## cueBeam: acoustic beam simulation using Huygens's principle.

cueBeam is a Matlab/CUDA code for simulation of acoustic field pressure distribution. It calculates pressure in frequency domain (meaning, assumes that the radiators continuously radiate) using the Huygen's principle.

Radiators are described in terms of spatial XYZ location, amplitude and phase; despite them being point-like, they physically represent perfectly baffled pistons.

Propagation medium is described in terms of wavenumber k. Radiating sources are treated as point like. For each field probing point, value is computed by summing the contributions from each transmitter, shifted by appropriate phase delay and reduced by respective distance between the transmitter and probed point:

for each pixel

pressure=complex zero;

for each radiator

distance=distance(radiator,pixel);

phase\_shift=wavenumber\*distance+radiator\_phaseshift;

amplitdue\_decayed=radiator\_amplitude/distance;

pressure=pressure+...

complex(amplitdue\_decayed,phase\_shift);

end for each radiator

out(pixel)=abs(pressure); % store absolute value

end for each pixel

The following inputs are required:

* locations of the centre of radiating elements -x,y,z. These should be packed into a n\*6 matrix:

tx=[elem\_x elem\_y elem\_z zeros() abs(SVect(:)) angle(SVect(:))];

SVect is a 'steering vector' - a complex number describing amplitude and phase of radiation for each element. The size of the SVect must be n\*1

* wavenumber k in the medium (single frequency only)
* description of the location of the field probing points. There are two versions: XZ and Lambert.

For XZ the probing points (where pressure is calculated) is always a regular grid described by: (x0,y0,z0) - location of a corner of the grid; (dx,dy,dz) - distance between points, and (nx,ny,nz) : number of points in each direction. ny=1 always in this implementation.

The way to call the calculation function is:

cueBeam\_xz(tx',k,x0,y0,z0,nx,ny,nz,dx,dy,dz);

Note that all inputs must be of class single. (default for Matlab is double, so conversion is needed). This is both for speed and compatibility with early CUDA cards. No ill conditioned math is involved so single precision numbers deliver approximately 80dB of precision.

