1. Summary
- We accelerate the execution of Octave programs with the Cell Broadband Engine.
- The Cell Broadband Engine is a microprocessor architecture found in Sony’s PS3 game console and in the fastest computer in the world, the IBM Roadrunner.
- Octave is an open source implementation of MATLAB.

2. Motivation
- MATLAB and Octave are the languages of choice for scientists and engineers due to their simplicity of use.
- Problem: These languages can run slowly for large input sizes because only a single core of a processor is utilised during execution.
- The physical performance limits of single-core processors have been reached. As a result modern parallel architectures have introduced many cores, e.g.: Multi-core CPUs General Purpose GPUs The Cell Broadband Engine.
- Problem: These modern parallel architectures are difficult to program due to concurrency issues and hardware idiosyncrasies.
- Question: How can we parallelise a matrix language program for modern parallel architectures without burdening the user.

3. PS³
a. We introduce a framework called PS³ for automatically parallelising Octave programs for the Cell Broadband Engine.
b. PS³ is easy to use and exploits 4 types of parallelism in an Octave program to achieve high performance:
   1. Data parallelism by partitioning large matrices into sub-matrices which can be operated on in parallel.
   2. Instruction level parallelism by executing independent matrix operations in parallel.
   3. Pipeline parallelism by overlapping communication between cores with computation of matrix operations.
   4. Task parallelism by overlapping execution of the Octave interpreter with execution of matrix operations.

4. Results
- Extensive evaluation of PS³ performed with 9 Octave benchmark programs.
- Compared benchmark runtime on a default installation of Octave on an Intel Core2 vs. runtime on our system on the Cell processor.
- Despite the Intel Core2 being a more recent and more expensive architecture than Cell, we achieved speedups of up to 12 times with our system.

5. Simplicity of Use
Our framework employs a data-type extension to Octave. To utilise our framework, a user only needs to wrap all matrix declarations in the \texttt{ps3\_matrix()} function.

6. Lazy Evaluation
Matrix operations in an Octave program are lazily evaluated, deferring their computation until the result is required. A trace of unexecuted matrix operations are represented by a data dependency graph which shows dependencies between operations. By examining these data dependencies, instruction level parallelism can be elicited (i.e. matrix operations that can be computed concurrently).

7. Operation Partitioning
Operations on large matrices are partitioned into operations on small matrices (lowering). This enables data parallelism to be exploited within a matrix operation. We introduce a new partitioning scheme which is efficient and maximises the amount of parallelism in the trace.

8. Operation Scheduling
Partitioned matrix operations are scheduled among the cores of the Cell processor prior to their execution. This is done in a way that: (1) satisfies the data dependencies between operations, and (2) minimises the total execution time (makespan) of the trace. The scheduling problem is NP-hard. We develop a heuristic algorithm to find a good schedule at run-time and an Integer Linear Programming formulation to find the optimal solution.

9. Operation Execution
Scheduled matrix operations are sent to the Computation Engine running on the Cell processor. Each of the cores of the Cell processors executes the operations assigned to it by the schedule.

The University of Sydney