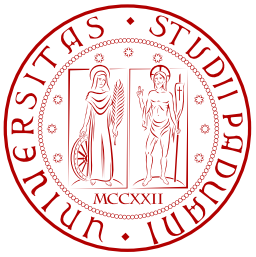

Università degli studi di Padova

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA ELETTRONICA

Microelettronica

Laboratorio di Microelettronica
Homework, 7/6/2012

Studente: Marco Piovesan

Matricola: 1012992

Prof. Andrea Cester

Prof. Gaudenzio Meneghesso

Anno Accademico 2011/2012

Indice

Capitolo 1

Introduzione

Lo scopo di questo lavoro è la simulazione del comportamento di una giunzione pn in condizioni di non equilibrio servendosi del linguaggio MATLAB per risolvere l'equazione di Poisson.

1.1 Calcolo dei valori teorici

Si considera una giunzione p-n con i seguenti drogaggi:

- Drogaggio p: $N_A = 5 \cdot 10^{16} cm^{-3}$;
- Drogaggio n: $N_D = 2 \cdot 10^{17} cm^{-3}$;

Si calcolano preliminarmente i **valori teorici** di larghezza della regione di carica spaziale RCS e di campo elettrico massimo con le formule classiche. Si parte innanzitutto dal potenziale di built-in ϕ:

$$\phi_i = \frac{kT}{q} \ln(\frac{N_A \cdot N_D}{{n_i}^2}) = 0.8144V. \tag{1.1}$$

Si passa poi a calcolare la larghezza della regione di carica spaziale RCS:

$$x_D = \sqrt{\frac{2\epsilon_{Si}}{q} \cdot (\frac{1}{N_A} + \frac{1}{N_D}) \cdot \phi_i} = 1.6227 \cdot 10^{-5} cm^{-2} \Rightarrow 0.16227 \mu m \tag{1.2}$$

e infine si passa al valore di campo elettrico:

$$E_{MAX} = \frac{2 \cdot \phi_i}{x_D} = -1.0038 \cdot 10^5 V cm^{-1}. \tag{1.3}$$

1.2 Ipotesi usate nelle simulazioni

La lunghezza scelta per la giunzione p-n nelle simulazioni è di $L = 2\mu m = 2 \cdot 10^{-4} cm$ per avere una buona risoluzione dell'area d'interesse ovvero la regione di carica spaziale RCS. Il procedimento logico si basa innanzitutto sulla *discretizzazione* delle equazioni differenziali e sulla loro risoluzione per via numerica attraverso il metodo degli elementi finiti con iterazioni fino alla convergenza.
Il dominio scelto è di 10^4 punti con una tolleranza di 10^{-9} nel quale si sono imposte le condizione di Dirichlet come *condizioni al contorno* per le regioni quasi neutre.
C'è poi la *normalizzazione* di tutte le lunghezze rispetto alla *lunghezza di Debye*:

$$L_{D0} = \sqrt{\frac{V_t e_r e_0}{q n_i}} = 3 \cdot 10^{-4} cm, \tag{1.4}$$

le concentrazioni vengono normalizzate alla *concentrazione intrinseca*:

$$n_i = 1.45 \cdot 10^{10} cm^{-3} \tag{1.5}$$

e i potenziali vengono rapportati alla *tensione termica*:

$$V_t = 25.9 mV. \tag{1.6}$$

Si passa poi a discretizzare il dominio rappresentante la lunghezza del semiconduttore in 10^4 intervalli di ampiezza $\Delta x = \frac{L}{10^4} \ll L_{D0}$. L'algoritmo usato per risolvere l'equazione di Poisson in condizioni di non equilibrio è di tipo perturbativo, nello specifico si usa l'algoritmo di Gummel che consente di risolvere separatamente le equazioni differenziali di continuità per il potenziale, per la densità di corrente di lacune e per la densità di corrente di elettroni trascurando il contributo della generazione e della ricombinazione nella RCS. Si parte dalla soluzione trovata precedentemente in condizioni di equilibrio che ha la caratteristica di avere un unico livello di Fermi e la si perturba con una piccola variazione di tensione di valore inferiore alla tensione termica $\Delta V_s < V_t$. Dopo l'applicazione della perturbazione il sistema si porta in un nuovo stato in cui sono presenti due livelli di Fermi al posto di quello unico: uno per gli elettroni E_{fn} e uno per le lacune E_{fp}.
Per assicurare la convergenza dell'algoritmo devono essere imposte delle opportune condizioni iniziali e finali oltre al valore piccolo di ΔV_s scelto pari a $\Delta V_s = 10 mV$. A ogni ciclo si ha un valore aggiornato del potenziale e il procedimento si ripete fino ad arrivare alla convergenza.
Il campo elettrico si trova facilmente dalla relazione:

$$\phi(2) - \phi(1) = -\int E(x) dx \tag{1.7}$$

e lo stesso vale per la distribuzione di densità di carica elettrica:

$$E(2) - E(1) = \int \frac{\rho(x)}{\epsilon_{Si}} dx. \tag{1.8}$$

Attraverso una serie di operazioni di derivazione è possibile risalire ai grafici della densità di carica e di campo elettrico.

Per disegnare la caratteristica $J_D vs. V_D$ si è ripetuto l'algoritmo perturbativo due volte: una partendo dalla condizione di equilibrio e andando verso tensioni positive e l'altra andando verso tensioni negative. All'interno dell'algoritmo si calcolano separatamente i contributi di densità di corrente di diffusione e di deriva per poi prenderne la somma J_D che risulta costante in condizioni di stazionarietà. Memorizzando per ogni iterazione n dell'algoritmo il valore di $J_D(n)$ relativo a un incremento $V_D(n) = 0 + n \cdot \Delta V_D$ si può graficare al termine la caratteristica $J_D vs. V_D$.

Capitolo 2

Simulazioni di una giunzione p-n

2.1 Tracciare il diagramma a bande all'equilibrio

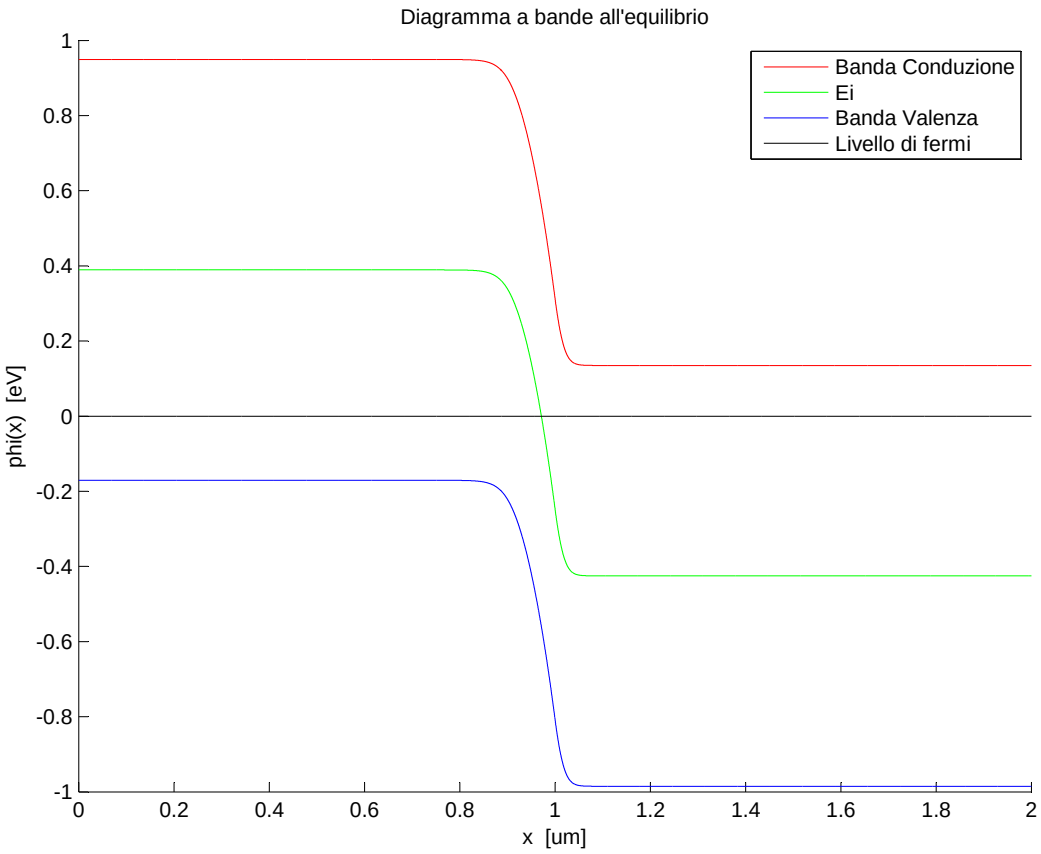

Figura 2.1: Diagramma a bande all'equilibrio

Una volta ricavato con l'algoritmo la situazione di potenziale all'equilibrio si può graficare il diagramma a bande della giunzione sapendo che l'andamento della banda di conduzione

e valenza hanno l'andamento del potenziale cambiato di segno e traslato di $\pm\frac{E_i}{2}$.

