Parallel DYNARE Toolbox
FP7 Funded
Project MONFISPOL Grant no.: 225149

Marco Ratto
European Commission, Joint Research Centre, Ispra, ITALY
with Ivano Azzini, Houtan Bastani, Sebastien Villemot

November 8, 2010

JRC
EUROPEAN COMMISSION

Outline

JRC EUROPEAN COMMISSION

Introduction

Parallelize portions of code that require a minimal (i.e. start-end communication) or no communications between different processes, denoted in the literature as "embarrassingly parallel" (Goffe and Creel, 2008; Barney, 2009), with the following criteria:

- provide the necessary input data to any sequence;
- distribute the workload, automatically balancing between the computational resources;
- collect the output data;
- ensure the coherence of the results with the original serial execution.

```
...
n=2;
m=10^6;
Matrix= zeros(n,m);
for i=1:n,
    Matrix(i,:)=rand(1,m);
end,
Mse= Matrix;
...
```

Example 1

```
            ...
            n=2;
            m=10^6;
            <provide to CPU1 and CPU2 input data m>
-
<Execute on CPU1>              <Execute on CPU2>
Matrix1 = zeros(1,m);           Matrix2 = zeros(1,m);
Matrix1(1,:)=rand(1,m);         Matrix2(1,:)=rand(1,m);
save Matrix1                    save Matrix2
-
            retrieve Matrix1 and Matrix2
            Mpe(1,:) = Matrix1;
            Mpe(2,:) = Matrix2;
```

Example 2

The `for` cycle has disappeared and it has been split into two separated sequences that can be executed in parallel on two CPUs. We have the same result (`Mpa=Mse`) but the computational time can be reduced up to 50%.

The DYNARE environment I

Routines suitable for parallelization:

1. the RW Metropolis with multiple chains (`random_walk_metropolis_hastings.m`) (`independent_metropolis_hastings.m`);
2. the diagnostic tests (`McMCDiagnostics.m`);
3. posterior IRF's (`posteriorIRF.m`).
4. posterior statistics for filtered and smoothed variables, forecasts, smoothed shocks, etc.. (`prior_posterior_statistics.m`).
5. plotting utility (`pm3.m`).

The DYNARE environment II

MATLAB does not support multi-threads, without the use (and purchase) of MATLAB Distributed Computing Toolbox.

When the execution should start in parallel:

1. *pass the control to the operating system (Windows/Linux);*
2. *launched concurrent threads (i.e. MATLAB instances) on different processors/cores/machines;*
3. *when parallel is concluded the control is given back to the original MATLAB session .*

Requirements I

For a Windows grid:

1. a standard Windows network (SMB) must be in place;
2. PsTools (Russinovich, 2009) must be installed in the path of the master Windows machine;
3. the Windows user on the master machine has to be user of any other slave machine in the cluster, and that user will be used for the remote computations.

Requirements II

For a UNIX grid

1. SSH must be installed on the master and on the slave machines;
2. the UNIX user on the master machine has to be user of any other slave machine in the cluster, and that user will be used for the remote computations;
3. SSH keys must be installed so that the SSH connection from the master to the slaves can be done without passwords, or using an SSH agent.

The interface I

Put the configuration of the cluster in a config file different from the MOD file, and to trigger the parallel computation with option(s) on the `dynare` command line. The configuration file is designed as follows:

- be in a standard location
  - `$HOME/.dynare` under Unix;
  - `c:\Documents and Setting\<username>\Application Data\dynare.ini` on Windows;
- have provisions for other Dynare configuration parameters unrelated to parallel computation
- allow to specify several clusters, each one associated with a nickname;

The interface II

Node Options	type	default	Win		Unix	
			Local	Remote	Local	Remote
Name	string	(stop)	*	*	*	*
CPUnbr	integer or array	(stop)	*	*	*	*
ComputerName	string	(stop)		*		*
UserName	string	empty		*		*
Password	string	empty		*		
RemoteDrive	string	empty		*		
RemoteDirectory	string	empty		*		*
DynarePath	string	empty				
MatlabOctavePath	string	empty				
SingleCompThread	boolean	true				

Cluster Options	type	default	Meaning	Required
Name	string	empty	name of the node	*
Members	string	empty	list of members in this cluster	*

