HITU_Simulator v2.0 User's Manual

Joshua Soneson

February 13, 2019

This software package was developed at the
US Food and Drug Administration.

Terms and Conditions

In exchange for this free software, the author requests that he and the FDA are acknowledged any time the results of this software or its derivatives are included in a publication, report, presentation, etc. In the interest of improving the package, the author greatly appreciates user feedback.

This package is still under development and has not been exhaustively tested and validated. Therefore no guarantee of the accuracy and/or reliability of the results is expressed or implied.

Official FDA Disclaimer

Contents

1 Changes to Version 2.0

The following updates have been made to the current version:
• Most of the original code was rewritten in a simpler way to facilitate user modification.
• This version can simulate a broader range of axisymmetric fields including shallowly focused (low f-number), unfocused, and diverging beams.
• User now defines location of the "effective source" for spherical cap tranducers.
• User defines the fraction of pressure attenuation due to absorption.
• Provisions for arbitrarily many tissue layers.
• Phase-velocity dispersion is now calcuated exactly for power-law attenuating media with exponent between 0 and 2.
• Option for user-defined source pressure distribution.
• Automatic discretization accounts for smaller scales introduced by nonlinear effects.
• Option for spatially-averaging the field to mimic hydrophone measurements; user defines hydrophone element diameter.
• Provisions for complex treatment protocols (pulsed sonications) have been omitted.

2 What HITU_Simulator does

HITU_Simulator predicts many important characteristics of continuous wave, high-intensity therapeutic ultrasound (HITU) beams and their heating effects. This is done by integrating a high-order parabolic approximation of the axisymmetric Westervelt equation, A.K.A. the wide-angle Khokhlov-Zabolotkaya-Kuznetsov (WAKZK) equation, from the frequency-domain perspective. This results in a spatial distribution of pressure of each harmonic, taking into account beam diffraction, interference effects, power-law frequency-dependence of attenuation and the corresponding phase velocity dispersion, the fraction of energy loss that is converted to heat, the nonlinear effects of higher harmonic generation/wavefront steepening, and the corresponding augmented heat generation due to shocked waveforms. From these pressure fields the temporal average intensity and power density are calculated. The power density is then used as a source for the bioheat transfer (BHT) equation, which is integrated to determine the temperature and thermal dose fields. Efforts have been made to keep the software fast and light on system resources.

2.1 Model equations

The propagation equation is a high-order (wide-angle) parabolic approximation of the generalized one-way Westervelt equation:

$$\frac{\partial^2 p}{\partial t^2} - c^2\nabla^2 p + 2c\frac{\partial}{\partial t}\left[(\alpha(\omega) * p(\omega)\right] = \frac{\beta}{\rho c^2}\frac{\partial^2 p^2}{\partial t^2},$$

where p (Pa) is pressure, t is time (s), c (m/s) is the small-signal sound speed, ∇^2 (cm^{-2}) is the Laplacian in cylindrical coordinates, α (cm^{-1}) is the attenuation/dispersion function, ω is angular frequency (rad/s), β (dimensionless) is the nonlinear parameter, and ρ ($\mathrm{kg/m}^3$)

is mass density. The wide-angle parabolic approximation results in a one-way wave equation, so scattering and reflection is not taken into account. It results in a more tractible model which accurately represents off-axis propagation to about 45°, compared to about 20° for the standard parabolic approximation used to obtain the KZK equation. For layered media, the transmission coefficient is calculated at interfaces to account for reflection loss and c, α, β, and ρ are piecewise constant functions of the axial coordinate z. The attenuation assumes the power law form $\mathrm{Re}[\alpha(f)] = \alpha_0(f/f_0)^\eta$ and requires the value at f_0=1MHz α_0 and the exponent $\eta \in [0, 2]$ for each layer. From this the corresponding phase velocity dispersion (imaginary part of α) is calculated exactly using the differential form of the Kramers-Krönig relations. After p has been calculated it is used to obtain the power density

$$Q = \frac{1}{\rho c}\left(\phi \sum_k \mathrm{Re}(\alpha_k)|A_k|^2 - \frac{d}{dz}\left\langle p^2 \right\rangle\right),$$

where $p = \sum_k \left[A_k e^{i\omega k(z/c-t)} + A_k^* e^{-i\omega k(z/c-t)}\right]$, ϕ (dimensionless) is the fraction of attenuation due to absorption, and k is the harmonic number. The summed terms result from viscous heating while the derivative of the time average of the squared pressure accounts for nonlinear loss. With the power density known, the temperature dynamics are calculated using the bioheat transfer equation

$$\rho C_p \frac{\partial T}{\partial t} = \kappa \nabla^2 T - wT + Q.$$

Here C_p (J/kg°C) is heat capacity, T (°C) is temperature rise above equilibrium, κ (W/m°C) is thermal conductivity, and w (kg/m^{3}s) is the blood perfusion rate. From the temperature dynamics, thermal damage is estimated using the dose metric

$$D(t) = \int_0^t 2^{T(t')-43} dt',$$

where t has units of cumulative equivalent seconds of exposure at 43°C. This is divided by 60 to obtain the familiar unit CEM (cumulative equivalent minutes).

2.2 Features

• Easy to use in MATLAB environment (The MathWorks, Inc., Natick, MA).
• Frequency-domain representation is suited for continuous wave simulations and roughly square-envelope pulses for which the pulse duration is much longer than the acoustical cycle.
• Accommodates power-law attenuation vs. frequency relationship. The corresponding phase velocity dispersion necessary to maintain causality is determined using the Kramers-Kronig relations.
• Accommodates any source pressure distribution corresponding to an axisymmetric transducer.
• Absorbing boundary conditions prevent spurious reflections from artificial boundaries of the computational domain.

• Includes bioheat transfer equation solver to determine temperature and thermal dose information.
• Automatically produces plots of many important quantities (compressional and rarefactional pressure, temperature rise, thermal dose, etc).
• Computed quantities are accessible by the user for postprocessing.
• Matlab scripts are documented and easy to understand to facilitate user customization.
• Updated plots of salient quantities.
• Option for checking the computed temperature rise of a linear field against the Bacon-Shaw formula.

2.3 System Requirements

Any modern computer running a contemporary version of MATLAB should work. The package was developed on MATLAB Version 9.4.0.813654 (R2018a); backward-compatibility has not been tested.

3 Getting started

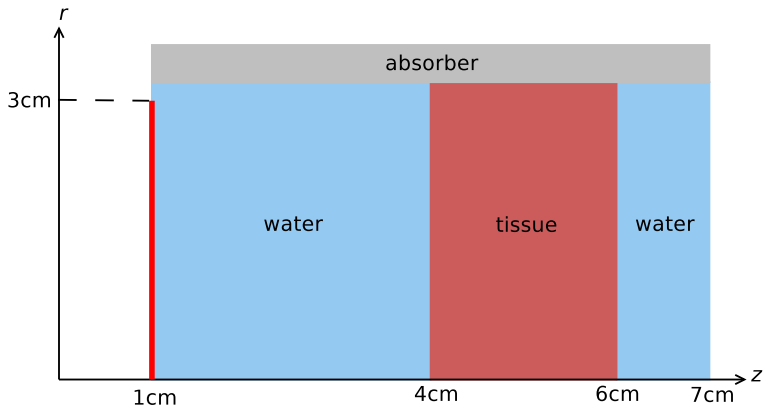
After obtaining and expanding `HITU_Simulator_v2.0.tar`, you should have a collection of files in a HITU_simulator_v2.0 directory. The following subsection includes a list of those files along with brief descriptions.

