rbmatmod

Paolo Luchini
CPLcode.net

May 5, 2022

Abstract

Distributed-memory extension of the rbmat.cpl library that solves a banded linear system by distributing the work among multiple nodes.

Description

The classical Gauss substitution algorithm for the solution of a linear system can be subdivided into a sequence of three phases: 1) LU decomposition of the coefficient matrix, 2) division of the right-hand-side vector by the triangular L part; 3) division of the intermediate result by the triangular U part to get the final result. Each of these can be performed in place and programmed so as to account for the banded structure of the matrix. Once 1) is performed, 2) and 3) can be repeated multiple times onto different right-hand-sides. Serial routines for each phase are provided together with CPL in the `rbmat.cpl` library, which at the same time defines an algebraic syntax for the required operations.

Each of the above three phases scans sequentially through the array of variables and only uses neighbouring-component information up to the length of the band. In a distributed-memory environment, the array can be split into multiple sections, one for each computing node, and only a number of components as large as the band needs to be transmitted from one node to the next (or previous) one, thus minimizing communication.

The Gauss algorithm as described above is still serial and not parallel insofar as each node has to wait for information from the previous one before continuing. However, if several linear systems are to be solved simultaneously, the computation of each phase for all the systems can be calculated (sequentially, or in groups if local parallelism is available) at the same time that another node works on a different group of linear systems (see example below). Distributed parallelism is thus achieved.

Usage

Routines `LUdecompStep`, `LeftLUDivStep1`, `LeftLUDivStep2`, `RightLUDivStep1` and `RightLUDivStep2` have analogous functions to the corresponding routines in `rbmat.cpl`. The last two routines, in particular, perform division to the right rather than division to the left and are only needed if division to the right is of interest (for instance in adjoint computations).

A usage example and test is provided in the file `rbmatmodtest.cpl`. You can run it on a given set of nodes by for each including three parameters on the command line: sequential number of the present node (starting from 1), total number of nodes in the simulation, and the internet hostname of the next node (the one with the next higher number) in the chain. For test purposes you can just as well run several processes on the same node; for example, for two processes:

```
rbmatmodtest 1 2 localhost & rbmatmodtest 2 2
```