信州大学工学部

学士論文

到着期限とデータ量制約のある伝送システムにおける歪み最小化アルゴリズム

指導教員　西新 幹彦 准教授

学科　電気電子工学科
学籍番号　07T2024H
氏名　金澤 秀樹

2012 年 03 月 16 日

目次

1 序章

今日の社会では，インターネットは生活する上で欠かせないものになってきている．特に動画による情報のやり取りは重要な情報伝達の場として活用されている．動画の場合，情報量が多いので記事や画像情報と違いクリックしてすぐにファイル全体をダウンロードすることは出来ない．そこで昨今のインターネットでは動画を見ながらダウンロードするビデオ・オン・デマンドまたはストリーミングと呼ばれている手法が用いられている．

ビデオ・オン・デマンドでは，決められた時刻までにフレームを再生するためのデータが届いていないと，なめらかな動画の再生が出来ない．これはフレームデータに到着期限があることを意味する．その上で視聴者はできるだけ高画質の動画を見たいと思っている．再生フレームの画質を向上させるには，送信するデータ量を増やさなければならない．送信データが増えるほど再生されるフレームはオリジナルのフレームに近づくと考えてよい．再生されるフレームとオリジナルのフレームの差を歪みと呼ぶ．したがって視聴者の要求は，動画全体にわたるフレームの歪みの総和を最小にすることであるといえる．

Faridi and Ephremides[1] はこのような問題をいくつかのモデルで表現し，それぞれのモデルに対して最適な伝送計画を求めるアルゴリズムの提案を試みた．本研究では，彼らのモデルをさらに一般化したモデルに対して最適な伝送計画を求めるアルゴリズムを提案する．また，彼らの提案アルゴリズムのひとつに到着期限を守らない反例があることを指摘する．そのアルゴリズムが対象とするモデルは本研究で扱うモデルの特殊な例として表されるので，本研究の提案アルゴリズムはそのモデルに対しても真に最適な伝送計画を導出する．

従来研究のモデルやアルゴリズムの詳細は次章以降で論じる．

2 伝送システムと歪み

この章では伝送システムのモデルと歪みについて定義する．本論文に登場する主な記号の意味をあらかじめ表 1 にまとめる．

2.1 伝送システムのモデル

従来研究や本研究で考える伝送システムの概略図は図 1 のように表される．以下図 1 を参照しながら伝送システムのモデルを説明する．

送信したい動画は N 個のフレームからなるとし，それらは最初すべて情報源に格納されているとする．各フレームは到着期限 M_i $(i = 1, \ldots, N)$ を持っている．本研究では問題を一般化し，到着期限は等間隔であるとは限らない．ただし一般性を損なうことなく，フレームの順番は到着期限の切れる順とする．すなわち $M_i \leq M_2 \leq \cdots \leq M_N$ とする．

情報源に格納されている各フレームは符号器によって符号化され，パケットを単位として伝送路に送られる．一般性を損なうことなく，伝送路は単位時間あたりに 1 パケットを受信機に届けることができるとする．また，一般にパケットは伝送中に誤りを生じることがあり，正しく受信機に届く確率を独立同一分布の p とする．以降，本論文では $p = 1$ の場合と $p \leq 1$ の場合を分けて考える．

スケジューラは，符号器がどのフレームを符号化するかを制御し，到着期限を過ぎたフレームを符号化しないようにするとともに，復号器で再生される動画の歪みを最小化する．したがってスケジューラは情報源をみて適切な伝送計画を立てなければならない．さらに，伝送路に雑音がある $p \leq 1$ の場合，受信機が誤りの有無を検出してスケジューラにフィードバックするモデルも考えられる．この場合，スケジューラはフィードバックに応じて伝送計画を動的に変えなければならない．このスケジューラの動きを決定することが研究の主題である．

もうひとつ，モデルを区分する特徴がある．それは情報源に格納された各フレームを符号化する際のパケット数の上限の有無である．もしフレームが連続値で表現されていると情報量は無限大になり，パケット数には上限は設けられない．一方，フレームの情報量が有限であればパケット数には上限が存在する．本研究では，上限があるモデルについて考察するが，上限なしのモデルは上限を無限大とした特殊な場合である．