3.1 Included files

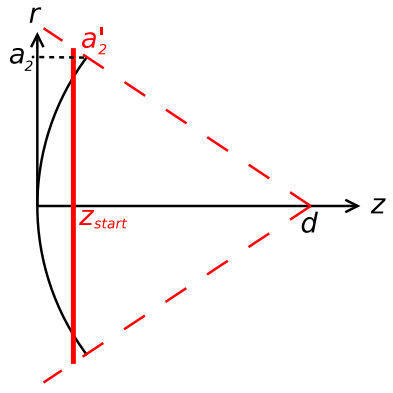
After unpacking the `tar` file, you should have the following files in a directory called HITU_Simulator_v2.0:
`HITU_Simulator_v2.0_manual.pdf` - User's manual for this software package
`WAKZK.m` - Axisymmetric wide-angle KZK integrator
`BuildPade11operators.m` - Second-order diffraction operators
`BuildPade12operators.m` - Third-order diffraction operators
`SourceFilterH` - Lowpass filter for source boundary condition
`SynthAxScan.m` - Spatial averaging on propagation axis
`SynthRadScan.m` - Spatial averaging on planes perpendicular to axis
`TDNL.m` - Time-domain nonlinear solver
`BHT.m` - Axisymmetric BHT integrator
`BuildBHTperipherals.m` - Diffusion and perfusion operators for bioheat equation
`vektorize.m` - Converts a matrix into a stacked column vector
`matrixize.m` - Converts a stacked column vector into a matrix
`LinearHeating.m` - Computes temp rise according to an exact solution of the heat equation given an ellipsoidal Gaussian power density

3.2 Example simulation

The code comes pre-configured to run a simulation which may facilitate the user's familiarity with the software. The two main parts of the package are the propagation (ultrasound field) and heating (temperature elevation) modules. The propagation part is pre-configured with the example of a spherical bowl transducer with inner radius a_1=1cm, outer radius a_2=3cm, and radius of curvature d=5cm radiating 200W acoustic power at 2MHz into water. A 2cm thick tissue slab is located at z=4cm, corresponding to 4cm from the apex of the transducer bowl (and in this case, 3cm from the aperture plane of the transducer bowl). The simulation terminates at z=7cm. The following cartoon depicts the computational domain:



The source of the field (indicated by the bold red line) is a projection onto the aperture plane of the bowl transducer located at axial position z=$d-\sqrt{d^2-a_2^2}$ using a formula from geometrical acoustics (high-frequency approximation). If you want to start your simulation at z=0cm or anywhere else, the "effective" source aperture a_2' changes as illustrated in the following diagram:



Here the effective source (red) is shown in relation to the physical shape of the transducer bowl. The true and effective aperture radii are also indicated in black and red text, respectively. All the parameters that inform the propagation module (some are also used by the heating module) are indicated in the following table:

	parameter	variable	value	unit
Transducer	frequency	`Tx.f`	2e6	Hz
	inner radius	`a_1`	1.0	cm
	outer radius	`a_2`	3.0	cm
	focusing depth	`Tx.d`	5.0	cm
	power	`Tx.P`	200	W
Computational domain	starting point	`z_start`	$d - \sqrt{d^2 - a_2^2}$	cm
	max axial distance	`Grid.Z`	7.0	cm
	max radius	`Grid.R`	$1.05a_2' + 2\lambda$	cm
	number of harmonics	`Grid.KK`	128	-
	radial grid density	`ppw_r`	15	-
	axial grid density	`ppw_z`	10	-
Hydrophone	element diameter	`hd`	0.2	mm
Graphical output	axial locations	`z_output`	4.65, 4.95, 5.25	cm
Material 1	material transition distance	`Layer(1).z`	0.0	cm
	small-signal sound speed	`Layer(1).c`	1482	m/s
	mass density	`Layer(1).rho`	1000	kg/m^3
	attenuation at 1 MHz	`Layer(1).alpha`	0.217	dB/m
	fraction due to absorption	`Layer(1).fraction`	0.0	-
	exponent of attenuation power law	`Layer(1).eta`	2.0	-
	nonlinear parameter	`Layer(1).beta`	3.5	-
	heat capacity	`Layer(1).Cp`	4180	J/kg/K
	thermal conductivity	`Layer(1).kappa`	0.6	W/m/K
	perfusion rate	`Layer(1).w`	0.0	kg/m^3/s
Material 2	material transition distance	`Layer(2).z`	3.0	cm
	small-signal sound speed	`Layer(2).c`	1629	m/s
	mass density	`Layer(2).rho`	1000	kg/m^3
	attenuation at 1MHz	`Layer(2).alpha`	58.0	dB/m
	fraction due to absorption	`Layer(2).fraction`	0.9	-
	exponent of attenuation power law	`Layer(2).eta`	1.0	-
	nonlinear parameter	`Layer(2).beta`	4.5	-
	heat capacity	`Layer(2).Cp`	4180	J/kg/K
	thermal conductivity	`Layer(2).kappa`	0.6	W/m/K
	perfusion rate	`Layer(2).w`	20	kg/m^3/s
Material 3	material transition distance	`Layer(3).z`	7.0	cm
	small-signal sound speed	`Layer(3).c`	1482	m/s
	mass density	`Layer(3).rho`	1000	kg/m^3
	attenuation at 1 MHz	`Layer(3).alpha`	0.217	dB/m
	fraction due to absorption	`Layer(3).fraction`	0.0	-
	exponent of attenuation power law	`Layer(3).eta`	2.0	-
	nonlinear parameter	`Layer(3).beta`	3.5	-
	heat capacity	`Layer(3).Cp`	4180	J/kg/K
	thermal conductivity	`Layer(3).kappa`	0.6	W/m/K
	perfusion rate	`Layer(3).w`	0.0	kg/m^3/s

3.2.1 Propagation

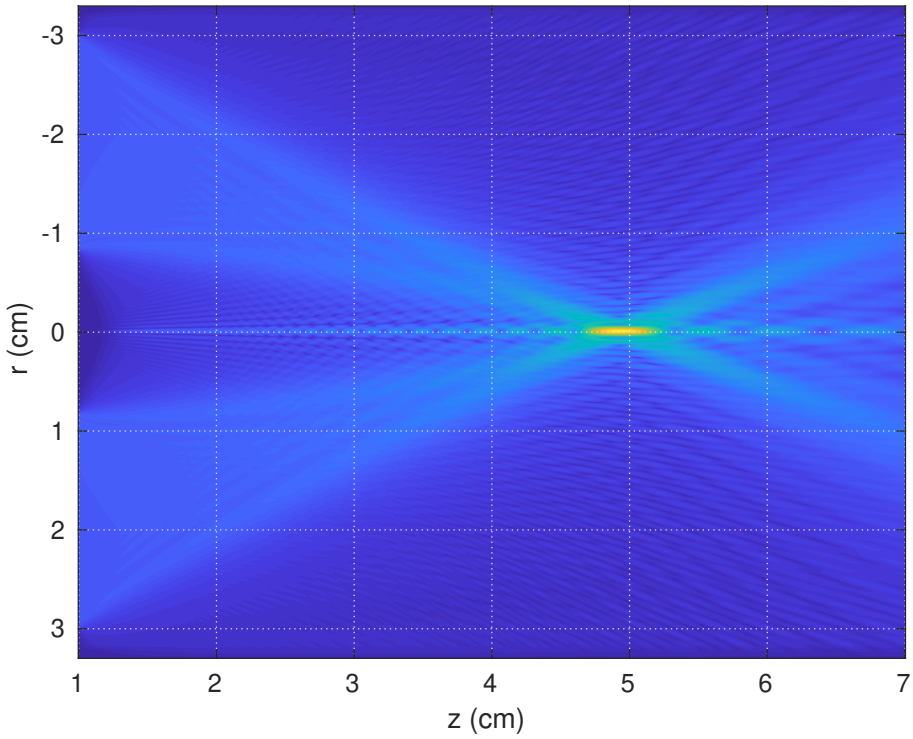
To run the example simulation, start MATLAB, go to the HITU_Simulator_v2.0 directory, and type the command:

```
[Grid,Layer,Q]=WAKZK();
```

