Concrete bounds for Chebyshev's prime counting functions

Manuel Eberl

September 16, 2024

Abstract

This entry derives explicit lower and upper bounds for Chebyshev's prime counting functions

$$\psi(x) = \sum_{\substack{p^k \leq x \\ k > 0}} \log p \qquad \vartheta(x) = \sum_{p \leq x} \log p \ .$$

Concretely, the following inequalities are proven:

- $\psi(x) \geq 0.9x$ for $x \geq 41$
- $\vartheta(x) \geq 0.82x$ if $x \geq 97$
- $\vartheta(x) \leq \psi(x) \leq 1.2x$ if $x \geq 0$

The proofs work by careful estimation of $\psi(x)$, with Stirling's formula as a starting point, to prove the bound for all $x \geq x_0$ with a concrete x_0, followed by brute-force approximation for all x below x_0.

An easy corollary of this is *Bertrand's postulate*, i.e. the fact that for any $x > 1$ the interval $(x, 2x)$ contains at least one prime (a fact that has already been shown in the AFP using weaker bounds for ψ and ϑ).

Contents

1 Concrete bounds for Chebyshev's prime counting functions

```
theory Chebyshev_Prime_Exhaust
imports
  "HOL-Decision_Procs.Approximation"
  "HOL-Library.Code_Target_Numeral"
  "Prime_Number_Theorem.Prime_Counting_Functions"
begin
```

The well-known Prime Number Theorem states that $\psi(x) \sim \theta(x) \sim (x)$, i.e. that both $\psi(x)$ and $\vartheta(x)$ are bounded by $(1 \pm \varepsilon)x$ for sufficiently large x for any $\varepsilon > 0$. However, these are asymptotic bounds without giving any concrete information on how ψ and ϑ behave for small x, or even how big x must be until these bound shold.

To complement this, we shall prove some concrete, non-asymptotic bounds. Concretely:

- $\psi(x) \geq 0.9x$ if $x \geq 41$
- $\theta(x) \geq 0.82x$ if $x \geq 97$
- $\theta(x) \leq \psi(x) \leq 1.2x$ if $x \geq 0$

Our formalisation loosely follows a blog entry by A.W. Walker: https://awwalker.com/2017/02/05/notes-on-the-chebyshev-theorem/

1.1 Brute-force checking of bounds for ψ and ϑ

1.1.1 Computing powers of a number

```
function powers_below_aux :: "nat ⇒ nat ⇒ nat ⇒ nat list" where
  "powers_below_aux ub n acc = (if acc = 0 ∨ n ≤ 1 ∨ acc > ub then []
else
     acc # powers_below_aux ub n (acc * n))"
  ⟨proof⟩
termination
  ⟨proof⟩

lemmas [simp del] = powers_below_aux.simps

lemma set_powers_below_aux:
  assumes "acc > 0" "n > 1"
  shows   "set (powers_below_aux ub n acc) = range (λi. acc * n ^ i)
∩ {..ub}"
  ⟨proof⟩

definition powers_below :: "nat ⇒ nat ⇒ nat list" where
```

```
  "powers_below ub n = powers_below_aux ub n n"

lemma set_powers_below:
  assumes "n > 1"
  shows   "set (powers_below ub n) = (λi. n ^ i) ` {1..} ∩ {..ub}"
⟨proof⟩

lemma distinct_powers_below_aux:
  assumes "n > 1" "acc > 0"
  shows   "distinct (powers_below_aux ub n acc)"
  ⟨proof⟩

lemma distinct_powers_below: "n > 1 ⟹ distinct (powers_below ub n)"
  ⟨proof⟩

lemma hd_powers_below_aux:
  assumes "acc ≤ ub" "n > 1" "acc > 0"
  shows   "hd (powers_below_aux ub n acc) = acc"
  ⟨proof⟩

lemma hd_powers_below:
  assumes "n ≤ ub" "n > 1"
  shows   "hd (powers_below ub n) = n"
  ⟨proof⟩
```

1.1.2 Computing prime powers

```
definition prime_powers_upto :: "nat ⇒ (nat × nat) list" where
  "prime_powers_upto n =
     sort_key fst (concat (map (λp. map (λk. (k, p)) (powers_below n p))
(primes_upto n)))"

lemma map_key_sort_key: "map f (sort_key f xs) = sort (map f xs)"
⟨proof⟩

lemma distinct_prime_powers_upto:
  "distinct (map fst (prime_powers_upto n))"
⟨proof⟩

lemma sorted_prime_powers_upto:
  "sorted (map fst (prime_powers_upto n))"
  ⟨proof⟩

lemma set_prime_powers_upto:
  "set (prime_powers_upto n) = {(q, aprimedivisor q) |q. primepow q ∧
q ≤ n}"
⟨proof⟩
```

