

PLTMG: A SOFTWARE PACKAGE FOR SOLVING ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

RANDOLPH E. BANK*

Abstract. *PLTMG 9.0* is a package for solving elliptic partial differential equations in general regions of the plane. It is based on continuous piecewise linear triangular finite elements, and features adaptive local mesh refinement, multigraph iteration, and pseudo-arclength continuation options for parameter dependencies. It also provides options for solving several classes of optimal control and obstacle problems. The package includes an initial mesh generator and several graphics packages. Support for the Bank-Holst parallel adaptive meshing strategy is also provided.

PLTMG is provided as Fortran (and a little C) source code, in both single and double precision versions. The code has interfaces to X-Windows, *MPI*, and Michael Holst's OpenGL image viewer *SG*. The X-Windows, *MPI*, and *SG* interfaces require libraries that are NOT provided as part of the *PLTMG* package. *PLTMG* is available from Netlib (<http://www.netlib.org/>), Mgnet (<http://www.mgnet.org/>), and the author's homepage (<http://www.cam.ucsd.edu/~reb>). The *SG* (socket graphics) OpenGL display tool is available from Michael Holst's homepage (<http://scicomp.ucsd.edu/~mholst/codes/index.html>). *MPICH* (a free implementation of *MPI*) is available from the *MPI* homepage (<http://www-unix.mcs.anl.gov/mmpi/mpich/>).

1. Problem Specification.. Consider the elliptic boundary value problem

$$(1.1) \quad -\nabla \cdot a(x, y, u, \nabla u, \lambda) + f(x, y, u, \nabla u, \lambda) = 0 \quad \text{in } \Omega,$$

with boundary conditions

$$(1.2) \quad \begin{aligned} u &= g_2(x, y, \lambda) && \text{on } \partial\Omega_2, \\ a \cdot n &= g_1(x, y, u, \lambda) && \text{on } \partial\Omega_1, \\ u, a \cdot n &\text{ continuous} && \text{on } \partial\Omega_0. \end{aligned}$$

Here Ω is a bounded region in $\mathcal{R}^2$, n is the unit normal, a is the vector $(a_1, a_2)^t$, a_1 , a_2 , f , g_1 , and g_2 are scalar functions. $\partial\Omega_0$ is a portion of $\partial\Omega$ where periodic boundary conditions are applied. In some problems solved by *PLTMG*, the parameter λ is not used, while in others $\lambda \in \mathcal{R}$ is a scalar parameter or $\lambda \in \mathcal{H}^1(\Omega)$, where $\mathcal{H}^1(\Omega)$ denotes the usual Sobolev space. Let

$$\begin{aligned} \mathcal{H}_p^1 &= \{\phi \in \mathcal{H}^1(\Omega) \mid \phi \text{ is continuous on } \partial\Omega_0\}, \\ \mathcal{H}_g^1 &= \{\phi \in \mathcal{H}_p^1 \mid \phi = g_2 \text{ on } \partial\Omega_2\}, \\ \mathcal{H}_e^1 &= \{\phi \in \mathcal{H}_p^1 \mid \phi = 0 \text{ on } \partial\Omega_2\}. \end{aligned}$$

Then the weak form of (1.1)-(1.2) is: find $u \in \mathcal{H}_g^1$ such that

$$(1.3) \quad a(u, v) = 0 \quad \text{for all } v \in \mathcal{H}_e^1,$$

where

$$(1.4) \quad a(u, v) = \int_{\Omega} a(u, \nabla u, \lambda) \cdot \nabla v + f(u, \nabla u, \lambda) v \, dx \, dy - \int_{\partial\Omega_1} g_1(u, \lambda) v \, ds.$$

*Department of Mathematics, University of California, San Diego, La Jolla, California 92093-0112. Email: rbank@ucsd.edu. The work of this author was supported by the National Science Foundation under contract DMS-0208449. The UCSD Scicomp Beowulf cluster was built using funds provided by the National Science Foundation through SCREMS Grant 0112413, with matching funds from the University of California at San Diego.

In some problems solved by *PLTMG*, a functional $\rho(u, \lambda)$ plays an important role. Functionals we consider are of the form

$$(1.5) \quad \rho(u, \lambda) = \int_{\Omega} p_1(x, y, u, \nabla u, \lambda) dx dy + \int_{\Gamma} p_2(x, y, u, \nabla u, \lambda) ds,$$

where p_1 and p_2 are scalar functions. Here $\Gamma = \partial\Omega \cup \Gamma_0$, where Γ_0 consists of certain internal curves specified by the user.

