On Squarefree Values of some Univariate Polynomials.

Helmut Preininger
Vienna, Austria
May 7, 2018
helmut.preininger [at] chello [dot] at
hosted at: www.vixra.org

Abstract

We consider univariate Polynomials, P(s), of the form (a1 * s + b1)*...*(ak * s + bk), where a1,..,ak,b1,..,bk are natural numbers and the variable s is squarefree. We give an algorithm to calculate, for a arbitrary s, the probability that the value of P(s) is squarefree.

1 Introduction

2 Some Notation and Constants

In the rest of the paper is:
$n_i \in \mathbb{N}_+$
$s_i \in \mathbb{S}$ ... the squarefree elements of $\mathbb{N}$
$p, p_i \in \mathbb{P}$ ... the prime elements of $\mathbb{N}$
$P(n) := P(X) \in \mathbb{N}[X]$, a polynomial where $X \in \mathbb{N}_+$
$P(s) := P(X) \in \mathbb{S}[X]$, a polynomial where $X \in \mathbb{S}$
$\mathfrak{P}(X; some\ constraints)$ ... the probability of $a \in X$ and $a \in \mathbb{S}$
$\mathfrak{P}(X + Y)$ ... the probability of $a \in X, b \in Y$ and $a + b \in \mathbb{S}$
$\mathfrak{P}(X * Y)$ ... the probability of $a \in X, b \in Y$ and $a * b \in \mathbb{S}$

$f(x) = \{a_1, a_2, \ldots, a_{p^2-1}\}$ is the distribution of the remainders of $n \equiv i \mod p^2$

Here are some constants:
$\mathfrak{P}(\mathbb{N}) = \prod_{i=1}^{\infty}(1 - \frac{1}{p_i^2}) = \prod_{i=1}^{\infty} \frac{p_i^2-1}{p_i^2} = \frac{6}{\pi^2} \approx 0.607..$
$\mathfrak{P}(\mathbb{S} + \mathbb{S}) = \prod_{i=1}^{\infty}(1 - \frac{1}{p_i^2-1}) = \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1} \approx 0.530..$
$\mathfrak{P}(\mathbb{S} * \mathbb{S}) = \prod_{i=1}^{\infty}(1 - \frac{1}{(p_i+1)^2}) = \prod_{i=1}^{\infty} \frac{p_i}{p_i+1} \approx 0.775..$
$\prod_{i=1}^{\infty}(1 - \frac{1}{p_i^2-1}) = \prod_{i=1}^{\infty} \frac{p_i^2-3}{p_i^2-2} \approx 0.388..$

3 $\mathfrak{P}((n + n_1) \wedge \cdots \wedge (n + n_m))$

Consider $\mathfrak{P}(n + n_0) \wedge (n + n_1')$ and $n_0 < n_1'$. We search for $s = (n + n_0) \in \mathbb{S}$ and $(s + (n_1' - n_0)) = (s + n_1) \in \mathbb{S}$. But $\mathfrak{P}(n + n_0) = 6/\pi^2$ and therefore is is enough to consider $\mathfrak{P}(s + n_1)$.

3.1 $\mathfrak{P}(s+n_1)$

In this subsection we proof the

Proposition 1. *Let* $n_1 = s_1 q_1$ *with* $q_1 = \prod_{j=1}^{m} p_j^{2\alpha_j}$ *then*

$$\mathfrak{P}(s+n_1) = \prod_{j=1}^{m} \frac{p_j^2-1}{p_j^2-2} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

Lemma 2.

$$\mathfrak{P}(s+s_1) = \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

Proof. We have $f(s) = \{0,1,\dots,1\}$ and $f(s_1) = \{0,\dots,0,1,0\dots,0\}$ (i.e. 1 at the i-th place $1 \le i < p^2$ and 0 otherwise).
We have p^2-1 possible pairs and 1 sufficient pair and get $\gamma = \{1/(p_1^2-1),\dots\}$ or $\mathfrak{P}(s+s_1) = \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$. □

Lemma 3. *Let* p_j *prime,* $k \in \mathbb{N}$ *and* $k > 1$ *then*

$$\mathfrak{P}(s+p_j^k) = \frac{p_j^2-1}{p_j^2-2} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

Proof. We have $f(s) = \{0,1,1,\dots\}$ and with $k > 1$

$$f(p_j^k) = \begin{cases} \{0,\dots,0,1,0,\dots\}, & p_j \neq p \text{ (1 at place } 0 \le i \le p^2, 0 \text{ otherwise)} \\ \{1,0,0,\dots\}, & \text{otherwise} \end{cases}$$

We get $\gamma = \{1/(p_1^2-1), 1/(p_{j-1}^2-1), 0, 1/(p_{j+1}^2-1),\dots\}$ and therefore

$$\mathfrak{P}(s+p_j^k) = \frac{\prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}}{1-\frac{1}{p_j^2-1}} = \frac{p_j^2-1}{p_j^2-2} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

□

Lemma 4. *Let* $q_1 = \prod_{j=1}^{m} p_j^{2\alpha_j}$ *then*

$$\mathfrak{P}(s+q_1) = \prod_{j=1}^{m} \frac{p_j^2-1}{p_j^2-2} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

Proof. It follows with Lemma 3 and the "Inclusion-Exclusion-Principle". Details □

Proof. of Proposition 1 The Proposition follows with Lemma 2 and Lemma 4. Details? □

3.2 $\mathfrak{P}(s+n_1; e(n_1)|s)$

Definition 5. *Let $n=\prod_{j=1}^{m} p_j^{\alpha_j}$. We define $e(n)$ as $e(n)=\prod_{j=1}^{m} p_j$.*

In this subsection we proof the

Theorem 6. *Let $n_1=\prod_{j=1}^{v} p_j^{\alpha_j}$, $A=\{p_j|\alpha_j=1\}$ and $B=\{p_j|\alpha_j>1\}$ then*

$$\mathfrak{P}(s+n_1; e(n_1)|s)=\prod_{p_a\in A}\frac{(p_a-2)(p_a+1)}{p_a^2-2}\prod_{p_b\in B}\frac{p_b^2-1}{p_b^2-2}\prod_{i=1}^{\infty}\frac{p_i^2-2}{p_i^2-1}$$

Lemma 7.

$$\mathfrak{P}(s+p_d;\, p_d|s)=\frac{(p_d-2)(p_d+1)}{p_d^2-2}\prod_{i=1}^{\infty}\frac{p_i^2-2}{p_i^2-1}$$

Proof. We have

$$f(s)=\begin{cases}\{0,\dots,0,1,0\dots0,1,0\dots\}, & \text{for } p_d=p;\ (1 \text{ at all places i for p divde i and } i>0)\\ \{0,1,\dots,1\}, & \text{otherwise}\end{cases}$$

and

$$f(p_d)=\begin{cases}\{0,\dots,0,1,0\dots\}, & \text{for } p_d=p;\ (1 \text{ at place } p_d)\\ \{0,\dots,0,1,0,\dots\}, & \text{for } p_d\neq p;\ (1 \text{ at place } \neq p_d)\end{cases}$$