以上のモデルのうち，本研究と本研究に関連する従来研究で扱うものについて表 2 にまとめる．

表 1　本研究で使われている記号の説明

記号	説明
N	フレームの総数
M_i	フレーム i の到着期限
p	データの伝送成功確率
$d(\cdot)$	歪み関数（単調減少で狭義に下に凸）
n	伝送アルゴリズム中，到着期限が残っているフレームの数
b_i	フレーム i に対して，すでに復号器に届いているデータ量
m_i	フレーム i の到着期限になるまでの残り時間
a_i	フレーム i のデータ量の上限
y_i	フレーム i に対する送信データ量（伝送計画）
B_i	フレーム i に対して，最終的に届くデータ量（確率変数）
x_i	フレーム i に対して，最終的に届くデータ量の期待値

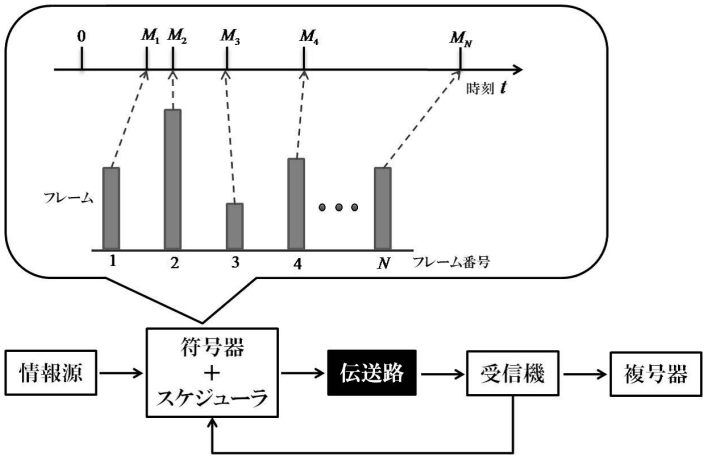

図 1　伝送モデル

表 2　従来研究の紹介

	伝送路の誤り	フィードバック	データ量上限	アルゴリズム
モデル 1	なし	—	なし	[3]
モデル 2	なし	—	あり	[1](反例)，本研究
モデル 3	あり	なし	なし	[1]
モデル 4	あり	あり	なし	[1](反例)，[4]
モデル 5	あり	あり	あり	本研究

2.2　歪み関数

動画の歪みは各フレームの歪みの総和で定義される．フレームの歪みを正確にモデル化しようとすれば，それは再生されたフレームとオリジナルのフレームの距離として定義されるべきであろう．しかし，本研究では，簡単化のため従来研究 [1] に従い，歪みを受け取ったパケット数で表される 1 変数関数とし，単調減少で狭義に下に凸な関数と仮定する．単調減少の妥当性は自明であるが，凸性の妥当性は次のように説明できる．送信するデータが画像であったとすると，データ量が多くなるほど画質はよくなる．つまり元の画像により近づくため歪みが小さくなると考えられる．このとき，データ量が大きくなる途中で画質が悪くなるということは起こりにくい．そのため，単調減少になると仮定する．凸性の妥当性は画質の悪い画像と綺麗な画像に同じ量のデータを追加したすると，その画質を両方ともよくすることができる．しかし悪い画像のほうが画質の改善の程度の変化が大きいと考えられる．そのため，下に凸であるということを仮定する．

