TI-86
Nodal Circuit Analysis

Terry Rogers

27 Sept 98

Description & Purpose

This program is a nodal circuit analysis for use in analysis of simple circuits. It can produce the voltage transfer function at a single frequency in list (text) output format and as a graphical frequency sweep.

Implementation & Limitations

The program is menu driven for analysis but makes use of the TI-86's built in matrix editor for component input and alternate means for viewing output. It has a graphical output but no graphical input. The user must draw the circuit separately, number the nodes and enter component values and connections into matrices titled R, L, C and GM which hold resistors, inductors, capacitors and dependent, voltage controlled current sources (transconductances). There are no other types of components. Dependent voltage sources and transistors must be constructed from transconductances and resistors.

The program extracts component values and connection nodes from component matrices, translates R into a real and L & C into a complex admittance dependent upon the frequency and loads an admittance matrix. It then hands off this matrix with a current vector having just one automatically created entry to the TI-86 simultaneous equation solving function receiving a voltage vector as an output. This complex voltage vector is used to create amplitude (decibels) and phase version of the transfer function which in most cases is just the translated values in the voltage matrix since the input node is almost always one volt. A menu permits viewing the complex ratio of any two node voltages as amplitude and angle without repeating the analysis. Built in TI-86 matrix editing may be used to view the admittance matrix or voltage vector directly. Also:

- You must number nodes from 1 to N where N is the number of nodes. You may not skip any nodes. That is, at least one component must be connected to each node.
- You do not need to use a resistor to ground on each node. Nodes might have only reactive (L & C) components connected to them (the TI-86 computes the solution using complex arithmetic and not by using real number matrix equivalencies as in many PC circuit analysis programs).
- A single, independent, 1 volt, zero degree angle source is always connected between node one and ground. This source is constructed from a one ampere source from ground to node one in parallel with a one ohm resistor. In the analysis output, this source appears to be a zero ohm, one volt source because the output is normalized (divided) by the actual node one voltage. This combination is for convenience in viewing the voltage vector.
- Components are elementary, lumped values. Element Q and parasitic resonances must be simulated by their Thevenin or Norton equivalents, by adding components in series or parallel. Transmission lines are not supported. The only asymmetric component is the transconductance. That is, the terminals of all R,L & C components may be reversed without effect but not those of transconductances.
- There are no non-linear elements.
- Component values are in ohms, henries, farads and MHOs for R, L, C and Gm respectively.
- Node zero is reserved for ground.

The size of the network is limited by available memory and your patience in waiting for the solution. Memory limitations are liberal so that if there were nothing else and matrix math did not eat it up during a solution, it might handle 300 parts and 50 nodes (but then the battery right run out before a solution). As a practical matter, a 20 node circuit requiring 2½ to 3 minutes for a single frequency in list analysis mode or a 7 node circuit with 10 frequency sweep in about the same time is about as large as one might want to consider. The program is written as one main and four sub-programs for efficiency and clarity. They are:

- Circuit: main program, handles user I/O
- zSOLVE: Component matrix parameter extraction. Calls zYval and zGval to load admittance matrix and TI-86 built in simult( ) for solution. Prefix z is to move it to the *back end* of the program menu because one does not run this program directly.
- zCLRY: initializes the admittance matrix
- zYVal: a sub-program that loads R,L,C to admittance matrix
- zGVal: a sub-program that loads Gm to admittance matrix

Requirements

A TI-86 and:

- 2900 bytes of free memory for the program
- 30 bytes of free memory for each component plus 9 bytes for each matrix overhead (50 bytes for transconductance)
- 40 bytes per node for vectors
- 20 x N^2 bytes for admittance matrix where N is number of nodes

Typically that would be ~4600 bytes for a 7 node circuit and ~13000 bytes for a 20 node circuit.

All of matrices R,L,C and GM must exist. If components of a given type (e.g. C) does not exist, then all values in that matrix should be zero. See section Component Matrix Formats.

Menu

The first five menu items appear immediately. The <more> menu key is required for the rest.

Freq	List	Param	Plot	Exit	Nodes	dB Nd

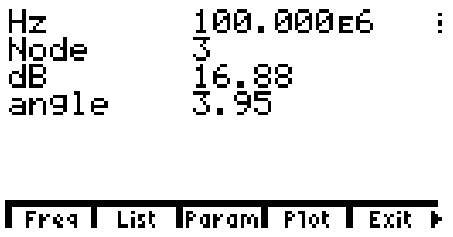
The menus are made for maximum convenience. For example, it's necessary to exit the program to change component values since the built in matrix editor is used for that purpose. However, after exiting the editor, just press <enter> key and the Circuit Analysis program starts up immediately (it was the last command). If one presses either List or Plot, the corresponding analysis is completed without having to reenter frequencies, plot parameters or nodes.

Freq(uency for list output analysis)

Enter the frequency in Hertz (e.g. 100E6 for 100 MHz) for the list output. This does not affect frequencies that will be used for the plot output sweep analysis. Note that you can do arithmetic in the entry line. E.g. 1.4*100E6. The resulting frequency will be printed out during list output analysis.

List (output analysis)

Performs circuit analysis at frequency entered above and lists the ratio of the voltage at the node entered under the Nodes menu with node number one (usually one volt at node one). You can get the ratio of other nodes using dB Nd menu below. *Angle* is in the preferred units you have set with *mode*.

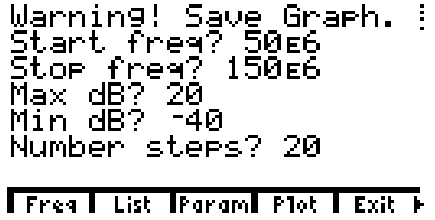


Also, it is possible to stop the program with *exit* and use the matrix editor to see the voltage vector V or just press <*alpha*> V and *enter*.

Param(eters for plot output sweep analysis)

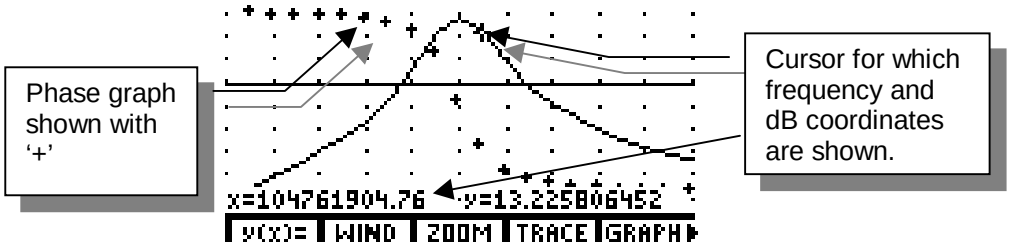
Enter parameters for controlling the plot output sweep analysis. A warning is issued that when Plot menu is pressed, the graphical image from the previous analysis will be lost and could be saved first (as a picture: Graph menu. See Storing Graphs). Also, **the built in function or other graphs will be turned off** (the functions will not be deleted) **and the graph control parameters will be altered**. You can save these parameters as Graphics Database under the Graph menu and submenu STGDB. You can recover those graphs with RCGDB. Doing so will recover functions and all graphing parameters completely as long as you save before pressing <plot>.

