Unbalanced Multiplication in GMP

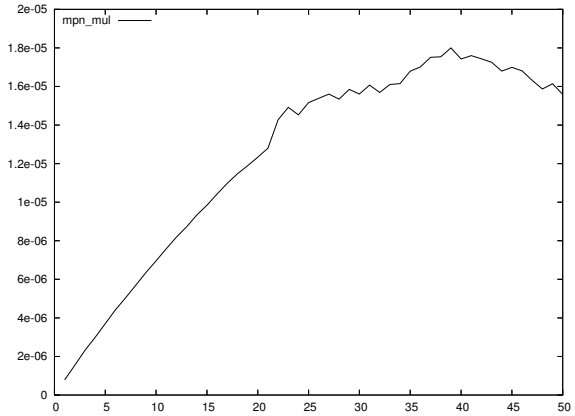
GMP 4.2.1

The following graphs show the behaviour of the `mpn_mul` function for operands of $N-n$ and n limbs respectively — with N fixed and $1 \leq n \leq N/2$ — for different values of the product size of N limbs. These tests were performed with gmp-4.2.1 on a Pentium M at 800Mhz, with the following default thresholds:

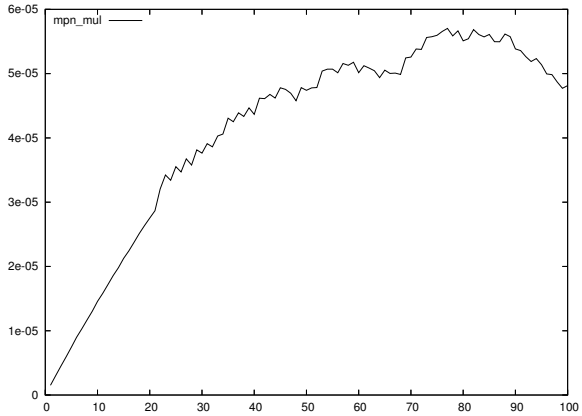
```
#define MUL_KARATSUBA_THRESHOLD          22
#define MUL_TOOM3_THRESHOLD              74
#define MUL_FFT_TABLE  { 464, 928, 1920, 4608, 10240, 24576, 98304, 393216,
                         1572864, 6291456, 0 }
#define MUL_FFT_MODF_THRESHOLD          480
#define MUL_FFT_THRESHOLD              3328
```

The strategy of gmp-4.2.1 to multiply two operands of m and n limbs, with $m \geq n$, is the following:

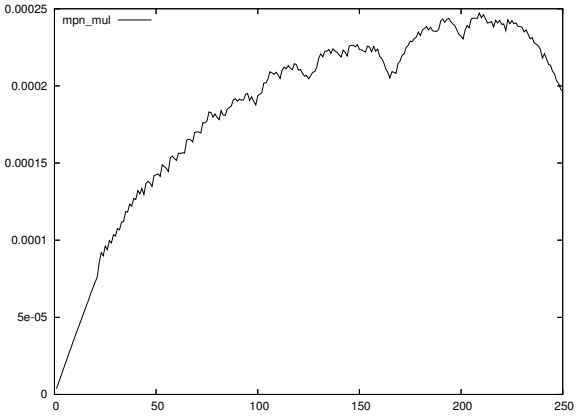
- if the smaller size n is smaller than the Karatsuba threshold, use the schoolbook multiplication;
- if the smaller size n is larger than the FFT threshold, perform one full FFT product of size $m+n$;
- if $m = qn + r$, perform q products of size $n \times n$, and one product of size $n \times r$.