This version of the *PLTMG* package address five major problem classes. These are briefly described below.

1.1. Elliptic Boundary Value Problem.. For this problem, *PLTMG* solves a discrete analog of (1.3). The parameter λ does not play a role in this problem. Let $\mathcal{T}$ denote a triangulation of Ω and let $\mathcal{M}$ be the space of C^0 piecewise linear polynomials associated with $\mathcal{T}$. *PLTMG* usually represents such a piecewise polynomial using the standard nodal basis; a function can then be specified by giving its values at the vertices. Let $\mathcal{I} : \mathcal{H}^1(\Omega) \rightarrow \mathcal{M}$ denote continuous piecewise linear interpolation operator that interpolates at the vertices of $\mathcal{T}$. Then

$$\begin{aligned} \mathcal{M}_p &= \{\phi \in \mathcal{M} \mid \phi \text{ is continuous on } \partial\Omega_0\}, \\ \mathcal{M}_g &= \{\phi \in \mathcal{M}_p \mid \phi = \mathcal{I}(g_2) \text{ on } \partial\Omega_2\}, \\ \mathcal{M}_e &= \{\phi \in \mathcal{M}_p \mid \phi = 0 \text{ on } \partial\Omega_2\}. \end{aligned}$$

The discrete equations solved by *PLTMG* are formulated as follows: find $u_h \in \mathcal{M}_d$ such that

$$(1.6) \quad a(u_h, v) = 0 \quad \text{for all } v \in \mathcal{M}_e.$$

1.2. Obstacle Problem.. The second class of problems addressed by *PLTMG* are the subset of variational inequalities known as obstacle problems. Let

$$\mathcal{K} = \{\phi \in \mathcal{H}_g^1 \mid \underline{u} \leq \phi \leq \bar{u}\}.$$

The obstacle problem is formulated as

$$(1.7) \quad \min_{u \in \mathcal{K}} \rho(u)$$

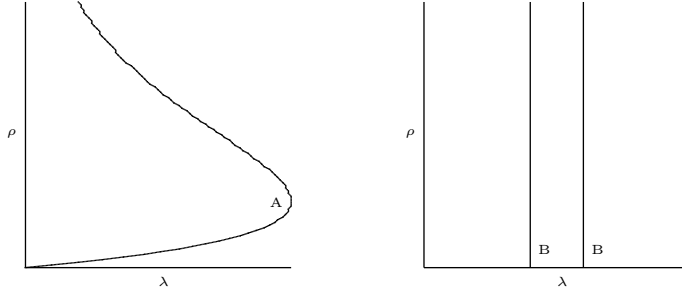
where ρ is a functional of the form (1.5). The parameter λ is not used in this problem. Implicit in our formulation of this problem is an assumption that the Frechet derivative of ρ corresponds to an elliptic boundary problem of the form (1.3). We also assume that the bound constraints are consistent with the boundary conditions.

The discrete form of this problem is as follows. Let

$$\mathcal{K}_h = \{\phi \in \mathcal{M}_g \mid \mathcal{I}(\underline{u}) \leq \phi \leq \mathcal{I}(\bar{u})\}.$$

We then seek $u_h \in \mathcal{K}_h$ that satisfies

$$(1.8) \quad \min_{u_h \in \mathcal{K}_h} \rho(u_h)$$

FIG. 1.1. Continuation curves $\rho = \rho(\lambda)$.

1.3. Continuation Problem.. Continuation problems addressed by *PLTMG* are all of the form (1.3), where the parameter $\lambda \in \mathcal{R}$. Continuation problems also require a functional ρ as in (1.5). Solutions of (1.3)–(1.5) in general define a family of curves on the (λ, ρ) plane. Typical curves are shown in Figure 1.1.

The singular point labeled “A” in the figure on the left is a limit (turning) point, and those labeled “B” in the figure on the right are bifurcation points (this figure corresponds to the special case of a linear eigenvalue problem). The purpose of the continuation process is to compute solutions (u, λ) corresponding to points on these curves.

