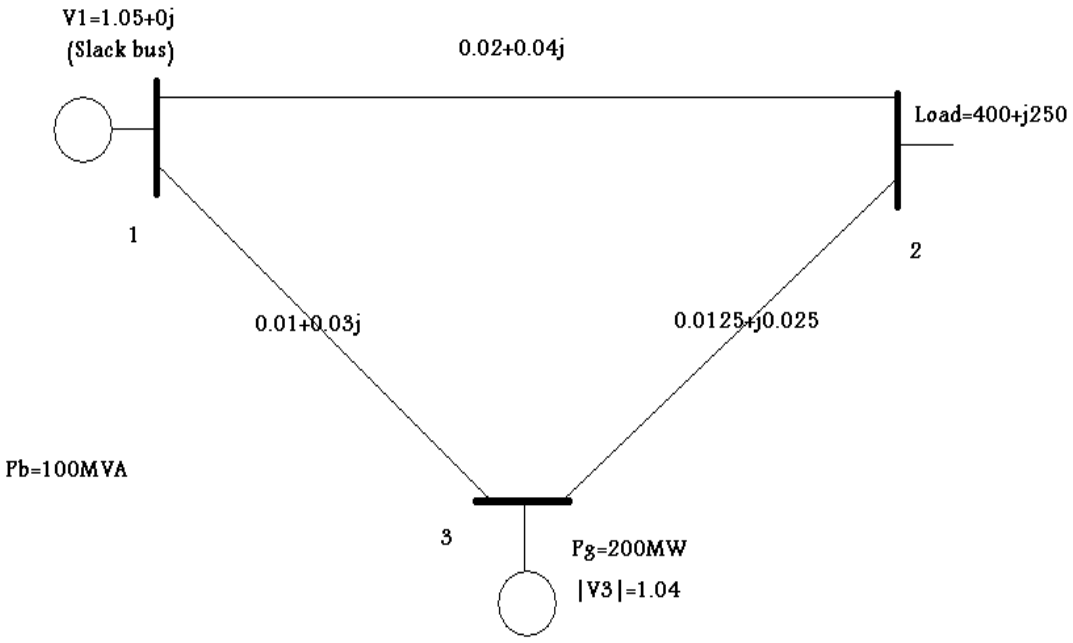
SALAM

What is LFNewton()?

"Load Flow using Newton Rophson Method"
Before anything I suggest you to study the LFGauss.pdf file which is included in LFGauss().zip package. By the way this package is completely independent.
As we will see through an example, learning how the program works is pretty easy. The same as LFGausse the formulas used in it are based on "Power System Analysis by Hadi Saadat" (more info.)
Actually the program uses a lot of RAM space if the network has more that 6 buses but it's not something to worry in TI-89 Titanium, you have much more RAM than needed, so I didn't compress the Ybus and vpqkxmvp matrices. (The second one is used to store information about voltages, active and reactive power and other characteristics of buses.)

How to install:
Just send the "shin_adm" and "LFNewton" files to you TI device. (Any folder but I use PSA for power system analyzing programs and functions.)
The shin_adm() is a program that generates YBUS matrix. This matrix will be stored in variable "xxx" and this is the only variable that is used by the program, other variables are local.
Another thing to mention is that both programs can be archived after running once, but the variable xxx must be unlocked so shin_adm can store YBUS in it.

OK, let's see **how it works:**
Imagine this system:



Before anything I suggest you to change all impedance values to admittance:

y12=10-20j

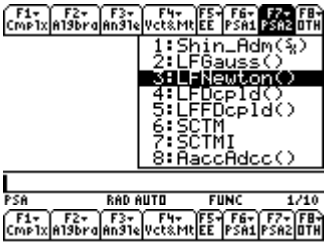
y13=10-30j

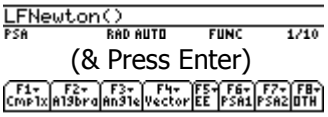
y23=16-32j

If you don't do this here, you will have to enter them in the program like this:

