Speeding up the legpts-command in CHEBFUN

Lukas Fath
KIT

10. Januar 2012

Speeding up the legpts-command in CHEBFUN

1. Introduction

In chapter 19 of the ATAP-book to the CHEBFUN-package for MATLAB you can find a comparison between the Gauß quadrature and the Clenshaw-Curtis quadrature. To calculate the nodes and weights for the Clenshaw-Curtis quadrature simple type

```
[nodes weights] = chebpts(n);
```

into the MATLAB's CLI. For the Gauß quadrature the nodes and weights are given by the following command:

```
[nodes weights] = legpts(n);
```

But while the *legpts*-function is implemented in rather slow MATLAB-code, the *chebpts*-function uses the highly optimized FFT built into MATLAB. In fact MATLAB uses the FFTW-library written in C. So it is not very surprising that *chebpts* is a lot faster[1] than *legpts*:

```
>> tic,[nodes weights] = chebpts(1000000); toc;
Elapsed time is 0.521917 seconds.

>> tic, [nodes weights] = legpts(1000000); toc;
Elapsed time is 39.290675 seconds.
```

So why not use the ability of MATLAB to call C++ code and speed up the *legpts*-command?

2. Calling C++-code in MATLAB

MATLAB provides a interface to use extern C++-subroutines. Out of the C++-code Matlab creates so-called MEX-files which can be called and executed like "'normal"' Matlab functions.

First you need to configure your system. Type

```
>> mex -setup
```

and choose a suitable compiler. A list of all supported compilers is available on the homepage of MathWorks.

Successfully completed the setup you can compile C++-source-files with the following commands:

[1] performed on a 64bit-OS, Core2Duo @ 2.0 GHz

```
>> mex <filename>.cpp
```

Now you have created a MEX-file and it can be called like the usual m-files. To handle the data transfer between Matlab and C++ the source file needs a special Gateway function called mexFunction. Details on how it works exactly can be found in the example files and descriptions on the MathWorks' hompage.

3. Speeding up the *legpts*-command

Using the possibilites shown in section 2 we compile *alg1_leg4.cpp* by typing

```
mex alg1_leg4.cpp
```

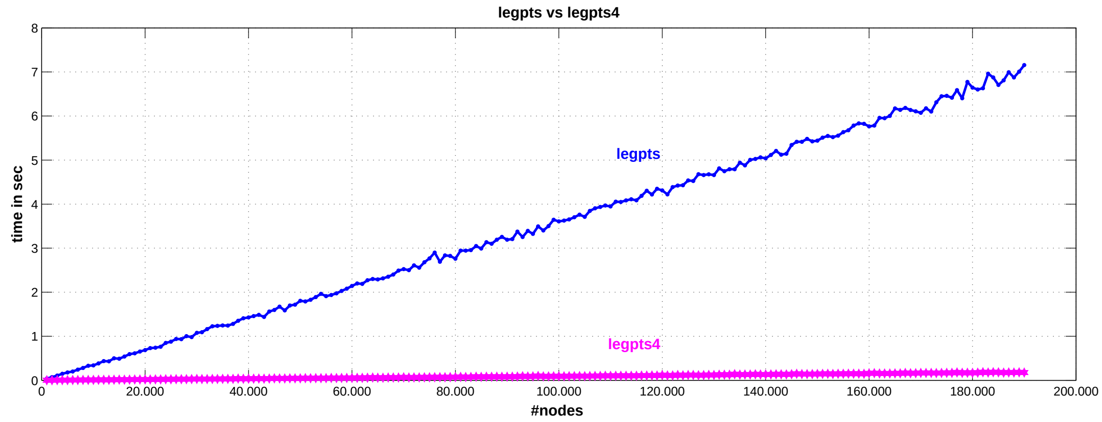
This C++-file contains an implementation of Glaser, Liu and Rohklin's fast algorithm to calculate the nodes and weights of the Gauß quadrature. Copy the file *legpts4.m* in the same directory. *Legpts4.m* offeres the same interface like the *legpts*-command in CHEBFUN, but instead it uses the algorithm in *alg1_leg4.cpp*:

```
>> tic; [nodes weights] = legpts4(1000000); toc
Elapsed time is 0.983534 seconds.
```