2.2 Tracciare l'andamento della carica elettrica e confrontare l'ampiezza con quella teorica

Di seguito si vede l'andamento della densità di carica elettrica nella regione di carica spaziale:

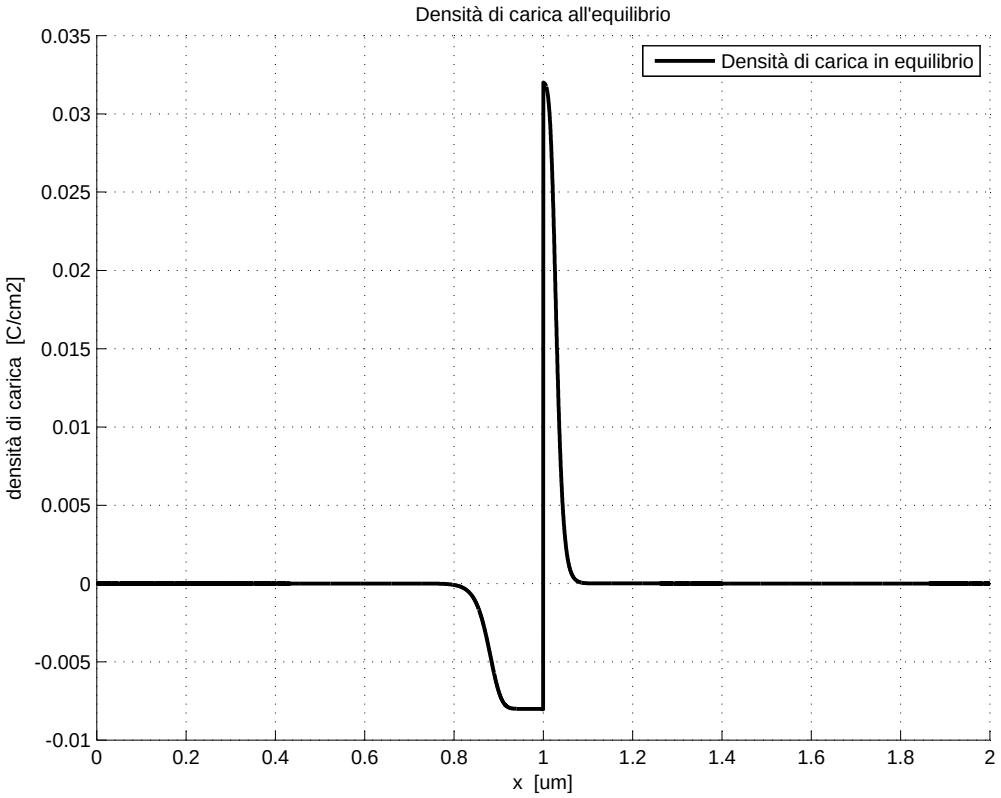

Figura 2.2: Andamento della densità di carica elettrica all'equilibrio

Come era prevedibile si nota che la RCS si estende di meno nella parte di silicio più drogata ovvero nell'n-Si (sezione a destra in figura). Inoltre, si vede che l'andamento in simulazione assomiglia poco all'ipotesi che si fa solitamente di svuotamento a cassetto. I valori di densità di carica nella simulazione corrispondono nel loro massimo a quelli che normalmente si utilizzerebbero nell'ipotesi di svuotamento a cassetto ovvero

$$Q_p = q \cdot N_A = -0.008[Ccm^{-2}] \quad (2.1)$$

$$Q_n = q \cdot N_D = +0.032[Ccm^{-2}] \quad (2.2)$$

Dalla simulazione si ottiene una larghezza di RCS pari a:

$$x_{D_{SIM}} = 1.2824 \cdot 10^{-5} cm \Rightarrow 0.12824 \mu m, \tag{2.3}$$

mentre il valore teorico era di:

$$x_{D_{TEOR}} = 1.6227 \cdot 10^{-5} cm \Rightarrow 0.16227 \mu m. \tag{2.4}$$

2.3 Stimare il campo elettrico massimo all'equilibrio e confrontarlo col valore teorico

Di seguito è possibile vedere l'andamento simulato del campo elettrico:

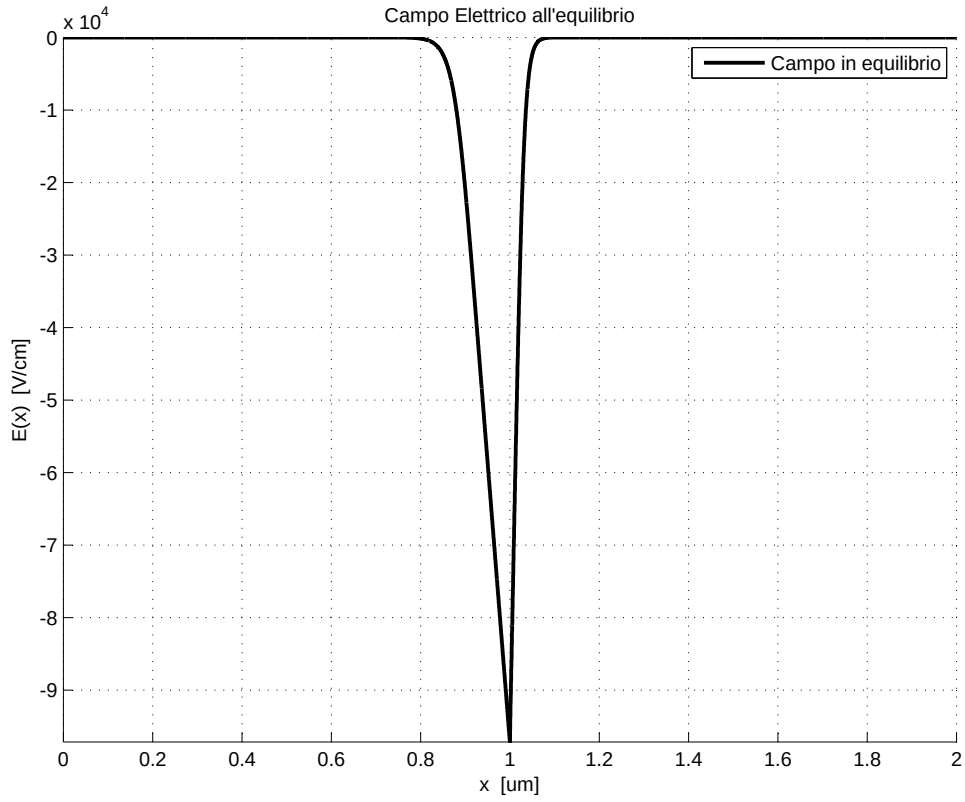

Figura 2.3: Campo elettrico all'equilibrio

Dalla simulazione si estrae il valore di picco del campo elettrico:

$$E_{MAX_{SIM}} = -9.7146 \cdot 10^4 V cm^{-1}, \tag{2.5}$$

mentre il valore teorico è di:

$$E_{MAX_{TEOR}} = -10.038 \cdot 10^4 V cm^{-1}. \tag{2.6}$$

2.4 Tracciare il diagramma a bande con $V_D = 0.5V$ e $V_D = -2V$

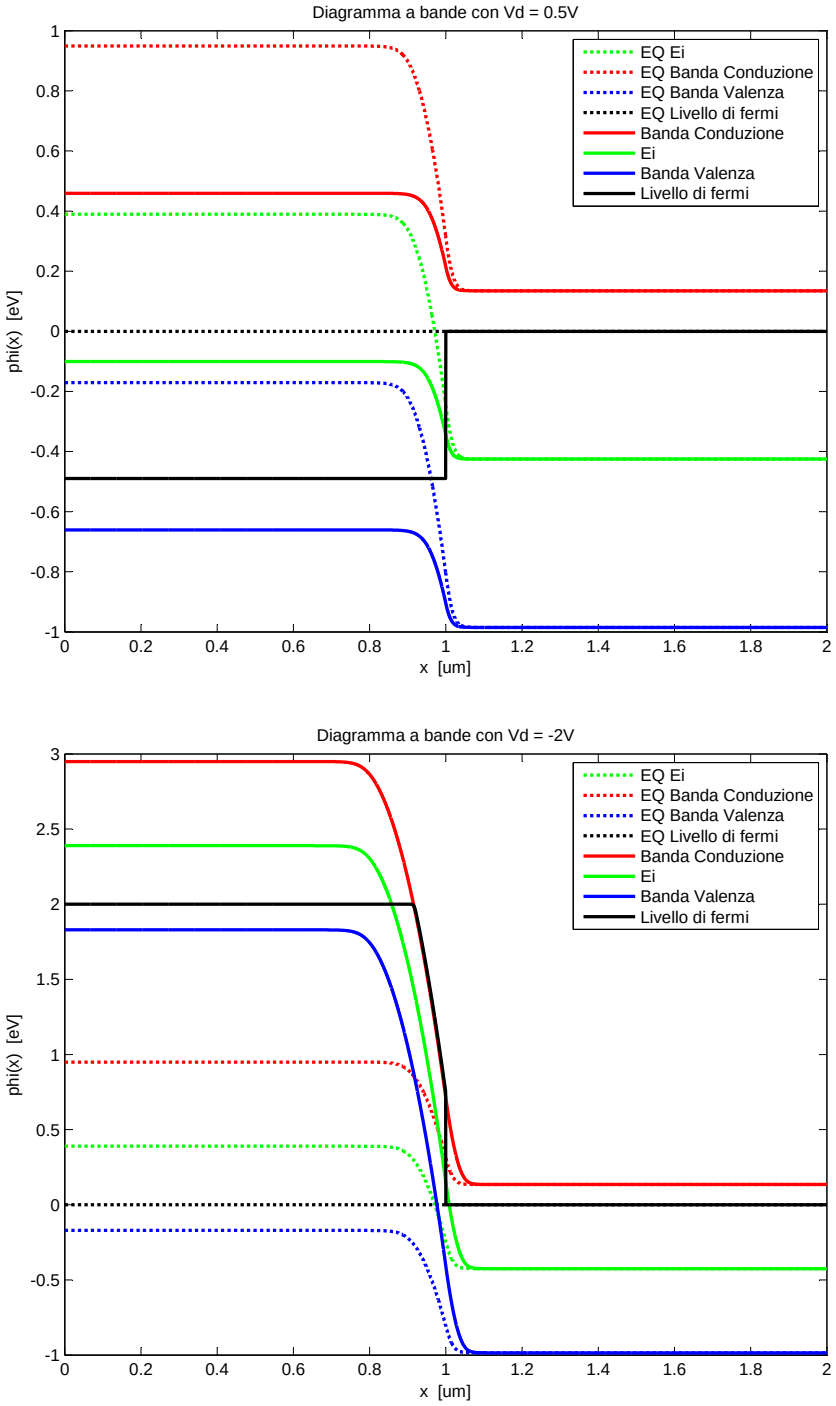

Figura 2.4: Diagrammi a bande con $V_D = 0.5V$ (in alto) e con $V_D = -2V$ (in basso).

Dai due grafici precedenti si possono vedere le situazioni corrispondenti a una situazione in cui la giunzione è fuori equilibrio a seguito della presenza di un potenziale esterno imposto al diodo. Quando la giunzione è fuori equilibrio si ha lo sdoppiamento del livello di Fermi e si ha in pratica la presenza dei quasi livelli di Fermi. Nelle due situazioni si legge direttamente dal grafico misurando la discontinuità della linea nera (livello di Fermi) il potenziale imposto: nel primo caso con $V_D = 0.5V$ le barriere si abbassano e rendono più probabile lo scambio di elettroni e lacune, in questa situazione il diodo è in diretta. Nel grafico successivo invece il diodo è in inversa perché si è applicata una tensione di $V_D = -2V$ che alza le barriere di una quantità $\phi_i - V_D$, anche in questa situazione la tensione esterna applicata si legge direttamente guardando la discontinuità del livello di Fermi.

2.5 Tracciare la caratteristica J_D vs. V_D del diodo da $-2V$ a $1V$

Caratteristica in diretta da zero a 1V:

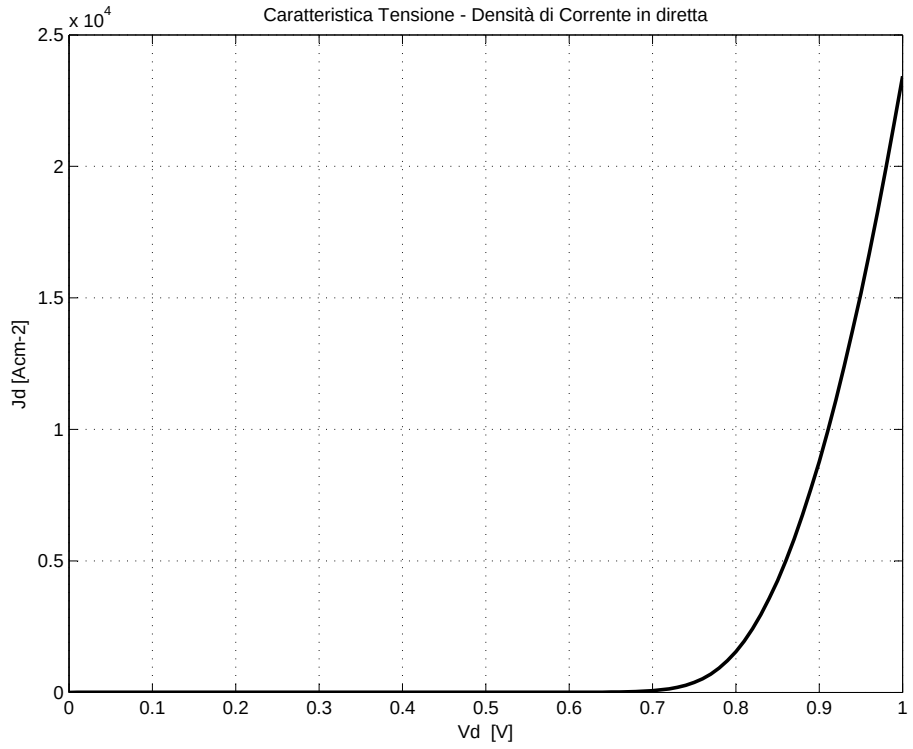

Figura 2.5: Caratteristica J_D vs. V_D del diodo in diretta

Nella figura precedente si vede il classico andamento esponenziale tipico di una giunzione p-n nell'ipotesi di bassa iniezione nella quale non si verificano cadute di tensione di natura resistiva dovute alle regioni quasi neutre.
Per disegnare poi la caratteristica in inversa del diodo ho ripetuto l'algoritmo perturbativo partendo nuovamente dalla situazione di equilibrio e procedendo verso la zona negativa.
Caratteristica in inversa da -2V a zero:

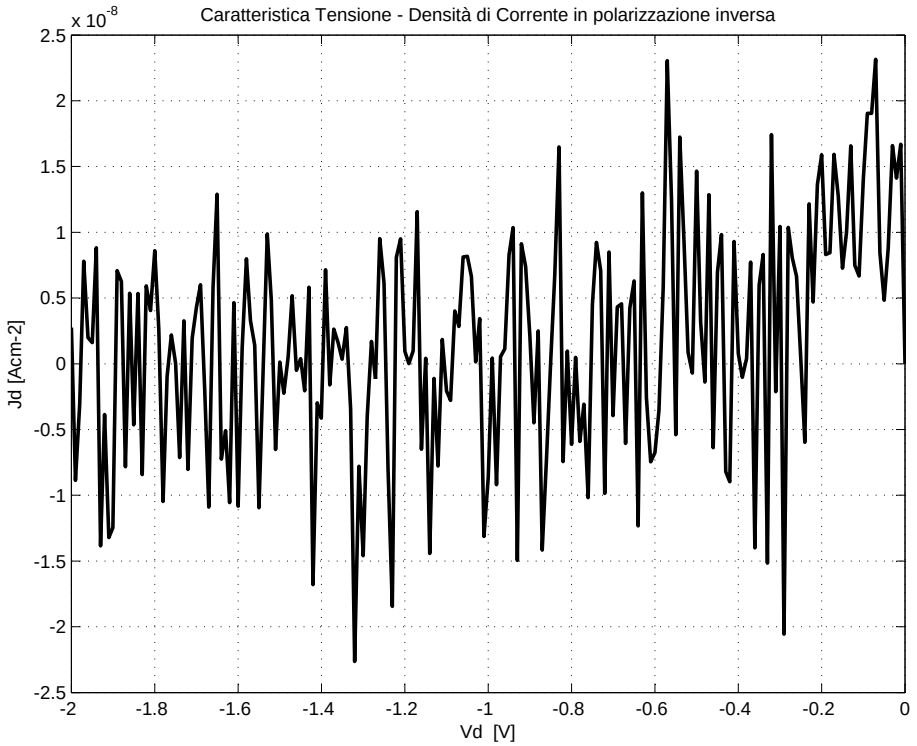

Figura 2.6: Caratteristica J_D vs. V_D del diodo in inversa

In questo caso si ottiene un andamento un po' strano vicino comunque a quello teorico, i valori assunti dalla caratteristica sono tutti nell'ordine di J_s e questo concorda con la relazione teorica:

