Journal of Convex Analysis
Volume 16 (2009), No. 3&4, 779–790

Numerical Computation of Fitzpatrick Functions

Bryan Gardiner
*Computer Science, I. K. Barber School of Arts & Sciences Unit # 4,
University of British Columbia Okanagan, 3333 University Way,
Kelowna BC V1V 1V7, Canada
khumba@interchange.ubc.ca*

Yves Lucet[*]
*Computer Science, I. K. Barber School of Arts & Sciences Unit # 4,
University of British Columbia Okanagan, 3333 University Way,
Kelowna BC V1V 1V7, Canada
yves.lucet@ubc.ca, http://people.ok.ubc.ca/ylucet/*

Dedicated to Stephen Simons on the occasion of his 70th birthday.

Received: May 15, 2008
Revised manuscript received: December 21, 2008

Fitzpatrick functions provide insights into the structure of operators. To help understand their information, we investigate their efficient numerical computation on a grid for operators with finite graphs defined on the real line. Our algorithms take advantage of existing computational Convex Analysis frameworks to improve previous worst-case time complexity results from quartic to quadratic. We also provide a linear-time algorithm for the computation of antiderivatives based on the Fitzpatrick function of infinite order.

2000 Mathematics Subject Classification: 52B55, 65D99

Introduction

Fitzpatrick functions were originally introduced in [12] to study the representation of operators using convex functions. They have generated a lot of interest very recently not only in studying such representations [16, 9, 10] but also in simplifying proofs by using convex analysis techniques [3, 4, 6, 17, 20, 21, 22, 24]. The Fitzpatrick function of infinite order is linked to cyclic monotonicity, and n-cyclic monotonicity is characterized very simply using the Fitzpatrick functions of order n, which have also proven very useful in the study of n-cyclically monotone operators [2, 7]. A recent application of Fitzpatrick functions, including its link to Rockafellar functions [19, 18], explained how to compute antiderivatives of cyclically monotone operators which preserve the symmetry induced by convex duality [5].

The present paper focuses on the efficient numerical computation *on a grid* of Fitzpatrick functions of two variables $F_{A,n} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ for $n = 2, \dots, +\infty$ with particular interest for the Fitzpatrick function of infinite order since it is an intrinsic antiderivative. We use fast numerical algorithms developed for computing the Fenchel conjugate [14]. The data

[*]Corresponding author. This work was partly supported by the second author NSERC Discovery grant.

ISSN 0944-6532 / $ 2.50 © Heldermann Verlag

structure to store piecewise linear functions and to evaluate them efficiently on a grid was introduced in [15] for piecewise linear-quadratic (PLQ) functions. Finally, we prove an explicit formula for the Fitzpatrick function of infinite order of two real variables with the second variable set to 0, which is similar to the explicit formula for Rockafellar functions of one variable as presented in [13, p. 145] and independently in [5, Theorem 3.14].

The paper is organized as follows: we first recall preliminary results in Section 1, give algorithms to compute Fitzpatrick functions in Section 2, and prove an explicit formula for a particular case of the Fitzpatrick function of infinite order in Section 3. Section 4 concludes the paper.

1. Preliminary

We recall definitions and previous results while setting our notations. A finite operator $A : \mathbb{R}^d \rightrightarrows \mathbb{R}^d$ has graph $\operatorname{gph} A = \{(\alpha_1, \alpha_1^*), \dots (\alpha_m, \alpha_m^*)\}$, range $\operatorname{Ran} A = \{a^* \mid \exists a, (a, a^*) \in \operatorname{gph} A\}$, and domain $\operatorname{Dom} A = \{a \mid \exists a^*, (a, a^*) \in \operatorname{gph} A\}$. Their respective sizes are denoted $m = |\operatorname{gph} A|$, $m_R = |\operatorname{Ran} A|$, and $m_D = |\operatorname{Dom} A|$. We always have $m_R \le m$ and $m_D \le m$. Except in Section 3 the letters $a, a_1, \dots, a_n$ (resp. $a^*, a_1^*, \dots, a_n^*$) indicate variables taking values in $\operatorname{Dom} A$ (resp. $\operatorname{Ran} A$). We denote $\langle \cdot, \cdot \rangle$ the standard dot product, and use it to emphasize results that extend beyond one dimension otherwise we revert to standard multiplication.

We recall the definition of n-cyclic monotonicity which is at the heart of Fitzpatrick functions.

Definition 1.1 ([1, 2, 7, 8, 23]). An operator A is *n-cyclically monotone* if the implication

$$\left.\begin{array}{c}(a_1, a_1^*) \in \operatorname{gph} A,\\ \vdots \\ (a_n, a_n^*) \in \operatorname{gph} A \\ a_{n+1} := a_1\end{array}\right\} \quad \Rightarrow \quad \sum_{i=1}^{n} \langle a_{i+1} - a_i, a_i^* \rangle \le 0 \tag{1}$$

holds.

The usual monotonicity notion corresponds to 2-monotonicity. Fitzpatrick functions are defined as follows.

