MPSpack user manual

Alex Barnett*and Timo Betcke†

July 17, 2009

Abstract

MPSpack is a fully object-oriented MATLAB toolbox for solving Laplace, Helmholtz, wave scattering, and related PDE boundary-value problems on piecewise-homogeneous 2D domains. The philosophy is to use basis functions which are particular solutions to the PDE in some region; solving is thus reduced to matching on the boundary (or on boundaries of subregions). This idea is known as the Method of Particular Solutions, or as Trefftz, ultra-weak, or non-polynomial methods in the FEM community. Basis functions include plane-wave, Fourier-Bessel, corner-adapted expansions, fundamental solutions. Layer potential representations and associated singular quadrature schemes are also integrated into this framework. It is designed to be simple to use, and to enable highly-accurate solutions. This is the user manual; for a more hands-on approach try the accompanying tutorial.

1 Overview

In numerical analysis there has been a recent flurry of activity on methods for solving PDEs where solutions are approximated by linear combinations of particular solutions to the PDE with good convergence properties. These methods are high-order (often exponentially convergent), efficient at high frequencies (the number of degrees of freedom scales linearly with wavenumber, in 2D), and are quite simple to implement. When geometries become more complicated, the domain needs to be split up into multiple subdomains, e.g.

*Department of Mathematics, Dartmouth College, Hanover, NH, 03755, USA
†Department of Mathematics, University of Reading, Berkshire, RG6 6AX, UK

one for each corner, and the implementation becomes cumbersome. The goal of this software toolbox is to make such an implementation simple and transparent, and create an intuitive, unifying framework in which many types of BVP, basis sets, boundary conditions, and domain geometries may be solved, explored and graphed with ease. Since in these methods, the number of subdomains is small and fixed, we leave the job of specifying subdomains, i.e. 'meshing', to the user (this contrasts with the philosophy of the large number of finite-element packages already in existence.)

We are most interested in the scalar Helmholtz equation in the plane,

$$(\Delta + k^2)u \;=\; 0 \qquad \text{in } \Omega \,, \tag{1}$$

where $\Omega \subset \mathbb{R}^2$ is an interior or exterior domain, k is the wavenumber, and certain imhomogeneous boundary conditions are imposed. Such problems arise in wave scattering and cavity resonances. More generally we may have multiple domains with different wavenumbers connected by homogeneous or inhomogeneous boundary conditions, as in transmission, dielectric-coated, or photonic crystal problems. With $k = 0$ we have Laplace's equation, with applications to electrostatics, steady-state heat flow, and probability.

In this release we discuss BVPs. It is a small extension to compute eigenvalue problems and periodic problems; we will document these in future releases.

Fig. 1 shows how a geometry is built up from objects, and the relationships between them.

Our influences include Driscoll's SC Toolbox, and the `chebfun` system by Trefethen et al.

1.1 Installation

Requirements:

- MATLAB version 7.6 (2008a) or newer is needed, since we make heavy use of recent object-oriented programming features.

The project is hosted at

`http://code.google.com/p/mpspack`

There are two alternative methods to download and unpack:

1. Get a gzip-compressed tar archive from

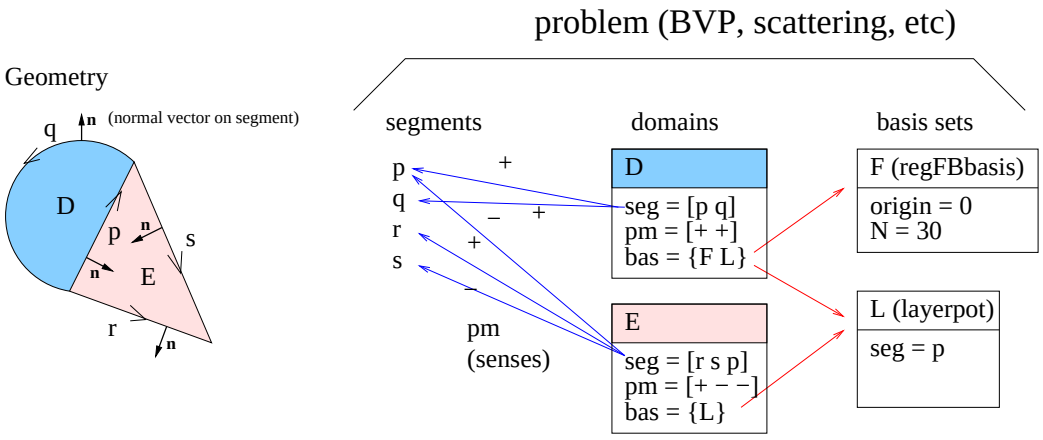

Figure 1: Relationship of `MPSpack` objects used to solve a typical BVP or EVP. The physical geometry is shown on the left, and the code structures on the right. Each segment has a normal direction. Each domain is built from segments with a list of signs (+ or −) which define the sense in which the segments are used to form the boundary. Basis sets affect the function values inside each domain in which they are referenced. A basis set may influence more than one domain.

`http://code.google.com/p/mpspack/downloads/list`

In a UNIX environment you may now unpack this with

`tar zxvf mpspack-r*.tar.gz`

This creates the directory `mpspack-r*` containing the toolbox, where `*` represents the revision (version) number.

2. First install the `subversion` (SVN) version control software from

   `http://subversion.tigris.org`

   Anonymous check out of `MPSpack` is then via the subversion command:

   `svn checkout http://mpspack.googlecode.com/svn/trunk/ mpspack`

   This creates a directory `mpspack` containing the toolbox.