$$J_D(V_D) = J_s(e^{\frac{V_D}{Vt}} - 1) \tag{2.7}$$

Che per tensioni negative risulta:

$$\lim_{V_D << 0} J_D(V_D) = -J_s. \tag{2.8}$$

Caratteristica complessiva da -2V a 1V:

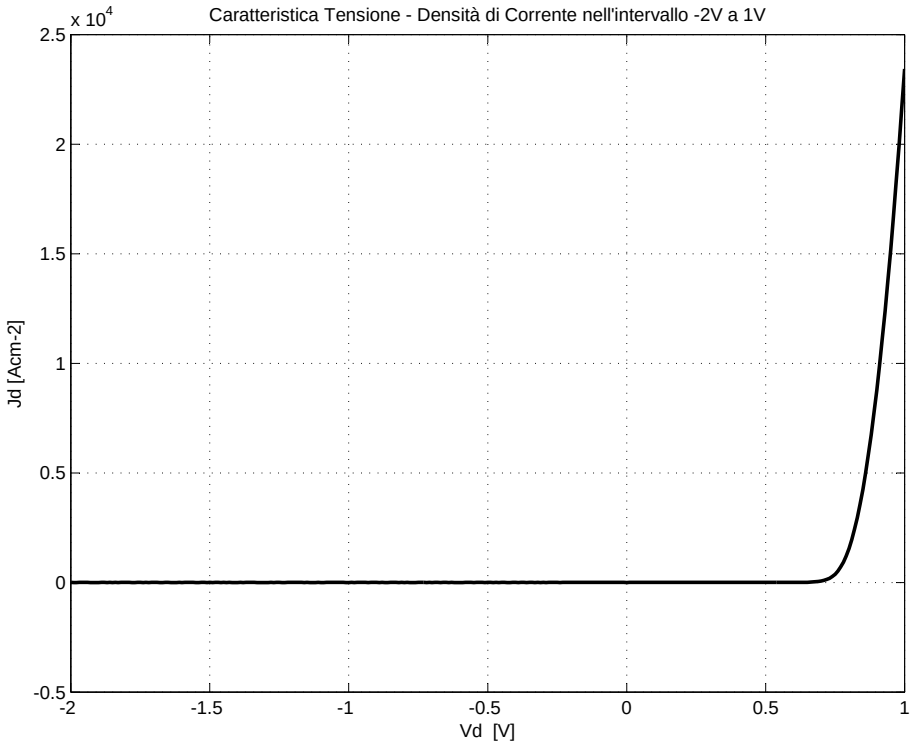

Figura 2.7: Caratteristica J_D vs. V_D del diodo

Dal grafico precedente si vede la densità di corrente in funzione della tensione nel range assegnato dall'esercitazione $[-2; 1]V$. Ovviamente essendo la parte negativa dell'ordine di 10^{-8} è impossibile distinguere la curva rispetto allo zero nelle tensioni negative. Diversamente si può pensare di stampare la curva in asse logaritmico. Caratteristica complessiva logaritmica da -2V a 1V:

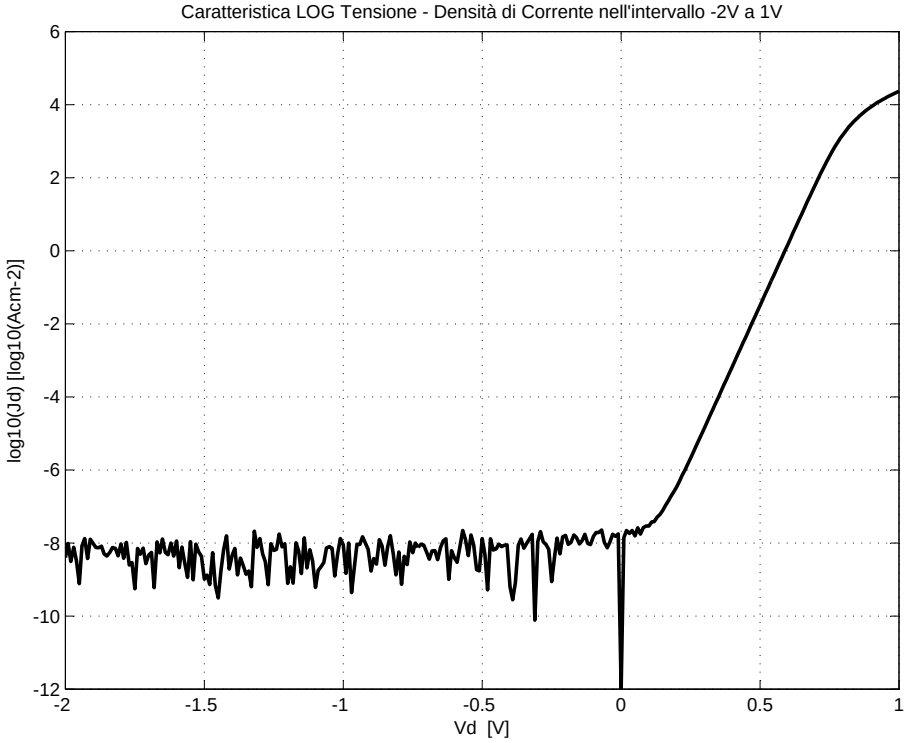

Figura 2.8: Caratteristica logaritmica J_D vs. V_D del diodo

Capitolo 3

Codice Matlab

- Main Program *MarcoGiunzionePN.m*,
- Funzione *LUsolve.m*,
- Funzione *Gummel.m*,

3.1 Main Program *MarcoGiunzionePN.m*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% HOMEWORK di Microelettronica Marco Piovesan 1012992              %
% studio di una giunzione pn                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all;
clear all;
clc;

%%costanti
global q Vt n_i Ld0 mu_n mu_p dx;

epsilon_vacuum = 8.854e-14;            %permittività dielettrica vuoto (F/cm)
epsilon_rel = 11.7;                    %permittività relativa silicio
epsilon = epsilon_vacuum*epsilon_rel;  %permittività silicio
k_bolt = 8.62e-5;                      %costante Boltzmann (eV/K)
temperature = 300;                     %temperatura (K)
Vt = k_bolt*temperature;               %energia termica KT (eV) o potenziale
                                       %termico KT/q (V) numericamente uguali
Eg = 1.12;                             %energygap (eV) a 300K (silicio)
Chi_s = 4.05;                          %affinità elettronica (V oppure eV)
Ei = Chi_s - Eg/2;                     %livello di fermi intrinseco
```

```
D_Eg = -2.7e-4;                          %dipendenza Eg dalla Temperatura (eV/K)
Eg = Eg - (temperature - 300)*D_Eg; %Eg con correzione temperatura
q = 1.602E-19;                           %carica dell'elettrone (C)
n_i = 1.45E10;                           %valore n_i diretto per il silicio