Definition 1.2 ([2, Proposition 2.3]). Let $A : \mathbb{R}^d \rightrightarrows \mathbb{R}^d$ be an operator, and let $n \in \{2, 3, \dots\}$. Then $F_{A,n} \colon \mathbb{R}^d \times \mathbb{R}^d \to [-\infty, +\infty]$ is defined by

$$F_{A,n}(x, x^*) = \sup_{\substack{(a_1,a_1^*)\in \operatorname{gph} A,\\ \vdots \\ (a_{n-1},a_{n-1}^*)\in \operatorname{gph} A}} \left(\sum_{i=1}^{n-2} \langle a_{i+1} - a_i, a_i^* \rangle + \langle x - a_{n-1}, a_{n-1}^* \rangle + \langle a_1, x^* \rangle \right), \tag{2}$$

and $F_{A,\infty} = \sup_{n\ge 2} F_{A,n}$

One of the useful properties of Fitzpatrick functions is their relation to cyclic monotonicity.

Fact 1.3 ([5, Fact 2.11]). *For an operator A and $2 \le n \le +\infty$,*

$$A \text{ is } n\text{-cyclically monotone} \Leftrightarrow F_{A,n} = \langle\cdot,\cdot\rangle \text{ on } \operatorname{gph} A.$$

In the formulation above, ∞-cyclic monotonicity corresponds to cyclic monotonicity.

For operators defined on $\mathbb{R}$, n-cyclic monotonicity reduces to monotonicity.

Proposition 1.4. *Assume $A : \mathbb{R} \rightrightarrows \mathbb{R}$ and n is an integer with $n \ge 2$. Then the following are equivalent.*

(*i*) *A is n-cyclically monotone.*
(*ii*) *A is monotone.*
(*iii*) *A is cyclically monotone.*
(*iv*) $\operatorname{gph} A$ *is a non-decreasing curve in the plane with respect to the coordinate-wise partial ordering.*

Proof. The equivalences (*ii*)–(*iv*) are stated in [18, p. 240]. The fact they imply A is n-cyclically monotone for any $n \ge 2$ comes from Fact 1.3 which gives

$$\forall (x,x^*) \in \operatorname{gph} A, \;\; \langle x,x^*\rangle = F_{A,2}(x,x^*) \le F_{A,3}(x,x^*) \le \cdots \le F_{A,\infty}(x,x^*) = \langle x,x^*\rangle,$$

since the sequence $(F_{A,n})_{n\ge 2}$ is increasing and pointwise convergent to $F_{A,\infty}$ (see [5, p. 7]). Hence for any $n \ge 2$, $F_{A,n} = \langle\cdot,\cdot\rangle$ on $\operatorname{gph} A$, so A is n-cyclically monotone. Conversely, if A is n-cyclically monotone, Fact 1.3 implies it is monotone. □

Cyclically monotone finite operators have only a finite number of distinct Fitzpatrick functions.

Fact 1.5 ([5, Theorem 2.16]). *Suppose that $A : \mathbb{R} \rightrightarrows \mathbb{R}$ is finite with $\operatorname{gph} A$ containing only n points. Then if A is n-cyclically monotone, it is $(n+1)$-cyclically monotone with*

$$F_{A,n+1} = F_{A,n+2} = \cdots = F_{A,\infty}.$$

So when A is finite cyclically monotone, the only Fitzpatrick functions $F_{A,n}$ to compute are for $2 \le n \le m$, and $n = +\infty$. However, when A is not cyclically monotone, the function $F_{A,\infty}$ is always equal to ∞ ([5, Remark 3.14]) but all the other functions are (potentially) distinct.

We are now ready to tackle the numerical computation of Fitzpatrick functions.

2. Numerical Computation

Our goal is to compute $F_{A,n}(x,x^*)$ for (x,x^*) belonging to a grid $X = X_x \times X_{x^*}$ of size $N = N_x \times N_{x^*}$, and $2 \le n \le +\infty$ when $A : \mathbb{R} \rightrightarrows \mathbb{R}$ is a finite operator with a graph of size m. Using the definition, computing $F_{A,n}$ on a grid of size N takes $O(nm^{n-1}N)$ worst-case time. Indeed, each of the $n-1$ variables takes m values giving m^{n-1} possibilities for which a sum of $n-2$ terms is computed, and the calculation has to be done for each (x,x^*) belonging to the grid.

The exponential complexity is reduced to polynomial time using the following recursive formula.

Fact 2.1 ([5, Proposition 2.13]). *Let $A : \mathbb{R} \rightrightarrows \mathbb{R}$, $n \in \{2, 3, \ldots\}$, and $(x, x^*) \in \mathbb{R}^2$. Then*

$$F_{A,n+1}(x, x^*) = \sup_{(a,a^*)\in \operatorname{gph} A} \left[F_{A,n}(a, x^*) + xa^* - aa^*\right]. \tag{3}$$

Remark 2.2. The recursion Formula (3) can be extended to $n=1$ by defining $F_{A,1}(x, x^*) = xx^*$.

