Solar cells with Quantum Dots

The team: Coord. Prof.: Răzvan Mihai Mitroi

Bosînceanu Andrada 315CC
Mirion Alina 312CC
Tiţa Andreea 314CC

CELLS AND QUANTUM DOTS

Introduction

The demands regarding the need of energy of the on growing society that must be met seem to be higher and the common means by which one can achieve its energetical target are not a real solution (gas, coal, wood, etc. all these are finite resources). That is why our attention has to be redirected. One of the most promising solutions is the use of solar energy. The reasons are simple:

A. We need ~ 30 TW of power and the sun gives us 120,000 TW
B. Solar cells are safe and have few non-desirable environmental impacts
C. Solar cells can replace coal
D. Solar cells provide electricity exactly when we need it the most

/*comentariul figurii*/

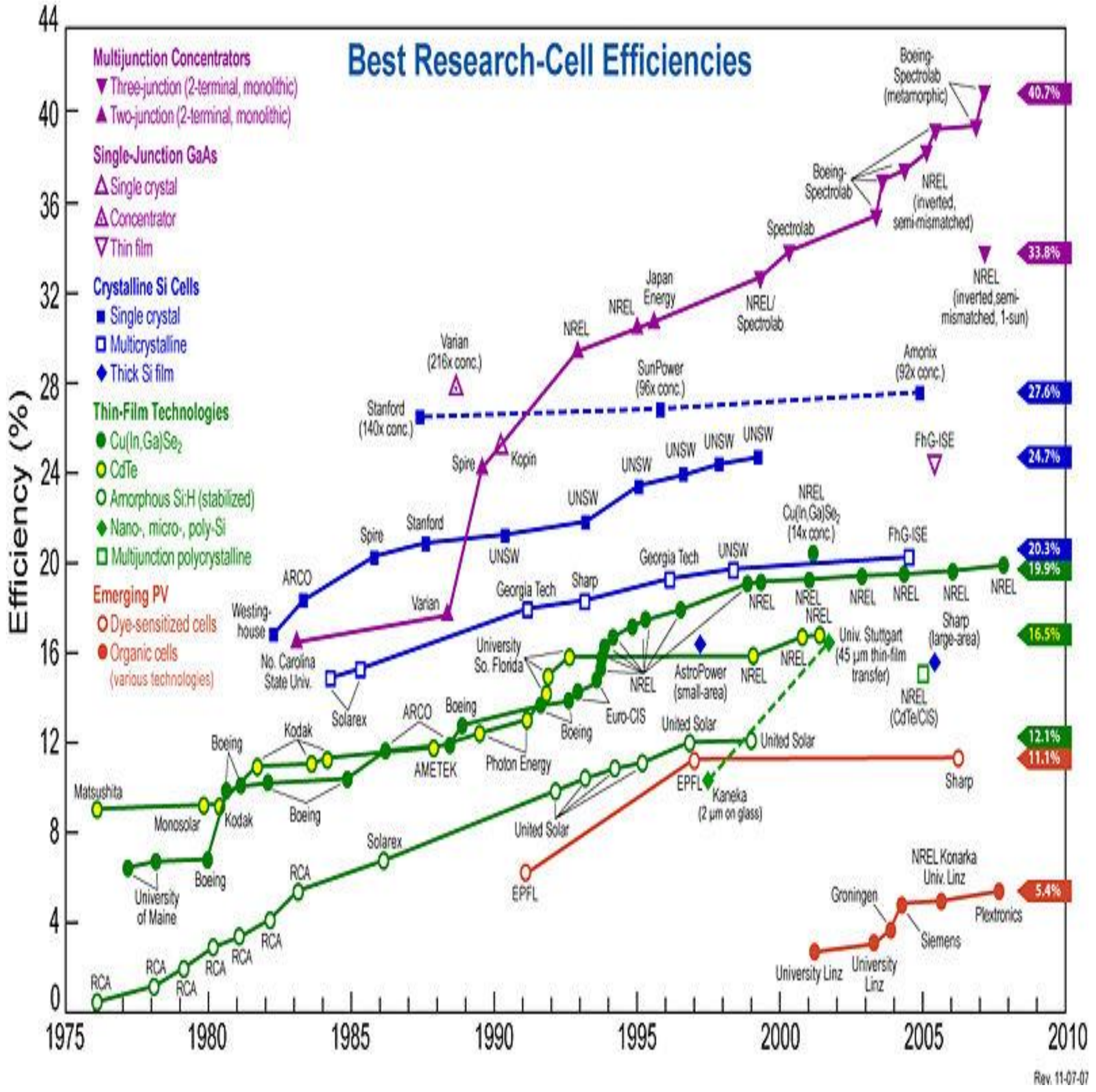
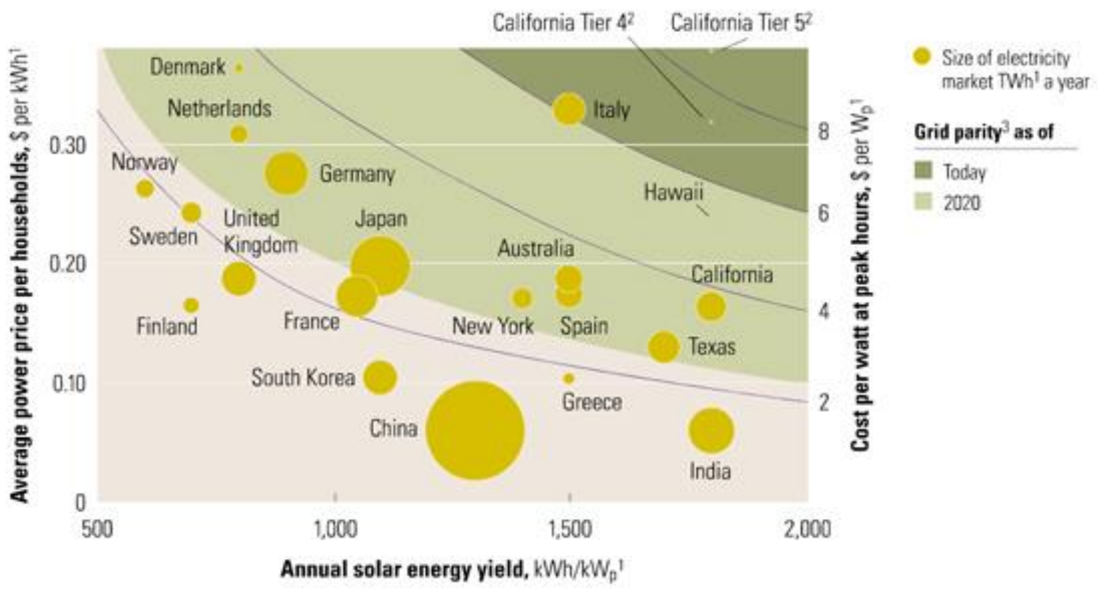