1.1.3 A generic checking function

```
locale chebyshev_check =
  fixes f :: "nat ⇒ real"
    and F :: "nat ⇒ 'a ⇒ float"
    and A :: "nat set"
    and plus :: "nat ⇒ float ⇒ float ⇒ float"
    and rel :: "real ⇒ real ⇒ bool"
    and P :: "nat ⇒ real ⇒ bool"
    and num :: "'a ⇒ nat"
  assumes plus: "⋀prec. rel X x ⟹ rel Y y ⟹ rel (plus prec X Y)
(x + y)"
  assumes P_rel: "⋀x y k. P k x ⟹ rel x y ⟹ P k y"
  assumes rel_0: "rel 0 0"
  assumes A: "0 ∉ A"
begin

definition S where "S n = (∑k∈A∩{..n}. f k)"
definition S' where "S' n = (∑k∈A∩{..<n}. f k)"

context
  fixes prec :: nat
begin

function check_aux :: "'a list ⇒ nat ⇒ nat ⇒ float ⇒ nat ⇒ bool"
where
  "check_aux ps lb ub acc n = (if n > ub then True else
     (let (acc', ps') =
        (if ps ≠ [] ∧ num (hd ps) = n then
           (plus prec acc (F prec (hd ps)), tl ps)
         else (acc, ps))
      in (n < lb ∨ P n (real_of_float acc')) ∧ check_aux ps' lb ub acc'
(n+1)))"
  ⟨proof⟩
termination
  ⟨proof⟩

definition check :: "'a list ⇒ nat ⇒ nat ⇒ bool" where
  "check xs lb ub =
     check_aux xs lb ub 0 (if xs = [] then lb else min lb (num (hd xs)))"

lemmas [simp del] = check_aux.simps

lemma check_aux_correct:
  assumes "sorted (map num ps)" "distinct (map num ps)"
  assumes "⋀p. p ≤ ub ⟹ p ∈ num ` set ps ⟷ p ∈ A ∧ p ≥ n"
  assumes "⋀x. x ∈ set ps ⟹ rel (real_of_float (F prec x)) (f (num
x))"
  assumes "rel (real_of_float acc) (S' n)"
  assumes "check_aux ps lb ub acc n"
```

```
  assumes "k ∈ {max lb n..ub}"
  shows   "P k (S k)"
  ⟨proof⟩

lemma check_correct:
  assumes "sorted (map num ps)" "distinct (map num ps)"
  assumes "⋀p. p ≤ ub ⟹ p ∈ num ‘ set ps ⟷ p ∈ A"
  assumes "⋀x. x ∈ set ps ⟹ rel (real_of_float (F prec x)) (f (num
x))"
  assumes "check ps lb ub"
  assumes "k ∈ {lb..ub}"
  shows   "P k (S k)"
⟨proof⟩

end

end
```