Configuration file syntax (Unix)

```
[cluster]
Name = c1
Members = n1 n2 n3
[cluster]
Name = c2
Members = n2 n3

[node]
Name = n1
ComputerName = localhost
CPUnbr = 1
```

```
[node]
Name = n2
ComputerName = karaba.cepremap.org
CPUnbr = 5
UserName = houtanb
RemoteDirectory = /home/houtanb/Remote
DynarePath = /home/houtanb/dynare/matlab
MatlabOctavePath = matlab
[node]
Name = n3
ComputerName = hal.cepremap.ens.fr
CPUnbr = 3
UserName = houtanb
RemoteDirectory = /home/houtanb/Remote
DynarePath = /home/houtanb/dynare/matlab
MatlabOctavePath = matlab
```

Configuration file syntax (Win)

```
[cluster]
Name = local
Members = n1

[cluster]
Name = vonNeumann
Members = n2

[cluster]
Name = c2
Members = n1 n2
```

```
[node]
Name = n1
ComputerName = localhost
CPUnbr = 4

[node]
Name = n2
ComputerName = vonNeumann
CPUnbr = [4:6]
UserName = COMPUTOWN\John
Password = *****
RemoteDrive = C
RemoteDirectory = dynare_calcs\Remote
DynarePath = c:\dynare\matlab
MatlabOctavePath = matlab
```

DYNARE command line options

- `conffile=<path>`: specify the location of the configuration file if it is not standard
- `parallel`: trigger the parallel computation using the first cluster specified in config file
- `parallel=<clustername>`: trigger the parallel computation, using the given cluster
- `parallel_slave_open_mode`: use the leaveSlaveOpen mode in the cluster
- `parallel_test`: just test the cluster, dont actually run the MOD file

Core functions I

`masterParallel` is the entry point to the parallelization system:

- It is called from the master computer, at the point where the parallelization system should be activated;
- all file exchange through the filesystem is concentrated in this `masterParallel` routine;
- there are two modes of parallel execution, triggered by option `parallel_slave_open_mode`;

`slaveParallel.m/fParallel.m`: are the top-level functions to be run on every slave;

Core functions II

`fMessageStatus.m`: provides the core for simple message passing during slave execution and typically replaces calls to `waitbar.m`;

`closeSlave.m` is the utility that sends a signal to remote slaves to close themselves;

Utilities

- `AnalyseComputationalEnviroment.m`;
- `InitializeComputationalEnviroment.m`;
- `distributeJobs.m`;
- generalized routines:

  `dynareParallelDelete.m` : generalized `delete`;
  `dynareParallelDir.m` : generalized `dir`;
  `dynareParallelGetFiles.m` : generalized copy FROM slaves TO master machine;
  `dynareParallelMkDir.m` : generalized `mkdir` on remote machines;
  `dynareParallelRmDir.m` : generalized `rmdir` on remote machined;
  `dynareParallelSendFiles.m` : generalized copy TO slaves FROM master machine;

JRC EUROPEAN COMMISSION

Parallel DYNARE: testing

We present here all tests performed with Windows Xp/Matlab. The model used for testing is a modification of Lubik (2003): a small scale open economy DSGE model with 6 observed variables, 6 endogenous variables and 19 parameters to be estimated.

Test 1

Bi-processor machine (Fujitsu Siemens, Celsius R630) powered with an Intel(R) Xeon(TM) CPU 2.80GHz Hyper Treading Technology.

