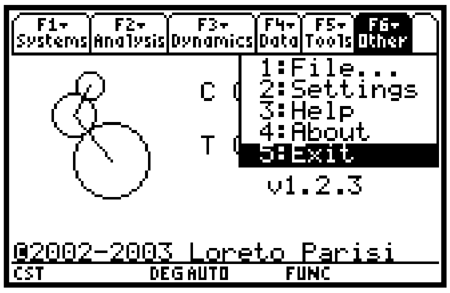

```
Control System Toolbox
Now all functions will be archivied
©2002 Loreto Parisi
CST    DEG AUTO    FUNC
```

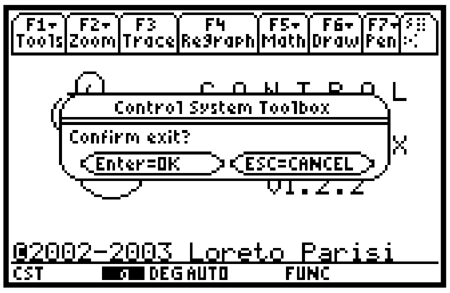

```
Control System Toolbox
Archiving GSTEP()
Now starts GSTEP()
When in, PRESS F7 to EXIT.
©2003 Loreto Parisi
CST    DEG AUTO    FUNC
```

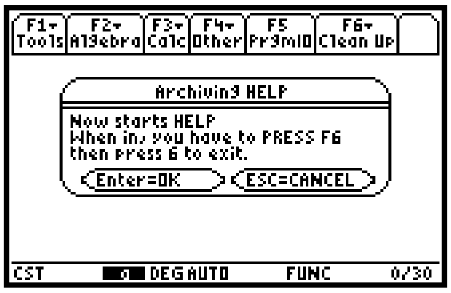

```
CST    DEG AUTO    FUNC
```

After archiving all functions, *install()* will run *gstep()* once before archiving it, so you have to exit it pressing F7. The it will run *cst()* once before archiving it, so you have to exit it pressing F6 then 5, or using arrow keys and select *exit* from *Other* menu (*F6*).


```
Control System Toolbox
Archiving CST
Now starts CST.
When in, you have to PRESS F6
then press 6 to exit.
©2002 Loreto Parisi
CST    DEG AUTO    FUNC
```

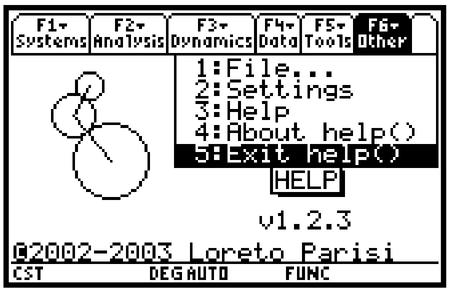

```
v1.2.3
©2002-2003 Loreto Parisi
CST    DEG AUTO    FUNC
```

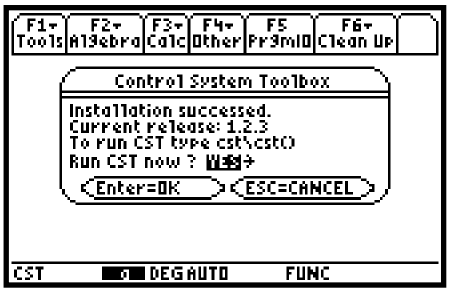

```
Control System Toolbox
Confirm exit?
©2002-2003 Loreto Parisi
CST    DEG AUTO    FUNC
```



Archiving HELP

Now starts HELP
When in, you have to PRESS F6
then press 6 to exit.





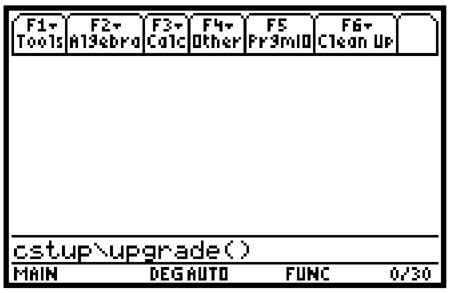
Control System Toolbox

Installation successed.
Current release: 1.2.3
To run CST type cst\cst()
Run CST now ? YES→

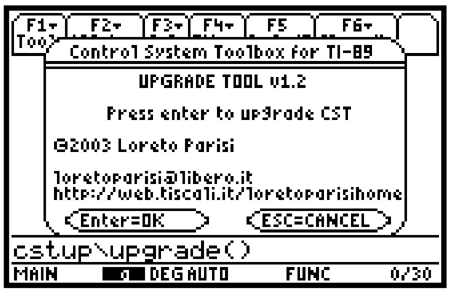
To exit *help()* press *F6* then *5* or select *exit help()* from *Other* menu (*F6)*. After installation you can delete *install()* from *CST* folder. To run CST type *CST\cst()* from HOME.

How to Upgrade

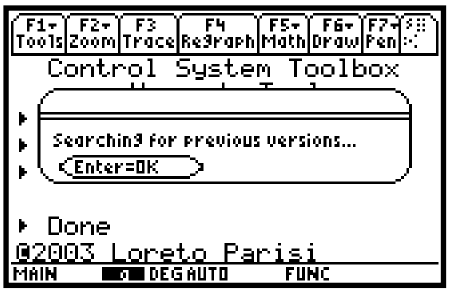
To upgrade CST, first send the upgrade group file CSTUPxxx.89G to the calculator. A folder named CSTUP will appear in VAR-LINK menu. Please select and load the program *upgrade()* from this folder. This will run the upgrade tool.



```
cstup\upgrade()
MAIN        DEG AUTO      FUNC      0/30
```

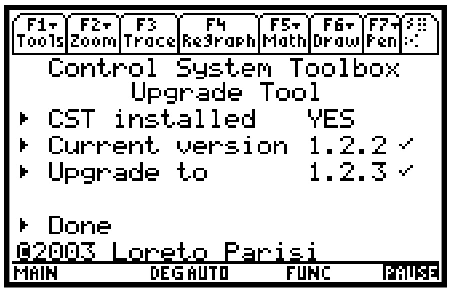


```
Control System Toolbox for TI-89
UPGRADE TOOL v1.2
Press enter to upgrade CST
©2003 Loreto Parisi
loretoparisi@libero.it
http://web.tiscali.it/loretoparisihome
Enter=OK      ESC=CANCEL
cstup\upgrade()
MAIN        DEG AUTO      FUNC      0/30
```

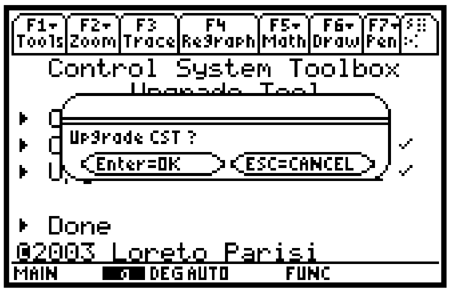


```
Control System Toolbox
Searching for previous versions...
Enter=OK
▸ Done
©2003 Loreto Parisi
MAIN        DEG AUTO      FUNC
```

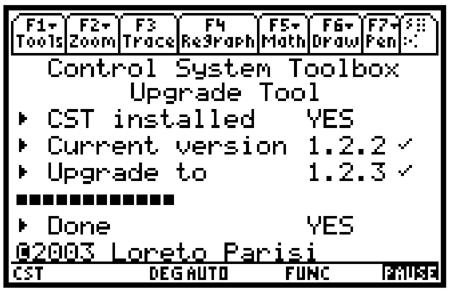
The program starts searching previous version of CST on the calculator and show the new upgrading version.If all informations are right, please select ENTER to upgrade.



```
Control System Toolbox
     Upgrade Tool
▸ CST installed    YES
▸ Current version  1.2.2 ✓
▸ Upgrade to       1.2.3 ✓

▸ Done
©2003 Loreto Parisi
MAIN        DEG AUTO      FUNC      PAUSE
```

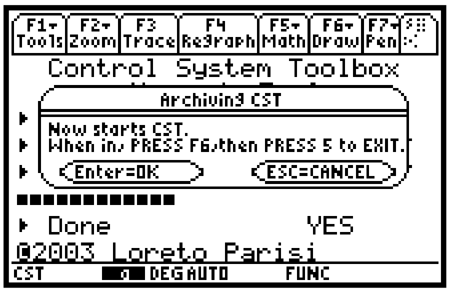


```
Control System Toolbox
Upgrade CST ?
Enter=OK   ESC=CANCEL
▸ Done
©2003 Loreto Parisi
MAIN        DEG AUTO      FUNC
```

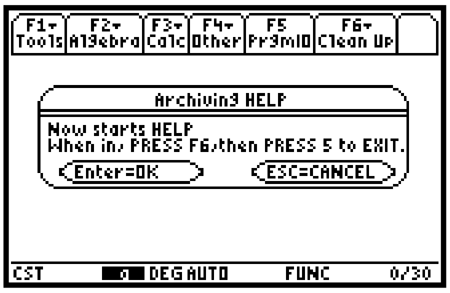


```
Control System Toolbox
     Upgrade Tool
▸ CST installed    YES
▸ Current version  1.2.2 ✓
▸ Upgrade to       1.2.3 ✓
■■■■■■■■■■■■■
▸ Done             YES
©2003 Loreto Parisi
CST         DEG AUTO      FUNC      PAUSE
```

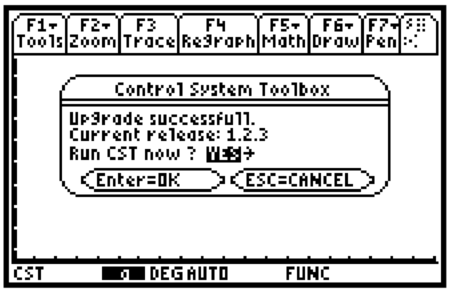
After upgrading, it it necessary to run once first archiving programs. So follow instruction to run *cst()*, *help()* and *gstep()* – this only for CSTUP123.89G – and exit these programs.



```
Control System Toolbox
Archiving CST
Now starts CST.
When in, PRESS F6,then PRESS 5 to EXIT.
Enter=OK      ESC=CANCEL
■■■■■■■■■■■■■
▸ Done             YES
©2003 Loreto Parisi
CST         DEG AUTO      FUNC
```



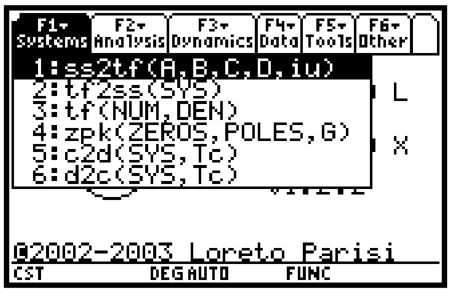
```
Archiving HELP
Now starts HELP
When in, PRESS F6,then PRESS 5 to EXIT.
Enter=OK      ESC=CANCEL
CST         DEG AUTO      FUNC      0/30
```



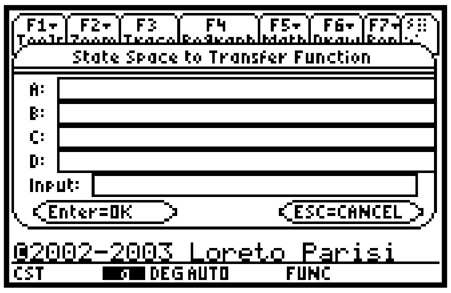
```
Control System Toolbox
Upgrade successfull.
Current release: 1.2.3
Run CST now ? YES→
Enter=OK   ESC=CANCEL
CST         DEG AUTO      FUNC
```

After upgrading, you can delete folder CSTUP and its contents. To run CST simply press ENTER or type *CST\cst()* from HOME.

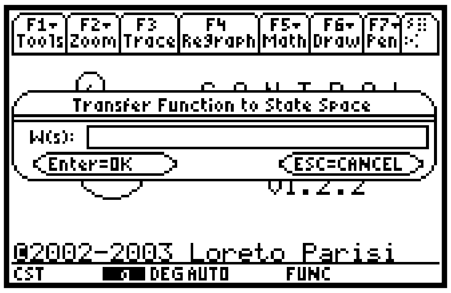
Systems



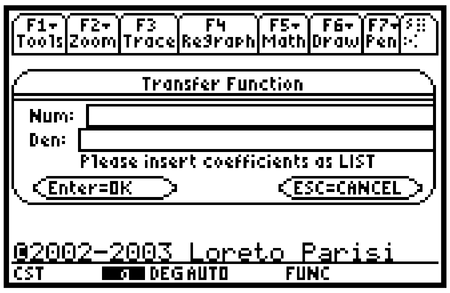
The *Systems* menu (*F1*) contains all the functions to build the model, using state-space or transfer function and to perform conversions from one representation to another, even from continuous time to discrete time model.



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).

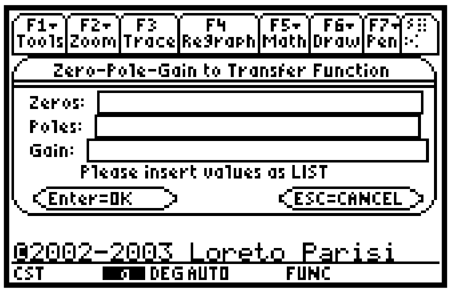


tf2ss(SYS)
Convert transfer function SYS in the state-space representation $\dot{x} = Ax + Bu$, $y = Cx + Du$, using the observability canonical form.

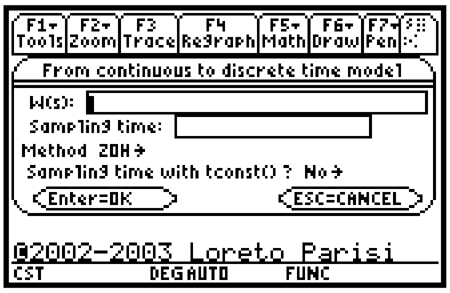


tf(NUM,DEN)
Calculates transfer function, where NUM and DEN are LIST of coefficients of numerator's and denominator's polynomial: NUM={b_0, b_1,…,b_n}, DEN={a_0,a_1,…a_n}, so

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

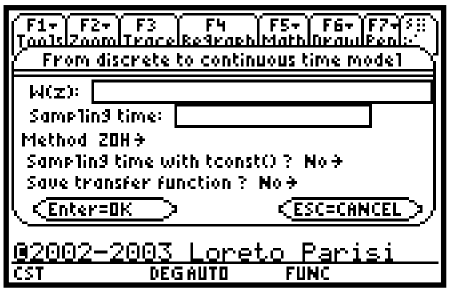


zpk(zeros,poles,gain)
Calculates transfer function W(s) in the zeros-poles-gain representation, where zeros, poles are LIST of zeros of numerator and denominator (poles), while gain is NUM and represents constant gain K.
Original version: *Francesco Orabona*
E-mail: bremen79@infinito.it
Homepage: http://web.genie.it/utenti/b/bremen79/



From continuous to discrete time model
W(s):
Sampling time:
Method ZOH
Sampling time with tconst() ? No
©2002-2003 Loreto Parisi

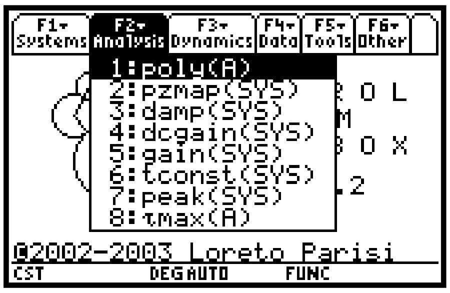
c2d(SYS,T_c)
Converts continuous time model SYS to the discrete time model, using sample time T_c and different methods: ZOH and FOH filters are implemented. Can use function *tconst(SYS)* to determinate sample time T_c. Use function sampler(A,B,T_c) to use ZOH method.



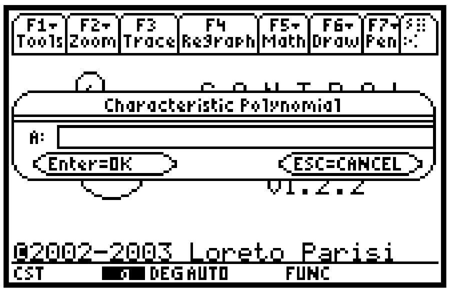
From discrete to continuous time model
W(z):
Sampling time:
Method ZOH
Sampling time with tconst() ? No
Save transfer function ? No
©2002-2003 Loreto Parisi

d2c(SYS,T_c)
Converts discrete time model SYS (in z) to the continuous time model, using sample time T_c and different methods: ZOH and FOH filters are implemented. Can use function *tconst(SYS)* to determinate sample time T_c. Can save the resulting continuous transfer function.

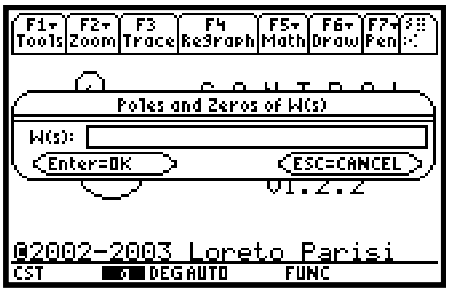
Analysis



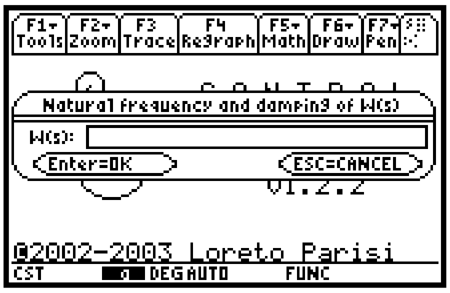
The *Analysis* menu (*F2*) contains all the tools to analyze the model you have created with *Systems'* tools. You can also analyze different models, using different SYS at time. This will not change current transfer function.



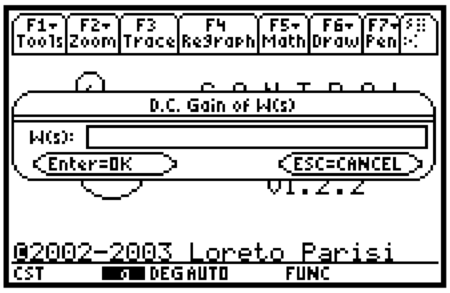
poly(A)
This function calculates characteristic polynomial of matrix A, as p(s)=|sI-A|, where |•| is determinat of a matrix.



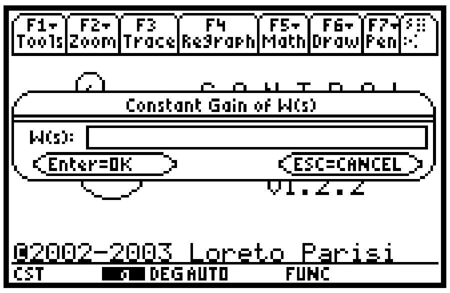
pzmap(SYS)
This function calculates poles and zeros of given transfer function SYS, where poles are zeros of denominator of SYS.