We get $\gamma=\{1/(p_1^2-1),\dots,1/(p_{d-1}^2-1),1/(p_d-1),1/(p_{d+1}^2-1),\dots\}$

$$\mathfrak{P}(s+p_d,\, p_d|s)=\frac{\prod_{i=1}^{\infty}\frac{p_i^2-2}{p_i^2-1}}{1-\frac{1}{p_d^2-1}}\left(1-\frac{1}{p_d-1}\right)=\frac{(p_d-2)(p_d+1)}{p_d^2-2}\prod_{i=1}^{\infty}\frac{p_i^2-2}{p_i^2-1}$$

□

Lemma 8. *Let $s_1=\prod_{j=1}^{m} p_j$ then*

$$\mathfrak{P}(s+s_1; s_1|s)=\frac{\prod_{j=1}^{m}(p_j-2)\prod_{j=1}^{m}(p_j+1)}{\prod_{j=1}^{m}(p_j^2-2)}\prod_{i=1}^{\infty}\frac{p_i^2-2}{p_i^2-1}$$

Proof. With Lemma 7 and the "Inclusion-Exclusion-Principle"we get the above formula. □

Lemma 9. *Let $k>1$ then*

$$\mathfrak{P}(s+p_d^k;\, p_d|s)=\frac{p_d^2-1}{p_d^2-2}\prod_{i=1}^{\infty}\frac{p_i^2-2}{p_i^2-1}$$

Proof. We have

$$f(s)=\begin{cases}\{0,\dots,0,1,0\dots0,1,0\dots\}, & \text{for } p_d=p;\ (1 \text{ at all places i for p divde i and } i>0)\\ \{0,1,\dots,1\}, & \text{otherwise}\end{cases}$$

and

$$f(p_d) = \begin{cases} \{1, 0, \dots\}, & \text{for } p_d = p \\ \{0, \dots, 0, 1, 0, \dots\}, & \text{for } p_d \neq p \end{cases}$$

We get $\gamma = \{1/(p_1^2 - 1), \dots, 1/(p_{i-1}^2), 0, 1/(p_{i+1}^2 - 1), \dots\}$

$$\mathfrak{P}(s + p_d^k; p_d | s) = \mathfrak{P}(s + p_d^2; p_d | s) = \frac{1}{\left(1 - \frac{1}{p_d^2 - 1}\right)} \prod_{i=1}^{\infty} \frac{p_i^2 - 2}{p_i^2 - 1} = \frac{p_d^2 - 1}{p_d^2 - 2} \prod_{i=1}^{\infty} \frac{p_i^2 - 2}{p_i^2 - 1}$$

□

Lemma 10. *Let* $q_1 = \prod_{j=1}^{m} p_j^{2\alpha_j}$ *then*

$$\mathfrak{P}((s + q_1); e(q_1) | s = \prod_{j=1}^{m} \frac{p_j^2 - 1}{p_j^2 - 2} \prod_{i=1}^{\infty} \frac{p_i^2 - 2}{p_i^2 - 1}$$

Proof. With Lemma 9 and the "Inclusion-Exclusion-Principle"we get the expected Lemma. □

Proof. With Lemma 8, Lemma 10 and the "Inclusion-Exclusion-Principle"we get the expected result. □

3.3 $\mathfrak{P}((s + n_1) \wedge (s + n_2))$

In this subsection we proof the

Theorem 11. *Let* $n_1, n_2 \in \mathbb{N}$, $n_2 > n_1$, $B_G = \{p \in \mathbb{P} | (p^2 | n_1) \wedge (p^2 | n_2)\}$, $B_1\{p \in \mathbb{P} | (p \notin B_G) \wedge (p^2 | n_1)\}$, $B_2 = \{p \in \mathbb{P} | (p \notin B_G) \wedge (p^2 | n_2)\}$, $B_{1,2} = \{p \in \mathbb{P} | (p \notin B_G) \wedge (p^2 | (n_2 - n_1))\}$ *and* $B_D = B_1 \cap B_2 \cap B_{1,2}$ *then*

$$\mathfrak{P}((s + n_1) \wedge (s + n_2)) = \prod_{p_g \in B_G} \frac{p_g^2 - 1}{p_g^2 - 3} \prod_{p_d \in B_D} \frac{p_d^2 - 2}{p_d^2 - 3} \prod_{i=1}^{\infty} \frac{p_i^2 - 3}{p_i^2 - 1}$$

Lemma 12. *Let* $s_1, s_2 \in \mathbb{S}$, $s_2 > s_1$ *and* $n_u = s_2 - s_1 = s_d q_d$ *with* $q_d = \prod_{j=1}^{m} p_j^{2\alpha_j}$ *then*

$$\mathfrak{P}((s + s_1) \wedge (s + s_2)) = \prod_{j=1}^{m} \frac{p_j^2 - 2}{p_j^2 - 3} \prod_{i=1}^{\infty} \frac{p_i^2 - 3}{p_i^2 - 1}$$

Proof. We have

$$f(s) = \{0, 1, \dots\}$$

Now we numerate all elements of $f(s)$ and get

$$f(s) = \{0_0, 1_1, 1_2, \dots, 1_k, \dots, 1_{p^2-1}\}$$

$$f(s + s_1) = \{1, 1, \dots, 1, 0, 1, \dots\} \text{ ; a circular right shift with } k_1 = s_1 \mod p^2$$

$$f(s + s_2) = \{1, 1, \dots, 1, 0, 1, \dots\} \text{ ; a circular right shift with } k_2 = s_2 \mod p^2$$

Note, we canceled in both shifts the first element.
Case 1: $p^2 \nmid (s_2 - s1)$: $k_1 \neq k_2$ we canceled two distinct elements.

$$\left(1 - \frac{2}{p^2-1}\right) = \frac{p^2-3}{p^2-1}$$

Case 2: $p^2|(s_2 - s_1)$: $k_1 = k_2$ we canceled only one element.

$$\left(1 - \frac{1}{p^2-1}\right) = \frac{p^2-2}{p^2-1}$$

Therefore we get (we replace the part $\frac{p_j^2-3}{p_j^2-1}$ in $\prod_{i=1}^{\infty} \frac{p_i^2-3}{p_i^2-1}$, $j = i$)

$$\mathfrak{P}((s+s_1) \wedge (s+s_2)) = \prod_{j=1}^{m} \frac{p_j^2-2}{p_j^2-3} \prod_{i=1}^{\infty} \frac{p_i^2-3}{p_i^2-1}$$

□

Proof. of Theorem 11
We saw the principle in Lemma 12. Now we have four cases:

Case B_G: We cancel no element and get 1.
Case B_1: We cancel one element and get $\frac{p^2-2}{p^2-1}$.
Case B_2: We cancel one element and get $\frac{p^2-2}{p^2-1}$.
Case $B_{1,2}$: We cancel one element and get $\frac{p^2-2}{p^2-1}$.