1. Start Freq?: Left side or start of frequency sweep in Hertz. Positive number.
2. Stop Freq?: Right side or end of frequency sweep in Hertz. Positive number.
3. Max dB?: The highest displayed magnitude in dB. This could be a negative number (use the (-) key and not subtraction key) but greater than item 4. Run list analysis at spot frequency first to find dynamic range.
4. Min dB?: The lowest displayed magnitude in dB. This could be a negative number and must be less than item 3.
5. Number of steps?: Expecting a positive integer. Other values will not be rejected but may cause unexpected results. A number higher than 125 is silly as that is the number of pixels used for the entire graph width. A number higher than 40 is exhausting. Except at the nose of a band-pass or zero of a filter, 10 steps gives fairly good results with good speed. One could then zoom in on this region by resetting parameters and still use 10 steps with good results.



Plot (sweep analysis)

Pressing plot starts a sweep analysis of which one is shown below. The solid line is amplitude in dB relative to node 1 and the plus signs show phase and the frequency interval at which analysis occurred.



.

In the graph above, menu <exit> has been pressed so that the program has halted bringing up the TI-86's native graph menu. Cursor keys have been used to move the cursor to 104.7MHz (shown in Hz) and 13.225dB which has been found graphically to be the 3dB bandwidth point. However, none of the graph menus shown work because they relate to function graphing and this screen was produced by using the *Line*( ) command (drawing on a graph). Other graph menus to the right (not shown) which refer to picture storage do work. The menu can be cleared by pressing <clear>. Cursor still operates.

- **Note:** If you set radian mode, the angle graph will be a straight line near zero. The angle graph is scaled only for degrees and goes from 180° at the top of the graph to -180° just above the menus as shown above.
- **Note:** The graphics cursor cannot be set to explore the angle graph, only the dB graph.
- **Note:** If you recall a picture, the graphics cursor readout will be correct only if you have previously run an analysis (pressed plot) after setting parameters the same as for the stored graph.

Exit

Stops the program so as to be able to:

1. View and inspect (with graphics cursor) graphs
2. Store & retrieve graphs (using graphic/STPIC and RCPIC menus)
3. Edit the network (using matrix editor)
4. Store and retrieve networks (by copying R,L,C & Gm matrices)

In most cases, it's possible to restart the program just by pressing <enter> since the last command was <Circuit>. In any other case, the program/name menu can be used to restart the program. All list and graphing analysis set-up is retained.

You may stop an analysis in progress and exit the program when the exit menu is not visible by pressing and holding the <on> key until the break message appears. Then press quit.

Nodes

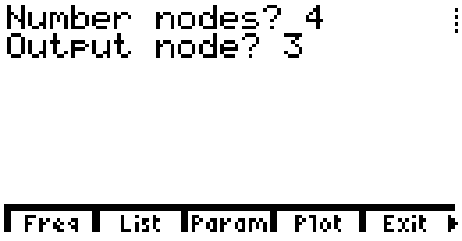
This menu is used to set the size of the network and the output node. Some extra code might have been able to set the number of nodes correctly but not the desired output node. Failure to set network size correctly (or at all) is a common source of error. If the network dimension is set too small, one will see:

ERROR 13 DIMENSION

after pressing either list or plot menus and while the screen is displaying *Loading Rs* or Ls, Cs or Gm. If the network dimension is too large, one will see:

ERROR 03 SINGULAR MAT

after all R, L, C and Gm have loaded and after it says *Solving*. The nodes entry display will look similar to:

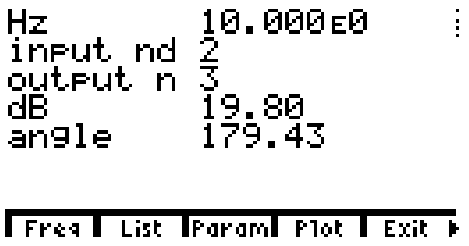


dB Nd (Nodes)

This menu permits indirect inspection of the V( ) or voltage vector without running another analysis. The *output node* set in the Nodes menu uses node one as a reference but perhaps you would like to know the ratio of some other two nodes. The menu might look like:

reference node? 2
output node? 3

After pressing <enter> the screen might look like:



angle is in whatever units are set using the *mode* editor.

Note: This menu does not set or reset the output node for List of Plot analysis.

Component Matrix Formats

This input format used to be called *free form* in that components may be entered in any order (at least in a given matrix). The net listing version can look like:

R10 1 2 10k

Which means the component is a resistor, number 10 (of resistors), connected from node 1 to 2 and which has a value of 10kΩ. The TI-86 is capable of handling strings and might have been programmed to store the input network in string or ASCII format as above or perhaps it could have been made to appear as above and just use R,L,C and GM matrices as in the present design. However, the TI-86 already has an editor for this purpose which can be easily called from the keyboard after pressing <exit>. Use <2nd><matrix> and then select <edit>. Now type or select either R,L,C or GM. Note that all of these four matrices must exist for the program to work. If one of the matrices does not exist, you will see:

ERROR 14 UNDEFINED

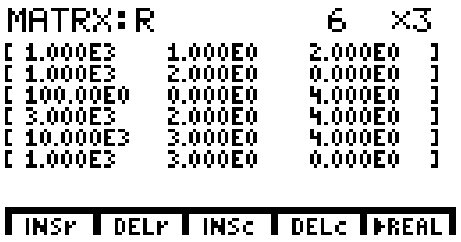
When running List or Plot analysis. If there are no components of a type, that matrix must still exist but all values should be set to zero.

R, L, C

These hold resistors, inductors and capacitors with values in ohms, henries and farads respectively. Each is a Nx3 matrix where N is the number of components of this type. If there are 3 resistors, then the R matrix is a 3x3, if 5 resistors, 5x3. The format is:

<value> <node 1> <node 2>

However, value may be = 0 in which case the component is removed from the circuit without changing the size of the matrix. Here is a particular R matrix.