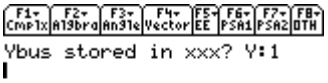
y12= (0.02+0.04j) ^-1

1. Run the program LFNewton() - (no parameters are needed.)

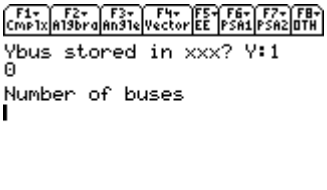




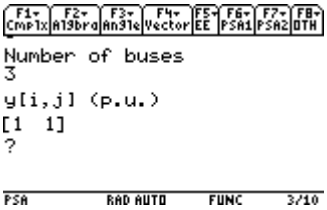
(& Press Enter)



2. The program is asking you about Ybus, if the question has given this matrix to you just store it in this variable and then when you reach hear enter 1, otherwise you should enter any number except 1 so the program automatically calls the other program shin_adm(n) and after a few steps it stores the Ybus in xxx. (n is the number of buses in network.)
   We assume that Ybus is not stored in xxx so for example I enter "0":

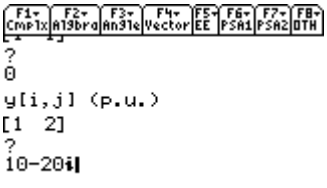


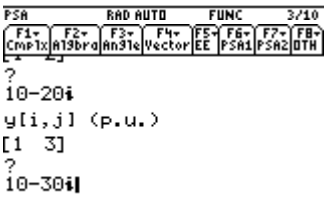
3. Enter number of buses. (3 in the above network)

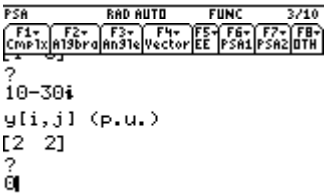


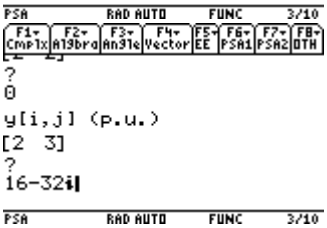
4. Entering the line data.
   - y [i , j] = the admittance between i and j buses. (Note that it's different from Yij in Ybus matrix.)
     The parallel admittance (if it exists in your line model) is not used here. You should just enter the amount of connected admittance to both i and j buses.
   - y [i , i] = the admittance between bus i and the earth (ground). If the line doesn't have parallel admittance (that is caused by capacitor between line and the earth) this amount is zero.

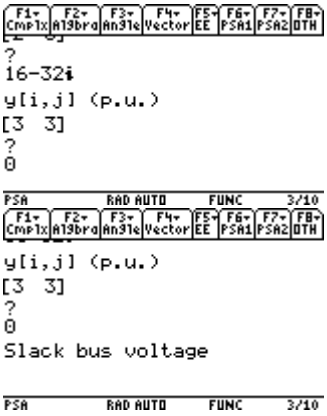
   So follow the pictures:









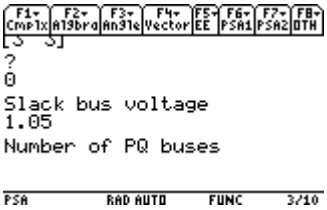


5. Up to now Ybus is created and stored in xxx.
   (You can generate this matrix in the same way by running shin_adm(n) program.)

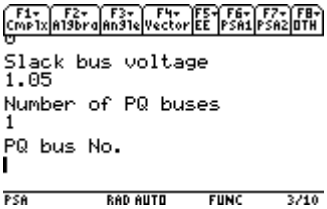
   As we saw the shin_adm(n) is a very usefull program to generate Ybus of a small network but if the network has more than 5 buses then this method of inputting data is not efficient, specially in real world problems. If you want to know why, read the blue text below:

   Another method of inputting line data:
   Another way is to input line data like this:
   [1,2,1+10j;1,3,4-5j;…;3,2;6-7j]
   And after inputting this matrix the program will understand that for example the admittance between bus 1 and bus 2 is 1+10j, and the same for bus 1 and 3, (4-5j), and all the other combinations which are not included are zero.
   If you test both methods you will find out that the one shown in forth step is faster for a small network. (Just try to input that matrix), but when we face a real world problem, it's totally different. Because many of buses are not connected to each other so you have to input zero many times; in this situation inputting line data in the form shown here is much faster. But our program is written to solve a problem in a test, and these kind of problems won't have more than 5 buses!
   Compressing Ybus:
   In real world problems we have lots of buses, for example, imagine a network with 30 buses, but every bus is only connected to a few other buses (usually something between 2 to 4 buses).
   Now most of the element of Ybus matrix become zero, in the above example Ybus is a 30*30 matrix, but each row has only 2 to 4 non zero elements, because the admittance between bus "i" and most of the other buses is zero (there is no line between them). In the past, computers didn't have enough memory to store this BIG matrix, so programmers suggested some ways to compress Ybus (it's not complicated at all, actually all zip programs do the same work, you can even write a program yourself to compress this matrix), but nowadays that computers have lots of memory it not needed to compress this matrix at all. (If you want to learn more about compressing this matrix I suggest you to study "Electric energy systems theory by O.I. Elgerd")

