



k-Wave

**A MATLAB toolbox for the time-domain
simulation of acoustic wave fields**

Felipe Wilker Grillo

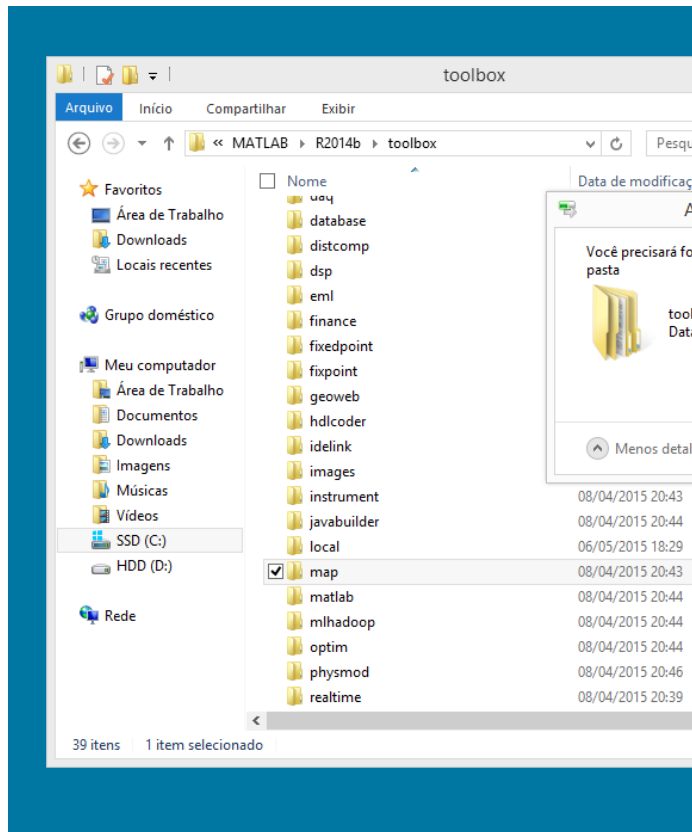
Monitor PAE- 2016

Potencial do k-Wave

- O software foi desenvolvido para simular a propagação de ondas e sistemas fotoacústicos em 1D, 2D ou 3D
- Exemplos de aplicações:
 - Propagação em meios homogêneos e heterogêneos,
 - funcionamento de transdutores comerciais e de formatos diversos,
 - efeito Doppler,
 - Difração, refração e reflexão
 - Fotoacústica, Modo-B etc

Instalação do k-Wave

- Download do arquivo
- Salvar pasta inteira em ./Matlab/R20XX/toolbox





k-Wave

A MATLAB toolbox for the time-domain simulation of acoustic wave fields

[home](#)
[download](#)
[installation](#)
[license](#)
[publications](#)
[documentation](#)
[forum](#)



FREE acoustics toolbox for MATLAB

Features

k-Wave is an [open source](#) acoustics toolbox for [MATLAB](#) and C++ developed by [Bradley Treeby](#) and [Ben Cox](#) (University College London) and [Jiri Jaros](#) (Brno University of Technology). The software is designed for time domain acoustic and ultrasound simulations in complex and tissue-realistic media. The simulation functions are based on the k-space pseudospectral method and are both fast and easy to use. The toolbox includes:

- 1 An advanced time-domain model of acoustic wave propagation that can account for nonlinearity, acoustic heterogeneities, and power law absorption (1D, 2D, and 3D)
- 2 The ability to model pressure and velocity sources, including photoacoustic sources, and diagnostic and therapeutic ultrasound transducers
- 3 The ability to specify arbitrary detection surfaces with directional elements, with options to record acoustic pressure, particle velocity, and acoustic intensity
- 4 An optimised C++ version of the code that maximises computational performance for large simulations
- 5 The option to use the forward model as a flexible time reversal image reconstruction algorithm for photoacoustic tomography with an arbitrary measurement surface
- 6 A fast, one-step, photoacoustic image reconstruction algorithm for data recorded on a linear (2D) or planar (3D) measurement surface
- 7 Optional input parameters to adjust visualisation and performance, including options to generate movies and to run the simulations on a graphics processing unit (GPU)
- 8 An extensive user manual and many simple to follow tutorial examples to illustrate the capabilities of the toolbox



Latest News



27th August 2016
An updated version of the user manual for k-Wave V1.1 is now available.



