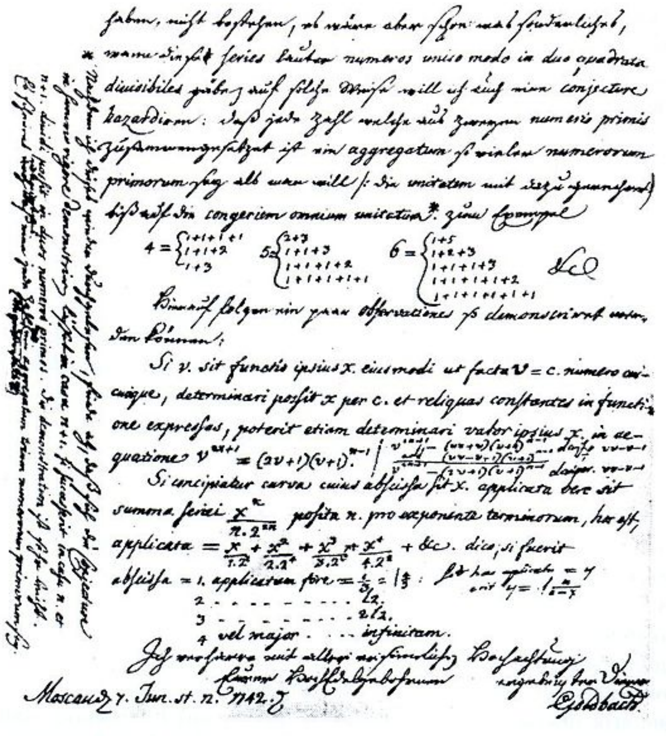
The Goldbach Conjecture

Sections

Summing Primes

Here we verify the conjecture for small numbers.

§**1.** ¡p¿On 7 June 1742, Christian Goldbach wrote a letter from Moscow to Leonhard Euler in Berlin making quot;eine conjecture hazardirenquot; that every even number greater than 2 can be written as a sum of two primes.¡sup id="fnred:1"¿¡a href="fn:1" rel="footnote"¿1¡/a¿¡/sup¿ Euler did not know if this was true, and nor does anyone else.¡/p¿


haben, nicht beweisen, ob wäre aber schon mahl sehr deutlich,
* wenn die einfache series lauter numeros unico modo in duo quadrata divisibiles gäbe, auf solche Weise will ich auch eine conjecture hazardiren: dass jede Zahl welche aus zweyen numeris primis zusammengesetzt ist ein aggregatum so vieler numerorum primorum seye als man will (die unitatem mit dazu gerechnet) bis auf die congeriem omnium unitatum.* zum Exempel

$$4 = \begin{cases}1+1+1+1\\1+1+2\\1+3\end{cases} \quad 5 = \begin{cases}2+3\\1+1+3\\1+1+1+2\\1+1+1+1+1\end{cases} \quad 6 = \begin{cases}1+5\\1+2+3\\1+1+1+3\\1+1+1+1+2\\1+1+1+1+1+1\end{cases}$$

&c.

Hierauf folgen ein paar observationes so demonstrirt werden können:

Si v. sit functio ipsius x. eiusmodi ut facta $v = c$. numero cuicunque, determinari possit x per c. et reliquas constantes in functione expressas, poterit etiam determinari valor ipsius x. in aequatione $v^{2x+1} = (2v+1)(v+1)^{x-1}$ [illegible]

Si concipiatur curva cuius abscissa sit x. applicata vero sit summa seriei $\frac{x^n}{n.2^{nn}}$ posito n. pro exponente terminorum, hoc est, applicata $= \frac{x}{1.2} + \frac{x^2}{2.2^4} + \frac{x^3}{3.2^9} + \frac{x^4}{4.2^{16}} + \&c$. dico, si fuerit abscissa $= 1$. applicatam fore $= \frac{1}{3} = 1\frac{1}{3}$ [illegible]

2 . . . . $l2$.
3 . . . . $2l2$.
4 vel major . . . . infinitam.

Ich verharre mit aller ersinnlichen Hochachtung
Ew. Hochedelgebohrnen ergebenster Diener
Goldbach

Moscau d. 7. Jun. st. n. 1742.

Nachdem ich dieses wieder durchgelesen, finde ich, dass sich die conjecture in summo rigore demonstriren lässet in casu n+1, si successerit in casu n. et n+1 dividi possit in duos numeros primos. Die demonstration ist sehr leicht. Es scheinet wenigstens, dass eine jede Zahl, die grösser ist als 2, ein aggregatum trium numerorum primorum sey.