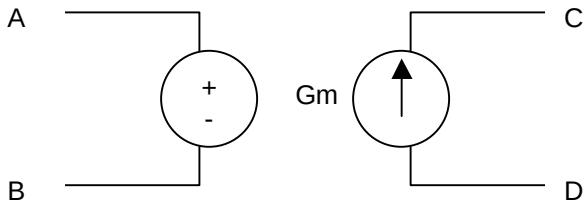
Which means that there is:

Resistor value ohms	1st node	2nd node
1000	1	2
1000	2	0 (ground)
100	0 (ground)	4
3000	2	4
10000	3	4
1000	3	0 (ground)

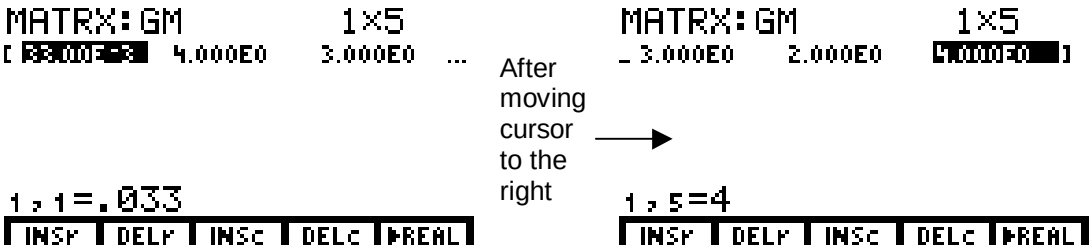
You can use the menus *INSr* and *DELr* to insert and delete rows to add or delete components or you can set the value in the 1st column to zero to delete a part. Press the <exit> key to quit editing. Usually just pressing <enter> key will restart the Circuit program.

Gm

This matrix holds transconductances. The circuit symbol is:



The convention is that a positive voltage (A – B) , causes a current to flow from D to C of magnitude Gm (A – B). If (A – B) were 2 volts and Gm = 4, then the current from D to C would be 8 amperes. Current is *conventional* (or Benjamin Franklin) current (positive charge) such that C is made more positive relative to D. Either A or B but not both may be node zero or ground. Either C or D but not both may be zero or ground. The program will not detect that an error has been made. You will just get an incorrect result. Here is an example:



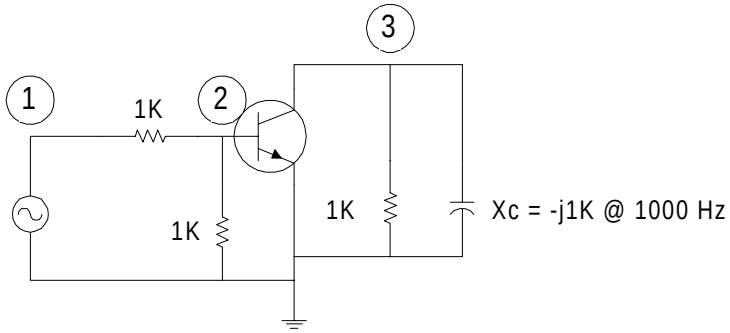
Which means:

Value MHOs	C node	D node	A node	B node
.033	4	3	2	4

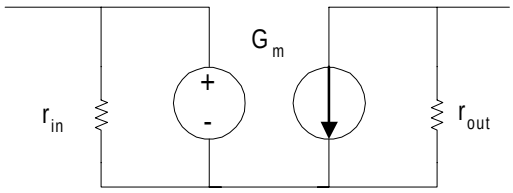
Two of the nodes may be tied together as shown above to produce a three terminal device. Note that the letter node designations are not in sequence. The current source is defined first followed by the voltage control nodes.

List Analysis Example

Here is a transistor amplifier showing input and output loads (not bias elements, bias evaluation is not supported). Nodes are numbered in the circles. The value of the output capacitor is not specified directly. It has a reactance of –j1000Ω at 1000 Hz.



Transistors as such are not supported in this simple nodal analysis, just transconductances. However, most data sheets contain enough information to translate the transistor to the following combination of transconductance with input and output load resistance which is valid at low frequencies (below the gain roll off point with the specified loads). More components must be added to the model for high frequencies.



Input and output resistances can be read directly from the data sheet. At low frequencies, typical values are r_{in} = 3KΩ and r_{out} = 10kΩ. Also a typical d.c. and low frequency current gain is β = 100.

Transistor transconductance is defined as the ratio of change in collector output current with change in base to emitter input voltage, output collector voltage being held constant (zero ohm load).

$$g_m = \frac{\partial i_c}{\partial v_{be}}\bigg|v_{ce} = cons\tan t$$

Equation 1

Also, β is defined as the ratio of the change in collector current to change in base current:

$$\beta = \frac{\partial i_c}{\partial i_b}\bigg|v_{ce} = cons\tan t$$

Equation 2

and r_{in} can be defined as the ratio of the change in base-emitter voltage to change in base current:

$$r_{in} = \frac{\partial v_{be}}{\partial i_b}$$

Equation 3

Solve Equation 3 for ∂v_{be} and substitute in the denominator of Equation 1.

$$g_m = \frac{\partial i_c}{r_{in}\partial i_b}$$

Equation 4

The circled term is just current gain or β so that Equation 4 may be written:

$$g_m = \frac{\beta}{r_{in}}$$

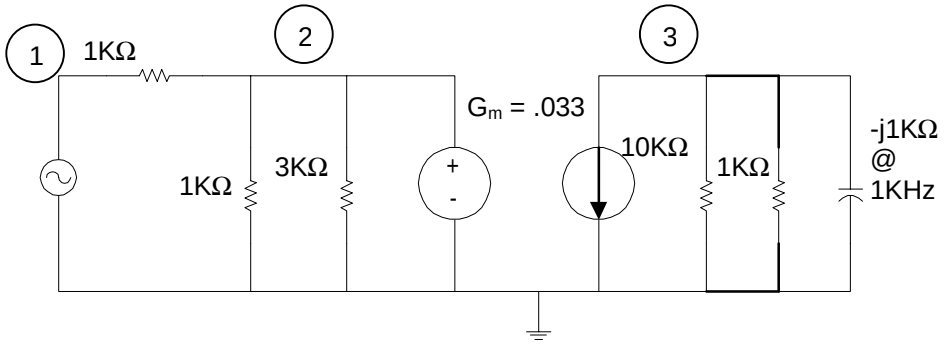
Equation 5

In this case:

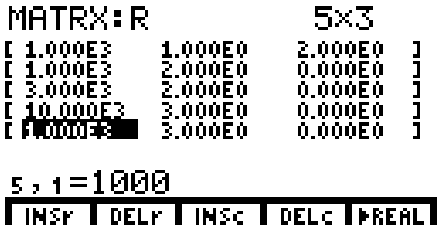
$$g_m = \frac{100}{3k} = .033$$

Equation 6

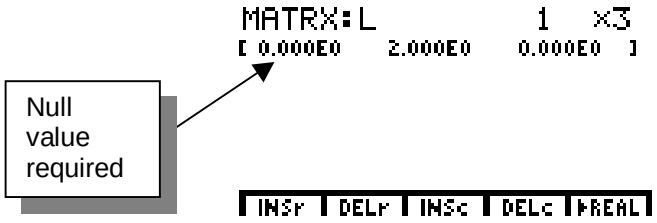
So that the translated model is as below.