18th August 2016
The native C++/CUDA code for graphics processing units (GPUs) is now available for download.



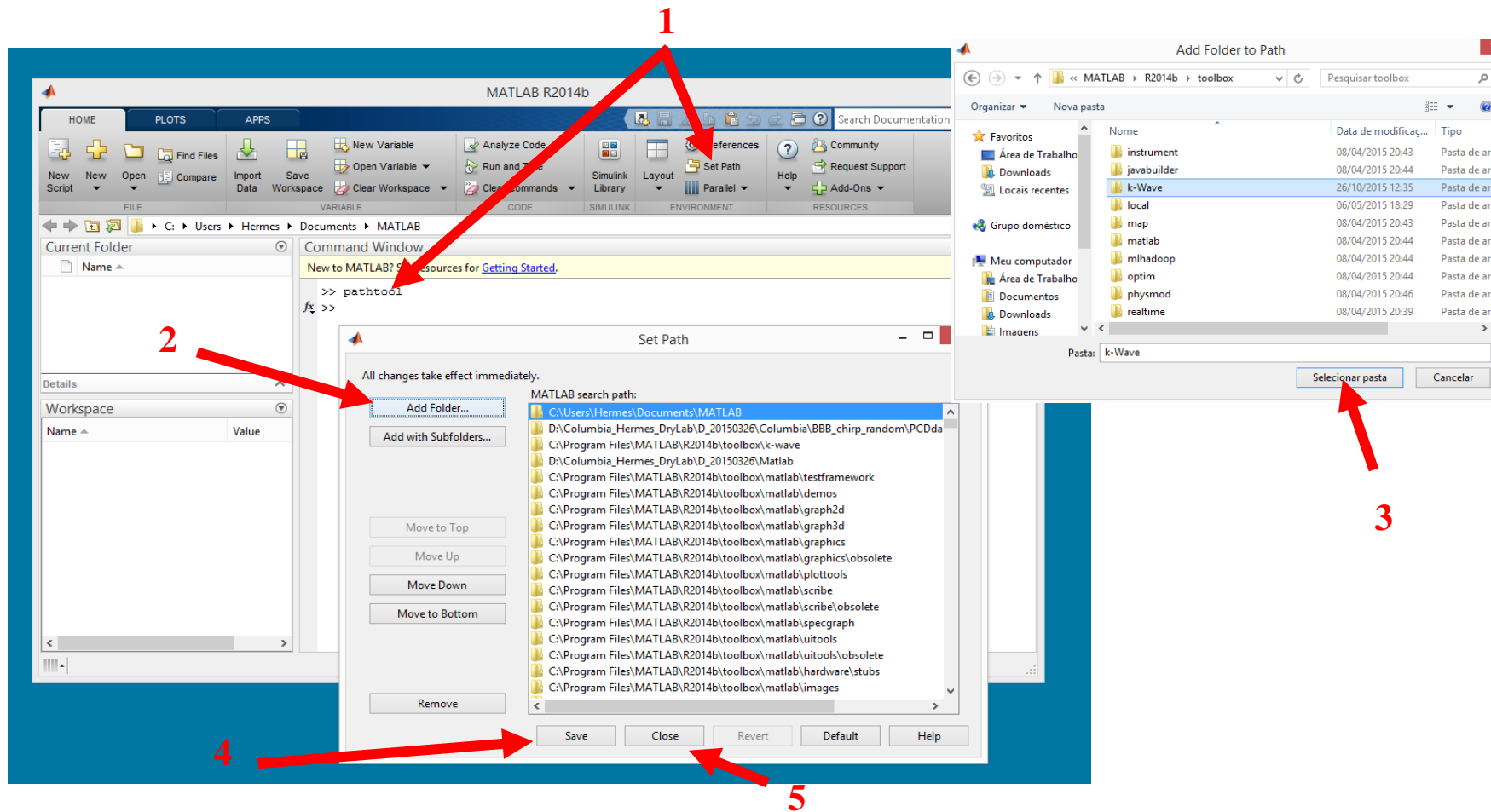
16th November 2015
The online content for supercomputing 2015 can be found [here](#).



