Kate Rush

Predictor-Corrector Algorithm for Tracing Curves Modeling Transistor Behavior

The PredCor procedure makes tracing curves modeling transistor behavior more efficient by taking the largest possible x-value step for each increment of graphing while keeping error below the inputted threshold.

Arguments:

```
PredCor {in1} {max1} {inc} {err} {conName} {supName} {gName} {cName} {yCalc}
```

- in1: the initial value of the independent variable
- max1: the final value of the independent variable
- inc: the size of the first step from in1 to the next value of the independent variable
- err: the maximum allowed percent error
- conName: the contact name (format: contact name=________)
- supName: the supply name (format: _______ supply=)
- gName: the graph name (format: chart graph=_______)
- cName: the curve name (format: curve=_______)
- yCalc: the name of the procedure used to calculate the dependent variable

A Taylor Approximation is used in the following manner, as can be seen in the procedure.

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2$$

(value of the curve at x) = (value of the equation of the line at x) +(secder/2)(x increment)^2
|error| = (secder/2)(x increment)^2
2(|error|)/(x increment)^2 = secder
ESTIMATED NEXT STEP (to maximize step size without exceeding max allowed error = SQR ROOT[(2)(|max allowed error|)/(secder)]

UNALTERED PROCEDURE (fill in as needed):

```
#all necessary procedures listed

#prework/ramping  completed

#write  procedure  for calculating the dependent variable value  (yCalc)


proc PredCor  {in1 max1 inc err conName  supName gName  cName yCalc} {


 #in1 - first x value

 #max1 - final x value

 #inc - first step

 #err - max allowed error (percent)

 #conName  - contact name=___

 #supName - ___ supply=
```

```
#gName - chart graph=___
#cName - curve=___
#yCalc - procedure to calculate y

set max [expr {$max1 + .00001}]

set loop 0

set err [expr {$err / 100.0}]

contact name=$conName $supName supply=$in1
device
device store

set dep [$yCalc]
chart graph=$gName curve=$cName xval=$in1 yval=$dep

#set first point values as x1 and y1
set x1 $in1
set y1 $dep

#set first reference value to independent variable val
#set next indep var to current val + initial increase
set ref $in1
set in [expr {$in1 + $inc}]

#loop until max
while {$loop == 0} {

  set error 1
  device restore
  #in case of matrix issue
  while {$error == 1} {

    set error 0

    if {[catch {contact name=$conName $supName supply=$in
      device}]} {

      device restore
      set error 1

      #cut increase in half and add it to previous indep var val
      set inc [expr {$inc/2}]
      set in [expr {$ref + $inc}]
    }
  }
  device store
```

```
#calc dep value after above in value
set dep [$yCalc]

#store results for later graphing
set ref $in
set ina $in
set depa $dep

#set second point values as x2 and y2
set x2 $in
set y2 $dep

#get slope
set slope [expr {($y2 - $y1)/($x2 - $x1)}]
#get b
set b [expr {$y2-($slope*$x2)}]

#get next indepvar value
set in [expr {$x2 + $inc}]
if {$in > $max} {
  set in $max
}
#set projected y on line using eq just constructed
set p1 [expr {($slope*$in) + $b}]

#get actual next y value
set error 1
device restore
while {$error == 1} {

  set error 0

  if {[catch {contact name=$conName $supName supply=$in
    device} ] } {

    device restore
    set error 1

    #cut increase in half and add it to previous indep var val
    set inc [expr {$inc/2}]
    set in [expr {$ref + $inc}]

    #set projected y on line using eq just constructed
    #use updated indep variable value
    set p1 [expr {($slope*$in) + $b}]
  }
}
```

```
#calc dep value after in from above; don't graph
set dep [$yCalc]

#store results for later graphing
set inb $in
set depb $dep

#difference between actual point and projected point
set error [expr {$p1 - $dep}]

set maxerr [expr {$err*$dep}]

if {$error < 0} {
  set abserror [expr {$error*(-1)}]
}
if {$error > 0} {
  set abserror $error
}
if {$error == 0} {
  set abserror $error
}
if {$abserror < $maxerr} {
  set go 1
}
if {$abserror >= $maxerr} {
  set go 0
}

#error within allowance; can proceed
if {$go == 1} {
  device store

  #calculate ideal next step size
  set diffsq [expr {$inc*$inc}]
  set secder [expr {(2*$error)/$diffsq}]
  if {$secder < 0} {
    set secder [expr {$secder*(-1)}]
  }
  set inc [expr {sqrt((2*$maxerr)/$secder)}]
  set in1 $in
  set x1 $in1
  set y1 $dep

  #graph ina with depa
  chart graph=$gName curve=$cName xval=$ina yval=$depa
  #graph inb with depb
  chart graph=$gName curve=$cName xval=$inb yval=$depb
```

```
}

#error  too high; cannot proceed
if {$go == 0} {
  device restore
  #graph ina with depa
  chart graph=$gName curve=$cName xval=$ina yval=$depa

  #step back and try again
  set inc [expr {$inc*.5}]
  set x1 $x2
  set y1 $y2
}

#finished
if {$in1 >= $max1} {
  set loop 1
  puts "max reached"
}

if {$in1 < $max1} {
  set in [expr {$in1 + $inc}]
  set ref $in1

  #take it to the end
  if {$in >= $max} {
    set loop 1
    set in $max

    set error 1

    #calc until converges and max has been reached
    device restore
    while {($error == 1) || ($in < $max)} {
      set error 0
      if {[catch {contact name=$conName $supName supply=$in
      device} ] } {
        device restore
        set error 1
        set inc [expr {$inc/2}]
        set in [expr {$ref + $inc}]
      }
      set dep [$yCalc]

      #graph each success until the end
      if {$error == 0} {
        chart graph=$gName  curve=$cName xval=$in yval=$dep
        set in [expr {$in + $inc}]
```

```
          device store
        }
      }
      puts "max reached"
    }
  }
 }
}
```

