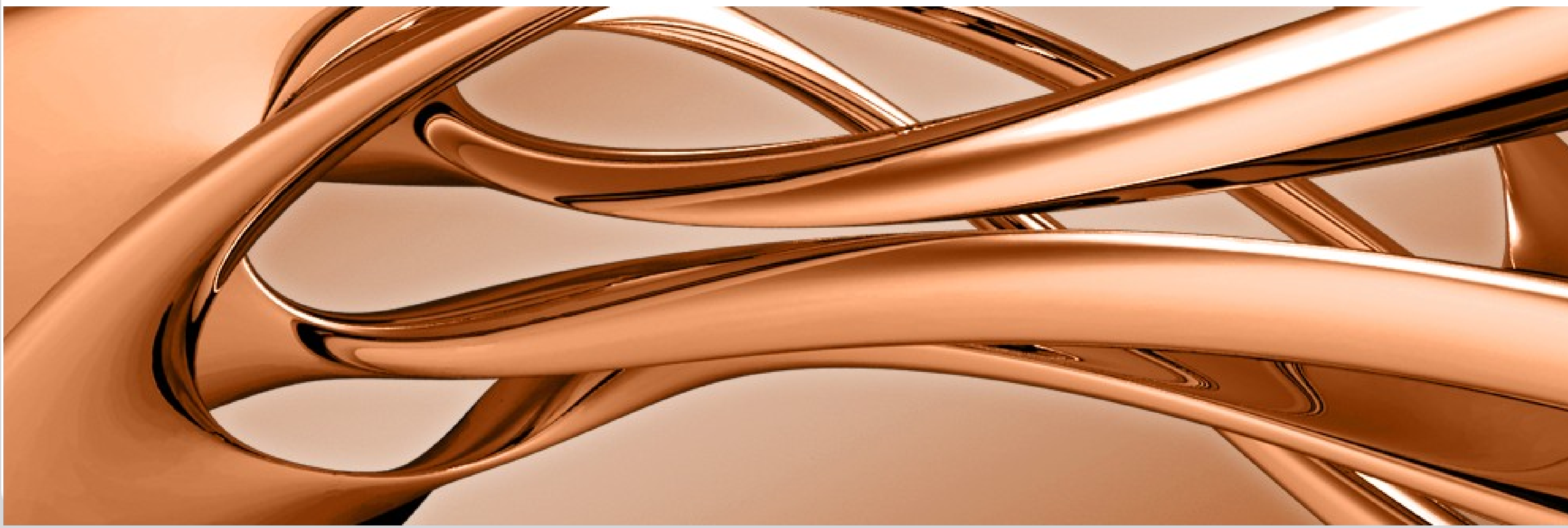
Karlsruhe Institute of Technology

Diffraktionstomographie innerhalb der Bornschen Naeherung

Marcus Hardt
Steinbuch Centre for Computing (SCC)
Karlsruhe Institute of Technology (KIT)

Forschungszentrum Karlsruhe
in der Helmholtz-Gemeinschaft

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825



Inhalt

- I: Infrastruktur für die Parallelverarbeitung
- II: Physik: Lösung der fundamentalen Gleichungen
- III: Implementierung der Lösung
- IV: 1001 Parameter
- V: Mehrfachstreuung

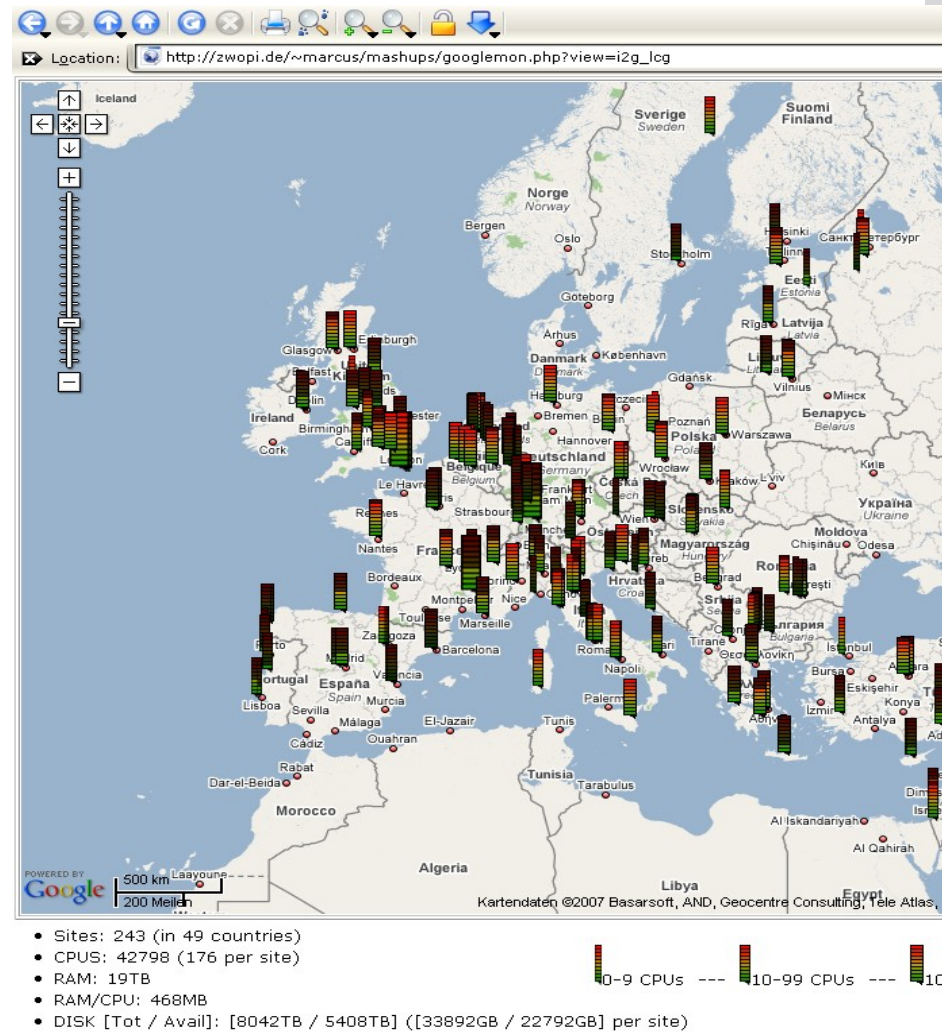
Part I:
Infrastruktur für die Parallelverarbeitung

- Ziele:
  1. aus Matlab heraus
  2. parallel
  3. möglichst viele CPUs

  zur Lösung von komplexen Aufgaben nutzbar machen.

Infrastruktur

- Glite grid Middleware
  - Viele Cluster, weltweit verteilt
  - Struktur:
    - Pro Grid
      Monitoring, Resource Broker
      Datenkatalog
    - Pro Rechenzentrum:
      1 Compute Element (CE)
      => N WorkerNodes (WN)
      1 Storage Element (SE)
      => Disk / Tape
- Eigenschaften:
  + Viele Ressourcen
  -- Komplexer Zugriff
  -- Job basiertes Paradigma
  -- Kein Zugriff aus Matlab



Marcus.Hardt@kit.edu | USCT Seminar | 28.8.2009

Verbesserung des Zugriffs

- GridSolve
  - Tool für (Remote Procedure Calls) RPC in global verteilten Umgebungen => Passt gut auf gLite.
  - ”Local” side
    - Schnittstellen für C, C++, ..., **Matlab**, Octave
  - ”Remote” side
    - Schnittstelle für **C** und Fortran
    - Via **Matlab Compiler** Einbindung von Matlab Code möglich

Source code in Matlab

- Easy to use: (Example in Matlab code)

```
y=problem(x) <=> y=gs_call('problem', x)
```

  - Transport input parameters to remote side
  - Execute “problem”
  - Transport result back
- Server executes C and Fortran libraries
  - Can be extended by the C-function `system`

=> Reduce complexity of the grid to one function call

Integration von GridSolve und gLite

- Integration
  - gLite jobs starten GridSolve Komponenten auf den gLite WorkerNodes (WN)
- Neue Probleme:
  - Infrastruktur:
    - Installation von GridSolve und Matlab Komponenten
    - Inkompatibilitäten der fundamentalen Bibliotheken (glibc)
  - Schnittstelle: Muss für jede remote funktion neu definiert werden
    - Lösung: Verallgemeinerte Schnittstelle

      => Bachelor / Masterarbeit, pending

Part II: The Basic equations

- Based on fundamental equations of physics:
  - Newton's law $F = ma$
  - Hooke's law $F = Ds$
- We can derive a differential equation for pressure and particle velocity (wave equation):

$$\partial_k p(x) + \rho(x)\partial_t v_k(x) = f_k$$
$$\partial_k v_k(x) + \kappa(x)\partial_t p(x) = q$$

- Exact solution:

$$\hat{p}^{sct}(x) = \int G(x-x')s^2\rho_o(\kappa_0-\kappa_s)\hat{p}^{tot}(x')dV(x') + \int G(x-x')\partial_k\left[\frac{\rho_0-\rho_s}{\rho_s}\partial_k\hat{p}^{tot}\right]dV(x')$$
$$\hat{v}_k^{sct}(x) = \int G(x-x')s^2\kappa_0(\rho_0-\rho_s)\hat{v}_k^{tot}(x')dV(x') + \int G(x-x')\partial_k\left[\frac{\kappa_0-\kappa_s}{\kappa_s}\partial_k\hat{v}_k^{tot}\right]dV(x')$$

$$\hat{p}^{tot} = \hat{p}^{sct} + \hat{p}^{inc}$$
$$\hat{v}^{tot} = -\frac{\partial_k\hat{p}^{tot}}{s\rho_s}$$

Approximations

- Exact solution:

$$\hat{p}^{sct}(x) = \int G(x-x')s^2\rho_o(\kappa_0-\kappa_s)\hat{p}^{tot}(x')dV(x') + \int G(x-x')\partial_k\left[\frac{\rho_0-\rho_s}{\rho_s}\partial_k\hat{p}^{tot}\right]dV(x')$$

$$\hat{v}_k^{sct}(x) = \int G(x-x')s^2\kappa_0(\rho_0-\rho_s)\hat{v}_k^{tot}(x')dV(x') + \int G(x-x')\partial_k\left[\frac{\kappa_0-\kappa_s}{\kappa_s}\partial_k\hat{v}_k^{tot}\right]dV(x')$$

- Born Approximation

  $p^{tot} = p^{inc}$ (on the right side)

- Kompressibility-only Approximation:

  $\rho_s(x) = \rho_0$

  - Not equivalent to neglecting soundspeed Variations:

    $$c = \frac{1}{\sqrt{\rho\kappa}}$$

- Results in the **"forward solution"** within the Born Approximation

$$\hat{p}^{sct}_{approx} = \int G(x-x')s^2\rho_0(\kappa_0-\kappa(x'))\hat{p}^{inc}(x')dV(x')$$