1.1.4 The ϑ function

```
context
begin

interpretation primes_theta: chebyshev_check
  "λn. ln (real n)"
  "λprec n. the (lb_ln prec (Float (int n) 0))"
  "{p. prime p}"
  "float_plus_down"
  "(≤)"
  "λk x. x ≥ c * (real k + 1)"
  "λn. n"
  for c :: real
⟨proof⟩


definition check_theta_lower_aux
  where "check_theta_lower_aux = primes_theta.check_aux"

definition check_theta_lower where
  "check_theta_lower c prec lb ub =
     primes_theta.check c prec (primes_upto ub) lb ub"

lemma check_theta_lower_aux_code [code]:
  "check_theta_lower_aux c prec ps lb ub acc n =
     (if ub < n then True else let (acc’, ps’) =
            if ps ≠ [] ∧ hd ps = n
            then (float_plus_down prec acc (the (lb_ln prec (Float (int
(hd ps)) 0))), tl ps)
            else (acc, ps)
```

```
      in (n < lb ∨ c * (real n + 1) ≤ real_of_float acc') ∧
         check_theta_lower_aux c prec ps' lb
          ub acc' (n + 1))"
  ⟨proof⟩

lemma check_theta_lower_code [code]:
  "check_theta_lower c prec lb ub = (let ps = primes_upto ub in
     check_theta_lower_aux c prec ps lb ub 0
       (if ps = [] then lb else min lb (hd ps)))"
  ⟨proof⟩

lemma check_theta_lower_correct:
  assumes "check_theta_lower c prec lb ub"
  shows   "∀x∈{real lb..real ub}. primes_theta x ≥ c * x"
⟨proof⟩

end




context
begin

interpretation primes_theta: chebyshev_check
  "λn. ln (real n)"
  "λprec n. the (ub_ln prec (Float (int n) 0))"
  "{p. prime p}"
  "float_plus_up"
  "(≥)"
  "λk x. x ≤ c * real k"
  "λn. n"
  for c :: real
⟨proof⟩


definition check_theta_upper_aux
  where "check_theta_upper_aux = primes_theta.check_aux"

definition check_theta_upper where
  "check_theta_upper c prec lb ub =
     primes_theta.check c prec (primes_upto ub) lb ub"

lemma check_theta_upper_aux_code [code]:
  "check_theta_upper_aux c prec ps lb ub acc n =
     (if ub < n then True else let (acc', ps') =
            if ps ≠ [] ∧ hd ps = n
            then (float_plus_up prec acc (the (ub_ln prec (Float (int
(hd ps)) 0))), tl ps)
```

```
          else (acc, ps)
      in (n < lb ∨ c * real n ≥ real_of_float acc') ∧
         check_theta_upper_aux c prec ps' lb
          ub acc' (n + 1))"
  ⟨proof⟩

lemma check_theta_upper_code [code]:
  "check_theta_upper c prec lb ub = (let ps = primes_upto ub in
     check_theta_upper_aux c prec ps lb ub 0
       (if ps = [] then lb else min lb (hd ps)))"
  ⟨proof⟩

lemma check_theta_upper_correct:
  assumes "check_theta_upper c prec lb ub" "c ≥ 0"
  shows   "∀x∈{real lb..real ub}. primes_theta x ≤ c * x"
⟨proof⟩

end
```

1.1.5 The ψ function

```
context
begin

interpretation primes_psi: chebyshev_check
  "λn. ln (real (aprimedivisor n))"
  "λprec x. the (lb_ln prec (Float (int (snd x)) 0))"
  "{p. primepow p}"
  "float_plus_down"
  "(≤)"
  "λk x. x ≥ c * (real k + 1)"
  "fst"
  for c :: real
⟨proof⟩


definition check_psi_lower_aux
  where "check_psi_lower_aux = primes_psi.check_aux"

definition check_psi_lower where
  "check_psi_lower c prec lb ub =
     primes_psi.check c prec (prime_powers_upto ub) lb ub"

lemma check_psi_lower_aux_code [code]:
  "check_psi_lower_aux c prec ps lb ub acc n =
     (if ub < n then True else let (acc', ps') =
            if ps ≠ [] ∧ fst (hd ps) = n
            then (float_plus_down prec acc (the (lb_ln prec (Float (int
(snd (hd ps))) 0))), tl ps)
```

```
            else (acc, ps)
      in (n < lb ∨ c * (real n + 1) ≤ real_of_float acc') ∧
         check_psi_lower_aux c prec ps' lb
          ub acc' (n + 1))"
  ⟨proof⟩

lemma check_psi_lower_code [code]:
  "check_psi_lower c prec lb ub = (let ps = prime_powers_upto ub in
     check_psi_lower_aux c prec ps lb ub 0
       (if ps = [] then lb else min lb (fst (hd ps))))"
  ⟨proof⟩

lemma check_psi_lower_correct:
  assumes "check_psi_lower c prec lb ub"
  shows   "∀x∈{real lb..real ub}. primes_psi x ≥ c * x"
⟨proof⟩

end


context
begin

interpretation primes_psi: chebyshev_check
  "λn. ln (real (aprimedivisor n))"
  "λprec x. the (ub_ln prec (Float (int (snd x)) 0))"
  "{p. primepow p}"
  "float_plus_up"
  "(≥)"
  "λk x. x ≤ c * real k"
  "fst"
  for c :: real
⟨proof⟩


definition check_psi_upper_aux
  where "check_psi_upper_aux = primes_psi.check_aux"

definition check_psi_upper where
  "check_psi_upper c prec lb ub =
     primes_psi.check c prec (prime_powers_upto ub) lb ub"

lemma check_psi_upper_aux_code [code]:
  "check_psi_upper_aux c prec ps lb ub acc n =
     (if ub < n then True else let (acc', ps') =
            if ps ≠ [] ∧ fst (hd ps) = n
            then (float_plus_up prec acc (the (ub_ln prec (Float (int
(snd (hd ps))) 0))), tl ps)
            else (acc, ps)
```

```
      in (n < lb ∨ c * real n ≥ real_of_float acc') ∧
         check_psi_upper_aux c prec ps' lb
          ub acc' (n + 1))"
  ⟨proof⟩
```