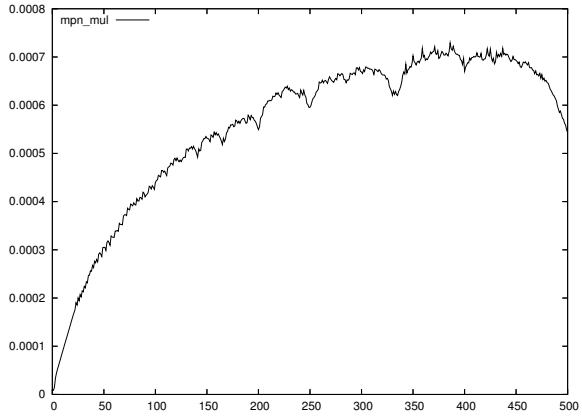
$N = 100$



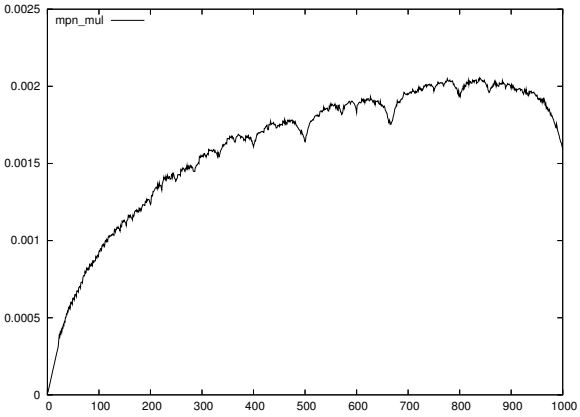
$N = 200$



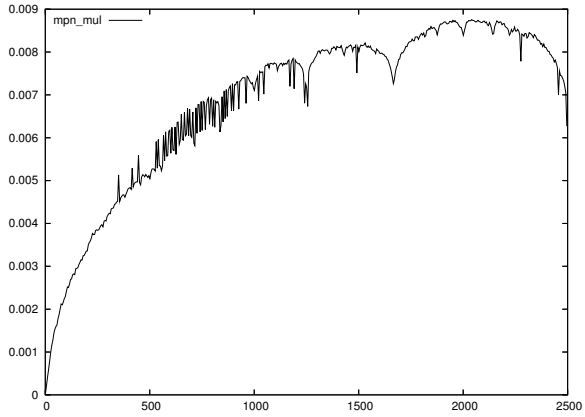
$N = 500$



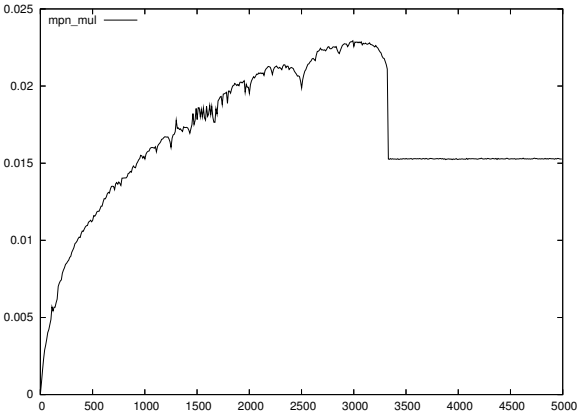
$N = 1000$



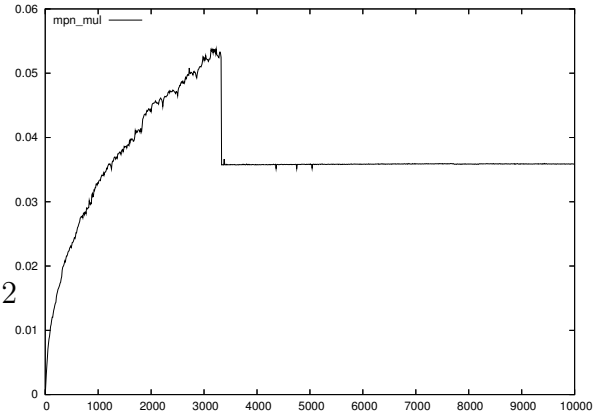
$N = 2000$



$N = 5000$



$N = 10000$



$N = 20000$

Proposed Improvements

It appears from the graphs for $N = 10000$ and $N = 20000$ that the FFT — which gives the lines on the right — could be used for smaller values of n, from $n = 1000$ approximately.

Assuming the FFT has complexity N^α to multiply two operands of total size N, to multiply two operands of size N/q and $N(1-1/q)$, the cost is N^α for one single FFT, and $(q-1)(2N/q)^\alpha$ for $q-1$ products of operands of size N/q.

