ノンパラメトリック予測推論に基づいたソフトウェアの適応的3ステップ若化方策

林　坂　弘一郎[*1]
土　肥　　正[*2]

概要

本論文ではエージング現象が観測されるようなソフトウェアシステムに対して最適ソフトウェア若化スケジュールを適応的に導出することを考える．具体的には，n 個の障害発生時間データからノンパラメトリック予測推論に基づいた $n+1$ 番目の障害発生時間に対する予測信頼度関数の上下限を定式化し，打切り時間データも考慮した上で $n+2$ 番目に対しても予測信頼度関数の上下限を定式化する．さらに，$n+3$ 番目の障害に対しては予測信頼度関数の上下限からシステムアベイラビリティの上下限を導出し，システムアベイラビリティを最大にする適応的な最適若化スケジュールについて議論する．またシミュレーション実験を通して提案手法の有効性を検証する．

キーワード：ソフトウェア信頼性，マンデルバグ，エージング関連バグ，ソフトウェアエージング，ソフトウェア若化，アベイラビリティ，ノンパラメトリック予測推論，ノンパラメトリック推定

1.　まえがき

コンピュータのシステムダウンによる障害はハードウェア障害よりもソフトウェア障害に起因することが多いことが知られており [1]，ソフトウェアシステムの信頼性を向上させ，これを維持していくことは非常に重要な課題である．ソフトウェアの開発段階においてソースコード上に作り込まれた欠陥はソフトウェアフォールトと呼ばれ，ソフトウェア障害を引き起こす原因となる．Gray [2] はソフトウェアフォールトをボーアバグ (Bohrgubs) とハイゼンバグ (Heisenbugs) に分類した．その後，Grottke and Trivedi [3] がボーアバグとマンデルバグ

[*1] 本学経営学部准教授
[*2] 広島大学大学院工学研究院教授

(Mandelbugs) に大別し，ハイゼンバグとエージング関連バグ (aging-related bugs) がマンデルバグに含まれると述べた．

Grottke and Trivedi [3] の定義によれば，ボーアバグは容易に除去が可能で，同じ環境下では再現が可能なフォールトである．ボーアバグは理論物理学者 Niels Bohr になぞらえて Gray [2] によって命名された．これに対してマンデルバグは原因が複雑かつ不明瞭であり，障害の再現やデバッグでの除去が困難なフォールトである．これはフラクタルの研究者 Benoît Mandelbrot になぞらえて名付けられた [4]．マンデルバグによる障害発生の複雑さには，(1) ソフトウェアシステムを実行するオペレーティングシステムやハードウェアといった他の環境に影響される，(2) フォールトが含まれるコードが実行されてから実際に障害に至るまでにタイムラグが存在する，という複雑さが存在する．

ソフトウェア障害が発生した後，システムをリスタートして障害が発生したコードを再実行すれば，今度は正常に動作し，障害を再現できないことがある．あるいは，ソフトウェア障害の発生後に原因究明のためデバッガを使用してシステムを起動しても，同じ障害を再現できないこともある．このように一時的に表面化するフォールトは，Gray [2] によってハイゼンバグと定義され，量子力学におけるハイゼンベルグの不確定性原理 (Heisenberg Uncertainty Principle) になぞらえて命名された．ハイゼンバグはソフトウェアシステムのユーザの環境で発生しても，開発者の環境では再現できないことがあったりするなど，その障害を必ずしも再現できないことからデバッグには困難が伴うことも多い．この意味で，ハイゼンバグはマンデルバグに含まれることになる．

また，アプリケーション内やシステム内部の動作環境において時間の経過とともにエラーが累積し，劣化が進むような現象をソフトウェアエージング (software aging) と呼び，この原因となるフォールトがエージング関連バグである．例えば，メモリリーク，ファイルロック解除漏れ，ストレージのフラグメンテーション，内部変数の丸め誤差，スレッドの暴走などが累積することで，徐々に劣化が進み，パフォーマンスの劣化やシステムのクラッシュといったシステム障害へ至ることになる．この場合，メモリリークなどを発生させるコードが実行されても直接システム障害に至るわけではなく，実際に障害が起こるまでにはタイムラグが存在し，この長さも確率的であるため，ソースコード上で障害の根本原因を特定することは困難である．よって，エージング関連バグもマンデルバグに含まれることになる．図１には４種類のフォールトの関係図を示す．

エージング関連バグはオペレーティングシステム，ミドルウェアシステム，サーバソフトウェアなどの数多くのシステムで観測されており，これに起因する障害も報告されている [1,5–18]．また，マンデルバグに包含されるエージング関連バグは，この根本原因をソフトウェアシステ