Resistors and other components may be in parallel. This is how the circuit appears in the matrix editor.



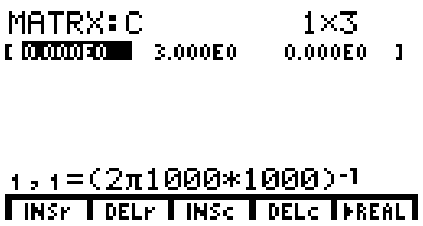
There are no Ls so a null value with any node numbers is required.



It's possible to do arithmetic in the entry line for the capacitor. It is -j1kΩ at 1kHz so it is:

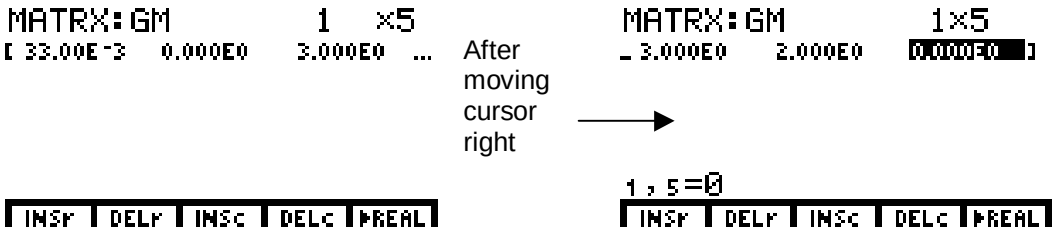
$$C = \frac{1}{2\pi f X_c}$$

Equation 7



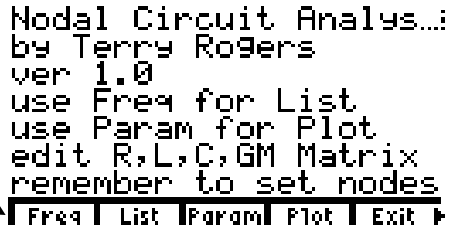
Which is 159nF.

…and G_m matrix:

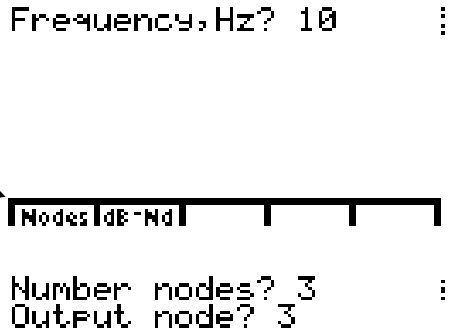


The circuit program is started:

Now press Freq and enter 10 Hz.

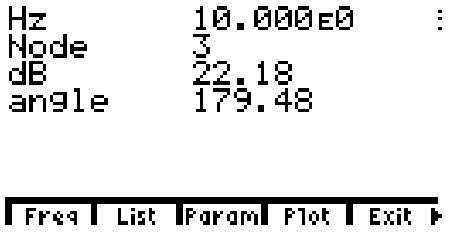


Now press <more> key and press Nodes key to set number of nodes (3) and the List analysis output node (also 3 in this case but it may be any node).



Now press <list> analysis.

You should get:



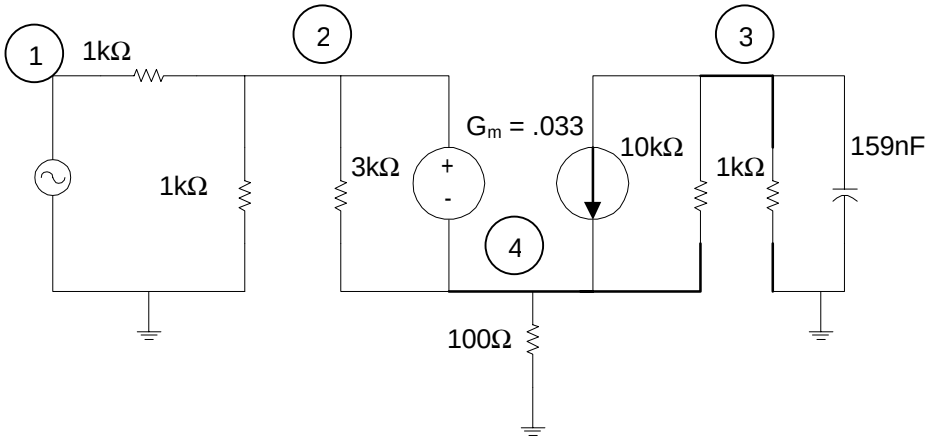
Note that bipolar transistors invert the phase at the collector (180°) and that voltage gain is $A_v = g_m r_l$ where g_m is .033 mhos and r_l (output load) is the combination (at frequencies << 1000Hz) of 10kΩ||1kΩ = 909Ω and .033 x 909 = 30 which is 29.54dB. However, the input resistor combination of 1kΩ in series with 1kΩ || 3kΩ divided the input voltage by a factor of .429 which is –7.36dB. Note that 29.54 + (-7.36) = 22.18dB, which is the answer given by the circuit analysis program.

Press <more> and then <*dB Nd*> or dB between nodes. Enter reference (input) node 2 and output node 3. You should get this screen.

```
reference node? 2
output node? 3

Hz          10.000E0
input nd    2
output n    3
dB          29.54
angle       179.48

Freq | List | Param | Plot | Exit
```

Without repeating the analysis, this is the gain between nodes 2 (base input) and node 3 (collector output). It is indeed 29.54dB as calculated above.

The circuit may be modified by placing an emitter resistor such that:



Edit the R matrix by pressing INSr which means *insert row*.

```
MATRX:R                    5×3
[ 1.000E3    1.000E0    2.000E0  ]
[ 1.000E3    2.000E0    0.000E0  ]
[ 3.000E3    2.000E0    0.000E0  ]
[ 10.000E3   3.000E0    0.000E0  ]
[ 1.000E3    3.000E0    0.000E0  ]

3,1=3000
INSr | DELr | INSc | DELc | ▶REAL
```

Then add the 100Ω resistor. Also remember to edit these two resistors to match the schematic nodes.