Ndrog_acc = 5E16;                        %drogaggio accettore
Ndrog_don = 2E17;                        %drogaggio donore
phy_0 = -Vt*log(Ndrog_acc/n_i);          % potenziale di equilibrio lato p
phy_1 = Vt*log(Ndrog_don/n_i);           % potenziale di equilibrio lato n

mu_p = 300; % mobilità lacune
mu_n = 600; % mobilità elettroni

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     DEFINIZIONE DOMINIO (ASSE X) E DISCRETIZZAZIONE    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Lsem = 2E-4;                    % lunghezza semiconduttore in cm
dx = Lsem/10000;                % passo di quantizzazione
x = 0:dx:Lsem;
x_um = x*10000;                 % conversione in cm
x_n = 0: dx : Lsem/2;
x_p = Lsem/2 : dx : Lsem;

n_dom = length(x);              % dimensione del dominio
%normalizzazione dominio (uso la lunghezza di Debye per normalizzare)
Ld0 = sqrt(epsilon*Vt/(q*n_i)); %costante di normalizzazione
y = x/Ld0;                      %dominio normalizzato
dy = y(2)-y(1);                 %passo quantizzazione dominio normalizzato
dy2 = dy*dy;
x_n = 0: dx : Lsem/2;
x_p = Lsem/2 : dx : Lsem;
l_n = length(x_n);
l_p = length(x_p);
p_dop = ones(l_p,1) * (-Ndrog_acc/n_i);
n_dop = ones (l_n,1) * (Ndrog_don/n_i);

dop = [p_dop; n_dop(1: l_n-1)]; % definisco il drogaggio
phy = Vt*log(abs(dop)).*sign(dop);       %potenziale
phy_in = ones(n_dom,1).*phy;
phy_in(1) = phy_0;

u0 = phy_in/Vt;      %normalizzo i potenziali utilizzando Vt
% concentrazioni all'equilibrio con potenziale già noralizzato
```

```
65 n = exp(u0);              %elettroni liberi
66 p = exp(-u0);             %lacune libere
67
68 %costruisco il sistema nella forma M*u=k con M matrice tridiagonale
69 %associata all'equazione di Poisson nel caso discreto, k vettore dei
70 %termini noti e u incognite (soluzione dell'equazione, andamento del
71 %potenziale cercato). Ragiono solo con le 3 diagonali, essendo la matrice
72 %tridiagonale
73 b = [0; ones(n_dom-2,1)/dy2; 0];        %colonna contenente la diagonale b
74                                         %inferiore (b2 ... b_n) con b_n=0
75                                         %per la condizione di Dirichlet
76 c = [0; ones(n_dom-2,1)/dy2; 0];        %colonna contenente la diagonale c
77                                         %superiore (c1 ... c_n-1) con c1=0
78                                         %per la condizione di Dirichlet
79 a = -n - p - 2/dy2;                     %colonna principale
80 a(1) = 1;  a(end) = 1;                  %impongo le condizioni Dirichlet
81 k = n - p - dop - u0.*(n + p);          %colonna dei termini noti
82 k(1) = u0(1);  k(end) = u0(end);        %impongo le condizioni al contorno
83
84
85 %eseguo le iterazioni per convergere ad una soluzione (andamento di phy(x))
86 Max_iter = 100;              %massimo numero di iterazioni
87 Tollerance = 1E-9;           %tolleranza (max discostamento tra le soluzioni
       )
88                             %il ciclo termina la distanza tra le ultime 2
89                             %soluzioni è inferiore alla tolleranza
90 for i=1:Max_iter
91     u = LUsolve(b,a,c,k);          %risolvo M*f=k ed ottengo una soluzione più
92                                    %vicina alla soluzione reale cercata
93     err = norm(u-u0,Inf);          %uso la norma infinita come riferimento
94                                    %dell'errore tra la soluzione calcolata e
95                                    %l'ultima disponibile
96     if (err < Tollerance)
97         break;        %esce dal ciclo, soluzione trovata (delta_u piccolo)
98     else                           %non raggiunta la tolleranza
99         %aggiorno la soluzione (sostituisco la soluzione attuale con la
100        %precedente e reitero. Si trova una soluzione se l'errore non
               diverge
101        u0(2:n_dom-1) = u(2:n_dom-1);     %aggiorno l'andamento del
               potenziale
102                                          %mantenendo le stesse condizioni
103                                          %al contorno
104        %anche n e p dipendono dal potenziale e vanno aggiornate
```

```
        n = exp(u0);              %elettroni liberi
        p = exp(-u0);             %lacune libere
        %anche la diagonale principale a ed il termine noto k dipendono dal
        %potenziale attraverso n e p, quindi anch'esse vanno aggiornate,
        %lasciando inalterate le condizioni al contorno
        a = -n - p - 2/dy2;                      %colonna principale
        a(1) = 1;  a(end) = 1;                   %impongo le condizioni Dirichlet
        k = n - p - dop - u0.*(n + p);      %colonna dei termini noti
        k(1) = u0(1);  k(end) = u0(end);      %impongo le condizioni al contorno
    end
    %itero usando risolvendo con i dati aggiornati
end
phy_fin_1 = u0*Vt;   % potenziale di equilibrio

u = u0;                 % salvo i valori corrispondenti all'equilibrio
uINV = u0;              % per poterli usare due volte nei due algoritmi perturbativi

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            BANDE ALL'EQUILIBRIO       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure();
hold on;
title('Diagramma a bande all''equilibrio');

Bconduzione = plot(x_um,-phy_fin_1+Eg/2, 'r');  %banda conduzione
EcB_Ei = plot(x_um,-phy_fin_1, 'g');                %banda Ei
Bvalenza = plot(x_um,-phy_fin_1-Eg/2, 'b');       %banda di valenza Ev
FerimiLevel = plot(x_um,zeros(n_dom,1),'k');      %livello di fermi Ef

xlabel('x  [um]');
ylabel('phi(x)  [eV]');
legend( 'Banda Conduzione', 'Ei','Banda Valenza', 'Livello di fermi');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CALCOLO DEL CAMPO ELETTRICO  ALL'EQUILIBRIO  e GRAFICO %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
e_eq = ones(n_dom, 1);
for i = 1: n_dom-1
    e_eq(i) = (phy_fin_1(i+1) - phy_fin_1(i))/dx;
end;
e_eq(end) = e_eq(end-1);
```

```
145
146 figure();
147 hold on;
148 plot(x_um, -e_eq, 'k','LineWidth',2);
149 grid on;
150 axis tight
151 title('Campo Elettrico all''equilibrio');
152 legend( 'Campo in equilibrio');
153 xlabel('x  [um]');
154 ylabel('E(x)  [V/cm]');
155 hold off;
156
157 [campo_elettrico1,v] = max(e_eq);  %trovo valore massimo [v]
158 Campo_elettrico_massimo_Simulato = campo_elettrico1
159
160 %%  calcolo i valori teorici
161 phi_teorico = Vt * log(Ndrog_acc*Ndrog_don/(n_i^2));
162 RCS_teorica = sqrt((2*epsilon/q)*(1/Ndrog_don + 1/Ndrog_acc)*phi_teorico);
163 Campo_elettrico_teorico = 2*phi_teorico / RCS_teorica
164
165 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
166 %CALCOLO DELLA DENSITA' DI CARICA ALL'EQUILIBRIO e GRAFICO %
167 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
168 q_eq = ones(n_dom,1); % trovo la densità derivando il campo elettrico
169 for i = 1: n_dom-1
170     q_eq(i) = (e_eq(i+1) - e_eq(i)).*epsilon./dx;
171 end;
172 q_eq(n_dom) = q_eq(n_dom-1);
173
174 figure();
175 hold on;
176 grid on;
177 plot(x_um, -q_eq, 'k','LineWidth',2);
178 xlabel('x  [um]');
179 ylabel('densità di carica  [C/cm2]');
180 title('Densità di carica all''equilibrio');
181 legend('Densità di carica in equilibrio');
182
183 hold off;
184
185 % misuro la larghezza della regione di carica spaziale
186 delta=10^(-6); % Delta per fissare uno zero
187 %ricerco primo zero da sinistra
```

```
flag= 1;
iSx=1;
while (flag)

    if q_eq(iSx) > delta
        flag = 0;
    else
        iSx= iSx + 1;
    end

end
%ricerco primo zero da destra
flag=1;
iDx = length(q_eq);

while (flag)

    if q_eq(iDx)< delta
        flag = 0;
    else
        iDx= iDx - 1;
    end
end
%calcolo dimensione RCS
Dimensione_RCS_in_um =(x_um(iDx)-x_um(iSx)) * 0.1

RCS_teorica_in_um = RCS_teorica * 10^4% confronto con quella teorica

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  POLARIZZAZIONE         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           ALGORITMO PERTURBATIVO            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Vs = 1;   %tensione applicata al diodo

delta_Vs = 1e-2; % era 1e-2         %incremento da dare ad ogni passo in V
steps = round(abs(Vs)/delta_Vs);    %numero passi da eseguire
%inizializzo le variabili n, p ed u ai valori all'equilibrio ideali
p_eq = exp(-u);
n_eq = exp(u);
mu_p = ones(n_dom, 1) * mu_p;
```