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



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



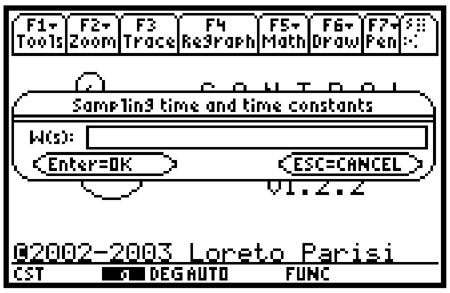
Constant Gain of W(s)

W(s):

v1.2.2

©2002-2003 Loreto Parisi

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.



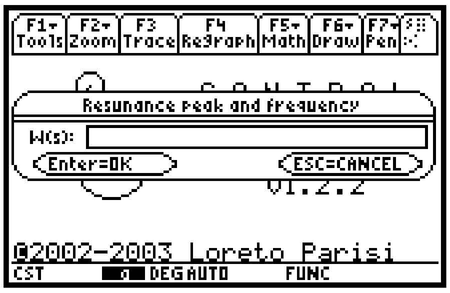
Sampling time and time constants

W(s):

v1.2.2

©2002-2003 Loreto Parisi

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 $T_c = 0.1\min\{\tau_i, \tau_h, T_h\}$.



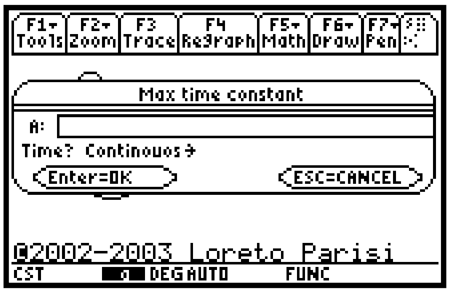
Resunance peak and frequency

W(s):

v1.2.2

©2002-2003 Loreto Parisi

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



Max time constant

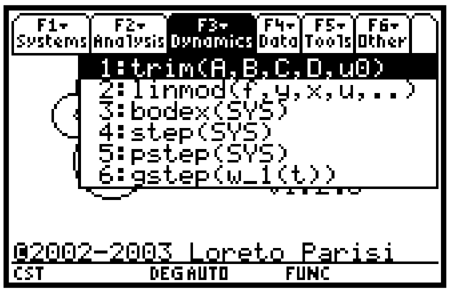
A:

Time? Continuos →

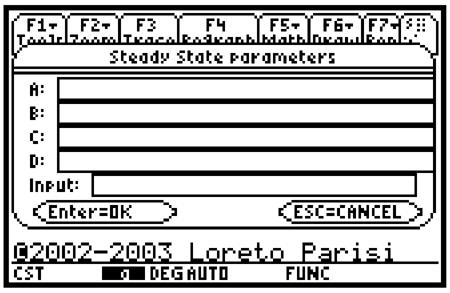
©2002-2003 Loreto Parisi

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



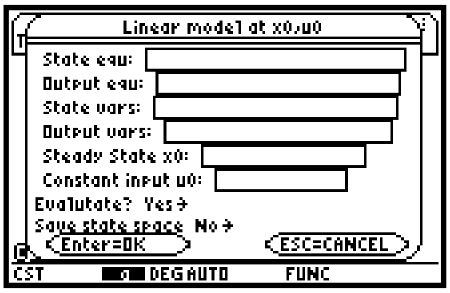
The *Dynamics* menu (*F3*)contains functions concerning dynamics of system for input, output and linearization of a non- linear model, and frequency analysis yet.



```
Steady State parameters
A:
B:
C:
D:
Input:
```

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

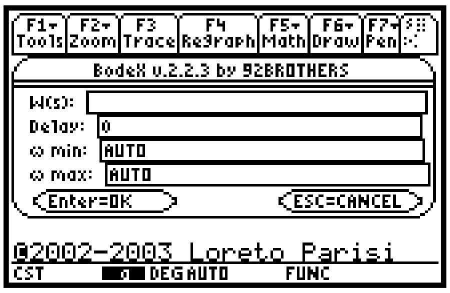
This function calculates the steady state x_0, relating to input u_0 for state-space $\dot{x} = Ax + Bu$, $y = Cx + Du$.



```
Linear model at x0,u0
State equ:
Output equ:
State vars:
Output vars:
Steady State x0:
Constant input u0:
Evalutate?  Yes→
Save state space  No→
```

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 $f=\{f_1(x,u),\ldots,f_n(x,u)\}$ and output equations y, such as $y=\{y_1(x,u),\ldots y_n(x,u)\}$, 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.

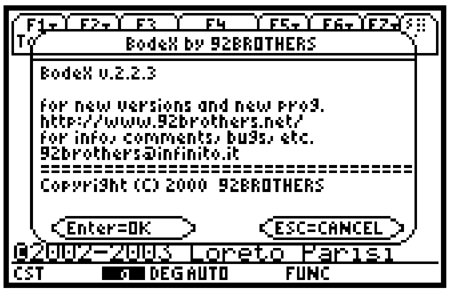


```
BodeX v.2.2.3 by 92BROTHERS
W(s):
Delay:  0
ω min:  AUTO
ω max:  AUTO
```

bodex(SYS)

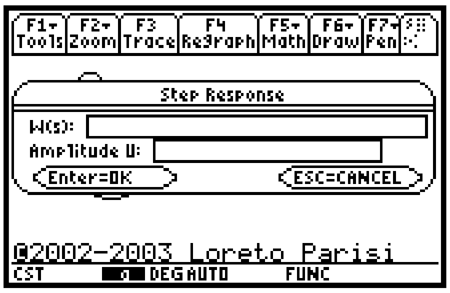
This program, made by 92BROTHERS, plots Bode diagrams of phase and magnitude and offers several tools to work with the plottoed diagrams.

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



```
BodeX by 92BROTHERS
BodeX v.2.2.3

for new versions and new prog.
http://www.92brothers.net/
for info, comments, bugs, etc.
92brothers@infinito.it
=====================================
Copyright (C) 2000  92BROTHERS
```

BodeX v.2.2.3
Copyright © 2000 92BROTHERS
For contacts please refers to:
e-mail: 92brothers@infinito.it
URL: http://www.92brothers.net/



Step Response

W(s):

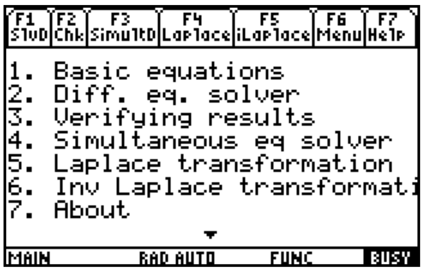
Amplitude U:

Enter=OK ESC=CANCEL

©2002-2003 Loreto Parisi

CST DEG AUTO FUNC

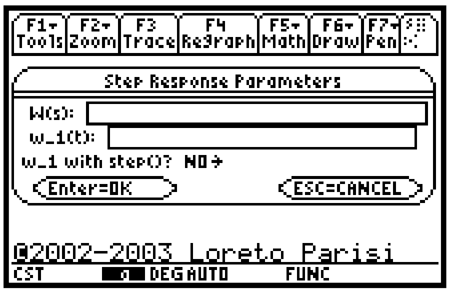
step(SYS)
This tool calculates the step response for SYS, as $U*w_{-1}(t) = L^{-1}(W(s)U/s)$, with amplitude U. It needs the tool DiffEq v. 2.04 or next by *Lars Frederiksen* to perform symbolic calculation of Laplace direct and inverse transformation.



1. Basic equations
2. Diff. eq. solver
3. Verifying results
4. Simultaneous eq solver
5. Laplace transformation
6. Inv Laplace transformati
7. About

MAIN RAD AUTO FUNC BUSY

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



Step Response Parameters

W(s):

w_1(t):

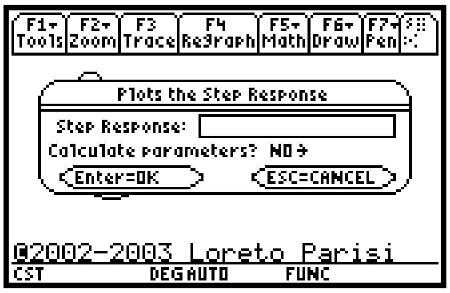
w_1 with step()? NO

Enter=OK ESC=CANCEL

©2002-2003 Loreto Parisi

CST DEG AUTO FUNC

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).



Plots the Step Response

Step Response:

Calculate parameters? NO

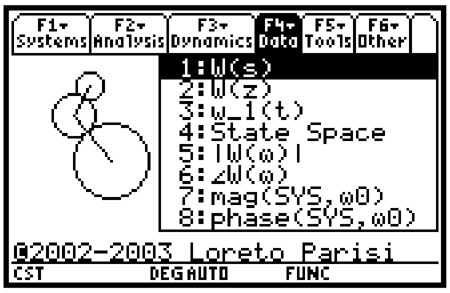
Enter=OK ESC=CANCEL

©2002-2003 Loreto Parisi

CST DEG AUTO FUNC

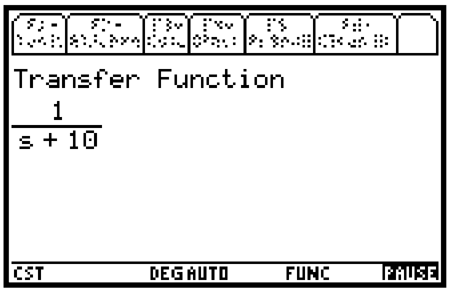
gstep(w_1(t))
This tool plots the step response w_1(t) calculated with *step(SYS)* or specified directly. Can use *pstep(SYS)* to evalutate w_1(t) in its typical parameters. Using parameters it applies the auto-scaling feature to obtain the correct zoom factor for the plot.

Data



```
1:W(s)
2:W(z)
3:w_1(t)
4:State Space
5:|W(ω)|
6:∠W(ω)
7:mag(SYS,ω0)
8:phase(SYS,ω0)
©2002-2003 Loreto Parisi
CST          DEG AUTO      FUNC
```

The *Data* menu (*F4*) gives access to current transfer function W(s), its discrete model W(z), the step response w-1(t), the current state space and magnitude and phase of W(s) or W(z).

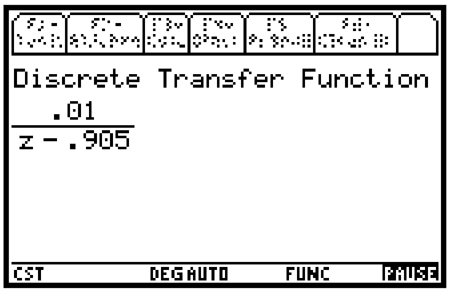


```
Transfer Function
```

$$\frac{1}{s+10}$$

```
CST          DEG AUTO      FUNC      PAUSE
```

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.

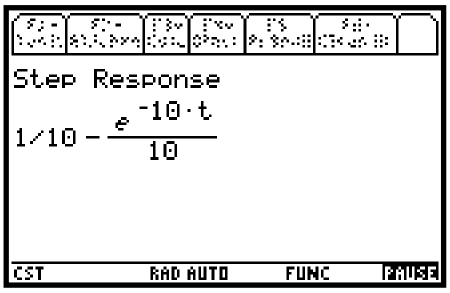


```
Discrete Transfer Function
```

$$\frac{.01}{z-.905}$$

```
CST          DEG AUTO      FUNC      PAUSE
```

W(z)
The discrete transfer function W(z), defined with c2d(SYS),Tc) using the current continuos transfer function W(s) or by *sampler(A,B,Tc)* using the current state space.

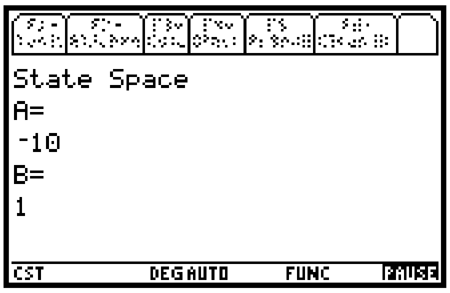


```
Step Response
```

$$1/10-\frac{e^{-10\cdot t}}{10}$$

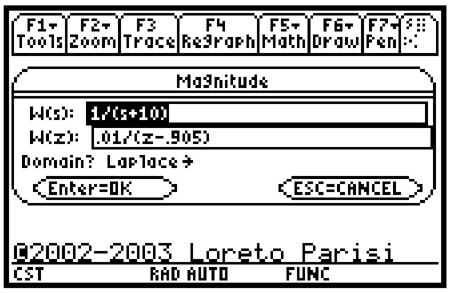
```
CST          RAD AUTO      FUNC      PAUSE
```

W_1(t)
Shows the current step response obtained with *step(SYS).*



```
State Space
A=
-10
B=
1
CST          DEG AUTO      FUNC      PAUSE
```

State Space
It displays the current state space, as defined from one of tools of *System* menu.



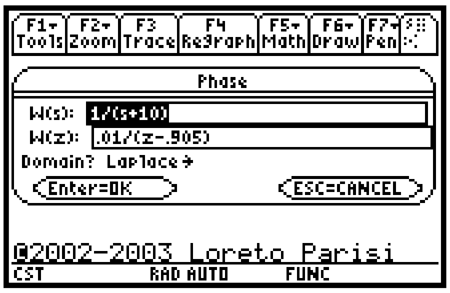
Magnitude

W(s): 1/(s+10)
W(z): .01/(z-.905)
Domain? Laplace→

©2002-2003 Loreto Parisi

|W(ω)|

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



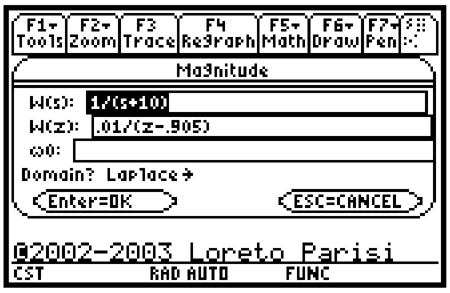
Phase

W(s): 1/(s+10)
W(z): .01/(z-.905)
Domain? Laplace→

©2002-2003 Loreto Parisi

∠W(ω)

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



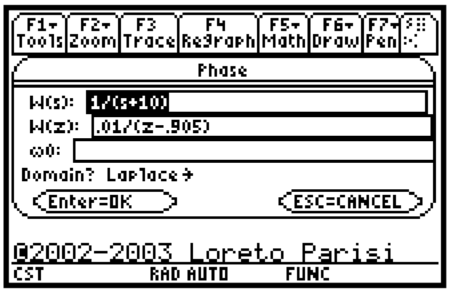
Magnitude

W(s): 1/(s+10)
W(z): .01/(z-.905)
ω0:
Domain? Laplace→

©2002-2003 Loreto Parisi

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

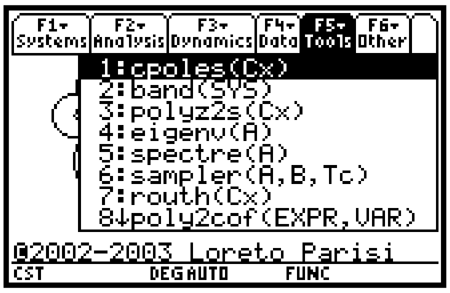
W(s): 1/(s+10)
W(z): .01/(z-.905)
ω0:
Domain? Laplace→

©2002-2003 Loreto Parisi

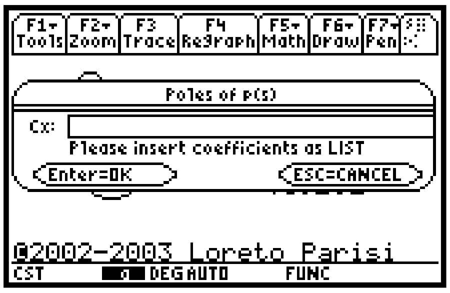
phase(SYS,ω$_0$)

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

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.

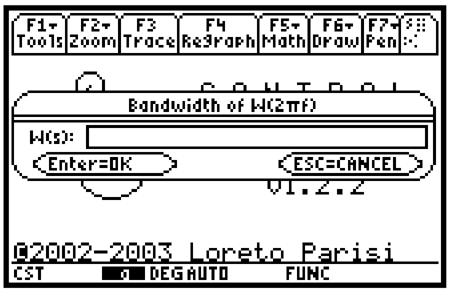

Poles of p(s)

Cx:

Please insert coefficients as LIST

cpoles(Cx)

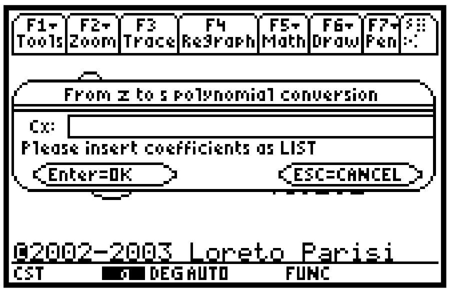
It calculates zeros of polinomyal given as LIST of coefficients, Cx.


Bandwidth of W(2πf)

W(s):

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).


From z to s polynomial conversion

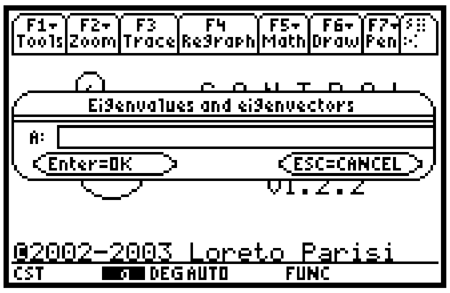
Cx:

Please insert coefficients as LIST

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

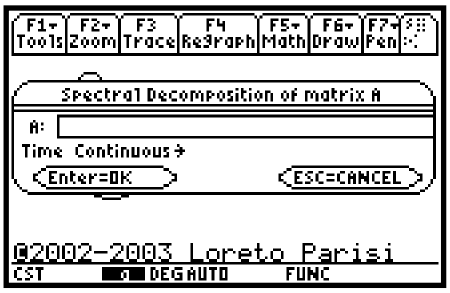
$$q(s) = (s-1)^n p(z)\Big|_{z=\frac{s+1}{s-1}}$$


Eigenvalues and eigenvectors

A:

eingev(A)

It calculates eigenvalues and eigenvectors of matrix A.