Marcus.Hardt@kit.edu | USCT Seminar | 28.8.2009

The underlying Model

- Green's function contains wave response of the system:

$$G(x, w) = e^{i2\pi f * \Delta x / c_{bg}}$$

  - No geometrical damping
  - Hygens scattering

  - Green's function can be adapted to reality

Marcus.Hardt@kit.edu | USCT Seminar | 28.8.2009

Part II:
The Basic equations

- Based on fundamental equations of physics:
  - Newton's law $F=ma$
  - Deformation Equation $F=Ds$
- We can derive a wave equation for pressure and particle velocity:

$$\partial_k p(x) + \rho(x)\partial_t v_k(x) = f_k$$
$$\partial_k v_k(x) + \kappa(x)\partial_t p(x) = q$$

$$(\nabla^2 - k^2)\hat{u}_k(x) = \hat{f}_k\hat{u}_k(x)$$

(Inhomogeneous Helmholtz Equation)

- Exact solution:

$$\hat{p}^{sct}(x) = \int G(x-x')s^2\rho_o(\kappa_0-\kappa_s)\hat{p}^{tot}(x')dV(x') + \int G(x-x')\partial_k\left[\frac{\rho_0-\rho_s}{\rho_s}\partial_k\hat{p}^{tot}\right]dV(x')$$
$$\hat{v}_k^{sct}(x) = \int G(x-x')s^2\kappa_0(\rho_0-\rho_s)\hat{v}_k^{tot}(x')dV(x') + \int G(x-x')\partial_k\left[\frac{\kappa_0-\kappa_s}{\kappa_s}\partial_k\hat{v}_k^{tot}\right]dV(x')$$

3.3 Single Step Inversion

The error made within this Approximation can be written as:

$$ERR = \sum_S \sum_R \sum_\omega \left| \hat{p}^{sct}_{real} - \underbrace{\int G(x-x')s^2\rho_0\chi(x')\hat{p}^{inc}(x')V(x')}_{=\hat{p}^{sct}_{approx}} \right|^2 \quad \text{with: } \chi(x) = \kappa_0 - \kappa_s(x)$$

$$= \Delta\chi(x)\cdot\alpha$$

This Error is positive definite, hence minimal, when the derivative against the variation parameter $\frac{\partial ERR}{\partial \alpha} = 0$. Solving this for α gives:

$$\alpha = \frac{Re\left\{\sum_{S,R,\omega} \hat{p}^{sct} \int \left(s^2\rho_o\Delta\chi G\hat{p}^{inc}\right)^* dV\right\}}{\sum_{S,R,\omega}\left|\int \Delta\chi G\hat{p}^{inc}s^2\rho_0 dV\right|^2}$$

α is maximal, if the numerator is maximal. This is the case when $\Delta\chi$ is parallel to the rest of the expression in the numerator. I.e.

$$\Delta\chi(x) = \sum_{S,R,\omega}\left(G(x-x')\hat{p}^{inc}(x')\right)^* \hat{p}^{sct}(x') \tag{11}$$

Inversion

- Inversion via "**Backpropagation**" **(BP)**

$$\Delta\chi(x) = \sum_{S,R,\omega} \left(G(x-x')\hat{p}^{inc}(x')\right)^* \hat{p}^{sct}(x')$$

  - Equivalent to first step of gradient method
- Similar result as for SAFT **but in frequency domain**
  - Advantage:
    - Frequency dependend corrections can be used
  - Disadvantage:
    - Each frequency needs to be computed individually
      => Factor ~ 500 slower
- Exploiting similarity of codes:
  - Re-use assembler parts of SAFT
  - Can benefit from preprocessing codes of SAFT

Part III: Implementation

- Data access mode
  - Data parallel
    => Send **subset of data** to a remote computer
    => **Return all voxels** of image
    => **Add images** locally

    + Re-use Assembler optimisations

    - Large data transfers for large images
  - Volume parallel
    => Send **all data** to each remote computer
    => **Return subset** of voxels
    => Combine **subimages** locally
    - One big data transfer at experiment start

      + Re-use input data for several experiments

      - Rewrite Assembler code
- Code Snippets
  - Forward Solution Kernel
  - Backward Solution Kernel

Implementation

- Forward Solution Kernel (data parallel)

```
%% Forward Solution:
for emit = emitterStart:emitterEnd
    for rec = 1:n_rec
        for freq = freq_start:freq_end
            pSctTemp = 0;
            i_zwopi_f_by_cbg = i * 2 * pi * frequency / c_bg;
            for x=1:x_max
                for y = 1:y_max
                    dist_src = sqrt((x - x_src(emit))^2 + (y - y_src(emit))^2);
                    dist_rec = sqrt((x - x_rec(rec)) ^2 + (y - y_rec(rec)) ^2);

                    greenValue = exp(-i_zwopi_f_by_cbg * (dist_rec + dist_src));

                    pSctTemp = pSctTemp + (chi(x,y) * greenValue) ...
                      * (p_inc_ft(emit, rec, freq) + p_sct_ft(emit, rec, freq));
                end
            end
            pSct(emit,rec,freq) = pSctTemp;
        end
    end
end
```


Marcus.Hardt@kit.edu | USCT Seminar | 28.8.2009

Implementation

- Backpropagation (BP) Solution Kernel (Volume parallel)

```
%% Backward Solution
for x=x_start:x_end;
    for y=y_start:y_end
    chi_tmp = 0;
        for freq = freq_start:freq_end
            i_zwopi_f_by_cbg = i * 2 * pi * frequency / c_bg;
            for emit=1:n_src
                for receiver=1:n_rec
                    dist_src = sqrt((x - x_src(emit)^2 + (y - y_src(emit)^2);
                    dist_rec = sqrt((x - x_rec(rec))^2 + (y - y_rec(rec))^2);

                    green = exp(i_zwopi_f_by_cbg*(dist_rec+dist_src));

                    chi_tmp = chi_tmp + green ...
                        * data_sct_ft(emitter, receiver, omega);
                end
            end
        end
        chi (x-x_start+1, y-y_start+1) = chi_tmp;
    end
end
```


Marcus.Hardt@kit.edu | USCT Seminar | 28.8.2009

Implementation

- Data parallel mode was accelerated in Assembler
  - Speedup ~ factor 50 (thanks to Michael Zapf)

- We can now
  - Run simulations to create A-Scans (for given potential)
  - Run reconstructions to find original potential (from A-Scans)

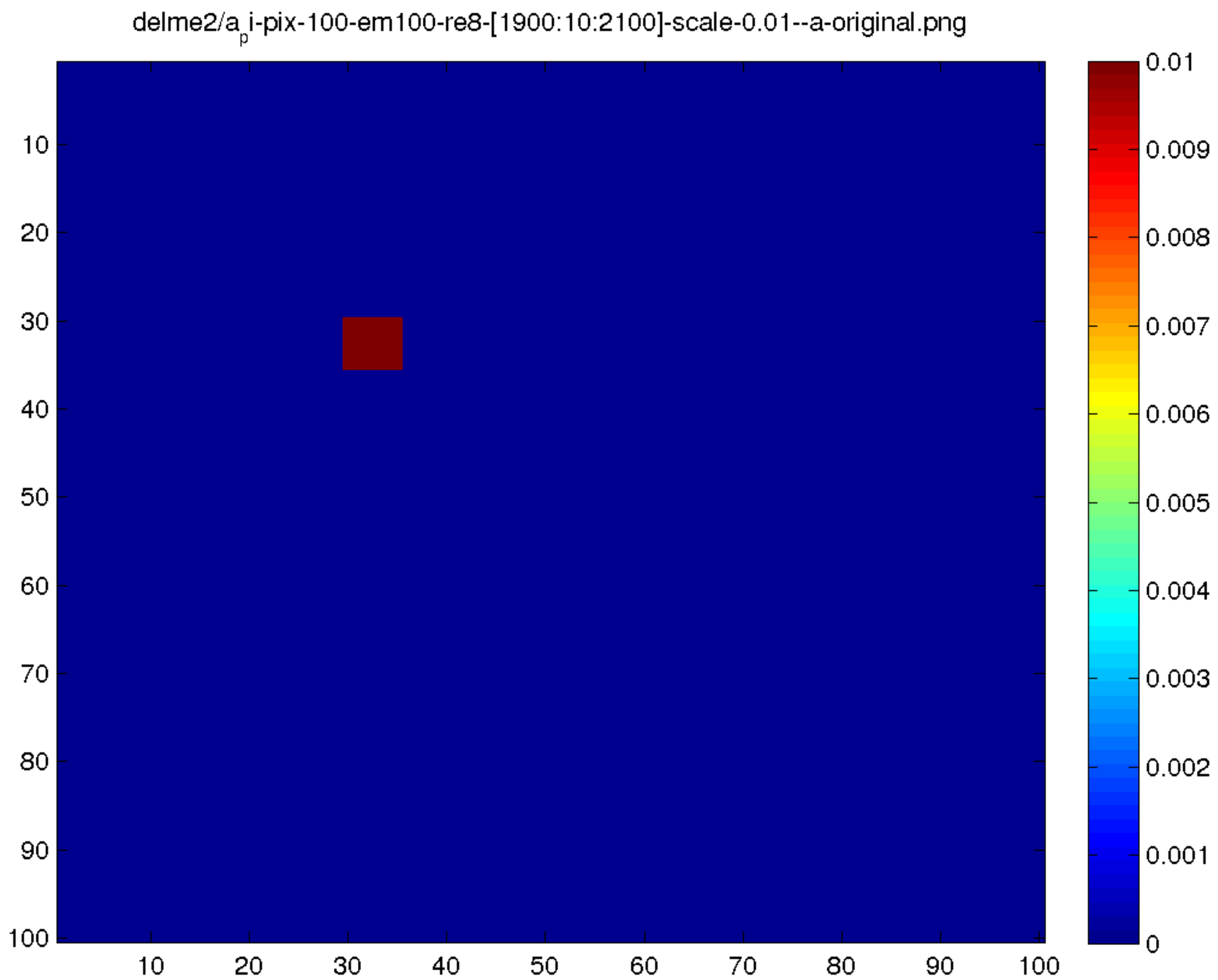
Implementation

- Data parallel mode was accelerated in Assembler
  - Speedup ~ factor 50 (thanks to Michael Zapf)