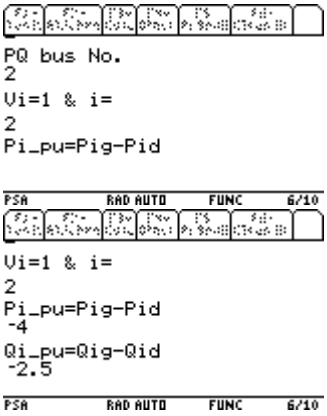
   Now is the time to enter bus data values:



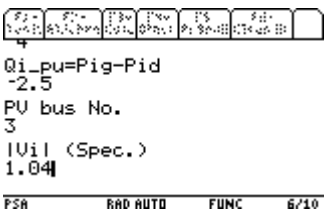
(I'm sure you know what is a PQ or a PV bus but just as a reminder, Slack bus, is a bus which is going to compensate the difference between load and generation. These kinds of power stations should have special characteristics that I'm not going to explain here. A PV bus is one which has a generator except the slack bus. PQ buses are the rest, they only have loads. Be careful that a capacitor bank changes a bus into generating bus or PV, because the capacitors generate reactive power.)



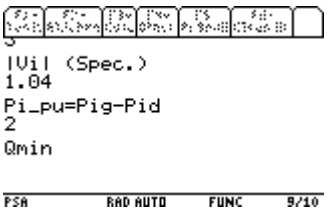
For every bus we assign a number, when using this program use number one for slack bus. (It's not a limit for program because the numbers are optional and in the other hand I can say that this is a standard, even if the question has given a different number for this bus just change the number with the number of a bus that is one.) On basis of the question's picture above, the PQ bus number is 2 and PV is 3.



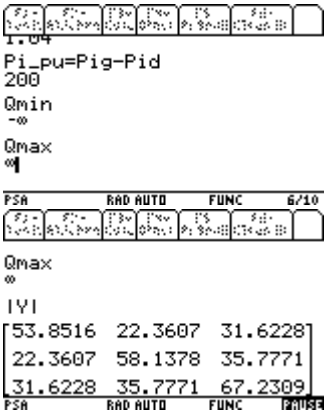
Another important point when using this or any other method of load flow is that everything should be "per-unite". (The base value of power in this system is 100MVA)



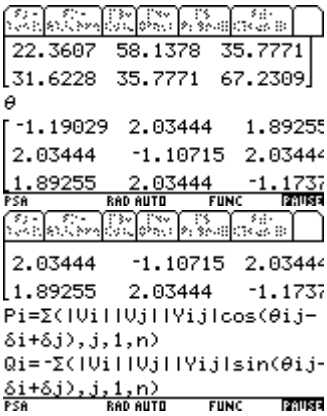
As you see the program automatically finds the number of PV buses, so you just need to enter bus number.



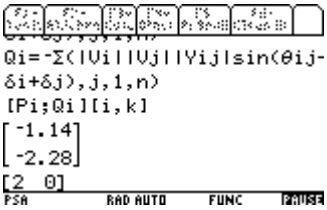
Most of generating buses have a reactive power generation limit, if there is also a load on the PV bus use Qmin=Qm-Qd in which Qm is the minimum generated reactive power by the generator and Qd is the demanded reactive power on this bus. If there is no limit to minimum reactive power for the generator, use -∞.

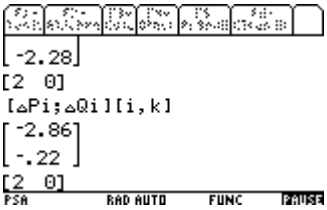


This is the absolute of Ybus matrix. If your system has more than 3 buses it probably won't fit on the screan, but don't worry, once you have finished you can generate this matrix by typing abs(xxx) at home screan. The same can done about angle of it: (angle(xxx))



(The program is showing the formulas which are going to be used. |Yij| and θij are the element of the matrixes shown in previous screens.)