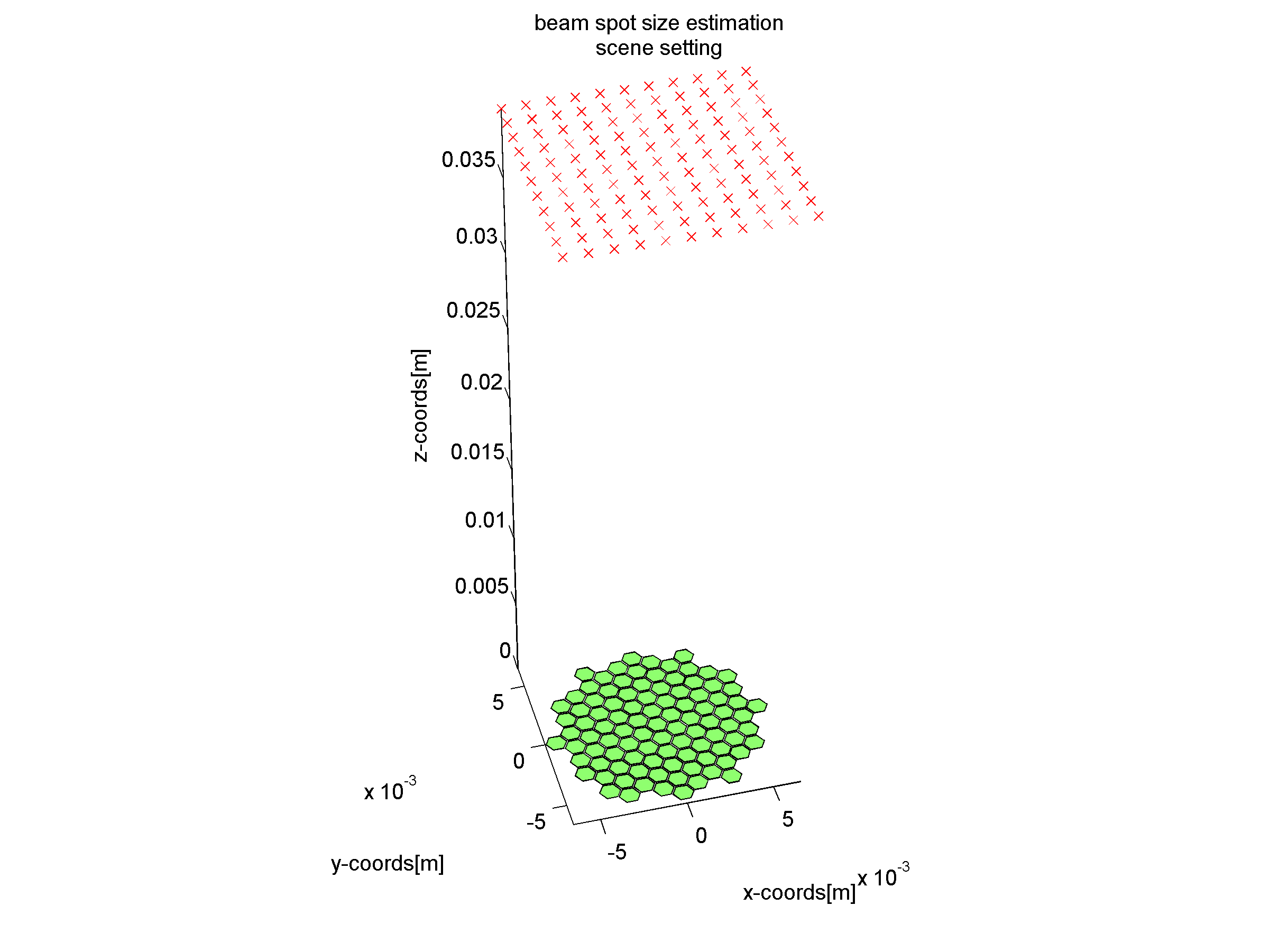


Figure 1. Scene setting for cueBEAM. Green: element locations(actually, points); red cross: "probe points" where field is calculated. Note that this is an old version of the figure and the default probing points are XZ only.

For beamsim Lambert the calling convention is simpler:

[img\_lambert lambert\_x lambert\_y lambert\_z]=cueBeam.cueBeam\_lambert

(tx', k, r, density);

This program automatically generates a mesh of points that are distributed over a hemisphere with radius R, and distance between real points "density", in such way, that the true area covered by a given point is equal for all points. The points are then mapped to a rectangle. The transformation rules are (after http://mathworld.wolfram.com/LambertAzimuthalEqual-AreaProjection.html )

Inverse transformation rules are used to calculate standard parallel phi_1 and central longitude on the sphere:

rho=sqrt(x^2+y^2)

c=2sin^(-1)(1/2rho).

phi=sin^(-1)(coscsinphi_1+(ysinccosphi_1)/rho)

lambda=lambda_0+tan^(-1)((xsinc)/(rhocosphi_1cosc-ysinphi_1sinc)),

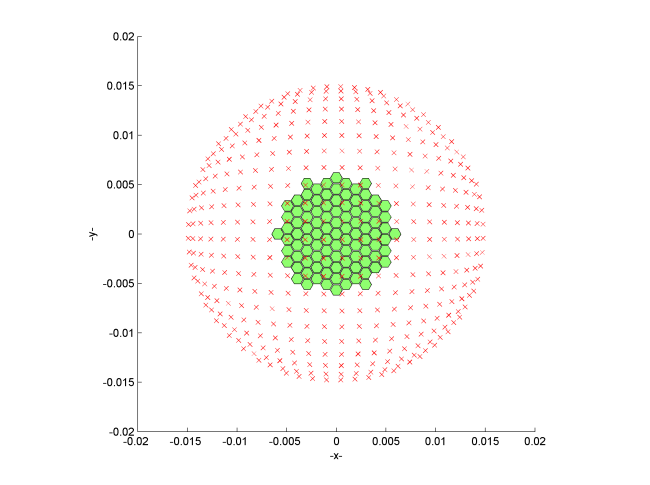
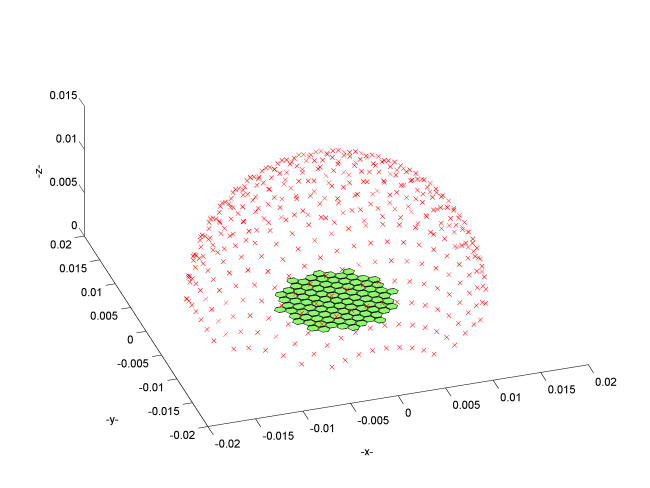


Figure 2. Lambert azimuthal equiareal map field probing point distribution. Left: perspective view. Right: top-down view.

Where r is the radius of the l'ambert sphere, and density is the distance between points on the l'ambert sphere. The resulting resolution of the field image can be calculated by:

number\_of\_points=ceil(2\*pi\*r/density);

d=2\*sqrt(2)/number\_of\_points;

n=ceil(2\*sqrt(2)/d); % note, the double scaling is needed to obtain correct rounding.

img\_lambert=zeros(n,n,'single');

When compared to regular orthogonal grid, the advantage of this approach is that true power of the sidelobes can be easily integrated and compared with the power of the main lobe. This reduces error of estimating the sidelobe level for classic beamforming, and therefore is more representative of the image contrast from the operator's point of view.

An example on how to define parameters for use with cueBEAM are provided in the MATLAB file:

cueBeam\_linear\_array\_basic