Therefore we get (we replace the part $\frac{p_j^2-3}{p_j^2-1}$ in $\prod_{i=1}^{\infty} \frac{p_i^2-3}{p_i^2-1}$, $j = i$)

$$\mathfrak{P}((s+n_1) \wedge (s+n_2)) = \prod_{p_g \in B_G} \frac{p_g^2-1}{p_g^2-3} \prod_{p_d \in B_D} \frac{p_d^2-2}{p_d^2-3} \prod_{i=1}^{\infty} \frac{p_i^2-3}{p_i^2-1}$$

□

3.4 Calculation of $\mathfrak{P}((s+n_1) \wedge \cdots \wedge (s+n_m))$

In this subsection we give an algorithm to calculate $\mathfrak{P}((s+n_1) \wedge \cdots \wedge (s+n_m))$. We split the algorithm in two parts and a helper procedure.

The Environment: Let $n_1, \ldots, n_m \in \mathbb{N}_+$, we call it coefficients, $n_1 < n_2 < \cdots < n_m$.

Helper 1: The Procedure FillCoeffs(aCoeffs(),tCoeffs)
The coefficients are given in the form $n_1, n_2, \ldots, n_m$ and the procedure FillCoeffs() fill the array aCoeffs() with the values $n_1, \ldots$.

```
; Fill Array of aCoeffs() (Array call by reference)
; Programming language: PureBasic
; tCoeffs: a String like "1,2,3" ->
;       aCoeffs(0) = 1, aCoeffs(1) = 2, aCoeffs(2)= 3
Procedure.i FillCoeffs(Array aCoeffs.i(1),tCoeffs.s)
  Protected kCount.i,kp.i
  kCount = CountString(tCoeffs,",") + 1
  If kCount <= 1
    ReDim aCoeffs(1)
    aCoeffs(1) = 0
  Else
    ReDim aCoeffs(kCount - 1)
  EndIf
  For kp = 1 To kCount
    aCoeffs(kp-1) = Val(StringField(tCoeffs,kp,","))
  Next
  SortArray(aCoeffs(),#PB_Sort_Ascending)
  ProcedureReturn kCount
EndProcedure
```

Part 1: The Procedure CntCancels(aCoeffs(),tPrime)
For a particular prime p, $f(s)$ (i.e. the distribution of the remainders $s \mod p^2$) is of the form

$$f(s) = \{0, 1, 1, \ldots, 1\}$$

In $f(s + n_1)$ is the 0, $n_1 \mod p^2$ times shifted to the right direction and the leftmost element of $f(s + n - 1)$ is, in general, a 1. All s with $(s + n_1) \equiv 0 \mod p^2$ are not squarefree and where canceled.
The procedure CntCancels() count the cancellations for a particular prime (note, without multiplicity).

```
; Return #Cancellations
; Programming language: PureBasic
Procedure.i CntCancels(Array aCoeffs.i(1),tPrime.i)
  Protected k2Prime.i,kp.i,kCancel.i
  k2Prime = tPrime * tPrime
  Dim aCancels.i(k2Prime - 1)
  kp = 0
  ; Zero Coeffs are automatically not considered
  ;   since we start with s that is squarefree
  While kp <= ArraySize(aCoeffs())
    ; we start with s -> {0,1,1,..}
    ;   (i.e. there are no s with (s % k2Prime) = 0)
    ; 0 + c shift the Zero on place (c % k2Prime)
    ;   and so there no s with (s+c) % k2Prime = 0
    If aCoeffs(kp) > 0
```

```
      aCancels(aCoeffs(kp) % k2Prime) = 1
    EndIf
    kp + 1
  Wend
  ; now we count all cancellations
  ;   (note: without the 0-te element)
  kp = 1
  While kp <= ArraySize(aCancels())
    kCancel + aCancels(kp)
    kp + 1
  Wend
  ProcedureReturn kCancel
EndProcedure
```

Part 2: The Procedure CalcWedge(Array aCoeffs.i(1))
We count for all $p^2 \leq n_m$ the cancellations and calculate the factor of the infinite product. There is no additional cancellation for $p^2 > n_m$.

```
; Calculate (s+n_1)^(s+n_2)^.........
; Programming language: PureBasic
Procedure.d CalcWedge(tCoeffs.s)
  Protected k2Prime.i,kPrime.i,kCancel.i,kMax.i,kCount
  Protected kWedge.d,kZero.i,kCoeffs.i,kp.i
  *gSieve = xNewSieve(2) ; a squarenumber sieve
  ; fill the aCoeffs Array
  Dim aCoeffs.i(1)
  kCount = FillCoeffs(aCoeffs(),tCoeffs)
  kWedge = 1.0
  kPrime = 2
  k2Prime = 4
  ; test only k2Prime <= kMax (saftey: 2 * Max(Coeff))
  ; (aCoeffs is sorted!!!)
  kMax = aCoeffs(ArraySize(aCoeffs())) * 2
  While k2Prime <= kMax
    kCancel = CntCancels(aCoeffs(),kPrime)
    If kCancel = (k2Prime - 1) ; no s exist
      kWedge = 0.0
      kZero = 1
      Break
    Else
      If kCancel > 0
        kWedge = kWedge * (Pow(kPrime,2) -
            (kCancel + 1.0)) / (Pow(kPrime,2) - 1.0)
      EndIf
    EndIf
    kPrime = xSVNextPrime(*gSieve,kPrime)
```

```
    k2Prime = kPrime * kPrime
  Wend
  If kZero = 0 ; some s exist
    ; Count the Coeffs > 0
    kCoeffs = 0
    For kp = 0 To ArraySize(aCoeffs())
      If aCoeffs(kp) > 0
        kCoeffs + 1
      EndIf
    Next
    ; Calc the Product for p >= kPrime
    ; kPrime < 1048000 of SegLength if Debugger ON
    While kPrime < 1048000
      kWedge = kWedge * (Pow(kPrime,2) -
          (kCoeffs + 1.0)) / (Pow(kPrime,2) - 1.0)
      kPrime = xSVNextPrime(*gSieve,kPrime)
    Wend
  EndIf
  ProcedureReturn kWedge
EndProcedure
```

4 Polynomials of the Form $P(n) = \prod_{i=1}^{m}(n + n_i)$

Let $P(n) \in \mathbb{N}[n]$ a polynomial of the form $P(n) = \prod_{i=1}^{m}(n + n_i)$, where $n_i \in \mathbb{N}$. The following observations about the values of $P(n)$ are useful:

PS1) $P(n) \in \mathbb{S} \Longrightarrow \forall_{i=1,..,m}(n + n_i) \in \mathbb{S}$
PS2) $P(n) \in \mathbb{S} \Longrightarrow \forall_{i,j=1,..,m;\, i<j} \gcd((n + n_i), (n + n_j)) = 1$
PS3) $P(n) \in \mathbb{S} \Longleftrightarrow \forall_{i,j=1,..,m;\, i<j}(n + n_i)(n + n_j) \in \mathbb{S}$
PS3') $P(n) \in \mathbb{S} \Longleftrightarrow \forall_{i,j=1,..,m;\, i<j}[(n + n_i) \in \mathbb{S}] \wedge [\gcd((n + n_i), (n + n_j)) = 1]$

PS2) implies, it is enough to study only a set of pairwise distinct $n_i's$. Since the addition and multiplication is commutative we set $n_1 < n_2 < \cdots < n_i < \cdots < n_m$.