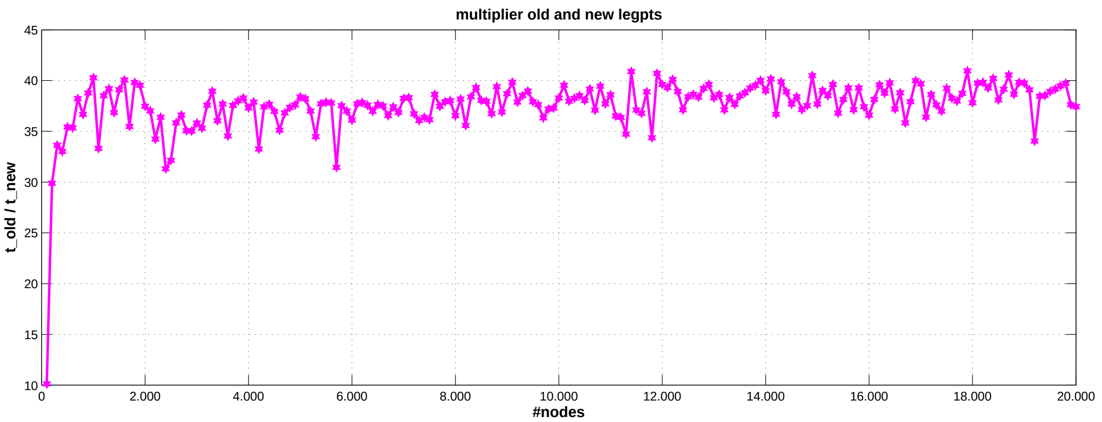
To show the exactness of the new algorithm we compare the nodes and weights with the results of the old implementation (as an example $n = 1.000.000$):

```
>> tic;[nodes_old weights_old]=legpts(1000000);toc
Elapsed time is 40.410179 seconds.

>> max(abs(nodes-nodes_old))
ans =
1.1102e-016

>> max(abs(weights-weights_old))
ans =
3.0934e-018
```

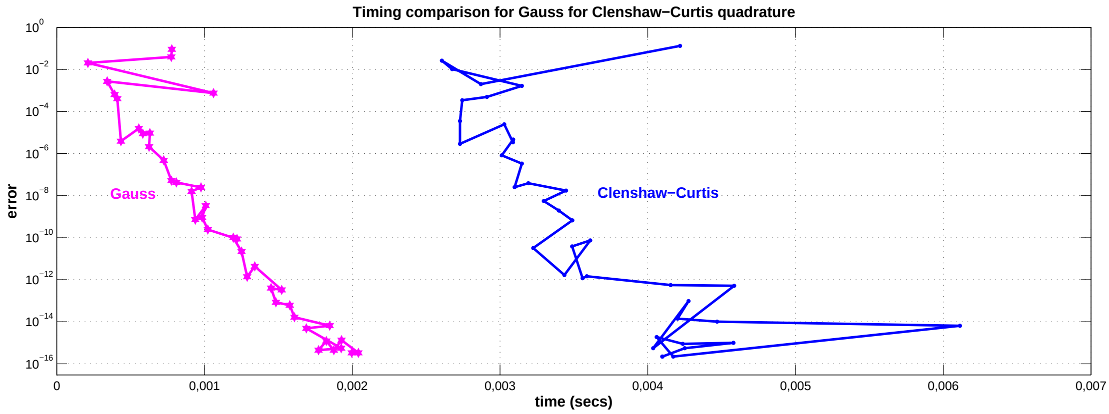
That looks perfect: the errors are around double precision. Let's have a closer look and compare the performance of both implementations. The first figure displays the computing time per nodes:



As expected we observe a tremendous increase in speed. The trouble of first having to compile the new function is definitely worth it.



For a wide range of nodes the new function is about 35-40 times faster than the old one! So here is the 'new' version of the time comparison of chapter 19 in the ATAP book (see [Tre11], p.141). It plots the accuracy as a function of the computing time for the harder integral (19.11):



The MATLAB code used to generate these comparisons can be found in *example.m*.

4. CONCLUSIONS/TODO

- legpts can easily be speeded up and compete with the FFT/chebpts on a DualCore system
- legpts4 uses SINGLE-CORE, FFT is 'cheating' and uses more CORES...but that's exactly one of the many strengths of FFT; is there a way to parallelise legpts4?
- advantage of FFT is clearly the ability of using all cores on multicore systems and that it is built-in in (nearly) every mathematical software package.
- code only tested on win64-architecture with MATLAB R2011b and MS VS 10.0. Does this work with older versions or on other systems?

References

[GLR07] Glaser, Andreas ; Liu, Xiangtao ; Rokhlin, Vladimir: A Fast Algorithm for the Calculation of the Roots of Special Functions. In: *SIAM J. Scientific Computing* 29 (2007), Nr. 4, S. 1420–1438

[THD11] Trefethen, L. N. ; Hale, N. ; Driscoll, T. A.: Chebfun version 4.1.1864, http://www2.maths.ox.ac.uk/chebfun/. (2011)

[Tre11] Trefethen, Lloyd N.: *Approximation Theory and Approximation Practice.* June 2011. – draft, Oxford University