```
231 mu_n = ones(n_dom, 1) * mu_n;
232
233 %inizializzo le variabili che ospiteranno le correnti
234 Jtot = zeros(n_dom,1); Jn =    zeros(n_dom,1);
235 Jp =    zeros(n_dom,1); Jn_drift = zeros(n_dom,1);
236 Jn_diffusion = zeros(n_dom,1); Jp_drift =      zeros(n_dom,1);
237 Jp_diffusion = zeros(n_dom,1);
238
239 %inizializzo i vettori per il calcolo della caratteristica J-V
240 Vs_vec = zeros(steps+1,1);
241 J_vec = zeros(steps+1,1);
242
243 Vs_step = 0;
244
245 for i = 1:steps
246     fprintf(['Step ', num2str(i), '/', num2str(steps), '\n']);%debug passo
          corrente
247     %aggiorno la condizione al contorno (potenziale all'estremità del
248     %diodo, lato P) di volta in volta (perturbazione data da delta_Vd)
249     %condizione al contorno (polarizzazione)
250     u(1) = u(1) + sign(Vs)*delta_Vs/Vt;
251
252     [u, n, p] = Gummel(u, n, p, dop, dy, Tollerance, Max_iter);%eseguo l'
          algoritmo di Gummel che mi restituisce le soluzioni n,p,u per
253     %una polarizzazione Vd_vec(i).
254
255     %%%%%%%%%%%%%correnti%%%%%%%%%%%%%%%%%%%%%%%%%%%%
256     J0n = mu_n(1:n_dom-1).*q.*n_i.*Vt./(dy.*Ld0);
257     J0p = mu_p(1:n_dom-1).*q.*n_i.*Vt./(dy.*Ld0);
258     %inizializzo le variabili (vettori n_dom elementi)
259     Jtot = zeros(n_dom,1);
260     Jn_drift = Jtot;  Jn_diffusion = Jtot;
261     Jp_drift = Jtot;  Jp_diffusion = Jtot;
262     %calcolo le correnti per gli elettroni
263     %NB: prendo i valori medi per ogni passo, in questo modo ottengo un
          vettore di
264     %n_dom-1 elementi più preciso rispetto a n(1:n_dom-1)
265     n_med = ((n(1:n_dom-1)+n(2:n_dom)))./2;
266     Jn_drift(1:n_dom-1) = -J0n.*n_med.*(u(2:n_dom)-u(1:n_dom-1));
267     Jn_drift(end) = Jn_drift(end-1);
268     Jn_diffusion(1:n_dom-1) = J0n.*(n(2:n_dom)-n(1:n_dom-1));
269     Jn_diffusion(end) = Jn_diffusion(end-1);
270     %NB: la media serve per livellare la curva (a causa di approssimazioni
```

```
    %potrebbe non risultare una corrente costante lungo x)
    Jn = ones(n_dom,1).*mean(Jn_drift + Jn_diffusion);
    %calcolo le correnti per le lacune
    %NB: prendo i valori medi per ogni passo, in questo modo ottengo un vettore di
    %n_dom-1 elementi più preciso rispetto a p(1:n_dom-1)
    p_med = ((p(1:n_dom-1)+p(2:n_dom)))./2;
    Jp_drift(1:n_dom-1) = -J0p.*p_med.*(u(2:n_dom)-u(1:n_dom-1));
    Jp_drift(end) = Jp_drift(end-1);
    Jp_diffusion(1:n_dom-1) = -J0p.*(p(2:n_dom)-p(1:n_dom-1));
    Jp_diffusion(end) = Jp_diffusion(end-1);
    %NB: la media serve per livellare la curva (a causa di approssimazioni
    %potrebbe non risultare una corrente costante lungo x)
    Jp = ones(n_dom,1).*mean(Jp_drift + Jp_diffusion);
    %Calcolo la corrente totale
    Jtot = Jn + Jp;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %calcolo il diagramma V-J
    Vs_step = sign(Vs)*delta_Vs*i;%tensione correntemente applicata al diodo
    Vs_vec(i+1) = Vs_step;
    J_vec(i+1) = Jtot(end);

    % eseguo il ciclo da 0 a 1V però quando arrivo a 0.5V salvo il valore u
    % in modo da avere il valore giusto per stampare il relativo diagramma
    % a bande
    if Vs_step < 0.5,     % steps/2 corrisponde a 0.5V
        u_05 = u;
        n_05 = n;
        p_05 = p;
    end
end
u_1 = u;     % potenziale relativo a 1V
phy_fin_1V = u_1*Vt; % potenziale di non equilibrio a Vs = 1V
phy_fin_2 = u_05*Vt; % potenziale di non equilibrio a Vs = 0.5V

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            BANDE ALL'EQUILIBRIO           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure();
plot(x_um,-phy_fin_1, ':g','LineWidth',2);          %banda Ei
hold on;
plot(x_um,-phy_fin_1+Eg/2, ':r','LineWidth',2);  %banda conduzione Ec
```

```
plot(x_um,-phy_fin_1-Eg/2, ':b','LineWidth',2);  %banda di valenza Ev
plot(x_um,zeros(n_dom,1),':k','LineWidth',2);     %livello di fermi Ef
xlabel('x  [um]');
ylabel('phi(x)  [eV]');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      BANDE FUORI EQUILIBRIO   V=0.5V      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%fuori_equilibrio =
plot(x_um, -phy_fin_2+Eg/2, 'r','LineWidth',2); % banda di conduzione fuori equilibrio
plot(x_um, -phy_fin_2, 'g','LineWidth',2);        % livello Ei fuori equilibrio
plot(x_um, -phy_fin_2-Eg/2, 'b','LineWidth',2); % banda di valenza fuori equilibrio

Efp = -Vt * (log(p_05)+ u_05);
Efn = Vt * (log(n_05) - u_05);

Ef_Tot=Efn;
for i = 1:length(Efn)/2
    Ef_Tot(i)=Efp(i);
end

plot(x_um, Ef_Tot, 'k', 'LineWidth',2);
title('Diagramma a bande con Vd = 0.5V');
legend('EQ Ei', 'EQ Banda Conduzione', 'EQ Banda Valenza',...
    'EQ Livello di fermi', 'Banda Conduzione', 'Ei', 'Banda Valenza', 'Livello di fermi');
grid off;
hold off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CARATTERISTICA TENSIONE CORRENTE DIODO IN INVERSA,
%   RIPETO L'ALGORTIMO PERTURBATIVO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u = uINV;                    % riparto dalla condizione di equilibrio

VsINV = -2;                  %tensione inversa applicata al diodo

steps = round(abs(VsINV)/delta_Vs);        %numero passi da eseguire
Vs_step = 0;
```

```
351
352 p_eq = exp(-u);%inizializzo le variabili n, p ed u ai valori all'equilibrio
        ideali
353 n_eq = exp(u);
354
355 %inizializzo le variabili che ospiteranno le correnti
356 Jtot = zeros(n_dom,1);  Jn = zeros(n_dom,1);  Jp = zeros(n_dom,1);
357 Jn_drift = zeros(n_dom,1); Jn_diffusion = zeros(n_dom,1);
358 Jp_drift = zeros(n_dom,1);  Jp_diffusion = zeros(n_dom,1);
359
360 %inizializzo i vettori per il calcolo della caratteristica J-V
361 Vs_vecINV = zeros(steps+1,1);
362 J_vecINV  = zeros(steps+1,1);
363
364 for i = 1:steps
365     fprintf(['Step ', num2str(i), '/', num2str(steps), '\n']);%debug passo
          corrente
366     %aggiorno la condizione al contorno (potenziale all'estremità del
367     %diodo, lato P) di volta in volta (perturbazione data da delta_Vd)
368     %condizione al contorno (polarizzazione)
369     u(1) = u(1) + sign(VsINV)*delta_Vs/Vt;
370
371     [u, n, p] = Gummel(u, n, p, dop, dy, Tollerance, Max_iter);%eseguo l'
          algoritmo di Gummel che mi restituisce le soluzioni n,p,u per
372     %una polarizzazione Vd_vec(i).
373
374     %%%%%%%%%%correnti%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
375     J0n = mu_n(1:n_dom-1).*q.*n_i.*Vt./(dy.*Ld0);
376     J0p = mu_p(1:n_dom-1).*q.*n_i.*Vt./(dy.*Ld0);
377     %inizializzo le variabili (vettori n_dom elementi)
378     Jtot = zeros(n_dom,1);
379     Jn_drift = Jtot;  Jn_diffusion = Jtot;
380     Jp_drift = Jtot;  Jp_diffusion = Jtot;
381     %calcolo le correnti per gli elettroni
382     %NB: prendo i valori medi per ogni passo, in questo modo ottengo un
          vettore di
383     %n_dom-1 elementi più preciso rispetto a n(1:n_dom-1)
384     n_med = ((n(1:n_dom-1)+n(2:n_dom)))./2;
385     Jn_drift(1:n_dom-1) = -J0n.*n_med.*(u(2:n_dom)-u(1:n_dom-1));
386     Jn_drift(end) = Jn_drift(end-1);
387     Jn_diffusion(1:n_dom-1) = J0n.*(n(2:n_dom)-n(1:n_dom-1));
388     Jn_diffusion(end) = Jn_diffusion(end-1);
389     %NB: la media serve per livellare la curva (a causa di approssimazioni
```