EXAMPLE 1: biasVbe.tcl, graphing Gain vs. Base Voltage

Original File:

```
# Device Simulation
DevicePackage
# Source Structure All models need Acceptor and Donor
options cm
set Vce 1.0
struct infile= "structures/biased/Vce$Vce.str"
contact Silicon supply= 0.0  name= emit
contact Silicon supply= 0.0  name= base
contact Silicon supply= $Vce name= coll
contact Silicon supply= 0.0  name= sub

window row=2 col=2 width= 500 height= 500
plot2d grid graph= grid
plot2d contact=coll graph= grid
plot2d contact=base graph= grid
plot2d contact=emit graph= grid
#plot2d contact=sub graph= grid
pen name=pD width=2 black
pen name=pE width=2 blue
pen name=pH width=2 red
pdbSetDouble Math iterLimit 200
##########################
# Equations
#########################
source [findFile equations.tcl]
##########################
# Solution
#########################
#pdbSet Math rhsLimit 1.0
pdbSetDouble Math rhsMax 1.0e-3
pdbSetDouble Math updateMax 1.0e-1
device

#do a DC Gummel Plot Sweep
```

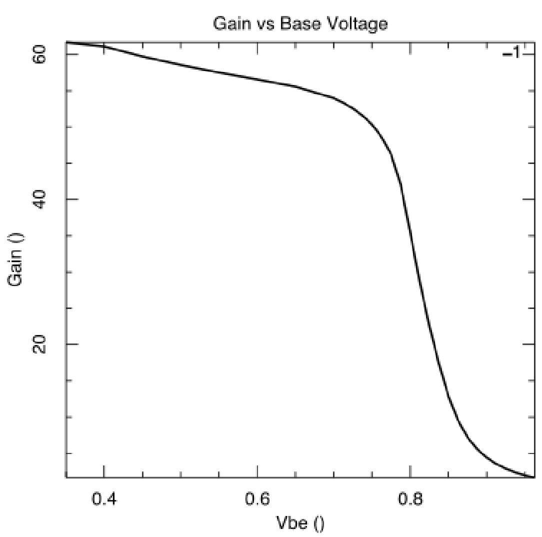
```
#     starting value    #  max


#*******CHANGES BEGIN******


for { set Vbe 0.050 } { $Vbe < 1.5025 } {set Vbe [expr ($Vbe < 0.95 && $Vbe > 0.690) ? $Vbe + 0.0125 : $Vbe + 0.05 ]} {
    contact name=base supply= $Vbe
    device
    set base [expr log10(abs([contact name=base sol=Qfn flux] - [contact name=base sol=Qfp flux]))]
    set coll [expr log10(abs([contact name=coll sol=Qfn flux] - [contact name=coll sol=Qfp flux]))]
    set gain [expr  abs([contact name=coll sol=Qfn flux] - [contact name=coll sol=Qfp flux]) / abs([contact name=base sol=Qfn flux] - [contact name=base sol=Qfp flux]) ]
    set gain [expr  $gain ]
  if {$Vbe > 0.31 } {
      #chart curve= Ib graph= Gummel xval= $Vbe yval= $base ylab= "Current ()" xlab= "Voltage ()" title= "Gummel"
      #chart curve= Ic graph= Gummel xval= $Vbe yval= $coll
      #chart curve= 1 graph=gain xval= $base yval= $gain ylab= "Gain ()" xlab= "Ib ()" title= "Gain vs Base Current"
      chart curve= 1 graph=gain xval= $Vbe yval= $gain ylab= "Gain ()" xlab= "Vbe ()" title= "Gain vs Base Voltage"
  }
  #struct outfile= "structures/biased/Vce$Vce\Vbe[expr round(($Vbe)*100.)/100.].str"
  puts "Vce $Vce"
  puts "Vbe $Vbe"
  puts "Ib [expr abs([contact name=base sol=Qfn flux] - [contact name=base sol=Qfp flux]) ]"
  puts "Ic [expr abs([contact name=coll sol=Qfn flux] - [contact name=coll sol=Qfp flux]) ]"
}
puts "Vce $Vce"
puts "Vbe $Vbe"
puts "Ib Qfn Flux [expr ([contact name=base sol=Qfn flux] ) ]"
puts "Ic Qfn Flux [expr ([contact name=coll sol=Qfn flux] ) ]"
puts "Ib Qfp Flux [expr ([contact name=base sol=Qfp flux] ) ]"
puts "Ic Qfp Flux [expr ([contact name=coll sol=Qfp flux] ) ]"


puts "Ib [expr abs([contact name=base sol=Qfn flux] - [contact name=base sol=Qfp flux]) ]"
puts "Ic [expr abs([contact name=coll sol=Qfn flux] - [contact name=coll sol=Qfp flux]) ]"
gets stdin
set fil [open "csv/bias/Gummel.csv" "w"]
puts $fil [chart graph= Gummel dump ]
close $fil
set fil [open "csv/bias/Gain.csv" "w"]
puts $fil [chart graph=gain    dump ]
close $fil
gets stdin
```

