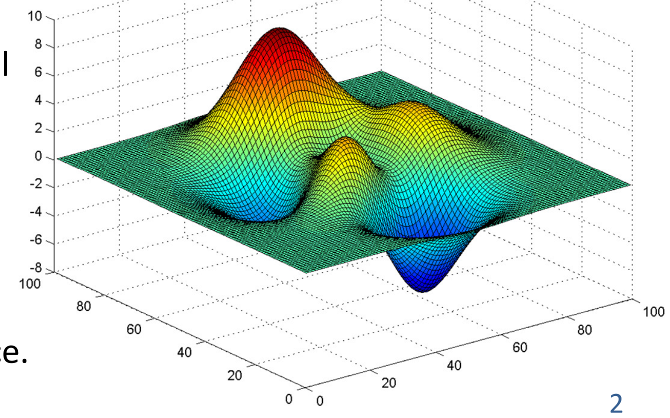
USPAS Fort Collins, June 2013
Design of Electron Storage and Damping Rings

Computer lab:

1. Introduction to Matlab and SAMM
   - Using Matlab to generate a phase space plot
   - Using SAMM to generate a phase space plot
2. Introduction to GUI programming in Matlab
   - Setting up a Graphical User Interface
   - Customising controls
3. **Optimisation routines in Matlab**

USPAS Fort Collins, June 2013
Design of Electron Storage and Damping Rings
Computer Lab 3: Optimisation in Matlab

- In general, an optimisation problem involves finding the minimum (or maximum) of some specified function of a number of variables.
- Optimisation problems occur routinely in accelerator design, e.g. when trying to find quadrupole strengths (and/or positions) to achieve particular values of beta function, phase advance, etc.
- Difficulties frequently occur when the "merit function" is not well-behaved (has many local minima, discontinuities, or long narrow valleys).
- We will discuss how to apply the built-in optimisation routines in Matlab to tuning a FODO cell to given values of the phase advance.



USPAS Fort Collins, June 2013
Design of Electron Storage and Damping Rings
Computer Lab 3: Optimisation in Matlab

1. First, define a function to calculate the tunes in a FODO cell:

```
function [nux, nuy] = FODOTunes(k1LQF, k1LQD)

    dLen = 1.0;

    tmQF = [   1      0    0      0 ;...
            -k1LQF   1    0      0 ;...
              0      0    1      0 ;...
              0      0   k1LQF   1];

    tmQD = [   1      0    0      0 ;...
            -k1LQD   1    0      0 ;...
              0      0    1      0 ;...
              0      0   k1LQD   1];

    tmD  = [   1  dLen  0   0    ;...
               0   1    0   0    ;...
               0   0    1  dLen  ;...
               0   0    0   1];

    tmFODO = tmD*tmQD*tmD*tmQF;

    nux = acos((tmFODO(1,1) + tmFODO(2,2))/2)/2/pi;
    nuy = acos((tmFODO(3,3) + tmFODO(4,4))/2)/2/pi;

end
```

USPAS Fort Collins, June 2013
Design of Electron Storage and Damping Rings
Computer Lab 3: Optimisation in Matlab

2. Next, define a merit function (this has a very general form):

```
function f = MeritFunction(x, a)

    [nux, nuy] = FODOTunes(x(1), x(2));

    f = (nux - a(1))^2 + (nuy - a(2))^2;

end
```

The merit function has a minimum (zero) when the quadrupole strengths in the variable “x” (an array) are such that the tunes in the FODO cell are equal to the constraints specified in “a” (also an array).

Note that in this case, the number of variables (two) is equal to the number of constraints (two): this is often advisable in accelerator design problems.

USPAS Fort Collins, June 2013
Design of Electron Storage and Damping Rings
Computer Lab 3: Optimisation in Matlab

3. Finally, we can use the Matlab function "fminsearch" to look for the values of the variables that minimise the merit function.

   At the Matlab command line, enter:

```
>> tuneTarget = [0.22,0.24];
>> f1 = fminsearch(@(x)MeritFunction(x,tuneTarget),[1.2,-1.2])
```

   This should produce the result:

```
f1 =

    1.2920   -1.3543
```

   To check that this is correct, enter:

```
>> [nux, nuy] = FODOTunes(1.2920, -1.3543)
```

USPAS Fort Collins, June 2013
Design of Electron Storage and Damping Rings
Computer Lab 3: Optimisation in Matlab

The syntax of fminsearch in Matlab is:

```
minimising_value = fminsearch( function_handle , initial_value )
```

- "initial_value" is the starting point for the search.
- "function_handle" is the *handle* of the function we want to minimise.
- "minimising_value" is the value that minimises the function.

A *function handle* is a variable associated with a particular function.

In Matlab, the operator "@" returns the handle of a function. For example:

```
>> g = @sin;
>> g(pi/2)

ans =

      1
```

USPAS Fort Collins, June 2013
Design of Electron Storage and Damping Rings
Computer Lab 3: Optimisation in Matlab

In Step 3 (on slide 5) we could have used the following commands:

```
>> tuneTarget = [0.22,0.24];
>> mf = @(x)MeritFunction(x,tuneTarget);
>> f1 = fminsearch(mf,[1.2,-1.2])
```

- Here, we define a function handle “mf” that is associated with the function “MeritFunction”...
- ...but “mf” takes only a single parameter: the value of “tuneTarget” is fixed, and different values of “tuneTarget” cannot be passed to “mf”.
- The fact that “mf” takes only a single parameter means that there is no ambiguity about which parameter to vary when we call fminsearch.

The syntax on slide 5 is simply a shorthand that avoids making an explicit definition of a function handle (that we will only use when calling fminsearch).

USPAS Fort Collins, June 2013
Design of Electron Storage and Damping Rings
Computer Lab 3: Optimisation in Matlab

It is possible to specify options in fminsearch, for example to set a tolerance on the minimum, or to limit the number of iterations.

Usually, it is wise to avoid an over-reliance on sophisticated optimisation algorithms that claim to deal with large numbers of variables and constraints:

- The simpler you can make the problem, the more efficient (and accurate) the optimisation will be: try to break large, complicated problems up into smaller, simpler pieces.
- When applying an optimiser to a beamline design problem, first try to satisfy yourself that a solution really exists (e.g. don’t try and tune a FODO cell for a phase advance greater than π): lattice optimisation should be the **final step** in a design process based on a good understanding of beam optics.

USPAS Fort Collins, June 2013
Design of Electron Storage and Damping Rings
Computer Lab 3: Optimisation in Matlab

Exercises for the student:

A. Modify the code in the previous slides to match the peak horizontal and vertical beta functions in a FODO cell (instead of the horizontal and vertical tunes).

B. Extend your optimisation routines so that you can match to specified values of the horizontal and vertical tunes, *and* the peak horizontal and vertical beta functions.
Hint: There are now four constraints. What are the appropriate variables?

C. Develop a GUI for design of a double-bend achromat cell, where the user can specify:

- dipole bending angle (i.e. number of cells in the complete ring);
- horizontal and vertical tunes;
- beta functions at the exit of the cell.

The code should tune the sextupoles for zero chromaticity, and calculate significant lattice parameters (natural emittance, damping times...)