   You might prefer a more user-friendly graphical subversion client such as those listed at

   `http://subversion.tigris.org/links.html#clients`

There are some optional fast basis evaluation routines (C and Fortran with MEX interfaces), which should be compiled in a UNIX environment as follows: from the `mpspack` directory type `make`

You should now add the `mpspack` subdirectory to your MATLAB path, for instance by adding the line

```
addpath 'path/to/mpspack';
```

to your MATLAB `startup.m` file. You are now ready to use `MPSpack` !

1.2 What kind of problems does `MPSpack` solve?

Let $\Omega \subset \mathbb{R}^2$

$$(\Delta + k^2)u = 0 \qquad \text{in } \Omega \tag{2}$$

$$au + bu_n = f \qquad \text{on } \partial\Omega \tag{3}$$

In our implementation we restrict to functions a, b that are constant on segments used to define the boundary $\partial\Omega$.

If Ω is an exterior domain, we may wish to impose additional boundary conditions at infinity, such as Sommerfeld's radiation condition. This may be achieved by choosing basis sets satisfying this condition.

See [1] for an example of using the Method of Fundamental Solutions to solve an interior Helmholtz problem.

To solve a BVP the usual flow is as follows:

1. define piecewise-analytic segments forming all boundaries
2. define domains using various of these segments as their boundaries
3. choose MPS basis set(s) within each domain
4. set up inhomogeneous boundary or matching conditions on each segment
5. solve a linear system to get the basis coefficient vector
6. evaluate solution on desired points
7. plot solution or compute error estimates

The accompanying tutorial document is a good way to be taken through these steps in the context of examples. The rest of this manual documents in a more theoretical way the data structures, organized by topic such as sements, domains, etc.

2 Representing geometry

Fig ** outlines the objects.

A segment is specified as a parametrized function $z(t)$ for $0 \leq t \leq 1$, and $z'(t)$ is also needed. From this all information follows such as locations and normal vectors at quadrature points. Since our focus is on high-order and accurate methods, we feel that forcing the user to go to the trouble of providing z and z' is reasonable.[1] The benefit is that the use may simply switch between quadrature point numbers and types.

When domains are built from segments, each segment stores information about which domain, if any, it is connected to on each side. If segments are to be reused to construct new domains, while erasing any old domains, the current way to do this is via the `s.disconnect` command, where `s` is a segment handle or array of such. This clears these segments of any connections with domains.

The segment object field responsible for this is `s.dom`, which is a 1-by-2 cell array of handles to the domain on the segment's natural normal side and back side respectively. If there is no domain on a particular side, the cell element is empty.

2.1 Boundary conditions

A segment Γ may contain a boundary condition on only one of its sides, of the form

$$au(s) + bu_n(s) = f(s) \qquad s \in \Gamma, \quad a, b \in \mathbb{C}$$

or, alternatively matching conditions of the form

$$a^+u^+(s) + a^-u^-(s) = f(s) \tag{4}$$
$$b^+u_n^+(s) + b^-u_n^-(s) = g(s) \tag{5}$$

[1] In future releases we may allow z and z' to be automatically generated as high-order polynomial fits to a set of boundary points such as might be available from an engineering or CAD package.

where $a^+, a^-, b^+, b^- \in \mathbb{C}$, and where u^+ is the limiting value approaching the segment from its positive normal side. In the above f and g are functions on the boundary. In our implementation the user may supply these as handles to functions of segment parameter $t \in [0, 1]$, or as a data column vector on the segment quadrature points.

The segment fields `a` and `b` store a and b for a boundary condition, or $[a^+, a^-]$ and $[b^+, b^-]$ respectively for a matching condition.

3 Basis sets and problem classes

FIX THIS TO REFLECT NEW basis.doms[] PROPERTY:

Each basis set has one (or possibly more) domain that it affects.

[2]. However, for a basis set to influence a domain `d`, its handle must appear in the cell array field `d.bas` This is set up automatically when you perform `d.addregfbbasis()` or any of the other

There will be more elaborate basis set types which naturally affect two domains, such as dielectric corner and transmission layer potential objects; their handles will appear in the `bas` cell array of more than one domain.

4 In-depth examples

5 Test routines

The code `testbvp.m` shows the main steps for solution of a BVP on a variety of domains.

6 Limitations

- Checking of whether inside a domain is approximate, based on approximating polygons.

[2]apart from layer potentials which require a segment on which their source density is placed

7 List of improvements to make

List to be prioritized:

- make `segment.bdryfunc` which creates u, u_n data given a field function and its gradient field function, useful for choosing BC data corresponding to exact fields. DONE - see tutorial. Vectorize it over s?
- make `domain.setbc` which uses one BC data function on all segments
- eigenvalue problems: MPS, scaling method
- segment methods to create analytic interpolant function from boundary point data, enabling user to specify a segment using points on a curve, as in Greengard.
- add better automated ways to choose MFS charge points, based on JCP paper
- write `segment.bdrysolution` which evaluates u, u_n on one or other side of a boundary. For layer potentials this would take into account jump relations. This should then be used in `problem.fillbcmatrix`
- keep discrepancy and evaluation matrices, for efficiency in multiple right-hand sides.
- classes which symmetrize basis sets for single reflection, C_4, etc symmetry, by wrapping the calls to basis evaluations using reflection points. This is only needed for eigenvalue problems.
- periodic boundary conditions connecting values and derivatives on given sides of two different segments, needed for photonic crystals
- make better addnufbbasis which attaches to a requested corner.
- basisgeom for nufbbasis, split off utils.plotfan from domain.plot and use.

8 Known bugs

Please alert the authors to any bugs you discover, including a description of how to reproduce the behavior, using the interface at

`http://code.google.com/p/mpspack/issues/list`

References

[1] A. H. Barnett and T. Betcke, *Stability and convergence of the Method of Fundamental Solutions for Helmholtz problems on analytic domains*, J. Comput. Phys., *in press* (2008). `arXiv:0708.3533 [math.NA]`.