Proposition 2.3. *Using the recursive formula* (3)*, computing $F_{A,n}$ on a grid of size N takes $O(nm^2N_{x^*} + mN)$ time.*

Proof. Note $T_n(N)$ the time required to evaluate $F_{A,n}(x, x^*)$ on a grid of size N. Formula (3) implies that $T_n(N) = T_{n-1}(mN_{x^*}) + mN$, and $T_{n-1}(mN_{x^*}) = T_{n-2}(mN_{x^*}) + m^2N_{x^*}$. Solving the recurrence equation gives $T_{n-1}(mN_{x^*}) = O(nm^2N_{x^*})$. Hence, $T_n(N) = O(nm^2N_{x^*} + mN)$. □

Remark 2.4. The typical case occurs when $O(N_x) = O(N_{x^*}) = O(m)$. Indeed, $\operatorname{gph} A$ is a one-dimensional object (a curve in the plane) while N is the cardinality of a grid in the plane (a two-dimensional object). So a good measure of the complexity is when all the one-dimensional objects are of size $\Theta(m)$.

For example, a typical plot requires $m = N_x = N_{x^*} = 100$. In that case, the recursive Formula (3) allows the computation of $F_{A,n}$ on a grid of size $N = N_x.N_{x^*} = 10,000$ in $1,000,000(n+1)$ operations. Computing $F_{A,\infty} = F_{a,m+1}$ requires $102,000,000$ operations.

We now explain how to further improve the complexity using fast transform algorithms such as the Linear-time Legendre Transform (LLT) [14] (alternatively one may use the Parabolic Envelope (PE) [11] algorithm which shares the same complexity). First, we establish a different recursion formula which clearly separates the variables (x, x^*) thus allowing us to take advantage of the LLT algorithm.

Proposition 2.5. *Let $A : \mathbb{R} \rightrightarrows \mathbb{R}$, let $n \in \{4, 5, \ldots\}$, and let $(x, x^*) \in \mathbb{R}^2$. Then*

$$\begin{aligned} F_{A,n}(x, x^*) = \max_{a_1, a^*_{n-1}} \big(& a_1x^* + xa^*_{n-1} \\ &+ \max_{a^*_1, a_{n-1}} \left[-a_{n-1}a^*_{n-1} - a_1a^*_1 - I(a_1, a^*_1) - I(a_{n-1}, a^*_{n-1}) + F_{A,n-2}(a_{n-1}, a^*_1)\right] \big) \end{aligned} \tag{4}$$

where $I(a, a^*)$ is the indicator function of $\operatorname{gph} A$ *at (a, a^*) ($I(a, a^*) = 0$ when $(a, a^*) \in$* $\operatorname{gph} A$, *$+\infty$ otherwise).*

Proof. Changing the names of variables and applying (2) gives

$$F_{A,n-2}(a_{n-1}, a^*_1) = \max_{\substack{(a_2,a^*_2)\in \operatorname{gph} A, \\ \vdots \\ (a_{n-2},a^*_{n-2})\in \operatorname{gph} A}} \left[a_2a^*_1 + \sum_{i=2}^{n-2}(a_{i+1} - a_i)a^*_i\right].$$

The result follows by applying Formula (2) to $F_{A,n}$. □

We now recall the complexity of the LLT Algorithm for bivariate functions.

Fact 2.6 ([14, p. 177]). *Assume $u : \mathbb{R}^2 \to \mathbb{R}$ is a bivariate function, $S = S_1 \times S_2 \subset \mathbb{R}^2$ (resp. $X = X_1 \times X_2$) a grid with cardinality $|S_i| = m_i$ (resp. $|X_i| = n_i$). The LLT2d algorithm computes*

$$\begin{aligned} u^*_X(s_1, s_2) &= \max_{x_1 \in X_1, x_2 \in X_2} [s_1 x_1 + s_2 x_2 - u(x_1, x_2)] \\ &= \max_{x_1 \in X_1} \left[s_1 x_1 + \left(\max_{x_2 \in X_2} s_2 x_2 - u(x_1, x_2) \right) \right] \end{aligned}$$

for all values $(s_1, s_2) \in S$ in $O(n_1 n_2 + n_1 m_2 + m_1 m_2)$.

A first consequence of Fact 2.6 is that

$$F_{A,2}(x, x^*) = \max_{a,a^*} [x a^* + a x^* - a a^* - I(a, a^*)]$$

can be evaluated for all $(x, x^*) \in X$ using the LLT2d algorithm in $O(m^2 + m N_{x^*} + N)$ time. When $N = m^2$ this gives $O(m^2)$ which is also the size of the output and so is optimal.

Similarly we can compute

$$\begin{aligned} F_{A,3}(x, x^*) &= \max_{a_1, a_2^*} [x a_2^* + a_1 x^* + f(a_1, a_2^*)], \\ f(a_1, a_2^*) &= \max_{a_1^*, a_2} [-a_1 a_1^* - a_2 a_2^* + a_2 a_1^* - I(a_1, a_1^*) - I(a_2, a_2^*)] \end{aligned}$$

in $O(m^2 + m N_{x^*} + N)$ time using Algorithm 2.9 to compute f for all the m^2 values, and the LLT2d Algorithm.