届いたデータ量が x のフレームに対し，その歪みを $d(x)$ と表す．

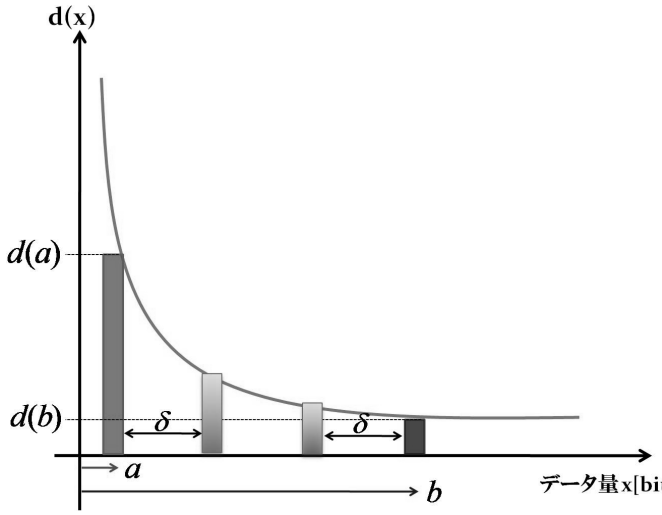

図 2　歪み関数

2.3　狭義に下に凸な減少関数の性質

狭義に下に凸な減少関数には補題 1 のような性質がある．

補題 1　$d(\cdot)$ が狭義に下に凸な減少関数であったとき，$a, b, \delta \in \mathbb{R}$, そして $0 < \delta < b - a$ であるとき，次式が成立する．(図 2)

$$d(a+\delta) + d(b-\delta) < d(a) + d(b) \tag{1}$$

証明: 関数が狭義に下に凸であるとは

$$d\big((1-\lambda)a + \lambda b\big) < (1-\lambda)d(a) + \lambda d(b) \tag{2}$$

を満たすことである．ここで λ は $0 < \lambda < 1$ となる任意の定数である．そのため

$$d\big(\lambda a + (1-\lambda)b\big) < \lambda d(a) + (1-\lambda)d(b) \tag{3}$$

も成立する．式 (2)(3) を足すと,

$$d\big(a + (b-a)\lambda\big) + d\big(\lambda(a-b) + b\big) < d(a) + d(b) \tag{4}$$

となる．ここで $\lambda = \frac{\delta}{b-a}$ とおくと, 式 (4) は

$$d(a+\delta) + d(b-\delta) < d(a) + d(b) \tag{5}$$

となり，補題 1 が証明された．□

補題 1 は 2 つのフレームにおいて双方のデータ量の和が等しいならば双方のデータ量の差が小さい方が歪みの総和が小さいと言うことを意味する．

3 開ループシステム

ここでは表 2 のモデル 2 に対する従来研究 [1] を紹介し，反例を示す．モデル 2 では伝送路に誤りがないので，フィードバックの必要がない．またそれゆえ，スケジューラが始めに立てた伝送計画は途中で変更されずにそのまま実行される．

入力: $p,\ n,\ \boldsymbol{m},\ \boldsymbol{a}$
出力: $\boldsymbol{x}$

1. $e_1 \stackrel{\text{def}}{=} \{1\}, e_2 \stackrel{\text{def}}{=} \{1,2\}, \ldots, e_N \stackrel{\text{def}}{=} \{1,2,,\ldots,n\}$
2. $\boldsymbol{e} = \{1,2,\ldots,N\}$
3. $\mu_i = \frac{m_i}{i}\ (j = 1,\ldots,n)$
4. $\alpha = min\{\mu_1, \mu_2, \ldots, \mu_N, a_1, a_2, \ldots, a_N\}$
5. $a_i = \alpha$ なる $i \in \boldsymbol{e}$ に対して $x_i = \alpha$
6. $\hat{\jmath} = max\{j \in \boldsymbol{e} \mid \mu_{\hat{\jmath}} = \alpha\}$
7. $i \in e_j \bigcap e$ に対して $x_i = \alpha$
8. $\boldsymbol{e}$ から決まったものを除く。
9. $i \in \boldsymbol{e}$ に対して　$\mu_j = \frac{m_i - \sum_{i \in e_j \bigcap e} x_i}{|e_j \bigcap e|}$

終了.

図 3　上限あり開ループアルゴリズム

モデル 2 に対する従来研究 [1] の提案アルゴリズムを図 3 に示す．従来研究 [1] のアルゴリズムが出力する伝送計画は到着期限が過ぎたフレームに対してパケットを送信することがある．その反例は $n = 5,\ \boldsymbol{m} = \{1,\ 1.1,\ 2,\ 2,\ 2.51\},\ \boldsymbol{a} = \{10,\ 0.5,\ 10,\ 0.49,\ 10\}$ である．これに対する上限有り開ループアルゴリズムの出力は $\boldsymbol{x} = \{71/150,\ 1/2,\ 71/150,\ 49/100,\ 71/150\}$ となる．このとき $\sum_{i=1}^{4} x_i = 601/300 > \sum_{i=1}^{4} b_i + pm_4 = 2$ となり フレーム 4 の到着期限が過ぎたあとでもフレーム 4 のパケットを送信している．