¡p¿Goldbach, a professor at St Petersburg and tutor to Tsar Peter II, wrote in several languages in an elegant cursive script, and was much valued as a letter-writer, though his reputation stands less high today.¡sup id="fnred:2"¿¡a href="fn:2" rel="footnote"¿2¡/a¿¡/sup¿ All the same, the general belief now is that primes are just plentiful enough, and just evenly-enough spread, for Goldbach to be right. It is known that:¡/p¿ ¡p¿(a) every even number is a sum of at most six primes (Ramar, 1995), and (b) every odd number is a sum of at most five (Tao, 2012).¡/p¿ ¡ul class="inwebfootnotetexts"¿¡li class="footnote" id="fn:1"¿¡p class="inwebfootnote"¿¡sup id="fnref:1"¿¡a href="fn:1" rel="footnote"¿1¡/a¿¡/sup¿ quot;Greater than 2quot; is our later proviso: Goldbach needed no such exception because he considered 1 a prime number, as was normal then, and was sometimes said as late as the early twentieth century.¡a href="fnref:1" title="return to text"¿ x21A9;¡/a¿¡/p¿ ¡/li¿ ¡li class="footnote" id="fn:2"¿¡p class="inwebfootnote"¿¡sup id="fnref:2"¿¡a href="fn:2" rel="footnote"¿2¡/a¿¡/sup¿ Goldbach, almost exactly a contemporary of Voltaire, was a good citizen of the great age of Enlightenment letter-writing. He and Euler exchanged scholarly letters for over thirty years, not something Euler would have kept up with a duffer. Goldbach was also not, as is sometimes said, a lawyer. See: http://mathshistory.st-andrews.ac.uk/Biographies/Goldbach.html. An edited transcription of the letter is at: http://eulerarchive.maa.org//correspondence/letters/OO0765.pdf¡a href="fnref:2" title="return to text"¿ x21A9;¡/a¿¡/p¿ ¡/li¿¡/ul¿

§2. ¡p¿Computer verification has been made up to around $10^{18}, but by rather better methods than the one we use here. We will$
$/p >$

define 100

```
#include <stdio.h>


int main(int argc, char *argv[]) {
    for (int i=4; i<RANGE; i=i+2) /* stepping in twos to stay even */
        ⟨Solve Goldbach's conjecture for i 2.1⟩ ;
}
```

§2.1. ¡p¿This ought to print:¡/p¿

```
    $ goldbach/Tangled/goldbach
    4 = 2+2
    6 = 3+3
    8 = 3+5
    10 = 3+7 = 5+5
    12 = 5+7
    14 = 3+11 = 7+7
    ...
```

¡p¿We'll print each different pair of primes adding up to i. We only check in the range 2 lt;= j lt;= i/2 to avoid counting pairs twice over (thus 8 = 3+5 = 5+3, but that's hardly two different ways).¡/p¿

```
    \pdfdest num 100 fit $\langle${\xreffont\pdfliteral direct{1 1 0 0 k}Solve Goldbach's conjecture
for i {\sevenss 2.1}}\special{PDF:0 g}$\rangle$ $\equiv$

    printf("%d", i);
    for (int j=2; j<=i/2; j++)
        if ((isprime(j)) && (isprime(i-j)))
            printf(" = %d+%d", j, i-j);
    printf("\n");
```

This code is used in §2.

The Sieve of Eratosthenes

A fairly fast way to determine if small numbers are prime, given storage.

S1 Storage; S2 Primality

§**1.** ¡p¿This technique, still essentially the best sieve for finding prime numbers, is attributed to Eratosthenes of Cyrene and dates from the 200s BC. Since composite numbers are exactly those numbers which are multiples of something, the idea is to remove everything which is a multiple: whatever is left, must be prime.¡/p¿ ¡p¿This is very fast (and can be done more quickly than the implementation below), but (a) uses storage to hold the sieve, and (b) has to start right back at 2 - so it can't efficiently test just, say, the eight-digit numbers for primality.¡/p¿

```
int still_in_sieve[RANGE + 1];
int sieve_performed = FALSE;
```

§**2.** ¡p¿We provide this as a function which determines whether a number is prime:¡/p¿

define 1

define 0

```
int isprime(int n) {
    if (n <= 1) return FALSE;
    if (n > RANGE) { printf("Out of range!\n"); return FALSE; }
    if (!sieve_performed) ⟨Perform the sieve 2.1⟩ ;
    return still_in_sieve[n];
}
```

§**2.1.** ¡p¿We save a little time by noting that if a number up to ¡code¿RANGE¡/code¿ is composite then one of its factors must be smaller than the square root of ¡code¿RANGE¡/code¿. Thus, in a sieve of size 10000, one only needs to remove multiples of 2 up to 100, for example.¡/p¿

```
    \pdfdest num 100 fit $\langle${\xreffont\pdfliteral direct{1 1 0 0 k}Perform the sieve {\sevenss
2.1}}\special{PDF:0 g}$\rangle$ $\equiv$
```

```
    ⟨Start with all numbers from 2 upwards in the sieve 2.1.1⟩ ;
    for (int n=2; n*n <= RANGE; n++)
        if (still_in_sieve[n])
            ⟨Shake out multiples of n 2.1.2⟩ ;
    sieve_performed = TRUE;
```

This code is used in §2.

§**2.1.1.** `\pdfdest num 100 fit $\langle${\xreffont\pdfliteral direct{1 1 0 0 k}Start with all numbers from 2 upwards in the sieve {\sevenss 2.1.1}}\special{PDF:0 g}$\rangle$ $\equiv$`

```
    still_in_sieve[1] = FALSE;
    for (int n=2; n <= RANGE; n++) still_in_sieve[n] = TRUE;
```

This code is used in §2.1.

§2.1.2. \pdfdest num 100 fit $\langle${\xreffont\pdfliteral direct{1 1 0 0 k}Shake out multiples of n {\sevenss 2.1.2}}\special{PDF:0 g}$\rangle$ $\equiv$

```
    for (int m= n+n; m <= RANGE; m += n) still_in_sieve[m] = FALSE;
```

This code is used in §2.1.