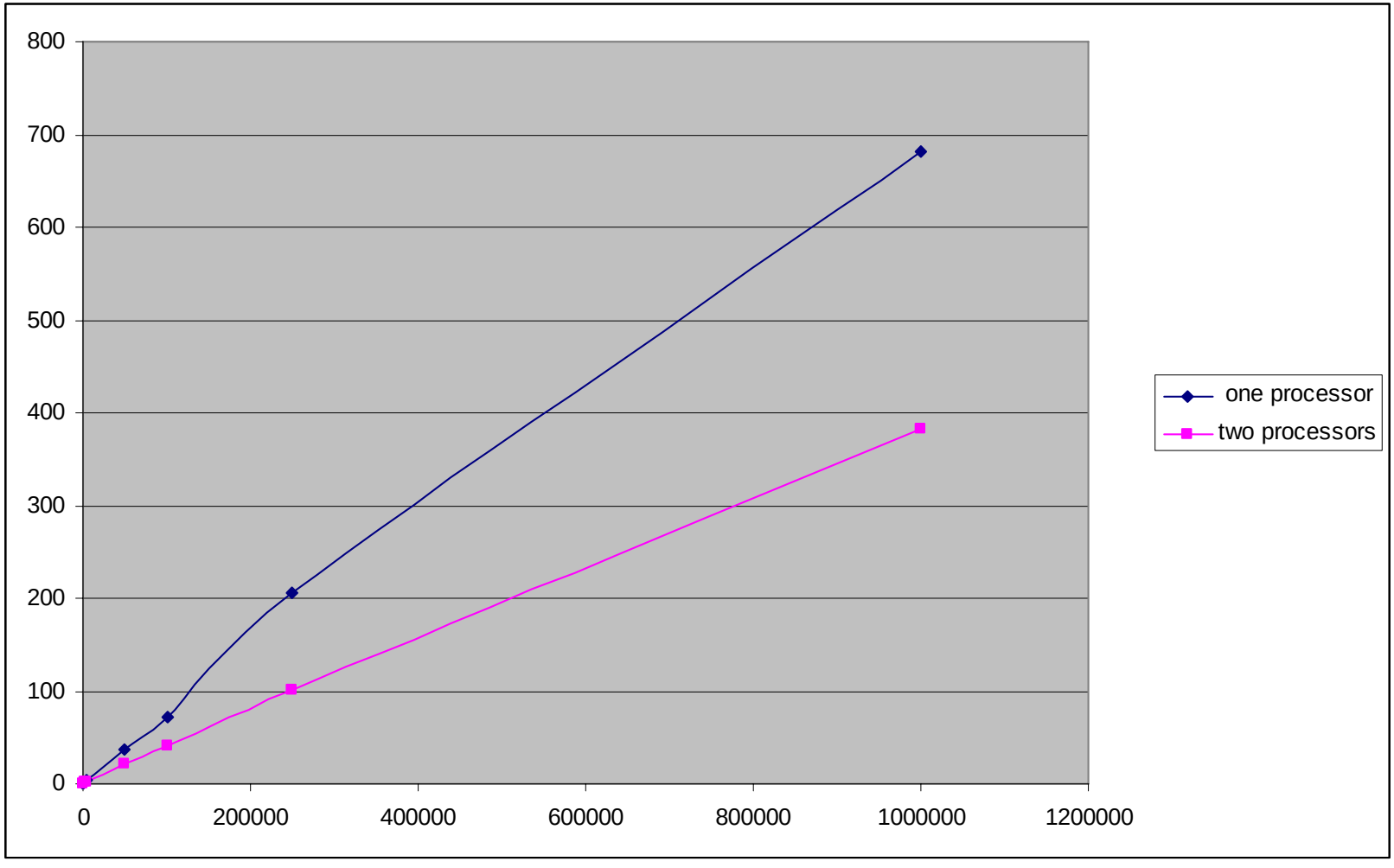

Figure: Computational time (in minutes) versus chain length for the serial and parallel implementation (Metropolis with two chains).