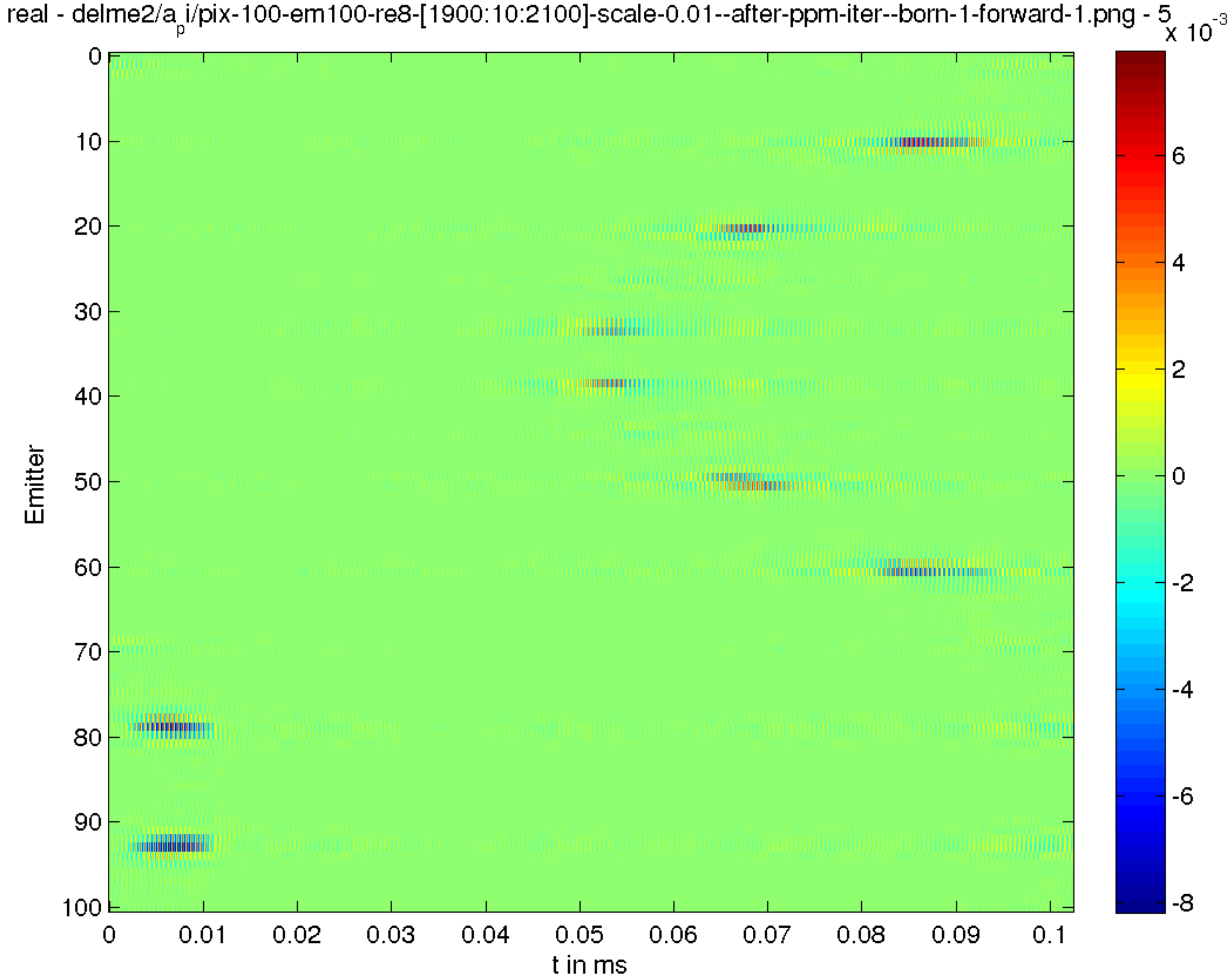
- We can now
  - Run simulations to create A-Scans (for given potential)
  - Run reconstructions to find original potential (from A-Scans)

- First images looked like these

Original Potential

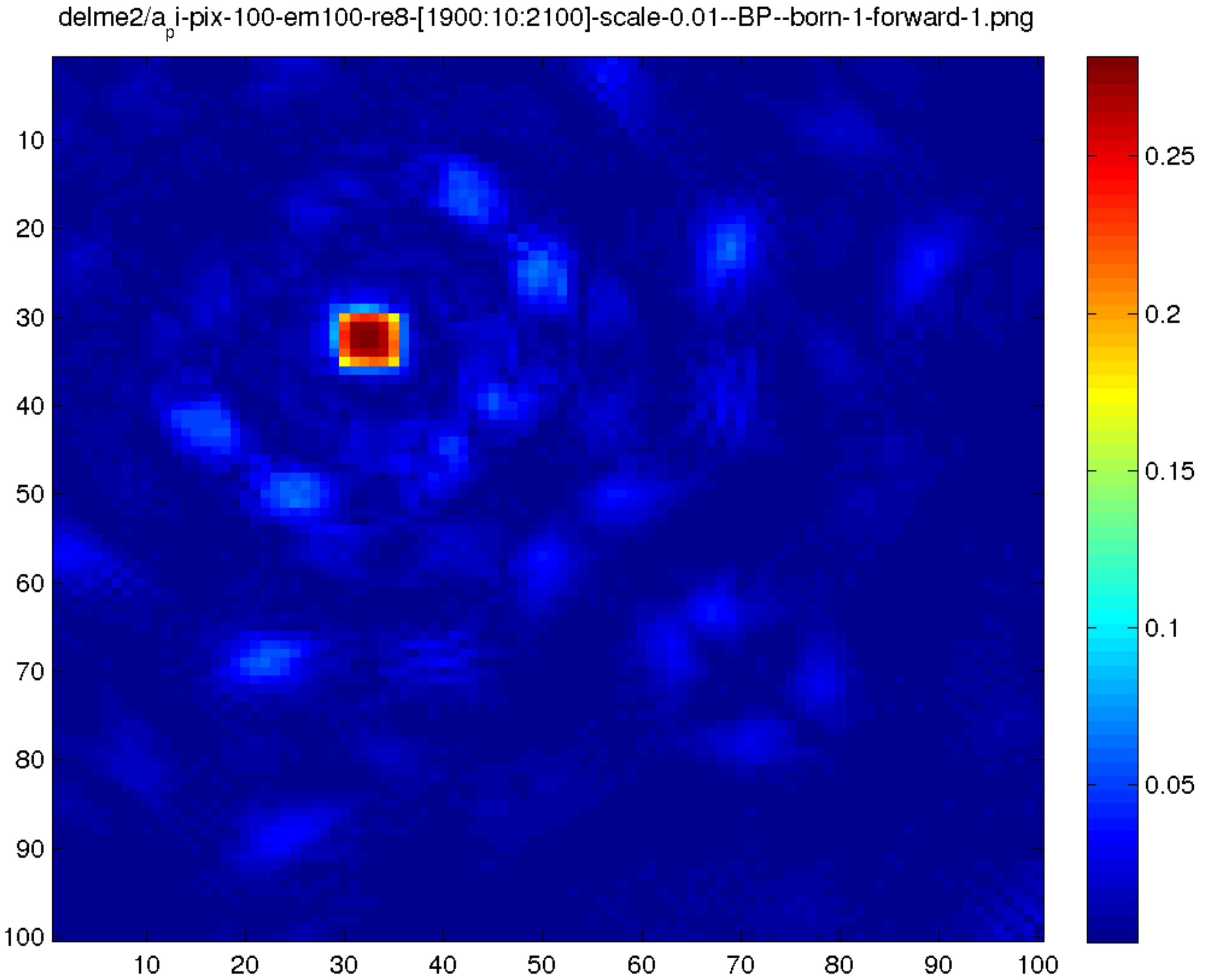


B-Scan

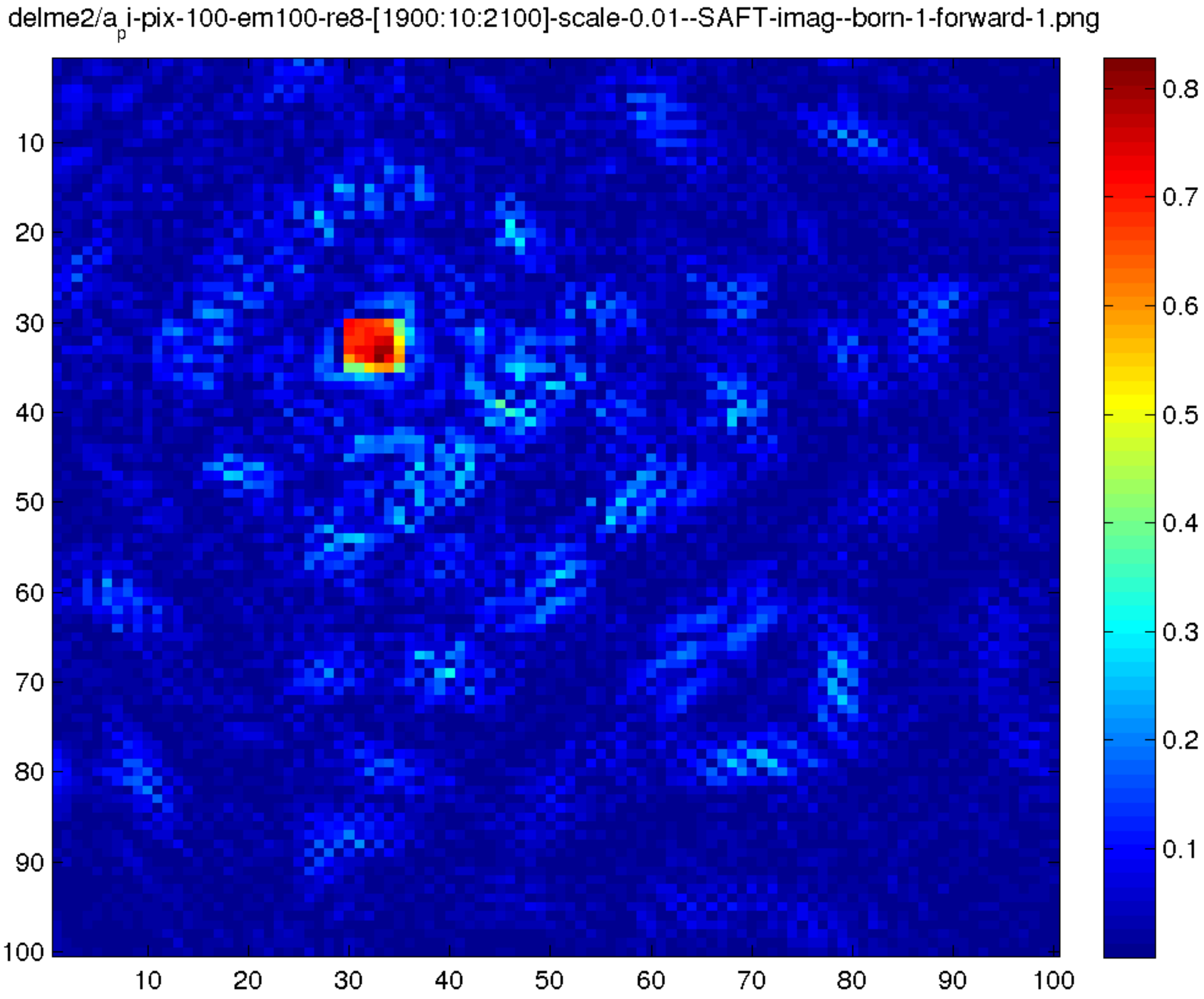


real - delme2/a_p i/pix-100-em100-re8-[1900:10:2100]-scale-0.01--after-ppm-iter--born-1-forward-1.png - 5
x 10^-3
Emitter
t in ms
0
10
20
30
40
50
60
70
80
90
100
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
6
4
2
0
-2
-4
-6
-8