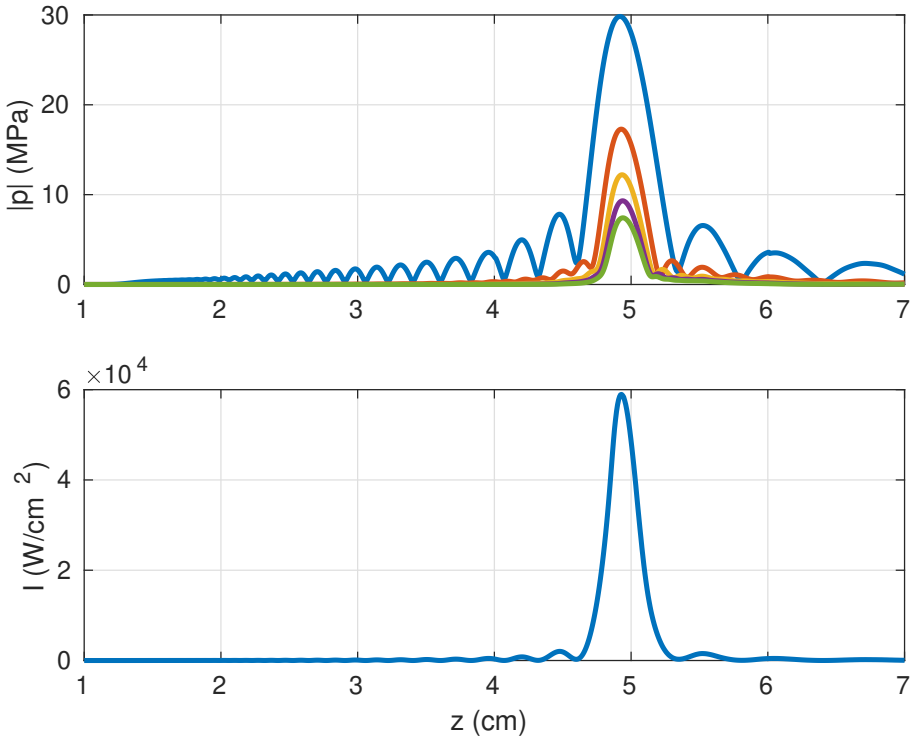
at the prompt. The ouput variables include two structures `Grid` and `Layer`, which contain the computational domain and media parameters, respectively, as well as the power density `Q`. These are used to inform the heat transfer model discussed in the next subsection. As the ultrasound field simulation runs, the following information is printed to the screen:

```
Wavelength = 0.74 mm
Node count
        Axial 810
        Radial 3649
Grid stepsize
        dz = 0.07 mm
        dr = 0.01 mm
        dt = 1.96 ns
Took 17.0 minutes.
```

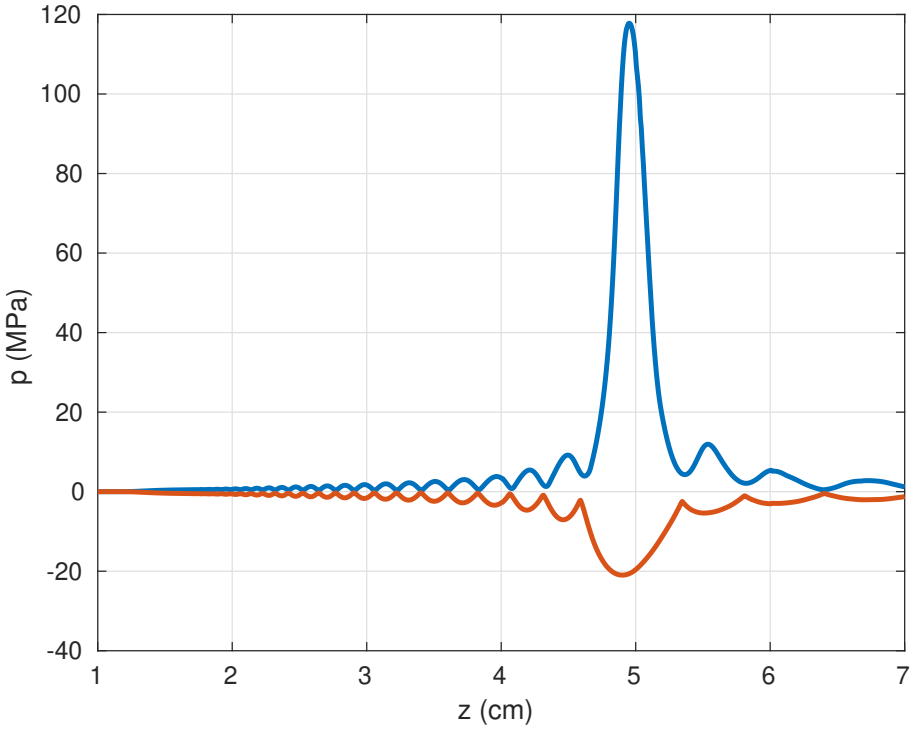
On my laptop (Dell E7270, Intel Core i7-8650U processor, 16 GB ram) this simulation required 17 minutes. Once the propagation simulation is complete, the code produces eight figures:



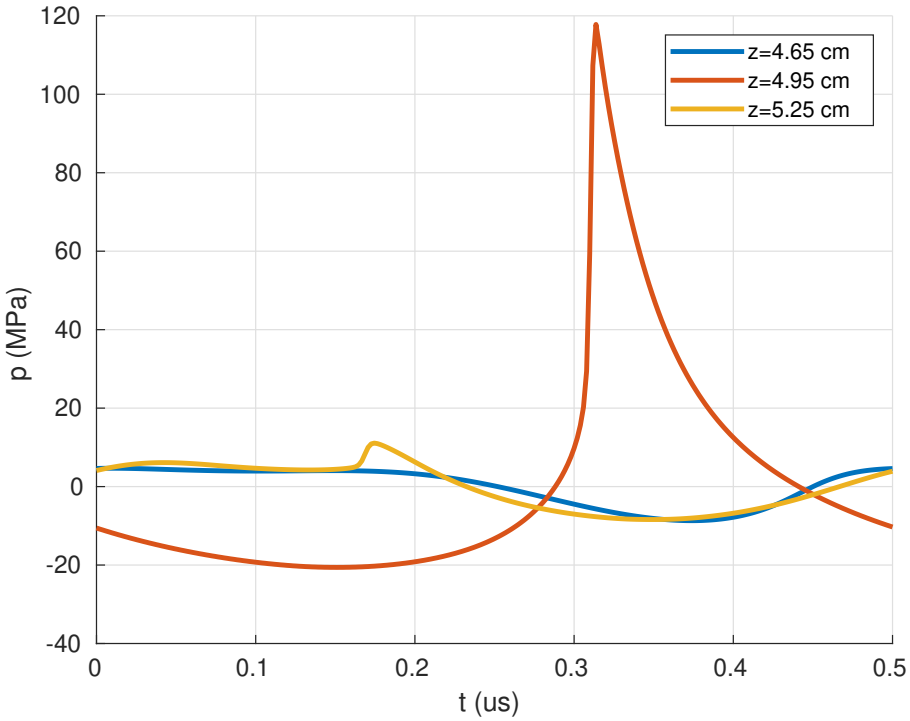
A qualitative plot of the spatial distribution of the ultrasound field. This is actually the intensity raised to the power 0.2, which highlights the low-intensity interference patterns in the field, but still shows hotspots.