Fig.1 The progress of solar cells efficiency

In figure 1 we can see the conversion efficiency of solar energy in electrical energy using solar cells with quantum dots

A solar cell is a solid state electrical device that converts the energy of light directly into electricity by the photovoltaic effect.
A quantum dot is a portion of matter (e.g. semiconductor) whose excitons –an exciton is a bound state of an electron and hole which are attracted to each other by the electrostatic Coulomb force – are confined in all three spatial dimensions. In other words, quantum dots are very small crystals with very unusual properties and a solar cell is a mean of storing light.

Besides creating more efficient LED's , which consume less energy and imaging cells in the human body, quantum dots play an important role in creating solar power cells because they have bandgaps that are tunable across a wide range of energy levels by changing the quantum dot size. This is in contrast to bulk materials where the bandgap is fixed by the choice of material composition (e.g. silicon, copper indium gallium selenide – CIGS – or CdTe).



[1]kWh = kilowatt hour; kW_P = kilowatt peak; TWh = terawatt hour; W_P = watt peak; the annual solar yield is the amount of electricity generated by a south-facing 1 kW peak-rated module in 1 year, or the equivalent number of hours that the module operates at peak rating.
[2]Tier 4 and 5 are names of regulated forms of electricity generation and usage.
[3]Unsubsidized cost to end users of solar energy equals cost of conventional electricity.

Source: CIA country files; European Photovoltaic Policy Group; Eurostat; Pacific Gas & Electric (PG&E); Public Policy Institute of New York State; McKinsey Global Institute analysis

Fig.2

In figure 2 we can observe the annual solar energy yield

The potential performance of quantum dot approach has led to widespread research in the field. Recent efforts have also use successive ionic layer absorption and reaction for semiconductor deposition.

Their efficiency of 5.4% is among the highest observed for QDSCs and, although quite low compared to that of commercial bulk silicon cells (about 17%), it has a potential for improvement beyond silicon cells.

Quantum dots can be precisely controlled to do all kinds of useful things. School-level physics tells us that if you give atomenergy, you can "excite" it: you can boost an electron inside it to a higher energy level. When the electron returns to a lower level, the atom emits a photon of light with the same energy that the atom originally absorbed. The color (wavelength and frequency) of light an atom emits depends on what the atom is; iron looks green when you excite its atoms by holding them in a hot flame, while sodium looks yellow, and that's because of the way their energy levels are arranged. The rule is that different atoms give out different colors of light. All this is possible because the energy levels in atoms have set values; in other words, they are **quantized**.

Quantum dots do the same trick (see fig.3)—they also have quantized energy levels—but dots made from the same material (say, silicon) will give out *different* colors of light depending on how big they are. The biggest quantum dots produce the highest wavelengths (and shortest frequencies), while the smallest dots make shorter wavelengths (and higher frequencies); in practice, that means big dots make red light and small dots make blue, with intermediate-sized dots producing green light (and the familiar spectrum of other colors too). The explanation for this is (fairly) simple. A small dot has a bigger **band gap** (crudely speaking, that's the minimum energy it takes to free electrons so they'll carry electricity through a material), so it takes more energy to excite it; because the frequency of emitted light is proportional to the energy, smaller dots with higher energy produce higher frequencies (and lower wavelengths). Larger dots have more (and more closely) spaced energy levels, so they give out lower frequencies (and higher wavelengths).