If you have downloaded this program you probably are familiar with Newton-Rophson method in load flow so you should know what is the usage of ΔQs and ΔPs, if you don't refer to the Saadat's book.
As you can see bus number (i) and iteration counter (k) are shown in the matrix below the screen.

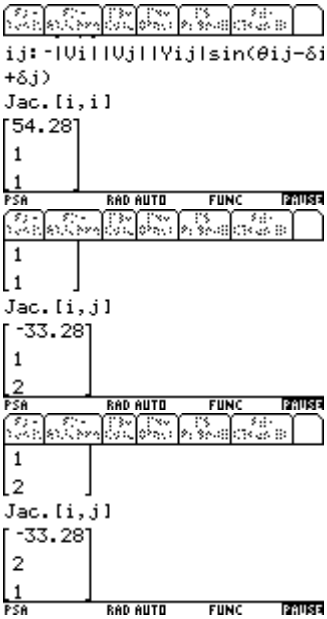
These are the formulas to calculate the elements of Jacobean matrix. In some references this matrix is shown like this:

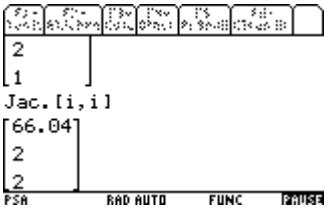
[H, N; J, L]

But here we show them like this:

[J1, J3; J2, J4]

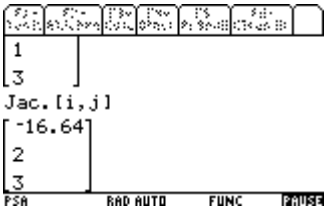
(The same as Hadi Saadat)



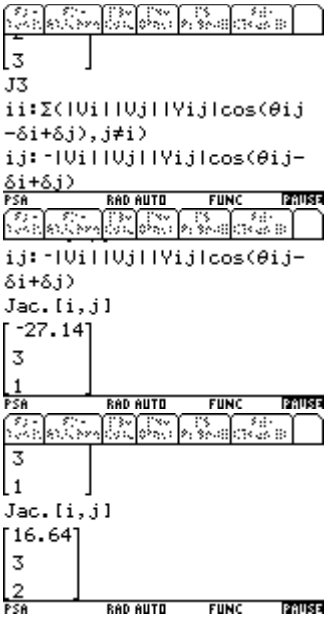


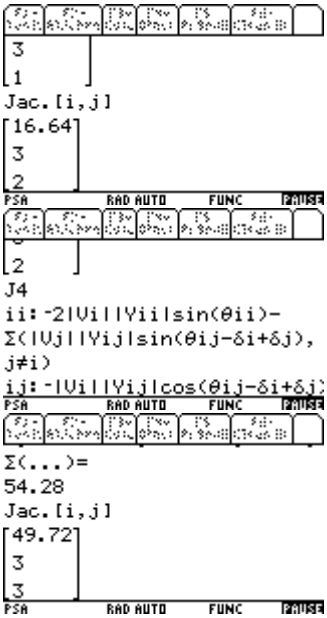
(Finished Calculating J1, the same process for J2, J3 and J4)

At the top of the screen you see the answer of sigma when calculating J2 [1, 1].

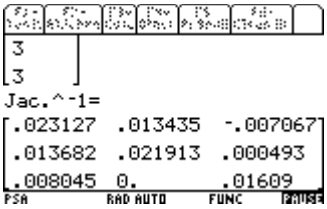


In this example J2 had only two elements.