PLTMG provides a suite of options for solving continuation problems. Among them are options for following a solution curve to a target value in λ or ρ , locating limit and bifurcation points, and switching branches at bifurcation points. Because some problems might have more than one parameter of interest, *PLTMG* also has options for switching parameters and functionals (changing the definitions of λ and ρ) during the calculation, as a means of exploring higher dimensional spaces.

1.4. Parameter Identification Problem.. In this problem, a partial differential equation of the form (1.3) appears as a constraint in an optimization problem. Here we seek $\lambda \in \mathcal{R}$ and $u \in \mathcal{H}_g$ that satisfy

$$(1.9) \quad \min \rho(u, \lambda)$$

subject to the constraint (1.3) and the simple bounds

$$(1.10) \quad \underline{\lambda} \leq \lambda \leq \bar{\lambda}.$$

We define the Lagrangian

$$(1.11) \quad L(u, v, \lambda) = \rho(u, \lambda) + a(u, v),$$

where $v \in \mathcal{H}_e$ is a Lagrange multiplier. We can solve the optimization problem by seeking stationary points of $L(u, v, \lambda)$ constrained by the simple bounds (1.10).

In the discretized problem, we seek $u_h \in \mathcal{M}_g$, a discrete Lagrange multiplier $v_h \in \mathcal{M}_e$, and $\lambda_h \in \mathcal{R}$ that correspond to a stationary point of $L(u_h, v_h, \lambda_h)$, constrained by the simple bounds

$$(1.12) \quad \underline{\lambda} \leq \lambda_h \leq \bar{\lambda}.$$

As in the case of continuation problems, a problem of the form (1.9)–(1.10) may involve more than one parameter of interest. At present, *PLTMG* does not allow

λ to be a vector of parameters, but it does allow parameter switching (redefining the meaning on λ) during the course of the calculation. Thus one can sequentially minimize (1.9) with respect to one of the parameters, holding the others fixed.

1.5. Optimal Control Problem.. This problem is very similar to the parameter identification problem, except now $\lambda \in \mathcal{H}^1(\Omega)$ (or perhaps some weaker space where pointwise values of (1.14) below are defined). Thus we seek $u \in \mathcal{H}_g$ and $\lambda \in \mathcal{H}^1(\Omega)$ that satisfy

$$(1.13) \quad \min \rho(u, \lambda)$$

subject to the constraint (1.3) and the simple bounds

$$(1.14) \quad \underline{\lambda}(x, y) \leq \lambda \leq \bar{\lambda}(x, y)$$

for $(x, y) \in \Omega$. As before, we define the Lagrangian

$$(1.15) \quad L(u, v, \lambda) = \rho(u, \lambda) + a(u, v),$$

where $v \in \mathcal{H}_e$ is a Lagrange multiplier. We seek stationary points of $L(u, v, \lambda)$ constrained by the simple bounds (1.14).

In the discretized problem, we seek $u_h \in \mathcal{M}_g$, a discrete Lagrange multiplier $v_h \in \mathcal{M}_e$, and $\lambda_h \in \mathcal{M}$ that correspond to a stationary point of $L(u_h, v_h, \lambda_h)$. constrained by the simple bounds

$$(1.16) \quad \mathcal{I}(\underline{\lambda}) \leq \lambda_h \leq \mathcal{I}(\bar{\lambda}).$$

2. Main Subroutines. The software package consists of six primary subroutines. These main routines and their functions are summarized in Table 2. The package uses two basic data structures to specify the domain Ω : the triangulation and the skeleton. Loosely speaking, a triangulation specifies the domain Ω as the union of triangles. A skeleton specifies the domain as the union of one or more subdomains and requires only a description of the boundary of each subdomain. The user can specify the domain as either a triangulation or a skeleton. Specifying a triangulation generally requires less data only for simple domains that can be triangulated with very few triangles. If the domain has a complicated geometry or has internal interfaces that the user would like the triangulation to respect, then it is usually easier to specify the domain as a skeleton.

Subroutine	Main Function
<i>TRIGEN</i>	Mesh generation and modification
<i>PLTMG</i>	Solve partial differential equation
<i>TRIPLT</i>	Display solution or related function
<i>INPLT</i>	Display input data
<i>GPHPLT</i>	Display performance statistics
<i>MTXPLT</i>	Display sparse matrix

TABLE 2.1

The main subroutines in the package.