JRC EUROPEAN COMMISSION

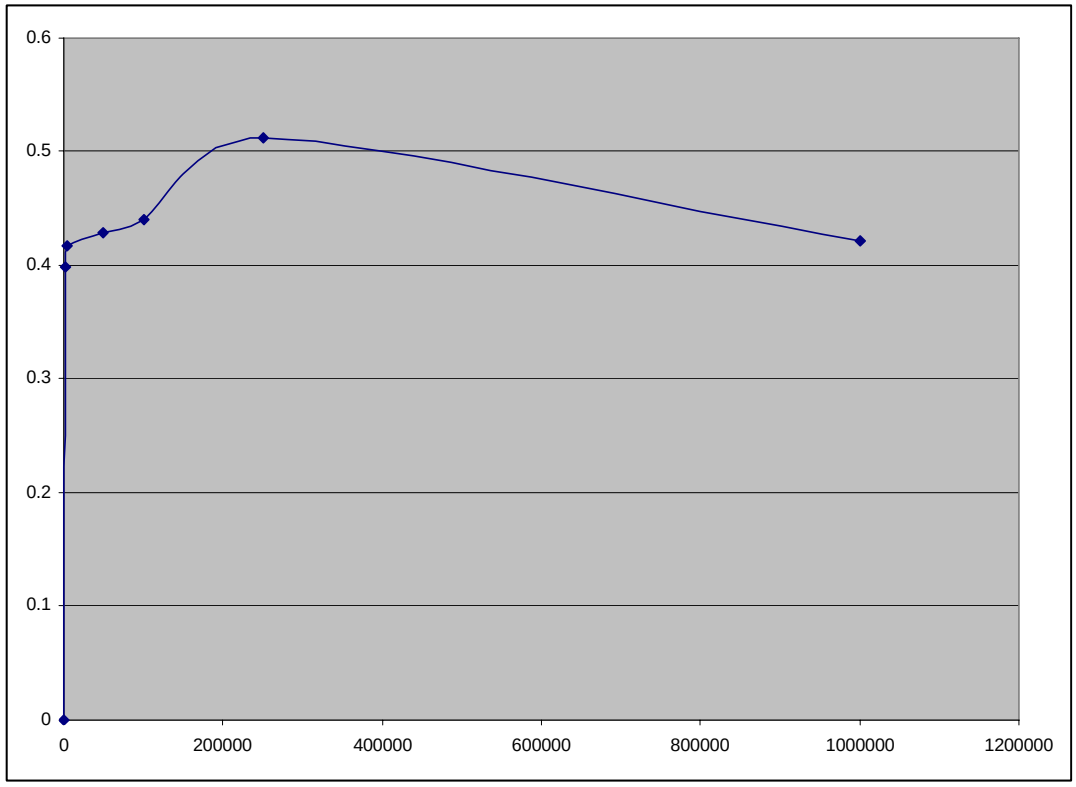

Figure: Reduction of computational time (i.e. the 'time gain') using the parallel coding versus chain length. The time gain is computed as $(T_s - T_p)/T_p$, where T_s and T_p denote the computing time of the serial and parallel implementations respectively.

Test 2 I

Test different hardware platforms, with chain lengths of 1,000,000 runs on the following hardware platforms:

- Single processor machine: Intel(R) Pentium4(R) CPU 3.40GHz with Hyper Treading Technology (Fujitsu-Siemens Scenic Esprimo);
- Bi-processor machine: two CPU's Intel(R) Xeon(TM) 2.80GHz Hyper Treading Technology (Fujitsu-Siemens, Celsius R630);
- Dual core machine: Intel Centrino T2500 2.00GHz Dual Core (Fujitsu-Siemens, LifeBook S Series).

Test 2 II

Machine	Single-processor	Bi-processor	Dual core
Parallel	8:01:21	7:02:19	5:39:38
Serial	10:12:22	13:38:30	11:02:14
Speed-Up rate	1.2722	1.9381	1.9498
Ideal Speed-UP rate	$\sim$1.5	2	2

Table: Trail results with normal PC operation. Computing time expressed in h:m:s. Speed-up rate is computed as T_s/T_p, where T_s and T_p are the computing times for the serial and parallel implementations.

Test 2 III

Environment	Computing time	Speed-up rate w.r.t. Table 1
Parallel Waitbar Not Visible	5:06:00	1.06
Parallel waitbar Not Visible, Real-time Process priority, Unplugged network cable.	4:40:49	1.22

Table: Trail results with different software configurations (optimized operating environment for computational requirements).

Test 3

We used a very simple computer class, quite diffuse nowadays: Netbook personal Computer. In particular we used the Dell Mini 10 with Processor Intel Atom Z520 (1,33 GHz, 533 MHz), 1 GB di RAM (with Hyper-trading).