F1 Tools | F2 Zoom | F3 Trace | F4 ReGraph | F5 Math | F6 Draw | F7 Pen

Spectral Decomposition of matrix A

A:

Time Continuous →

Enter=OK ESC=CANCEL

©2002-2003 Loreto Parisi

CST DEG AUTO FUNC

spectre(A)

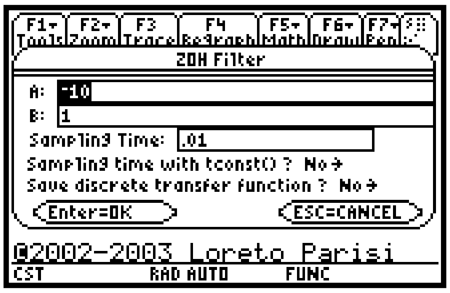
This tool calculates the spectral decomposition of matrix A, even in the continuous (e^{At}) and in the discrete time (A^k), relating to real eigenvalues and complex eigenvalues.

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 $u_h = u_{ha} \pm u_{hb}$.



F1 Tools | F2 Zoom | F3 Trace | F4 ReGraph | F5 Math | F6 Draw | F7 Pen

ZOH Filter

A: -10

B: 1

Sampling Time: .01

Sampling time with tconst() ? No →

Save discrete transfer function ? No →

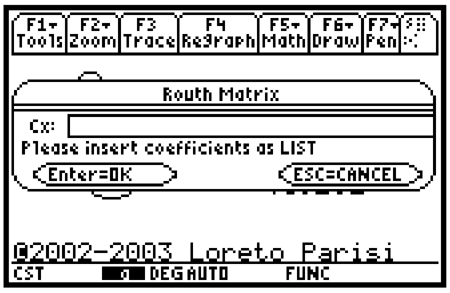
Enter=OK ESC=CANCEL

©2002-2003 Loreto Parisi

CST RAD AUTO FUNC

sampler(A,B,T_c)

This function performs the discrete time conversion of continuous time model with state-space $\dot{x} = Ax + Bu$ at sample time T_c, using the ZOH (Zero Order Hold) method. It permits to use sample time T_c calculated with function *tconst(SYS)* for current transfer function SYS. It can save the resulting discrete transfer function W(z).



F1 Tools | F2 Zoom | F3 Trace | F4 ReGraph | F5 Math | F6 Draw | F7 Pen

Routh Matrix

Cx:

Please insert coefficients as LIST

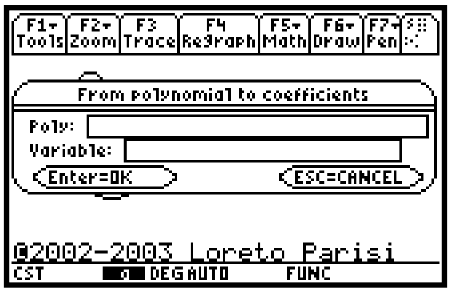
Enter=OK ESC=CANCEL

©2002-2003 Loreto Parisi

CST DEG AUTO FUNC

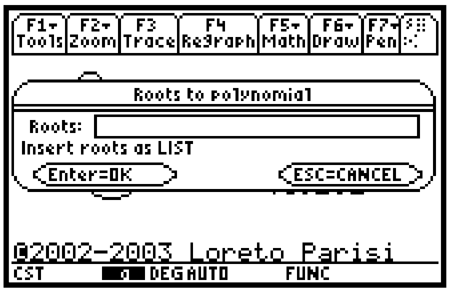
routh(Cx)

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



From polynomial to coefficients
Poly:
Variable:
Enter=OK ESC=CANCEL
©2002-2003 Loreto Parisi
CST DEG AUTO FUNC

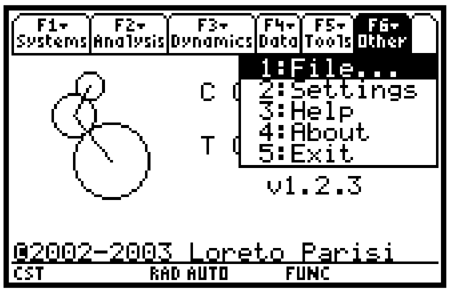
poly2cof(expr,var)
Gives the LIST of coefficients of the polynomial given in *expr* in the variable *var*.
Function *poly2cof(expr,var)* included in *CST for TI-89* with permission of *Francesco Orabona*. For contacts:
E-mail: bremen79@infinito.it
Homepage: http://web.genie.it/utenti/b/bremen79/



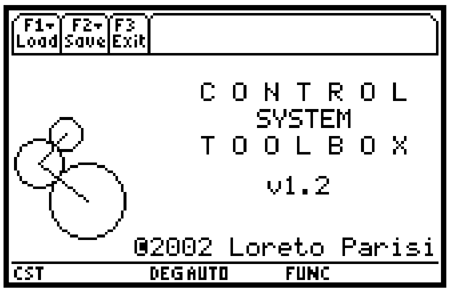
Roots to polynomial
Roots:
Insert roots as LIST
Enter=OK ESC=CANCEL
©2002-2003 Loreto Parisi
CST DEG AUTO FUNC

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

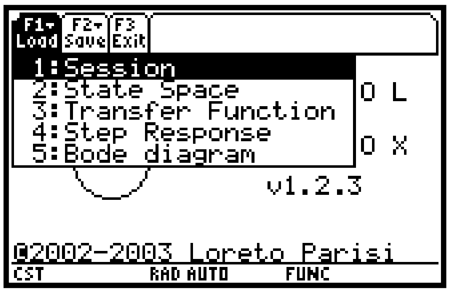
Other



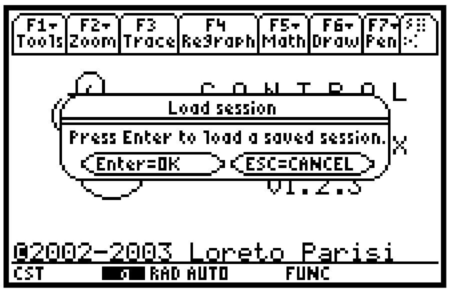
The *Other* menu (*F6*) gives tools to manage files, the current working session, the *Settings*, to access to on-line help tool with *help*(), some information about CST, and the way to *exit* CST.



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.

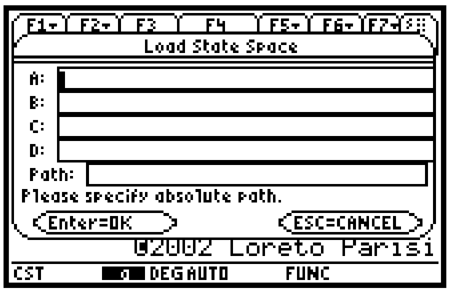


The *Load* menu (F1) permits to load the current working session, the State Space, Transfer Function, Step Responde and bode diagrams from the specified path.



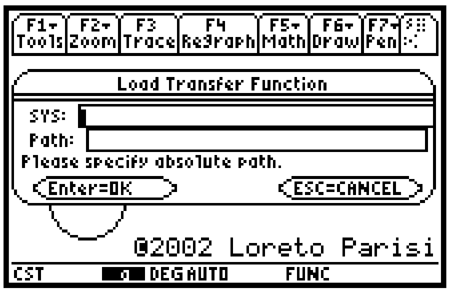
Press Enter to load a saved session.

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



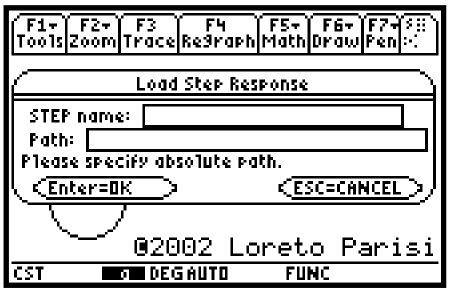
Please specify absolute path.

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.



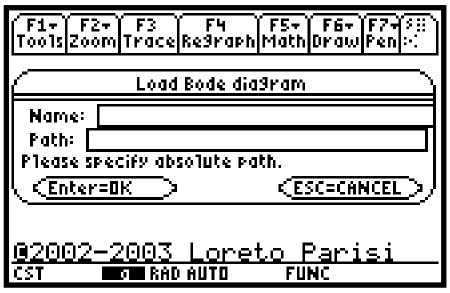
Load Transfer Function
SYS:
Path:
Please specify absolute path.
©2002 Loreto Parisi

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



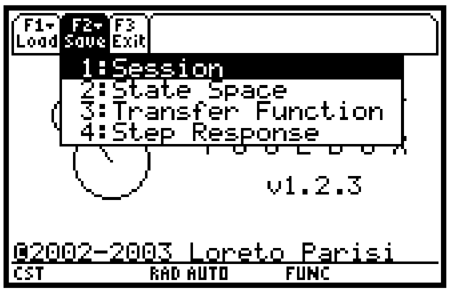
Load Step Response
STEP name:
Path:
Please specify absolute path.
©2002 Loreto Parisi

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



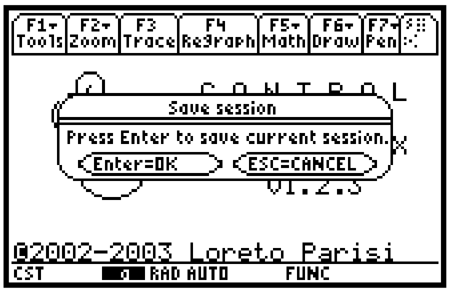
Load Bode diagram
Name:
Path:
Please specify absolute path.
©2002-2003 Loreto Parisi

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



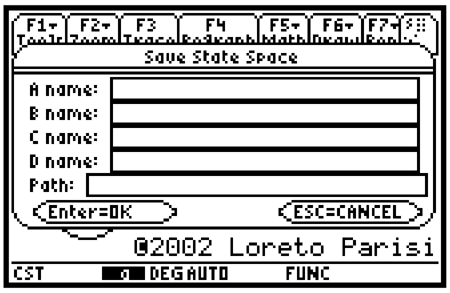
v1.2.3
©2002-2003 Loreto Parisi

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



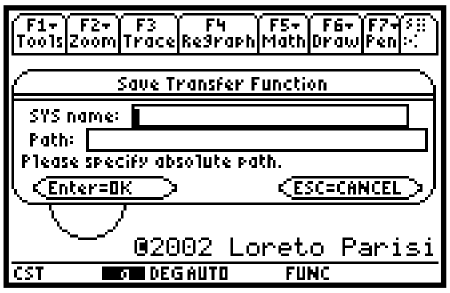
Save session
Press Enter to save current session.
©2002-2003 Loreto Parisi

Save session
It saves the current working session (i.e. transfer functions W(s) and W(z),State space, w_1(t), Tc, step response parameters,ecc.).



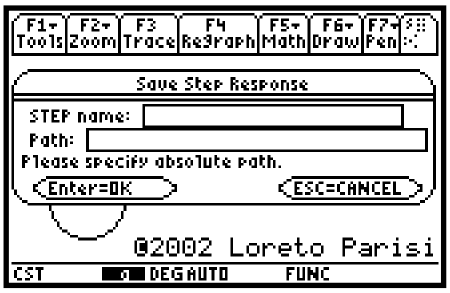
Save State Space
A name:
B name:
C name:
D name:
Path:
©2002 Loreto Parisi

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



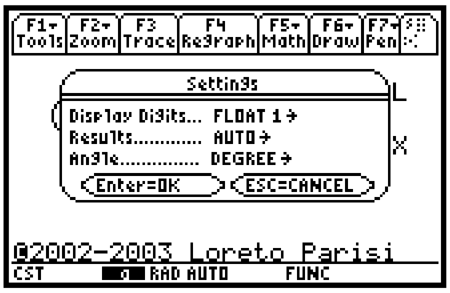
Save Transfer Function
SYS name:
Path:
Please specify absolute path.
©2002 Loreto Parisi

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



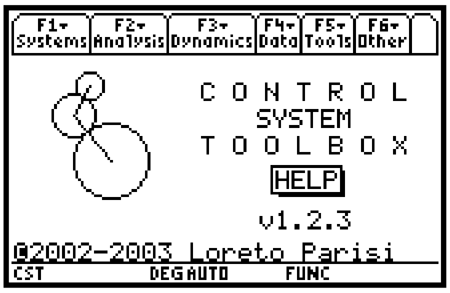
Save Step Response
STEP name:
Path:
Please specify absolute path.
©2002 Loreto Parisi

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

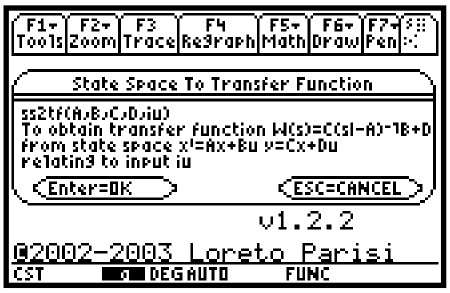


Settings
Display Digits... FLOAT 1 →
Results............ AUTO →
Angle.............. DEGREE →
©2002-2003 Loreto Parisi

Settings
It permits to modify some settings of the calculator, such as the *display digits,* the *angle,* and the format of *results.*

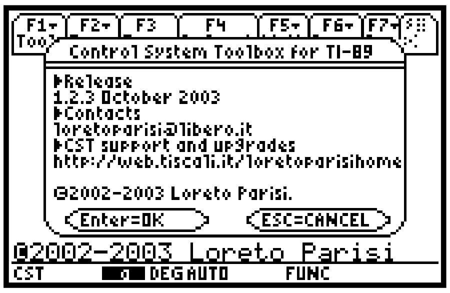


help()
The program *help()* loads the on-line help tool, which gives an instant help on all the functions of CST suite.
Note that *help*() is a standalone program also, so you can recall it from HOME, typing *CST/help().*



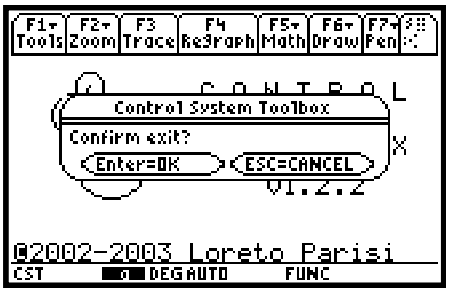
```
State Space To Transfer Function
ss2tf(A,B,C,D,iu)
To obtain transfer function W(s)=C(sI-A)^-1B+D
from state space x'=Ax+Bu y=Cx+Du
relating to input iu
v1.2.2
©2002-2003 Loreto Parisi
```

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



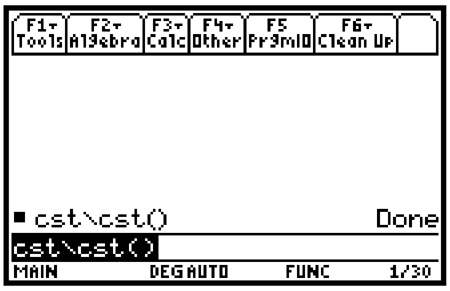
```
Control System Toolbox for TI-89
▸Release
1.2.3 October 2003
▸Contacts
loretoparisi@libero.it
▸CST support and upgrades
http://web.tiscali.it/loretoparisihome

©2002-2003 Loreto Parisi.
©2002-2003 Loreto Parisi
```

About
Gives the current version of CST *for TI-89*, the way to contact the author and to obtain support and upgrades.



```
Control System Toolbox
Confirm exit?
©2002-2003 Loreto Parisi
```

Exit
Exit Control System Toolbox *for TI-89.* All previous settings of the calculator will be restored.



```
■ cst\cst()                Done
cst\cst()
```

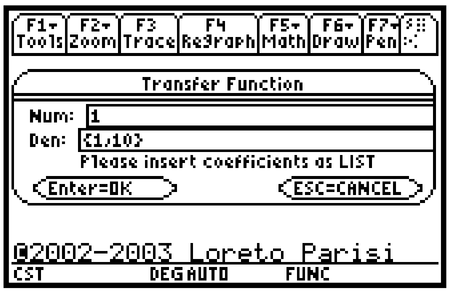
cst()
Enjoyed the journey ?
We'll be pleased.

Examples

Example 1. *First order LPF*

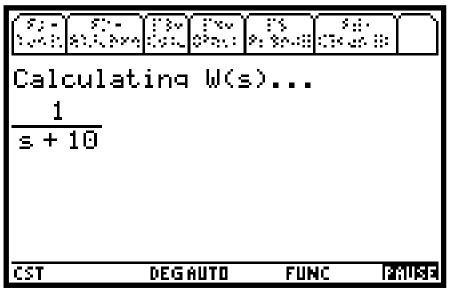
Considers a sample low pass filter, with transfer function W(s)=$\frac{1}{s+10}$.

See what can be done in *CST.*



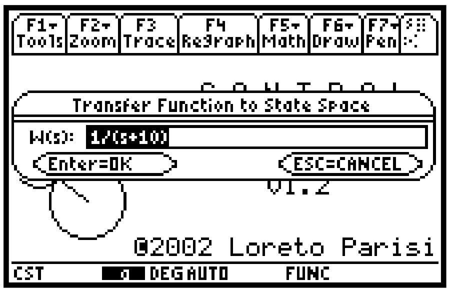
```
Transfer Function
Num: 1
Den: {1,10}
Please insert coefficients as LIST
©2002-2003 Loreto Parisi
```

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}.



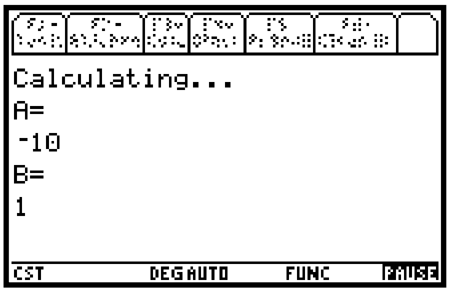
```
Calculating W(s)...
  1
------
s + 10
```

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



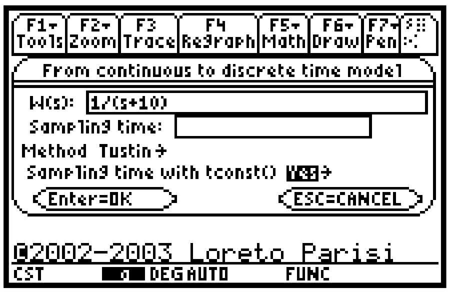
```
Transfer Function to State Space
W(s): 1/(s+10)
v1.2
©2002 Loreto Parisi
```

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.