このように従来研究のアルゴリズムは到着期限を過ぎた伝送計画を導く場合がある．到着期限を過ぎてしまった原因は以下のように考えられる．4 フレーム目に送信する予定のデータ量が上限値に決定した後に前のフレームに送信するデータ量が増加した．そのときにすでに送信するデータ量が決まったフレームが到着期限を越えていないかのチェックが抜けていたことが原因と考えられる．5 章で説明する本研究の提案法はモデル 2 に対しても正しく最適解を導くことができる．

1. 現状に対して最適な伝送計画を見つける.
2. 伝送計画をもとにパケットを送信するフレームを決定する.
3. パケットを送信し，受信状況を受け取る.
4. 現状を更新する.
5. 到着期限が切れたフレームは対象から取り除く.
6. 到着期限が残っているフレームがあるなら手順 (1) に戻る.
7. 終了.

図 4　伝送アルゴリズムの概略

4　閉ループシステム

この章では表 2 のモデル 5 に対するシステムの詳細を説明する．モデル 5 はモデル 4 に対してデータ量上限の制約を追加して一般化したものになっている．従ってモデル 5 はモデル 4 を含んでいる．以降，モデル 4 と共通する部分については文献 [4] による説明を引用する．

このモデルでは，受信機は誤りの有無を検出してスケジューラにフィードバックする．よってスケジューラは最終的な動画の歪みを最小化するため，フィードバックの内容に従って以降の伝送計画を見直さなくてはならない.

そこで従来研究 [1] では，確実性等価制御 (Certainty Equivalent Controllers，CEC)[2] の考えを用いて伝送計画を随時更新することを提案している．CEC は，観測していないランダムな値の期待値を求め，各段階で最適解を毎回計算する発見的手法である.

本モデルに CEC の手法を用いると，システムの動きは以下のようになる．まず，スケジューラは現在の時点で現状に基づき，歪みの期待値が最小となるように現在以降の伝送計画を立てる．これをもとにフレームを選んでパケットを送信する．次の時刻では，送信したパケットが無事に復号器に届いたかどうかに基づき，スケジューラは把握している現状を更新する．また，時刻が進んでいることから，各フレームの到着期限も更新する．もし到着期限が残っているフレームがあれば，それらに対して再び伝送計画を立てる．これをすべての（最後の）フレームの到着期限が切れるまで繰り返す.

この手順をまとめたのが，図 4 の伝送アルゴリズムである．現状に対する伝送計画と，パケットを送信するフレームの選び方（伝送指針）については次節で詳しく説明する．また，伝送計画を見つけるにあたって，到着期限が 0 以下となったフレームにはデータを送信することはできないため考える必要はない [1]. しかし，データはパケット単位で送信されるため，到着期限が 1 パケットを送信する時間に満たない場合，そのフレームにデータを送信することはできない．そのため，伝送アルゴリズムの概略の手順 5 の到着期限が切れたフレームとは，到着期限が 1 パケットを送信する時間に満たないフレームを指す.

4.1 伝送計画

各パケットに対し，すでに届いているデータ量と到着期限までの残り時間をそれぞれ $\boldsymbol{b} = (b_1, \ldots, b_n)$, $\boldsymbol{m} = (m_1, \ldots, m_n)$ とする．これらはスケジューラが把握している現状を表しており, 初期値は $b_i = 0, m_i = M_i, n = N$ である．現時点での伝送計画を $\boldsymbol{y} = (y_1, \ldots, y_n)$ と表す．

本来のシステムの目的からすれば，計画通りに送信する場合に最終的に届くデータ量を確率変数 B_i とおいて，現状における歪みの条件付き期待値

$$\mathbb{E}\left[\sum_{i=1}^{n} d(B_i) \middle| \boldsymbol{b}, \boldsymbol{y}\right] \tag{6}$$

を最小化する伝送計画 $\boldsymbol{y}$ を求めなければならない．しかし，このような伝送計画を見つけることは計算量的に高価となる．そこで，(6) の代わりに B_i の条件付き期待値を用いて

$$\sum_{i=1}^{n} d\left(\mathbb{E}[B_i | b_i, y_i]\right) \tag{7}$$

を最小化する伝送計画 $\boldsymbol{y}$ を用いることにする．歪み関数の性質を利用すると，(7) を最小化するために歪み関数 $d(\cdot)$ の値を評価する必要がない．したがって計算量的に非常に安価に計算することができることが利点である．また，(7) を最小化する伝送計画を用いたときの最終的な全体の歪みは，(6) を用いたときの歪みに匹敵することが実験的に示されている [1].