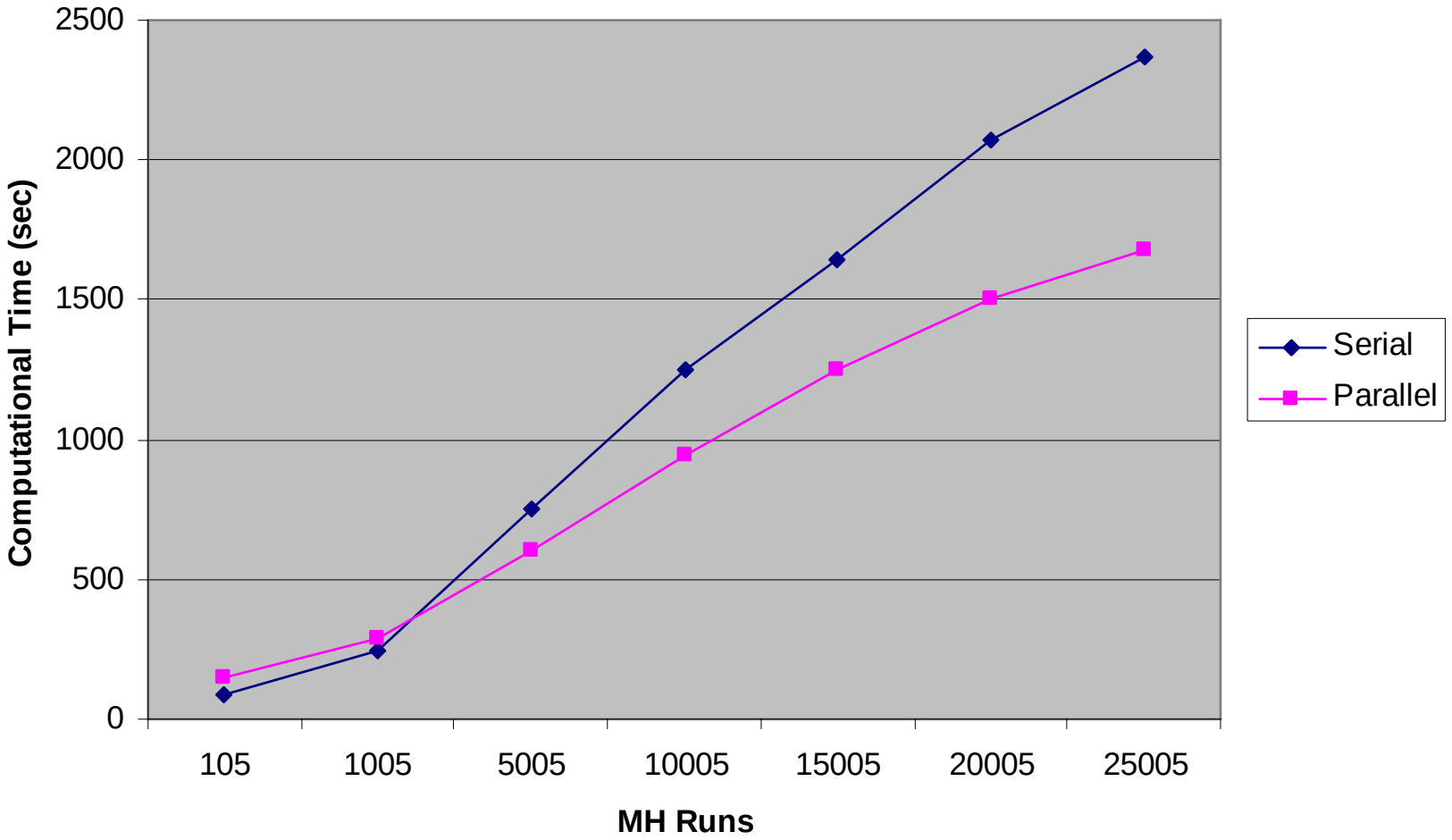

Figure: Computational Time (s) versus Metropolis length, running all the parallelized functions in DYNARE and the basic parallel implementation (the 'Open/Close' strategy). (Lubik, 2003).