A second consequence is that the Fitzpatrick functions can be computed recursively using Algorithm 2.7.

Algorithm 2.7 (Fitzpatrick functions Algorithm).

```
    Input: gph $A$, $X$, $n$
    Output: $F_{A,n}(x, x^*)$ for all $(x, x^*) \in X$
1   begin
2       if n=2 or n=3 then
3           Compute $F_{A,n}$ directly
4       else
6           Compute recursively $F_{A,n-2}(a_{n-1}, a_1^*)$ for all $(a_{n-1}, a_1^*) \in \operatorname{Dom} A \times \operatorname{Ran} A$;
8           Compute for all values $(a_1, a_{n-1}^*)$ the function
                $f(a_1, a_{n-1}^*) = \max_{a_1^*, a_{n-1}} \big[ -a_{n-1} a_{n-1}^* - a_1 a_1^* - I(a_1, a_1^*) - I(a_{n-1}, a_{n-1}^*) + F_{A,n-2}(a_{n-1}, a_1^*) \big]$
10          Compute for all values $(x, x^*)$
                $F_{A,n}(x, x^*) = \max_{a_1, a_{n-1}^*} \big( a_1 x^* + x a_{n-1}^* + f(a_1, a_{n-1}^*) \big)$
11      end
12  end
```

Proposition 2.8. *Algorithm* 2.7 *runs in* $O(nm^2 + mN_{x^*} + N)$.

Proof. Denote by $T_n(N)$ the cost to compute $F_{A,n}$ on a grid of size N. Algorithm 2.7 runs in $O(m^2 + mN_{x^*} + N + m^2 + T_{n-2}(m^2))$ since Line 6 takes $T_{n-2}(m^2)$ by calling the algorithm recursively, Line 10 takes $O(m^2 + mN_{x^*} + N)$ using the LLT2d algorithm, and we can compute Line 8 in $O(m^2)$ using Algorithm 2.9. Considering the case where $n-2$ is odd and the case where $n-2$ is even we can solve the recurrence equation to find $T_{n-2}(m^2) = O(nm^2)$ since $T_2(N) = T_3(N) = O(m^2 + mN_{x^*} + N)$ as shown above. So we deduce $T_n(N) = O(nm^2 + mN_{x^*} + N)$. □

Algorithm 2.9 (Algorithm for Line 8 of Algorithm 2.7).

```
Input: $\operatorname{gph} A = \{(\alpha_1, \alpha_1^*), \ldots, (\alpha_m, \alpha_m^*)\}$
Output: $f(a_1, a_{n-1}^*)$ for all $a_1 \in \{\alpha_i | i = 1, \ldots, m\}$ and all $a_{n-1}^* \in \{\alpha_j^* | j = 1, \ldots, m\}$
begin
    for $i = 1..m$ do
        for $j = 1..m$ do
            $F(i,j) = -\alpha_i\alpha_i^* - \alpha_j\alpha_j^* + \alpha_j\alpha_i^*$ ;
        end
    end
end
```

Remark 2.10. Continuing our typical case from Remark 2.4, Algorithm 2.7 takes around $10,000n$ operations to compute $F_{A,n}$ and about $1,000,000$ operations for $F_{A,\infty} = F_{A,m+1}$, a reduction by a factor of 100.

To further improve the complexity of computing $F_{A,\infty}$, we turn to piecewise linear-quadratic (PLQ) algorithms and Rockafellar functions. Our next algorithm, Algorithm 2.14, relies on the following explicit formula for Rockafellar functions of finite operators $A : \mathbb{R} \rightrightarrows \mathbb{R}$.

Fact 2.11 ([5, Theorem 3.14]). *Suppose that the graph of* $B\colon \mathbb{R} \rightrightarrows \mathbb{R}\colon x \mapsto \operatorname{conv}(Ax)$ *is*

$$\bigcup_{i=1}^{m} \big(\{a_i\} \times [b_i^-, b_i^+]\big),$$

where $m \in \{1, 2, \ldots\}$, $a_1 < a_2 < \cdots < a_m$, *and* $b_1^- \le b_1^+ \le b_2^- \le \cdots \le b_m^- \le b_m^+$. *Set* $a_0 := -\infty$ *and* $a_{m+1} := +\infty$. *Suppose that* $k \in \{1, \ldots, m\}$. *Then* $R_{A,a_k} : \mathbb{R} \to \mathbb{R}$ *is given by*

$$R_{A,a_k}(x) = \begin{cases} (x - a_i)b_i^- + \displaystyle\sum_{j=i+1}^{k} (a_{j-1} - a_j)b_j^-, & \text{if } a_{i-1} < x \le a_i \le a_k; \\ (x - a_i)b_i^+ + \displaystyle\sum_{j=k}^{i-1} (a_{j+1} - a_j)b_j^+, & \text{if } a_k \le a_i \le x < a_{i+1}, \end{cases} \tag{5}$$

Note that using these notations implies that the operator A is cyclically monotone, and $m_D = m$.