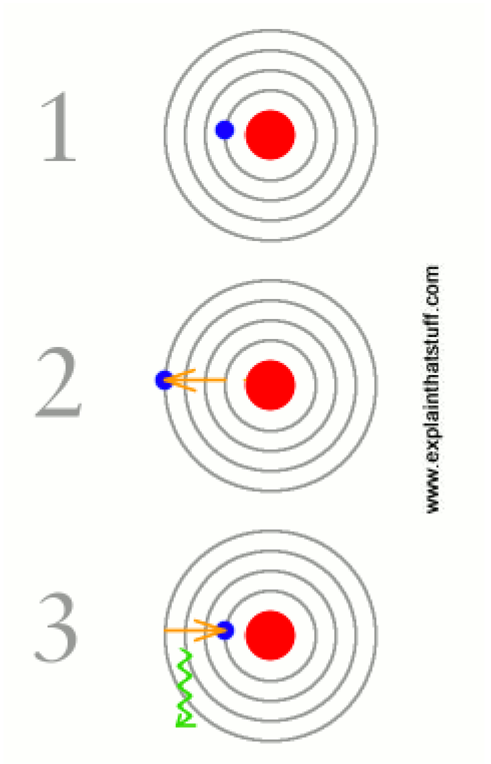


Fig. 3

Figure 3: How atoms make light. After absorbing energy (1), an electron inside an atom is promoted to a higher energy level further from the nucleus (2). When it returns, the energy is given out as a photon of light (3). The color of the light depends on the energy levels and varies from one atom to another. Quantum dots produce light in a similar way because the electrons and holes constrained inside them give them similarly discrete, quantized energy levels. However, the energy levels are governed by the size of the dot rather than the substance from which it's made.

New semiconductor materials for solar cells

Two Dimensional Electron Gas (2DEG)

Switching gears, we turn to how to create something like a 2D particle in a box experimentally. A recent approach is to use semiconductors to trap a layer of free electrons within a thin flat wafer. By combining layers of semiconductor materials that have been treated, or "doped," with additional atoms of different materials, scientists can create a system where electron are free to move in a plane, but are trapped in the third, z dimension. Such as system is called a "two-dimensional electron gas", or "2DEG." These systems have become a wonderful way to study quantum mechanical effects, and as a platform for engineering quantum systems at the nanoscale.

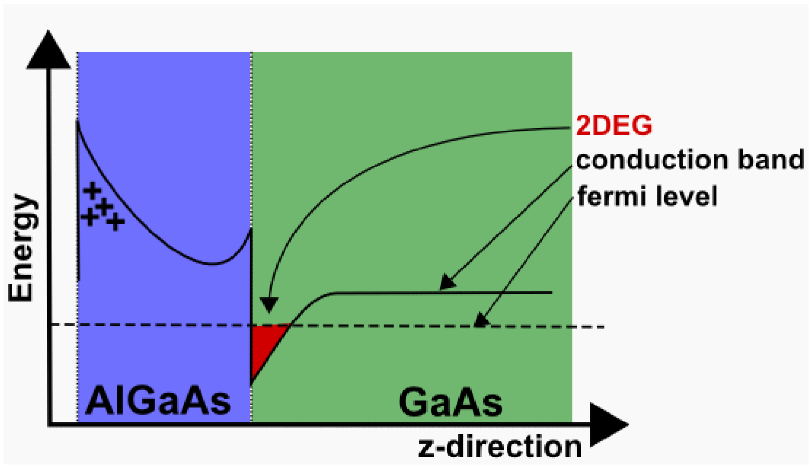
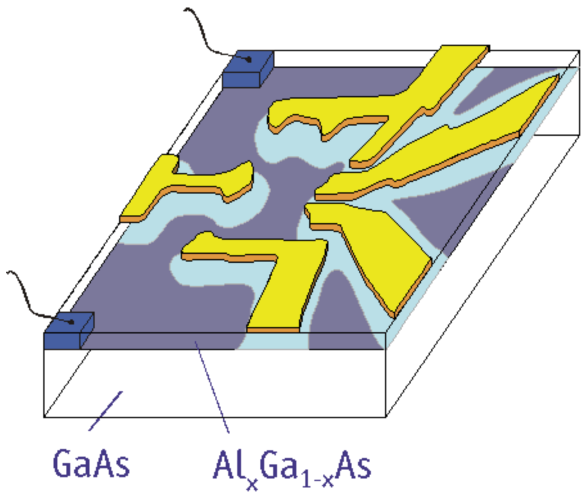


Fig. 4
From http://www.phys.unsw.edu.au/QED/research/2D_scattering.htm and http://meso.seas.harvard.edu/research/spm.html

A wonderful quality of 2DEGs is that within the narrow region that conducts, a free electron moves as if in free space, and all the laws of quantum mechanics hold to a good approximation (as seen in Fig. 4). The only adjustment is that the mass of the electron appears to be significantly less in these systems. For a typical material used creating 2DEGs, Galium Arsenide (GaAs), the electron's effective mass is only 7% of its real mass. Thus is seems lighter and more "quantum" in nature, and quantum effects can be seen on a length scale larger than in free space.

However, for many or most experiements with 2DEGS you can only measure the current flow for electrons that have a certain small band of energies, the Fermi energy, E_F of the system. Thus in these types of experiments, the focus is on the behavior of electrons at this energy, since at other energies the electrons are essentially "invisible" to direct measurement.

Lateral Quantum Dots



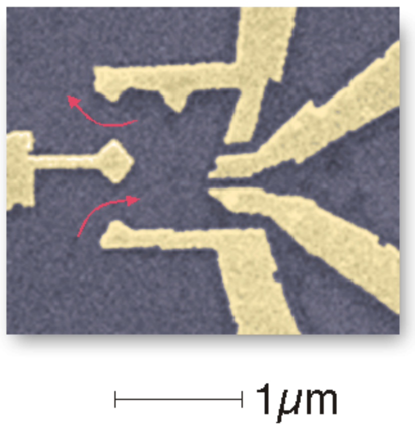


Fig. 5 Fig. 6

From the Marcus Lab, http://marcuslab.harvard.edu/research.shtml.