JRC EUROPEAN COMMISSION

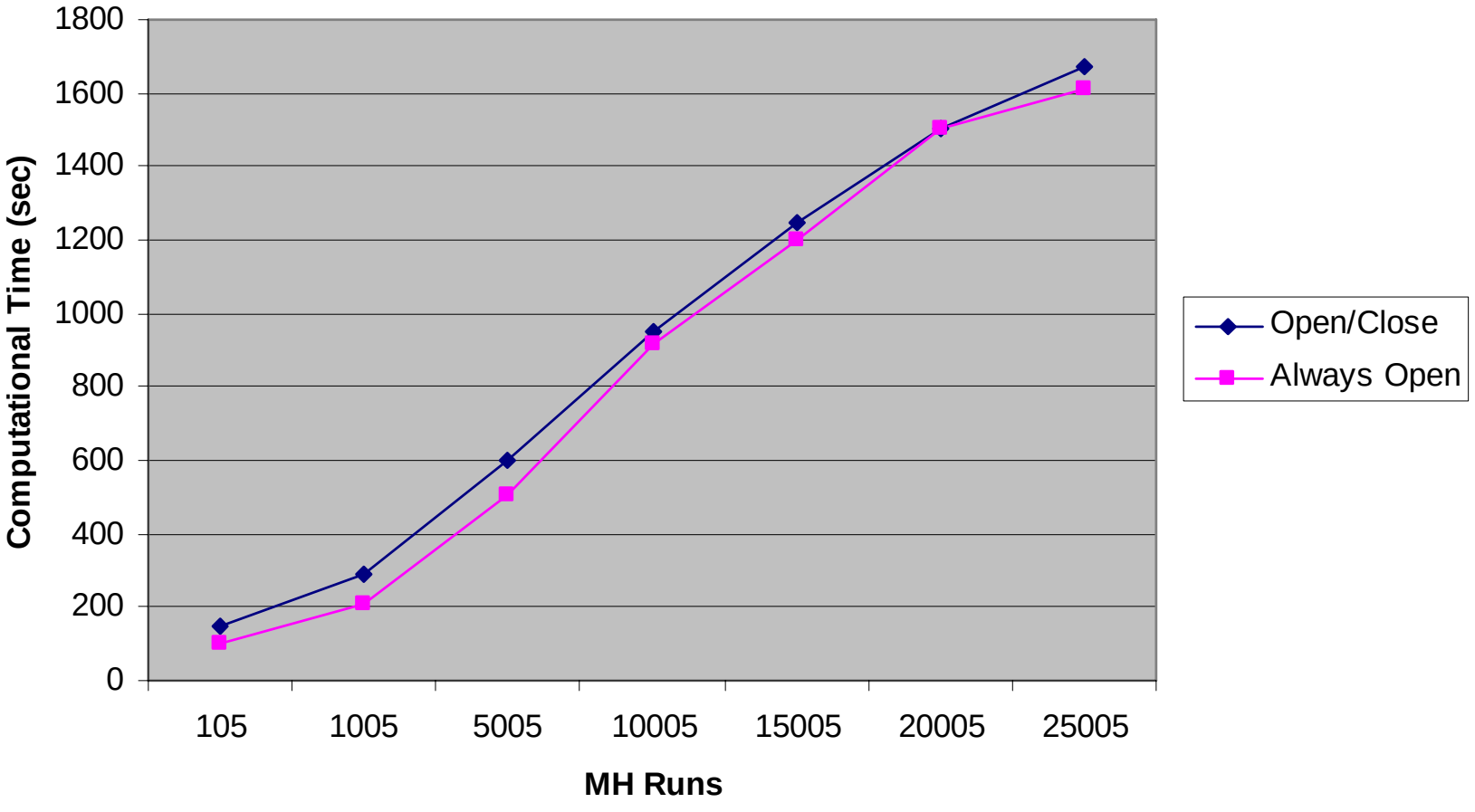

Figure: Comparison of the 'Open/Close' strategy and the 'Always-open' strategy. Computational Time (s) versus Metropolis length, running all the parallelized functions in DYNARE (Lubik, 2003).

JRC EUROPEAN COMMISSION

Tets 4 I

Here we increase the dimension of the test model, using the QUEST III model (Ratto et al., 2009), using a more powerful Notebook Samsung Q 45 with an Dual core Processor Intel Centrino.