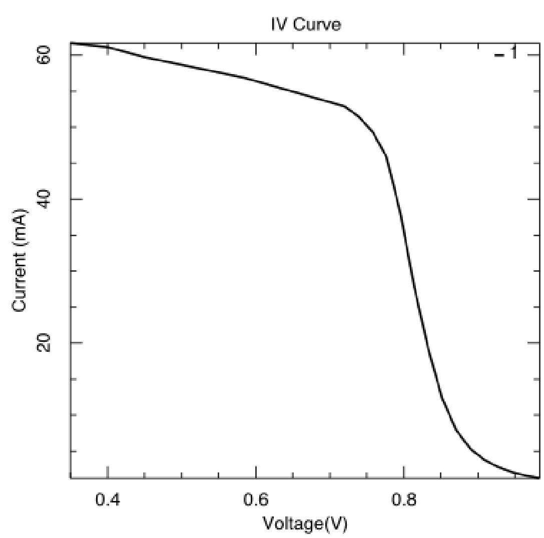
Using PredCor Procedure:

```
#all remains same until
#*******CHANGES BEGIN******
device store
set Vbe1 0.05
set inc 0.05
#ramp to starting graphable value
```

```
while {$Vbe1 <= .31} {
 device restore
 puts "Ramping Vbe to $Vbe1"
 contact name=base supply= $Vbe1
 device

 set base [expr log10(abs([contact name=base sol=Qfn flux] - [contact name=base sol=Qfp flux]))]
 set coll [expr log10(abs([contact name=coll sol=Qfn flux] - [contact name=coll sol=Qfp flux]))]
 set gain [expr abs([contact name=coll sol=Qfn flux] - [contact name=coll sol=Qfp flux]) / abs([contact name=base sol=Qfn flux] - [contact name=base sol=Qfp flux]) ]
 set gain [expr $gain]

 set Vbe1 [expr $Vbe1 + $inc]
 device store
}
#procedure to calculate Gain
proc gainCalc {} {
 set base [expr log10(abs([contact name=base sol=Qfn flux] - [contact name=base sol=Qfp flux]))]
 set coll [expr log10(abs([contact name=coll sol=Qfn flux] - [contact name=coll sol=Qfp flux]))]
 set gain [expr abs([contact name=coll sol=Qfn flux] - [contact name=coll sol=Qfp flux]) / abs([contact name=base sol=Qfn flux] - [contact name=base sol=Qfp flux]) ]
 set gain [expr $gain]
 return $gain
}
proc PredCor {in1 max1 inc err conName supName gName cName yCalc} {
#PROCEDURE CONTENT
}
#******UTILIZE PREDCOR PROCEDURE******
PredCor {.35} {1} {.05} {10} {base} {} {gain} { 1} {gainCalc}
```