Lemma 2.12. *Computing* R_{A,a_k} *using Formula* (5) *takes* $\Theta(m)$ *time.*

	Definition	Recursive Formula	Fast Algorithm	PLQ Algorithm
$F_{A,n}$	$O(nm^{n-1}N)$	$O(nm^2N_{x^*}+mN)$	$O(nm^2+mN_{x^*}+N)$	
$F_{A,\infty}$	$O(m^{m+1}N)$	$O(m^3N_{x^*}+mN)$	$O(m^3+mN_{x^*}+N)$	$O(m^2+mN_x+N)$

Table 2.1: Complexity results for computing $F_{A,n}$ on a grid of size $N=N_xN_{x^*}$.

Proof. Since the function is written as a piecewise linear function with m pieces, we only need to show that each piece can be computed in $O(1)$. This is achieved by precomputing each sum $S^-=\sum_{j=2}^{k}(a_{j-1}-a_j)b_j^-$ and $S^+=\sum_{j=k}^{m-1}(a_{j+1}-a_j)b_j^+$, and updating them as x sweeps through intervals defined by the sequence $(a_i)_i$. $\square$

The link between Fitzpatrick functions and Rockafellar functions is provided by the following result.

Fact 2.13 ([5, Theorem 3.15]). *Let $A\colon \mathbb{R}\rightrightarrows\mathbb{R}$. Then*

$$\forall(x,x^*)\in\mathbb{R}\times\mathbb{R},\ \ F_{A,\infty}(x,x^*)=\sup_{a\in\operatorname{Dom}A}\left[ax^*+R_{A,a}(x)\right].$$

Algorithm 2.14 (Fitzpatrick function of infinite order Algorithm).

```
Input: gph $A$, $X$
Output: $F_{A,\infty}(x,x^*)$ for all $(x,x^*)\in X$
begin
    for $k=1..m$ do
        Compute $R_{A,a_k}$ using the explicit formula (5);
        Evaluate $R_{A,a_k}(x)$ for all $x\in X_x$;
    end
    for all $x\in X_x$ do
        Compute $F_{A,\infty}(x,x^*)=\max_{a\in\operatorname{Dom}A}[ax^*+R_{A,a}(x)]$ ;
    end
end
```

Proposition 2.15. *Algorithm* 2.14 *computes $F_{A,\infty}(x,x^*)$ for all $(x,x^*)\in X$ in $O(m^2+N+mN_x)$ time.*

Proof. The correctness of the algorithm is ensured by Fact 2.11 and Fact 2.13. For each $a\in\operatorname{Dom}A$, computing $R_{A,a}$ and storing it as a PLQ function takes $O(m)$ using the explicit formula while evaluating a PLQ function with m nodes at N_x points takes $O(m+N_x)$ using the plq_eval function [15]. So the first loop runs in $O(m(m+N_x))$. The second loop runs in $O(N_x(m+N_{x^*}))$ operations using the LLT1d algorithm for each value of x. Adding both complexities concludes the proof. $\square$

Table 2.1 summarizes our results while Table 3.1 emphasizes the typical case $N_x=N_{x^*}=m$. The computation of $F_{A,\infty}$ using the Fast LLT Algorithm is reduced to computing $F_{A,m+1}$ when the operator is cyclically monotone (otherwise $F_{A,\infty}$ is identically $+\infty$).

3. Antiderivatives

We now focus on the application of Fitzpatrick functions to the computation of antiderivatives.

Definition 3.1 ([5, Definition 3.1]). Let $A\colon \mathbb{R} \rightrightarrows \mathbb{R}$ and let f be convex lower-semi-continuous, and proper. Then f is an *antiderivative* of A if

$$\operatorname{gph} A \subset \operatorname{gph} \partial f. \tag{6}$$

The following facts will be useful in our computation of antiderivatives.

Fact 3.2 ([5]). *Assume $A : \mathbb{R} \rightrightarrows \mathbb{R}$ is a finite operator.*

(i) *The Rockafellar function $R_{A,a}$ is convex lower semicontinuous proper polyhedral, and a continuous antiderivative A.*

(ii) *If A is cyclically monotone, the function $f = F_{A,\infty}(\cdot, 0)$ is polyhedral continuous with full domain, and an antiderivative of A with*

$$F_{A,\infty}(\cdot, 0) = \max_{a \in \operatorname{Dom} A} R_{A,a}.$$

(iii) *If the operator A is not cyclically monotone, it does not admit antiderivatives. In that case, the functions $R_{A,a}$ and $F_{A,\infty}(\cdot, 0)$ are identically equal to $+\infty$ for any $a \in \operatorname{Dom} A$.*

While the computation of Rockafellar functions of a finite operator $A : \mathbb{R} \rightrightarrows \mathbb{R}$ takes $O(m)$, the computation of $F_{A,\infty}(\cdot, 0)$ takes $O(m^2 + mN_x)$ using the PLQ Algorithm. We now show how to reduce that computation to $\Theta(m + N_x)$ by computing an explicit formula.