In order to make these otherwise open system appears like closed systems, scientists apply metal gates and voltage to the surface of the material to hem in the electrons, or at least restrict their flow. An example from the lab of Charlie Marcus at Harvard in pictured above (see fig. 5 and fig. 6).

Thus 2DEGs provide an incredible "quantum playground" for engineering and science.

THEORETICAL MODEL

The Schrodinger Equation

At the heart of modern quantum mechanics is the Schrodinger equation, which relates the energy of a state (or "wavefunction") to a mathematical expression (or "operator") applied to the the wavefunction:

$$\hat{H}\psi = \frac{-\hbar^2}{2m}\nabla^2\psi + V(\vec{r})\psi = E\psi$$

Here the psi is the wavefunction, the first term on the right hand side represents the kinetic energy term, and the second term relates the potential energy of the state. The sum of the two, i.e. H is called the "Hamiltonian," a term drawn from classical mechanics. There is a clear classical analog to this, E = KE(r) + V(r). The quantum version is essentially the generalization of the classical version, taking into account that instead of a point particle, we are dealing with an object "psi" that is spread out in space.

- **The Schrodinger Equation in 1D**

In one dimension, the Schrodinger equation only depends on the coordinate x, and is simplifed as follows:

$$\hat{H}\psi(x) = \frac{-\hbar^2}{2m}\frac{\partial^2}{\partial x^2}\psi(x) + V(x)\psi(x)$$

Now we just need to contend with a single partial derivative of the x coordinate.

- **The Schrodinger Equation in 2D**

In two dimensions, the Schrodinger equation is generalized simply by making the various functions and operators include the y coordinate as well:

$$\hat{H}\psi(\vec{r}) = \frac{-\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2}\psi(\vec{r}) + \frac{\partial^2}{\partial y^2}\psi(\vec{r})\right) + V(\vec{r})\psi(\vec{r})$$

where the r vector indicates both x and y coordinates, r = (x, y).

Numerical simulation

To solve for eigenstates and energies of these types of dots, we'll need to numerically solve the Schrodinger equation. Thus we'll need to somehow numerically perform second derivatives on a grid. The Schrodinger equation very closely resembles wave equations for other fields, such as electromagnetism and acoustics, and as such there has been a great deal of knowledge and study about how to solve such systems.

- **One Dimensional Finite Difference**

For our pedagogic case, we'll choose the simplest and most direct approach, which is to perform a finite difference on a uniform grid. The system is broken up into a mesh (see Annex 2) with each cell the same size, and the derivative terms written as difference between the values in neighboring cells.

The second derivative at some point x_i is hence:

$$\frac{\partial^2}{\partial x^2}\psi(x_i) \approx \left(\frac{\psi(x_{i+1})-\psi(x_i)}{\Delta x} - \frac{\psi(x_i)-\psi(x_{i-1})}{\Delta x}\right) = \frac{1}{\Delta x^2}\left(\psi(x_{i+1}) - 2\psi(x_i) + \psi(x_{i-1})\right)$$

The way to do this derivative to lay out our wavefunction as a vector, and then prepare a matric to multiple by that will do this operation:

$$\frac{\partial^2 \psi}{\partial x^2} = \frac{1}{\Delta x^2}\begin{pmatrix} -2 & 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & & \dots \\ 0 & 1 & -2 & 1 & 0 & \\ \vdots & 0 & 1 & \ddots & & \\ & & & & \ddots & 1 \\ 0 & \dots & & & 1 & -2 \end{pmatrix}\psi$$

Our potential in this representation takes the form of a diagonal matrix, with the values at each point laid at the corresponding point on the diagonal. The end result is a matrix equation as follows, which we solve for eigensolutions:

$$\hat{H}\psi = \begin{pmatrix} -\frac{\hbar^2}{m\Delta x^2}+V_1 & \frac{\hbar^2}{2m\Delta x^2} & 0 & 0 & \dots & 0 \\ \frac{\hbar^2}{2m\Delta x^2} & -\frac{\hbar^2}{m\Delta x^2}+V_2 & \frac{\hbar^2}{2m\Delta x^2} & 0 & \dots & \dots \\ 0 & \frac{\hbar^2}{2m\Delta x^2} & -\frac{\hbar^2}{m\Delta x^2}+V_2 & \frac{\hbar^2}{2m\Delta x^2} & 0 & \vdots \\ \vdots & 0 & \frac{\hbar^2}{2m\Delta x^2} & \ddots & & \\ & & & & \ddots & \\ 0 & \dots & & & & -\frac{\hbar^2}{m\Delta x^2}+V_N \end{pmatrix}\psi$$

This is the finite difference Schrodinger equation in 1D. (see Annex 6)

- **Two Dimensional Finite Difference**

For two dimensions, the same operation is done, except that when we convert the wavefunction, have have to unravel it from a two dimensional grid into a long vector. This operation takes a bit of bookkeeping. (see Annex 5).