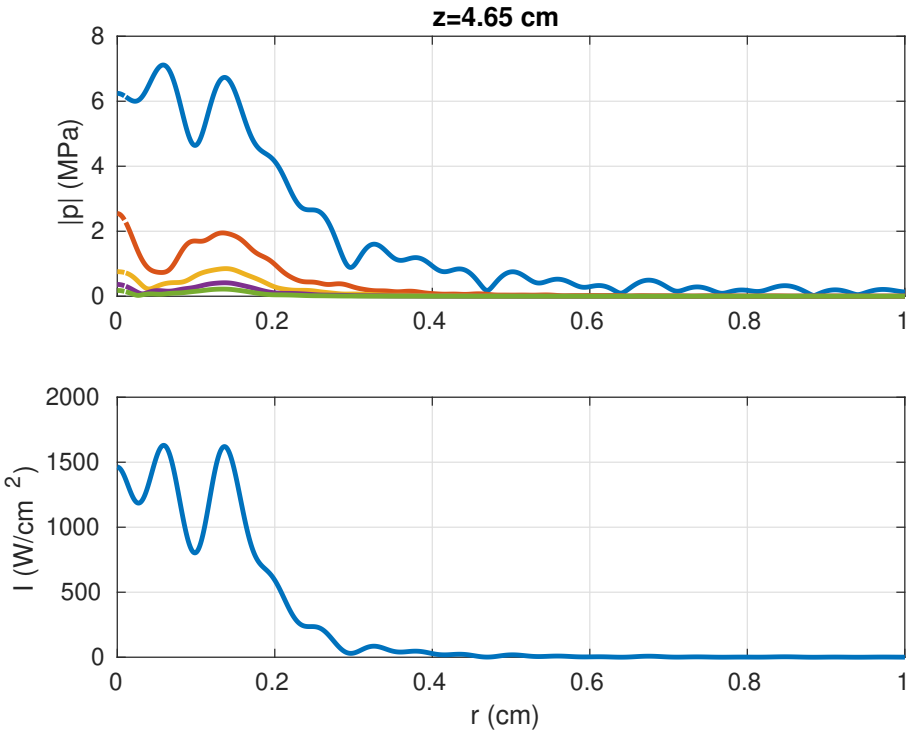
The axial distribution of the first five harmonic pressure amplitudes as well as the intensity.

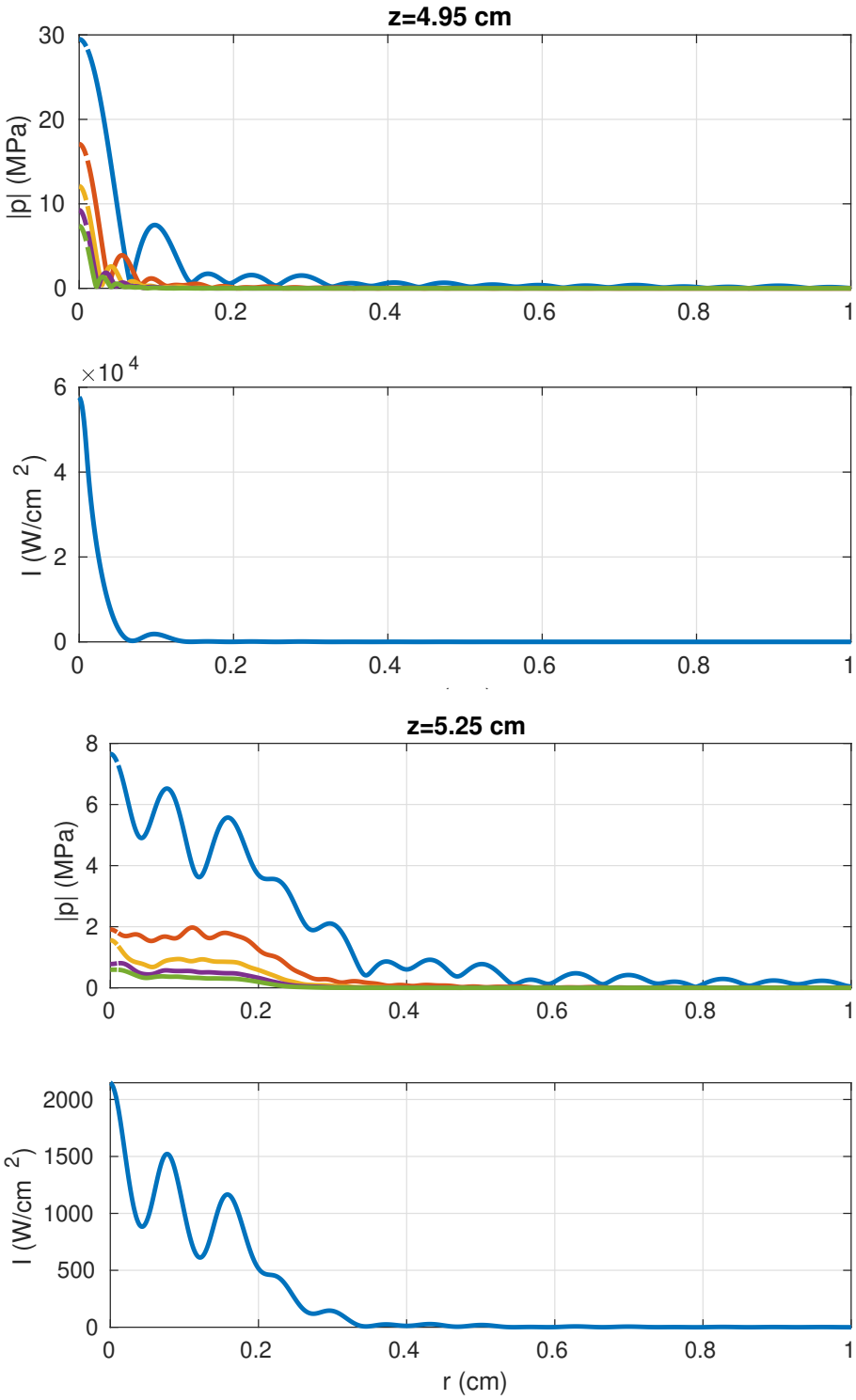


Axial distribution of peak compressional (blue) and rarefactional (red) pressures.

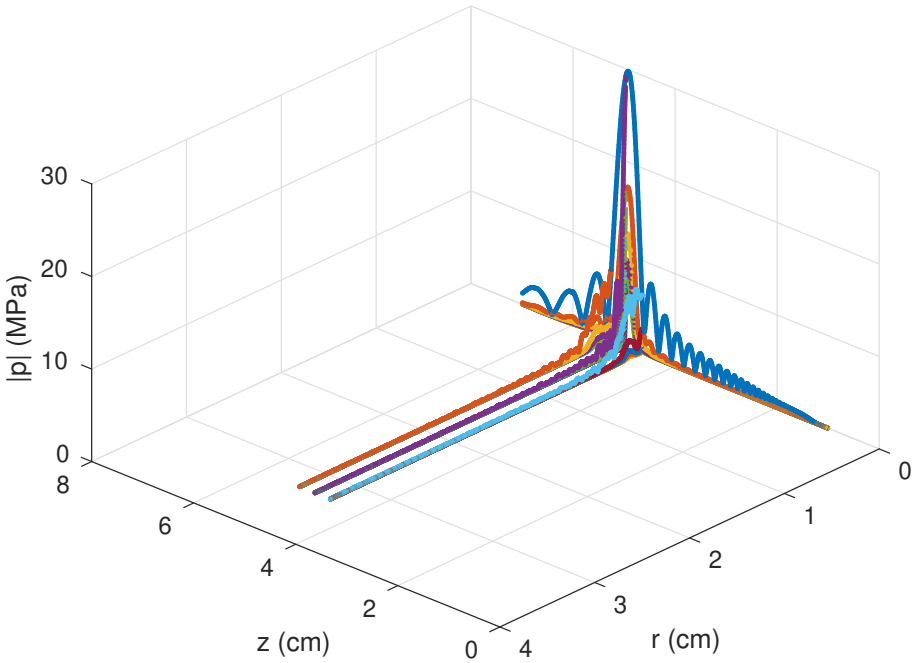


Pressure waveforms at user-defined axial locations.