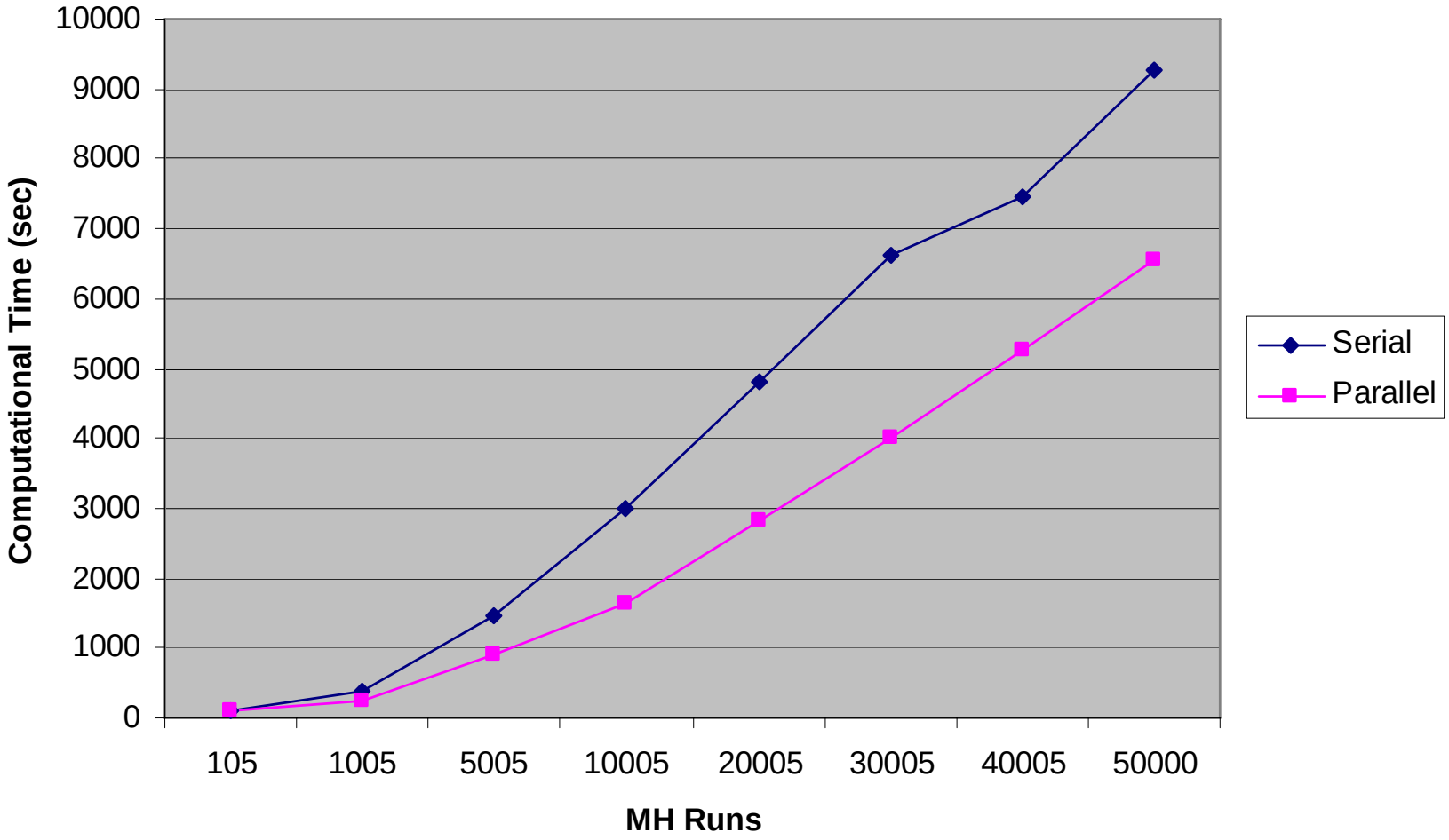

Figure: Computational Time (s) versus Metropolis length, running all the parallelized functions in DYNARE and the basic parallel implementation (the 'Open/Close' strategy). (Ratto et al., 2009).

