1 Interface between Reachability and Model Construction

The current interace between reachability code and model construction code is a function

```
models = model_create(circuit,lp,methods)
```

circuit variable is an object from mSpice, which represents the circuit to verify. *lp* is a Coho linear program which is the convex hull of a projectagon. *methods* specifies which linearization function to use. The function computes a valid model within the space specified as *lp* for each linearization method. *model* is a structure with fields *integ.L*, *integ.c* and *integ.errLP*

$$\dot{x} \quad = \quad Lx + c + errLP$$

Currently, $errLP$ is a hyperrectangle.

For each transistor, the function calls the linearization method

```
[c,err] = linfit(model,bbox,lp,nodes,method,isconv)
```

model is the 3D ids table from hspice simulation. *bbox* is the lower and upper bounds of terminal voltages. *lp* and *nodes* restrict the region by a linear program. *nodes* can be removed easily to simplify the interface. *method* specifies which linearization method to use. *isconv* specifies whether *model* is convex or not. It can be merged in to *model* variable later. *c* and *err* is for the model of prederivatives

$$\dot{x_i} = c[x_i; 1] \pm err$$

Therefore, a simple interface can be provided

```
[c,err] = linfit(model,bbox,lp,method)
```

For the projectagon geometric shape object, it already has a similar interface with $projected\_shape = shape.project(self, dim\_index)$. The interface $points = shape.createGrid(dx)$ and $shape = bounding_shape(points)$ is easy to implement if necessary. There might be two problems, the general APE model generator might be too expensive to be used in Coho, and projectagon is implemented as a structure rather than an object now (it should not be a problem to convert it to object).