```
    %potrebbe non risultare una corrente costante lungo x)
    Jn = ones(n_dom,1).*mean(Jn_drift + Jn_diffusion);
    %calcolo le correnti per le lacune
    %NB: prendo i valori medi per ogni passo, in questo modo ottengo un vettore di
    %n_dom-1 elementi più preciso rispetto a p(1:n_dom-1)
    p_med = ((p(1:n_dom-1)+p(2:n_dom)))./2;
    Jp_drift(1:n_dom-1) = -J0p.*p_med.*(u(2:n_dom)-u(1:n_dom-1));
    Jp_drift(end) = Jp_drift(end-1);
    Jp_diffusion(1:n_dom-1) = -J0p.*(p(2:n_dom)-p(1:n_dom-1));
    Jp_diffusion(end) = Jp_diffusion(end-1);
    %NB: la media serve per livellare la curva (a causa di approssimazioni
    %potrebbe non risultare una corrente costante lungo x)
    Jp = ones(n_dom,1).*mean(Jp_drift + Jp_diffusion);
    %Calcolo la corrente totale
    Jtot = Jn + Jp;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %calcolo il diagramma V-J
    Vs_vecINV(i+1) = sign(VsINV)*delta_Vs*i; %tensione correntemente applicata al diodo
    J_vecINV(i+1) = Jtot(end);
end

phy_fin_INV = u*Vt; % potenziale di non equilibrio a -2V

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              BANDE ALL'EQUILIBRIO          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure();
plot(x_um,-phy_fin_1, ':g','LineWidth',2);          %banda intrinseco Ei
hold on;
plot(x_um,-phy_fin_1+Eg/2, ':r','LineWidth',2);   %banda conduzione Ec
plot(x_um,-phy_fin_1-Eg/2, ':b','LineWidth',2);   %banda di valenza Ev
plot(x_um,zeros(n_dom,1),':k','LineWidth',2);       %livello di fermi Ef
xlabel('x  [um]');
ylabel('phi(x)  [eV]');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    BANDE FUORI EQUILIBRIO   V=-2V        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot(x_um, -phy_fin_INV+Eg/2, 'r','LineWidth',2); % banda di conduzione fuori equilibrio
```

```
429 plot(x_um, -phy_fin_INV, 'g','LineWidth',2);        % livello Ei fuori
    equilibrio
430 plot(x_um, -phy_fin_INV-Eg/2, 'b','LineWidth',2); % banda di valenza fuori
    equilibrio
431
432 Efp = -Vt *(log(p)+u);  % calcola i quasi livelli di Fermi
433 Efn = Vt * (log(n)-u);
434
435 Ef_Tot=Efn;
436 for i = 1:length(Efn)/2
437     Ef_Tot(i)=Efp(i);
438 end
439
440 plot(x_um, Ef_Tot, 'k', 'LineWidth',2); % livello di Fermi fuori equilibrio
441 title('Diagramma a bande con Vd = -2V');
442 legend('EQ Ei', 'EQ Banda Conduzione', 'EQ Banda Valenza',...
443     'EQ Livello di fermi', 'Banda Conduzione', 'Ei', 'Banda Valenza', '
        Livello di fermi');
444 grid off;
445 hold off;
446
447 %% CARATTERISTICHE J - V
448
449 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
450 %   CARATTERISTICA TENSIONE CORRENTE DIODO IN DIRETTA
451 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
452 figure();
453 plot(Vs_vec, J_vec, 'k','LineWidth',2);
454 title('Caratteristica Tensione - Densità di Corrente in diretta');
455 xlabel('Vd  [V]');
456 ylabel('Jd  [Acm-2]');
457 grid on;
458
459 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
460 %   CARATTERISTICA TENSIONE CORRENTE DIODO IN INVERSA  %
461 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
462 figure();
463 plot(Vs_vecINV, J_vecINV, 'k','LineWidth',2);
464 title('Caratteristica Tensione - Densità di Corrente in polarizzazione
    inversa');
465 xlabel('Vd  [V]');
466 ylabel('Jd  [Acm-2]');
467 grid on;
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    CARATTERISTICA TENSIONE CORRENTE DIODO COMPLESSIVA -2V a 1V      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%unisco le due caratteristiche per plottare un unico grafico
Tensioni(length(Vs_vec) + length(Vs_vecINV)) =0 ; % dimensione 302 = 101 + 201

for i = 1:length(Vs_vecINV)
    Tensioni(i)=Vs_vecINV(length(Vs_vecINV) - i + 1);
end
for i = length(Vs_vecINV) : length(Vs_vecINV)+length(Vs_vec)-1
    Tensioni(i+1)=Vs_vec(i-length(Vs_vecINV)+1);
end

DensitaDiCorrenti(length(J_vec) + length(J_vecINV)) = 0;

for i = 1:length(J_vecINV)
    DensitaDiCorrenti(i)=J_vecINV(length(J_vecINV) - i +1);
end
for i = length(J_vecINV):length(J_vecINV)+length(J_vec)-1
    DensitaDiCorrenti(i+1)=J_vec(i-length(J_vecINV)+1);
end

figure();
plot(Tensioni, DensitaDiCorrenti, 'k','LineWidth',2);
title('Caratteristica Tensione - Densità di Corrente nell''intervallo -2V a 1V');
xlabel('Vd  [V]');
ylabel('Jd [Acm-2]');
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    LOGAR CARATTERISTICA TENSIONE CORRENTE DIODO COMPLESSIVA -2V a 1V  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% aggiunge l'asisntoto per il logaritmo in zero
for i = 1:length(DensitaDiCorrenti)
    if abs( DensitaDiCorrenti(i) ) < 1e-12;
        DensitaDiCorrenti(i) = 1e-12;
    end
end

figure();
```

```
509 plot(Tensioni,log10(abs(DensitaDiCorrenti)), 'k','LineWidth',2);
510 title('Caratteristica LOG Tensione - Densità di Corrente nell''intervallo
    -2V a 1V');
511 xlabel('Vd  [V]');
512 ylabel('log10(Jd) [log10(Acm-2)]');
513 grid on;
```

MarcoPiovesan1012992pn/MarcoGiunzionePN.m

3.2 Funzione *LUsolve.m*

```
1  %%Algoritmo per la soluzione dell'equazione M*f-k=0 --> f=M^-1*k
2  %tramite metodo LU b, a e c cono le diagonali della matrice TRIDIAGONALE M
3  %da invertire, mentre k è il termine noto.
4  %NB: b,a,c,k sono vettori Cononna di n elementi anche se
5  %contengono meno di n elementi (ad esempio b e c contengono n-1 elementi).
6  function f = LUsolve(b,a,c,k)
7  n_matr = length(a);          %dimensione matrice ricavata dal numero di
8                               %elementi della diagonale principale
9  %vettore contenente i valori di alpha (n valori, da alpha_1 ad alpha_n)
10 alpha = zeros(n_matr,1);
11 %vettore contenente i valori di beta (n-1 valori, da beta_2 a beta_n)
12 beta = zeros(n_matr, 1);
13 %vettore contenente i valori di gamma (n-1 valori, da gamma_1 a gamma_n-1)
14 gamma = zeros(n_matr, 1);
15
16 %Scompongo la matrice M in L ed U tali che M=L*U, In poche parole calcolo
17 %le diagonali delle matrici L ed U a partire dalle diagonali di M
18 %la cosa può essere fatta con 2n iterazioni
19 gamma = c;              %la diagonale gamma della matrice U è pari alla colonna
20                         %c della matrice M
21 %calcolo i primi elementi di alpha e beta
22 alpha(1) = a(1);
23 beta(1) = 0;            %(beta parte dall'indice 2)
24 %calcolo gli altri elementi rimanenti
25 for i=2:n_matr
26     beta(i) = b(i)/alpha(i-1);
27     alpha(i) = a(i) - gamma(i-1)*beta(i);
28 end
29 %L'equazione M*f=k diventa L*U*f=k, quindi si può porre y = U*f e quindi il
30 %sistema risulta L*y=k. L'incognita è f, quindi determinando y si può
31 %calcolare f invertendo U (molto semplice) --> f=U^-1*y
```

