SIPPI

Ed. version 1.0

COLLABORATORS

	TITLE : SIPPI	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Thomas Mejer Hansen and Knud Skou Cordua	April 3, 2014	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

About

SIPPI is a Matlab toolbox (compatible with GNU Octave) that allow sampling the solution of non-linear inverse problems with realistic a priori information.

In order to make use of SIPPI one has to

- Install and setup SIPPI
- Define the prior model, in form of the prior data structure
- Define the forward model, in form of the forward data structure, and the sippi_forward.m m-file
- Define the data and noise model, in form of the prior data structure
- Choose a method for sampling the a posteriori probability density.

Chapter 1

Installation

1.1 SIPPI

Download the latest version of SIPPI from http://sippi.sourceforge.net.

Unpack ZIPPI_1.0.zip somewhere, for example to 'c:\Users\tmh\SIPPI'. Then setup the Matlab path to point to the appropriate SIPPI directories:

```
addpath c:\Users\tmh\SIPPI
sippi_set_path
```

1.1.1 SGeMS (optional)

To make use of the SISIM and SNESIM type priori models SGeMS needs to be available.

Currently only SGeMS version 2.1 (download) for Windows is supported.

Chapter 2

Setting up SIPPI

2.1 The a priori model

2.2 The data and the noise

2.3 The forward model

Chapter 3

The a posteriori distribution

3.1 Sampling the a posteriori probability density

3.1.1 The rejection sampler

3.1.2 The extended Metropolis sampler

3.1.2.1 The extended independent Metropolis sampler

3.1.3 linear least squares

3.2 Simulated Annealing

Chapter 4

Examples

4.1 Line fitting

The forward problem consists of computing the y-value as a function of the x-position of the data, and the polynomial coefficients determining the line. sippi_forward_linefit.m:

```
% sippi_forward_linefit Line fit forward solver for SIPPI
%
% [d,forward,prior,data]=sippi_forward_linefit(m,forward,prior,data);
%
function [d,forward,prior,data]=sippi_forward_linefit(m,forward,prior,data);

if length(m)==1;
    d{1}=forward.x*m{1};
elseif length(m)==2;
    d{1}=forward.x*m{1}+m{2};
else
    d{1}=forward.x.^2*m{1}+forward.x*m{2}+m{3};
end
```

4.2 Covariance model inference

4.3 Cross hole tomography

4.4 Reflection seismic inversion