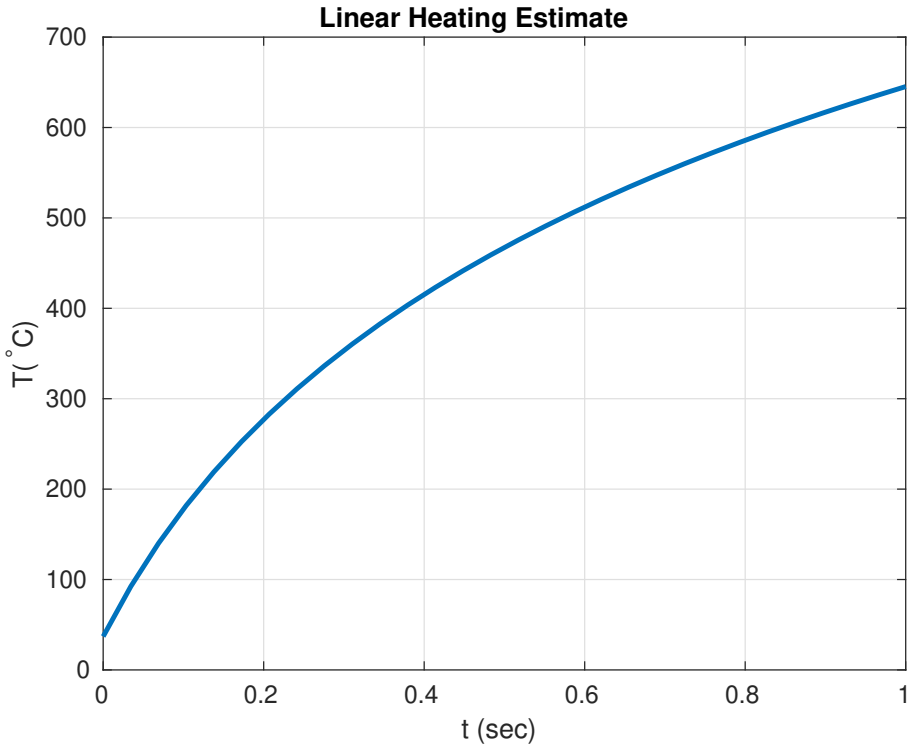


Radial distribution of the first five harmonic pressure amplitudes at user-defined axial locations.



A plot which combines the axial and radial pressure distributions.

The



A plot showing the expected temperature elevation from a linear field. This may be included to verify the heat transfer module is working correctly, at least in the linear case.

Note: for linear simulations, the number of harmonics should be set `Grid.KK=1`, and fewer plots are produced since some are only relevant to nonlinear propagation.

3.2.2 Heating

After the ultrasound field simulation is complete, it produces the power density of the field which then informs the heat transfer model. This example is a single 2-second sonication using a 0.8% duty cycle, followed by a 3-second cooling period. All the parameters that inform this model are indicated in the following table: Before running the heating module, I recommend downloading Factorize, "A simple-to-use object-oriented method for solving linear systems and least-squares problems" by Tim Davis, available for free here:

`https://www.mathworks.com/matlabcentral/fileexchange/24119`

Indicate its location in your computer by updating the argument of the `path` function in the BHT.m script. Linking to this package allows the heating module to run *much* faster. To run the heating example, type the command:

```
BHT(Grid,Layer,Q/125);
```

at the MATLAB prompt. Note that the power density `Q` is scaled by 125 to represent the 1/125 = 0.8% duty cycle. This starts the BHT solver using the additional thermal parameters listed in the following table:

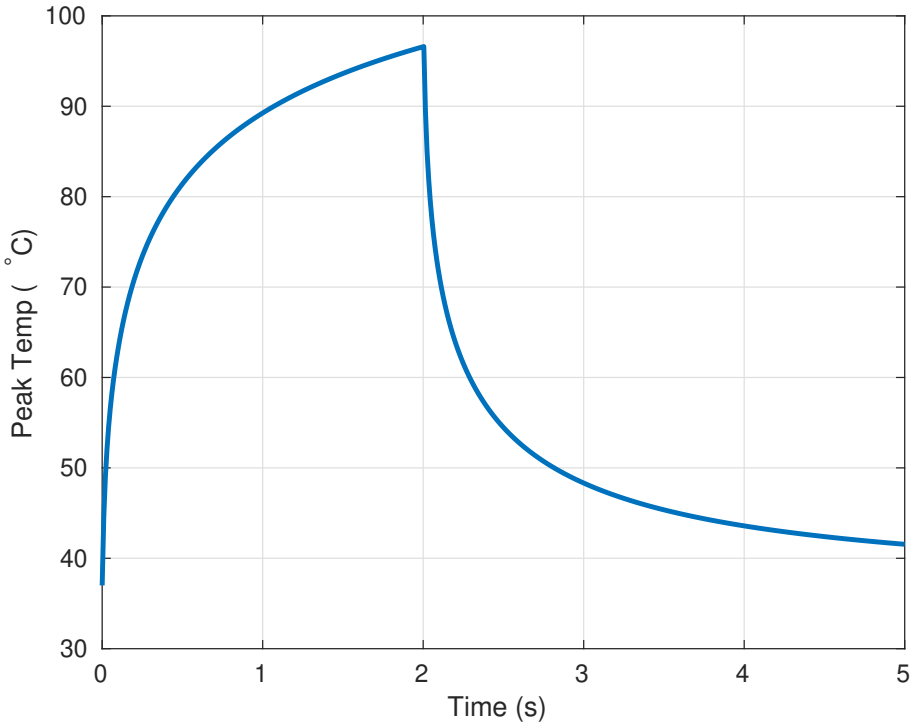
parameter	variable	value	unit
sonication duration	`t_h`	2.0	sec
cooling duration	`t_c`	3.0	sec
baseline temperature	`Teq`	37	°C
dose threshold for safety	`safety`	80	CEM_{43}
dose thershold for efficacy	`efficacy`	240	CEM_{43}

Typically the discretization of the BHT equation is coarser than that of the KZK equation, so grid coarsening is employed to speed integration of BHT. Reduction factors for coarsening in each direction may be set by the user. As the simulation runs, the following is printed to the screen:

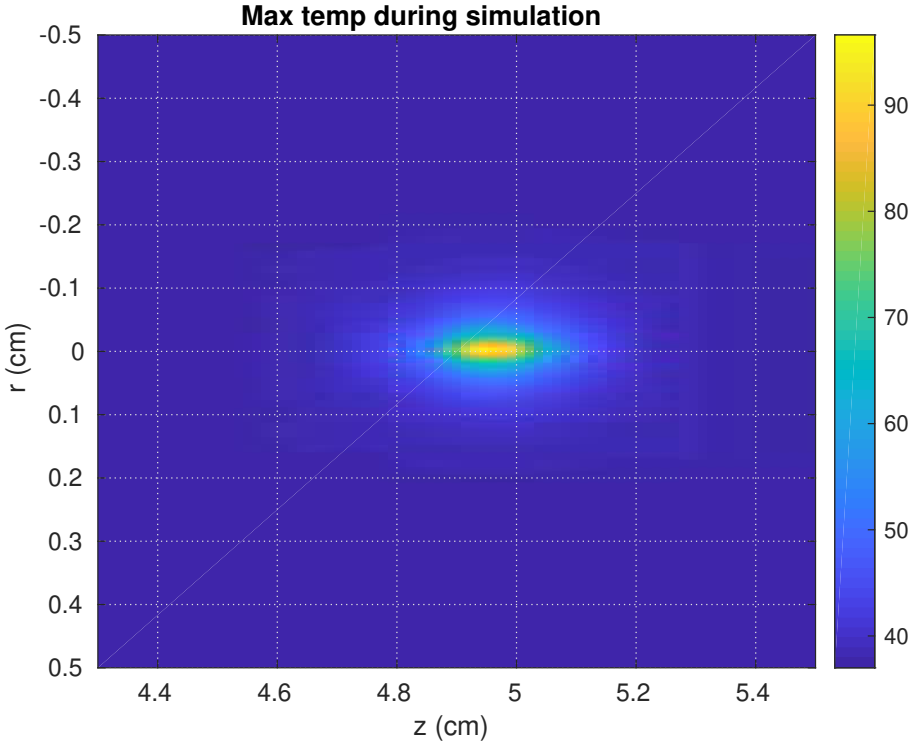
```
Grid stepsize
        dz = 0.22 mm
        dr = 0.02 mm
        dt = 0.01 s
Node count
        Axial 270
        Radial 1825
        Temporal 500
Took 84.4 seconds.
```

The timestep size `dt` and total number of steps taken `Temporal` are displayed as are the coarsened spatial grid dimensions. Coarsening is specified in `BuildBHTperipherals.m`, the control variables are `r_skip` and `z_skip`.