Subroutine *TRIGEN* is mainly concerned with transforming the data structures defining the domain. *TRIGEN* also provides a posteriori error estimates for the

solution in the $\mathcal{H}^1(\Omega)$ and $\mathcal{L}^2(\Omega)$ norms. *TRIGEN* provides options for creating triangulation and skeleton data structures, and adaptively modifying the triangulation data structure. *TRIGEN* also provides options for various tasks related to parallel processing, namely partitioning the mesh, broadcasting a given mesh to all processors, reconciling a fine mesh distributed among several processors, and (possibly) collecting a fine mesh from many processors onto just one.

Subroutine *PLTMG* uses finite element discretizations based on C^0 piecewise linear triangular finite elements and includes algorithms to address each of the problem classes described above. In the case of parallel processing, *PLTMG* includes a domain decomposition solver for each problem class.

Subroutine *TRIPLT* provides graphical displays of the solution and other grid functions. Three-dimensional color surface/contour plots with shading and an arbitrary viewing perspective are available. Subroutine *INPLT* provides a graphical display of the mesh data (triangulation or skeleton) defining Ω . Subroutine *GPHPLT* provides a variety of graphical displays of convergence histories, statistical data, and other interesting output from *PLTMG*. Subroutine *MTXPLT* displays the stiffness matrix A or the (approximate) *LDU* factorization of A in a graphical format.

An elementary interactive test driver, *ATEST*, provides options for calling each of the main routines, as well as other useful functions such as writing and reading data files, resetting parameters, and executing problem specific subroutines provided by the user. Several short machine dependent routines are required for timing and graphics. Some examples problems data sets are included with the source code.

PLTMG was originally conceived as a prototype program to study the theoretical and practical aspects of the multigrid iterative method, adaptive grid refinement and error estimation procedures, and their interaction. As such, *PLTMG* was designed to (formally) handle a wide class of elliptic operators and reasonably general domains. The boundary of the problem class has expanded as problems were encountered that required its enlargement to be solved. The problem class addressed by this version of *PLTMG* should not be interpreted as the limit of the class of problems that could be successfully solved by the techniques embodied by this package. Conversely, one should not assume that every problem (formally) within this class can be solved using the existing code.

As with other versions of the package, time efficiency is a secondary consideration to robustness, versatility, and ease of maintenance. While *PLTMG* is probably not the fastest code that could be used for any particular problem, we believe that it will deliver reasonable execution times in most environments.

3. Installation.. The package is provided in both single and double precision versions. The code development was done in single precision, and the program *S2D* of Jim Meyering (available from Netlib) was used to create the double precision version. The source code is contained in several files as indicated in Table 3. The majority of the source code is machine independent. The X-Windows interface is based on the Motif widget set and can be used only on systems which support X-Windows. Certain X-Windows libraries must be loaded along with the *PLTMG* software. The OpenGL graphics program *SG* of Michael Holst has been integrated as one of several available graphics devices. *SG* is available elsewhere, and its *MALOC* library must be loaded along with the *PLTMG* software. Finally, the parallel processing options in *PLTMG* are based on *MPI*, and the *MPI* library must also be loaded in order to resolve all external names.

In *MPI* is not available or not desired, one can substitute the supplied *stub* inter-

File	Contents
pltmg.f	most source code
mgmpi.f (mgmpi_stubs.f)	<i>MPI</i> interface
mgvio.f (mgvio_stubs.f)	<i>SG</i> interface
xgui.c (xgui_stubs.c)	X-Windows interface
mgxdr.c	XDR interface
atest.f	test driver program
burger.f, battery.f, circle.f, control.f domains.f, ident.f, jcn.f, message.f mnsurf.f, naca.f, ob.f, square.f	test problem data sets

TABLE 3.1
Files in the basic distribution.

face routines. The stub routines are a set of *MPI* interface routines with all calls to *MPI* library functions and subroutines deleted. By using the stub routines in place of the regular interface, one can create an executable with no unresolved external references without loading the *MPI* library. In this case, however, all the parallel options of *PLTMG* are disabled.

In a similar fashion, if *SG* is not available or not desired, one can use the stub routines in place of standard interface routines. If the stub routines are used, the *MALOC* library is not needed, but the *SG* OpenGL and BH file graphics devices are disabled. Finally, if the X-Windows libraries are not available, one can replace the X-Windows interface with stub routines. In this case, the graphical user interface and the corresponding X-Windows graphics devices are all disabled, but the X-Windows libraries are not needed.