Lemma 13.

$$\mathfrak{P}(\prod_{i=1}^{m}(n + a_i)) = \mathfrak{P}(n\prod_{i=2}^{m}(n + (n_i - n_1))) = \mathfrak{P}(\mathbb{N})\mathfrak{P}(s\prod_{i=2}^{m}(s + (n_i - n_1)))$$

Proof. Claim 1: $\mathfrak{P}(n + n_1) = \mathfrak{P}(\mathbb{N})$, $n_1 \in \mathbb{N} \cup \{0\}$
Proof of Claim 1: We have $f(n) = \{1, 1, \ldots, 1\}$ and $f(n_1) = \{0, \ldots, 0, 1, 0 \ldots, 0\}$ (i.e. 1 at the i-th place $0 \leq i < p^2$ and 0 otherwise) .
We have p^2 possible pairs and 1 sufficient pair and get $\gamma = \{1/p_1^2, \ldots\}$ or $\mathfrak{P}(n + n_1) = \mathfrak{P}(\mathbb{N})$.
Claim 2: $\mathfrak{P}((n + n_1)(n + n_2) = \mathfrak{P}(n(n + (n_2 - n_1)))$, $n_1, n_2 \in \mathbb{N}$, $n_1 < n_2$
Proof of Claim 2: 1) With the Claim 1, we get for both polynomials $(p^2)^2$ possible pairs and p^2

sufficient pairs.

Except of the first n_1 numbers, $(n+n_1)(n+n_2)$ and $n(n+(n_2-n_1))$ run over the same numbers, therefore we have the same $\gcd's$.
Complete the proof: Since the values of the factors of the polynomial are squarefree numbers (PS1), it is enough to consider only the set $\mathbb{S}$. □

4.1 Polynomials of the Form $P(s) = s(s+n_1)$.

We proof the

Theorem 14. *Let* $n_1 = \prod_{j=1}^{m} p_j^{\alpha_j}$ *with* $\alpha_j > 0$ *then*

$$\mathfrak{P}(s(s+n_1)) = \frac{\prod_{j=1}^{m} p_j(p_j-1)}{\prod_{j=1}^{m}(p_j^2-2)} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

Since $s(s+n_1) \in \mathbb{S} \Longrightarrow \gcd(s, s+n_1) = 1$, we count the cases where $\gcd(s, s+n_1) > 1$ and subtract it from $\mathfrak{P}(s+n_1)$.

Proposition 15. *Let* $s_1 = \prod_{j=1}^{m} p_j$ *then*

$$\mathfrak{P}(s(s+s_1)) = \frac{\prod_{j=1}^{m} p_j(p_j-1)}{\prod_{j=1}^{m}(p_j^2-2)} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

Proof. With Lemma 2 we have $\mathfrak{P}(s+s_1) = \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$, but until now we have not considered the case $\gcd(s, s+s_1) > 1$.
Since $\gcd(s, s+s_1) > 1 \Longrightarrow \exists p_j : p_j | s$, with Lemma 8, [PRE02] Proposition 6 (there are $s/(p+1)$ squarefree numbers with $p|s$) and the "Inclusion-Exclusion-Principle"finally we get:

$$\mathfrak{P}(s(s+s_1)) = \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1} \left(1 - \frac{1}{\prod_{j=1}^{m}(p_j+1)} \cdot \frac{\prod_{j=1}^{m}(p_j-2)(p_j+1)}{\prod_{j=1}^{m} p_j^2-2}\right)$$

and we get

$$\mathfrak{P}(s(s+s_1)) = \frac{\prod_{j=1}^{m} p_j(p_j-1)}{\prod_{j=1}^{m}(p_j^2-2)} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

□

Proposition 16. *Let* $q_1' = \prod_{j=1}^{m} p_j^{2\alpha_j}$, $q_1 = \prod_{j=1}^{m} p_j^2$ *and* $q_1 | q_1'$ *then*

$$\mathfrak{P}(s(s+q_1')) = \mathfrak{P}(s(s+q_1)) = \frac{\prod_{j=1}^{m} p_j(p_j-1)}{\prod_{j=1}^{m}(p_j^2-2)} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

Proof. With Lemma 4 we have $\mathfrak{P}(s+q_1') = \mathfrak{P}(s+q_1) = \frac{\prod_{j=1}^{m}(p_j^2-1)}{\prod_{j=1}^{m}(p_j^2-2)} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$, but until now we have not considered the case $\gcd(s, s+q_1) > 1$.

Since $\gcd(s, s+q_1) > 1 \longrightarrow \exists p_j \,:\, p_j | s$, with Lemma 9, [PRE02] Proposition 6 (there are $s/(p+1)$ squarefree numbers with $p|s$) and the "Inclusion-Exclusion-Principle"finally we get:

$$\mathfrak{P}(s(s+q_1')) = \mathfrak{P}(s(s+q_1)) = \frac{\prod_{j=1}^{m}(p_j^2-1)}{\prod_{j=1}^{m}(p_j^2-2)} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1} \prod_{j=1}^{m} \left(1 - \frac{1}{p_j+1}\right)$$

□

Proof. of Theorem 14
The Theorem follows with Proposition 15 and Proposition 16. □

4.2 Polynomials of the Form $P(s) = (s+n_1)(s+n_2)$

In this section we proof the

Proposition 17. *Let $n_1, n_2 \in \mathbb{N}$, $n_2 > n_1$, $G = \{p \in \mathbb{P} | (p|n_1) \wedge (p|n_2)\}$ and $B = \{p \in \mathbb{P} | (p \notin G) \wedge (p|(n_2 - n_1))\}$ then*

$$\mathfrak{P}((s+s_1)(s+s_2)) = \prod_{p_k \in G} \frac{p_k^2 - p_k}{p_k^2 - 3} \cdot \prod_{p_j \in B} \frac{p_j^2 - p_j - 1}{p_j^2 - 3} \prod_{i=1}^{\infty} \frac{p_i^2 - 3}{p_i^2 - 1}$$

Lemma 18. *Let $s_1, s_2 \in \mathbb{S}$, $k = 1, 2, \ldots$, p_d prime, $p_d^k = s_2 - s_1$ and s_1, s_2, p_d mutually prime then*

$$\mathfrak{P}((s+s_1)(s+s_2)) = \frac{p_d^2 - p_d - 1}{p_d^2 - 3} \prod_{i=1}^{\infty} \frac{p_i^2 - 3}{p_i^2 - 1}$$

Proof. Since a $\gcd(s+s_1, s+s_2) > 1$ only occur if $p = p_d$, therefore we only consider this case:
We start with $f(s) = \{0, 1, 1, \ldots\}$, $f(s+s_1) = \{1, \ldots, 1, 0, 1, \ldots\}$ and $f(s+s_2) = \{1, \ldots, 1, 0, 1, \ldots\}$.
Only pairs of "1's" at place i in $f(s+s_1)$ and $(i+p) \mod p^2$ in $f(s+s_2)$ are canceled. Since $p \nmid s_1$ and $p \nmid s_2$ we cancel p pairs and get

$$1 - \frac{p_d}{p_d^2 - 1} = \frac{p_d^2 - p_d - 1}{p_d^2 - 1}$$

we replace the part $\frac{p_d^2-3}{p_d^2-1}$ in $\prod_{i=1}^{\infty} \frac{p_i^2-3}{p_i^2-1}$

$$\frac{p_d^2 - p_d - 1}{p_d^2 - 1} \cdot \frac{p_d^2 - 1}{p_d^2 - 3} = \frac{p_d^2 - p_d - 1}{p_d^2 - 3}$$

and finally

$$\mathfrak{P}((s+s_1)(s+s_2)) = \frac{p_d^2 - p_d - 1}{p_d^2 - 3} \prod_{i=1}^{\infty} \frac{p_i^2 - 3}{p_i^2 - 1}$$