9th October 2015
k-Wave Version 1.1.1 (bug fix update) is now available for download.

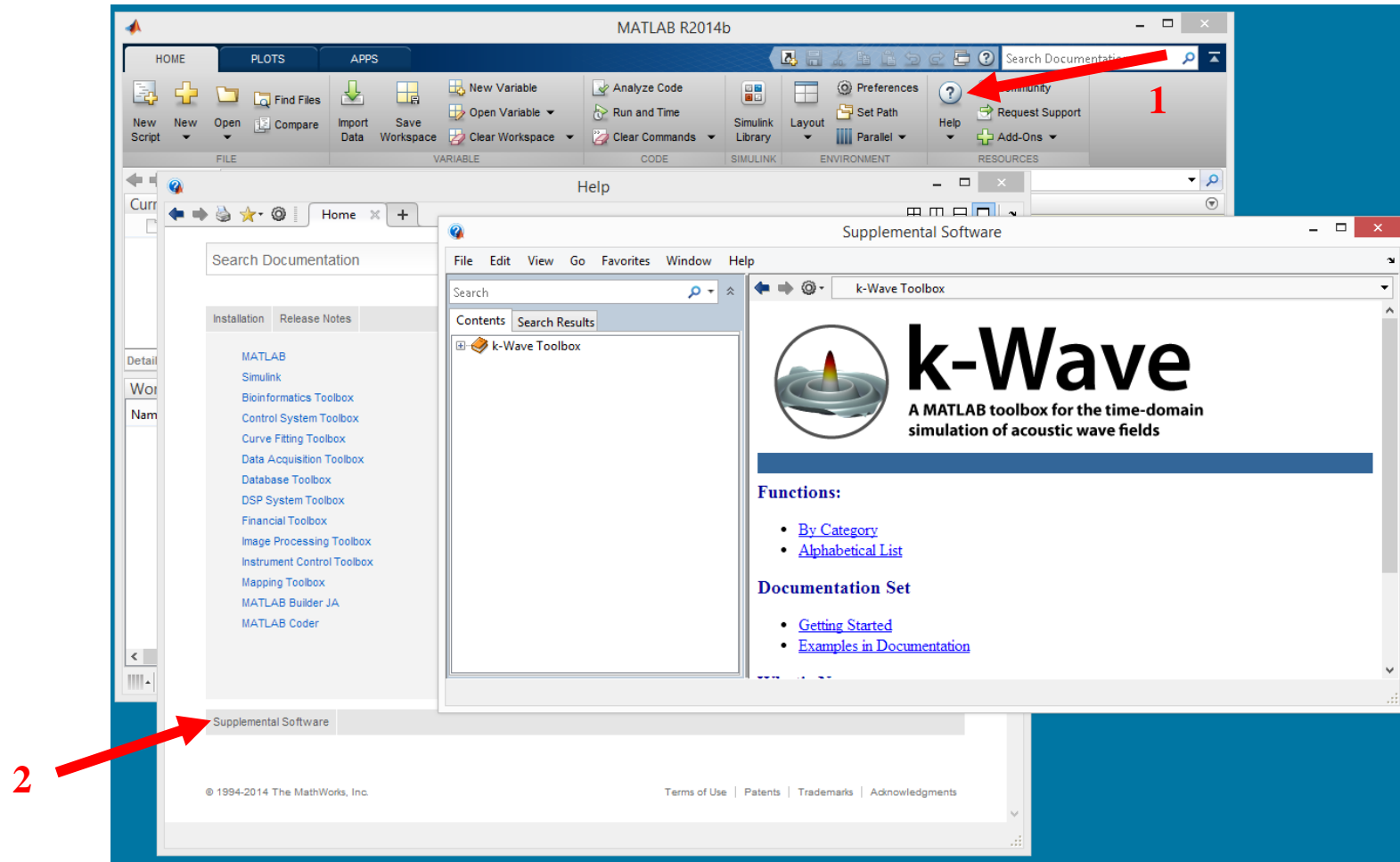
Instalação do k-Wave

Abrir Matlab e incluir caminho para toolbox <File><Set Path><Add Folder><Save>



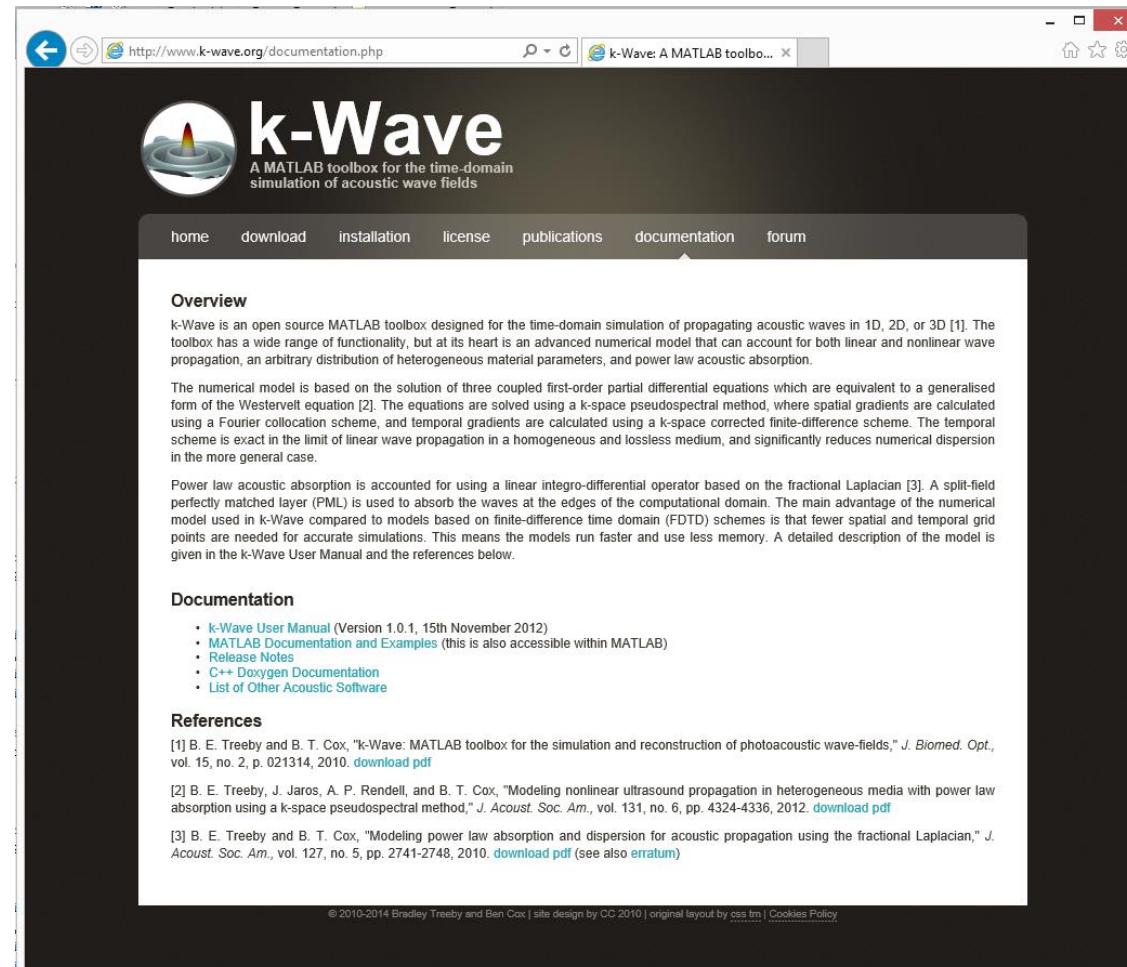
Instalação do k-Wave

Reiniciar Matlab e abrir Help para ver se k-wave foi indexado na busca



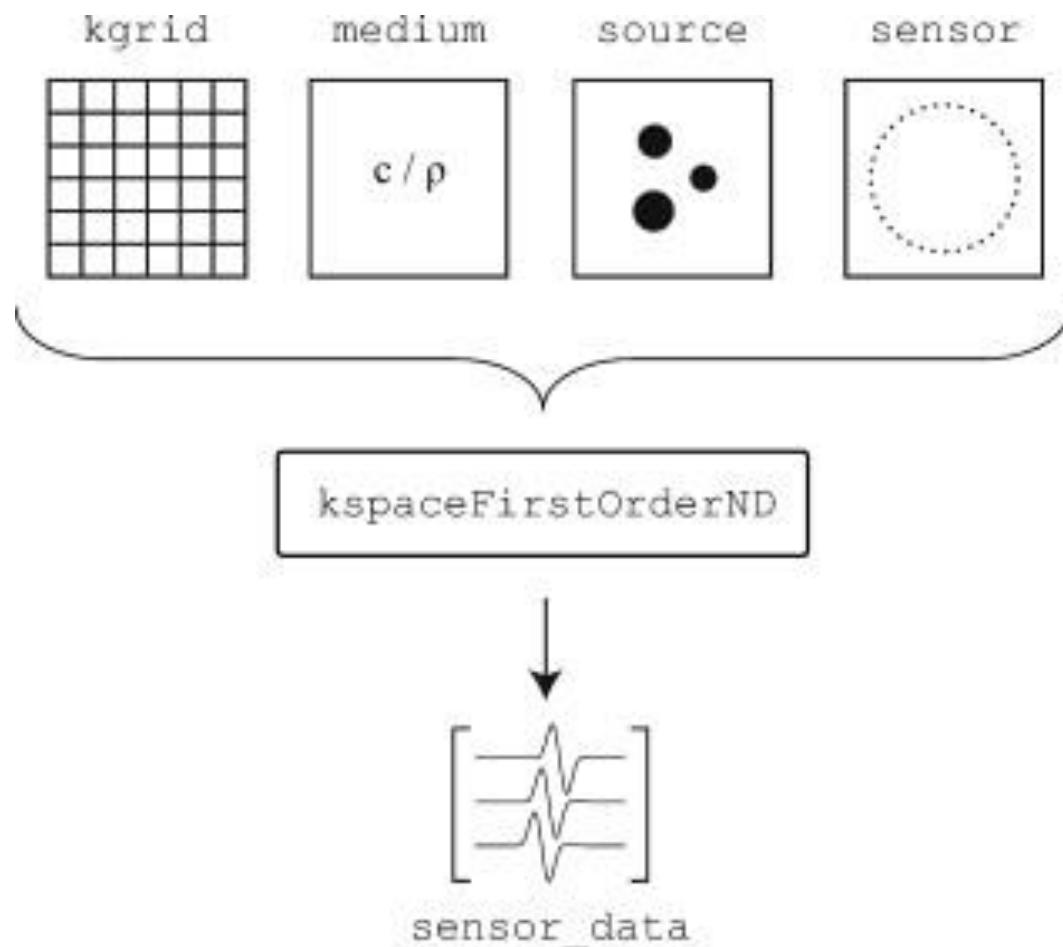
Instalação do k-Wave

• Informações também disponíveis online



Algoritmo básico

- 1 - Definição:
 - dos parâmetros do espaço a ser calculado
 - dos Parâmetros do meio de propagação
 - da fonte de ultrassom
 - do sensor
- 2 - Rodar simulação