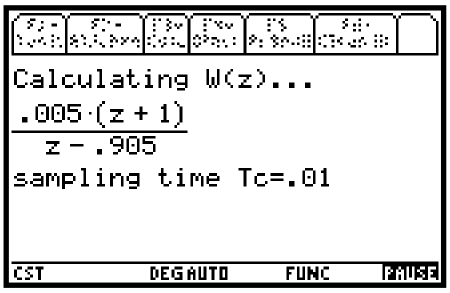
```
Calculating...
A=
 -10
B=
1
```

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



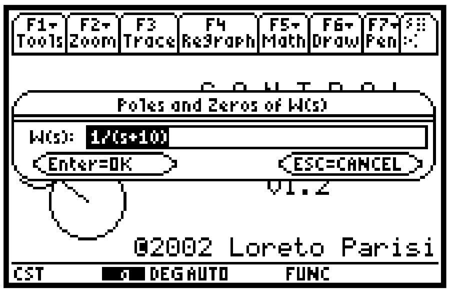
```
From continuous to discrete time model
W(s): 1/(s+10)
Sampling time:
Method  Tustin→
Sampling time with tconst() YES→
©2002-2003 Loreto Parisi
```

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.



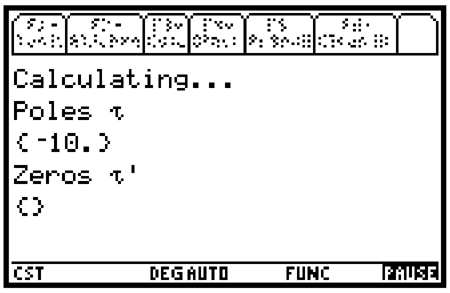
```
Calculating W(z)...
.005·(z + 1)
------------
  z - .905
sampling time Tc=.01
```

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



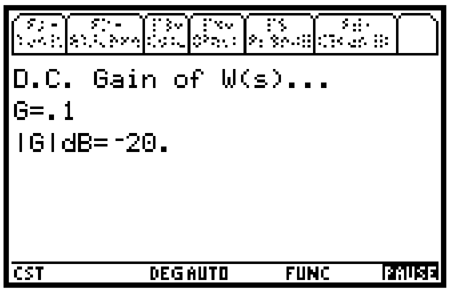
```
Poles and Zeros of W(s)
W(s): 1/(s+10)
©2002 Loreto Parisi
```

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.



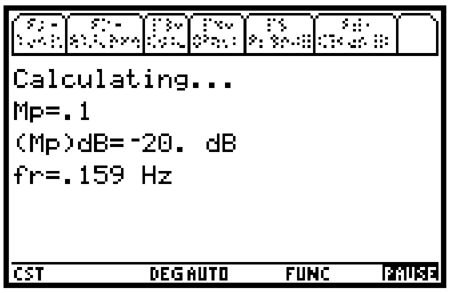
```
Calculating...
Poles ι
{-10.}
Zeros ι'
{}
```

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



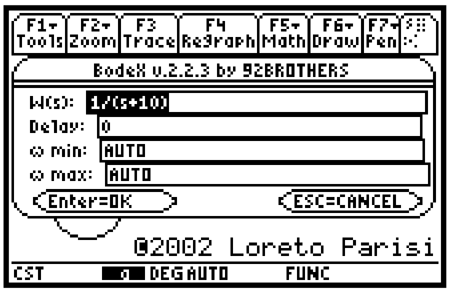
```
D.C. Gain of W(s)...
G=.1
|G|dB=-20.
```

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



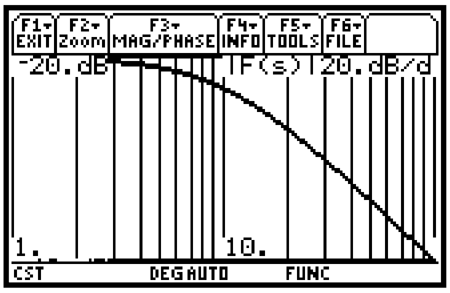
```
Calculating...
Mp=.1
(Mp)dB=-20. dB
fr=.159 Hz
```

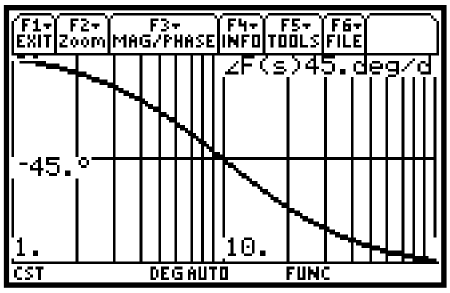
In the same way, with function *peak(SYS)* we'll obtain the resonance peak M_P (in this case it matches with previous d.c. gain G) and relating resonance frequency f_r that results .159 Hz.



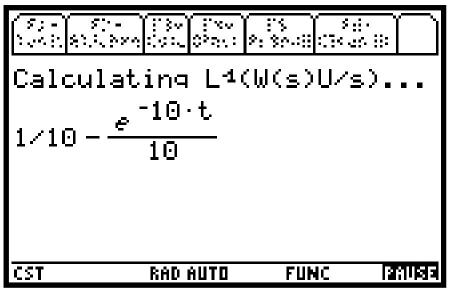
```
BodeX v.2.2.3 by 92BROTHERS
W(s): 1/(s+10)
Delay: 0
ω min: AUTO
ω max: AUTO
Enter=OK      ESC=CANCEL
©2002 Loreto Parisi
```

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.

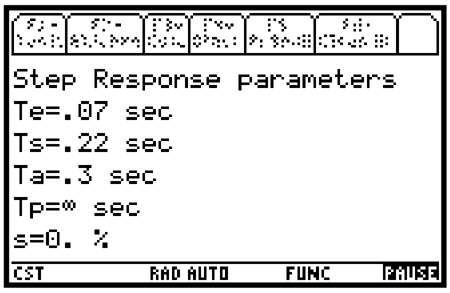




Now we'll save those plots and exit *bodex()*.



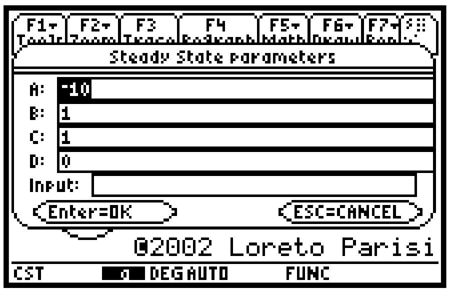
```
Calculating L⁻¹(W(s)U/s)...
1/10 - e^(-10·t)/10
```



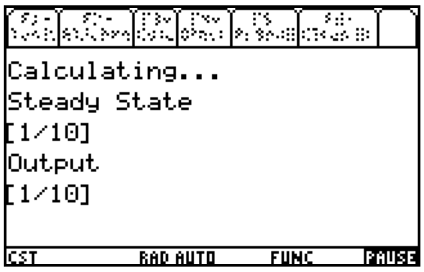
```
Step Response parameters
Te=.07 sec
Ts=.22 sec
Ta=.3 sec
Tp=∞ sec
s=0. %
```

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

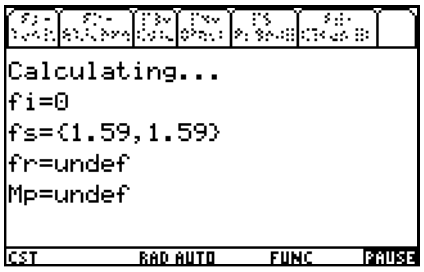
To get steady state we'll use function *trim(A,B,C, D,u_0)*. Notice that x0 and y0 matches with w_{-1}(t).

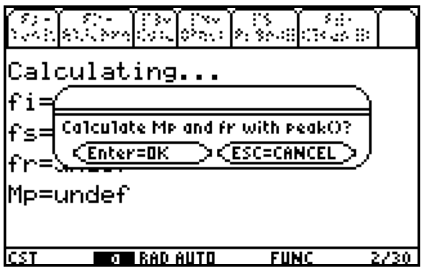


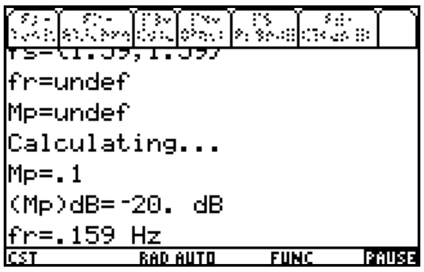
```
Steady State parameters
A: -10
B: 1
C: 1
D: 0
Input:
Enter=OK      ESC=CANCEL
©2002 Loreto Parisi
```



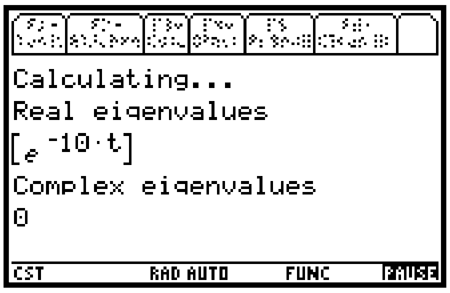
```
Calculating...
Steady State
[1/10]
Output
[1/10]
```

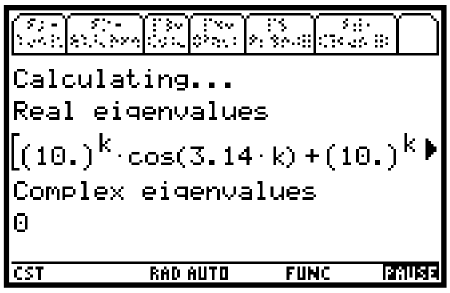

```
Calculating...
fi=0
fs={1.59,1.59}
fr=undef
Mp=undef
```


```
Calculating...
fi=
fs=
fr=
Mp=undef
Calculate Mp and fr with peak()?
```

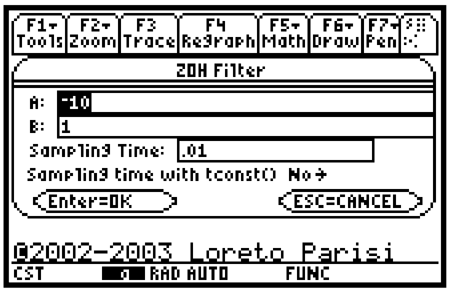

```
fs={1.59,1.59}
fr=undef
Mp=undef
Calculating...
Mp=.1
(Mp)dB=-20. dB
fr=.159 Hz
```

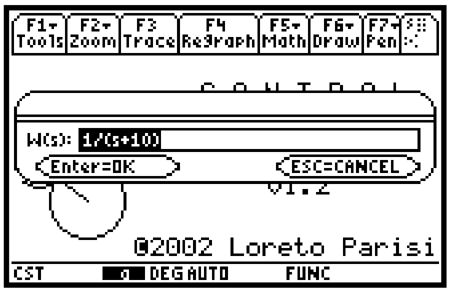
We have calculated the bandwith choosing *band(SYS)* from *Tools* menu (*F5).* Notice that M_P and f_r results from *peak(SYS)* when *band(SYS)* fails.

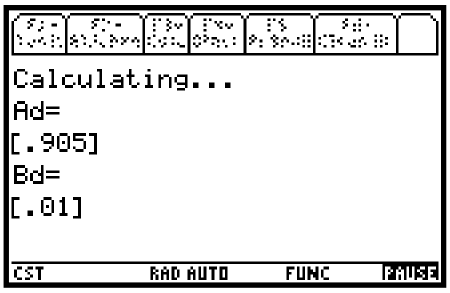

```
Calculating...
Real eigenvalues
[e^-10·t]
Complex eigenvalues
0
```


```
Calculating...
Real eigenvalues
[(10.)^k·cos(3.14·k)+(10.)^k ▸
Complex eigenvalues
0
```

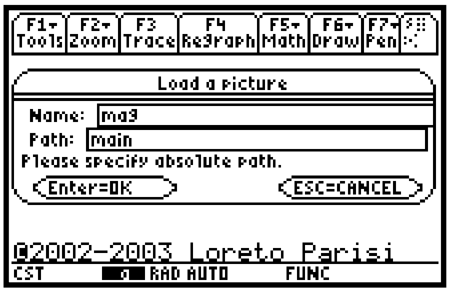
With *spectre*(A) we'll obtain e^{At} and A^K. Because of real pole, we have not complex expansion.

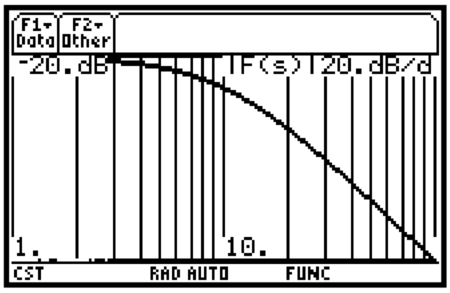

```
ZOH Filter
A: -10
B: 1
Sampling Time: .01
Sampling time with tconst() No→
©2002-2003 Loreto Parisi
```

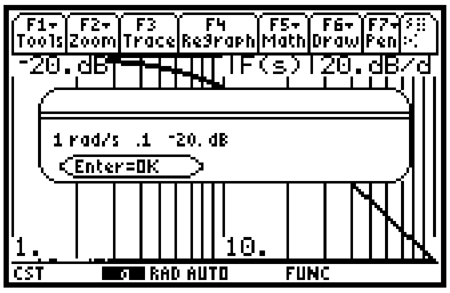

```
W(s): 1/(s+10)
CONTROL
v1.2
©2002 Loreto Parisi
```


```
Calculating...
Ad=
[.905]
Bd=
[.01]
```

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).


```
Load a picture
Name: mag
Path: main
Please specify absolute path.
©2002-2003 Loreto Parisi
```

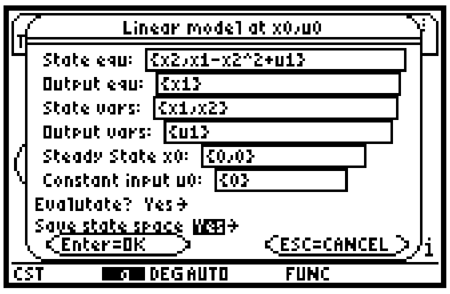



```
1 rad/s .1 -20. dB
```

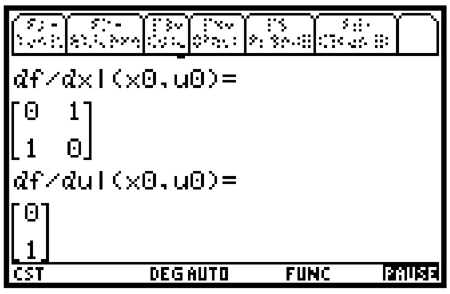
From *Load* menu (*F1)* we'll choose *Picture* to load previous magnitude Bode plot. Here we can evalutate magnitude (or phase) at ω_0 choosing from *Data* menu (*F1)* function *mag(SYS,*ω_0*).*

Example 2. *Linearization*

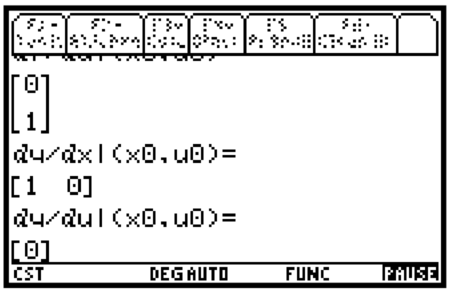
Now consider a non linear model: $\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_1 - x_2^{\,2} + u \\ y = x_1 \end{cases}$. To work with it, we need to linearize around a steady state x_0 relating to constant input u_0. First, we have to calculate the steady state x_0. It results: $x_0 = \begin{pmatrix}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



```
Linear model at x0,u0
State equ:          {x2,x1-x2^2+u1}
Output equ:         {x1}
State vars:         {x1,x2}
Output vars:        {u1}
Steady State x0:    {0,0}
Constant input u0:  {0}
Evalutate?  Yes→
Save state space  Yes→
<Enter=OK>                <ESC=CANCEL>
CST          DEG AUTO       FUNC
```

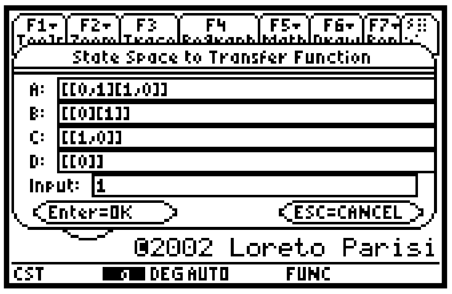


```
df/dx|(x0,u0)=
[0  1]
[1  0]
df/du|(x0,u0)=
[0]
[1]
CST          DEG AUTO       FUNC      PAUSE
```

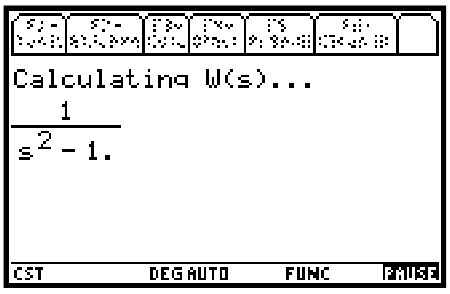


```
[0]
[1]
dy/dx|(x0,u0)=
[1  0]
dy/du|(x0,u0)=
[0]
CST          DEG AUTO       FUNC      PAUSE
```

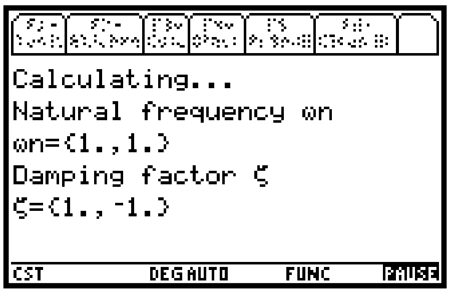
From *Dynamics* menu (*F3)* we choose *linmod*(f,y,x,u,x_0,u_0) where we have f={x_2,x_1-$x_2^{\,2}$+u_1}, y={x_1}, x={x_1,x_2}, u={u_1}, x_0={0,0} and u_0={0}. We have decided to evalutate jacobian matrixex in x_0 and u_0 to save the state-space of obtained linearized model. Now, we can get transfer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



```
State Space to Transfer Function
A:      [[0,1][1,0]]
B:      [[0][1]]
C:      [[1,0]]
D:      [[0]]
Input:  1
<Enter=OK>                <ESC=CANCEL>
          ©2002 Loreto Parisi
CST          DEG AUTO       FUNC
```

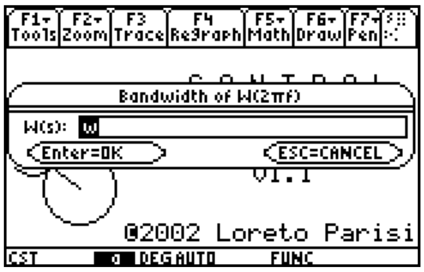


```
Calculating W(s)...
    1
 -------
 s^2 - 1.
CST          DEG AUTO       FUNC      PAUSE
```

