MATLAB® News & Notes

Picking up the Pace with the MATLAB Profiler

by Ned Gulley

The new MATLAB JIT-Accelerator can significantly accelerate your code by transforming high-level MATLAB commands into native machine instructions.

This article explains how you can achieve optimum speed using the MATLAB Profiler. The Profiler shows where your code is slow and can offer specific suggestions about how to take advantage of the JIT-Accelerator.

Profiling for Speed

This script computes a familiar fractal, the Mandelbrot set. (We found this file at www.students.tut.fi/~warp/MandScripts/.)

```
1 xmax=2;
2 steps=200;
3 maxiter=32;
4 Z=0;
5
6 for m=1:steps
7 for n=1:steps
8 c=-xmax+2*xmax*n/steps-.5 + i*(-xmax+2*xmax*m/steps);
9 z=c;
10 for r=0:maxiter
11 z=z*z+c;
12 if abs(z)>2 break
13 end
14 end
15 Z(m,n)=sqrt(r/maxiter);
16 end
17 end
```

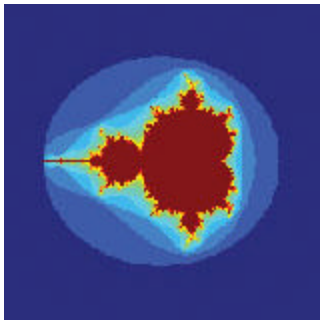

Figure 1. The Mandelbrot set in the variable Z looks like this when we plot it with the image command.

To find out how long this script takes to run, bring up the Profiler and run the code.

```
>> profile viewer
```

File mandelbrot1, time = 3.175 sec

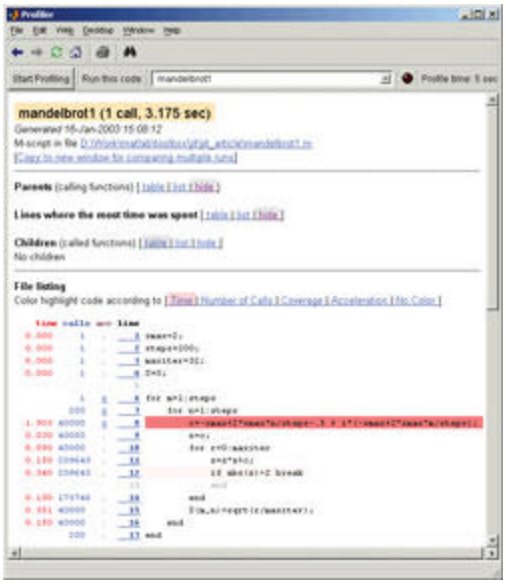

Figure 2: The Profiler display. Click on image to see enlarged view.

The file runs in 3.175 seconds. The Profiler display shows the most time-consuming line with the darkest background color. Line 8 took the most time, 1.903 seconds.

```
8 c=-xmax+2*xmax*n/steps-.5 + i*(-xmax+2*xmax*m/steps);
```

One way to speed up that line is to read the messages provided by the MATLAB JIT-Accelerator. Look at the file listing at the bottom of the Profiler, and you will see four columns:

- `time` the amount of time spent on a given line
- `calls` the number of calls made to that line
- `acc` indicating whether or not the line has been successfully accelerated
- `line` the line number

A dot `(.)` in the acc column means the line accelerated; an `x` means that it did not. Clicking on the x shows why that line did not accelerate. For example:

```
Reference to 'i' in a script cannot be accelerated unless 'i' is a
defined variable. To apply the builtin 'i', use the notation '1i'.
```

Define i

The problem here is that we haven't defined i explicitly. MATLAB can figure out that we mean `sqrt(-1)`, but we lose time in doing that. We can save MATLAB the trouble by typing `1i` instead of `i`, or simply by adding a new line `saying i = sqrt(-1)`.