- 3 - Visualizar resultados



Algoritmo básico para utilização do K-Wave

Criação do seu ambiente
de simulação (GRID)

```
% create the computational grid
Nx = 128;           % number of grid points in the x (row) direction
Ny = 128;           % number of grid points in the y (column) direction
dx = 0.1e-3;        % grid point spacing in the x direction [m]
dy = 0.1e-3;        % grid point spacing in the y direction [m]
kgrid = makeGrid(Nx, dx, Ny, dy);
```

Definir os parâmetros do
meio

```
% define the properties of the propagation medium
medium.sound_speed = 1500;      % [m/s]
medium.alpha_coeff = 0.75;      % [dB/(MHz^y cm)]
medium.alpha_power = 1.5;
```

Criando um estímulo

```
% create initial pressure distribution using makeDisc
disc_magnitude = 5;           % [Pa]
disc_x_pos = 50;               % [grid points]
disc_y_pos = 50;               % [grid points]
disc_radius = 8;               % [grid points]
disc_1 = disc_magnitude*makeDisc(Nx, Ny, disc_x_pos, disc_y_pos, disc_radius);

disc_magnitude = 3;           % [Pa]
disc_x_pos = 80;               % [grid points]
disc_y_pos = 60;               % [grid points]
disc_radius = 5;               % [grid points]
disc_2 = disc_magnitude*makeDisc(Nx, Ny, disc_x_pos, disc_y_pos, disc_radius);

source.p0 = disc_1 + disc_2;
```


