Advanced example: elliptic curve factorization

Elliptic Curve Method (ECM) factorization is a powerful, beautifully parallelizable method for factoring large integers. It was with ECM that Zilla found two new factors of F13 in early
1991 (see "General" help). The idea is to run a certain ECM program on each machine, but also to give each machine a unique seed parameter (which technically speaking determines the "elliptic curve" being used).

Our ECM example uses an executable called 'ecm', designed by R. E. Crandall for parallel factoring runs. The ecm program resides in the Zilla.app directory. Also used in this example is a feeder program which we provide. The feeder source is in the Zilla.app directory under the name 'ecmfeed.m', and the pre-compiled ecm.feed is ready for use.

The ECM routines use some number theory packages that permit numbers up to 2^1000000 (that is one million bit numbers) to be handled. Because of FFT large multiply techniques, the time for a multiplication grows roughly linearly over the upper parts of this range. Thus, if you can do 100 elliptic curves per day for a 1000 digit number-to-be-factored, you can do (with the same input curve parameters) 10 per day for 10000 digit numbers.

Instructions for the ECM example

1. Add to your network a number of machines, on which you have at least Partial Permission. By default, these machines will use the username and password which you entered when Zilla was first started. If you need to use a different account on any of the machines which you have added, enter the proper username and password using the host inspector panel.

The feeder program which we will use does the following. For each machine in the network, a different command line is created, with different parameters to the 'ecm' program. The feeder then sends that command line to Zilla, and tells Zilla to launch all of the commands.

2. To load the feeder, select 'Command Setup' from the 'Control' menu.
3. Click on the radio button next to 'From feeder'.
4. Click the 'set...' button next to the 'From feeder' button. An open panel will then be brought up.
5. Choose 'ecm.feed' in the Zilla.app directory. This is the pre-compiled example feeder, and it is here where you would load in your own custom feeder.
6. Create a directory called Zfactor in the home directory of the user (usually you) who was assigned to the machines (ie. the user whose name you entered in the 'login:' field of the inspector panel).
7. Move the executable "ecm" to the Zfactor directory. That is where each machine will
find the executable.

(The provided ecm.feed example attempts to find factors of F7 = 2^128 + 1. If you are
doing this F7 example, the following step 8 is entirely unnecessary because the provided
feeder does not require an Nfile.)
8. (This step is only necessary if the number to be factored is neither of the Mersenne form
2^n ± 1, nor of the Fermat form 2^n + 1. Thus, if you are factoring a number not of either form, you need to specify the digits of the number in a file. The ecm program figures out these digits for you for the cases 2^n +± 1; and, indeed, runs much faster in such special cases)

Create Nfile and place it inside the Zfactor directory. Nfile should contain the number to be factored in ASCII format, followed by a linefeed. Again, since F7 is of the Fermat form, this step 8 is
entirely unnecessary, as the ecm program will generate its own internal representation of F7.

9. Use 'Run...' from the 'Control' menu to start up the factorization.

At this point, Zilla will execute the Feeder program, which will tell Zilla how to proceed. After a while, several files should appear in the Zfactor directory where you placed Nfile. These files, named log...machine (where machine will be replaced by the name of the machine on which this process ran) will contain the results of the factoring.

Technical details about the factoring are discussed next.

The ecm program

The program ecm will, given enough time and/or enough machines, find non-trivial factors of a large composite integer N. The program is to be run on one or more machines, each machine using some unique seed.

The simplest way to use ecm is to run on the m-th machine the executable:

> ecm 0 0 seedm b < Nfile > outputm ; if the number to be factored is in Nfile and not of the form 2^q +± 1
> ecm 1 q seedm b > outputm ; if the number to be factored is of the form 2^q + 1.
> ecm ±1 q seedm b > outputm ; if the number to be factored is of the form 2^q±1.

where "seedm" is a machine-dependent seed, and "outputm" is a machine-dependent filename. The parameter "b" is the smoothness limit in the ECM, which amounts to a certain internal limit on the calculations. The best rule of thumb is: b = 10000 is suitable when the unknown factor has about15 digits, and b = 100000 is suitable for about 25 digits. Run time is roughly proportional to b (per curve). Here are some typical output files "outputm" for ecm runs on a typical day:

-rw-r--r-- 1 richard 140 Mar 20 19:52 log.f10.scrawl

```
-rw-r--r--   1 richard        389 Mar 20 19:52 log.f10.cauldron
-rw-r--r--   1 richard        436 Mar 20 19:52 log.f10.eugene
-rw-r--r--   1 richard        683 Mar 20 19:52 log.f10.alexandria
-rw-r--r--   1 richard        121 Mar 20 19:52 log.f10.trilithon
```

You can see that the feeder program arranges to put the output data into a filename that involves the hostname, for ease in maintenance of large directories. The ecmfeed.m example in Zilla.app shows how this naming is arranged.

The output of the ecm program is a series of elliptic curve parameters used, with any discovered factors showing up with exclamation mark prefix. For example, the command:

```
> ecm ±1 101 666 1000  > log.m101.mymachine
```

will find factors of M101 = 2^101 ± 1, which appear next to exclamation marks in the log file:

```
> cat   log.m101.mymachine
A:2860448219137
!7432339208719
A:2860448219138
!7432339208719
A:2860448219139
!341117531003194129
>
```

This output means that elliptic curve parameters (all three starting 2860...) were tried, and in all three cases a factor was found. Indeed,

2^101 ± 1 = 7432339208719 * 341117531003194129

so that only three elliptic curves sufficed. What is most common, though, is hundreds or thousands of curves (therefore that many occurrences of A:) with none,

or one, or some small number of lucky factors, i.e. a small number of exclamation marks (if any) in the output file.

Challenge to Zilla users

The mathematical community would like to see certain numbers factored. You can try to find new factors of the following numbers (but of course there are many, many more numbers of interest beyond what we list here):

M445 = 2^445 ± 1, has two prime factors, each (very probably) greater than 27 digits long,
neither of which has been found.
Other Mersenne numbers that remain incompletely factored include M467, M479, M481.

F10 = 2^1024 + 1, has only two known factors, 45592577 and 6487031809; the remaining
part is, however, composite and must have more factors.
F14 = 2^(2^14) + 1 = 2^16384 + 1, is the mysterious fourteenth Fermat Number. Unlike its little cousin F10 above, F14 has no known factors and yet it must have factors as has
been proven.
F15, F16, F17, F18 etc. are gigantic, largely unresolved Fermat numbers that provide good
challenges. The "ecm" program run on Zilla is especially efficient for these cases because
it uses a special, FFT-based large multiply.

Numerators of Bernoulli Numbers (Numerator[BernoulliB[n]] in *Mathematica*), for even n, are good challenges, especially for n in the region 100-to-200.
Repunit numbers, made up of all 1's in decimal, are very tough if you take large enough cases.
Some of these are primes, for example the repunit having19 ones is prime.

For these numbers best represented in decimal, you would modify the feeder ecmfeed.m so that
the command given be:

> ecm 0 0 seedm b < Nfile > outputm

where Nfile has the Bernoulli numerator, or a repunit series of 1's, in ASCII, followed by a single
newline.

You might even get started with factoring by trying to reproduce the discovery of the new factor 2663848877152141313 of F13 (see "General" help). This would involve a command line:

> ecm 1 8192 seedm 20000 > outputm

which is easy to arrange by simple modification of ecmfeed.m. It is not unlikely that you can find a brand new factor of F13 in this way, especially if there exist such a factor with no more than about 30 digits.