Graph Comparison (Left is Original, Right is PredCor Produced): *titles are not currently changeable*





EXAMPLE 2: mosdrain.tcl, graphing Current vs. Gate Voltage

Original File:

```
#convergence issue. run fdsoi first!

#math device package
math diffuse dim=1 umf none col !scale
pdbSetDouble Math iterLimit 50
DevicePackage
math device dim=1 row bcgs ilu tol=1e-30

#constants
set T    300
set k    1.38066e-23
set q    1.619e-19
set Vt   [expr {$k*$T/$q}]
set ni   1.1e10
set esi  [expr 11.8 * 8.85418e-14]
set eox  [expr 3.9 * 8.85418e-14]
set eps  [expr $esi / $q]
set epo  [expr $eox /  $q]
set Emob 350.0
set Hmob 150.0
set small 1.0
set Nc 3.2e+19
set Nv 1.8e+19

#add solutions
solution add name=DevPsi pde solve negative damp continuous
solution add Silicon name=Elec   pde solve !negative
solution add Silicon name=Hole   pde solve !negative

solution add Oxide   name=Elec const solve val=($small)
solution add Oxide   name=Hole const solve val=($small)

#set equation
#set equations
set eqnP "$eps * grad(DevPsi) + Doping - Elec + Hole"
set eqnE "ddt(Elec) + 400.0  * 0.025 * sgrad(Elec, DevPsi/0.025)"
set eqnH "ddt(Hole) + 200.0 * 0.025 * sgrad(Hole,  -DevPsi/0.025)"
pdbSetDouble Silicon DevPsi DampValue 0.025
pdbSetDouble Silicon DevPsi Abs.Error 1.0e-10
pdbSetString Silicon DevPsi Equation $eqnP
pdbSetDouble Silicon Elec Abs.Error 1.0e5
pdbSetString Silicon Elec Equation $eqnE
pdbSetDouble Silicon Hole Abs.Error 1.0e5
pdbSetString Silicon Hole Equation $eqnH


pdbSetDouble Silicon Eg 1.1
```

```
pdbSetDouble Silicon Affinity 4.05

set eqnP "$epo * grad(DevPsi) - (1.0e-10)"
#set eqnP "$epo * grad(DevPsi)"
pdbSetDouble Oxide DevPsi DampValue $Vt
pdbSetDouble Oxide DevPsi Abs.Error 1.0e-9
pdbSetString Oxide DevPsi Equation $eqnP

#set contact equations
pdbSetBoolean drain Elec   Fixed 1
pdbSetBoolean drain Hole   Fixed 1
pdbSetBoolean drain DevPsi Fixed 1
pdbSetString drain Elec Equation {Doping - Elec + Hole}
pdbSetString drain Hole Equation {DevPsi + 0.025*log((Hole+1.0e-10)/1.0e10)   - drain}
pdbSetString drain DevPsi Equation {DevPsi - 0.025*log((Elec+1.0e-10)/1.0e10)  - drain}
pdbSetDouble  drain Elec   Flux.Scale 1.619e-19
pdbSetDouble  drain Hole   Flux.Scale 1.619e-19

pdbSetBoolean src Elec   Fixed 1
pdbSetBoolean src Hole   Fixed 1
pdbSetBoolean src DevPsi Fixed 1
pdbSetString src  Elec Equation {Doping - Elec + Hole}
pdbSetString src Hole Equation {DevPsi + 0.025*log((Hole+1.0e-10)/1.0e10)   - src}
pdbSetString src DevPsi Equation {DevPsi - 0.025*log((Elec+1.0e-10)/1.0e10)  - src}
pdbSetDouble  src Elec   Flux.Scale 1.619e-19
pdbSetDouble  src Hole   Flux.Scale 1.619e-19

pdbSetBoolean sub Elec   Fixed 1
pdbSetBoolean sub Hole   Fixed 1
pdbSetBoolean sub DevPsi Fixed 1
pdbSetString sub Hole Equation {Doping  - Elec + Hole}
pdbSetString sub DevPsi Equation {DevPsi + 0.025*log((Hole+1.0e-10)/1.0e10)  - sub}
pdbSetString sub Elec Equation {DevPsi - 0.025*log((Elec+1.0e-10)/1.0e10)  - sub}
pdbSetDouble  sub Elec   Flux.Scale 1.619e-19
pdbSetDouble  sub Hole   Flux.Scale 1.619e-19


pdbSetDouble Aluminum 0.0
proc MetalContact {Contact Mat} {
    pdbSetBoolean $Contact DevPsi Flux  1
    pdbSetBoolean $Contact DevPsi Fixed 1
    pdbSetString  $Contact DevPsi Equation "DevPsi - $Contact"; # Efm+WFN=Ei=-DevPsi, Efm=-$Contact
}
MetalContact front Oxide

    #set grid
    line x loc=-0.05 spac=0.01   tag=TopOx
    line x loc=0.03 spac=0.001  tag=TopSi
```

```
   line x loc=0.04 spac=0.0025
   line x loc=0.06 spac=0.01
   line x loc=0.09 spac=0.005
   line x loc=0.10 spac=0.01
   line x loc=0.12 spac=0.025
   line x loc=0.19 spac=0.05
   line x loc=3.00 spac=0.1    tag=Bottom

   # Vertical lines
   line y loc=0.00 spac=0.05   tag=Left
   line y loc=0.10 spac=0.01
   line y loc=0.15 spac=0.01
   line y loc=0.25 spac=0.007  tag=GLE
   line y loc=0.35 spac=0.05
   line y loc=1.25 spac=0.1
   line y loc=2.15 spac=0.05
   line y loc=2.25 spac=0.007  tag=GRE
   line y loc=2.35 spac=0.01
   line y loc=2.40 spac=0.01
   line y loc=2.50 spac=0.05   tag=Right

   #add  material and region
   mater add name=Oxide
   mater add name=Silicon

   region  Oxide   xlo=TopOx  xhi=TopSi  ylo=GLE yhi=GRE
   region  Silicon xlo=TopSi xhi=Bottom ylo=Left yhi=Right
   init

   #declare contact
   contact name=front      oxide   xlo=-0.055  xhi=-0.045 ylo=0.25 yhi=2.25 add depth=1.0  width=1.0
   contact name=src       silicon xlo=0.02905 xhi=0.030005 ylo=0.00 yhi=0.10 add depth=1.0 width=1.0
   contact name=drain     silicon xlo=0.02905 xhi=0.030005 ylo=2.45 yhi=2.50 add depth=1.0 width=1.0
   contact name=sub silicon xlo=2.950 xhi=3.01 ylo=0.00 yhi=2.50 add depth=1.0 width=1.0

# Define initial bias
contact name=front      voltage supply=0.0
contact name=src       voltage supply=0.0
contact name=drain     voltage supply=0.0
contact name=sub voltage supply=0.0

#set doping
set buff 1.0e10
sel z=((1.0e20)*(x<0.10)*((y<0.25)+(y>2.25)) )    name=ND
sel z=(1.0e15*(x>-0.01)+$buff)  name=NA
sel z=ND-NA  name=Doping

#initial guess
```