Algoritmo para utilização do K-Wave

Definir sensor de leitura

```
% define a centered circular sensor
```

```
sensor_radius = 4e-3; % [m]
```

```
num_sensor_points = 50;
```

```
sensor.mask = makeCartCircle(sensor_radius, num_sensor_points);
```

Executar a simulação

```
% run the simulation
```

```
sensor_data = kspaceFirstOrder2D(kgrid, medium, source, sensor);
```

```
% plot the simulated sensor data
```

```
figure;
```

```
imagesc(sensor_data, [-1, 1]);
```

```
colormap(getColorMap);
```

```
ylabel('Sensor Position');
```

```
xlabel('Time Step');
```

```
colorbar;
```

Algoritmo para utilização do K-Wave

Adicionando a variável
tempo

```
% define a single source point
```

```
source.p_mask = zeros(Nx, Ny);
```

```
source.p_mask(end - Nx/4, Ny/2) = 1;
```

```
% define a time varying sinusoidal source
```

```
source_freq = 0.25e6; % [Hz]
```

```
source_mag = 2; % [Pa]
```

```
source.p = source_mag*sin(2*pi*source_freq*kgrid.t_array);
```

```
% define the acoustic parameters to record
```

```
sensor.record = {'p', 'p_final'};
```

```
% run the simulation
```

```
sensor_data = kspaceFirstOrder2D(kgrid, medium, source, sensor);
```

Exemplo de criação de um transdutor ultrassônico

```
% define properties of the input signal
```

```
source_strength = 1e6;           % [MPa]
```

```
tone_burst_freq = 0.5e6;    % [Hz]
```

```
tone_burst_cycles = 5; % create the input signal
```

using toneBurst

```
input_signal      =      toneBurst(1/kgrid.dt,      tone_burst_freq,
tone_burst_cycles);
```

% scale the source magnitude by the source_strength divided by the

% impedance (the source is assigned to the particle velocity)

input signal =

Exemplo de criação de um transdutor ultrassônico

% physical properties of the transducer

```
transducer.number_elements = 72;    % total number of transducer elements
transducer.element_width = 1;        % width of each element [grid points]
transducer.element_length = 12;      % length of each element [grid points]
transducer.element_spacing = 0;      % spacing (kerf width) between the elements [grid points]
transducer.radius = inf;              % radius of curvature of the transducer [m]
```

% calculate the width of the transducer in grid points

```
transducer_width = transducer.number_elements*transducer.element_width +
(transducer.number_elements - 1)*transducer.element_spacing;
```

% use this to position the transducer in the middle of the computational grid

```
transducer.position = round([1, Ny/2 - transducer_width/2, Nz/2 - transducer.element_length/2]);
```

% properties used to derive the beamforming delays

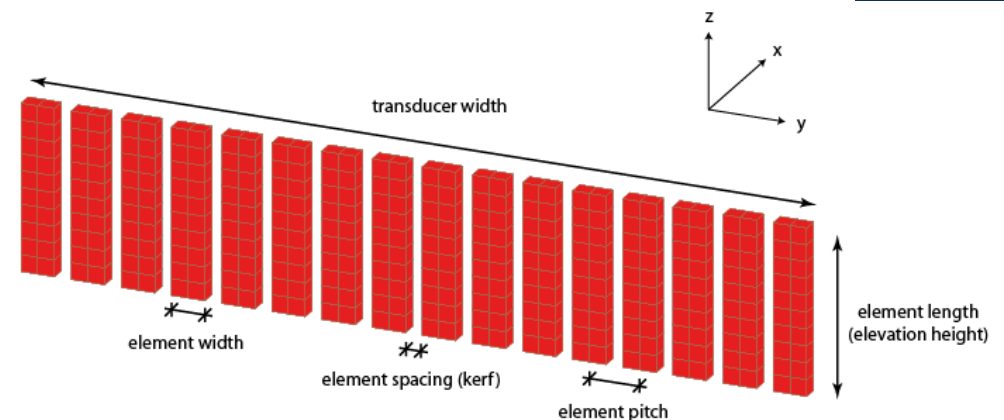
```
transducer.sound_speed = 1540;        % sound speed [m/s]
transducer.focus_distance = 20e-3;     % focus distance [m]
transducer.elevation_focus_distance = 19e-3; % focus distance in the elevation plane [m]
transducer.steering_angle = 0;         % steering angle [degrees]
```