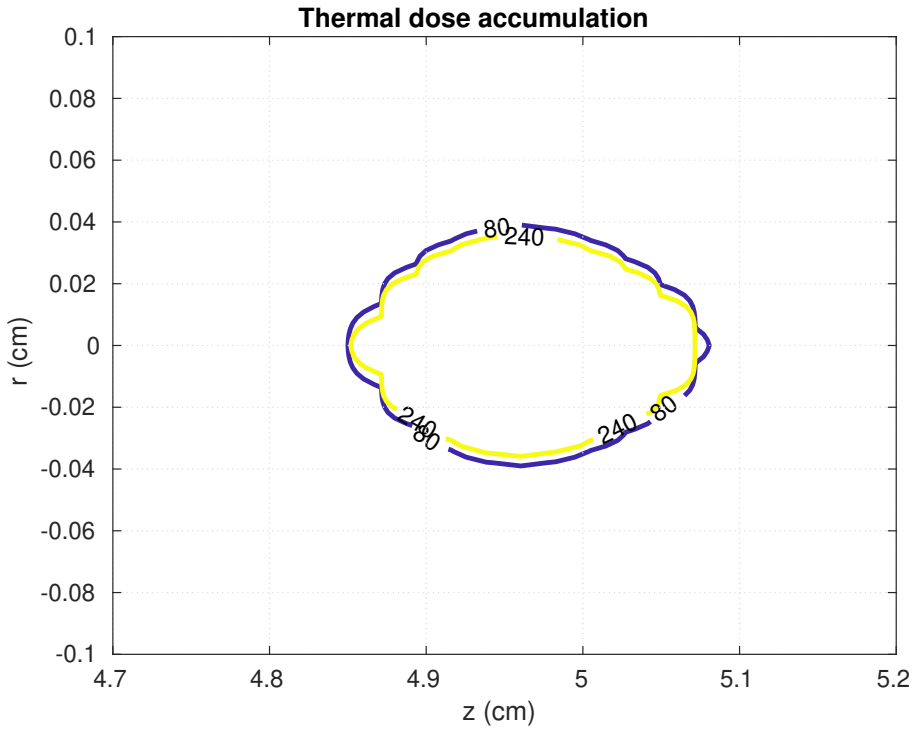
At the end of the simulation, three figures are produced:



Peak temperature vs. time.



Temperature distribution at the time peak temperature occurs.



Thermal dose calculation at the end of the simulation. Two curves are shown, one at 80 CEM indicating the region where no thermal effect occurs (safety threshold) and one at 240 CEM which shows the region of total necrosis (efficacy threshold). These threshold values are set by the user in the bioheat transfer script `BHT.m`. Note the ripples in the countours are a discretization artifact. Increasing grid resolution reduces this effect.

3.3 Setting up your HITU beam simulation

To simulate your setup, you will need to edit `WAKZK.m` to specify transducer and medium characteristics (including thermal characteristics), the computational domain, whether spatial averaging is performed, and locations where the transverse field and waveforms are plotted. The input parameters are defined near the top of the script while the "guts" are further down. In the following subsections the use of the propagation code's features is discussed.

3.3.1 Computational grid parameters

The size of the computational domain in the axial dimension is determined by `z_start` and `Grid.Z`; these are the locations of the effective source and the distance to which the field is propagated, respectively. In the radial dimension, the domain goes from zero to a value determined by software, which is the radius of the effective source plus the thickness of the absorbing layer.

The three parameters which control the grid resolution are `Grid.KK`, `ppw_r`, and `ppw_z`. These are the number of harmonics and the node density (points per fundamental wavelength) in the r- and z-directions, respectively. They are set at values that are appropriate for the default parameters. However, a simple convergence analysis in which these values are doubled and output values compared should always be performed. If the output values are too disparate,

refine and run again.

3.3.2 Spatial averaging

This new feature mimics the effect of a hydrophone measurement using a finite element size, specified by the user and indicated in the code by `hd` (hydrophone sensing element diameter in mm). In the current state it assumes the effective element size is constant with frequency. Measureable acoustic field quantities such as axial and radial pressure and pressure waveforms are affected; but the computed quantity acoustic intensity is not. Try running the default simulation with `hd=0` and compare to see the effect of spatial averaging.

3.3.3 Graphical output locations

The transverse pressure and intensity fields as well as the on-axis waveforms are plotted in as many axial locations as the user specifies. This is controlled by the array `z_output` which is given the values `[4.65 4.95 5.25]` corresponding to prefocal, focal, and post focal locations.

3.3.4 Discretization

In the radial direction, the asymptotic result that for focused beams the width of the kth harmonic goes like $k^{-0.3}$ is used to modify the discretization. You may specify `ppw_r = 20`, for example, and for linear simulations `Grid.KK = 1` the code will use 20 points per wavelength. But if 256 harmonics are used, $256^{0.3} = 106$ points per wavelength is used to account for the narrowing of the higher harmonic fields.

3.3.5 User-specified source

The default configuration involves a spherical cap transducer which is very common in HIFU research and clinical applications. However, the software has provisions for user-specified sources as well, which may result from acoustic holography, hydrophone scans, or alternative theoretical pressure distributions. In the supplied example the expression for a spherically-converging wave is used to determine the phase and amplitude distribution. It is then filtered and scaled appropriately to prevent numerical artifacts occuring.

The user may specify any complex pressure distribution $A = A(r)$ and the same process will ensue. The code will filter and rescale according to the user-specified total acoustic power. This part of the `WAKZK.m` starts on line 130:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Equivalent source %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This source is a converging spherical wave at z=z_start bounded by a2p.
% Note: it's dimensionless:
A = Tx.d*exp(-i*Layer(1).k*sqrt(Grid.r.^2+(Tx.d-z_start)^2)).*(Grid.r<a2p) ...
   ./sqrt(Grid.r.^2+(Tx.d-z_start)^2);
if(Tx.a1 ~= 0)          % if there's a hole in the Tx
  a1p = Tx.a1*(Tx.d-z_start)/sqrt(Tx.d^2-Tx.a1^2);
```

```
  A = A.*(Grid.r>a1p);
end



% The user could specify a custom source here [any complex function A=A(r)].
%A = ;



% Apply a low-pass filter to the source:
A = SourceFilterH(Grid.r,A,Layer(1).k);

% Next scale the source by the appropriate pressure coefficient so that it has
% the proper total acoustic power:
integral = 2*pi*dr*trapz(abs(A).^2.*Grid.r);
integral = 1e-4*integral;                % convert units from cm^2 to m^2
p0 = sqrt(2*Layer(1).rho*Layer(1).c*Tx.P/integral);
A = p0*A;       % dimensionalize the boundary condition (units of Pa)
```

Simply comment out the definition of A and apply your own. Note that you may need to specify `a2p`, the radius of your source function to ensure your source is within the computational domain. It is specified on lines 109-112 of `WAKZK.m`.

3.3.6 Higher-order approximation of the diffraction operator

The propagation part of the code is set up with a second-order approximation of the diffraction operator which is good for propagation up to about $\theta = \tan^{-1}(a_2'/d) = 35°$ off-axis in focused-beam applications. The code includes provisions for a third-order approximation good to θ=45°. This added accuracy comes at some computational expense so expect run times to increase by at most 50%. It is simple to use this higher-order method but you must edit the integration loop starting around line 260 in `WAKZK.m`:

```
%% integration loop:
for ii=1:II
  %build operators for Layer ii:
  for kk=1:Grid.KK
    %[M(kk).P1,M(kk).P2,M(kk).P3] = ...
    %  BuildPade12operators(A,kk,dz,Layer(ii).k,Grid.JJ);
    [M(kk).P1,M(kk).P2] = ...
      BuildPade11operators(A,kk,dz,Layer(ii).k,Grid.JJ);
  end
  mu = (Layer(ii).beta/2/Layer(ii).rho/Layer(ii).c^3)*(0.01*dz/dt);
  cutoff = Layer(ii).alpha(1)*Layer(ii).rho*Layer(ii).c^2 ...
```

```
        / Layer(ii).beta/Layer(ii).k; % cutoff for nonlinearity
for nn=Layer(ii).index:Layer(ii+1).index-1
  %integrate nonlinear term:
  [p,w(nn+1,:),Ppos(:,nn+1),Pneg(:,nn+1),I_td(:,1)] = ...
    TDNL(q,w(nn+1,:),Y,Grid.KK,Grid.JJ,mu,cutoff,Ppos(:,nn),Pneg(:,nn),I_td(:,1))
  %attenuation/dispersion term:
  p(:,kk) = p(:,kk).*exp(-Layer(ii).alpha(kk)*dz);
  %diffraction term:
  for kk=1:Grid.KK
  %p(:,kk) = M(kk).P1 \ (M(kk).P2 \ (M(kk).P3*p(:,kk)));    % for Pade 12
    p(:,kk) = M(kk).P1 \ (M(kk).P2*p(:,kk));                 % for Pade 11
  end

   ⋮
```

Note the two areas of red text. The color highlights the parts to change. The default mode uses the second order approximation and `BuildPade11operators` is called while `BuildPade12operators` is commented. Simply comment out the second-order parts and uncomment the third-order parts.

3.3.7 Linear heating

In the process of code verification and validation it is necessary to have comparators against which one can check the results of computer models. At runtime, `WAKZK.m` can call the script `LinearHeating.m` which produces a plot of the temperature rise at the focus of a linear field. This may be used to check the results of the heat transfer part of this package in the case of a linear ultrasound field. `LinearHeating.m` plots the formula

$$\Delta T_{ellipse}(t) = \frac{\alpha I_0 a^2 b}{K\sqrt{b^2-a^2}}\left[\tanh^{-1}\frac{b}{\sqrt{b^2-a^2}} - \tanh^{-1}\sqrt{\frac{b^2+4\kappa t}{b^2-a^2}}\right],$$

where α is absorption in Np/m at the source frequency, K is conductivity, κ is diffusivity, and t is time. This formula results from an elliptic Gaussian intensity distribution:

$$I(r,z) = I_0 \exp[-(r/a)^2 - (z/b)^2]. \tag{1}$$

This distribution approximates that of a focused ultrasound beam. The linear heating function is called at line 320 of `WAKZK.m`:

```
LinearHeating(Layer(2),Grid,I);          % assumes a focused beam and that
                                         % the focus is in layer 2
```

and is informed by the tissue characteristics of the layer in which the focus occurs, grid information, and the computed intensity distribution. It assumes an equilibrium temperature of 37°C and is set up for a 1-second continuous wave exposure. Of course these can be changed by editing `LinearHeating.m`. Note the package is supplied with this function disengaged. Uncomment the line to perform this verification step.

3.3.8 Note

There is no formula that predicts how many harmonics should be included in a given simulation. But the number increases with higher gain frequency, material nonlinearity, and drive level. The value of `Grid.KK` should be an integer power of 2 since the fast Fourier transform is used in this part of the computation. Other numbers will work but simulation speed will suffer. For nonlinear solutions (even weakly nonlinear), `Grid.KK` ≥ 32 is recommended in order to adequately resolve the temporal waveform.

3.4 Setting up your heating simulation

Edit `BHT.m` to specify heating and cooling duration, equilibrium temperature, and thermal dose thresholds. The input parameters are defined near the top of the script and again the "guts" are further down. The first few lines of the script follow:

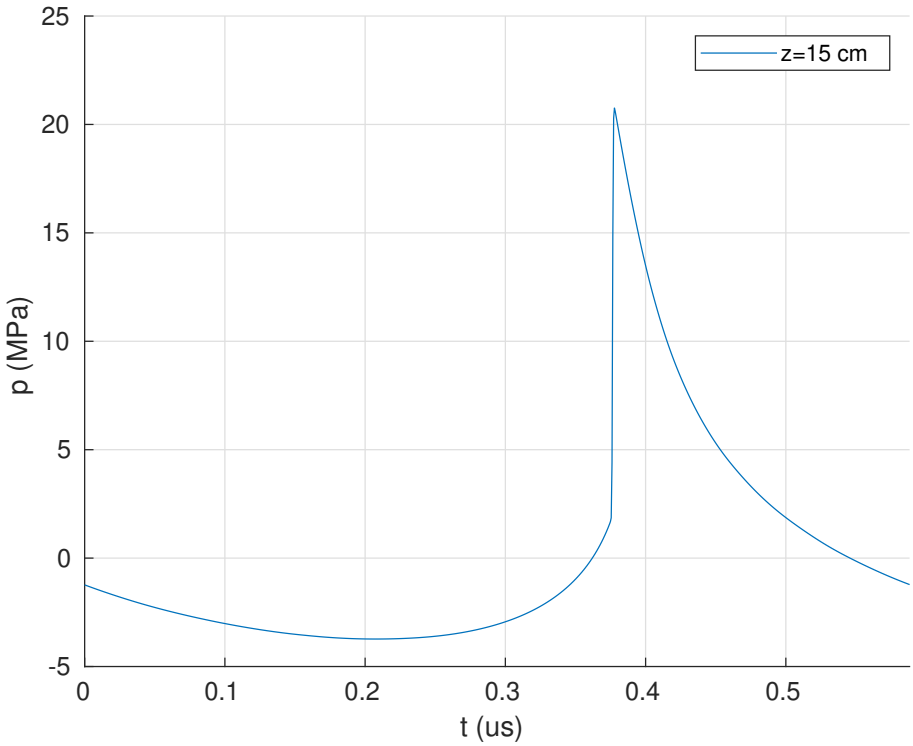
```
function[] = BHT(Grid,Layer,Q)
tic;
%% set path to include Factorize for shorter run times:
path(path,'/home/josh/Downloads/Factorize/')


% set heating and cooling durations:
t_h = 2;        % heating time (s)
t_c = 3;        % cooling time (s)

% set equilibrium temperature:
Teq = 37;       % celcius degrees

% Thermal damage thresholds:
safety = 80;    % dose (CEM43) where 0% cell necrosis occurs; safety threshold
efficacy = 240; % dose at which 100% cell nectosis occurs