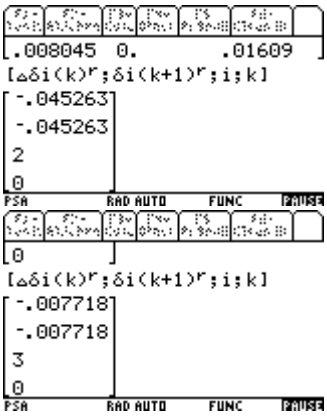


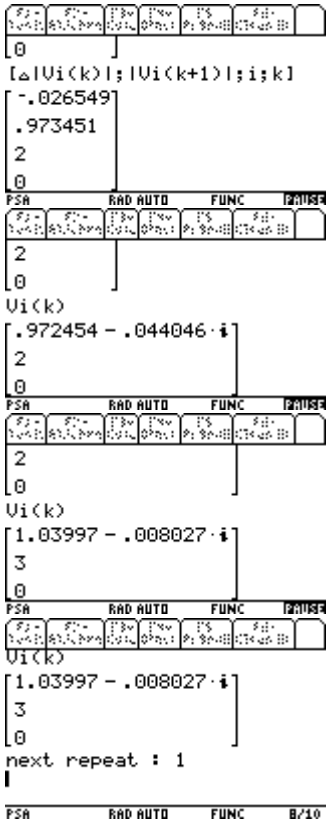


Calculating Jacobean matrix is finished.



(The Jacobean inverse is shown in this screen; if your Jacobean matrix doesn't fit to screen then if you want to generate this matrix you should do it manually in the main menu on basis of the Jacobean elements you have, however in usual problems (in standard tests I mean) the question is made in a way that usually this matrix wont exceed 3*3, for example if you have another PV in this question, then the row and column of that bus in Jacobean matrix will be omitted so your Jacobean would have only one element!
Do you know why?
Because in PV buses the voltage is constant (it's specified in the question) and in the other hand reactive power is constant too so we won't have ΔV and ΔQ.
If you still don't get what I mean or why ΔQ=0 just email me.)





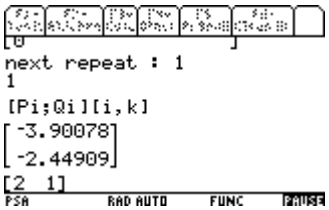
OK! First iteration finished.

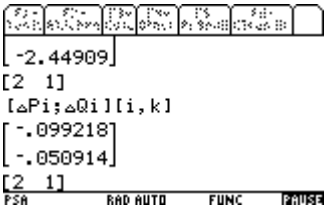
A little boring huh? Yes, Newton-Rophson method has long steps but only a few steps are needed to reach the final answer.

For example if you have solved this problem using GS method then you know reaching the answer with an error of less than 5*10^-5, need at least seven steps but here we will see that only after 3 steps we will reach the final answer.

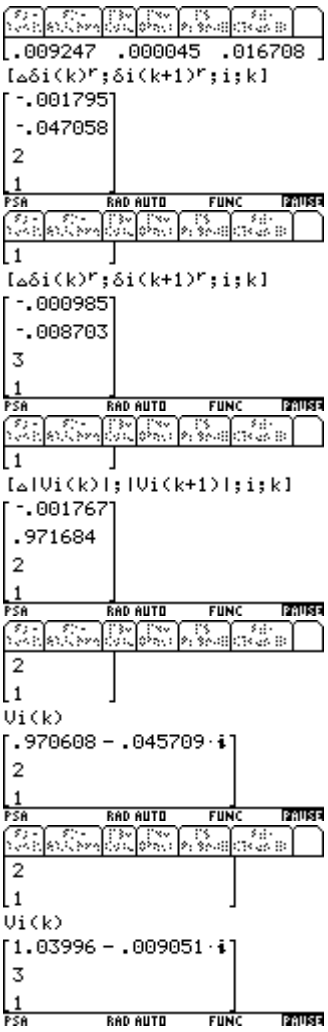
But there are other advantages that make NR much more reliable. For instance in a few system the iteration may not become converge when using GS method but the probability for such situation is very less in NR method. (Besides the method you use being convergent or not is very relevant to your first guess of voltages; as a standard in programs we assume that all PQ voltages are 1p.u. because in a real system and it's usual work (I mean except short circuit situations) it is really something near 1p.u. .)

If you want to calculate more Vi's with more repeats enter 1, any other number will lead you to next loop.





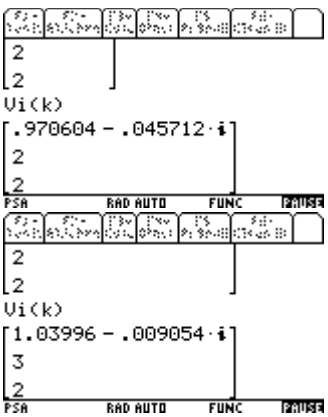
...