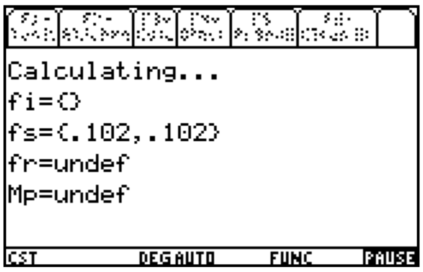


```
Calculating...
Natural frequency ωn
ωn={1.,1.}
Damping factor ζ
ζ={1.,-1.}
CST          DEG AUTO       FUNC      PAUSE
```

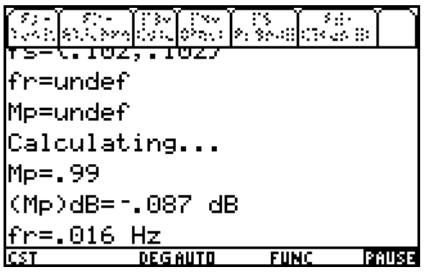
For this linear model, we'll get natural frequencies ω_{nh} and damping factors ζ_{nh} using function *damp(SYS)* from *Analysis* menu (*F2*). Now we'll use *band(SYS)*:



```
Bandwidth of W(2πf)
W(s): w
V1.1
©2002 Loreto Parisi
```

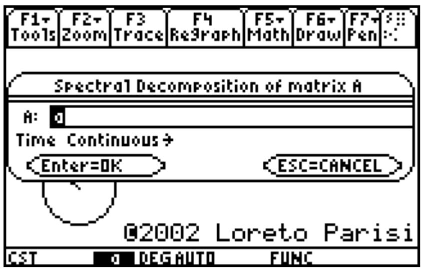


```
Calculating...
fi={}
fs={.102,.102}
fr=undef
Mp=undef
```

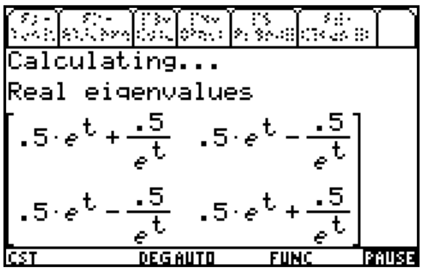


```
fs={.102,.102}
fr=undef
Mp=undef
Calculating...
Mp=.99
(Mp)dB=-.087 dB
fr=.016 Hz
```

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

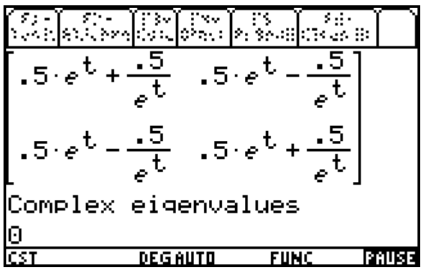


```
Spectral Decomposition of matrix A
A: a
Time Continuous
©2002 Loreto Parisi
```



Calculating...
Real eigenvalues

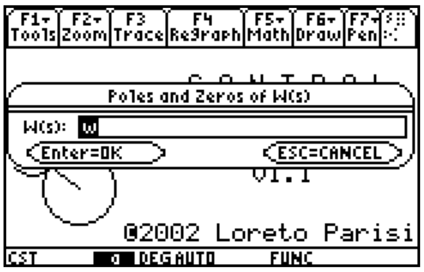
$$\begin{bmatrix} .5\cdot e^{t}+\frac{.5}{e^{t}} & .5\cdot e^{t}-\frac{.5}{e^{t}} \\ .5\cdot e^{t}-\frac{.5}{e^{t}} & .5\cdot e^{t}+\frac{.5}{e^{t}} \end{bmatrix}$$



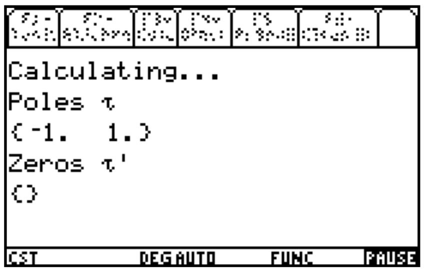
$$\begin{bmatrix} .5\cdot e^{t}+\frac{.5}{e^{t}} & .5\cdot e^{t}-\frac{.5}{e^{t}} \\ .5\cdot e^{t}-\frac{.5}{e^{t}} & .5\cdot e^{t}+\frac{.5}{e^{t}} \end{bmatrix}$$

Complex eigenvalues
0

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



```
Poles and Zeros of W(s)
W(s): w
V1.1
©2002 Loreto Parisi
```

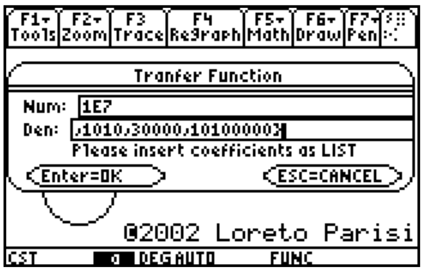


```
Calculating...
Poles τ
{-1.  1.}
Zeros τ'
{}
```

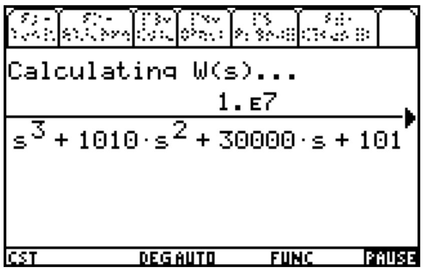
As we can see, there is a positive pole who causes instability of the model.

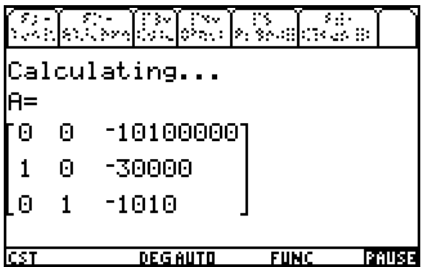
Example 3. *3rd order LPF*

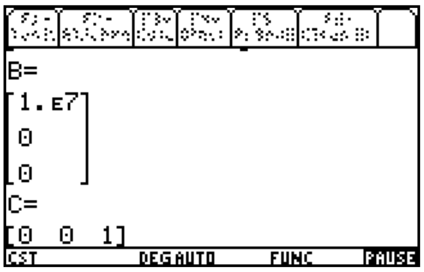
Consider a 3rd order low pass filter $W(s)=\dfrac{10000000}{s^3+1010s^2+30000s+10100000}$

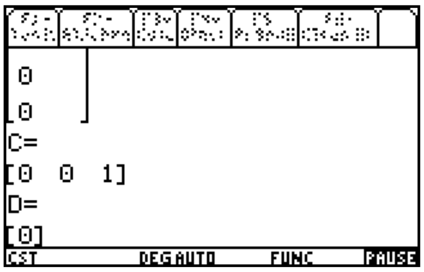


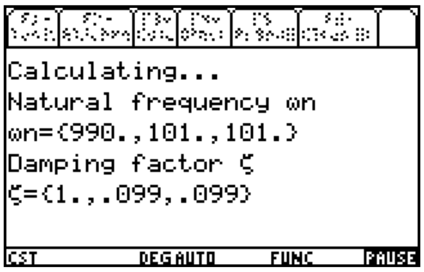
Tranfer Function
Num: 1E7
Den: ,1010,30000,10100000}
Please insert coefficients as LIST
©2002 Loreto Parisi

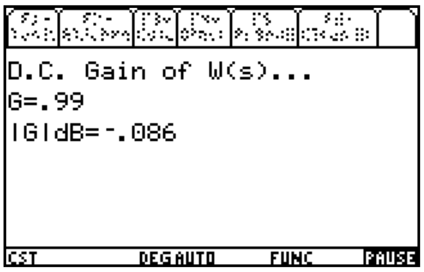

```
Calculating W(s)...
              1.E7
s^3 + 1010·s^2 + 30000·s + 101
```

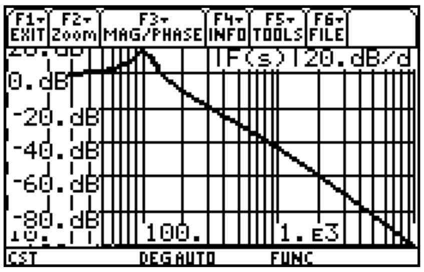

```
Calculating...
A=
[0  0  -10100000]
[1  0  -30000   ]
[0  1  -1010    ]
```

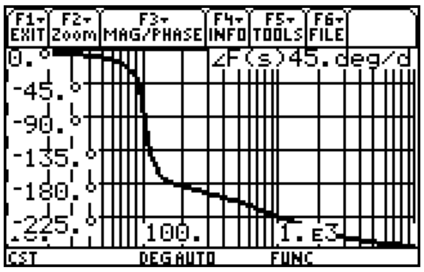

```
B=
[1.E7]
[0   ]
[0   ]
C=
[0  0  1]
```

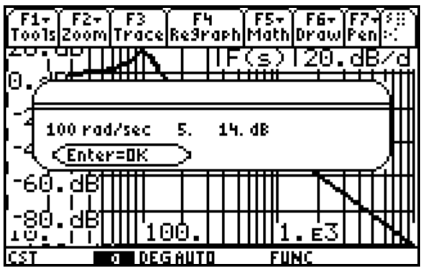

```
 0
 0
C=
[0  0  1]
D=
[0]
```

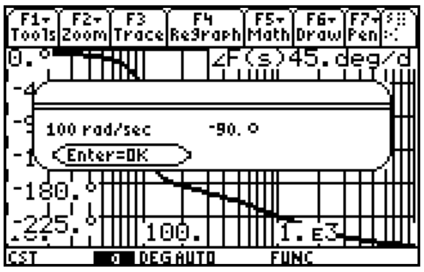

```
Calculating...
Natural frequency ωn
ωn={990.,101.,101.}
Damping factor ζ
ζ={1.,.099,.099}
```

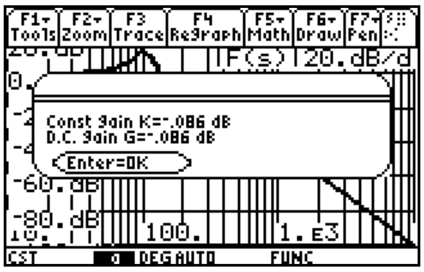

```
D.C. Gain of W(s)...
G=.99
|G|dB=-.086
```

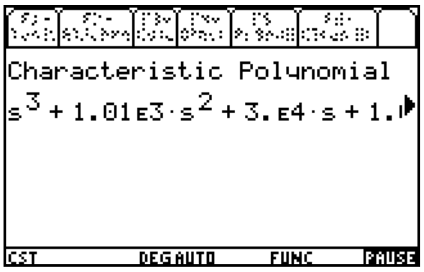


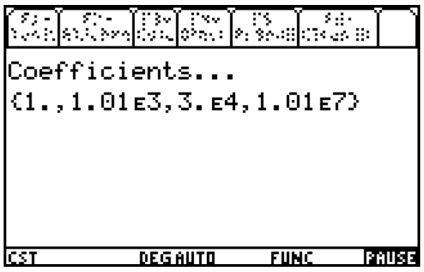


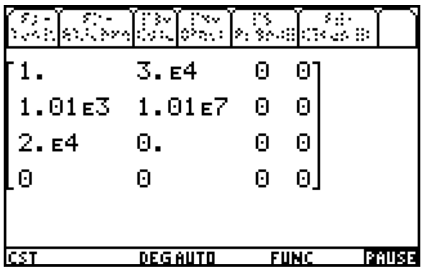

```
100 rad/sec    5.   14. dB
```

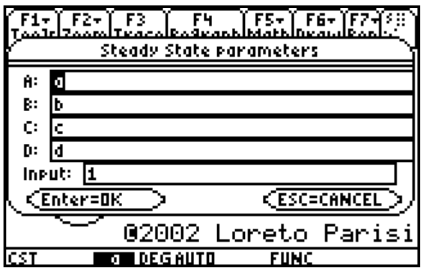

```
100 rad/sec       -90. °
```


```
Const Gain K=-.086 dB
D.C. Gain G=-.086 dB
```

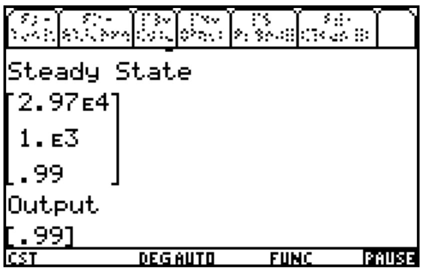

```
Characteristic Polynomial
s^3 + 1.01E3·s^2 + 3.E4·s + 1.
```


```
Coefficients...
{1.,1.01E3,3.E4,1.01E7}
```

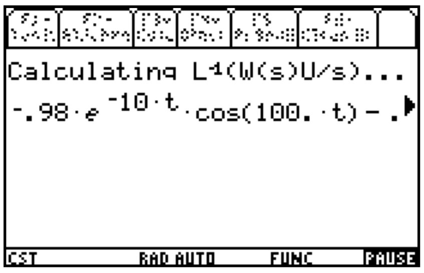

```
[1.      3.E4    0  0]
[1.01E3  1.01E7  0  0]
[2.E4    0.      0  0]
[0       0       0  0]
```



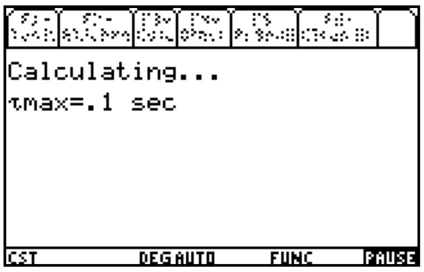
```
Steady State parameters
A: a
B: b
C: c
D: d
Input: 1
©2002 Loreto Parisi
```



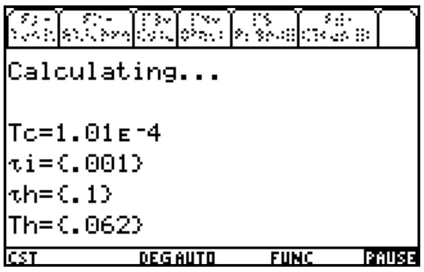
```
Steady State
[2.97E4]
[1.E3  ]
[.99   ]
Output
[.99]
```



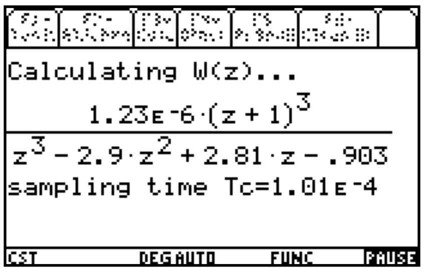
```
Calculating L⁻¹(W(s)U/s)...
-.98·e^(-10·t)·cos(100.·t) - .
```



```
Calculating...
τmax=.1 sec
```



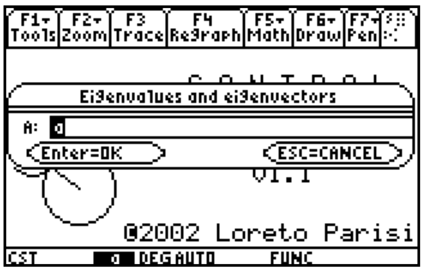
```
Calculating...

Tc=1.01E-4
τi={.001}
τh={.1}
Th={.062}
```



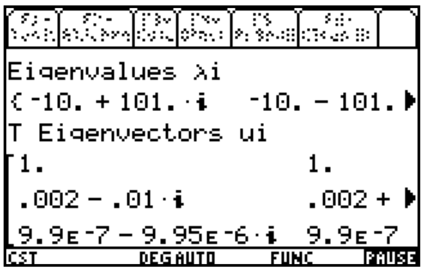
```
Calculating W(z)...
```

$$\frac{1.23\mathrm{E}{-6}\cdot(z+1)^3}{z^3-2.9\cdot z^2+2.81\cdot z-.903}$$

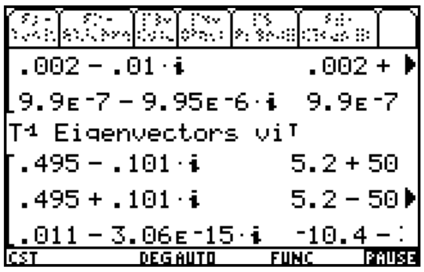
```
sampling time Tc=1.01E-4
```



```
Eigenvalues and eigenvectors
A: a
©2002 Loreto Parisi
```



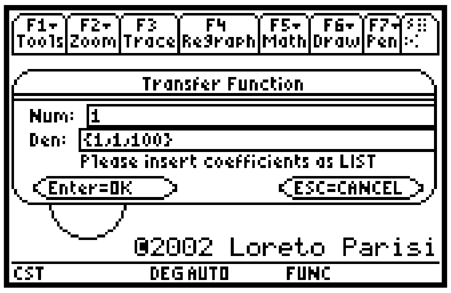
```
Eigenvalues λi
{-10. + 101.·i    -10. - 101.
T Eigenvectors ui
[1.                    1.
 .002 - .01·i          .002 +
 9.9E-7 - 9.95E-6·i    9.9E-7
```

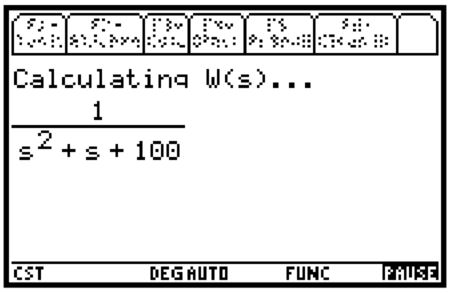


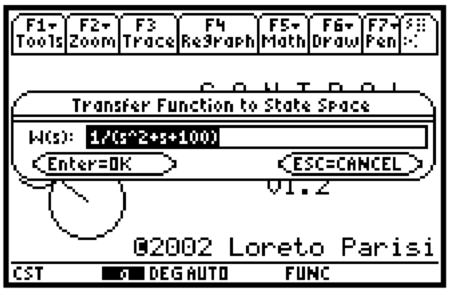
```
 .002 - .01·i          .002 +
 9.9E-7 - 9.95E-6·i    9.9E-7
T⁻¹ Eigenvectors viᵀ
[.495 - .101·i         5.2 + 50
 .495 + .101·i         5.2 - 50
 .011 - 3.06E-15·i     -10.4 -
```

Example 4. *2nd order LPF*

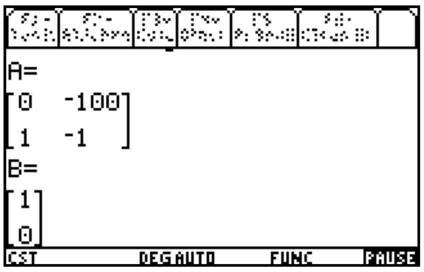
Consider a 2nd order low pass filter W(s)=$\frac{1}{s^2+s+100}$;