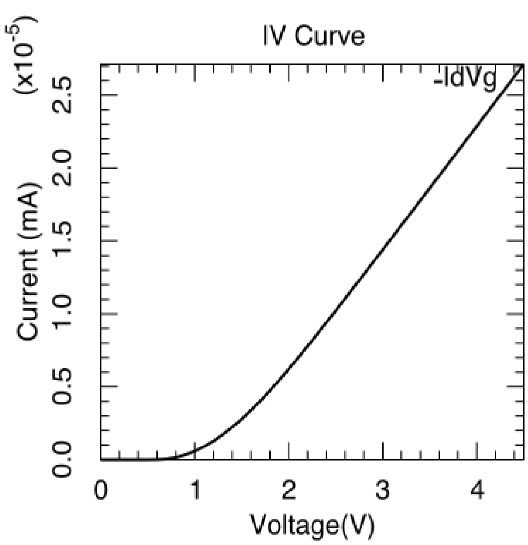
```
sel z=0.5*(Doping+sqrt(Doping*Doping+4.0e20))/1.0e10   name=arg

sel z=0.025*log(abs(arg))  name=DevPsi

sel z=1.0e10*exp(DevPsi/0.025)  name=Elec

sel z=1.0e10*exp(-DevPsi/0.025) name=Hole


#0 bias

device


#*******CHANGES BEGIN******


set v 0.1

#Ramp drain bias

if {1} {

   set delta 0.5

   for {set bias 0.0} {$bias <= 1.3} {set bias [expr {$bias + $delta}]} {

       contact name=drain voltage supply=$bias

       puts "Ramping drain to $bias"

       device

   }

}


window row=1 columns=3 width=400 height=400

#Ramp gate bias, extract Gm, AC bias

if {1} {

       set delta 0.1

       for {set bias 0.0} {$bias <= 4.5} {set bias [expr {$bias + $delta}]} {

           contact name=front  voltage supply=$bias

           puts "Ramping gate to $bias"

           device

     device


     sel z=log10(Elec)

     plot1d yv=1.25 graph=Elec


           set cur [expr ([contact name=drain sol=Hole flux] - [contact name=drain sol=Elec flux])]

           chart graph=Win_IdVg curve=IdVg xval=$bias yval=$cur

  }

}
```

