MatCL - OpenCL MATLAB Interface

OpenCL

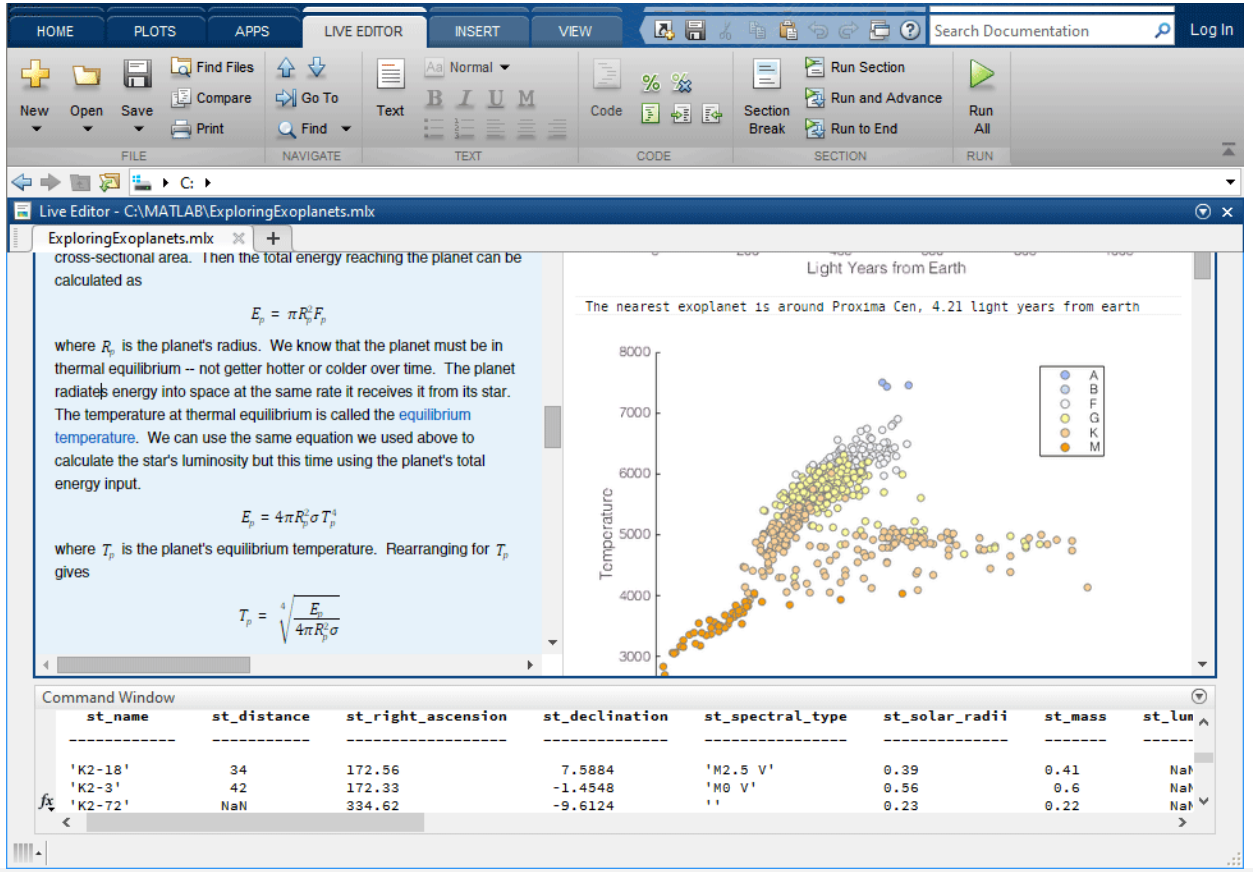
MatCL - OpenCL MATLAB Interface

- OpenCL toolkit for Mathworks MATLAB/SIMULINK
- Compile & Run OpenCL Kernels
- Handles OpenCL memory management

MathWorks MATLAB

- Commercial Numerical Computing Environment
- Engineering, Science, Economics
- Free GNU Octave clone

- Native C/C++ interface “MEX”
- Support for CUDA (2010)
- No OpenCL support



MatCL – Easy to use MATLAB OpenCL toolkit

- Wraps OpenCL low-level API
- Single instruction to copy memory & build, run Kernels
- Support all standard OpenCL devices
- Intended for Kernel development & MATLAB acceleration