lemma *check_psi_upper_code [code]:*

```
  "check_psi_upper c prec lb ub = (let ps = prime_powers_upto ub in
     check_psi_upper_aux c prec ps lb ub 0
       (if ps = [] then lb else min lb (fst (hd ps))))"
  ⟨proof⟩
```

lemma *check_psi_upper_correct:*

```
  assumes "check_psi_upper c prec lb ub" "c ≥ 0"
  shows   "∀x∈{real lb..real ub}. primes_psi x ≤ c * x"
⟨proof⟩
```

end

end

theory *Chebyshev_Prime_Bounds*

imports

```
  "Prime_Number_Theorem.Prime_Counting_Functions"
  "Prime_Distribution_Elementary.Prime_Distribution_Elementary_Library"
  "Prime_Distribution_Elementary.Primorial"
  "HOL-Decision_Procs.Approximation"
  "HOL-Library.Code_Target_Numeral"
  Chebyshev_Prime_Exhaust
```

begin

1.2 Auxiliary material

context *comm_monoid_set*

begin

lemma *union_disjoint':*

```
  assumes "finite C" "A ∪ B = C" "A ∩ B = {}"
  shows   "f (F g A) (F g B) = F g C"
  ⟨proof⟩
```

end

lemma *sum_mset_nonneg:*

```
  fixes X :: "'a :: ordered_comm_monoid_add multiset"
  shows "(⋀x. x ∈# X ⟹ x ≥ 0) ⟹ sum_mset X ≥ 0"
  ⟨proof⟩
```

lemma *of_int_sum_mset: "of_int (sum_mset M) = sum_mset (image_mset of_int M)"*

```
  ⟨proof⟩
```

lemma *sum_sum_mset:* "(∑x∈A. ∑y∈#B. f x y) = (∑y∈#B. ∑x∈A. f x y)"
⟨*proof*⟩

lemma *sum_mset_diff_distrib:*
fixes *f g :: "'a ⇒ 'b :: ab_group_add"*
shows "(∑x∈#A. f x - g x) = (∑x∈#A. f x) - (∑x∈#A. g x)"
⟨*proof*⟩

lemma *sum_mset_neg_distrib:*
fixes *f :: "'a ⇒ 'b :: ab_group_add"*
shows "(∑x∈#A. -f x) = -(∑x∈#A. f x)"
⟨*proof*⟩

1.3 Bounds for the remainder in Stirling's approximation

definition *ln_fact_remainder :: "real ⇒ real"* **where**
"ln_fact_remainder x = ln (fact (nat ⌊x⌋)) - (x * ln x - x)"

lemma *ln_fact_remainder_bounds:*
assumes *x: "x ≥ 3"*
shows "ln_fact_remainder x ≤ ln x / 2 + ln (2 * pi) / 2 + 1 / (12 * ⌊x⌋)"
and "ln_fact_remainder x ≥ -ln x / 2 + ln (2 * pi) / 2 - 1 / (2 * x)"
⟨*proof*⟩

lemma *abs_ln_fact_remainder_bounds:*
assumes *x: "x ≥ 3"*
shows "|ln_fact_remainder x| < ln x / 2 + 1"
⟨*proof*⟩

1.4 Approximating ψ

unbundle *prime_counting_notation*

lemma *primes_psi_lower_rec:*
fixes *f :: "real ⇒ real"*
assumes "⋀x. x ≥ x0 ⟹ f x ≤ f (x / c) + h x"
assumes "x0 > 0" "x * c ≥ x0 * c ^ n" "c ≥ 1"
shows "f x ≤ f (x / c ^ n) + (∑k<n. h (x / c ^ k))"
⟨*proof*⟩