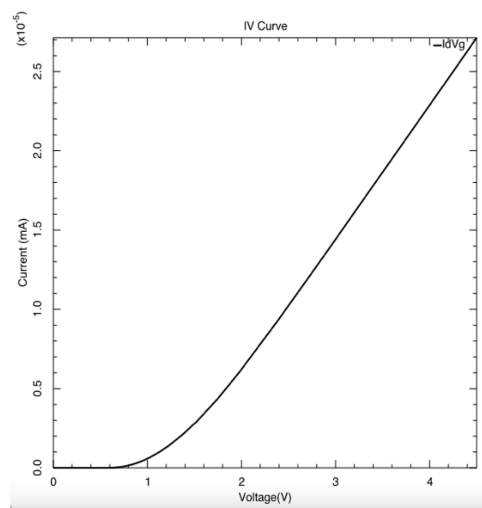
Using PredCor Procedure:

```
#all remains same until

#*******CHANGES BEGIN******

set v 0.1

#ramp drain

if {1} {

   set delta 0.5
```

```
  for {set bias 0.0} {$bias <= 1.3} {set bias [expr {$bias + $delta}]} {
    contact name=drain voltage supply=$bias
    puts "Ramping drain to $bias"
    device
  }
}

#calculate current
proc currCalc {} {
 expr ([contact name=drain sol=Hole flux] - [contact name=drain sol=Elec flux])
}

proc PredCor {in1 max1 inc err conName supName gName cName yCalc} {
#PROCEDURE CONTENT
}
#******UTILIZE PREDCOR PROCEDURE******
PredCor {0.0} {4.5} {.1} {10} {front} {voltage} {Win_IdVg} {IdVg} {currCalc}
```

Graph Comparison (Left is Original, Right is PredCor Produced):