□

Lemma 19. *Let $s_1, s_2 \in \mathbb{S}$ where $s_d = s_2 - s_1 \in \mathbb{S}$, $s_d = \prod_{j=1}^{m} p_j$ and s_1, s_2, s_d are mutually prime then*

$$\mathfrak{P}((s+s_1)(s+s_2)) = \frac{\prod_{j=1}^{m}(p_j^2 - p_j - 1)}{\prod_{j=1}^{m}(p_j^2 - 3)} \prod_{i=1}^{\infty} \frac{p_i^2 - 3}{p_i^2 - 2}$$

Proof. The Lemma follows with Lemma 18 and the "Inclusion-Exclusion-Principle". □

Proof. of Proposition 17
Case 1: $(p \in G) \Longrightarrow p|(n_2 - n_1)$ we cancel $p-1$ pairs

$$\left(1 - \frac{p-1}{p^2-1}\right) = \frac{p^2-p}{p^2-1}$$

Case 2: $(p \notin G) \wedge (p|(n_2 - n_1))$ we cancel p pairs (see Lemma 18)

$$\left(1 - \frac{p}{p^2-1}\right) = \frac{p^2-p-1}{p^2-1}$$

Finally we replace $\frac{p^2-3}{p^2-2}$ with Case 1 or Case 2 and get the expected result. □

4.3 Calculation of $\mathfrak{P}((s+n_1)\cdots(s+n_m))$

In this subsection we give an algorithm to calculate $\mathfrak{P}((s+n_1)\cdots(s+n_m))$.

The Environment: Let $n_1, \ldots, n_m \in \mathbb{N}_+$, we call it coefficients, $n_1 < n_2 < \cdots < n_m$.

```
; First step: like CntCancels
; Second step: consider Gcd-Cancels
;  there are only Gcd's if:
;       (Coff_p Mod tPrime) = (Coeff_m Mod tPrime)
; Programming language: PureBasic
Procedure.i CntGcdCancels(Array aCoeffs.i(1),tPrime.i)
  ; Return #Cancellations in respect with Gcd
  Protected k2Prime.i,kp.i,kGcds.i,km.i,kDelta.i,kq.i
  k2Prime = tPrime * tPrime
  Dim aGcds.i(k2Prime - 1)
  kp = 0
  ; Zero Coeffs are automatically not considered
  ; Consider the Cancels like CntCancels
  While kp <= ArraySize(aCoeffs())
    aGcds(aCoeffs(kp) % k2Prime) = 1
    kp + 1
  Wend
  kp = 0
  ; Consider all combination c_p - c_m  p < m for Gcd-Cancels
```

```
  While kp < ArraySize(aCoeffs())
    km = kp + 1
    While km <= ArraySize(aCoeffs())
      ; there are only Gcd-cancellations if
      ;     (c_i % tPrime) = (c_j % tPrime)
      If (aCoeffs(kp) % tPrime) = (aCoeffs(km) % tPrime)
        kq = aCoeffs(kp) % tPrime
        While kq < k2Prime
          aGcds(kq) = 1
          kq + tPrime
        Wend
      EndIf
      km + 1
    Wend
    kp + 1
  Wend
  kp = 1
  While kp <= ArraySize(aGcds())
    kGcds + aGcds(kp)
    kp + 1
  Wend
  ProcedureReturn kGcds
EndProcedure

; Programming language: PureBasic
Procedure.d CalcProd(tCoeffs.s)
  Protected k2Prime.i,kPrime.i,kGcds.i,kMax.i,kCount
  Protected kProd.d,kZero.i,kCoeffs.i,kp.i
  *gSieve = xNewSieve(2)
  Dim aCoeffs.i(1)
  kCount = FillCoeffs(aCoeffs(),tCoeffs)
  If kCount = 1 ; only one Coeff -> CalcWedge
    ProcedureReturn CalcWedge(tCoeffs)
  EndIf
  kProd = 1.0
  kPrime = 2
  k2Prime = 4
  kMax = aCoeffs(kCount - 1) * aCoeff(kCount - 1)
  ; If k2Prime > kMax => no Gcd-Cancels are possible
  While k2Prime <= kMax
    kGcds = CntGcdCancels(aCoeffs(),kPrime)
    If kGcds = (k2Prime - 1)
      kProd = 0.0
      kZero = 1
      Break
```

```
    Else
      If kGcds > 0
        kProd = kProd * (Pow(kPrime,2) -
            (kGcds + 1.0)) / (Pow(kPrime,2) - 1.0)
      EndIf
    EndIf
    kPrime = xSVNextPrime(*gSieve,kPrime)
    k2Prime = kPrime * kPrime
  Wend
  If kZero = 0
    ; Count the Coeffs > 0
    kCoeffs = 0
    For kp = 0 To ArraySize(aCoeffs())
      If aCoeffs(kp) > 0
        kCoeffs + 1
      EndIf
    Next
    ; Calc the Product for p >= kPrime
    ; kPrime < 1048000 of SegLength if Debugger ON
    While kPrime < 1048000
      kProd = kProd * (Pow(kPrime,2) -
          (kCoeffs + 1.0)) / (Pow(kPrime,2) - 1.0)
      kPrime = xSVNextPrime(*gSieve,kPrime)
    Wend
  EndIf
  ProcedureReturn kProd
EndProcedure
```

5 $\mathfrak{P}((c_1n + d_1) \wedge \cdots \wedge (c_mn + d_m); c_i, d_i \in \mathbb{N})$

5.1 $\mathfrak{P}((c_1s + d_1); c_1 \in \mathbb{N}, d_1 \in \mathbb{S})$

In this subsection we proof the

Proposition 20. *Let* $c_1 \in \mathbb{N}_+$, $d_1 \in \mathbb{S}$, $B_2 = \{p \in \mathbb{P} | (p^2|c_1)\}$ *and* $B_G = \{p \in \mathbb{P} | (p \notin B_2) \wedge (p|c_1) \wedge (p|d_1)\}$ *then*

$$\mathfrak{P}(c_1s + d_1; c_1 \in \mathbb{N}_+, d_1 \in \mathbb{S}) = \prod_{p_g \in B_G} \frac{p_g^2 - p_g - 1}{p_g^2 - 2} \prod_{p_b \in B_2} \frac{p_b^2 - 1}{p_b^2 - 2} \prod_{i=1}^{\infty} \frac{p_i^2 - 2}{p_i^2 - 1}$$

Lemma 21. *Let* p *prime then*

$$\mathfrak{P}(ps + p) = \mathfrak{P}(p(s + 1)) = \frac{p^2 - p - 1}{p^2 - 2} \prod_{i=1}^{\infty} \frac{p_i^2 - 2}{p_i^2 - 1}$$