B_i の条件付き期待値を x_i と表すと

$$x_i = \mathbb{E}[B_i | b_i, y_i] = b_i + p y_i \tag{8}$$

が成り立つ．このことから $\boldsymbol{x} = (x_1, \ldots, x_n)$ のことも伝送計画と呼ぶ．負のデータ量を送ることはできないことと，データ量の上限を超えて送信することはできないことから

$$a_i \geq x_i \geq b_i \quad (i = 1, \ldots, n) \tag{9}$$

が成り立たなければならない．また，データは到着期限の近い順に送信し，到着期限を越えてデータを送信しないので

$$\sum_{j=1}^{i} y_j \leq m_i \tag{10}$$

でなくてはならない．ただしここで，1 パケットを送るのに単位時間を要することを用いた．(10) は x_i を用いると

$$\sum_{j=1}^{i} x_j \leq \sum_{j=1}^{i} b_j + p m_i \quad (i = 1, \ldots, n) \tag{11}$$

と表される．(9) と (11) を満たす伝送計画を**実行可能**であるという．

(7) より，最適な伝送計画を求める最適化問題を次のように定式化する．

$\mathbf{P_{CEC}}$：

$$\sum_{i=1}^{n} d(x_i) \tag{12}$$

を最小化する実行可能な伝送計画 $\boldsymbol{x}$ を求めよ．

4.2 伝送指針

$\mathrm{P_{CEC}}$ の解から (8) によって計算した最適な伝送計画 $\boldsymbol{y}$ は今後どれだけ各フレームにデータを送信すれば歪みを最小化できるかを示しているのであって，今からどのフレームのパケットを送信すればいいかは示していない．伝送計画が導くデータ量は連続量であり，パケットの大きさは固定されているからである．そのため，パケットを送信するフレームを決める伝送指針が必要となる．

伝送指針は，パケットが無事に届いたとしても次の時刻に

$$b_1 \geq \cdots \geq b_n \tag{13}$$

$$a_i \geq b_i \quad (i = 1, \ldots, n) \tag{14}$$

が成り立つようにフレームを選択する．この選択の仕方によって，到着期限前の 2 つのフレームの間の最適性に損失は発生しない．なぜなら，始めのフレームの到着期限は，後ろのフレームの到着期限より前のため，始めのフレームよりも後ろのフレームにデータを多く送っても歪みを小さくすることはできないからである．

このような伝送指針 ψ の例として次がある [1]．

$$\psi(\boldsymbol{y}) = \min\left\{\arg\max_{1 \leq i \leq j^*} \{y_i\}\right\} \tag{15}$$

$$j^* = \min\left\{j \,\middle|\, \sum_{i=1}^{j} y_i \geq 1\right\} \tag{16}$$

5 閉ループシステムに対する提案法

この章ではモデル 5 において最適な伝送計画を求めるアルゴリズムを提案する．提案アルゴリズムを図 5 に示す．提案アルゴリズムは，伝送計画が実行可能であることを常に保証しながら，送信データの少ないフレームに対して少しずつ送信データを追加する．補題 1 が意味するように，少ないデータ量のフレームに対してデータ量を増加させることがもっとも歪みを減らす効果のあることから，提案アルゴリズムは最適な伝送計画を導くことが分かる．

入力: $p,\ n,\ \boldsymbol{b},\ \boldsymbol{m},\ \boldsymbol{a}$
出力: $\boldsymbol{x}$