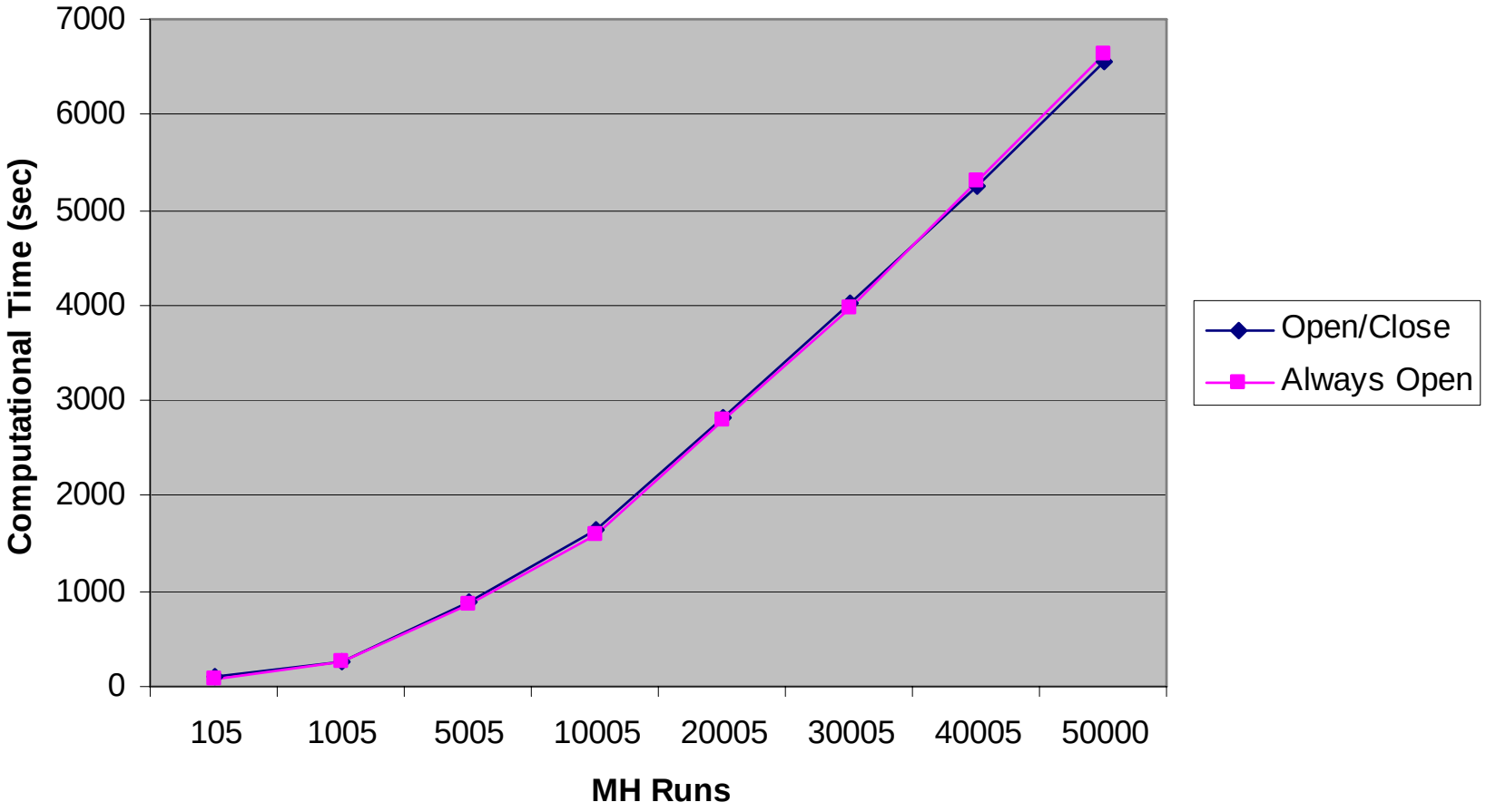

Figure: Comparison of the 'Open/Close' strategy and the 'Always-open' strategy. Computational Time (s) versus Metropolis length, running all the parallelized functions in DYNARE (Ratto et al., 2009).

Conclusions

The methodology identified for parallelizing MATLAB codes within DYNARE proved to be effective in reducing the computational time of the most extensive loops. This methodology is suitable for 'embarrassingly parallel' codes, requiring only a minimal communication flow between slave and master threads. The parallel DYNARE is built around a few 'core' routines, that act as a sort of 'parallel paradigm'. Based on those routines, parallelization of expensive loops is made quite simple for DYNARE developers. A basic message passing system is also provided, that allows the master thread to monitor the progress of slave threads. The test model `ls2003.mod` is available in the folder `\tests\parallel` of the DYNARE distribution, that allows running parallel examples.

Bibliography I

B Barney. *Introducing To Parallel Computing (Tutorial)*. Lawrence Livermore National Laboratory, 2009.

W.L. Goffe and M. Creel. Multi-core CPUs, clusters, and grid computing: A tutorial. *Computational economics*, 32(4): 353–382, 2008.

T. Lubik. Investment spending, equilibrium indeterminacy, and the interactions of monetary and fiscal policy. Technical Report Economics Working Paper Archive 490, The Johns Hopkins University, 2003.

Bibliography II

Marco Ratto, Werner Roeger, and Jan in 't Veld. QUEST III: An estimated open-economy DSGE model of the euro area with fiscal and monetary policy. *Economic Modelling*, 26(1):222 – 233, 2009. doi: DOI:10.1016/j.econmod.2008.06.014. URL http://www.sciencedirect.com/science/article/B6VB1-4TC8J5F-1/2/7f22da17478841ac5d7a77d06f13d13e.

M. Russinovich. *PsTools v2.44*, 2009. available at Microsoft TechNet, http://technet.microsoft.com/en-us/sysinternals/bb896649.aspx.