Reconstruction via backprojection



Reconstruction via SAFT

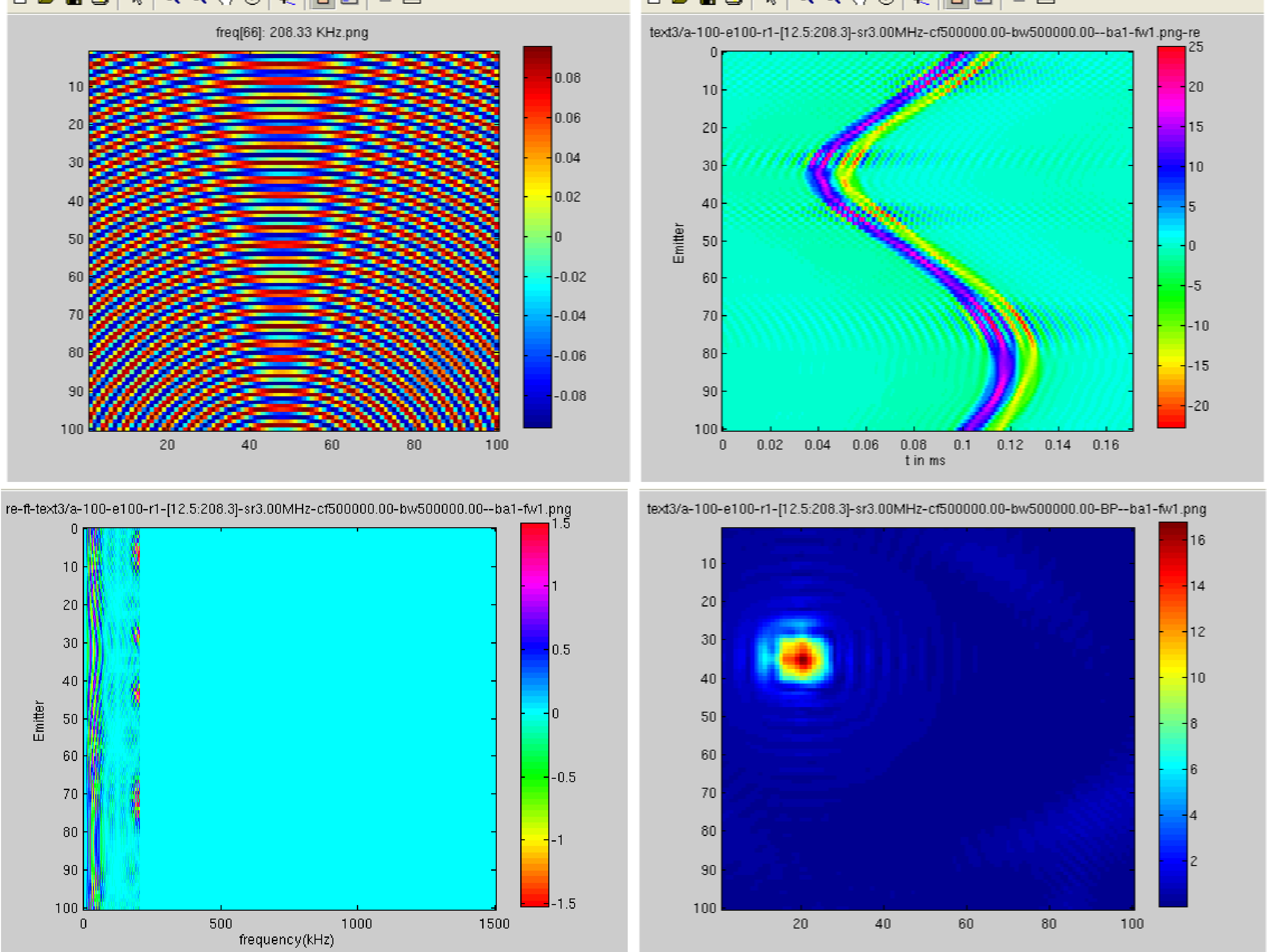


Results from first images

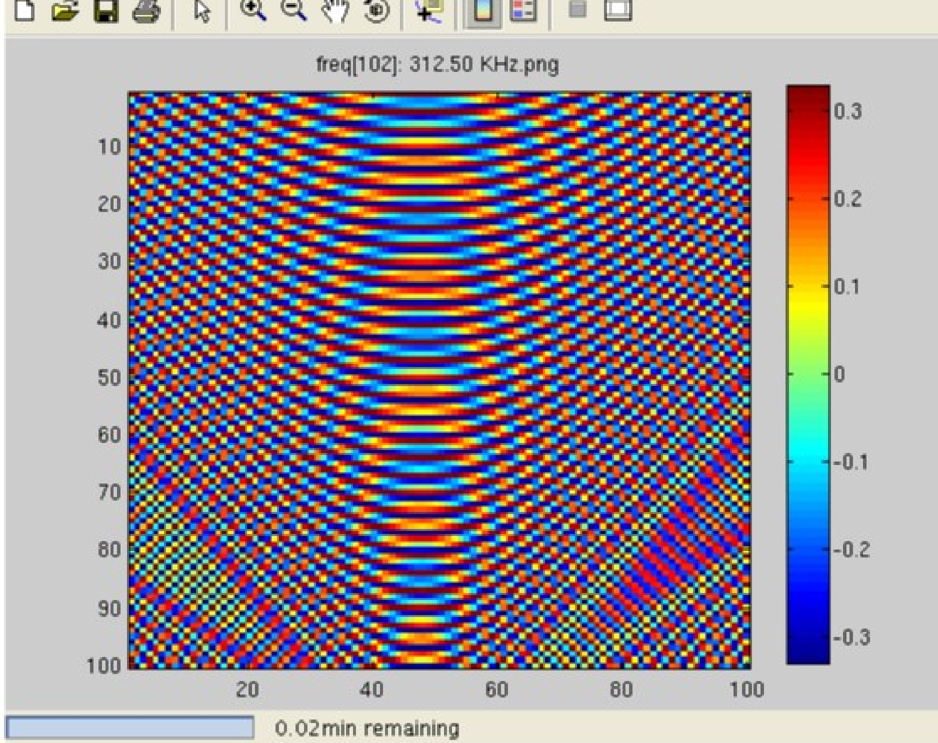
- SAFT produces similar images like BP
- Physics and Signalprocessing Theory not respected so far
  - Artefacts
  - No correlation between simulation parameters and reality
  - Impossible to compare results
- Consequence
  - Major rewrite of code (to properly respect reality)
  - Improved understanding of parameters and artefacts
  - Limits for the frequencies used

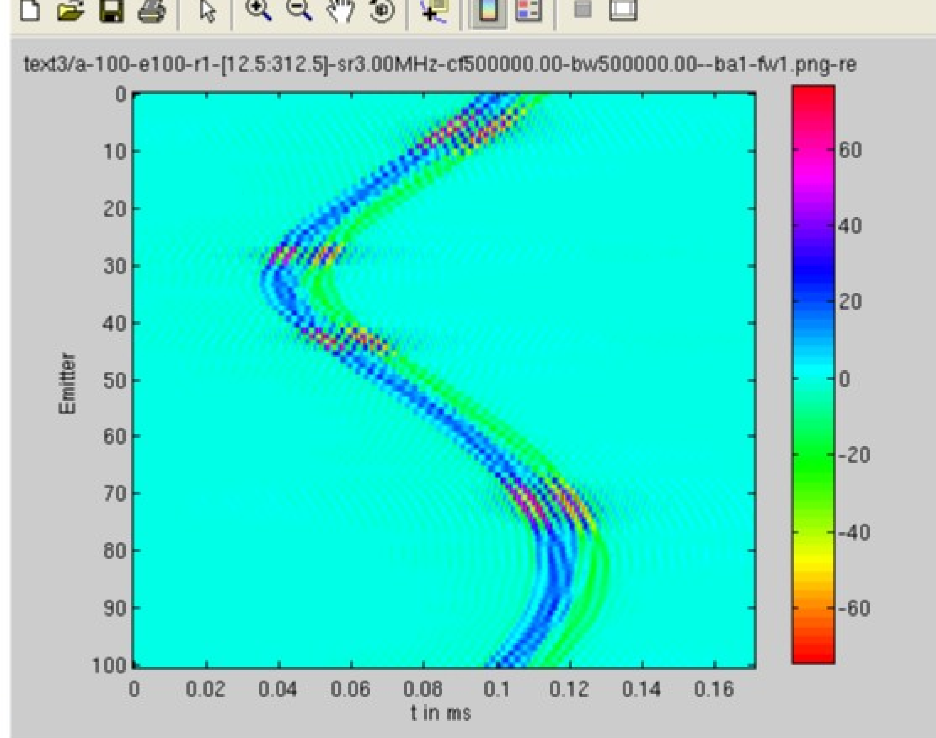
Limits for the frequency range ($c = \lambda f$)

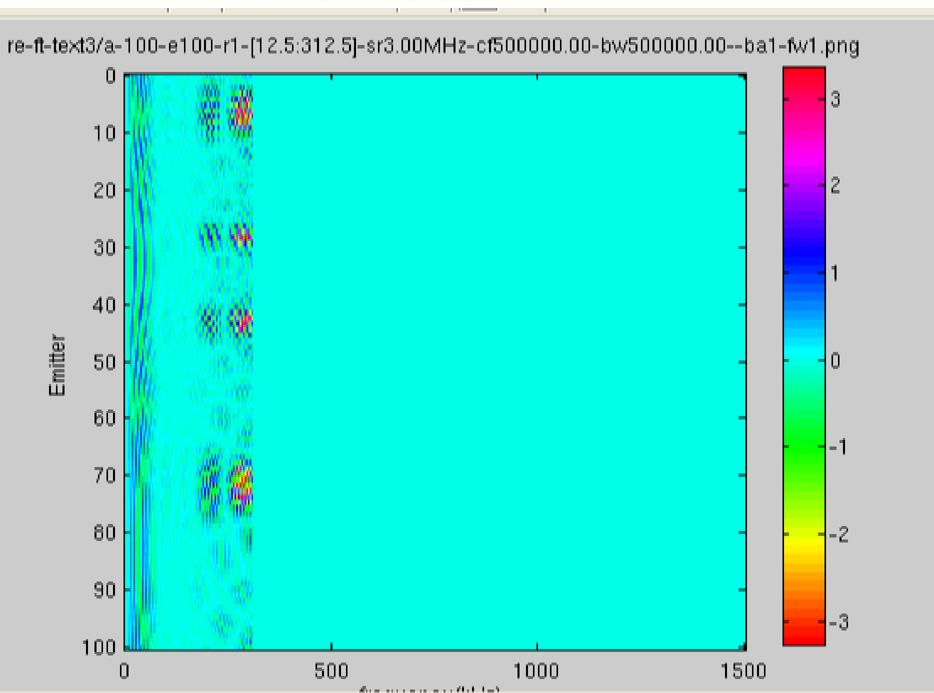
- Diameter of USCT $\lambda_{max} < D$
- Pixelsize $\lambda_{min} > 2 \cdot dr$
- Samplingrate (Nyquist) $f_{max} < \frac{1}{2} f_{sample}$
- Spacial Nyquist $f_{max} < \frac{c \cdot ROI}{2\sqrt{2} \cdot dr \cdot D}$
- Grating Lobes $\lambda_{min} > 2 * d_{Sensor-Sensor} = \frac{2\pi \cdot D}{N_{Sensors}}$
- Center frequency
  - In range of simulation frequencies
- Bandwidth
  - Small bandwidth => Long signals => Bad Images