1. $C_j \leftarrow \sum_{i=1}^{j} b_i + pm_j\ (j = 1, \ldots, n)$
2. $\boldsymbol{x} \leftarrow \boldsymbol{b}$
3. $e_i \leftarrow \begin{cases} 1 & (x_i = \min x_j) \\ 0 & (otherwise) \end{cases}$
4. $e_i = 0\{i = 1, \ldots, n\}$ なる $i \in (1, \ldots, n)$ があれば $x_k = \min\{x_j, e_i = 0, i = 1, \ldots, n\}$ を選ぶ。
5. $\alpha^{(1)} \leftarrow x_k - \min x_j$
6. $e_i = 1$ なる各 i に対して $\alpha_i^{(2)} \leftarrow a_i - x_i$
7. $e_i = 1$ なる各 i に対して $\alpha_i^{(3)} \leftarrow \frac{c_j - \sum_{i=1}^{i} x_j}{\sum_{j=1}^{i} e_j}$
8. $\alpha^{(1)}, \alpha_i^{(2)}, \alpha_i^{(3)}$ の中で最小値を探す
9. $\alpha^{(1)}$ が最小値ならば手順 (9.1) を，$\alpha_i^{(2)}$ ならば (9.2) を，$\alpha_i^{(3)}$ ならば (9.3) を実行する。
   - (9.1) $e_i = 1$ なる i に対して $x_i \leftarrow x_i + \alpha^{(1)}$ として $e_k \leftarrow 1$ 手順 (4) を実行。
   - (9.2) $e_i = 1$ なる i に対して $x_i \leftarrow x_i + \alpha_2^{(1)}$ として $e_j \leftarrow 0$
   - (9.3) $e_i = 1$ なる i に対して $x_i \leftarrow x_i + \alpha_3^{(1)}$ として $e_1, \ldots, e_j \leftarrow 0$
10. $\boldsymbol{e} = 0$ ならば終了。そうでないなら手順 (4) を実行。

図 5　上限あり CEC アルゴリズム

定理 1　単調減少で狭義に下に凸な歪み関数 $d(\cdot)$ のもとで，提案アルゴリズムの出力は $\mathrm{P}_{\mathrm{CEC}}$ の最適解である．

モデル 5 の伝送路では一般に誤りがあるが $(p \leq 1)$，$p = 1$ の特殊な場合はフィードバックは不要になり，モデル 2 に帰着する．さらに確実性等価制御 (CEC) も不要であるので，最初に立てた伝送計画は途中で修正されず，したがって初期値 $\boldsymbol{b} = 0$ を用いて 1 度計算すればよい．このように，モデル 2 はモデル 5 の特殊な場合であるので，3 章で指摘した反例のあるアルゴリズムに代わり，提案アルゴリズムはモデル 2 においても最適な伝送計画を導く．

また，モデル 5 におけるデータ量上限を無限大としてもアルゴリズムは正しく動く．したがってデータ量上限のないモデル 4 に対しても提案アルゴリズムは最適な伝送計画を導く．

一方，表 2 に掲げたように，モデル 4 に対するアルゴリズムとして文献 [4] の方法がある．よって，モデル 4 に対しては 2 つのアルゴリズムがあることになる．しかし，文献 [4] の方法は先頭のフレームから伝送計画が決まっていくので提案法より速度が速いと思われる．そのためにモデル 4 の伝送計画を行うなら文献 [4] のアルゴリズムを用いるのが良いと思われる．

6　まとめ

本研究では伝送モデルの違いによって問題要素に違いがある．問題要素の組み合わせの中で上限あり，伝送路に誤りが無い場合の従来法 [1] でのアルゴリズムに反例を発見した．また，さらに一般化した上限あり，伝送路に誤りあり，フィードバックありの問題要素のときのアルゴリズムを発見した．

謝辞

本研究を行うにあたり，数多くの助言，指導をしていただいた指導教員西新幹彦准教授，また西新研究室の皆様に感謝の意を表する．

参考文献

[1] Azadeh Faridi, Anthony Ephremides, “Distortion Control For Delay-sensitive Sources,” IEEE Transaction on information, Vol.54, No.8, pp.3399–3406, August 2008.

[2] D.P. Bertsekas, *Dynamic Programming and Optimal Control*, vol.I, 3rd ed., Athena Scentific, 2005.

[3] 原博紀, “到着期限のあるデータ伝送における歪み最小化アルゴリズム”, 信州大学工学部学士論文, 2010.

[4] 岩井祐斗, “到着期限のある閉ループ伝送システムにおける歪み最小化アルゴリズム”, 信州大学工学部学士論文, 2011.