Schematically, to take the 2nd derivative, you need to include nearby points in the x and y direction on the grid (fig. 7)

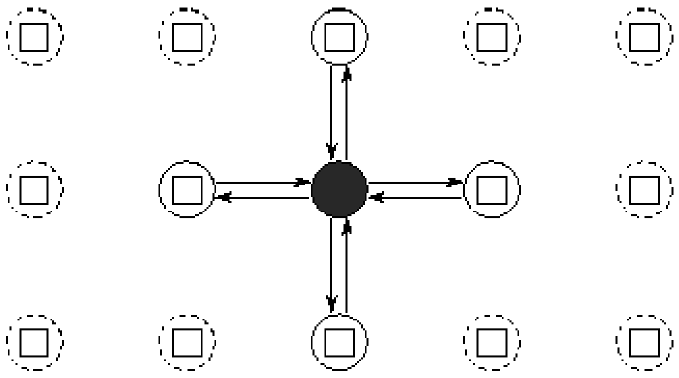


Fig. 7

This is called a "five point finite difference" scheme. The resulting equation in two dimension for the derivative

$$\vec{\nabla}^2\psi(x_i,y_j) \approx \frac{\psi(x_{i+1},y_j) - 2\psi(x_i,y_j) + \psi(x_{i-1},y_j)}{\Delta x^2} + \frac{\psi(x_i,y_{j+1}) - 2\psi(x_i,y_j) + \psi(x_i,y_{j-1})}{\Delta y^2}$$

Now we need to take into account that there are two indices, one for each dimension, but that the matrix H needs to multiply by psi(x,y) on one side. We need to turn our 2D mesh into a single vector psi$_a$ with a single index a. This involves some bookkeeping. In Matlab, the last index of a matrix is the one where the values are closest together in memory, so in that case we would flatten out our x$_i$ and y$_i$ grid so that the index a of a position (i,j) in the *flattened* version of the grid is

$$a = N_y(i-1) + j$$

A schematic picture of how we need to layout the H matrix is as follows:

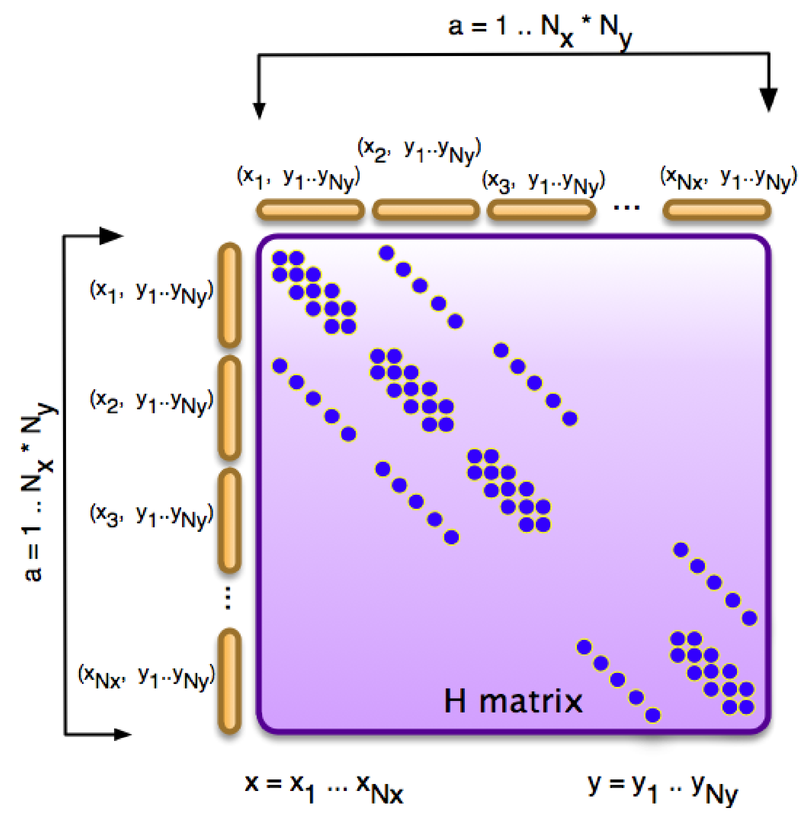


Fig. 8

Here out bigger matrix H (see Annex 7) is of size N_x * N_y, and ranges over the y indices quickly (within the orange segments) and the x index slowly. The blue circles hint at what the resulting matrix will look; each is a non-zero matrix element (as seen in Fig. 8).

The trick now is now to populate H accordingly. The most naive way is to start with an empty H matrix. Then you directly iterate over all a and a' values (along both directions of the H matrix) and calculate i and j as well as i' and j' along the other side of the matrix. For each iteration test for the condition that both i=i' and j=j',j'+1,j'-1 are true, or the same condition which the i's and j's switched. If true, then this element of the matrix will have a non-zero value, and you set it.

Such an approach in order N^4, and slow can be pretty slow for large N. However, it's easy to conceive and implement. An alternate approach that utilizes the symmetry of the H matrix is used below. Note that this matrix, as N grows, is mostly zeros. Because of this, we can solve for larger system (larger N) by using a sparse matrix representation, where instead of storing all the values of the matrix, we only store the non-zero values, along with their indices in the matrix. Sparse matrices are a first tier object in Matlab, and it comes with a suite of routine specifically meant to manipulate sparse matrices and solve sparse systems. This greatly increases the size of the systems we can treat.