First we need to define the following indices

$$j_0 = \max\{j = 0, \dots, m+1 | b_j^- < 0\} \quad \text{and} \quad j_1 = \max\{j = 0, \dots, m+1 | b_j^+ < 0\}, \tag{7}$$

where $b_0^- = b_0^+ = -\infty$ and $b_{m+1}^- = b_{m+1}^+ = +\infty$. We also simplify our notation by writing $R_{A,a_k} = R_{a_k}$ when there is no ambiguity on the operator A.

The relationship between j_0 and j_1 will be needed in our discussion later.

Lemma 3.3. *Assuming j_0 and j_1 are defined by (7) with b_i given as in (5), we have $j_1 \le j_0 \le j_1 + 1$.*

Proof. The relation $b_j^- \le b_j^+$ gives $j_1 \le j_0$. Now assume $j_0 > j_1 + 1$ i.e. $j_0 \ge j_1 + 2$. Then we have $b_{j_1}^+ < 0 \le b_{j_1+1}^+ \le b_{j_1+2}^- \le b_{j_0}^- < 0$, a contradiction which concludes the proof. □

We now split the computation of $\max_{1\le k\le m} R_{a_k}$ between computing $\max_{1\le k\le i} R_{a_k}$ and $\max_{i+1\le k\le m} R_{a_k}$.

Lemma 3.4. *Assume $a_i < x < a_{i+1}$. If $1 \le j_1 \le i$ then $\max_{1\le k\le i} R_{a_k} = R_{a_{j_1+1}}$, otherwise $i+1 \le j_1 \le m$ and $\max_{1\le k\le i} R_{a_k} = R_{a_i}$.*

Proof. Assume $1 \le k \le i$. We have

$$R_{a_k}(x) = (x - a_i)b_i^+ + \sum_{j=k}^{i-1}(a_{j+1} - a_j)b_j^+.$$

Now if $i+1 \le j_1 \le m$, the coefficients $b_j^+ < 0$ so $\max_k R_{a_k} = R_{a_i}$. Otherwise $1 \le j_1 \le i$ and the fact that $\max_k R_{a_k} = R_{a_{j_1+1}}$ is a consequence of the following formulas.

$$R_{a_k}(x) = R_{a_{j_1+1}}(x) + \sum_{j=k}^{j_1}(a_{j+1} - a_j)b_j^+,$$

$$R_{a_{j_1+1}}(x) = R_{a_k}(x) + \sum_{j=j_1+1}^{i-1}(a_{j+1} - a_j)b_j^+.$$

The first establishes $R_{a_k} \le R_{a_{j_1+1}}$ for $1 \le k \le j_1$ and the second for $j_1 < k$. □

Lemma 3.5. *Assume $a_i < x < a_{i+1}$. If $1 \le j_0 \le i$ then $\max_{i+1\le k\le m} R_{a_k} = R_{a_i+1}$, otherwise $i+1 \le j_0 \le m$ and $\max_{i+1\le k\le m} R_{a_k} = R_{a_{j_0}}$.*

Proof. Assume $i+1 \le k \le m$. The Rockafellar functions can be written as

$$R_{a_k}(x) = (x - a_{i+1})b_{i+1}^- + \sum_{j=i+2}^{k}(a_{j-1} - a_j)b_j^-.$$

So when $1 \le j_0 \le i$ we have $b_j^- \ge 0$ so $R_{a_k} \le R_{a_{i+1}}$. Otherwise $i+1 \le j_0 \le m$ and $\max_{i+1\le k\le m} R_{a_k} = R_{a_{j_0}}$ using the same argument as in the previous proof. □

We can now write the explicit formula for $F_{A,\infty}(x,0) = \max\limits_{1\le k\le m} R_{a_k}(x)$.

Proposition 3.6. *Assume $a_i < x < a_{i+1}$. Then*

$$\max_{1\le k\le m} R_{a_k}(x) = \begin{cases} R_{a_{j_1+1}}(x) = (x-a_i)b_i^+ + \displaystyle\sum_{j=j_1+1}^{i-1}(a_{j+1}-a_j)b_j^+ & \textit{when } j_0 < i, \\ R_{a_{j_0}}(x) = (x-a_{i+1})b_{i+1}^- + \displaystyle\sum_{j=i+2}^{j_0}(a_{j-1}-a_j)b_j^- & \textit{when } j_0 > i, \\ R_{a_{j_1+1}}(x) = (x-a_i)b_i^+ & \textit{when } j_0 = i \textit{ and } b_i^+ = b_{i+1}^-, \end{cases} \tag{8}$$

and when $j_0 = i$ but $b_i^+ < b_{i+1}^-$ we have

$$\max_{1\le k\le m} R_{a_k}(x) = \begin{cases} R_{a_{i+1}}(x) = (x-a_{i+1})b_{i+1}^- & \textit{when } \bar{x} \le x < a_{i+1}, \\ R_{a_{j_1+1}}(x) = (x-a_i)b_i^+ & \textit{when } a_i < x \le \bar{x}; \end{cases} \tag{9}$$

with $\bar{x} = (a_i b_i^+ - a_{i+1}b_{i+1}^-)/(b_i^+ - b_{i+1}^-)$ when $i > 0$, $\bar{x} = -\infty$ when $i = 0$, and $\bar{x} = +\infty$ when $i = m$.