付録 A　ソースコード

ここでは，5 章で提案したアルゴリズムのプログラムを記載する．

A.1　提案アルゴリズムのプログラム

```
#include <stdio.h>
#define FRAMEBOUND 100


int n;
double b[FRAMEBOUND];
double pm[FRAMEBOUND];
double a[FRAMEBOUND];
double x[FRAMEBOUND];

void
cec_proc(void)
{
    double c[FRAMEBOUND];
    double b_cum;
    double x_min;
    int x_min_idx;
    int e[FRAMEBOUND];
    double bnd1;
    double bnd2;
    double bnd3;
    int bnd1_idx;
    int bnd2_idx;
    int bnd3_idx;
    double x_cum;
    int e_cum;
    int i;

    for (b_cum = 0, i = 0; i < n; i++){
        b_cum += b[i];
        c[i] = b_cum + pm[i];
        x[i] = b[i];
    }

    x_min_idx = 0;
    x_min = x[0];
    for (i = 0; i < n; i++){
        if (x[i] < x_min){
            x_min_idx = i;
            x_min = x[i];
        }
    }

    for (i = 0; i < n; i++){
        e[i] = (x[i] == x_min)? 1: 0;
    }

LOOP1:

    for (i = 0; e[i] == 0; i++){
        ;
    }
    bnd1_idx = i;
    bnd1 = a[i] - x[i];
    for (i++; i < n; i++){
        if (e[i] == 1 && a[i] - x[i] < bnd1){
            bnd1_idx = i;
            bnd1 = a[i] - x[i];
        }
    }

    x_cum = 0;
    for (i = 0; e[i] == 0; i++){
        x_cum += x[i];
```

```
    }
    x_cum += x[i];
    e_cum = 1;
    bnd2_idx = i;
    bnd2 = (c[i] - x_cum) / e_cum;
    for (i++; i < n; i++){
        double tmp;
        x_cum += x[i];
        e_cum += e[i];
        tmp = (c[i] - x_cum) / e_cum;
        if (tmp < bnd2){
            bnd2_idx = i;
            bnd2 = tmp;
        }
    }

    for (i = 0; e[i] == 1 && i < n; i++){
        ;
    }
    if (e[i] == 0){
        bnd3_idx = i;
        bnd3 = x[i];
        for (i++; i < n; i++){
            if (e[i] == 0 && x[i] < bnd3){
                bnd3_idx = i;
                bnd3 = x[i];
            }
        }
        bnd3 -= x_min;
        if (bnd3 > 0 && bnd3 < bnd1 && bnd3 < bnd2){
            for (i = 0; i < n; i++){
                if (e[i] == 1){
                    x[i] += bnd3;
                }
            }
            x_min += bnd3;
            e[bnd3_idx] = 1;
        } else {
            goto BOUND1OR2;
        }
    } else {
    BOUND1OR2:
        if (bnd1 < bnd2){
            for (i = 0; i < n; i++){
                if (e[i] == 1){
                    x[i] += bnd1;
                }
            }
            x_min += bnd1;
            e[bnd1_idx] = 0;
        } else {
            for (i = 0; i < n; i++){
                if (e[i] == 1){
                    x[i] += bnd2;
                }
            }
            x_min += bnd2;
            for (i = 0; i <= bnd2_idx; i++){
                e[i] = 0;
            }
        }
    }

    for (i = 0; i < n; i++){
        if (e[i] == 1){
            goto LOOP1;
        }
    }

    return;
}

int
main()
{
    int i;
```

```
    printf("n=");
    scanf("%d", &n);
    if (n <= 0 || n > FRAMEBOUND){
        goto ERROR;
    }
    for (i = 0; i < n; i++){
        printf("b[%d]=",i);
        scanf("%lf", b + i);
    }
    for (i = 0; i < n; i++){
        printf("pm[%d]=",i);
        scanf("%lf", pm + i);
    }
    for (i = 0; i < n; i++){
        printf("a[%d]=",i);
        scanf("%lf", a + i);
    }
    cec_proc();
    for (i = 0; i < n; i++){
        printf("x[%d]=",i);
        printf(" %f\n", x[i]);
    }
    printf("\n");
    return(0);
ERROR:
    return(-1);
}
```