EXAMPLE 3: GateCap.tcl, graphing Drain Current vs. Gate Voltage

Original File:

```
#math device package
math diffuse dim=1 umf none col !scale
pdbSetDouble Math iterLimit 50
DevicePackage

#Two versions - one with SG and one with QF
```

```
#They can also be run with both the quad and tri meshes

set T    300
set k    1.38066e-23
set q    1.619e-19
set Vt   [expr {$k*$T/$q}]
set esi  [expr 11.8 * 8.85418e-14]
set eox  [expr 3.9 * 8.85418e-14]
set eps  [expr $esi / $q]
set epo  [expr $eox / $q]
set Emob 350.0
set Hmob 150.0
set small 1.0e-10
set Nc 3.2e+19
set Nv 1.8e+19
set Eg 1.1
set ni   [expr sqrt($Nc*$Nv) * exp(-0.5 * $Eg / $Vt)]

proc OhmicSG {name} {
    global ni Vt
    #set contact equations
    pdbSetBoolean $name Elec   Fixed 1
    pdbSetBoolean $name Hole   Fixed 1
    pdbSetBoolean $name DevPsi Fixed 1
    pdbSetString $name Elec Equation "Doping - Elec + Hole"
    pdbSetString $name DevPsi Equation "DevPsi - $Vt * log(Elec/$ni) - $name"
    pdbSetString $name Hole Equation "DevPsi + $Vt * log(Hole/$ni) - $name"
    pdbSetDouble  $name Elec   Flux.Scale 1.619e-19
    pdbSetDouble  $name Hole   Flux.Scale 1.619e-19
}
proc SetSGEqn {} {
    global eps epo Vt ni Emob Hmob
    #add solutions
    solution add name=DevPsi pde solve negative damp continuous
    solution add Silicon name=Elec   pde solve !negative
    solution add Silicon name=Hole   pde solve !negative

    set e "DevPsi - ($Vt) * log( Elec / $ni ) "
    solution add name=Qfn solve Silicon const val = "($e)"

    set e "DevPsi + ($Vt) * log( Hole / $ni ) "
    solution add name=Qfp solve Silicon const val = "($e)"

    #set equations
    set eqnP "$eps * grad(DevPsi) + Doping - Elec + Hole"
    set eqnE "ddt(Elec) + 400.0 * 0.025 * sgrad(Elec, DevPsi/0.025)"
    set eqnH "ddt(Hole) + 200.0 * 0.025 * sgrad(Hole, -DevPsi/0.025)"
    pdbSetDouble Silicon DevPsi DampValue 0.025
```

```
    pdbSetDouble Silicon DevPsi Abs.Error 1.0e-10
    pdbSetString Silicon DevPsi Equation $eqnP
    pdbSetDouble Silicon Elec Abs.Error 1.0e5
    pdbSetString Silicon Elec Equation $eqnE
    pdbSetDouble Silicon Hole Abs.Error 1.0e5
    pdbSetString Silicon Hole Equation $eqnH

    set eqnP "$epo * grad(DevPsi)"
    pdbSetDouble Oxide DevPsi DampValue $Vt
    pdbSetDouble Oxide DevPsi Abs.Error 1.0e-9
    pdbSetString Oxide DevPsi Equation $eqnP

    ##############################################
    OhmicSG drain
    OhmicSG src
    OhmicSG substrate
    OhmicSG gate
}

proc OhmicQF {name} {
    global ni Vt
    #set contact equations
    pdbSetBoolean $name Qfn   Fixed 1
    pdbSetBoolean $name Qfp   Fixed 1
    pdbSetBoolean $name DevPsi Fixed 1
    pdbSetString $name DevPsi Equation "Doping - $ni * exp((DevPsi-Qfn)/$Vt)  + $ni * exp((Qfp-DevPsi)/$Vt)"
    pdbSetString $name Qfn Equation "Qfn - $name"
    pdbSetString $name Qfp Equation "Qfp - $name"
    pdbSetDouble  $name Qfn   Flux.Scale 1.619e-19
    pdbSetDouble  $name Qfp   Flux.Scale 1.619e-19
}
proc SetQFEqn {} {
    global eps epo Vt ni Emob Hmob
    #add solutions
    solution add name=DevPsi pde solve negative damp continuous
    solution add Silicon name=Qfn   pde solve
    solution add Silicon name=Qfp   pde solve

    set e "($ni * exp( (DevPsi-Qfn)/$Vt  ) + 1.0)"
    solution add name=Elec solve Silicon const val = "($e)"

    set e "($ni * exp( (Qfp-DevPsi)/$Vt  ) + 1.0)"
    solution add name=Hole solve Silicon const val = "($e)"

    #set equations
    set eqnP "$eps * grad(DevPsi) + Doping - Elec + Hole"
    set eqnE "ddt(Elec) + 400.0 * Elec * grad(Qfn)"
    set eqnH "ddt(Hole) - 200.0 * Hole * grad(Qfp)"
```

```
    pdbSetDouble Silicon DevPsi DampValue 0.025
    pdbSetDouble Silicon DevPsi Abs.Error 1.0e-10
    pdbSetString Silicon DevPsi Equation $eqnP
    pdbSetDouble Silicon Qfn Abs.Error 1.0e-10
    pdbSetString Silicon Qfn Equation $eqnE
    pdbSetDouble Silicon Qfp Abs.Error 1.0e-10
    pdbSetString Silicon Qfp Equation $eqnH

    set eqnP "$epo * grad(DevPsi)"
    pdbSetDouble Oxide DevPsi DampValue $Vt
    pdbSetDouble Oxide DevPsi Abs.Error 1.0e-9
    pdbSetString Oxide DevPsi Equation $eqnP

    ############################################