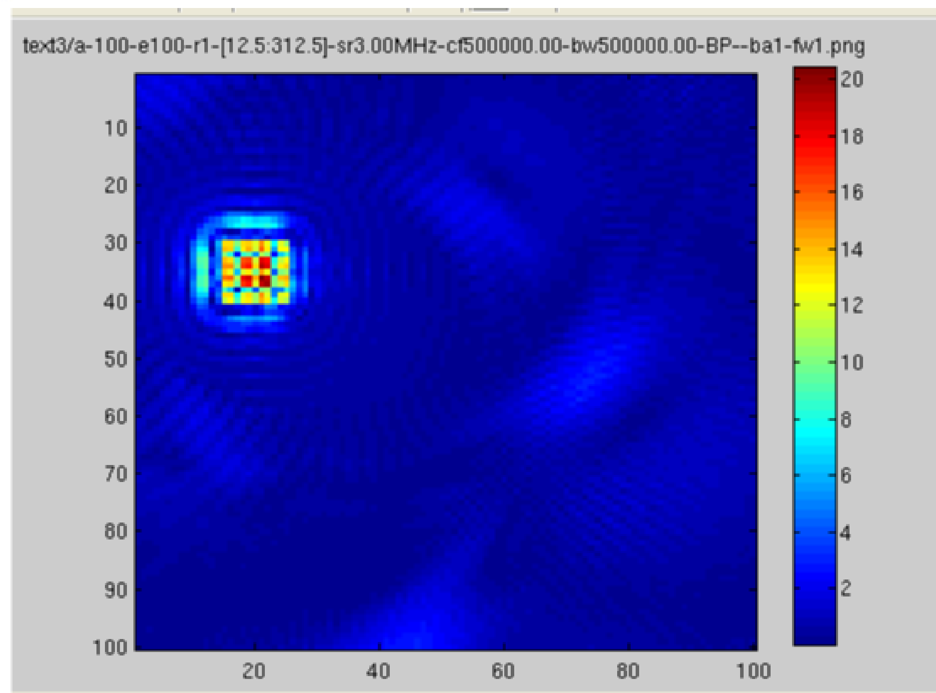


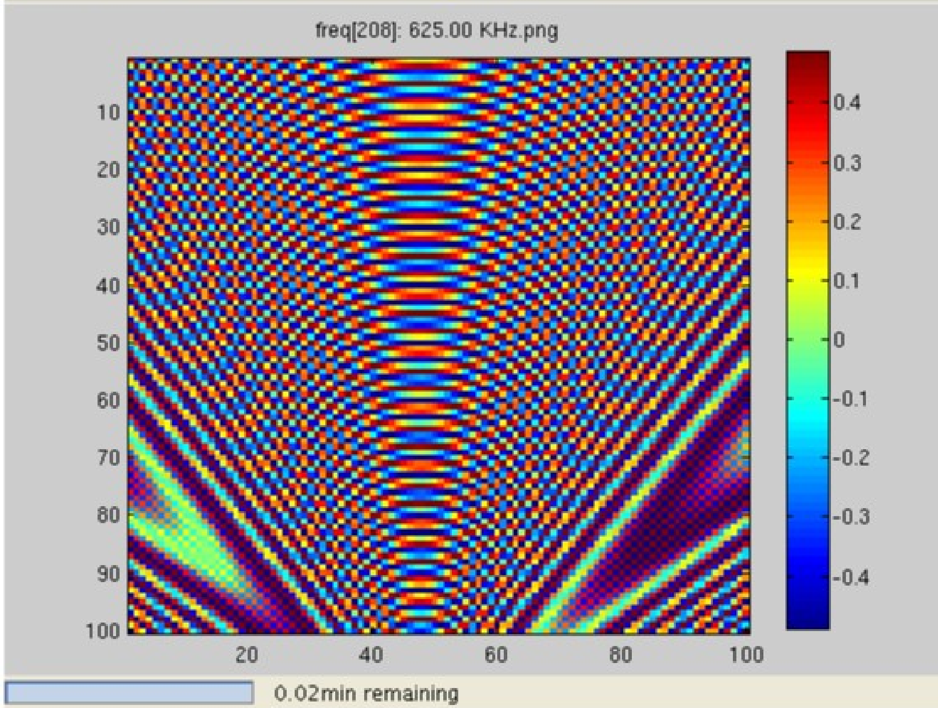
freq[66]: 208.33 KHz.png
text3/a-100-e100-r1-[12.5:208.3]-sr3.00MHz-cf500000.00-bw500000.00--ba1-fw1.png-re
Emitter
t in ms
re-ft-text3/a-100-e100-r1-[12.5:208.3]-sr3.00MHz-cf500000.00-bw500000.00--ba1-fw1.png
Emitter
frequency(kHz)
text3/a-100-e100-r1-[12.5:208.3]-sr3.00MHz-cf500000.00-bw500000.00-BP--ba1-fw1.png

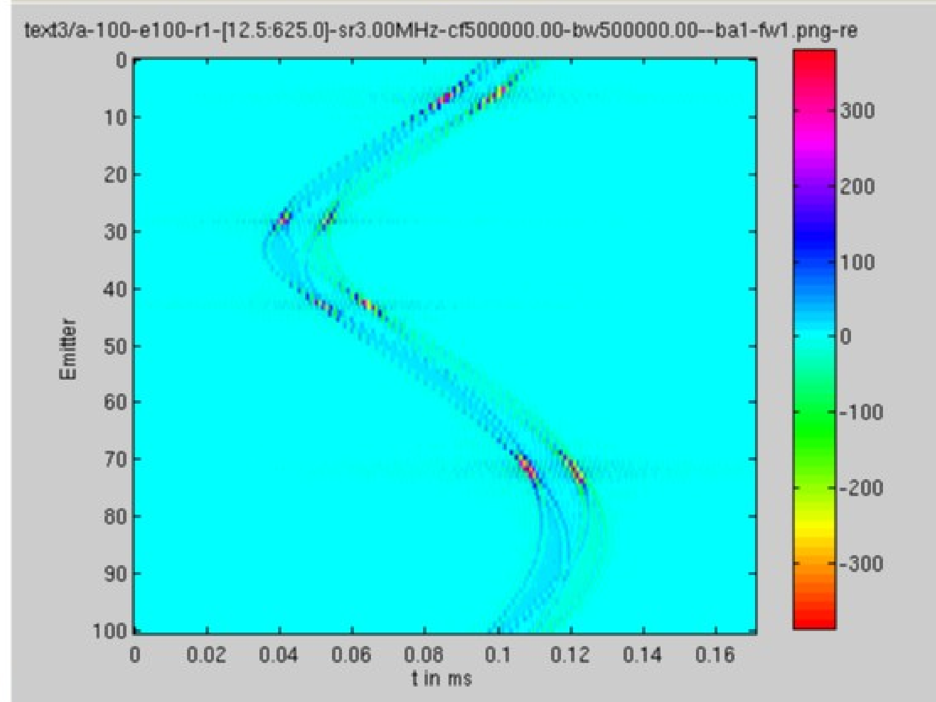

freq[102]: 312.50 KHz.png
0.02min remaining

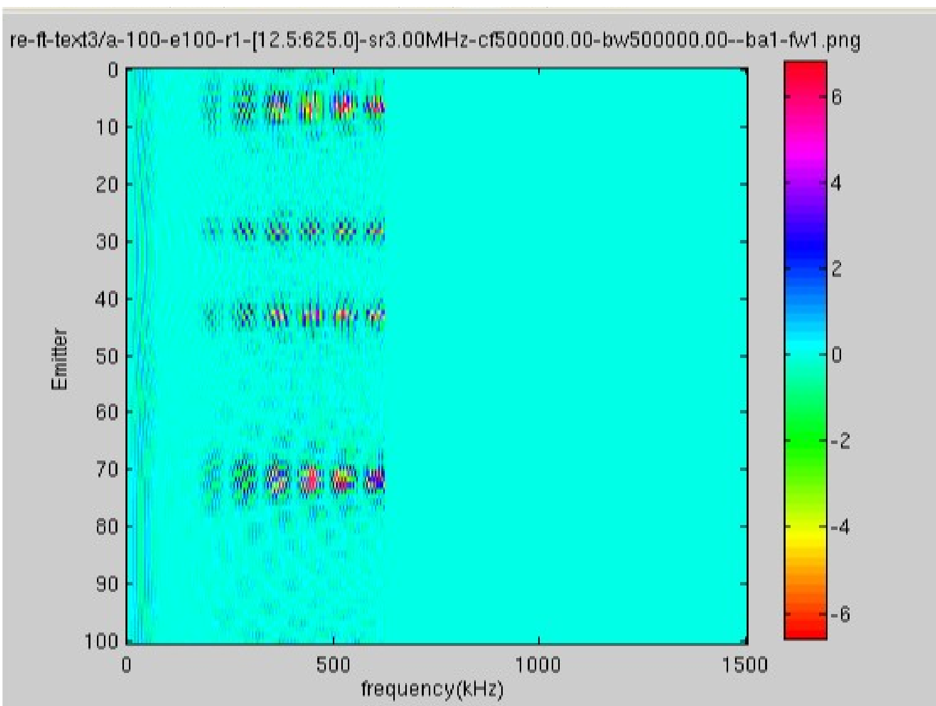

text3/a-100-e100-r1-[12.5:312.5]-sr3.00MHz-cf500000.00-bw500000.00--ba1-fw1.png-re
Emitter
t in ms