Proof. First we consider the following two cases.

Assume $1 \le j_1 \le j_0 < i$. Then the two candidates for the maximum are

$$R_{a_{j_1+1}}(x) = (x-a_i)b_i^+ + \sum_{j=j_1+1}^{i-1}(a_{j+1}-a_j)b_j^+ \quad \text{and} \quad R_{a_{i+1}}(x) = (x-a_{i+1})b_{i+1}^-.$$

	Definition	Recursive Formula	Fast Algorithm	PLQ Algorithm	Explicit Formula
$F_{A,n}$	$O(nm^{n+1})$	$O(nm^3)$	$O(nm^2)$		
$F_{A,\infty}$	$O(m^{m+3})$	$O(m^4)$	$O(m^3)$	$\Theta(m^2)$	
$F_{A,\infty}(\cdot,0)$	$O(m^{m+3})$	$O(m^4)$	$O(m^3)$	$O(m^2)$	$\Theta(m)$

Table 3.1: Complexity results for computing $F_{A,n}$ when $N_x = N_{x^*} = m$.

Since $i+1 > j_0$, $b^-_{i+1} \geq 0$; and $i > j_1$ implies $b^+_i \geq 0$. Moreover, for all $j > j_1$ we have $b^+_j \geq 0$. Taking the sign of all coefficients b_j into account we deduce $R_{a_{i+1}} \leq 0 \leq R_{a_{j_1+1}}$.

Now assume $i+1 \leq j_1 \leq j_0 \leq m$. The maximum is taken between the two functions

$$R_{a_{j_0}}(x) = (x - a_{i+1})b^-_{i+1} + \sum_{i+2}^{j_0}(a_{j-1} - a_j)b^-_j \quad \text{and} \quad R_{a_i}(x) = (x - a_i)b^+_i.$$

A similar argument gives $R_{a_i} \leq 0 \leq R_{a_{j_0}}$.

Now using Lemma 3.3 we can partition the values taken by indexes j_0 and j_1 as follows: either $j_0 = j_1$ then the cases $j_0 < i$ and $i < j_0$ have been treated above and we only need to consider the case $j_0 = i$, or $j_0 = j_1 + 1$ and again the only remaining case is $j_0 = i$ since $j_0 < i$ and $j_0 > i$ are covered in the previous two cases.

Assume $j_0 = i$. Our two candidates are now

$$R_{a_{j_1+1}}(x) = (x - a_i)b^+_i \quad \text{and} \quad R_{a_{i+1}}(x) = (x - a_{i+1})b^-_{i+1}.$$

So if $j_0 = j_1 + 1$ then $b^+_i = b^+_{j_1+1} \geq 0$ and the maximum is $R_{a_{j_1+1}} \geq 0 \geq R_{a_i+1}$, which proves the first part of the proposition.

The last case is when $j_0 = j_1 = i$. Then $b^+_i \neq b^-_{i+1}$ otherwise $0 > b^+_{j_1} = b^-_{j_0+1} \geq 0$ which is impossible. So $b^+_i < b^-_{i+1}$, we can compute $\bar{x}$ and the result follows as the function $x \mapsto (x - a_i)b^+_i$ is decreasing from 0 when $x = a_i$, and the function $x \mapsto (x - a_{i+1})b^-_{i+1}$ is increasing to 0 (reached when $x = a_{i+1}$). $\square$

Corollary 3.7. *Computing the Fitzpatrick function of infinite order $F_{A,\infty}$ on a set $(x, 0) \in X$ takes $\Theta(m + N_x)$.*

Proof. The size of the input is $\Omega(m + N_x)$, which gives the trivial lower bound. Proposition 3.6 allows one to compute $F_{A,\infty}$ in $O(m)$ and store it as in PLQ format. Then evaluating on the grid using the PLQ evaluation function takes $O(m + N_x)$. $\square$

Hence we reduce the computation of $F_{A,\infty}(x, 0)$ for all $x \in X_x$ when $N_x = m = 100$ from 10^{206} (using the definition) to 10^8 (using recursive Formula 3) to 10^6 (using the LLT2d fast algorithm through Algorithm 2.7) to 10^4 (using the PLQ Algorithm 2.14) to the optimal value of 10^2 (using Proposition 3.6). Table 3.1 summarizes our complexity results.

4. Conclusion

While Fitzpatrick functions have become the subject of several works recently, very little is known on how to compute them efficiently even in the case of finite operators. We tackle this problem and provide polynomial time algorithms to compute Fitzpatrick functions of order n and of infinite order. Our algorithms build on the Linear-time Legendre Transform algorithm, to compute the Legendre-Fenchel conjugate in linear time, and on PLQ algorithms, to evaluate efficiently piecewise linear functions on a grid. Considering it has important applications for computing antiderivatives, we also provided an explicit formula to compute $F_{A,\infty}(\cdot, 0)$ in $\Theta(m + N)$ on a grid of size N, which is optimal.