    OhmicQF drain
    OhmicQF src
    OhmicQF substrate
    OhmicQF gate
}
proc MosGrid {flag} {
    #set grid
    line x loc=-0.03 spac=0.005 tag=P
    line x loc=0.0 spac=0.001 tag=F
    line x loc=0.01 spac=0.001 tag=I1
    line x loc=0.04 spac=0.001 tag=I2
    line x loc=0.05 spac=0.005
    line x loc=1.0 spac=0.1 tag=B

    line y loc=0.0 spac=0.01 tag=Left
    line y loc=0.1 spac=0.01 tag=GL
    line y loc=0.4 spac=0.01 tag=GR
    line y loc=0.5 spac=0.01 tag=Right

    #add material and region
    mater add name=Oxide
    mater add name=Silicon

    region Silicon   xlo=P xhi=F ylo=GL yhi=GR
    region Oxide   xlo=F xhi=I1 ylo=Left yhi=Right
    region Silicon   xlo=I1 xhi=B ylo=Left yhi=Right

    if {$flag} {
  init quad
    } else {
  init
    }
    contact name=gate Silicon xlo=-0.035 xhi=-0.025 ylo=0.1 yhi=0.4 add supply=0.0
    contact name=substrate Silicon xlo=0.95 xhi=1.05 ylo=-0.05 yhi=0.55 add supply=0.0
```

```
    contact name=src Silicon xlo=0.01 xhi=0.03 ylo=-0.05 yhi=0.05 add supply=0.0
    contact name=drain Silicon xlo=0.01 xhi=0.03 ylo=0.45 yhi=0.55 add supply=0.0
    sel z=((1.0e19)*((y<0.1)+(y>0.4))*(x<0.04))+(1.0e20)*(x<0.005)    name=ND
    sel z=1.0e17 name=NA
    sel z=ND-NA  name=Doping
}
proc InitSG {} {
    # Define initial bias
    contact name=gate      voltage supply=0.0
    contact name=src       voltage supply=0.0
    contact name=drain     voltage supply=0.0
    contact name=substrate voltage supply=0.0

    #initial guess
    sel z=0.5*(Doping+sqrt(Doping*Doping+4.0e20))/1.0e10   name=arg
    sel z=0.025*log(abs(arg))  name=DevPsi
    sel z=1.0e10*exp(DevPsi/0.025)  name=Elec
    sel z=1.0e10*exp(-DevPsi/0.025) name=Hole
    #0 bias
    device init
    device
}
proc InitQF {} {
    # Define initial bias
    contact name=gate      voltage supply=0.0
    contact name=src       voltage supply=0.0
    contact name=drain     voltage supply=0.0
    contact name=substrate voltage supply=0.0

    #initial guess
    sel z=0.0 name=Qfn
    sel z=0.0 name=Qfp
    sel z=0.0 name=DevPsi
    newton Silicon eqn=Doping-Elec+Hole var=DevPsi damp=0.025

    #0 bias
    device init
    device
}
#*******CHANGES BEGIN******
proc RampPlot {Name Sol} {
    #Ramp drain bias
    set delta 0.05
    for {set bias 0.0} {$bias <= 0.5} {set bias [expr {$bias + $delta}]} {
        contact name=drain voltage supply=$bias
        puts "Ramping drain to $bias"
        device
    }
```

```
    #Ramp gate bias, extract Gm, AC bias
    set delta 0.1
    for {set bias 0.0} {$bias <= 4.5} {set bias [expr {$bias + $delta}]} {
  contact name=gate voltage supply=$bias
  puts "Ramping gate to $bias"
  device

  set cur [expr (-[contact name=drain sol=$Sol flux])]
  chart graph=Win_IdVg curve=IdVg$Name xval=$bias yval=$cur leg.left

  set v 1.0
  contact name=gate voltage supply= $v acreal
  contact name=gate voltage supply=0.0 acimag
  set f 1.0e5
  device freq=$f
  set re_curg [expr (abs([contact name=gate sol=Qfn flux acreal])/($v*2*3.14159*$f))]
  set im_curg [expr (abs([contact name=gate sol=Qfn flux acimag])/($v*2*3.14159*$f))]
  chart graph=WinC curve=g#dg#u xval=$bias yval=$re_curg xlab=V#dg#u ylab=Capacitance/um#u2#d leg.right leg.bottom title= "C#dg#u v. V#dg#u"
  chart graph=WinC curve=C#dg#u xval=$bias yval=$im_curg
    }
}
window row=1 columns=2 width=400 height=400
#SetSGEqn
#MosGrid 0
#InitSG
#set SGTime [time {
    #RampPlot TriSG Elec
#}]
SetQFEqn
MosGrid 0
InitQF
set QFTime [time {
    RampPlot TriQF Qfn
}]
```

Using PredCor Procedure:

```
#all remains same until
#*******CHANGES BEGIN******
#ramp bias
set delta 0.05
for {set bias 0.0} {$bias <= 0.5} {set bias [expr {$bias + $delta}]} {
  contact name=drain voltage supply=$bias
  device
  device store
}

#procedure to calculate current
proc IdVgp {} {
```

```
  expr (-[contact name=drain sol=Qfn flux])
}
proc PredCor {in1 max1 inc err conName supName gName cName yCalc} {
#PROCEDURE CONTENT
}
#******UTILIZE PREDCOR PROCEDURE******
PredCor {0.0} {4.5} {.1} {10} {gate} {voltage} {Win_IdVg} {IdVgTriQF} {IdVgp}
```

Graph Comparison (Left is Original, Right is PredCor Produced):