Proof. We consider only the distribution of the remainders of p^2. We start with

$$f(s) = \{0, 1, 1, \dots\}$$

and

$$f(s+1) = \{1, 0, 1, 1, \dots\}$$

We cancel all "1" at positions i with $i \equiv 0 \mod p$ (p is a factor of all this squarefree numbers). Therefore we cancel "1" p times and get

$$\left(1 - \frac{p}{p^2-1}\right) = \frac{p^2-p-1}{p^2-1}$$

replace $\frac{p^2-2}{p^2-1}$

$$\frac{p^2-p-1}{p^2-1} \cdot \frac{p^2-1}{p^2-2} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

□

Lemma 22. *Let p prime, $k = 2, 3, 4, \dots$ then*

$$\mathfrak{P}(p^k s + p) = \frac{p^2-1}{p^2-2} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

Proof. We consider only the distribution of the remainders of p^2. We start with

$$f(s) = \{0, 1, 1, \dots\}$$

and

$$f(p^k s) = \{p^2-1, 0, 0, \dots\}$$

we get

$$f(p^k s + p) = \{0, \dots, 0, p^2-1, 0, \dots\}$$

where $p^2 - 1$ is on the i-te place. Therefore no "1" is canceled and we only replace $\frac{p^2-2}{p^2-1}$ □

Lemma 23. *Let $c_1 = d_1 = s_1$, $s_1 = \prod_{j=1}^{m} p_j$ then*

$$\mathfrak{P}(s_1(s+1)) = \prod_{j=1}^{m} \frac{p_j^2-p_j-1}{p_j^2-2} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

Proof. With Lemma 21 and the "Inclusion-Exclusion-Principle"we get the expected result. □

Lemma 24. *Let $c_1, d_1 \in \mathbb{S}$, $\gcd(c_1, d_1) = 1$ and $c_1 = \prod_{j=1}^{m} p_j$ then*

$$\mathfrak{P}(c_1 s + d_1; \gcd(c_1, d_1) = 1) = \prod_{j=1}^{m} \frac{p_j^2-1}{p_j^2-2} \prod_{i=1}^{\infty} \frac{p_i^2-2}{p_i^2-1}$$

Proof. We consider only the distribution of the remainders of p_j^2. We start with

$$f(s) = \{0, 1, 1, \dots\}$$

Now $p_j \cdot s \equiv x \mod p_j^2$ where $x = 0, 1, \dots, p_j - 1$. For a cancellation it holds $p_j \cdot x + d_1 = k \cdot p_j^2$ and we get $d_1 = p_j(p_j - x)$ but $p_j \nmid d_1$. Therefore we have no cancellation and we get the expected result. □

Lemma 25. *Let* $c_1 = q_1 \in \mathbb{N}$, $d_1 \in \mathbb{S}$, $\gcd(q_1, d_1) = 1$ *and* $q_1 = \prod_{j=1}^{m} p_j^{2\alpha_j}$ *then*

$$\mathfrak{P}(q_1 s + d_1; \gcd(q_1, d_1) = 1) = \prod_{j=1}^{m} \frac{p_j^2 - 1}{p_j^2 - 2} \prod_{i=1}^{\infty} \frac{p_i^2 - 2}{p_i^2 - 1}$$

Proof. With Lemma 22 and the "Inclusion-Exclusion-Principle"we get the expected result. □

Proof. of Proposition 20
With Lemma 24 and Lemma 25 we get the expected result. □

5.2 $\mathfrak{P}((c_1 s + d_1); c_1, d_1 \in \mathbb{N}_+, d_1 > 1)$

Lemma 26. *Let* $d_1 = q_1$ *and* $q_1 = \prod_{j=1}^{m} p_j^{2\alpha_j}$ *then*

$$\mathfrak{P}(s + q_1) = \prod_{j=1}^{m} \frac{p_j^2 - 1}{p_j^2 - 2} \prod_{i=1}^{\infty} \frac{p_i^2 - 2}{p_i^2 - 1}$$

Proof. We consider only the distribution of the remainders of p_j^2. We start with

$$f(s) = \{0, 1, 1, \dots\}$$

Now we have

$$f(s + p_j^{2\alpha_j}) = \{0, 1, 1, \dots\}$$

i.e. the addition with $p_j^{\alpha_j}$ change nothing and we have no additional cancellation. Replacing the term $\frac{p_j^2-2}{p_j^2-1}$ with 1 and the "Inclusion-Exclusion-Principle"we get the expected result. □

Theorem 27. *Let* $c_1, d_1 \in \mathbb{N}$, $B_{2C} = \{p \in \mathbb{P} | (p^2 | c_1)\}$, $B_{1C} = \{p \in \mathbb{P} | (p \notin B_{2C}) \wedge (p | c_1)\}$, $B_{2D} = \{p \in \mathbb{P} | (p^2 | d_1)\}$, $B_{1D} = \{p \in \mathbb{P} | (p \notin B_{2D}) \wedge (p | d_1)\}$, *then*

$$\mathfrak{P}(c_1 s + d_1; c_1, d_1 \in \mathbb{N}) = \prod_{i=1}^{\infty} \frac{p_i^2 - 2}{p_i^2 - 1} \cdot \begin{cases} 0 & (p \in B_{2C}) \wedge (p \in B_{2D}) \\ \frac{p^2-1}{p^2-2} & (p \in B_{2C}) \wedge (p \notin B_{2D}) \\ \frac{p^2-p}{p^2-2} & (p \in B_{1C}) \wedge (p \in B_{2D}) \\ \frac{p^2-p-1}{p^2-2} & (p \in B_{1C}) \wedge (p \in B_{1D}) \\ \frac{p^2-1}{p^2-2} & (p \in B_{1C}) \wedge (d_1 = 1) \\ \frac{p^2-1}{p^2-2} & (c_1 = 1) \wedge (p \in B_{2D}) \\ 1 & \textit{otherwise} \end{cases}$$

Proof. We start with

$$f(s) = \{0, 1, 1, \dots\}$$

and consider all cases:
Case $(p \in B_{2C}) \wedge (p \in B_{2D})$:
We have $c_1 = p^2 c_1'$ and $d_1 = p^2 d_1'$ and therefore $p^2(c_1' s + d_1')$ which is not squarefree.
Case $(p \in B_{2C}) \wedge (p \notin B_{2D})$: See Lemma 22
Case $(p \in B_{1C}) \wedge (p \in B_{2D})$:
Since

$$f(ps) = \{p-1, p^2, \dots, p^2, 0, \dots\}$$

(p^2 $p-1$ times) and $f(p^2)$ do not change this we get

$$\left(1 - \frac{p-1}{p^2-1}\right) \cdot \frac{p^2-1}{p^2-2} = \frac{p^2-p}{p^2-2}$$

Case $(p \in B_{1C}) \wedge (p \in B_{1D})$: See Lemma 21
Case $(p \in B_{1C}) \wedge (d_1 = 1)$: See Lemma 22
Case $(c_1 = 1) \wedge (p \in B_{2D})$: See Lemma 26