Results

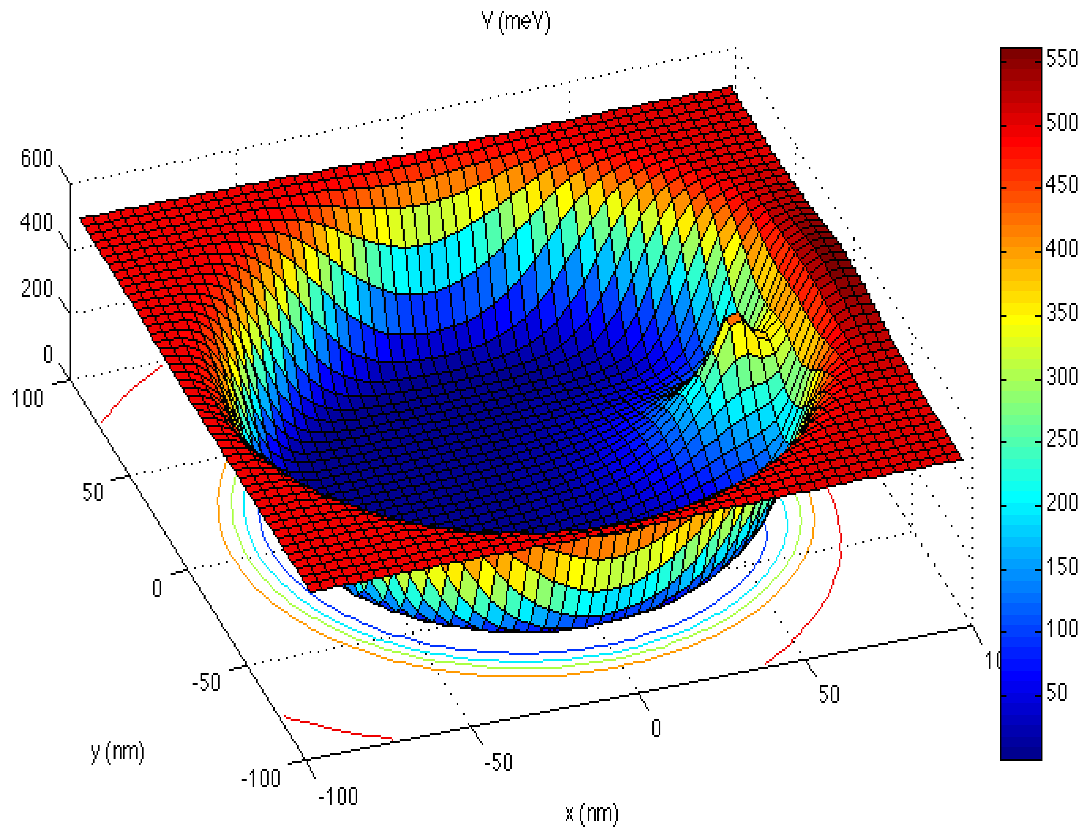


Fig.9

In figure 9 we can see the potential variation across a quantum dot from a GaAs. (see annex 9)

Figures 7 and 8 present the results of the simulation for one and two dimension.

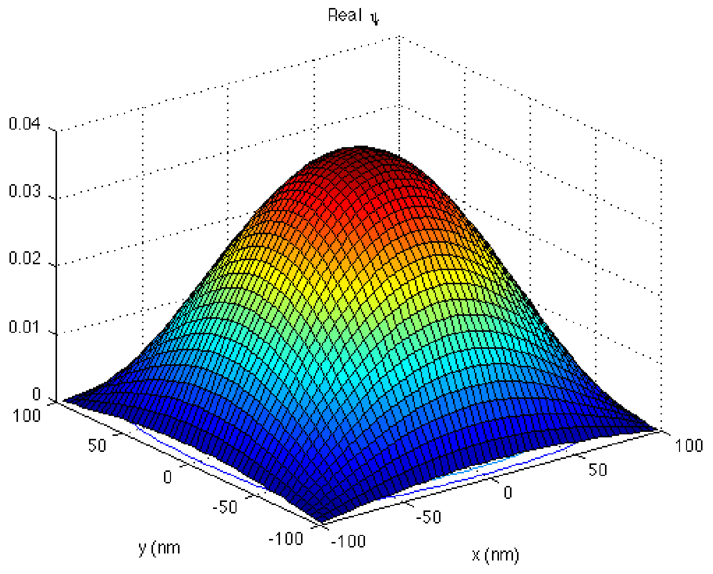


Fig.10

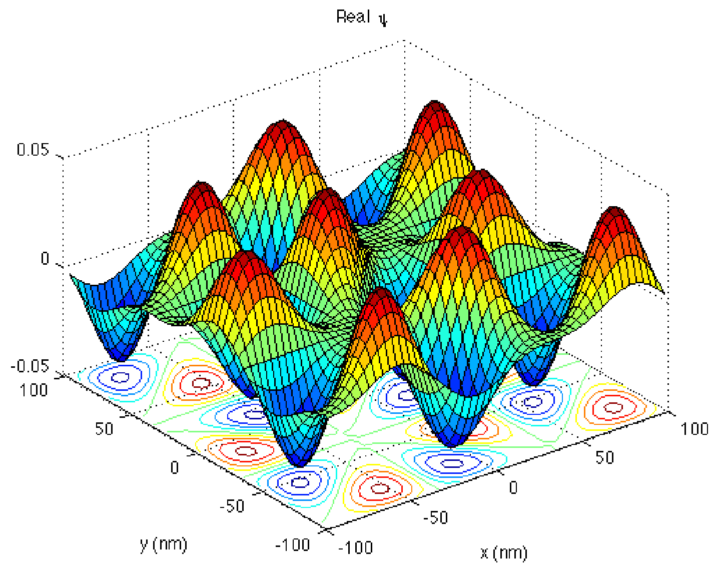


Fig. 11

Fig. 10 and Fig. 11 Solutions of the 2d Particle in a Box system

Conclusion

Solar cells with quantum dots hold promise for low-cost solar cells because they can be made using simple inexpensive chemical reactions. Scientists have calculated that quantum dots could be used to make thin film photovoltaics that are at least as efficient as conventional silicon cells, and possible more efficient. The higher possible efficiency is because nanocrystals made of certain semiconductors can emit more than one electron for every photon absorbed. Plus, tweaking their size and shape changes the colors of light they absorb.

