Weekly Report(11/11/2002~15/11/2002)

Zhao Li

In this week, I've been implementing variable-step numerical integration methods for transient simulation.

First, I finished the variable-step method following what SPICE has done. The only difference is that I use the successive-chord method to accomplish nonlinear iteration rather than the Newton-Raphson method. The benefit is that only one LU decomposition is needed for each time step. The drawback is that more iterations are needed to achieve the same convergence tolerance.

Secondly, the semi-implicit integration method is plugged in to reduce the number of LU decompositions. The basic idea is that we still follow the SPICE variable-step integration method. But when the Trapezoid method is used, the new time step will be compared to the immediately previous time step. If they are close enough, the semi-implicit integration method will be used so that no LU decomposition is needed at the new time step. To ensure the semi-implicit method generates an accurate result, the LTE due to the semi-implicit integration method will be checked after each time step. If the LTE is not satisfied, we will use the general Trapezoid method to redo this time step. If the LTE is satisfied, we will used the general Trapezoid LTE formula to predict the next time step size. Tests on some simple circuits have shown that the above method does save LU decomposition times. For example,

RCL circuit, user-defined time step size 0.01sec, total simulation time 160sec, which means 16001 time points for a fixed-step integration method. SPICE-like integration method finished this by 337 simulation points with 409 LU decomposition times. Our program finished this by 336 simulation points with 252 LU decomposition times, and encountered 6 failed time points with semi-implicit integration. One extra cost for our program is the LTE checking using semi-implicit integration formula, but this part can be combined with the time-step prediction part. The other extra cost is the failed trials with semi-implicit method, so some good prediction methods should be found to know in advance whether the semi-implicit method should be used for the next time step or not.

In a summary, the key idea in this implementation is that – if possible, we prefer to use semi-implicit integration methods rather than general integration methods to reduce the LU decomposition times during the transient simulation, which should save much time for a stiff system. The top controlling scheme is still general BE-TRAP integration method (like SPICE).

Finally, I also tried to use the semi-implicit method as the top controlling scheme. In other words, the general trapezoid method will be replaced by the semi-implicit method totally, including the prediction part. However, I just found this is too hard. On one hand, the semi-implicit formula is too complex to be used as an efficient time-step prediction formula. On the other hand, it's impossible to use one fixed time step during the simulation – typically, we need to restart the LU decomposition every several time points. Then why not just use the general trapezoid method to predict the next time step while using the semi-implicit method to calculate the result? So, I think the previous implementation way should be better.

For the next week, I will focus on : 1) Optimize the implementation; 2) Find enough and appropriate test cases.