→Easy way to try out & learn OpenCL

MatCL Features

- MATLAB R2014a and up (GNU Octave support)
- Windows, Linux, macOS
- Native support for MATLAB datatypes (scalar & vector)
- Compiler output & Kernel ‘printf’ redirection
- Support for OpenCL 2.x pipes
- Precompiled binaries available on GitHub
  https://github.com/philipheinisch/MatCL

MatCL Basic Usage

- Get Device info:

```
[dev_name,dev_type,max_mem,wg_size,lw_size,compute_units]=cl_get_devices;
```

  - Available devices with type (CPU,GPU)
  - Available memory
  - Possible WG & LW sizes
  - Number of compute units per device

MatCL Basic Usage

- Compile & Run kernel automatically

```
[trun,tcopy]=cl_run_kernel(dev,'kernel.cl','-cl-std=CL2.0','test_func',global_range,local_range,var1,var2,varn,[0 1 1]);
```

  - Quick & Easy way to test kernels

- Compile & Run Kernel separately

```
[tcomp,kernel_names]=cl_run_kernel(dev,'kernel.cl','-cl-std=CL2.0');
[trun,tcopy]=cl_run_kernel(dev,'test_func',global_range,local_range,var1,var2,varn,[0 1 1]);
```

  - Intended to run kernels repeatedly
  - MATLAB acceleration

Kernel Development & Validation

- Problem:
  - Intended Host application unsuitable for development (HPC Clusters, Embedded Platforms, FPGAs…)
  - Develop Kernels using predefined input data & check results

- Advantages of MATLAB:
  - Provides data I/O (HDF5, XML, MPEG, JPG, PNG, SQL…)
  - Provides graphical output/plotting tools
  - Comprehensive numerical toolbox for data generation & validation