Despite these advantages, no one has succeeded in making efficient quantum-dot solar cells. For that, you need n-type and p-type nanocrystals, says Eran Rabani, a chemistry professor at Tel Aviv University who was involved in the new work. In solar cells, the electrons and holes that are created when photons are absorbed have to be separated so that the electrons can travel out of the semiconductor to the external electric circuit. Some electrons and holes inevitably combine, but they combine much faster in quantum dots than in large silicon crystals. Doping semiconductor nanocrystals would provide a way for creating p-n junctions that separate electrons and holes efficiently.

ANNEX

Annex 1

Let's get the basics out of the way, and setup our various constants and grid as follows:

```
%
%  Matlab code to solve for states of a closed quantum dot in 1D or 2D
%

%
% Some unit conversions
%
au2nm = 5.2918e-2;
nm2au = au2nm^-1;
au2meV   = 27211.6;
meV2au   = au2meV^-1;

% Physical constants
hbar = 1.0;
me   = 0.07;

% Define our parameters
Nx = 50;
Ny = 50;
Nenergy = 10;

xmin = -100 * nm2au;
xmax =  100 * nm2au;
ymin = -100 * nm2au;
ymax =  100 * nm2au;

% Fermi Energy
Ef = 50  * meV2au;
```

Annex 2

Now that we've defined the size of our problem, we want to setup the mesh. To do so, we use the linspace function in Matlab to create an array of uniformaly spaced values from the lower to the upper value.

```
% Define our "space"
Lx = (xmax-xmin);
Ly = (ymax-ymin);
dx = Lx/Nx;
dy = Ly/Ny;
xvec = linspace(xmin+0.5*dx,xmax-0.5*dx,Nx);
yvec = linspace(ymin+0.5*dy,ymax-0.5*dy,Ny);
[ x, y ] = meshgrid(xvec,yvec);
r =  sqrt( x.^2 + y.^2 );
```

Note that, in order to ensure that x_{min} and x_{max} are at the edges of our grid, and that dx = L/Nx, we need to tell linspace to start at the center of the first grid cell, and end at the center of the last grid cell. That's why we trim the values a bit when we put then into linspace.

Once created, we use meshgrid on these vector to create a 2D matrix that runs over these values. This makes it easy to create expressions that look like scalar expressions (i.e. sin(xy)) but that really evaluate the expression over the entire mesh. An example is the expression for r above, which is the distance from the origin on the grid.

Annex 3

Now comes the fun part: setting up our potential. We want to emulate something like a lateral quantum dot, but with a siple expression. To do so, we use a hyperbolic tangent at a certain radius to emulate soft walls of the dot. In addition, we add in a gaussian bump to make the system less symmetric. The expression without the bump is

$$V_{clean}(x,y) = \frac{V_o}{2}\left(1 + \tanh\left(\frac{r-a}{\sigma}\right)\right)$$

The parameter a represents the radius of the dot, while sigma is the "softness" of the wall. This expression drops to very near zero at the center of the dot, and goes to V_o for large r.

The expression for the bump is

$$V_{bump}(x,y) = V_o e^{\left((x-a)^2+y^2\right)/(2\sigma)}$$

Here b is the center of the bump on the x axis, and it has the same width sigma as the walls.

When put together the code looks like

Annex 4

```
% parameters of the potential
sigma =   8 * nm2au;
a     =  80 * nm2au;
b     =  0.8 * a;
Vo    = 500.0 * meV2au;

% Create our potential
Vclean = Vo * 0.5 * ( 1.0 + tanh((r-a)/sigma ));
Vbump1 = Vo * exp ( - sqrt( (x-b).^2 + y.^2 )/(2 * sigma));
V = Vclean + Vbump1;
```

When written this way, the matrix V is evaluated across the entire grid, and can be visuallized easily by using a surface plot (see Fig.1-Annex).

```
surfc(xvec*au2nm,yvec*au2nm,V*au2meV);
colorbar;
title('V (meV)'); xlabel('x (nm)'); ylabel('y (nm)');
```

Annex 5

Now, let's do a full 2D treatment. This involves building up the KE matrix (the derivative part) and then conerting the potential to a sparse 1D diagonal matrix, and adding the two to create the sparse Hamiltonian matrix H.