locale *chebyshev_multiset =*
fixes *L :: "int multiset"*
assumes *L_nonzero: "0 ∉# L"*
begin

```
definition chi_L :: "real ⇒ int" ("χL")
  where "chi_L t = (∑l∈#L. sgn l * ⌊t / |l|⌋)"

definition psi_L :: "real ⇒ real" ("ψL")
  where "psi_L x = sum_upto (λd. mangoldt d * chi_L (x / d)) x"

definition alpha_L :: real ("αL")
  where "alpha_L = -(∑l∈#L. ln |l| / l)"

definition period :: nat
  where "period = nat (Lcm (set_mset L))"

lemma period_pos: "period > 0"
⟨proof⟩

lemma dvd_period: "l ∈# L ⟹ l dvd period"
  ⟨proof⟩

lemma chi_L_decompose:
  "χL (x + of_int (m * int period)) = χL x + m * int period * (∑l∈#L.
1 / l)"
⟨proof⟩

lemma chi_L_floor: "chi_L (floor x) = chi_L x"
  ⟨proof⟩

end


locale balanced_chebyshev_multiset = chebyshev_multiset +
  assumes balanced: "(∑l∈#L. 1 / l) = 0"
begin

lemma chi_L_mod: "χL (of_int (a mod int period)) = χL (of_int a)"
⟨proof⟩

sublocale chi: periodic_fun_simple chi_L "of_int period"
⟨proof⟩


definition psi_L_remainder where
  "psi_L_remainder x = (∑l∈#L. sgn l * ln_fact_remainder (x / |l|))"

lemma abs_sum_mset_le:
  fixes f :: "'a ⇒ 'b :: ordered_ab_group_add_abs"
  shows "|∑x∈#A. f x| ≤ (∑x∈#A. |f x|)"
  ⟨proof⟩

lemma psi_L_remainder_bounds:
```

```
  fixes x :: real
  assumes x: "x ≥ 3" "⋀l. l ∈# L ⟹ x ≥ 3 * ¦l¦"
  shows "¦psi_L_remainder x¦ ≤
           ln x * size L / 2 - 1/2 * (∑l∈#L. ln ¦l¦) + size L"
⟨proof⟩

lemma psi_L_eq:
  assumes "x > 0"
  shows   "psi_L x = α_L * x + psi_L_remainder x"
⟨proof⟩

lemma primes_psi_lower_bound:
  fixes x C :: real
  defines "x0 ≡ Max (insert 3 ((λl. 3 * ¦l¦) ` set_mset L))"
  assumes x: "x ≥ x0"
  assumes chi_le1: "⋀n. n ∈ {0..<period} ⟹ χ_L (real n) ≤ 1"
  defines "C ≡ 1 / 2 * (∑l∈#L. ln ¦l¦) - size L"
  shows   "ψ x ≥ α_L * x - ln x * size L / 2 + C"
⟨proof⟩

end

lemma psi_lower_bound_precise:
  assumes x: "x ≥ 90"
  shows   "ψ x ≥ 0.92128 * x - 2.5 * ln x - 1.6"
⟨proof⟩

context balanced_chebyshev_multiset
begin

lemma psi_upper_bound:
  fixes x c C :: real
  defines "x0 ≡ Max ({3, 55 * c} ∪ {3 * ¦l¦ |l. l ∈# L})"
  assumes x: "x ≥ x0"
  assumes chi_nonneg: "⋀n. n ∈ {0..<period} ⟹ χ_L (real n) ≥ 0"
  assumes chi_ge1: "⋀n. real n ∈ {1..<c} ⟹ χ_L (real n) ≥ 1"
  assumes c: "c > 1" "c ≤ period"
  assumes "α_L ≥ 0"
  shows "ψ x ≤ c / (c - 1) * α_L * x + (3 * size L) / (4 * ln c) * ln
x ^ 2 + ψ x0"
⟨proof⟩

end
```

1.5 Final results

```
theorem psi_lower_ge_9:
  assumes x: "x ≥ 41"
  shows   "ψ x ≥ 0.9 * x"
```

⟨*proof*⟩

theorem *primes_theta_ge_82:*
  assumes *"x ≥ 97"*
  shows *"ϑ x ≥ 0.82 * x"*
⟨*proof*⟩

corollary *primorial_ge_exp_82:*
  assumes *"x ≥ 97"*
  shows *"primorial x ≥ exp (0.82 * x)"*
⟨*proof*⟩

theorem *primes_psi_le_111:*
  assumes *"x ≥ 0"*
  shows *"ψ x ≤ 1.11 * x"*
⟨*proof*⟩

corollary *primes_theta_le_111:*
  assumes *"x ≥ 0"*
  shows *"ϑ x ≤ 1.11 * x"*
  ⟨*proof*⟩

As an easy corollary, we obtain Bertrand's postulate: For any real number $x > 1$, the interval $(x, 2x)$ contains at least one prime.

corollary *bertrands_postulate:*
  assumes *"x > 1"*
  shows *"∃p. prime p ∧ real p ∈ {x<..<2*x}"*
⟨*proof*⟩

unbundle *no_prime_counting_notation*

end