% apodization

```
transducer.transmit_apodization = 'Rectangular';
transducer.receive_apodization = 'Rectangular';
```

% define the transducer elements that are currently active

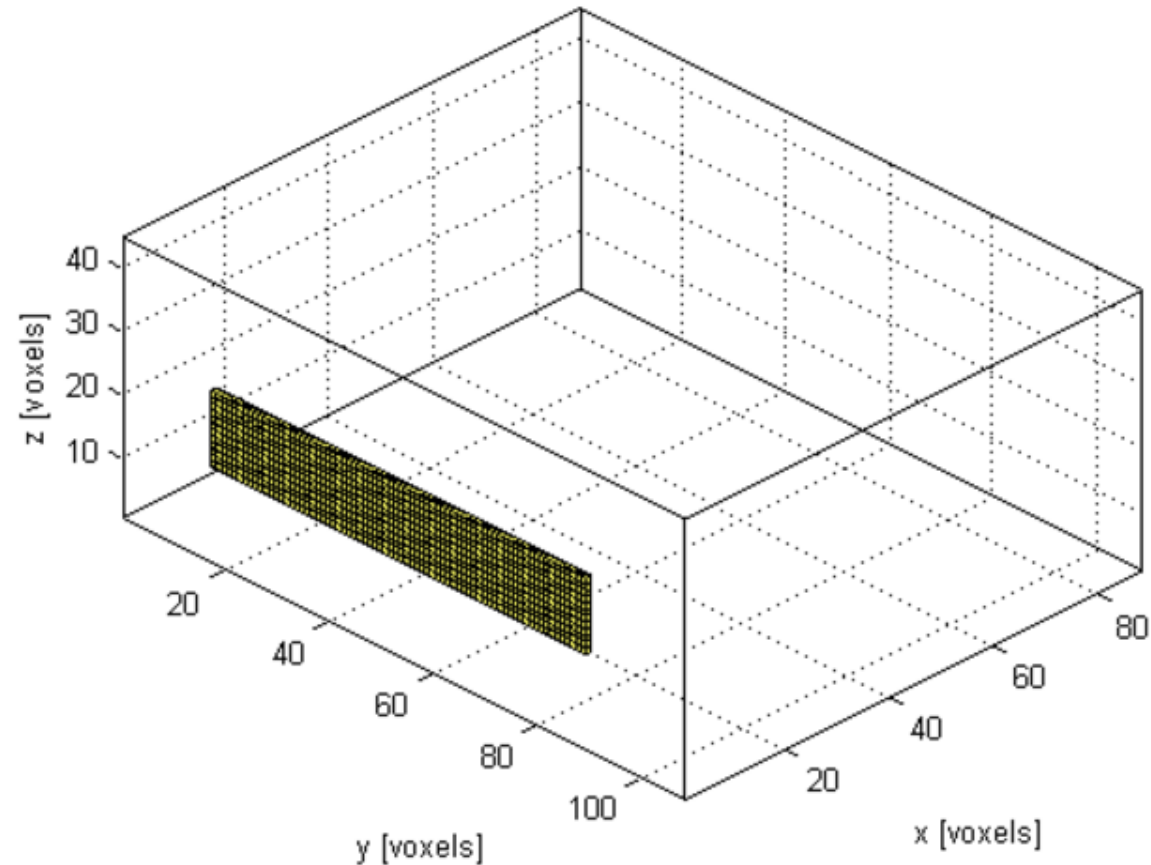
```
transducer.active_elements = zeros(transducer.number_elements, 1);
transducer.active_elements(21:52) = 1;
```



Exemplo de criação de um transdutor ultrassônico

Definindo o transdutor – Função
makeTransducer

```
% append input signal used to drive the transducer  
transducer.input_signal = input_signal;  
% create the transducer using the defined settings  
transducer = makeTransducer(kgrid, transducer);
```



Exemplo de criação de um transdutor ultrassônico

Executando a simulação

% run the simulation

```
[sensor_data] = kspaceFirstOrder3D(kgrid, medium, transducer, sensor, input_args{:})
```