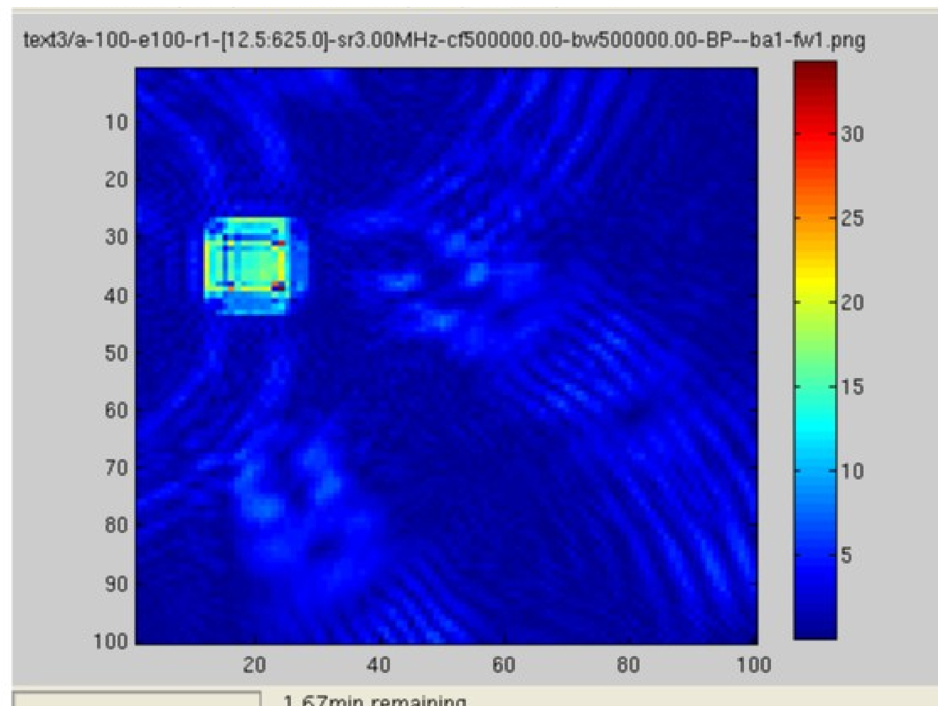

re-ft-text3/a-100-e100-r1-[12.5:312.5]-sr3.00MHz-cf500000.00-bw500000.00--ba1-fw1.png
Emitter


text3/a-100-e100-r1-[12.5:312.5]-sr3.00MHz-cf500000.00-bw500000.00-BP--ba1-fw1.png

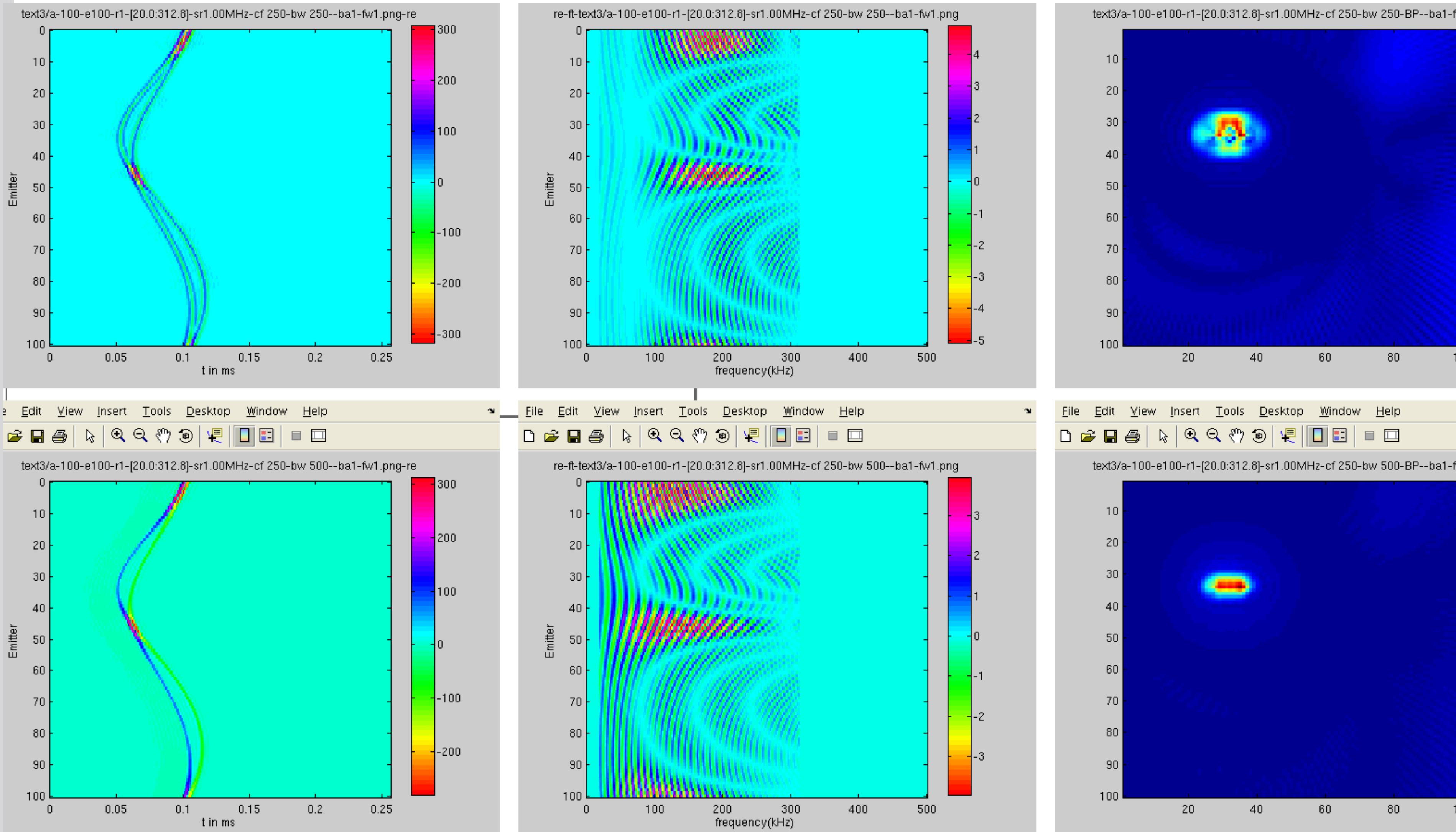

freq[208]: 625.00 KHz.png
0.02min remaining

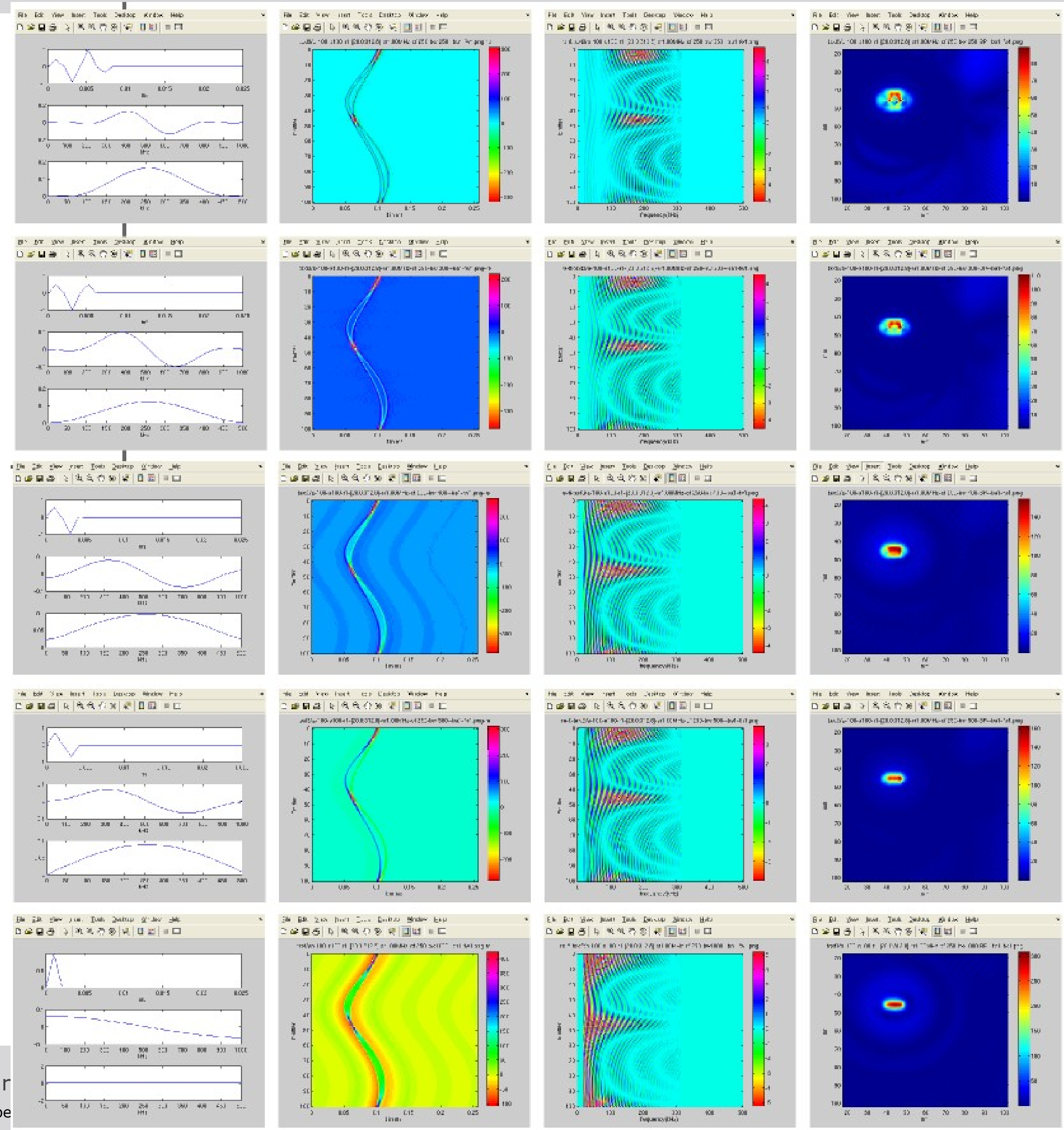

text3/a-100-e100-r1-[12.5:625.0]-sr3.00MHz-cf500000.00-bw500000.00--ba1-fw1.png-re
Emitter
t in ms


re-ft-text3/a-100-e100-r1-[12.5:625.0]-sr3.00MHz-cf500000.00-bw500000.00--ba1-fw1.png
Emitter
frequency(kHz)


text3/a-100-e100-r1-[12.5:625.0]-sr3.00MHz-cf500000.00-bw500000.00-BP--ba1-fw1.png
1.67min remaining