        ⋮
```

The `path` function call is highlighted in blue. This is where you indicate where your copy of Factorize is in your computer. If you choose not to use Factorize, simply comment this out and go to line 94 of `BuildBHTperipherals.m` to change the definition of the CN1 matrix. It will give the same results but it'll take a while.

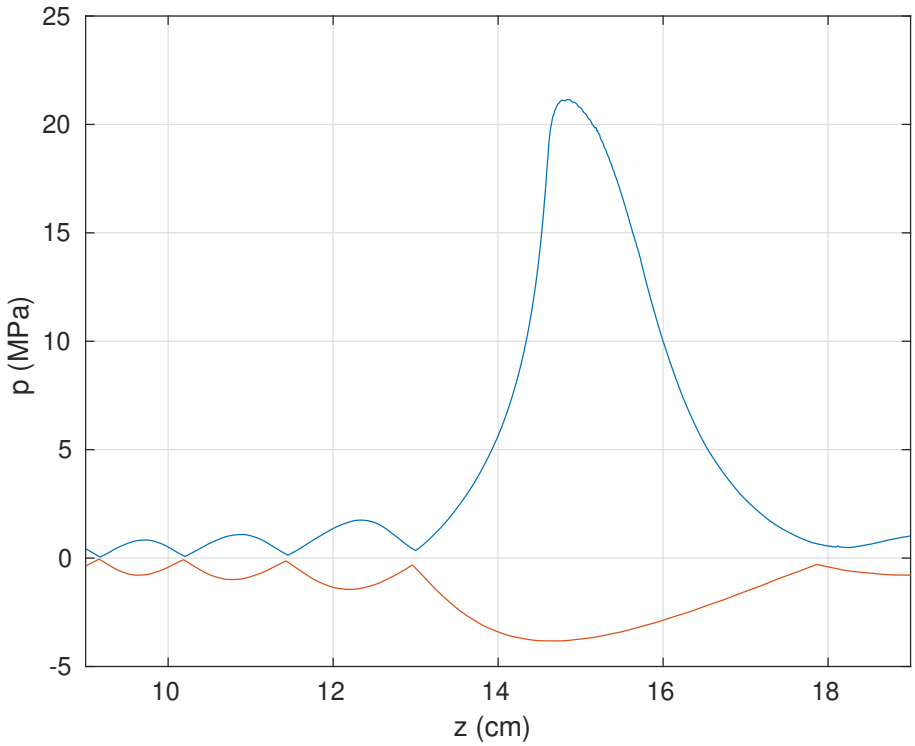
Next in the script are the parameter definitions which are few and self-explanatory. Because of the smoothing nature of the BHT equation, its resolution requirements are lower than those of the KZK equation Thus, to speed the temperature calculation, provisions to coarsen the computational grid are included. The user may adjust the coarsening by changing the factors `r_skip` and `z_skip` in the script `BuildBHTperipherals.m`. For example, $\Delta r \rightarrow$`r_skip`$\times \Delta r$.

4 Verification and Validation

To check the accuracy of the nonlinear modeling, a low-gain, high nonlinearity scenario was tested using the parameters detailed in Yuldashev PV and Khokhlova VA, *Simulation of three-dimensional nonlinear fields of ultrasound therapeutic arrays*, Acoust. Phys. 57, pp. 334-347 (2010). This is a simulation of a single-element transducer with an 8.2cm aperture, 15cm radius of curvature, driven at 1.7MHz with 40W acoustic power into water. The paper reports close agreement of the focal waveform and compressional and rarefactional pressures on axis produced by two different algorithms. HITU_Simulator2.0 was used to produce the following plots of the same quantities which appear below. The results are virtually identical:



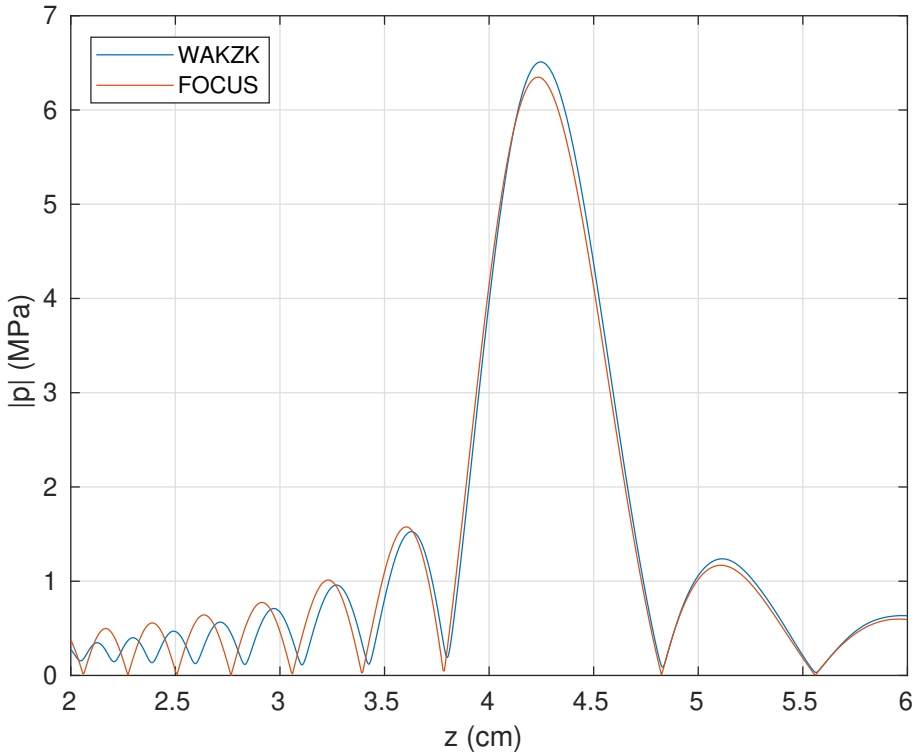
Temporal waveform at geometrical focus ($z = d$).



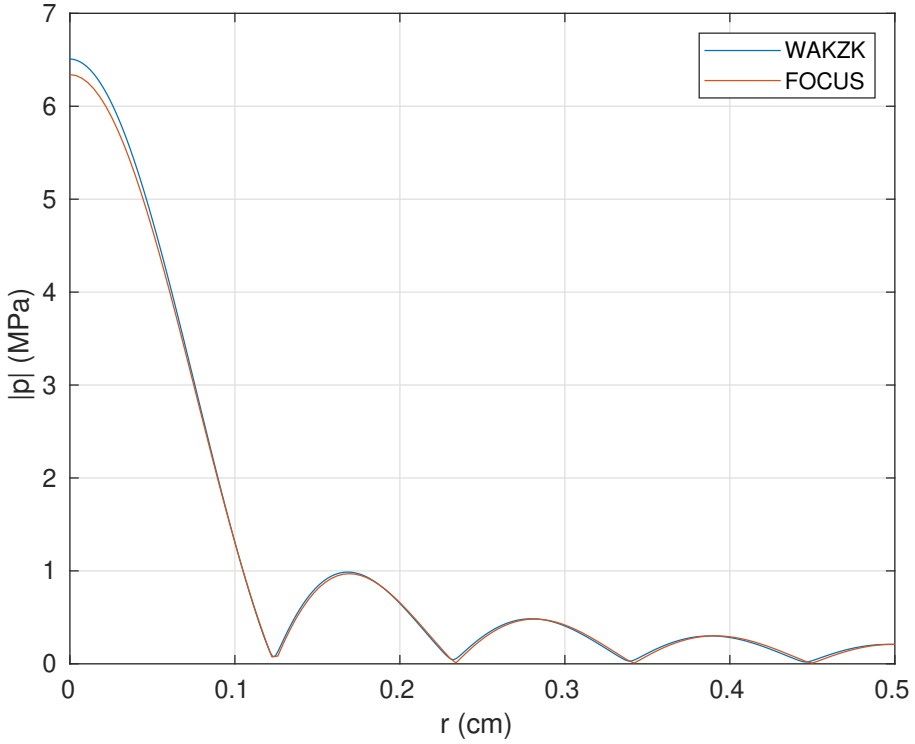
Compressional and rarefactional pressure along central axis.

I cannot reproduce the plots from the paper here, but I invite the reader to compare. Note that the scaling is different; the initial pressure p_0 used in the paper is 0.15MPa.

To check the focusing capability of HITU_Simulator2.0, the third-order approximation was used to simulate the linear field of a very shallowly-focused spherical bowl transducer. The transducer was specified with 6cm aperture diameter and 4.25cm radius of curvature driven at 1MHz with 20W acoustic power. This is in lossless media with sound speed 1500m/s. The results of HITU_Simulator2.0 are compared with FOCUS, the Fast Object-Oriented C++ Ultrasound Simulator developed at Michigan State University by Robert McGough's team (https://www.egr.msu.edu/ fultras-web/), which uses the fast nearfield method and is equivalent to a Rayleigh integral model.



Linear pressure amplitude along central axis at low F-number.



Linear pressure amplitude in focal plane (z=4.25cm) at low F-number.

The simulations were run at very high accuracy to push discretization error far below the level of model form error. HITU_Simulator2.0 was run using `ppw_r=ppw_z=100` and FOCUS was run using ndiv=500. The result shows the error commited by the third-order wide-angle parabolic approximation at θ=45°.

5 Reporting bugs

Mail to `joshua.soneson@fda.hhs.gov` with bug reports, questions, or other information related to this software.