Example: Kernel Development

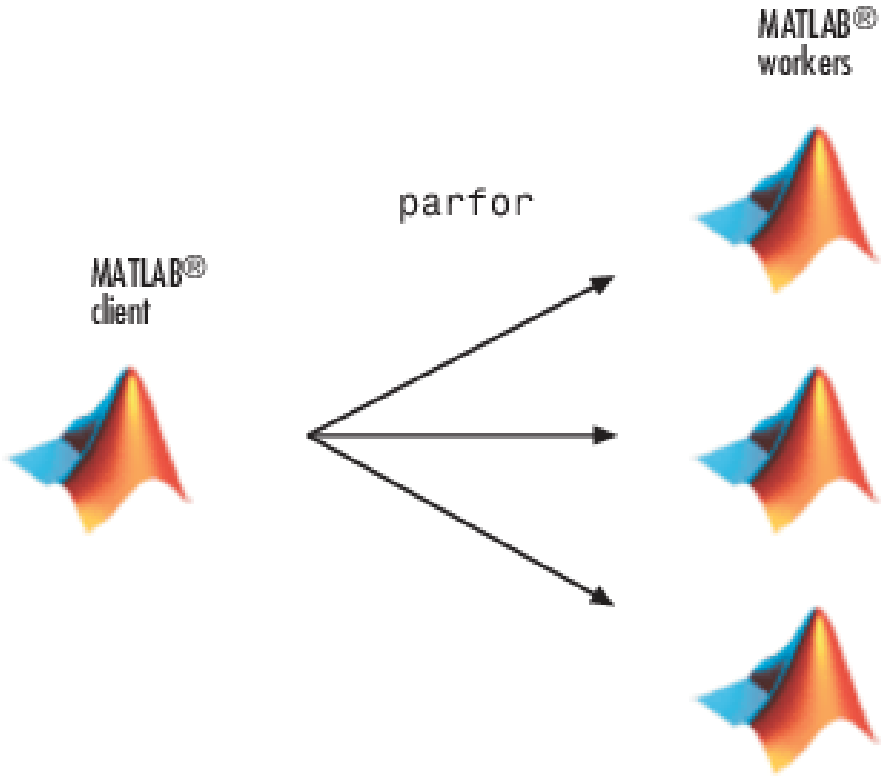
- Numerical Plasma Simulation
  - Complex indexing
  - Multi-Layer boundary handling
  - Requires specific input data

```
kernel solver_O2_HR_split(uint ix,  uint iy, uint iz, global REAL4 *d_field_b_old, global REAL4 *d_field_u, global REAL *d_field_et

  uint idx =  calc_idx(ix,iy,iz);

  convec.x = d_field_b_old[idx].x*(d_field_u[increment_x(idx,1)].x - d_field_u[increment_x(idx,-1)].x)/((2.0f*(REAL)DX))
             +d_field_b_old[idx].y*(d_field_u[increment_y(idx,1)].x - d_field_u[increment_y(idx,-1)].x)/((2.0f*(REAL)DY))
             -0.5*d_field_b_old[idx].x*((d_field_u[increment_x(idx,1)].x - d_field_u[increment_x(idx,-1)].x)/(2.0f*(REAL)DX)
             +(d_field_u[increment_y(idx,1)].y - d_field_u[increment_y(idx,-1)].y)/((2.0f*(REAL)DY)))
             -0.5*(d_field_u[idx].x*(d_field_b_old[increment_x(idx,1)].x - d_field_b_old[increment_x(idx,-1)].x)/(2.0f*(REAL)DX)
             +d_field_u[idx].y*(d_field_b_old[increment_y(idx,1)].x - d_field_b_old[increment_y(idx,-1)].x)/(2.0f*(REAL)DY))
             -0.5*((d_field_u[increment_x(idx,1)].x*d_field_b_old[increment_x(idx,1)].x
             -d_field_u[increment_x(idx,-1)].x*d_field_b_old[increment_x(idx,-1)].x)/(2.0f*(REAL)DX)
             +(d_field_u[increment_y(idx,1)].y*d_field_b_old[increment_y(idx,1)].x
             -d_field_u[increment_y(idx,-1)].y*d_field_b_old[increment_y(idx,-1)].x)/(2.0f*(REAL)DY));

  convec.y = d_field_b_old[idx].x*(d_field_u[increment_x(idx,1)].y - d_field_u[increment_x(idx,-1)].y)/((2.0f*(REAL)DX))
             +d_field_b_old[idx].y*(d_field_u[increment_y(idx,1)].y - d_field_u[increment_y(idx,-1)].y)/((2.0f*(REAL)DY))
             -0.5*d_field_b_old[idx].y*((d_field_u[increment_x(idx,1)].x - d_field_u[increment_x(idx,-1)].x)/(2.0f*(REAL)DX)
             +(d_field_u[increment_y(idx,1)].y - d_field_u[increment_y(idx,-1)].y)/((2.0f*(REAL)DY)))
             -0.5*(d_field_u[idx].x*(d_field_b_old[increment_x(idx,1)].y - d_field_b_old[increment_x(idx,-1)].y)/(2.0f*(REAL)DX)
             +d_field_u[idx].y*(d_field_b_old[increment_y(idx,1)].y - d_field_b_old[increment_y(idx,-1)].y)/(2.0f*(REAL)DY))
             -0.5*((d_field_u[increment_x(idx,1)].x*d_field_b_old[increment_x(idx,1)].y
             -d_field_u[increment_x(idx,-1)].x*d_field_b_old[increment_x(idx,-1)].y)/(2.0f*(REAL)DX)
             +(d_field_u[increment_y(idx,1)].y*d_field_b_old[increment_y(idx,1)].y
             -d_field_u[increment_y(idx,-1)].y*d_field_b_old[increment_y(idx,-1)].y)/(2.0f*(REAL)DY));
```