```
32 %Per determinare y si deve invertire L, cosa molto semplice che richiede
33 %solo n iterazioni:  y=L^-1*k  dove k è il vettore colonna dei termini noti
34 y = zeros(n_matr, 1);              %vettore colonna contenente i coeff y_i
35 y(1) = k(1);                       %primo elemento
36 %elementi rimanenti
37 for i = 2:n_matr
38     y(i) = k(i) - beta(i)*y(i-1);
39 end
40
41 %L'ultimo passo per la soluzione del sistema M*f=k consiste come detto
42 %nella soluzione di y = U*f --> f=U^-1*y. L'inversione di U viene fatta
43 %in n iterazioni
44 f = zeros(n_matr,1);          %inizializzo f
45 %calcolo l'ultimo elemento f_n  (si deve andare a ritroso perchè f_n
       dipende fa f_n+1)
46 f(n_matr) = y(n_matr)/alpha(n_matr);
47 for i=(n_matr-1):-1:1
48     f(i) = (y(i) - gamma(i)*f(i+1))/alpha(i);
49 end
50 % f è la soluzione del sistema in uscita dalla funzione
51 end
```

MarcoPiovesan1012992pn/LUsolve.m

3.3 Funzione *Gummel.m*

```
1  %%Funzione per la soluzione dell'equazione di continuità e delle equazioni
2  %%drift-diffusion in modo autoconsistente a partire dalle condizioni
3  %%iniziali all'equilibrio (ideali). L'algoritmo Gummel trova le soluzioni
4  %%u (potenziale normalizzato), n (concentrazione elettroni) e p
5  %%(concentrazione lacune) nel semiconduttore, tali che risolvano tutte e
6  %%tre le equazioni differenziali (in forma discreta).
7  %%Ho ipotizzato U=0 (trascuro effetti centri G/R)
8  function [u,n,p] = Gummel(u,n,p,N_dop,dy,Tollerance,Max_iter)
9  n_dom = length(u);            %dimensione dominio monodimensionale
10 dy2 = dy*dy;
11 b1 = zeros(n_dom,1);  c1=b1;  a1=b1;  k1=b1;      %inizializzo a,b,c,k
12 %Eseguo il ciclo iterativo che calcola la soluzione (u,n,p) delle 3
13 %equazioni simultaneamente (per convergenza).
14 b2 = zeros(n_dom,1);  c2=b2;  a2=b2;  k2=b2;
15 b3 = zeros(n_dom,1);  c3=b3;  a3=b3;  k3=b3;
16 %NB: alla prima chiamata della funzione u,n,p sono i valori all'equilibrio
```

```
17 for i = 1:Max_iter
18     %PASSO 1: SOLUZIONE EQUAZIONE POISSON NEL CASO DI NON EQUILIBRIO
19     %costruisco il sistema nella forma M*u=k con M matrice tridiagonale
20     %associata all'equazione di Poisson nel caso discreto, k vettore dei
21     %termini noti e u incognite. Nel caso di non equilibrio n e p non
22     %dipendono solo da u ma anche dalla polarizzazione (quasi livelli di
23     %fermi) e quindi dalle equazioni drift-diffusion. Al primo passo vale
24     %n=n_equilibrio e p=p_equilibrio, verranno poi successivamente
25     %raffinate risolvendo le equazioni drift-diffusion (passi che portano
26     %alla convergenza, soluzione autoconsistente)
27     b1(2:n_dom-1) = ones(n_dom-2,1)/dy2;      %colonna contenente la
           diagonale b
28     c1(2:n_dom-1) = ones(n_dom-2,1)/dy2;      %colonna contenente la
           diagonale c
29     a1 = -n - p - 2/dy2;                      %colonna principale
30     k1 = n - p - N_dop - u.*(n + p);          %colonna dei termini noti
31     a1(1) = 1;  a1(end) = 1;                   %impongo le condizioni
           Dirichlet
32     k1(1) = u(1);  k1(end) = u(end);
33     %risolvo il sistema lineare determinando una u più vicina alle
           condizioni date (n,p,u precedenti)
34     u_new = LUsolve(b1,a1,c1,k1);
35
36     %PASSO 2: SOLUZIONE DELL'EQUAZIONE DI CONTINUITà PER GLI ELETTRONI
37     %costruisco il sistema nella forma M*n=k con M matrice tridiagonale
38     %associata all'equazione drift-diffusion per elettroni nel caso
           discreto,
39     %k vettore dei termini noti e n incognite (distribuzione concentrazione
40     %degli elettroni. L'equazione utilizza il valore di u calcolato
41     %precedentemente per calcolare un valore di n più vicino alla soluzione
42     %finale
43     b2(2:n_dom-1) = 1 + (u_new(2:n_dom-1)-u_new(1:n_dom-2))./2;
44     c2(2:n_dom-1) = 1 - (u_new(3:n_dom)-u_new(2:n_dom-1))./2;
45     a2(2:n_dom-1) = -2 - (u_new(3:n_dom)-u_new(2:n_dom-1))./2 + (u_new(2:
           n_dom-1)-u_new(1:n_dom-2))./2;
46     k2(2:n_dom-1) = zeros(n_dom-2,1);
47     %impongo le condizioni al contorno di dirichlet (n0 agli estremi)
48     a2(1) = 1;  a2(end) = 1;
49     k2(1) = n(1);  k2(end) = n(end);
50     %risolvo il sistema lineare determinando la n associata all'ultimo u
           calcolato
51     n_new = LUsolve(b2,a2,c2,k2);
52
```

```
53      %PASSO 3: SOLUZIONE DELL'EQUAZIONE DI CONTINUITà PER LE LACUNE
54      %costruisco il sistema nella forma M*p=k con M matrice tridiagonale
55      %associata all'equazione drift-diffusion per le lacune nel caso
            discreto,
56      %k vettore dei termini noti e p incognite (distribuzione concentrazione
57      %delle lacune. L'equazione utilizza il valore di u calcolato
58      %precedentemente per calcolare un valore di p più vicino alla soluzione
59      %finale
60      b3(2:n_dom-1) = 1 - (u_new(2:n_dom-1)-u_new(1:n_dom-2))./2;
61      c3(2:n_dom-1) = 1 + (u_new(3:n_dom)-u_new(2:n_dom-1))./2;
62      a3(2:n_dom-1) = -2 + (u_new(3:n_dom)-u_new(2:n_dom-1))./2 - (u_new(2:
            n_dom-1)-u_new(1:n_dom-2))./2;
63      k3(2:n_dom-1) = zeros(n_dom-2,1);
64      %impongo le condizioni al contorno di dirichlet (p0 agli estremi)
65      a3(1) = 1;  a3(end) = 1;
66      k3(1) = p(1);  k3(end) = p(end);
67      %risolvo il sistema lineare determinando la p associata all'ultimo u
            calcolato
68      p_new = LUsolve(b3,a3,c3,k3);
69
70      %PASSO 4: controllo se ho converso alla soluzione finale controllando
71      %se l'errore tra la soluzione corrente e l'ultima soluzione è inferiore
72      %alla tolleranza indicata
73      err_u = norm(u_new-u,inf);              %errore su u
74      err_n = norm(log(n_new./n),inf);        %errore su n
75      err_p = norm(log(p_new./p),inf);        %errore su n
76      %se tutti gli errori sono più piccoli della tolleranza allora ho
77      %trovato le soluzioni u,n,p cercate e posso uscire dal ciclo
78      if (err_u < Tollerance && err_n < Tollerance && err_p < Tollerance)
79          break;
80      end
81      %se gli errori non sono più piccoli della tolleranza devo aggiornare
82      %u,n,p agli ultimi valori calcolati (più vicini alla soluzione) ed
83      %eseguire una ulteriore iterazione
84      u = u_new;  n = n_new;  p = p_new;
85  end
86  %A questo punto ho  i valori di u,n e p.
87  end
```

MarcoPiovesan1012992pn/Gummel.m