Variation of the Bandwidth

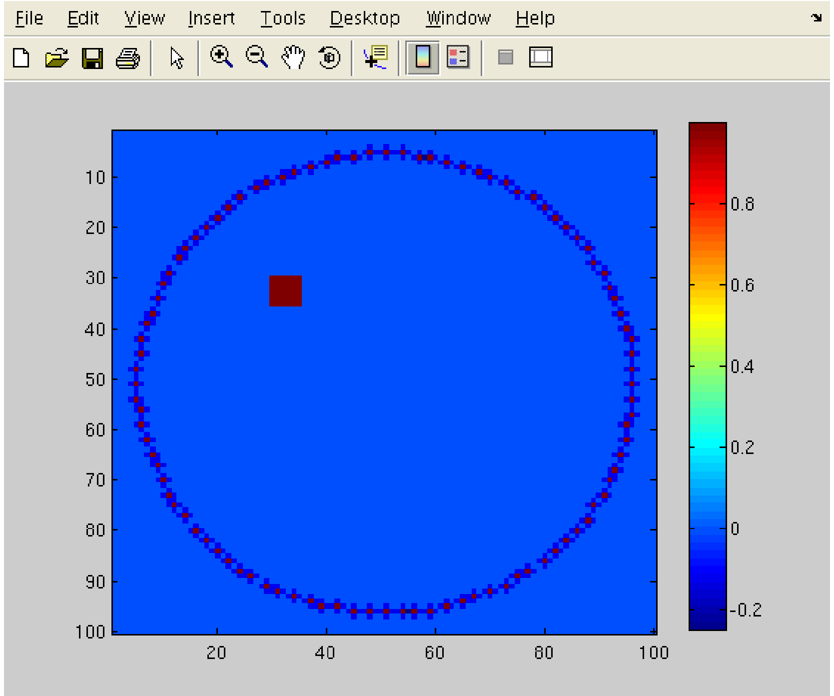


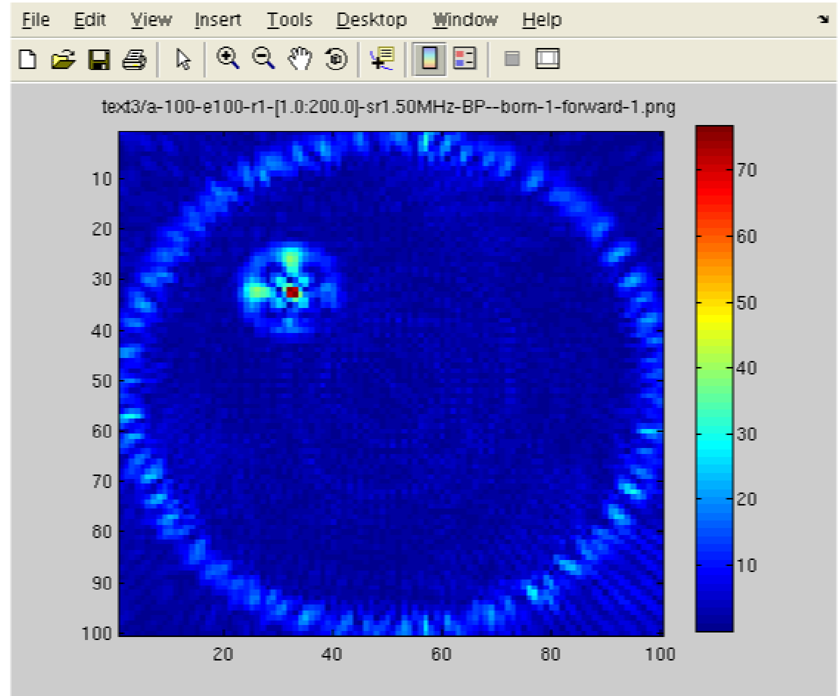


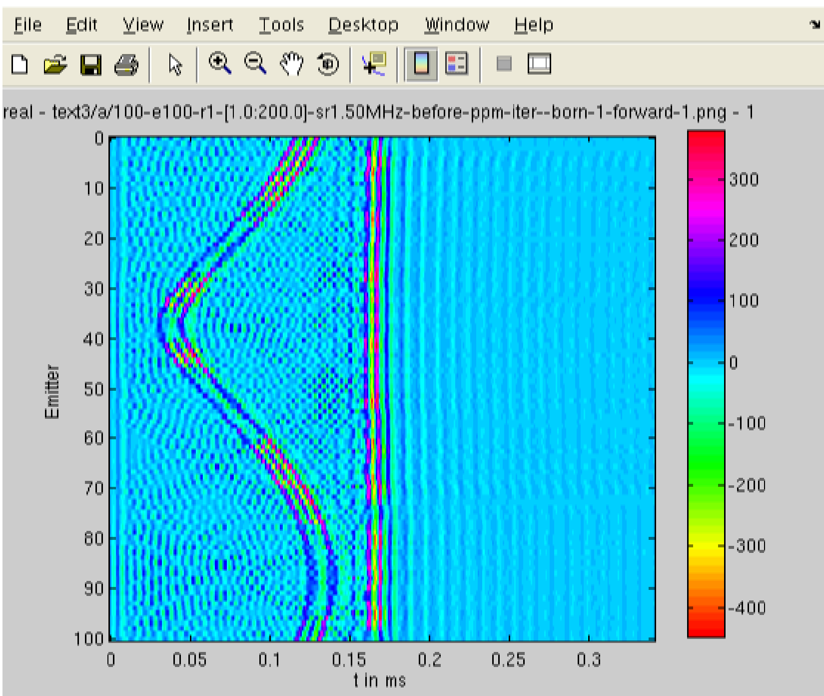
Parameters defined for verification with Brno

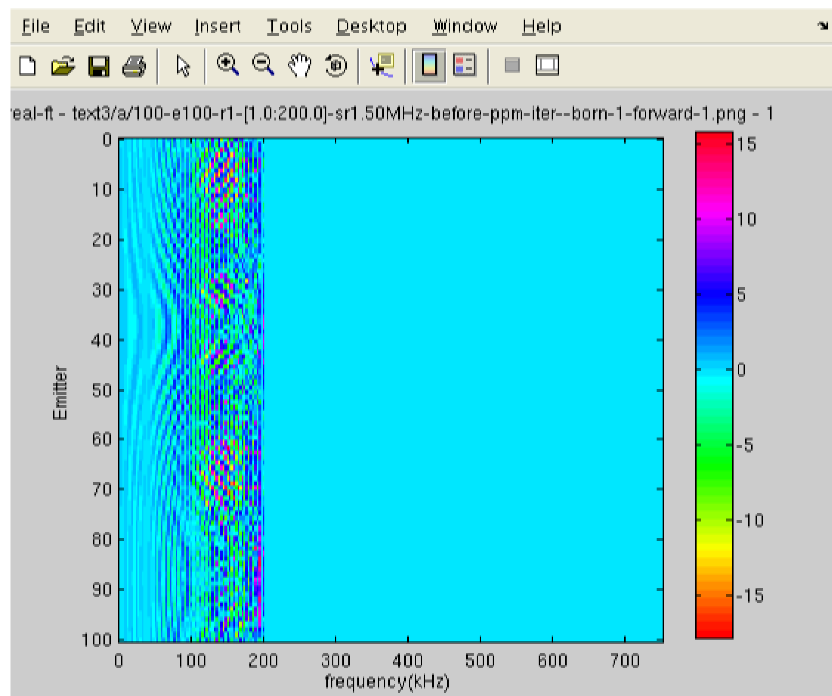
Karlsruhe Institute of Technology

- Samplerate: 1 MHz
- Resolution: 1 x 100 x 100
- Centerfrequency: 250 kHz
- Bandwidth: 500 kHz
- Simulation range: 20 – 313 kHz (149 steps)


File Edit View Insert Tools Desktop Window Help
10 20 30 40 50 60 70 80 90 100
20 40 60 80 100
0.8 0.6 0.4 0.2 0 -0.2


File Edit View Insert Tools Desktop Window Help
text3/a-100-e100-r1-[1.0:200.0]-sr1.50MHz-BP--born-1-forward-1.png
10 20 30 40 50 60 70 80 90 100
20 40 60 80 100
70 60 50 40 30 20 10


File Edit View Insert Tools Desktop Window Help
real - text3/a/100-e100-r1-[1.0:200.0]-sr1.50MHz-before-ppm-iter--born-1-forward-1.png - 1
Emitter
0 10 20 30 40 50 60 70 80 90 100
0 0.05 0.1 0.15 0.2 0.25 0.3
t in ms
300 200 100 0 -100 -200 -300 -400


File Edit View Insert Tools Desktop Window Help
real-ft - text3/a/100-e100-r1-[1.0:200.0]-sr1.50MHz-before-ppm-iter--born-1-forward-1.png - 1
Emitter
0 10 20 30 40 50 60 70 80 90 100
0 100 200 300 400 500 600 700
frequency(kHz)
15 10 5 0 -5 -10 -15

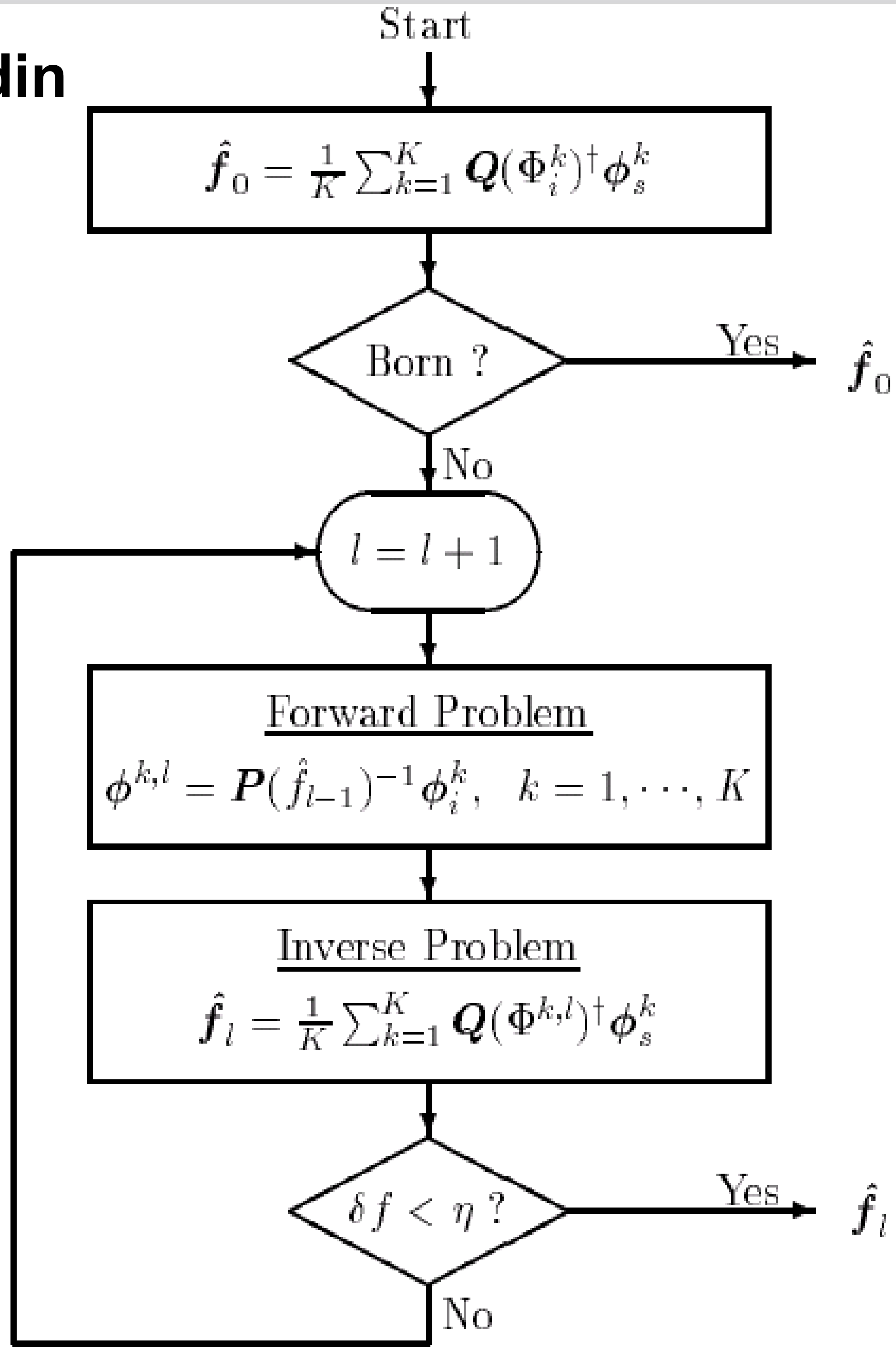
Part IV:
Multiple Scattering (work in progress)