A FFT splitting operands in 2^k parts has complexity $\alpha = \frac{k}{k-2}$. Solving the equation $(q-1)(2/q)^{k/(k-2)} = 1$ gives the following values:

k	5	6	7	8	9	10
q	3.3	5.2	8.2	12.4	18.7	27.6

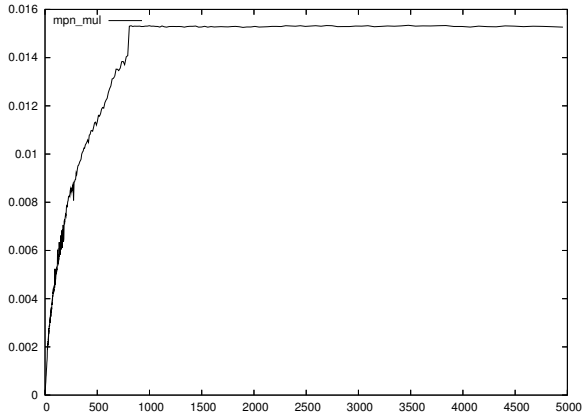
We propose to replace the following lines in `mpn/generic/mul.c`:

```
  if (ABOVE_THRESHOLD (vn, MUL_FFT_THRESHOLD))
    {
      mpn_mul_fft_full (prodp, up, un, vp, vn);
      return prodp[un + vn - 1];
    }
```

by:

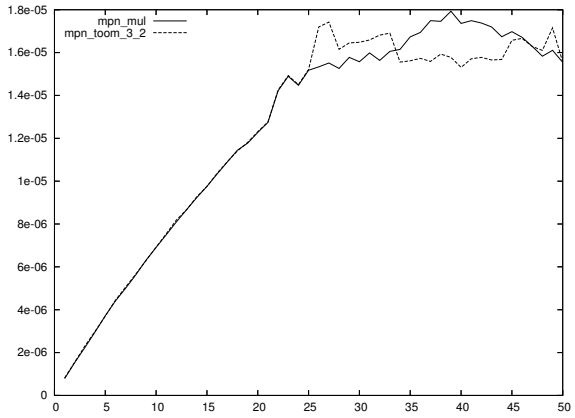
```
    double Q[] = { 2.0, 3.3, 5.2, 8.2, 12.4, 18.7, 27.6 };
    unsigned long k = mpn_fft_best_k (un + vn, 0);
    if (ABOVE_THRESHOLD ((un + vn) / 2, MUL_FFT_THRESHOLD)
&& (double) (un + vn) / (double) vn <= Q[(k > 10) ? 6 : k - 4])
      {
mpn_mul_fft_full (prodp, up, un, vp, vn);
return prodp[un + vn - 1];
      }
```

We then get the following curve for $N = 10000$:

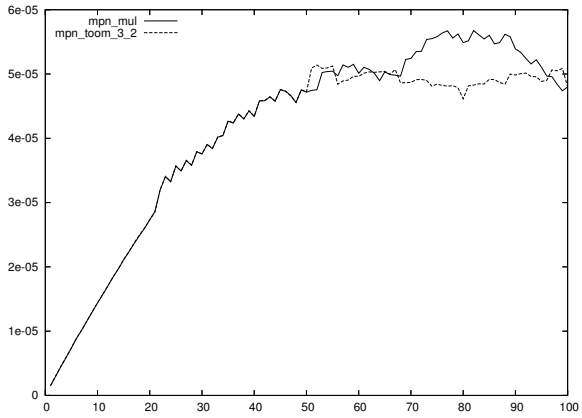


$N = 10000$

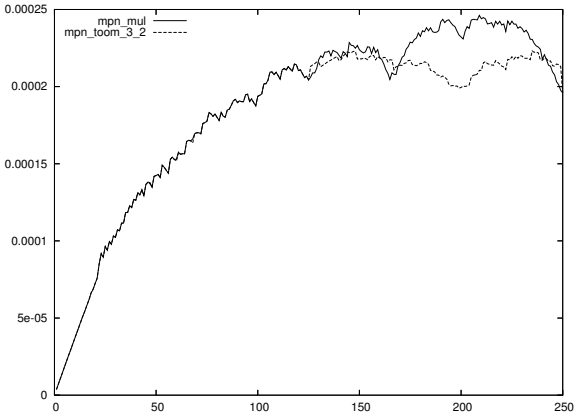
With the Toom-2.5 algorithm, which multiplies one operand of $2n < a \leq 3n$ limbs with another one of $n < b \leq 2n$ limbs, we get the following curves (for $N = 10000$ and $N = 20000$, Toom-2.5 is slower than the FFT):