To begin, we'll want to remove or comment out any code from the 1D optional step from above. Once done, we will first build our KE sparse matrix:

To begin, we'll want to remove or comment out the section from above where we made the problem 1D. Once done, we will build our full 2D Hamiltonian matrix

```
% Setup the Hamiltonian
coeff = -hbar^2/(2*me);
KE  = (2/dx/dx + 2/dy/dy) * coeff * speye(Nx*Ny);
Vsp = sparse(diag(reshape(V, [Nx*Ny,1])));


% work on the y-to-y coupling
ijs = [];
for ix = 1:Nx
   % y-to-y coupling
   is = ( 1:(Ny-1) ) + Ny*(ix-1);
   ijs = [ ijs; [ is'+1 , is' ] ];
end

% include all points flipped across the diagonal
ijs = [ ijs; flipdim(ijs,2) ];
ps  = ijs(:,1) + Nx*Ny*(ijs(:,2)-1);

KE(ps) = -coeff/dy/dy;

% work on the x-to-x coupling
ijs = [];
for iy = 1:Ny
   % x-to-x coupling
   is = ( 1:Ny:((Nx-1)*Ny)) + (iy-1);
   ijs = [ ijs; [ is'+Ny , is' ] ];
end

ijs = [ ijs; flipdim(ijs,2) ];
ps  = ijs(:,1) + Nx*Ny*(ijs(:,2)-1);

KE(ps) = -coeff/dx/dx;
```

We won't go into detail here, but this code is a generalization of the 1D version to 2D.

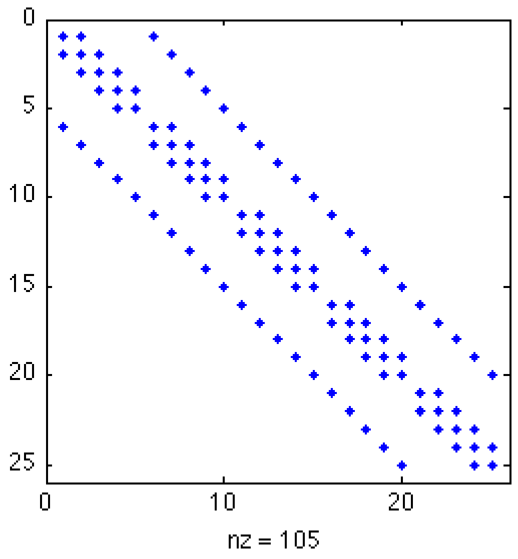
The eye function create an identity matrix (am matrix with the only non-zero elements being ones on the diagonal); here the speye function creates a sparse version of that.

Next we convert our potential on the grid to a 1D vector, and put it on the diagonal of a sparse matrix:

Here we take our Nx by Ny potential on the grid, string it out into a 1D vector of length Nx*Ny, and using the diag function, put it our the diagonal of a matrix Nx * Ny on a side. Then we convert it to sparse form with the sparse command.

Annex 6

Lastly we add them together to form our sparse Hamiltonion matrix. Note how much detail in manipulating these sparse matrices is handled behind the scenes by Matlab. Use the spy command to see the structure of the matrix. Here's an example for a small number of mesh points:



Annex 7

Pseudo-code for populating the H matrix:

```
For i = 1 ..  Nx
 for ip = 1 .. Nx
  for j = 1 ..  Ny
   for jp = 1 .. Ny
    a  = Ny * (i-1)  + j
    ap = Ny * (ip-1) + jp

    if ( i = ip && j = jp  ) H(a,ap) = ...
    if ( i = ip && ( j = jp-1 || j = jp+1) ) H(a,ap) = ...
    if ( j = jp && ( i = ip-1 || i = ip+1) ) H(a,ap) = ...
```

Annex 8

Now that we've setup the matrix, we want to solve the system. We'll do so numerically using the sparse version of the eigensystem solver, eigs. This function, based on routines from the ARPACK library, can function in a variety of ways. In our case, we want to find a number of solutions that are near a target energy value.

Here's the code

```
% (E - H) * Psi = 0
[psi, energy] = eigs(H, Nenergy, Ef);
```

The first argument is the matrix to solve for, in our case the Hamiltonian. The next is the number of solutions requested, and the last is the target energy to find solutions near.

The results are return in a matrix of dimensions (Nx*Ny) by Nenergy, so we can pull out each state solution as

```
ie = 2;
this_psi = reshape(psi(:,ie), [Ny, Nx]);
```

and then we can examine the solution. The eigenvalues energy are the diagonal values of a square matrix Nenergy on a side. You can access these value simply by the index:

```
ie = 2;
this_energy = energy(ie);
```

Annex 9

Once you have a set of solutions, you can access and reshape them each individually to see the structure of the solution:

```
ie = 3;
myPsi = reshape(psi(:,ie), [Ny, Nx])
figure
surfc(xvec*au2nm,yvec*au2nm,real(myPsi))
figure
imagesc(xvec*au2nm,yvec*au2nm,real(myPsi))
```

After some time of looking at your cool results, this gets tiresome, so let's automate the operation by cycling through all our results and plotting them one by one:

```
Figure
for ie = 1:Nenergy
   myPsi = reshape(psi(:,ie), [Ny, Nx])
   subplot(1,2,1)
   imagesc(xvec*au2nm,yvec*au2nm,real(myPsi))
   title('Real \psi')
   subplot(1,2,2)
   imagesc(xvec*au2nm,yvec*au2nm,abs(myPsi))
   title('Magnitude \psi')
   pause
end
```

Bibliography:

- http://en.wikipedia.org/wiki/Quantum_mechanics
- http://en.wikipedia.org/wiki/Particle_in_a_box
- http://en.wikipedia.org/wiki/Wavefunction
- http://en.wikipedia.org/wiki/Mathematical_formulations_of_quantum_mechanics
- Feynman, Richard P.; Leighton, Robert B.; Sands, Matthew (1965). The Feynman Lectures on Physics.
- http://meso.seas.harvard.edu/
- http://marcuslab.harvard.edu/
- http://www-heller.harvard.edu/