- In principle (=literature) very simple:
  - Iterations over forward and backward solution
  - Changes are propagated via updates to the image and via the measured field

Marcus.Hardt@kit.edu | USCT Seminar | 28.8.2009

Based on PhD of O.S.Haddadin

1. Compute inverse solution (image) from measured data
2. Compute forward solution to update the total pressure
3. Compute inverse solution with updated pressure



- I know, this causes nightmares for experimentalists...

My formulation of the iterative scheme

- Correction of the assumption of the total field by solving the forward problem
- Re-computing the inverse problem

$$\hat{p}^{sct,(l)}_{\kappa,Born} = \int G\chi_\kappa^{(l-1)}(\hat{p}^{inc} + \hat{p}^{sct,(l-1)}_{\kappa,Born})dV$$

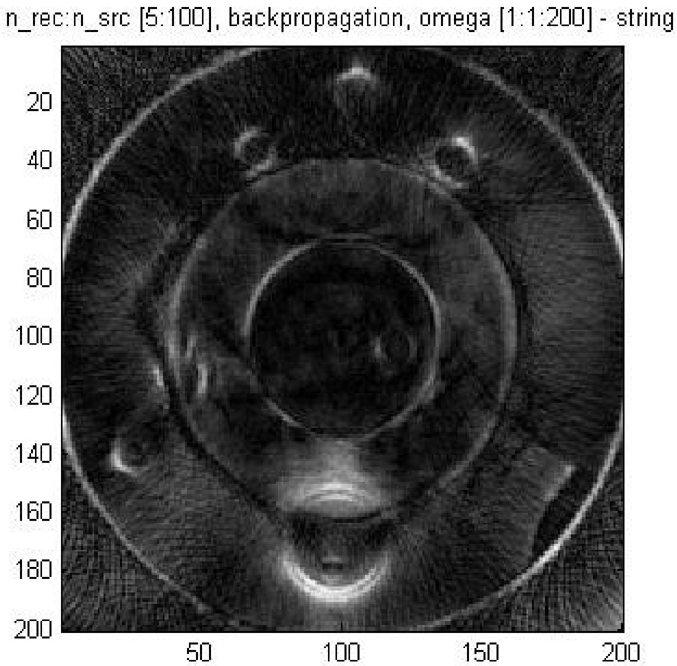
$$\Delta\chi_\kappa^{(l)} = \sum_{S,R,\omega} \left(G\hat{p}^{inc}\right)^* \hat{p}^{sct,(l-1)}_{\kappa,Born}$$

Iterations $l = 1, \ldots, L$

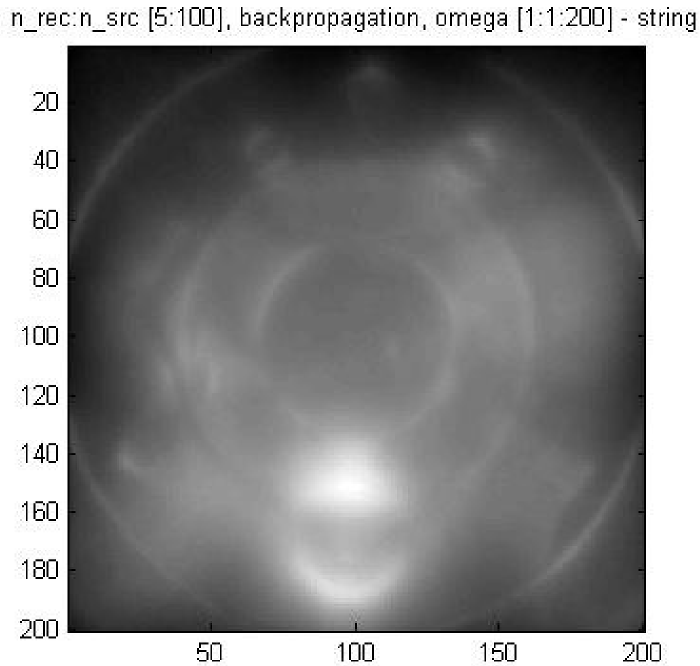
- This is basically ”averaging”

First Results

- „Milk-effect“ ☹



Iteration 0



Iteration 1

Current work

- Modification of the **forward solution**:
  - Interpretation of:

$$\hat{p}^{sct,(l)}_{\kappa,Born} = \int G\chi^{(l-1)}_{\kappa}(\hat{p}^{inc} + \hat{p}^{sct,(l-1)}_{\kappa,Born})dV$$

  - Previously: Assumed surface integral over ”dV”
  - Now: Volume Integral inside ”dV”
    => Simulate pressure at every pixel
    (instead of: at every receiver)
    - For 1 x 100 x 100 Voxels: Factor 10.000 slower

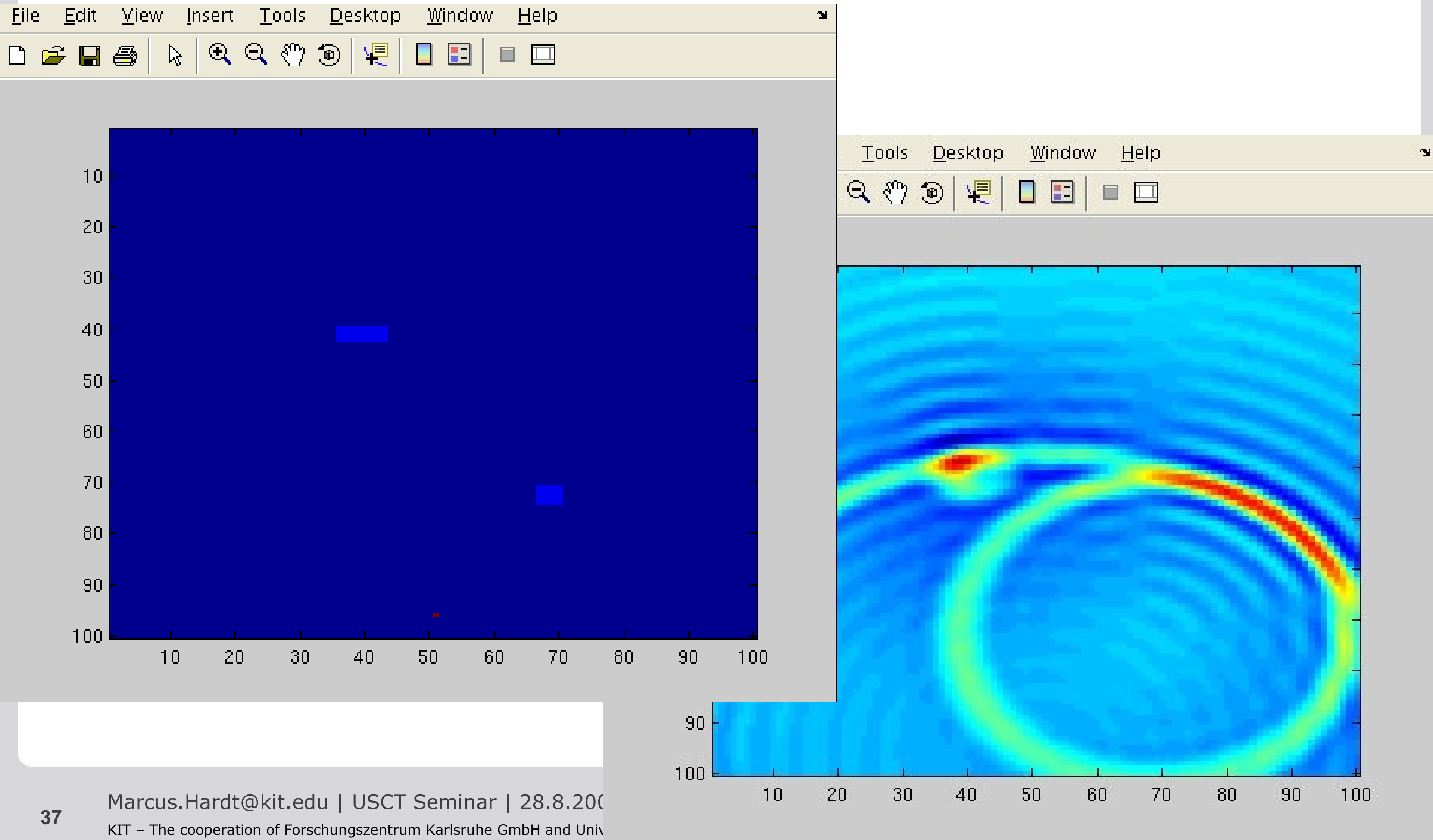
=> Able to ”Send off” secondary waves in **forward iterations**

Marcus.Hardt@kit.edu | USCT Seminar | 28.8.2009

Karlsruhe Institute of Technology

Example

- file://home/marcus/grid/talks/2009-08-28-IPE-Seminar



Conclusions

- Parallel computing facilities integerated into Matlab
- Simulations correlate to realistic parameters
- Reconstructions of BP and SAFT show similar images
- First order scattering can be simulated
  - Higher order scattering in progress
- No noise yet
- No Matrix operations required

Questions?