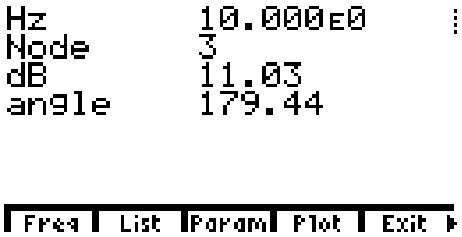
```
MATRX:R                    6×3
[ 1.000E3    1.000E0    2.000E0  ]
[ 1.000E3    2.000E0    0.000E0  ]
[ 100.00E0   0.000E0    4.000E0  ]
[ 3.000E3    2.000E0    4.000E0  ]
[ 10.000E3   3.000E0    0.000E0  ]
[ 1.000E3    3.000E0    0.000E0  ]

5,3=4
INSr | DELr | INSc | DELc | ▶REAL
```

Also edit the G_m matrix to make node connections correct.

```
MATRX:GM                   1×5
[ 3.000E-3   4.000E0    3.000E0  ...  -3.000E0   2.000E0   0.000E0  ]

1,1=.033
INSr | DELr | INSc | DELc | ▶REAL
```

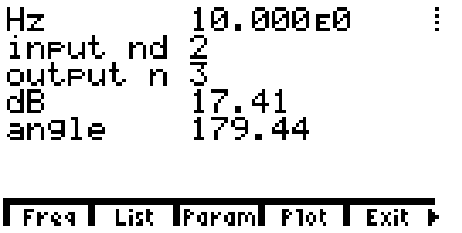
```
MATRX:GM                   1×5
[ 3.000E-3   4.000E0    3.000E0  ...  -3.000E0   2.000E0   4.000E0  ]

1,5=4
INSr | DELr | INSc | DELc | ▶REAL
```

Exit the matrix editor and just press <enter> (if the last command was *circuit*, otherwise restart *circuit* program). Now press *Nodes* and enter 4 nodes but keep the output node as 3. Now press *List*. (If you forgot to reset the size of the network, you got ERROR 13 DIMENSION) You should see this screen.

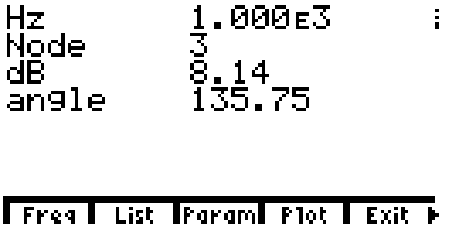


We can inspect the gain from node 2 as before. We know the open loop gain was (from above) 29.54dB or a voltage gain of 30. The feed back fraction or β is 0.1. The feed back equation for voltage gain is $A_v' = \frac{A_v}{1+\beta A_v}$ which in this case is

$$A_v' = \frac{30}{1+0.1*30} = 7.5 = 17.5dB .$$

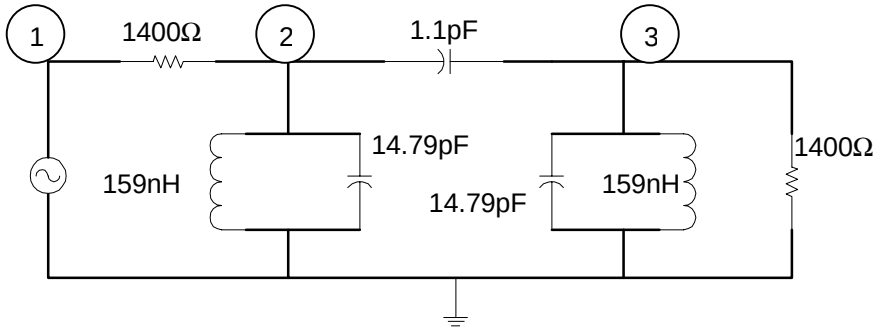


Now change the frequency (*Freq* key) to 1000Hz and press *List* again. You should see this screen. (Compare gain to two screens above). The gain at 10Hz was 11.03dB and 3dB less (the capacitor has the same reactance as the load resistor at this frequency) is about 8.14dB. Also 180-45 (lag) = 135°.



Graphical Analysis

Here is another circuit which is a two pole 100 Mhz band pass filter.



See section Storing Circuits for how to save the previous circuit if you want and then enter the circuit above by using the matrix editor for matrices R, L and C. Remember to enter a g_m value of 0 (zero) for the only entry in the GM matrix. You cannot delete any of the matrices R, L, C or GM. A value of zero effectively removes a component (or use DELr matrix editing command for all but the last component of a type.

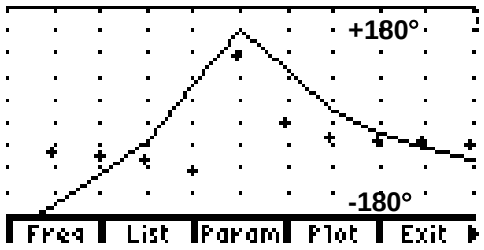
This circuit is resonant at 100 MHz. Start the circuit program and press *Freq* to set 100 MHz. Then set the nodes menu to 3 nodes and output node is 3. Now press *List*. You should see:

```
Hz          100.000E6   ⋮
Node        3
dB          -6.03
angle       92.14

 Freq | List | Param | Plot | Exit ▸
```

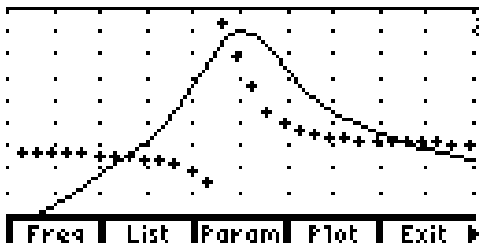
The voltage gain –6.02dB is standard for a lossless network with the same input and output impedances. If you use the *dB Nd* menu with node in = 2 and node out = 3, you will get nearly zero loss at -.29dB.