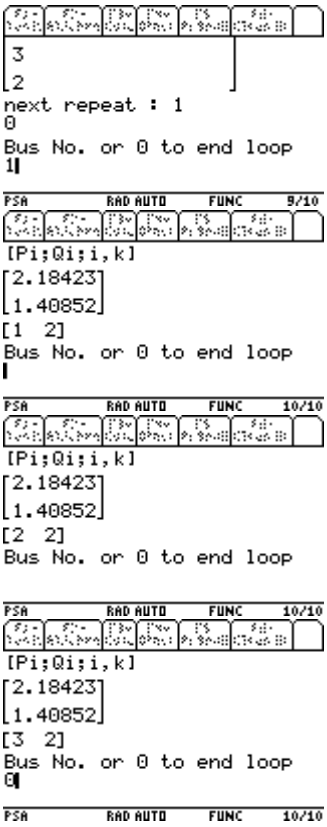
...

Only one more step! ☺

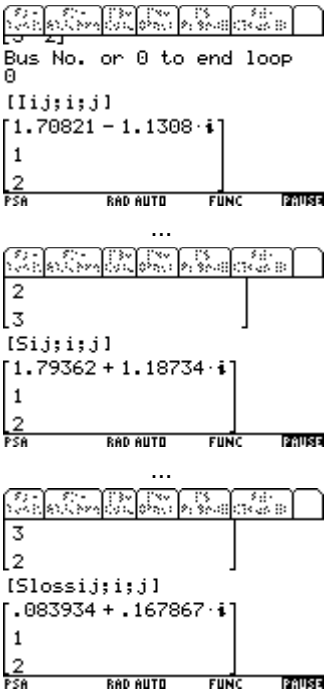
...



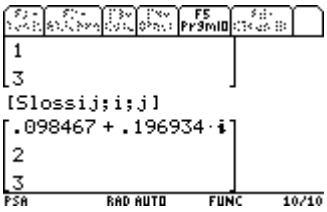
We wanted to know how much power should the slack bus produce. In some kind of problems the power produced by other PV buses is questioned too, so the program is designed in a way that can give you the power of any bus!



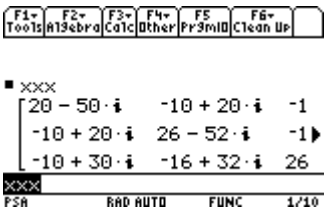
The rest of program just gives a few useful information about the system (Iij, Sij and Sloss,ij) in which Sloss,ij is the Power loss in line i to j.



...



After you finish you can see the Ybus by typing xxx:



That's all.
The only **limit** I found when using this program was that it's not programmed to calculate the effect of tab-changing of transformers. (Of coarse there is a big difference between a real system and what we just explained, but as a program to help you solve your problems I think it's very useful.)

Other related programs:
I have also written 3 other programs which use Newton-Rophson, Decoupled and Fast Decoupled methods to solve load flow problem. (The DC load flow is too simple and I'm not going to program it.)
Here is a list of programs I've written:

- Load flow using Gauss-Seidel method.
- Load flow using Decoupled method.
- Load flow using fast decoupled method.
- SCTM (Symmetrical components transformation matrix created by doctor C.L.Fortescue.) & SCTMI (Inverse of SCTM)
- puBasech((a function to calculate per unit values after change of base value.)
- linegc() (calculates line general coefficients for short and medium length lines.)
- llinegc() the same as linegc() but for long lines (uses hyperbolic functions.)
- GMD, GMR, GMRb (Geometric mean distance and radius of lines, these programs are slow I will be very thankful if you can send me any suggestions to improve these programs.)
- OCT() and SCT(). (Open circuit test and short circuit test for a DC machine.)
- IEEE (finds IEEE model of Induction machine.)
- W2D and D2W (delta<->why)
- …

Sorry for my bad English, if there is anything that I should add or any dictation or grammar errors just email me. Thanks. ;D

Any suggestions, any errors, anything, just contact me:
mailto:ali.db65@gmail.com

Ali Dehghan Banadaki
(A university student in Islamic Azad University of Gonabad.)
Thursday, February 07, 2008