□

5.3 Calculation of $\mathfrak{P}((c_1 n + d_1) \wedge \dots \wedge (c_m n + d_m); c_i, d_i \in \mathbb{N}, c_i > 0)$

```
; this hold a Bracket (i.e. (kFact * s + kCoeff))
; Programming language: PureBasic
Structure sBracket
  kFact.i
  kCoeff.i
EndStructure

; Programming language: PureBasic
Procedure.i cwCountBracket(Array aBracket.sBracket(1))
  ; Count the # of Bracket with Fact > 0
  ; IF Coeff = 0 => no additional Cancellation
  Protected kBrac.i,kp.i
  kBrac = 0
  For kp = 0 To ArraySize(aBracket())
    If (aBracket(kp)\kFact > 0) And (aBracket(kp)\kCoeff > 0)
      kBrac + 1
    EndIf
  Next
  ProcedureReturn kBrac
EndProcedure

; Programming language: PureBasic
Procedure cwCleanBracket(Array aBrackett.sBracket(1))
```

```
  ; Delete: multiple elements
  Protected kp.i,kcp.i,km.i
  SortStructuredArray(aBrackett(),#PB_Sort_Descending,
        OffsetOf(sBracket\kCoeff),TypeOf(sBracket\kCoeff))
  kp = 0
  ; Delete multiple Brackets
  While kp < ArraySize(aBrackett())
    If aBrackett(kp)\kFact > 0
      kcp = aBrackett(kp)\kCoeff
      km = kp + 1
      While (km <= ArraySize(aBrackett())) And
            (aBrackett(km)\kCoeff = kcp)
        If aBrackett(kp)\kFact = aBrackett(km)\kFact
          aBrackett(km)\kFact = 0
        EndIf
        km + 1
      Wend
    EndIf
    kp + 1
  Wend
  SortStructuredArray(aBrackett(),#PB_Sort_Descending,
        OffsetOf(sBracket\kFact),TypeOf(sBracket\kFact))
  kp = ArraySize(aBrackett())
  While (kp >= 0) And (aBrackett(kp)\kFact = 0)
    kp - 1
  Wend
  If kp < 1
    kp = 1
  EndIf
  ReDim aBrackett(kp) ; delete zero elements
EndProcedure

; Programming language: PureBasic
Procedure.i cwFillBracket(Array aBrackett.sBracket(1),tBracket.s)
  ; "f_1,c_1 # f_2,c_2 # ..." ->
  ; aBracket(0)\kFact = f_1, aBracket(0)\kCoeff = c_1,...
  Protected kBrak.i,kp.i,stBrak.s,km.i,kcp.i,kcm.i
  ; Calc #Brackets
  kBrak = CountString(tBracket,"#") + 1
  If kBrak = 1
    ReDim aBrackett(kBrak)
  Else
    ReDim aBrackett(kBrak - 1)
  EndIf
  ; Clear Array
```

```
  For kp = 0 To ArraySize(aBrackett())
    aBrackett(kp)\kCoeff = 0
    aBrackett(kp)\kFact = 0
  Next
  ; FillArray
  For kp = 1 To kBrak
    stBrak = StringField(tBracket,kp,"#")
    If CountString(stBrak,",") = 1
      aBrackett(kp-1)\kFact = Val(StringField(stBrak,1,","))
      aBrackett(kp-1)\kCoeff = Val(StringField(stBrak,2,","))
    Else
      Debug stBrak + " is not a bracket!"
      End ; Terminate the program
    EndIf
  Next
  ProcedureReturn kBrak
EndProcedure

; Programming language: PureBasic
Procedure.i cwCntCancels(Array aBracket.sBracket(1),tPrime.i)
  ; Count the Cancellations for one Prime
  Protected k2Prime.i,kBrak.i,kp.i,kMod.i,kCancels.i,kFree.i,kx.i
  k2Prime = tPrime * tPrime
  Dim aCancel.i(k2Prime - 1)
  kBrak = 0 ; inspect all brackets (Fact * s + Coeff)
  While kBrak <= ArraySize(aBracket())
    If aBracket(kBrak)\kFact > 0 ; only brackets with Fact > 0
      For kp = 1 To (k2Prime - 1)
        ; we count only (Fact * kp + Coeff) = 0 Mod k2Prime
        ; (s Mod p^2) = kp and kp > 0 (s is squarefree !)
        ; only kernel(Fact * kp + Coeff) is cancelled
        kMod = aBracket(kBrak)\kFact * kp + aBracket(kBrak)\kCoeff
        If (kMod % k2Prime) = 0
          ; we count each cancel. only once
          aCancel(kp) = 1
        EndIf
      Next
    EndIf
    kBrak + 1
  Wend
  ; now we count all cancellations
  kCancels = 0
  For kp = 1 To ArraySize(aCancel())
    kCancels + aCancel(kp)
  Next
```

```
  ProcedureReturn kCancels
EndProcedure

; Programming language: PureBasic
Procedure cwWedge(tBracket.s)
  ; Return the Wedge Calculation
  Protected kWedge.d,kPrime.i,k2Prime.i,kp.i,kMax.i,kCancels.i
  Protected kBrac.i,kIsMod.i
  *gSieve = xNewSieve(2) ; Init the squarefree Sieve
  Dim aBracket.sBracket(1)
  cwFillBracket(aBracket(),tBracket) ; Extract the Brackets
  cwCleanBracket(aBracket()) ; del. Bracket with Fact = 0
  kBrac = cwCountBracket(aBracket())
  ; kBrac = #Cancellation in allmost all cases
  kMax = 1 ; Calc upper bound for kPrime
  For kp = 0 To ArraySize(aBracket())
    If aBracket(kp)\kFact > kMax
      kMax = aBracket(kp)\kFact
    EndIf
    If aBracket(kp)\kCoeff > kMax
      kMax = aBracket(kp)\kCoeff
    EndIf
  Next
  kWedge = 1.0
  kPrime = 2 ; test only Primes <= kMax
  While kPrime <= kMax
    k2Prime = kPrime * kPrime
    kIsMod = 0
    kCancels = kBrac
    For kp = 0 To ArraySize(aBracket())
      If aBracket(kp)\kFact > 0
        ; kCancel <> kBrac only if ...
        If ((aBracket(kp)\kFact % kPrime) = 0) Or
                          ((aBracket(kp)\kCoeff % kPrime) = 0)
          kIsMod = 1
          Break
        EndIf
      EndIf
    Next
    If kIsMod = 1 ; Calc Cancellations explicit
      kCancels = cwCntCancels(aBracket(),kPrime)
      If kCancels >= (k2Prime - 1)
        kWedge = 0.0
        Break
      EndIf
```

```
    EndIf
    kWedge = kWedge * (k2Prime - 1.0 - kCancels) / (k2Prime - 1.0)
    kPrime = xSVNextPrime(*gSieve,kPrime)
  Wend
  ; Calc. up to a max Prime
  kBrac + 1
  While kPrime < 104800
    k2Prime = kPrime * kPrime
    kWedge = kWedge * (k2Prime - kBrac) / (k2Prime - 1.0)
    kPrime = xSVNextPrime(*gSieve,kPrime)
  Wend
EndProcedure
```