While our results are limited to operators on the real line, they already reveal the complexity of such computations. Future work will focus on extending them to higher dimensions, which requires extending PLQ algorithms to higher dimensions, a subject of active investigation.

References

[1] E. Asplund: A monotone convergence theorem for sequences of nonlinear mappings, in: Nonlinear Functional Analysis (Chicago, 1968), Proc. Sympos. Pure Math. 18, Part 1, Amer. Math. Soc., Providence (1970) 1–9.

[2] S. Bartz, H. H. Bauschke, J. M. Borwein, S. Reich, X. Wang: Fitzpatrick functions, cyclic monotonicity and Rockafellar's antiderivative, Nonlinear Anal., Theory Methods Appl. 66A (2007) 1198–1223.

[3] H. H. Bauschke: Fenchel duality, Fitzpatrick functions and the extension of firmly nonexpansive mappings, Proc. Amer. Math. Soc. 135 (2007) 135–139.

[4] H. H. Bauschke, J. M. Borwein, X. Wang: Fitzpatrick functions and continuous linear monotone operators, SIAM J. Optim. 18 (2007) 789–809.

[5] H. H. Bauschke, Y. Lucet, X. Wang: Primal-dual symmetric intrinsic methods for finding antiderivatives of cyclically monotone operators, SIAM J. Control Optim. 46 (2007) 2031–2051.

[6] H. H. Bauschke, D. A. McLaren, H. S. Sendov: Fitzpatrick functions: inequalities, examples, and remarks on a problem by S. Fitzpatrick, J. Convex Analysis 13 (2006) 499–523.

[7] H. H. Bauschke, X. Wang: A convex-analytical approach to extension results for n-cyclically monotone operators, Set-Valued Anal. 15 (2007) 297–306.

[8] J. M. Borwein: Maximal monotonicity via convex analysis, J. Convex Analysis 13 (2006) 561–586.

[9] J. M. Borwein, C. H. Hamilton: Symbolic Fenchel conjugation, Math. Program., Ser. B 116 (2009) 17–35.

[10] R. I. Boţ, E. R. Csetnek, G. Wanka: A new condition for maximal monotonicity via representative functions, Nonlinear Anal., Theory Methods Appl. 67A (2007) 2390–2402.

[11] P. F. Felzenszwalb, D. P. Huttenlocher: Distance transforms of sampled functions, Tech. Rep. TR2004-1963, Cornell Computing and Information Science (Sept. 2004).

[12] S. Fitzpatrick: Representing monotone operators by convex functions, in: Functional Analysis and Optimization, Workshop / Miniconference (Canberra, 1988), Proc. Cent. Math. Anal. Aust. Natl. Univ. 20, Australian National University, Canberra (1988) 59–65.

[13] D. Lambert, J.-P. Crouzeix, V. H. Nguyen, J.-J. Strodiot: Finite convex integration, J. Convex Analysis 11 (2004) 131–146.

[14] Y. Lucet: Faster than the fast Legendre transform, the linear-time Legendre transform, Numer. Algorithms 16 (1997) 171–185.

[15] Y. Lucet, H. H. Bauschke, M. Trienis: The piecewise linear-quadratic model for computational convex analysis, Comput. Optim. Appl. 43 (2009) 95–118.

[16] J.-P. Penot, C. Zălinescu: Continuity of the Legendre-Fenchel transform for some variational convergences, Optimization 53 (2004) 549–562.

[17] S. Reich, S. Simons: Fenchel duality, Fitzpatrick functions and the Kirszbraun-Valentine extension theorem, Proc. Amer. Math. Soc. 133 (2005) 2657–2660.

[18] R. T. Rockafellar: Convex Analysis, Princeton University Press, Princeton (1970).

[19] R. T. Rockafellar: On the maximal monotonicity of subdifferential mappings, Pacific J. Math. 33 (1970) 209–216.

[20] S. Simons: Dualized and scaled Fitzpatrick functions, Proc. Amer. Math. Soc. 134 (2006) 2983–2987.

[21] S. Simons: The Fitzpatrick function and nonreflexive spaces, J. Convex Analysis 13 (2006) 861–881.

[22] S. Simons, C. Zălinescu: Fenchel duality, Fitzpatrick functions and maximal monotonicity, J. Nonlinear Convex Anal. 6 (2005) 1–22.

[23] M. D. Voisei: Extension theorems for k-monotone operators, Stud. Cercet. Ştiinţ., Ser. Mat., Univ. Bacău 9 (1999) 235–242

[24] C. Zălinescu: A new proof of the maximal monotonicity of the sum using the Fitzpatrick function, in: Variational Analysis and Applications, F. Giannessi, A. Maugeri (eds.), Nonconvex Optim. Appl. 79, Springer, New York (2005) 1159–1172.