Now try 50 MHz, the lowest frequency to be sweeped. You should get –59.24dB and for 150MHz, the highest frequency to be sweeped, -39.31dB. This defines the range for the graphical parameters. Press *Param* and set:

```
Warning! Save Graph. ⋮
Start freq? 50E6
Stop freq? 150E6
Max dB? 0
Min dB? -60
Number steps? 10

 Freq | List | Param | Plot | Exit ▸
```

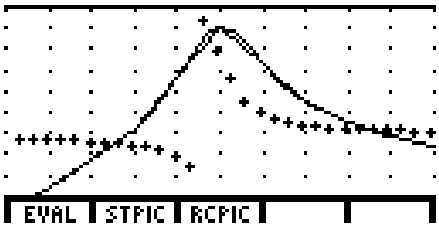


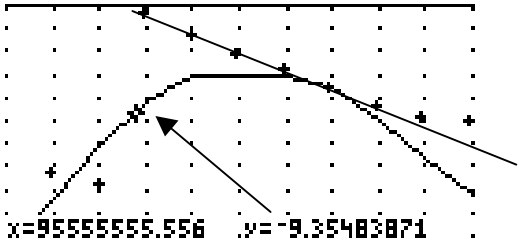
The warning is a note to save either a previous circuit analysis graph (see section Storing Graphs) or the built in graphing parameters (see section Param(eters for plot output sweep analysis)). 10 steps is nice to begin, a reasonable compromise between accuracy and speed. Most of the time, 10 steps will be adequate. Now press *Plot*. At first nothing seems to happen (but see the *wooly worm* crawling in the upper right of the display indicating that the processor is hard at work). Then this display traces out across the screen. I added +/-180° labels latter. They are approximately in the correct positions and indicate the approximate phase traced out by '+' signs. Phase *wraps* in this example at -180° to +180° near the center of the graph. The solid line is dB. Review the section Plot (sweep analysis) for information on how to stop the program and use built in cursor inspection of the graph.



Stop the program by pressing *Exit* and use the *more* key to get to *STPIC*. Store this picture under *N3* (note: each picture uses 1020 bytes). Now return to the *circuit* program and reset *Params* the same way but 30 steps. Press *Plot* (and go get a cup of coffee). You should get the graph at the right. Exit and inspect 3dB band width. Then use the graphical menu to recall graph N3.

You can see differences between 10 point and 30 point sweeps are not as great as one might think. All of the difference is at the *nose* area. Therefore, it is better to just *zoom* into the nose area by resetting *Params* such that you are sweeping from 90MHz to 110MHz with max dB = 0 and min dB = -20dB and 10 steps. You get the graph below.

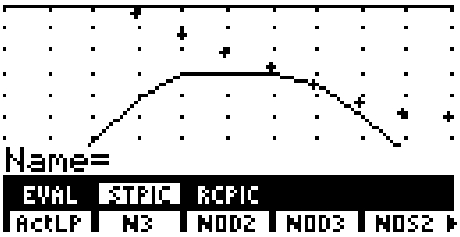




In the graph at the left, *clear* was pressed to remove the graphing menu after which cursor keys explored 3dB band width as shown. The circuit was intended to be +/- 5MHz centered at 100 MHz but it is skewed a little high, as is typical of capacitive coupling. I drew a straight line over the phase graph (you can just hold a straight edge to the screen). If the graph departs even slightly from a straight line, the group delay is varying at the frequency in a significant way. Here, group delay is bad (for pulse and TV signals anyway) at the +/-3dB points.

Storing Graphs

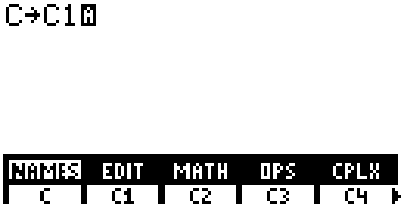
Storing graphs was mentioned in the section above. Press <*graph*> if necessary to restore the graph menu (the *circuit* program must be stopped) and <*more*> twice to get the menu shown two graphs above. Press *STPIC* (store picture) and either press an existing name to over-write or type in a new name. You can use the menu *RCPIC* to recall any of the named pictures.



- **Note:** Recalling the graph does not recall scaling parameters. If you have been using a different *Params* set up (other than number of steps), then the displayed scale is wrong.
- **Note**: You can't delete a picture/graph from the graph menu. You must exit and use the *MEM* menu to do that.
- **Note:** Each graph takes about 1020 bytes.

Storing Circuits

Store and recall circuits by storing and recalling sets of R,L,C, GM matrices. The easiest way is just to name sets as 1,2,3… as in R2,L2,C2 and GM2 etc. Press *matrix* and then *names*. Select C, press *STO→* and then C again but add the digit 2. Then press *enter*. Do this for each matrix of R, L, C and GM. The amount of storage is proportional to the number of elements as described in the section Requirements. Recalling circuits is similar but press C1 first, then *STO→* and then C and *enter*. Also do so for the other matrices.



- **Note:** I was stupid enough not to store plot *params* in a matrix and so you cannot store that as well. You have to remember what the analysis *param* or *Freq* and node set up was. This programming exercise is left for the student.

Appendix

How you could make it better

Want to do some programming? Here are some ways to make this program more useful.

- Change the frequency sweep to interlaced so that it solves and plots points only for frequencies 1,4,7,10…, then 2,5,8,11…, then 3,6,9,12… Then connects all the points in a line graph. This way, parts of the response at many frequencies are seen sooner so an analysis with a bad result can be stopped. If it is OK, then you get high resolution without restarting the analysis.
- Alternately, provide a sweep analysis that takes finer steps in the middle of the sweep or when the slope changes rapidly (2^{nd} derivative of dB graph becomes relatively large).
- Make components so that values may be complex. In this way, some parasitic elements may be included and there would be some approximation of high frequency operation for transistors.
- Provide s parameter definition for transistors. Consult circuit analysis texts for s to y parameter conversion and make s to y conversion transparent. Make transconductance values complex. Once converted to y parameters, values can be added to Y matrix in the same way and still use TI-86's simult() for solution.
- Allow arbitrary node numbering and automatic network sizing.
- Call the matrix editor from inside the circuit analysis program so that it appears to be a part of it. Addresses for many routines can be found on the TI-86 web site.
- Make it possible to save analysis parameters by perhaps placing them in a matrix and saving it the same way as R,L,C, GM.
- Make network saving and recovery a menu item.

Variables Used

An attempt was made to avoid commonly used variables (at least what I thought was common). Most variables are lower case (I noticed most of the listings used upper case) or start with cXXX for *circuit* analysis. You can inspect the programs below for the names of variables used (the program is not *structured* or *designed* in any way. I wrote it during lunch hours munching away so design documentation is sparse).

Commented Source Code

This program is not copy writed. I place it in the public domain. It is supplied *as is.* No claims are made or implied for its fitness for any purpose nor can any claims be made for damages or consequences of its use. The author informs you that it is inadvisable to use analysis output of this program for any circuit or device used in a critical function, especially in a life support role as the program has not been thoroughly tested.