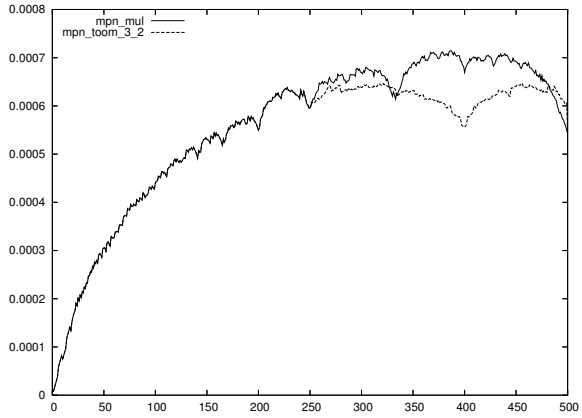
$N = 100$



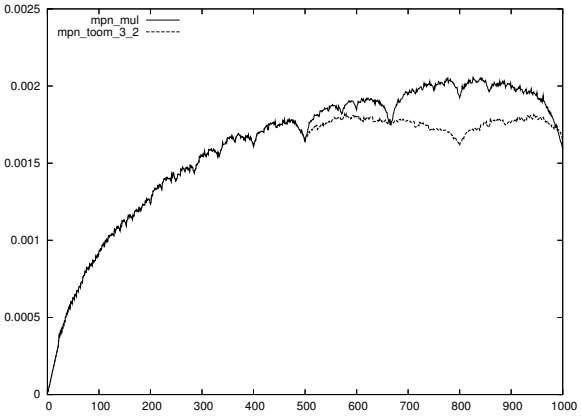
$N = 200$



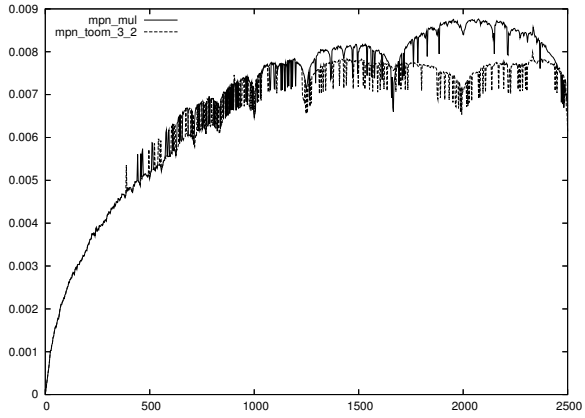
$N = 500$



$N = 1000$



$N = 2000$



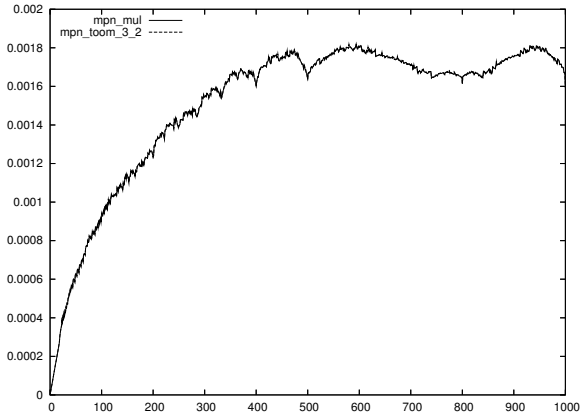
$N = 5000$

The curves suggest that Toom-2.5 may be used instead of `mpn_mul` for $a/3 < b < a$ in the Karatsuba or Toom-Cook range. This corresponds to the following additional code, just after the above test for FFT in file `mul.c`:

```
if (2 * ((vn + 1) / 2) < un && (un + 2) / 3 < vn)
  {
    mpn_toom_3_2 (prodp, up, un, vp, vn);
    return prodp[un + vn - 1];
  }
```

(Note that `mpn_toom_3_2` calls recursively `mpn_mul` for the product of the leading coefficients.)

This gives the following graph:



$N = 2000$