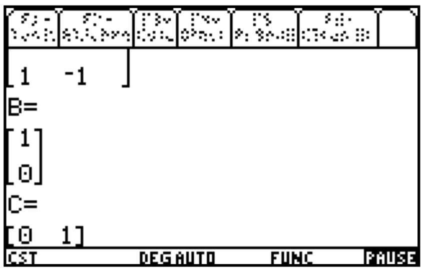

```
Transfer Function
Num: 1
Den: {1,1,100}
Please insert coefficients as LIST
Enter=OK          ESC=CANCEL
©2002 Loreto Parisi
```

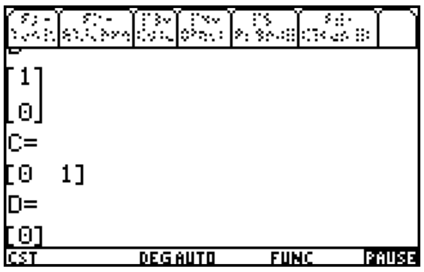

```
Calculating W(s)...
```
$$\frac{1}{s^2+s+100}$$


```
Transfer Function to State Space
W(s): 1/(s^2+s+100)
Enter=OK          ESC=CANCEL
V1.2
©2002 Loreto Parisi
```

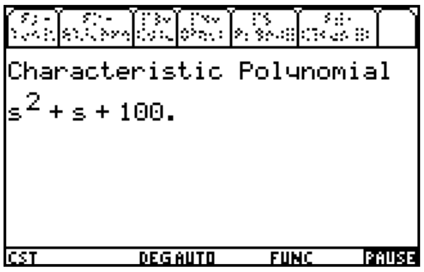
First, we'll define transfer function and calculates state-space representation:

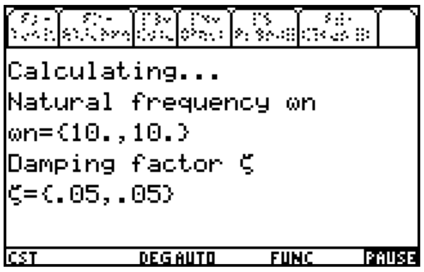

```
A=
[0  -100]
[1  -1  ]
B=
[1]
[0]
```


```
[1  -1  ]
B=
[1]
[0]
C=
[0  1]
```

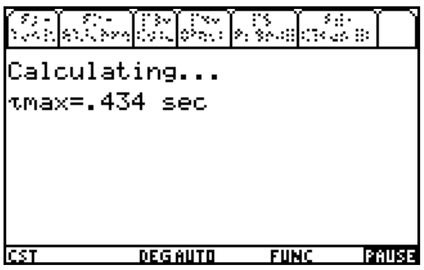

```
[1]
[0]
C=
[0  1]
D=
[0]
```

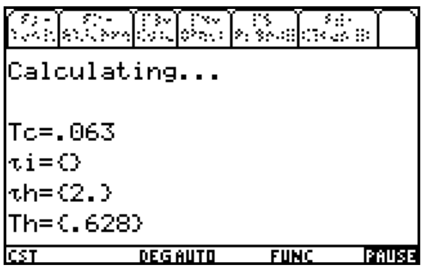
Now we'll get the characteristic polynomial, natural frequency and damping factor:

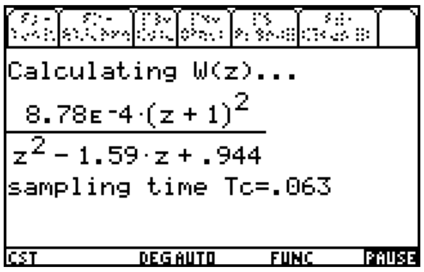

```
Characteristic Polynomial
s^2 + s + 100.
```


```
Calculating...
Natural frequency ωn
ωn={10.,10.}
Damping factor ζ
ζ={.05,.05}
```

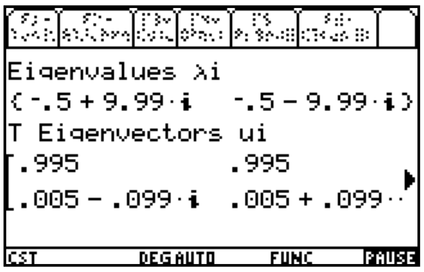
Then we'll calculate maximum time constant, sample time and other time constants, and the discrete model relating to previous sample time with Tustin method:


```
Calculating...
τmax=.434 sec
```

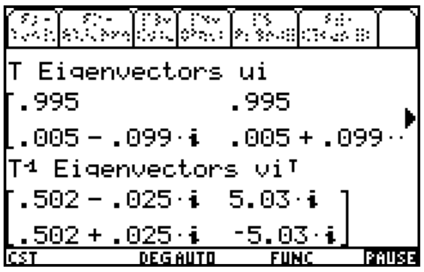

```
Calculating...

Tc=.063
τi={}
τh={2.}
Th={.628}
```


```
Calculating W(z)...
```
$$\frac{8.78E^{-4}\cdot(z+1)^2}{z^2-1.59\cdot z+.944}$$
```
sampling time Tc=.063
```

Eigenvalues and eigenvector :

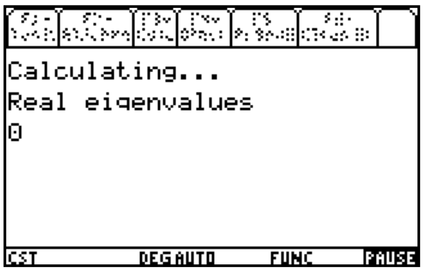


```
Eigenvalues λi
{-.5 + 9.99·i   -.5 - 9.99·i}
T Eigenvectors ui
[.995            .995
[.005 - .099·i   .005 + .099··
CST        DEG AUTO    FUNC   PAUSE
```

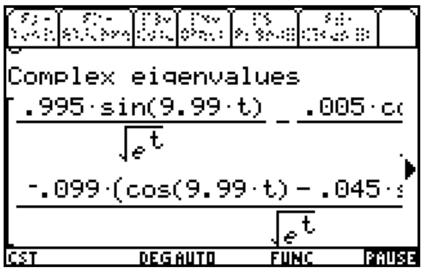


```
T Eigenvectors ui
[.995            .995
[.005 - .099·i   .005 + .099··
T-1 Eigenvectors viT
[.502 - .025·i   5.03·i ]
[.502 + .025·i   -5.03·i]
CST        DEG AUTO    FUNC   PAUSE
```

Spectral decomposition of dynamics matrix A:

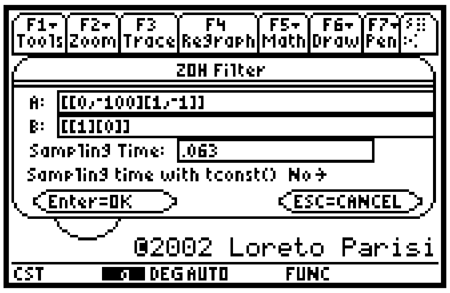


```
Calculating...
Real eigenvalues
0
CST        DEG AUTO    FUNC   PAUSE
```

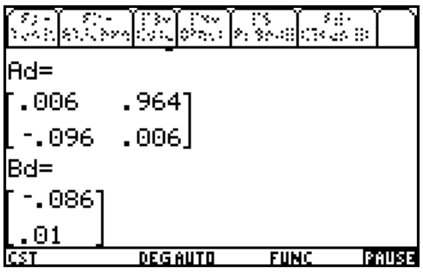


```
Complex eigenvalues
[.995·sin(9.99·t)   .005·c
 ----------------  -  ------
      √(e^t)
 -.099·(cos(9.99·t) - .045·s
 ---------------------------
                √(e^t)
CST        DEG AUTO    FUNC   PAUSE
```

Discrete model with ZOH method:

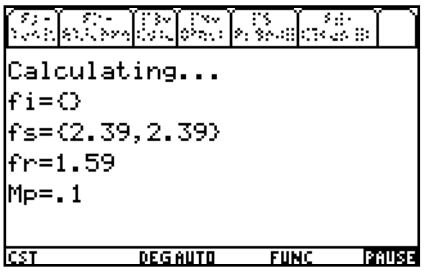


```
ZOH Filter
A: [[0,-100][1,-1]]
B: [[1][0]]
Sampling Time: .063
Sampling time with tconst()  No →
Enter=OK                ESC=CANCEL
©2002 Loreto Parisi
CST        DEG AUTO    FUNC
```

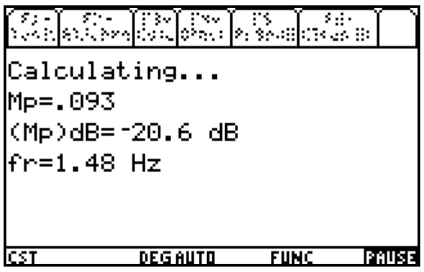


```
Ad=
[.006    .964]
[-.096   .006]
Bd=
[-.086]
[.01  ]
CST        DEG AUTO    FUNC   PAUSE
```

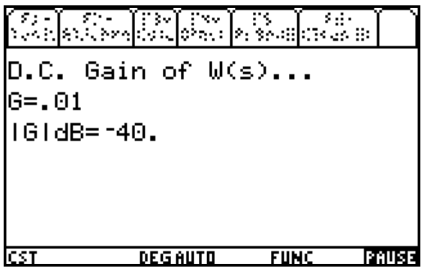
Bandwidth, resonance peak and d.c. gain:



```
Calculating...
fi={}
fs={2.39,2.39}
fr=1.59
Mp=.1
CST        DEG AUTO    FUNC   PAUSE
```



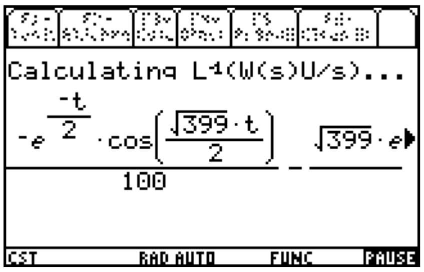
```
Calculating...
Mp=.093
(Mp)dB=-20.6 dB
fr=1.48 Hz
CST        DEG AUTO    FUNC   PAUSE
```



```
D.C. Gain of W(s)...
G=.01
|G|dB=-40.
CST        DEG AUTO    FUNC   PAUSE
```

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)*.

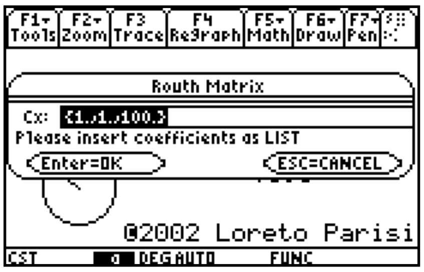
Step response:



Calculating L⁻¹(W(s)U/s)...

$$\frac{-e^{\frac{-t}{2}}\cdot\cos\left(\frac{\sqrt{399}\cdot t}{2}\right)}{100}-\frac{\sqrt{399}\cdot e}{}$$

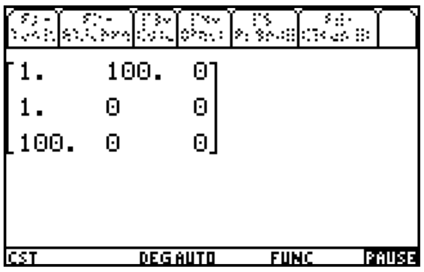
Routh matrix:



Routh Matrix

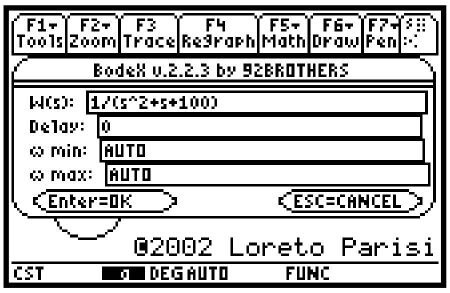
Cx: {1.,1.,100.}

Please insert coefficients as LIST



$$\begin{bmatrix} 1. & 100. & 0 \\ 1. & 0 & 0 \\ 100. & 0 & 0 \end{bmatrix}$$

Bode plots:



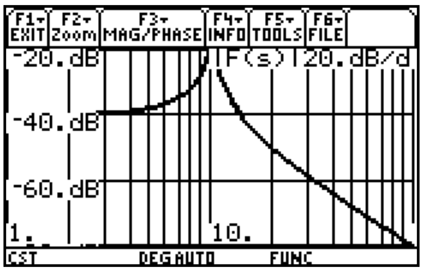
BodeX v.2.2.3 by 92BROTHERS

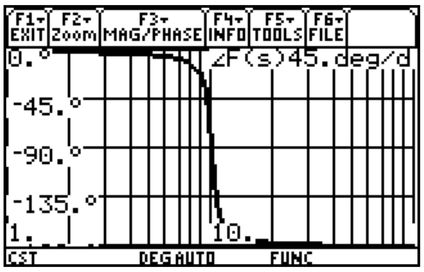
W(s): 1/(s^2+s+100)

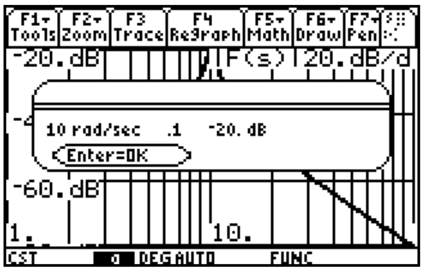
Delay: 0

ω min: AUTO

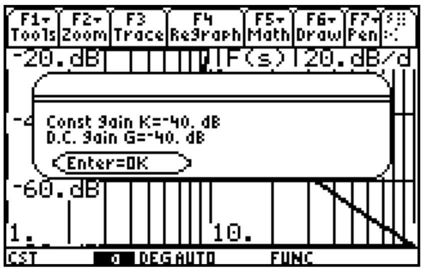
ω max: AUTO



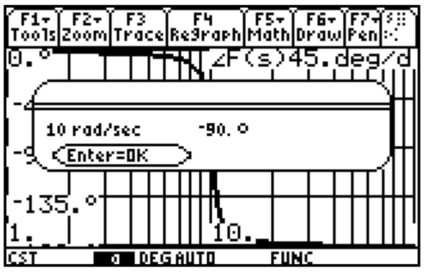




10 rad/sec .1 -20. dB



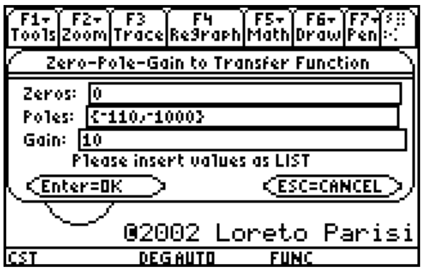
Const gain K=-40. dB
D.C. gain G=-40. dB



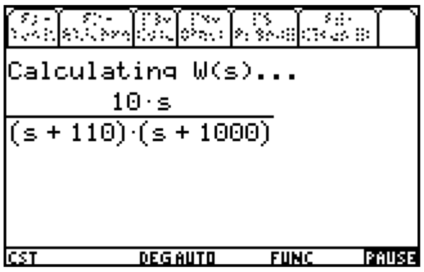
10 rad/sec -90. °

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:



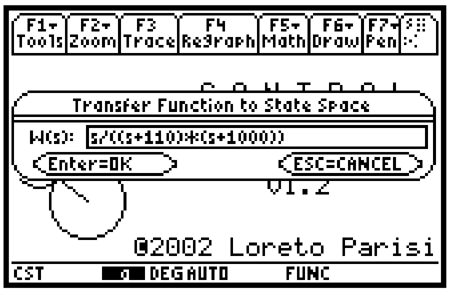
```
Zero-Pole-Gain to Transfer Function
Zeros: 0
Poles: {-110,-1000}
Gain: 10
Please insert values as LIST
©2002 Loreto Parisi
```



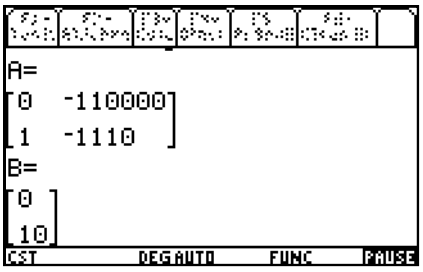
```
Calculating W(s)...
       10·s
(s + 110)·(s + 1000)
```

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

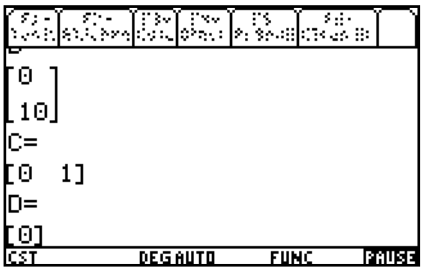
Now we need the state-space representation, so we use *tf2ss(SYS)*:



```
Transfer Function to State Space
W(s): s/((s+110)*(s+1000))
©2002 Loreto Parisi
```

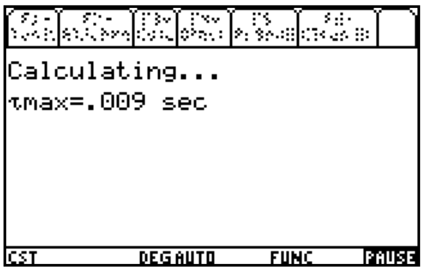


```
A=
[0  -110000]
[1  -1110  ]
B=
[0 ]
[10]
```

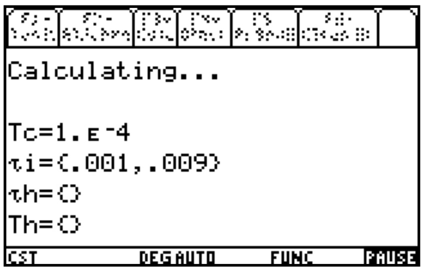


```
[0 ]
[10]
C=
[0  1]
D=
[0]
```

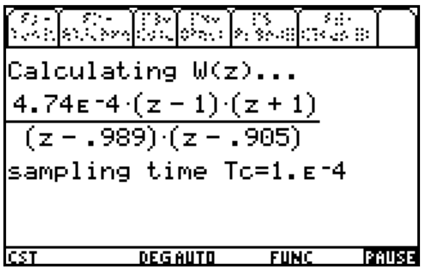
We can get the relating discrete model with *c2d(SYS,T_c)*, but we could need to obtain time constants too, so we use *tconst(SYS)* and *τmax(A)*:



```
Calculating...
τmax=.009 sec
```