The basic methodology is very simple. If you write a series of node current equations for a circuit using Kirchoff's law restricting yourself to independent or dependent current sources and R, L, C elements, you notice that the system of equations can be resolved into three matrices such that:

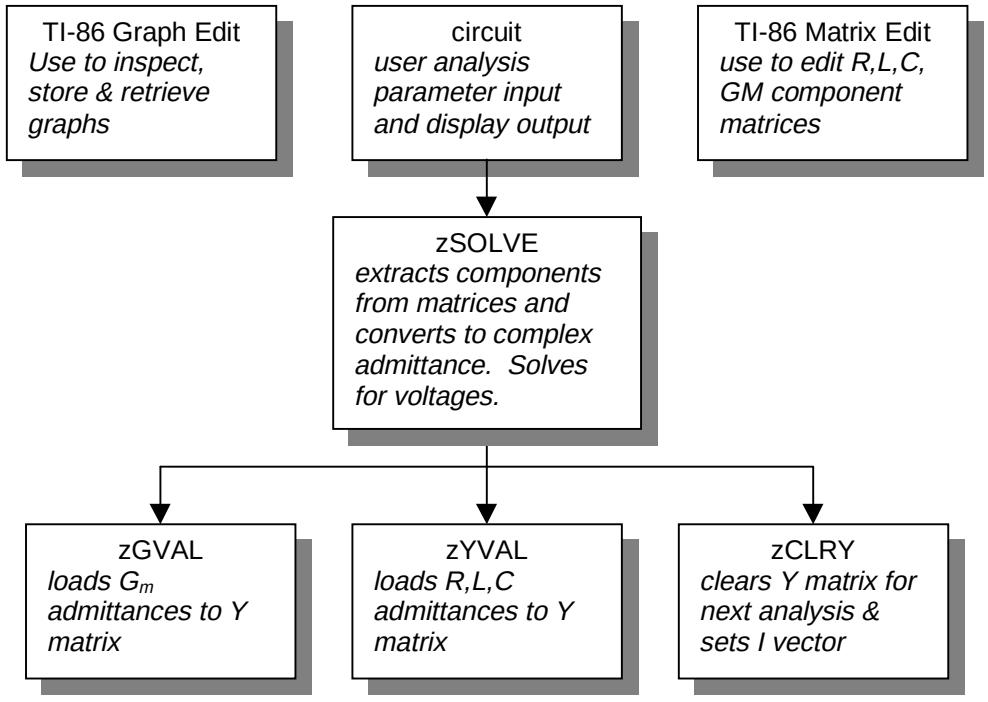
$$Y \times V = I$$

Where:

Y	is the admittance matrix of network components
V	is the unknown voltage vector
I	is the independent current vector (with just one entry in this program)

It's precisely this system of complex matrices (equations) that can be solved with the TI-86's *simult()* function. You will also see that there is a very methodical way to load the Y matrix. If a passive element is connected between, say nodes 2 and 3, then its complex admittance will be summed / added to the admittance of Y matrix elements $Y_{2,3}$ and $Y_{3,2}$. It will also be subtracted from elements $Y_{2,2}$ and $Y_{3,3}$ on the diagonal (or the other way around. If all signs are reversed on a row, a matrix remains the same). Therefore the diagonal must always be non-zero. (You can inspect the Y matrix with the editor like any other to see that this is so. You can also see V or I vectors with the vector editor.) Transconductances are not symmetric but you can find the rules for loading the Y matrix in any circuit analysis text or by inspecting the program. The point is that the rules are straight forward and that a large part of this program is used to extract components from the R,L,C, GM matrices and load them into the Y matrix. Finding the solution is only one line. Most of the rest of the program deals with displaying the results. The TI-86 has so much built in that this program is not very large. Certainly the smallest circuit analysis program that I have ever seen.

Here is the hierarchical (calling sequence) diagram.



In the text listing below, \ xxx \ means *escape* sequence where the characters between are edited into an internal TI-86 form. It's mostly obvious what the key entry would be. For example, \->\ means use the key STO→ and \pi\ means use π. If keying this program in through the TI-86 keyboard, you can use this listing. Otherwise, use the program files with this document and TI-Graph Link.

In this listing below, /* this is a comment */ and is not supported by the TI-86. Don't key in anything ***/**** between the comment characters */. Lines that would be obvious by reading the TI-86 manual are not commented.