MATLAB Code Acceleration

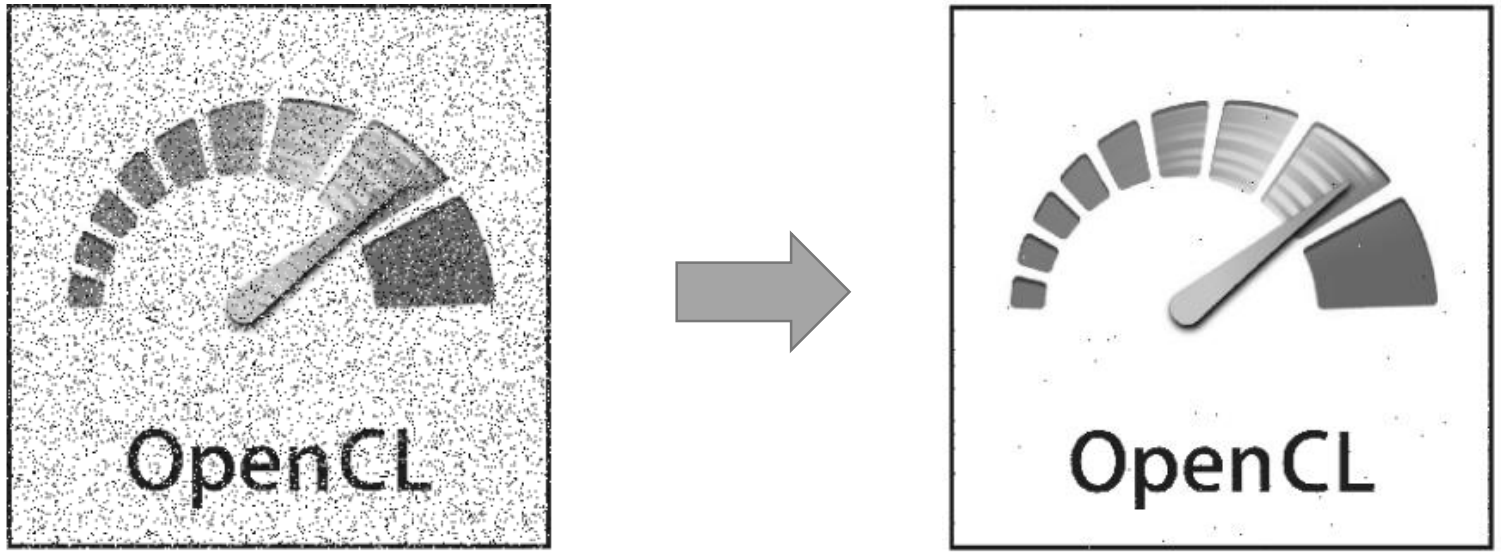
- MPI based code parallelization using ‘parfor’
  - Only data parallelism
  - No parallelization for built-in functions
- Native CUDA support (since 2010)
  - Some built-in functions supported directly
  - Support for custom CUDA code
  - Easy entry for CUDA beginners
- No support for AMD & Intel GPUs…

→ OpenCL support missing



Example: 2D Median Filter

- Typical use: image noise removal, edge detection
- Built-in MATLAB function: ‘medfilt2’
- MATLAB CUDA version available
- OpenCL implementation using MatCL



Example: 2D Median Filter

- Typical use: image noise removal, edge detection
- Problem size: 1200x1200 uint8
- Built-in MATLAB function: ‘medfilt2’
- Systems:
  - Core i7-7700HQ/GTX1050/Intel HD Graphics 630
  - Ryzen 5 1600X/GTX1060

	MATLAB (CPU)	MATLAB (CUDA)	MatCL - CPU	MatCL – GPU1	MatCL GPU2
Core i7-7700 GTX1050/HD 630	27 ms	3 ms	5 ms	1.8 ms	5 ms
Ryzen 5 1600X GTX1060	25 ms	3 ms	2 ms	2.5 ms	-

Example: Cross Correlation

- Typical use: pattern recognition, echo cancelation
- Problem size: 512000 float
- Built-in MATLAB function: ‘xcorr’
- Systems:
  - Core i7-7700HQ/GTX1050/Intel HD Graphics 630
  - Ryzen 5 1600X/GTX1060

	MATLAB (CPU)	MATLAB (CUDA)	MatCL - CPU	MatCL – GPU1	MatCL GPU2
Core i7-7700 GTX1050/HD 630	61 ms	70 ms	8 ms	6 ms	12 ms
Ryzen 5 1600X GTX1060	68 ms	745 ms(?)	9 ms	4 ms	-

Still ahead…

- Detailed documentation & more examples
- Complicated build system (library conflicts)
- No native 16-bit half-precision support (deep learning)
- Grapics support (context sharing)?
- Cooperation with MathWorks (official hardware support package)?
- Spread the word