
Subject: Re: GPULib 1.8 released

Posted by [Michael Galloy](#) on Thu, 05 Feb 2015 20:35:48 GMT

[View Forum Message](#) <> [Reply to Message](#)

On 2/5/15, 3:41 AM, superchromix wrote:

>
> hi Mike,
>
> I notice that there is a new function called GPULEASTSQUARES. Is
> this an implementation of the GPU-based least squares fitting which
> you have mentioned in the past? It would be great if you could
> include a usage example in the documentation.
>
> best Mark
>

The GPULib distribution includes a unit test for GPULEAST_SQUARES shown below (ignore the ASSERT statements, that is just checking to make sure things are as the test expects along the way):

```
function gpuleast_squares_ut::test_float
  compile_opt strictarr

  assert, !gpu.mode ne 0, 'GPULEAST_SQUARES not available in emulation
mode', /skip
  assert, !gpu.mode eq 0L || gpulmgr(/full), 'GPULib not licensed for
LAPACK calculations', /skip
  assert, !gpu.mode eq 0L || gpuMagmaPresent(), 'MAGMA not present', /skip

  a = [[1., 2., 1.], [4., 10., 15.], [3., 7., 1.]]
  dims = size(a, /dimensions)
  m = dims[0]
  n = dims[1]

  b = [2., 3., 7.]

  standard = la_least_squares(a, b)

  da = gpuputarr(a, ERROR=err)
  assert, err eq 0, 'CUDA error: %s', gpu_errormessage(err)

  db = gpuputarr(b, ERROR=err)
  assert, err eq 0, 'CUDA error: %s', gpu_errormessage(err)

  dresult = gpuleast_squares(da, db, ERROR=err)
  assert, err eq 0, 'CUDA error: %s', gpu_errormessage(err)

  result = gpugetarr(dresult, ERROR=err)
```

```

    assert, err eq 0, 'CUDA error: %s', gpu_errormessage(err)

    error = total(abs(standard - result), /preserve_type)
    assert, error lt self.tolerance * n * 10.0, 'incorrect result: error
= %g', error

    gpuFree, [da, db, dresult]

    return, 1
end

function gpuleast_squares_ut::test_double
    compile_opt strictarr

    assert, !gpu.mode ne 0, 'GPULEAST_SQUARES not available in emulation
mode', /skip
    assert, !gpu.mode eq 0L || gpulmgr(/full), 'GPULib not licensed for
LAPACK calculations', /skip
    assert, !gpu.mode eq 0L || gpuMagmaPresent(), 'MAGMA not present', /skip
    assert, !gpu.mode eq 0L || gpuDoubleCapable(), 'CUDA device not
double capable', /skip

    a = [[1.0D, 2.0D, 1.0D], [4.0D, 10.0D, 15.0D], [3.0D, 7.0D, 1.0D]]
    dims = size(a, /dimensions)
    m = dims[0]
    n = dims[1]

    b = [2.0D, 3.0D, 7.0D]

    standard = la_least_squares(a, b, /DOUBLE)

    da = gpuputarr(a, ERROR=err)
    assert, err eq 0, 'CUDA error: %s', gpu_errormessage(err)

    db = gpuputarr(b, ERROR=err)
    assert, err eq 0, 'CUDA error: %s', gpu_errormessage(err)

    dresult = gpuleast_squares(da, db, ERROR=err)
    assert, err eq 0, 'CUDA error: %s', gpu_errormessage(err)

    result = gpugetarr(dresult, ERROR=err)
    assert, err eq 0, 'CUDA error: %s', gpu_errormessage(err)

    error = total(abs(standard - result), /preserve_type)
    assert, error lt self.tolerance * n * 10.0, 'incorrect result: error
= %g', error

```

```
gpuFree, [da, db, dresult]
```

```
    return, 1  
end
```

Mike

--

Michael Galloy

www.michaelgalloy.com

Modern IDL: A Guide to IDL Programming (<http://modernidl.idldev.com>)

Research Mathematician

Tech-X Corporation