Program *circuit*

```
:ClLCD:Disp "Nodal Circuit Analysis"
:Disp "by Terry Rogers":Disp "ver 1.0"
:Disp "use Freq for List"
:Disp "use Param for Plot":Disp "edit R,L,C,GM Matrix"
:Disp "remember to set nodes"
:Dec  /* place calculator in decimal mode in case left in HEX or
other mode */
:Lbl Cstart
:Menu(1,"Freq",FENT,2,"List",LSOLVE,3,"Param",GPARAM,4,"Plot",GSO
LVE,5,"Exit",CQUIT,6,"Nodes",NOUT,7,"dB\(-)\Nd",cdbnode)
:Lbl FENT
:ClLCD:Input "Frequency,Hz? ",freq
:Goto Cstart
:Lbl NOUT
:ClLCD:Input "Number nodes? ",NN
:Input "Output node? ",OUTN
:Goto Cstart
:Lbl LSOLVE /* list analysis */
:2\pi\freq\->\P2F /* create 2πf parameter used for L & C value to
admittance conversion as in Yc = +j2πfC.  Avoid this
multiplication for each part. */
:1\->\cGRAPH /* flag for zSOLVE to display 'loading R' et.al. */
:zCLRY:zSOLVE     /* create & clear complex Y admittance matrix
and solve the network */
:Eng:Fix 3  /* switch to engineering notation for frequency
output */
:ClLCD:Outpt(1,1,"Hz"):Outpt(1,10,freq)
:Normal:Fix 0     /* switch to 'integer' mode for node display */
:Outpt(2,1,"Node"):Outpt(2,10,OUTN)
:Fix 2  /* switch to 2 decimal fixed for dB & angle display */
:Outpt(3,1,"dB"):Outpt(3,10,20(log abs V(OUTN)-log abs V(1)))
:Outpt(4,1,"angle"):Outpt(4,10,angle V(OUTN)
:Goto Cstart
:Lbl GPARAM /* Input graph sweep analysis parameters.  Values are
stored in non-system variables, i.e. not xMax so that the program
can be started w/o reentering params if function graphing is used
*/
:ClLCD:Disp "Warning! Save Graph."
:Input "Start freq? ",cF1
:Input "Stop freq? ",cF2
:Input "Max dB? ",cY1
:Input "Min dB? ",cY2
:Input "Number steps? ",cSTEPS
:(cF2-cF1)/cSTEPS\->\DELTAF
```

```
:125/cSTEPS\->\dxPIX
:Goto Cstart
:Lbl GSOLVE /* graphical sweep analysis */
:Func:RectGC:CoordOn    /* set graph to function & rectangular
coordinates (X vs. Y) and turn on coordinate display to inspec
graph with cursor.  The display might be in another mode like
polar coordinates. */
:GridOn:AxesOn:FnOff 1,2,3:ClDrw
:0\->\cGRAPH  /* flag to turn off 'loading R' et.al. display */
:cF1\->\xMin:cF2\->\xMax  /* set graph scale max & min */
:(xMax-xMin)/10\->\xScl  /* set x scale for grid */
:cY1\->\yMax:cY2\->\yMin
:(yMax-yMin)/10\->\ySCl
:cF1\->\f1  /* set first frequency for analysis */
:2\pi\f1\->\P2F:f1\->\xlast  /* 2πf parameter for admittance load
& remember last frequency for LINE() function coordinate */
:f1+DELTAF\->\f1  /* increment sweep frequency */
:zCLRY:zSOLVE     /* solve network */
:20(log abs V(OUTN)-log abs V(1))\->\ylast  /* find value to
graph by comparing output node voltage to node 1 on an absolute
basis.  Note that one could change this line to use the two node
parameters from Lbl cdbnode. */
:For(kk,1,cSTEPS) /* sweep analysis loop */
:2\pi\f1\->\P2F:f1\->\xnow  /* save current frequency for plot */
:f1+DELTAF\->\f1
:zCLRY:zSOLVE
:20log (abs (V(OUTN)/V(1)))\->\ynow  /* save current value for
plot */
:Line(xlast,ylast,xnow,ynow)  /* plot line segment */
:xnow\->\xlast:ynow\->\ylast  /* move current points to last
points */
:int (26-.139*angle (V(OUTN)/V(1)))\->\cRow  /* this rather
strange calculation scales angle amplitude and offset to the row
size of the TI-86 LCD display.  Graph plotting does not support
dual scales.  Note you could change the scale factor, .139, to
support radian angles. */
:int (dxPIX*kk)\->\cCol /* same strange calculation for columns
*/
:PxOn(cRow,cCol):PxOn(cRow+1,cCol)  /* all these PxOn()
statements draw a small '+' on the display for phase output */
:PxOn(cRow-1,cCol):PxOn(cRow,cCol+1)
:PxOn(cRow,cCol-1)
:End
:Goto Cstart
:Lbl CQUIT  /* exit program leaving all set up parameters as they
were.  However, numerical output format is put back to some
nominal value.  You could change that to make it what you want.
*/
:Eng:Fix 3
:Stop
:Lbl cdbnode:ClLCD      /* node ratio display */
:Input "reference node? ",n1
:Input "output node? ",n2:ClLCD:Eng:Fix 3
:Outpt(1,1,"Hz")
:Outpt(1,10,freq)
:Normal:Fix 0
:Outpt(2,1,"input nd")
```

```
:Outpt(2,10,n1)
:Outpt(3,1,"output n"):Outpt(3,10,n2)
:Fix 2
:Outpt(4,1,"dB"):Outpt(4,10,20log abs (V(n2)/V(n1))
:Outpt(5,1,"angle"):Outpt(5,10,angle (V(n2)/V(n1))
:Goto Cstart
```

Program *zSOLVE*

```
:If cGRAPH:Then:ClLCD:Disp "Loading Rs"
:End
:\(-)\1\->\Y(1,1) /* this is the 1 ohm resistor from node 1 to
ground in parallel with the 1 amp generator.  It's not necessary
if you are not going to view V() directly and you don't 'strike a
pole head on (infinite input impedance) */
:dim R:Ans(1)\->\aa     /* find the size of R and thus how many
resistors */
:For(ii,1,aa)     /* extract nodes and value */
:R(ii,2)\->\n1
:R(ii,3)\->\n2
:R(ii,1)\->\yval
:If yval\!=\0:Then      /* skip if value is zero */
:yval\^-1\\->\yval      /* otherwise load to Y matrix */
:zYVAL
:End
:End
:If cGRAPH:Disp "Loading Cs"
:dim C:Ans(1)\->\aa
:For(ii,1,aa)
:C(ii,2)\->\n1
:C(ii,3)\->\n2
:C(ii,1)\->\yval
:If yval\!=\0:Then
:(0,yval*P2F)\->\yval
:zYVAL
:End
:End
:If cGRAPH:Disp "Loading Ls"
:dim L:Ans(1)\->\aa
:For(ii,1,aa)
:L(ii,2)\->\n1
:L(ii,3)\->\n2
:L(ii,1)\->\yval
:If yval\!=\0:Then
:(0,(\(-)\P2F*yval)\^-1\)\->\yval
:zYVAL
:End
:End
:If cGRAPH:Disp "Loading GM"
:dim GM:Ans(1)\->\aa
:For(ii,1,aa)
:GM(ii,2)\->\n1
:GM(ii,3)\->\n2
:GM(ii,4)\->\n3
:GM(ii,5)\->\n4
:GM(ii,1)\->\yval
:zGVAL
```

```
:End
:If cGRAPH:Disp "Solving"
:simult(Y,II)\->\V      /* solve network, the pay-off line */
:Return
```

Program *zYVAL*

```
/* see program description above */
:If (n1\!=\0 and n2\!=\0)
:Then
:Y(n1,n2)+yval\->\Y(n1,n2)
:Y(n2,n1)+yval\->\Y(n2,n1)
:End
:If n1\!=\0
:Y(n1,n1)-yval\->\Y(n1,n1)
:If n2\!=\0
:Y(n2,n2)-yval\->\Y(n2,n2)
:Return
```

Program *zGVAL*

```
/* see program description above */
:If (yval\!=\0)
:Then
:If (n1\!=\0 and n4\!=\0)
:Y(n1,n4)-yval\->\Y(n1,n4)
:If (n1\!=\0 and n3\!=\0)
:Y(n1,n3)+yval\->\Y(n1,n3)
:If (n2\!=\0 and n3\!=\0)
:Y(n2,n3)-yval\->\Y(n2,n3)
:If (n2\!=\0 and n4\!=\0)
:Y(n2,n4)+yval\->\Y(n2,n4)
:End
:Return
```

Program *zCLRY*

```
/* see program description above */
:{NN,NN}\->\dim Y
:Fill((0,0),Y)
:NN\->\dim II
:Fill(0,II):\(-)\1\->\II(1)
:Return
```