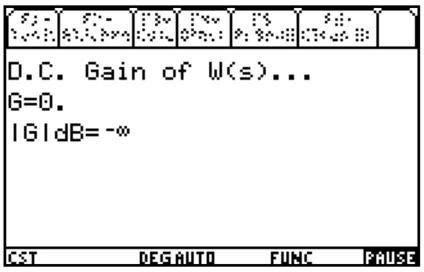


```
Calculating...

Tc=1.E-4
τi={.001,.009}
τh={}
Th={}
```

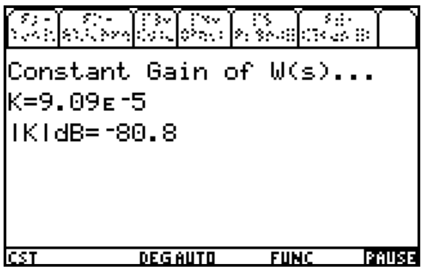


```
Calculating W(z)...
 4.74E-4·(z - 1)·(z + 1)
  (z - .989)·(z - .905)
sampling time Tc=1.E-4
```

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:

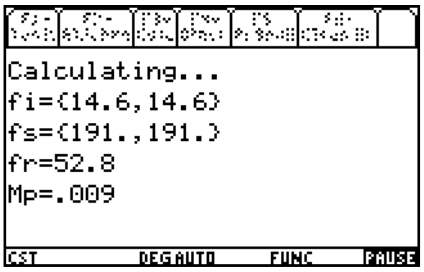


```
D.C. Gain of W(s)...
G=0.
|G|dB=-∞
```

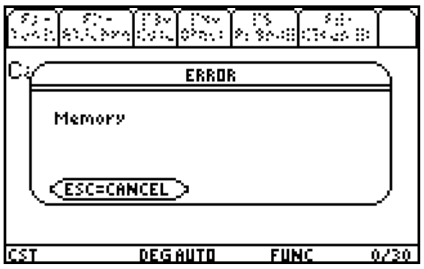


```
Constant Gain of W(s)...
K=9.09E-5
|K|dB=-80.8
```

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



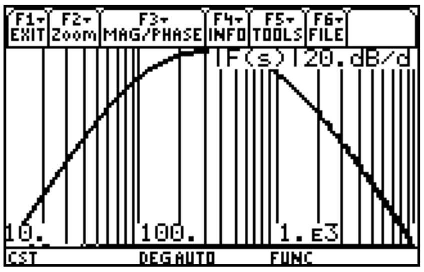
```
Calculating...
fi={14.6,14.6}
fs={191.,191.}
fr=52.8
Mp=.009
```

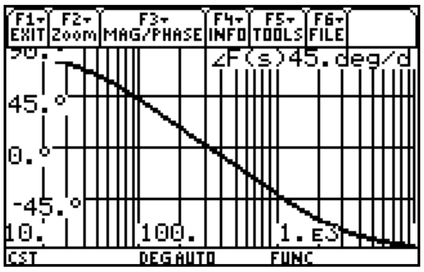


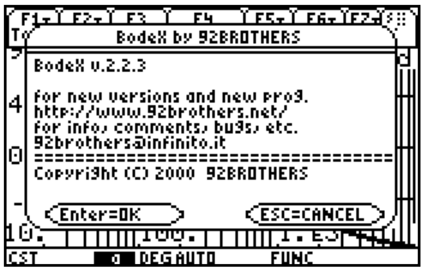
```
ERROR
Memory
ESC=CANCEL
```

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

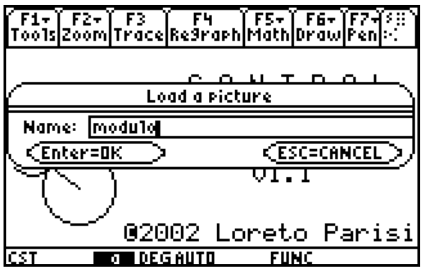
Peak(SYS) went to error because of lack of memory. This is due to too far away poles (one decade). It's our fault! We are working to this trouble and encouraging suggestions. Anyhow, we have obtained the bandwidth and peak. Now let's calculate Bode plots:



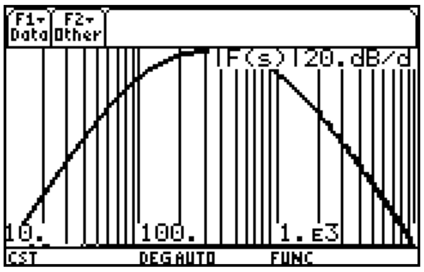


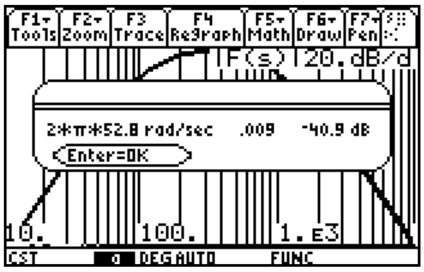


```
BodeX by 92BROTHERS
BodeX v.2.2.3

for new versions and new prog.
http://www.92brothers.net/
for info, comments, bugs, etc.
92brothers@infinito.it
=====================================
Copyright (C) 2000  92BROTHERS

Enter=OK                ESC=CANCEL
```



```
Load a picture
Name: modulo
Enter=OK                ESC=CANCEL
```

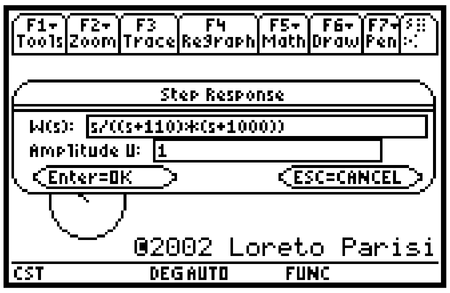




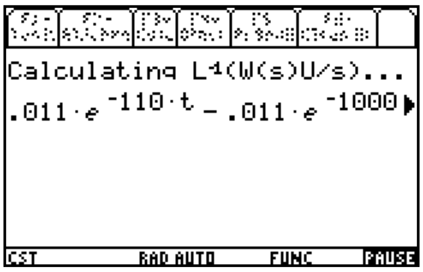
```
2*π*52.8 rad/sec   .009   -40.9 dB
Enter=OK
```

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

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

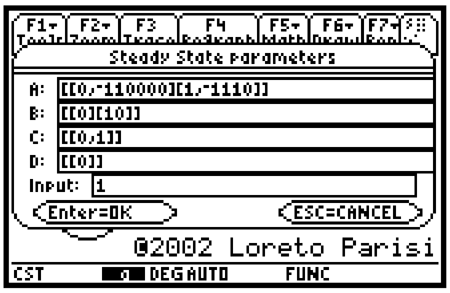


```
Step Response
W(s): s/((s+110)*(s+1000))
Amplitude U: 1
Enter=OK    ESC=CANCEL
©2002 Loreto Parisi
CST   DEG AUTO   FUNC
```

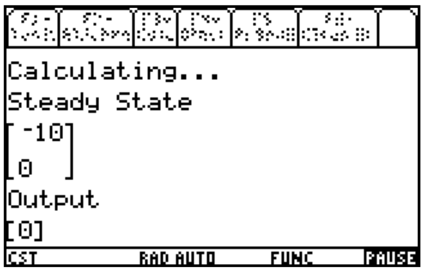


```
Calculating L⁻¹(W(s)U/s)...
.011·e^(-110·t) - .011·e^(-1000 ▸
CST   RAD AUTO   FUNC   PAUSE
```

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

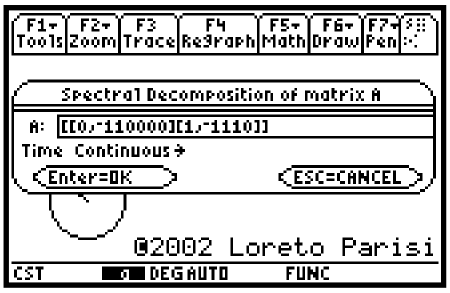


```
Steady State parameters
A: [[0,-110000][1,-1110]]
B: [[0][10]]
C: [[0,1]]
D: [[0]]
Input: 1
Enter=OK    ESC=CANCEL
©2002 Loreto Parisi
CST   DEG AUTO   FUNC
```

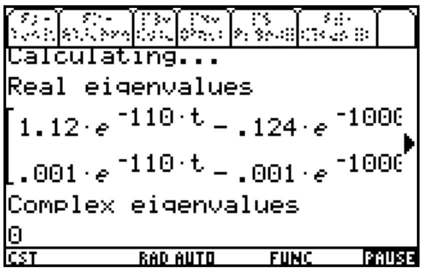


```
Calculating...
Steady State
[-10]
[0  ]
Output
[0]
CST   RAD AUTO   FUNC   PAUSE
```

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

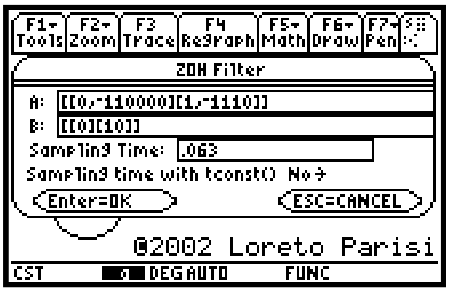


```
Spectral Decomposition of matrix A
A: [[0,-110000][1,-1110]]
Time Continuous →
Enter=OK    ESC=CANCEL
©2002 Loreto Parisi
CST   DEG AUTO   FUNC
```

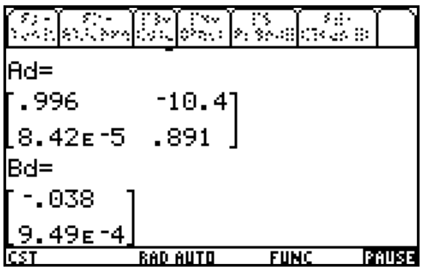


```
Calculating...
Real eigenvalues
[1.12·e^(-110·t) - .124·e^(-1000
[.001·e^(-110·t) - .001·e^(-1000 ▸
Complex eigenvalues
0
CST   RAD AUTO   FUNC   PAUSE
```

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



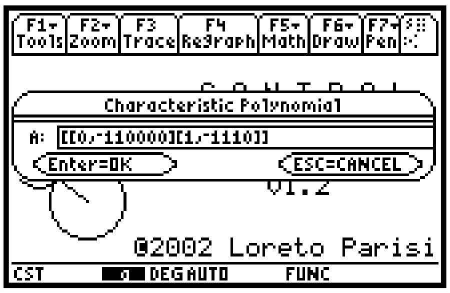
```
ZOH Filter
A: [[0,-110000][1,-1110]]
B: [[0][10]]
Sampling Time: .063
Sampling time with tconst() No →
Enter=OK    ESC=CANCEL
©2002 Loreto Parisi
CST   DEG AUTO   FUNC
```



```
Ad=
[.996      -10.4]
[8.42E-5   .891 ]
Bd=
[-.038  ]
[9.49E-4]
CST   RAD AUTO   FUNC   PAUSE
```

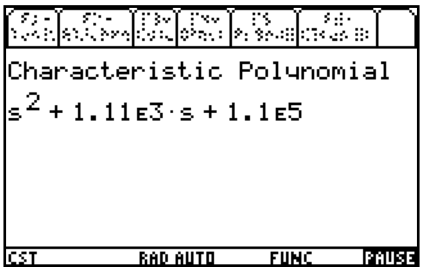
Notice that we have let *tconst(SYS)* to calculate the sample time T_c used in discretization.

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:



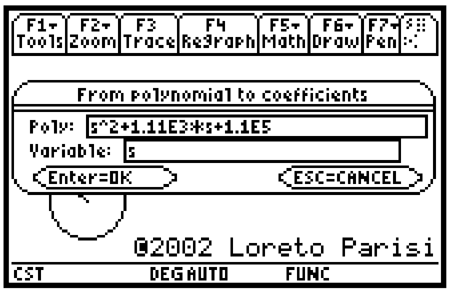
Characteristic Polynomial

A: [[0,-110000][1,-1110]]



```
Characteristic Polynomial
```
$s^2 + 1.11E3\cdot s + 1.1E5$

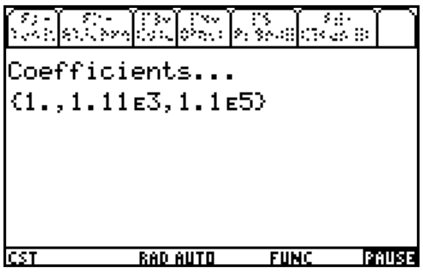
First, get p(s) with *poly(A)* from *Analysis* menu (*F2*).



From polynomial to coefficients

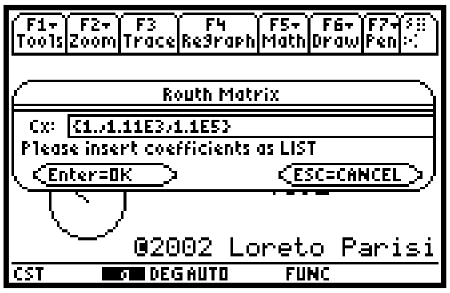
Poly: s^2+1.11E3*s+1.1E5

Variable: s



```
Coefficients...
{1.,1.11E3,1.1E5}
```

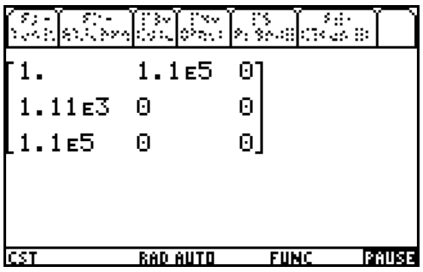
Second, get coefficients using function *poly2cof(expr, var)* from *Tools* menu (*F5*). These will appear in the next dialog window.



Routh Matrix

Cx: {1.,1.11E3,1.1E5}

Please insert coefficients as LIST



$$\begin{bmatrix} 1. & 1.1E5 & 0 \\ 1.11E3 & 0 & 0 \\ 1.1E5 & 0 & 0 \end{bmatrix}$$

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

Current Version

Current version of Control System Toolbox *for TI-89*: *version 1.2.3*
Date of release: *October 2003*
CST Reference Guide: *Fourth release, October 2003*

Improvements:

- Install Tool v. 1.2
- Error management
- Session management
- Data Menu
- c2d(SYS,Tc)
- sampler(A,B,Tc)
- gstep(w_1(t))

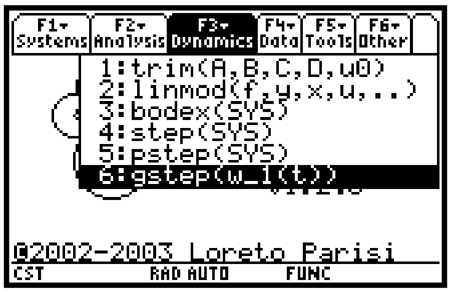
Details:

Install Tool v.1.2
See section *How to Install.*
Error Management
This latest version of CST come with a robust internal error management, that prevents the user to lose all data of the current working session, due to an unspected abort or error.
Session Management
The user can now save the current working session (i.e. the current transfer functions W(s) and W(z), the step response w_1(t), the state space, and all related and calculated parameters – Tc, step response parameters, ecc.) from *File* toolbox and *Save* menu. The saved session can be loaded from *Load* menu, simply selecting *Session*. All data of the latest saved working session will be restored. This overwrites the current working session, so it must be used only if you wish to load the last saved session.
- *Data Menu*
Now it is possible to show even the current discrete transfer function W(z), the current State Space and to obtain Magnitude and Phase for W(s) or W(z) in a simpler way.
- *c2d(SYS,Tc)*
This funcion now can save the resulting discrete transfer function as the current W(z) used in the current working session.
- *sampler(A,B,Tc)*
This function can now save the resulting state space as the current state space to be used in the current working session.
- *gstep(w_1(t))*
This new function plot the step response w_1(t) obtained with *step(SYS)* or directly specified in its menu. It can use the step response parameters previously calculated by

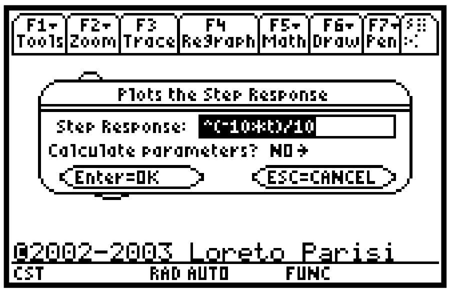
pstep(SYS) to evalutate w_1(t) on the plot and to use an auto-scaling function to calculate the correct zoom factor to show the plot. If there aren't calculated step response parameters, it will use standard values for the zoom factor. See section *New Feature* for more details.

New Features

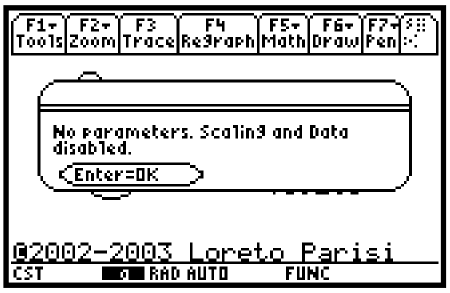
Let see *gstep(w_1(t))* in detail. After calculating the step response w_1(t) using *step(SYS)*, please select *Dynamics/gstep(w_1(t))* – F3 and then 6 – to plot the step response. In this case we have not calculated its parameters yet with *pstep(SYS)*.



```
1:trim(A,B,C,D,u0)
2:linmod(f,y,x,u,..)
3:bodex(SYS)
4:step(SYS)
5:pstep(SYS)
6:gstep(w_1(t))
©2002-2003 Loreto Parisi
```

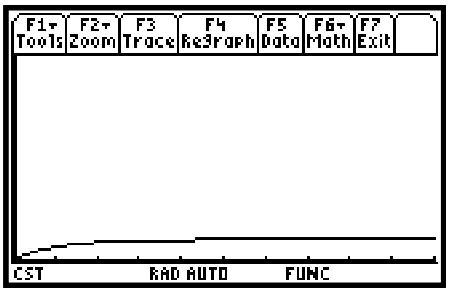


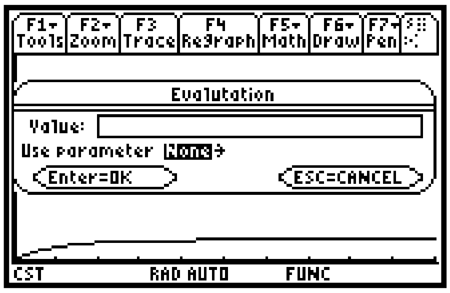
```
Plots the Step Response
Step Response: ^(-10*t)/10
Calculate parameters? NO
©2002-2003 Loreto Parisi
```



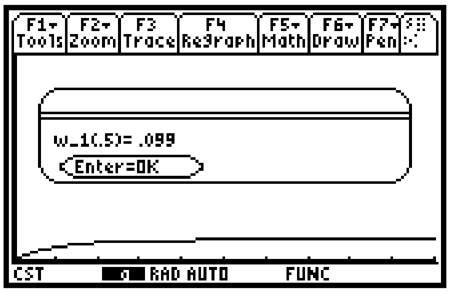
```
No parameters. Scaling and Data disabled.
©2002-2003 Loreto Parisi
```

The lack of parameters do not permit to use the auto-scaling feature to obtain a zoom that matches with the specified step response. Moreover it is not possible to use the *Data* menu to show step response parameters.