5.4 Calculation of $\mathfrak{P}((c_1 s + d_1); c_1, d_1 \in \mathbb{N}_+, d_1 > 1)$

```
; Programming language: PureBasic
Macro _Bound(tp,tm)
  Abs(aBracket(tp)\kFact * aBracket(tm)\kCoeff -
      aBracket(tm)\kFact * aBracket(tp)\kCoeff)
EndMacro

Macro _Value(tIndex,tRem)
  (aBracket(tIndex)\kFact * tRem + aBracket(tIndex)\kCoeff)
EndMacro

; Programming language: PureBasic
Procedure cwhSinkWedge(Array aBracket.sBracket(1),
          Array aCancel.i(1),t2Prime.i)
  ; mark the sink's through Wedge
  Protected kp.i,kBrak.i,kValue.i
  kBrak = 0
  ; a copy of the initial aCancel()
  ; run over aTmp and change aCancel() to
  ; procduce the Cancel's in the right position
  Dim aTmp.i(ArraySize(aCancel()))
  While kBrak <= ArraySize(aBracket())
    CopyArray(aCancel(),aTmp())
    kp = 1
    While kp < t2Prime
      If aTmp(kp) > 0 ; only 1 (i.e. up to now not cancelled)
        If (_Value(kBrak,kp) % t2Prime) = 0
          aCancel(kp) = 0
        EndIf
      EndIf
      kp + 1
```

```
    Wend
    kBrak + 1
  Wend
EndProcedure

; Programming language: PureBasic
Procedure cwhSinkGcd(Array aBracket.sBracket(1),
          Array aCancel.i(1),tp.i,tm.i,tPrime.i)
  ; mark all sink's through Gcd in respect to tPrime
  Protected ki.i,kr.i,kb.i
  ki = 0
  While ki < tPrime
    If (_Value(tp,ki) % tPrime) = 0
      While ki < ArraySize(aCancel())
        If (_Value(tm,ki) % tPrime) = 0
          aCancel(ki) = 0
        EndIf
        ki + tPrime
      Wend
      Break
    EndIf
    ki + 1
  Wend
EndProcedure

; Programming language: PureBasic
Procedure.i cwCntWithGcds(Array aBracket.sBracket(1),tPrime.i)
  Protected kp.i,km.i,k2Prime.i,kValue.i,kBrak.i,kCanc.i
  ; aCancel() Rem-Dist of s 0..threre are elments of s
  ; except aCancel(0) "the Rem Sink"
  k2Prime = tPrime * tPrime
  Dim aCancel.i(k2Prime - 1)
  FillMemory(aCancel(),8 * ArraySize(aCancel()) + 8,1,#PB_Integer)
  aCancel(0) = 0
  cwhSinkWedge(aBracket(),aCancel(),k2Prime)
  ; Test all possible Bracket-Pairs
  kp = 0
  While kp < ArraySize(aBracket())
    km = kp + 1
    While km <= ArraySize(aBracket())
      If tPrime <= _Bound(kp,km)
        cwhSinkGcd(aBracket(),aCancel(),kp,km,tPrime)
      EndIf
      km + 1
    Wend
```

```
    kp + 1
  Wend
  kCanc = 0
  For kp = 1 To ArraySize(aCancel())
    If aCancel(kp) = 0
      kCanc + 1
    EndIf
  Next
  ProcedureReturn kCanc
EndProcedure

; Programming language: PureBasic
Procedure.i cwUpperBound(Array aBracket.sBracket(1))
  ; Calc the upper bound Prime to test cwSinkGcd
  ; c1 * k + d1 = r1 * p AND c2 * k + d2 = r2 * p
  ; where 1 <= k < p^2
  ; k = (r1 * p - b1) / c1
  ; p = (c2 * b1 - c1 * b2) / (c2 * r1 - r2)
  ; we search the max p -> (c2 * r1 - r2) = +- 1
  Protected kMax.i,kp.i,km.i,kTmp.i
  ; but first (safety) we calc the upper bound for wedge
  ; easy only max Coeff
  kMax = 1
  For kp = 0 To ArraySize(aBracket())
    If aBracket(kp)\kCoeff > kMax
      kMax = aBracket(kp)\kCoeff
    EndIf
    If aBracket(kp)\kFact > kMax
      kMax = aBracket(kp)\kFact
    EndIf
  Next
  ; now Upper bound gcd
  kp = 0
  While kp < ArraySize(aBracket())
    km = kp + 1
    While km <= ArraySize(aBracket())
      kTmp = _Bound(kp,km)
      If kMax < kTmp
        kMax = kTmp
      EndIf
      km + 1
    Wend
    kp + 1
  Wend
  ProcedureReturn kMax
```

```
EndProcedure

; Programming language: PureBasic
Procedure.d cwCalc(tBracket.s,tExperimetal.b = #True,
          tWithGcd.b = #True)
  Protected kProd.d,kPrime.i,k2Prime.i,kp.i,kMax.i,kCancels.i,kBrac.i
  Protected kIsMod.i, kTime.i
  *gSieve = xNewSieve(2)
  Dim aBracket.sBracket(1)
  cwFillBracket(aBracket(),tBracket)
  If cwCleanBracket(aBracket()) = 1
    cwPrintBracket(aBracket())
    Debug StrD(0.0,#PG_Decimals) + " because multiple Brackets"
    End
  EndIf
  If cwHasMoreConformBraks(aBracket()) = 1
    Debug StrD(0.0,#PG_Decimals) +
          " because two or more conform brackets"
    End
  EndIf
  cwPrintBracket(aBracket())
  kBrac = cwCountBracket(aBracket(),#True)
  If kBrac < 2
    Debug "Too few Factors"
    End
  EndIf
  ; Calc upper bound for kPrime
  kMax = cwUpperBound(aBracket()); * 5
  kProd = 1.0
  kPrime = 2 ; test only Primes <= kMax
  While kPrime <= kMax
    k2Prime = kPrime * kPrime
    kIsMod = 1
    kCancels = kBrac
    If kIsMod = 1
      kCancels = cwCntWithGcds(aBracket(),kPrime)
      If kCancels >= (k2Prime - 1)
        kProd = 0.0
        Break
      EndIf
    EndIf
    kProd = kProd * (k2Prime - 1.0 - kCancels) / (k2Prime - 1.0)
    kPrime = xSVNextPrime(*gSieve,kPrime)
  Wend
  ;Calc. up to a max Prime
```

```
  kBrac + 1
  While kPrime < 104800
    k2Prime = kPrime * kPrime
    kProd = kProd * (k2Prime - kBrac) / (k2Prime - 1.0)
    kPrime = xSVNextPrime(*gSieve,kPrime)
  Wend
  ProcedureReturn kProd
EndProcedure
```

References

[PRE02] H. Preininger, *Distribution of the Residues and Cycle Counting*, 2017, http://vixra.org/pdf/1705.0289v1.pdf

[PRE04] H. Preininger, *Natural Squarefree Numbers: Statistical Properties*, 2017, http://vixra.org/pdf/1712.0441v1.pdf

[PRE05] H. Preininger, *Natural Squarefree Numbers: Statistical Properties II*, 2018, http://vixra.org/pdf/1801.0138v1.pdf