If we do the second of these options, our new version has the line

```
5 i=sqrt(-1);
```

Let's use the Profiler again to find out how fast it runs now.

```
File mandelbrot2, time = 1.382 sec
```

What else can we try? There are still x marks in the accelerator column on lines 7 and 8. Since

they're in the loop body, they have a big impact on the overall speed. Let's click on the x in line 8 to determine the problem.

```
Variable 'r', used as a FOR loop control variable, was referred to on
line 16 outside of the body of the FOR loop.
```

Confine the Loop Variable

So `r`, which is used as the loop control variable in the innermost loop, is also used in a calculation below the loop. We can fix this by introducing a new variable, `rmax`. This removes all references to `r` outside the loop. We need to be careful with the new variable `rmax`, though. It shouldn't be initially defined inside the loop, since the body of a loop might never be evaluated, as in the expression

```
>> for n=2:1, disp('This never runs'); end
```

Therefore, we define `rmax` above the loop since we intend to refer to it below the loop. Now, instead of the original code in the inner loop, we have

```
11 rmax=maxiter;
12 for r=0:maxiter
13 z=z*z+c;
14 if abs(z)>2
15 rmax=r;
16 break
17 end
18 end
19 Z(m,n)=sqrt(rmax/maxiter);
```

This saves a little time.

```
File mandelbrot3, time = 0.861 sec
```

Preallocate Matrices

Another change we might make to speed things is to preallocate the space allotted to large matrices. This saves you the time required to "grow" the matrix dynamically inside a loop. Let's preallocate the matrix Z, since we know exactly how big it's going to be.

```
4 Z=0;
```

becomes

```
4 Z=zeros(steps);
```

The new time
`File mandelbrot4, time = 0.631 sec`
is a significant improvement for a very small change.

Unroll the Complex Arithmetic

For one last change, we can take the complex math calculations that MATLAB is managing for us and turn them into non-complex calculations. That is, we can manage the imaginary and real components separately. This is somewhat inconvenient, but it pays off in this case because non-complex math runs much faster in MATLAB version 6.5.

```
File mandelbrot5, time = 0.291 sec
```

Here is the entire final program.

```
1 xmax=2;
2 steps=200;
```

```
3 maxiter=32;
4 Z=zeros(steps);
5
6 for m=1:steps
7 ci=(-xmax+2*xmax*m/steps);
8 for n=1:steps
9 cr=-xmax+2*xmax*n/steps-.5;
10 zr=cr;
11 zi=ci;
12 rmax=maxiter;
13 for r=0:maxiter
14 zrn=zr*zr-zi*zi+cr;
15 zin=2*zi*zr+ci;
16 zi=zin;
17 zr=zrn;
18 if (zr*zr+zi*zi)>4,
19 rmax=r;
20 break
21 end
22 end
23 Z(m,n)=sqrt(rmax/maxiter);
24 end
25 end
```

Run the Timing Loop

Let's time all the programs we've written both with and without the JIT-Accelerator. The speed-up is dramatic. Notice that preallocating memory is helpful in either case, but unrolling the complex arithmetic is a good idea only if you have the JIT-Accelerator. The end result of our optimization was to knock around 90% off the original time. Without the JIT-Accelerator, these changes would have actually slowed our code down by 80%.

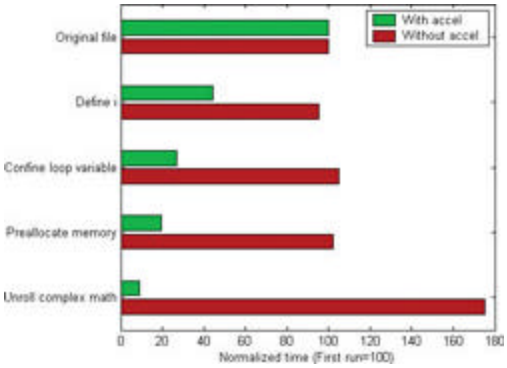

Figure 3. Comparing JIT and non-JIT timing. Click on image to see enlarged view.

Conclusion

The JIT-Accelerator is a useful addition to MATLAB, and the Profiler helps you take full advantage of it. Remember: optimize your code only according to your needs. Maximum speed might be more important than readability or maintainability, but if it isn't, then there's no need to alter code that is already well structured and fast enough. Finally, keep in mind that we will continue to improve MATLAB, including the JIT-Accelerator, making MATLAB faster even as it gets easier to use.

Next Article

related topics:

Using MathWorks Products For... | Training | MATLAB Based Books | Third-Party Products