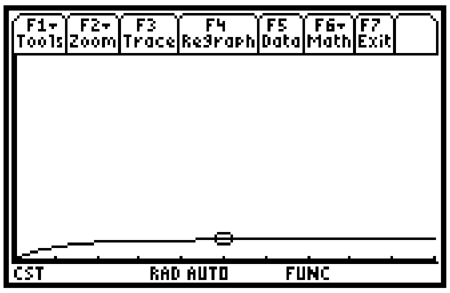


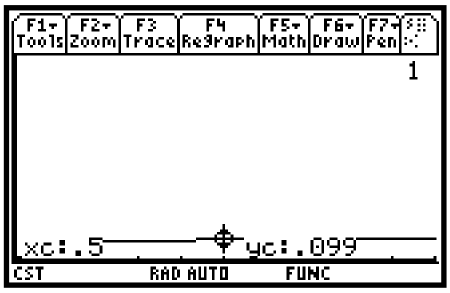
```
Evalutation
Value:
Use parameter None
```



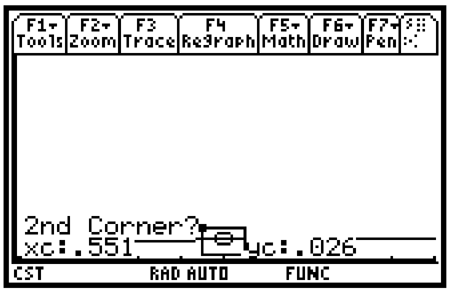
```
w_1(.5)= .099
```

It is possible to evalutate w_1(t) in a point and see this point on the plot. You can trace and zoom the function too.



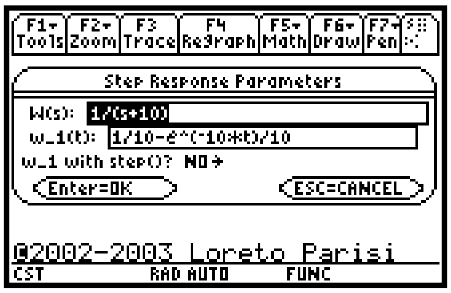


```
xc:.5 yc:.099
```

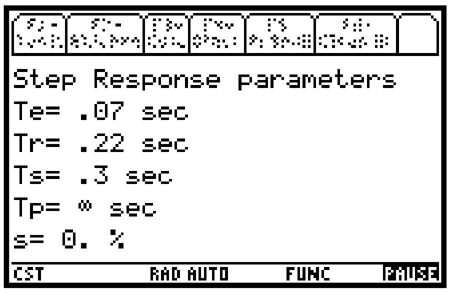


```
2nd Corner?
xc:.551 yc:.026
```

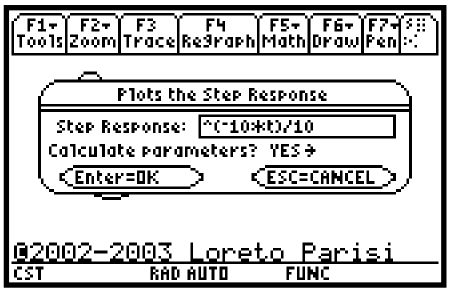
Suppose now to calculate parameters with *pstep(SYS)* and run *gstep(w_1(t))* selecting YES to use them.



```
Step Response Parameters
W(s): 1/(s+10)
w_1(t): 1/10-e^(-10*t)/10
w_1 with step()? NO
```

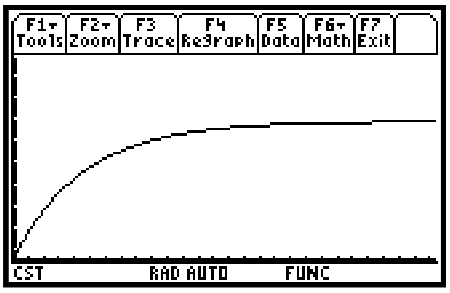


```
Step Response parameters
Te= .07 sec
Tr= .22 sec
Ts= .3 sec
Tp= ∞ sec
s= 0. %
```

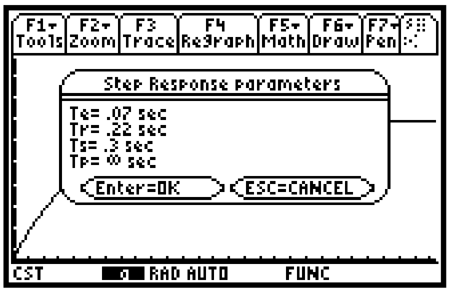


```
Plots the Step Response
Step Response: ^(-10*t)/10
Calculate parameters? YES
```

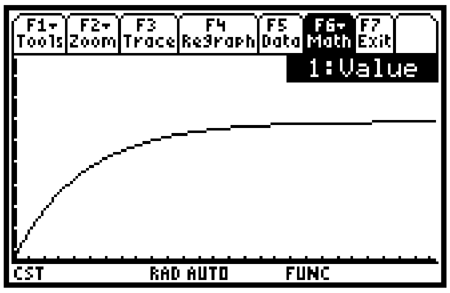
The auto-scaling feature is enabled and the zoom factor now matches with the order of the step response parameters.It is possible to use Data menu to show this parameters.



F1 Tools F2 Zoom F3 Trace F4 Regraph F5 Data F6 Math F7 Exit
CST RAD AUTO FUNC

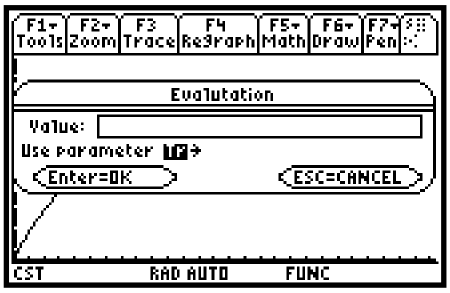


```
Step Response parameters
Te= .07 sec
Tr= .22 sec
Ts= .3 sec
Tp= ∞ sec
```

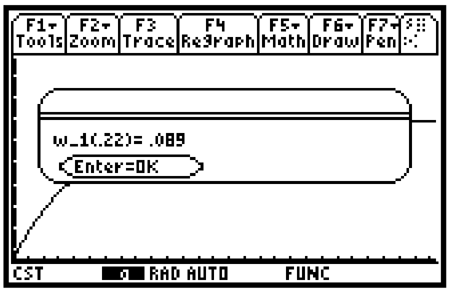


F1 Tools F2 Zoom F3 Trace F4 Regraph F5 Data F6 Math F7 Exit
1:Value
CST RAD AUTO FUNC

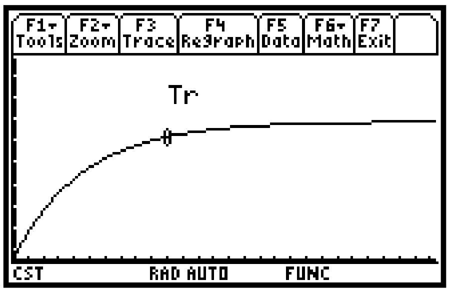
Now we can use *Math* – *Value* to evalutate w_1(t) in these specific points. Please select a parameter, such as Tr (raise time – as shown). We'll have this point on the plot.



```
Evalutation
Value:
Use parameter Tr
```

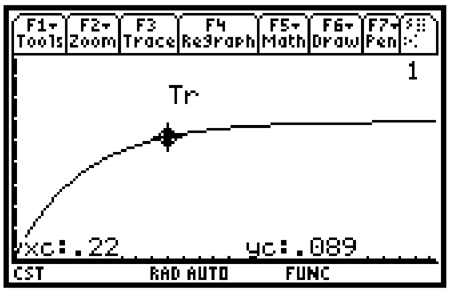


```
w_1(.22)= .089
```

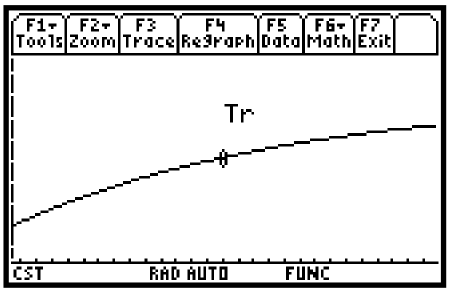


F1 Tools F2 Zoom F3 Trace F4 Regraph F5 Data F6 Math F7 Exit
Tr
CST RAD AUTO FUNC

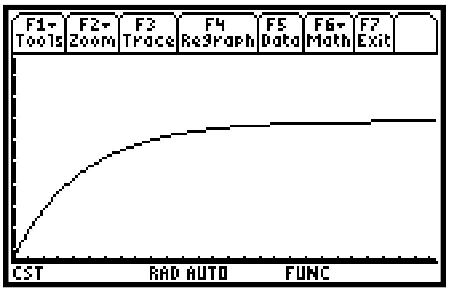
We can *Trace* the step response and zoom it. Use *Regraph* (F4) and *Zoomprev* (from the *Zoom* menu – F2, then 9) to restore the plot.



```
xc:.22          yc:.089
```

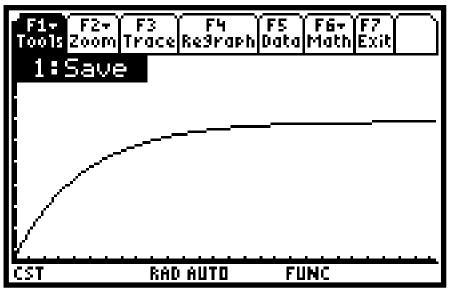


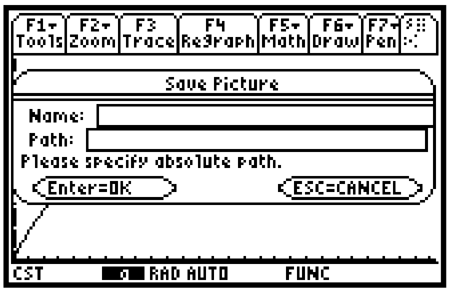
F1 Tools F2 Zoom F3 Trace F4 Regraph F5 Data F6 Math F7 Exit
Tr
CST RAD AUTO FUNC



F1 Tools F2 Zoom F3 Trace F4 Regraph F5 Data F6 Math F7 Exit
CST RAD AUTO FUNC

Finally, it is possible to save the plot as a picuture using the *Tools* menu and *Save* option – press F1, then 1.

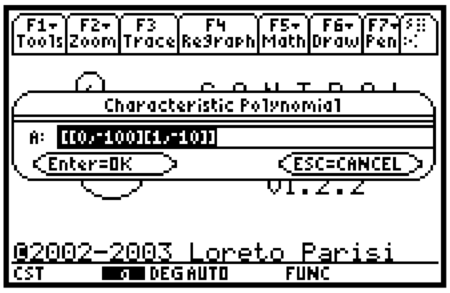


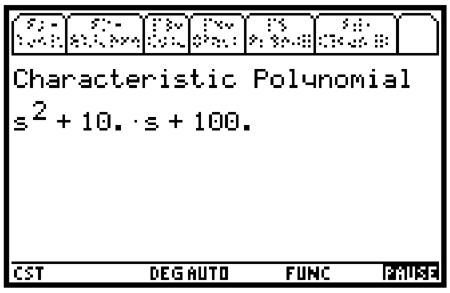


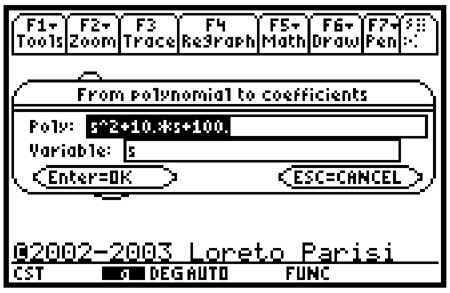
Name:
Path:
Please specify absolute path.

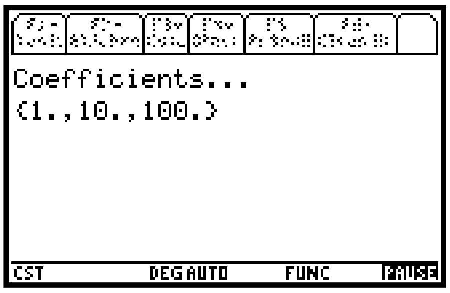
Tips

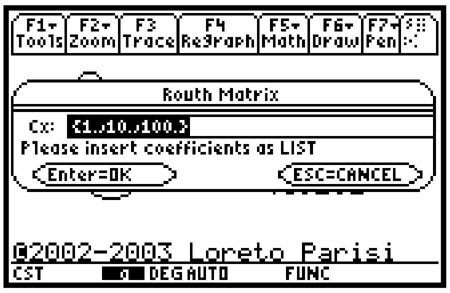
Question: *How can I get coefficients of characteristic polynomial to calculate poles or Routh matrix ?*
Solution: From *Analisys* menu (*F2*) choose *poly(A)* to get characteristic polynomial, then from *Tools* (*F5*) choose *poly2cof(EXPR,VAR)* and input ‘s’ as variable. Now choose *cpoles(Cx)* or *routh(Cx)* from *Tools* menu. Here are the sequence of operations:

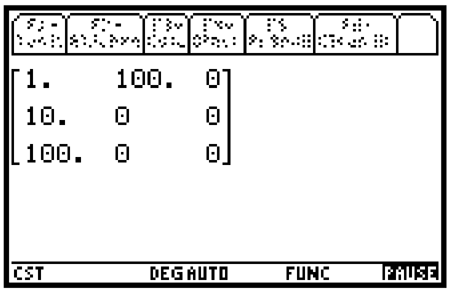

Characteristic Polynomial
A: [[0,-100][1,-10]]


```
Characteristic Polynomial
s^2 + 10. ·s + 100.
```

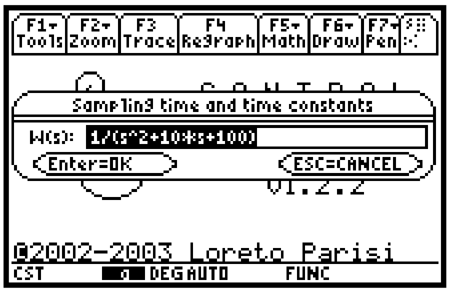

From polynomial to coefficients
Poly: s^2+10.*s+100.
Variable: s

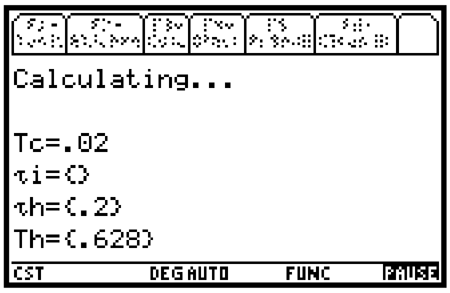

```
Coefficients...
{1.,10.,100.}
```

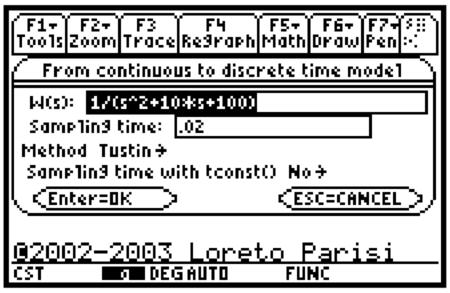

Routh Matrix
Cx: {1.,10.,100.}
Please insert coefficients as LIST

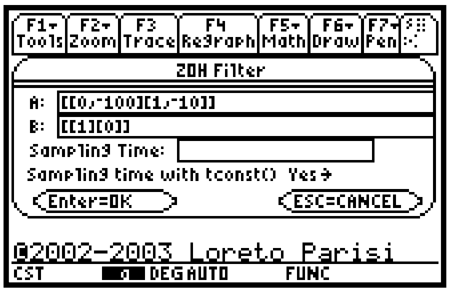

$$\begin{bmatrix} 1. & 100. & 0 \\ 10. & 0 & 0 \\ 100. & 0 & 0 \end{bmatrix}$$

Question: *How can I obtain sampling time T_c to be used with c2d(SYS,Tc), that performs continuous to discrete time model conversion using bilinear Tustin transformation, or with sampler(A,B,Tc) that generates a discrete time model using ZOH filter ?*
Solution: From *Analisys* menu (*F2*) choose *tconst(SYS).* So you get all time constants of your system. Now choose *c2d(SYS,Tc)* from *Systems* menu (*F1*) or *sampler(A,B,Tc)* from *Tools* menu (*F5*). Here are the steps:


Sampling time and time constants
W(s): 1/(s^2+10*s+100)


```
Calculating...

Tc=.02
τi={}
τh={.2}
Th={.628}
```


From continuous to discrete time model
W(s): 1/(s^2+10*s+100)
Sampling time: .02
Method Tustin →
Sampling time with tconst() No →



ZOH Filter

A: [[0,-100][1,-10]]
B: [[1][0]]
Sampling Time:
Sampling time with tconst() Yes →

Alternatively, you can directly choose from *c2d(SYS, Tc)* or *sampler(A,B,Tc)* menu the option *Sampling time with tconst()* with *Yes.* This is useful if you have not calculated T_c yet, or if you want to input a different sampling time – in this case simply choose *No* and input that value in *Sampling Time* text field.

Contents

After installation of *CST*, in the 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
cstse_	LIST
cstver_	STR
d2c	FUNC
damp	FUNC
db	FUNC
dcgain	FUNC
degzero	FUNC
eigenv	FUNC
gain	FUNC
gstep	PRGM
help	PRGM
install	PRGM
linmod	FUNC
linspace	FUNC
logspace	FUNC
mag	FUNC
magz	FUNC
mreach	FUNC

File name	*File type*
peak	FUNC
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.

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

The users

Who help CST to grow up better and faster!

Thanks all folks!

What next ?

Next version of Control System Toolbox *for TI-89 release 1.3* will improve global performances and will introduce some news, such as

- New proprietary function to plot Bode diagrams, *bode(SYS)*
- New function to calculate Root Locus, *rlocus(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