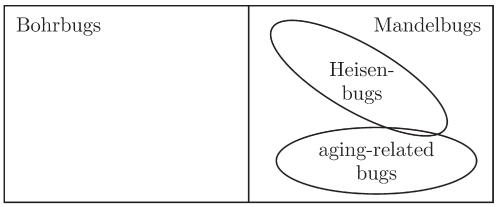

図 1　フォールトの関係図 [3]

ムのソースコード上で特定することが極めて困難であることから，その対処法について数多くの研究がなされてきた．特に，ソフトウェア若化 (software rejuvenation) と呼ばれる方策はエージング関連バグに起因する障害を予防するための有効な方法として認識されている [19]．ソフトウェア若化はソフトウェアシステムの稼働を一時的に停止し，その内部構造を浄化したあとにシステムを再稼働するという一連の予防保全手続きを意味する．ここで，ソフトウェアシステムの内部構造の浄化とは，ガーベジコレクションやオペレーティングシステムにおけるカーネルテーブルの洗浄，データ構造の初期化などを意味する．アプリケーションシステムの運用上，極端ではあるが最も頻繁に行われているソフトウェア若化の一例として，ハードウェアリブートが挙げられる．

Marshall [6] はパトリオットミサイル防衛システムによるスカッドミサイル迎撃失敗の原因がエージング関連バグであったことを報告した．この対処方法として，パトリオットシステムを 8 時間ごとに再起動するというソフトウェア若化手続きが採用されている．Grottke ら [1] や Matias and Filho [11] は Apache ウェブサーバソフトウェアのエージング現象について，Machida ら [16] は仮想マシンモニタにおけるエージング現象について報告した．また Yoshimura ら [17] は Linux OS のエージング現象を調査し，OS 全体を再起動しなくとも問題を引き起こしているプロセスを停止するだけで若化できる可能性があることを示した．また，2014 年 6 月に国内線 178 便の欠航を引き起こした日本航空のシステム障害は，「プログラムの潜在的な不具合により，不要なデータが滞留したのが原因だった」[18] ことから，これもエージング関連バグが原因であると推察される．

ここで問題となるのは，どのようなタイミングでシステムを予防的に若化するかにある．Huang ら [19] はソフトウェアシステムの時間的挙動を，正常稼働状態，障害発生可能な状態，障害の発生状態，ソフトウェア若化状態の 4 状態をもつ連続時間マルコフ連鎖で記述した．すなわちランダムなソフトウェア若化スケジュールのもとでシステムのアンアベイラビリティや定常状態における単位時間当たり運用費用を評価した．Dohi ら [20,21] は Huang ら [19] の連

続時間マルコフ連鎖モデルを連続時間および離散時間のセミマルコフモデルに拡張し，更に障害発生時間データから経験分布関数に基づいた標準総試験時間統計量を定義することにより最適な若化スケジュールを推定する統計的手法を開発している．Vaidyanathan ら [22] はシステム障害に至るまでの時間分布に２次アーラン分布を考えたセミマルコフモデルに対して，アベイラビリティとダウンタイムコストの観点から最適ソフトウェア若化スケジュールの導出を議論した．また，著者らは文献 [23] においてセミマルコフモデルで記述されたモデルに対して，文献 [24] においてはマルコフ再生過程で記述されたモデルに対して最適ソフトウェア若化スケジュールを核密度推定によって少数の障害発生時間データから高精度に推定するアルゴリズムを提案した．Zhao ら [25] はメモリリークの注入実験による加速寿命試験を行い，セミマルコフモデルに基づいた最適若化スケジュールを導出するためのパラメトリック／ノンパラメトリックモデルを提案した．

さらに，Pfening ら [26] はサーバ型のソフトウェアシステムに対し，サービスを継続するかソフトウェア若化を実施するかという意思決定をマルコフ決定過程としてモデル化し，そのアルゴリズムを提案した．Eto and Dohi [27–29] はサーバ型ソフトウェアについて多段階の劣化状態を考えたセミマルコフ決定過程モデルを提案し，管理限界型のソフトウェア若化方策が最適となることを示した．また Eto ら [30] は，同様の問題に対して Q 学習による強化学習アルゴリズムを提案し，適応的にコストを最小化する最適ソフトウェア若化スケジュールを導出した．Okamura and Dohi [31] はサーバ型ソフトウェアシステムにおけるトランザクションの到着にマルコフ到着過程を考え，マルコフ決定過程による最適なソフトウェア若化スケジュールを議論した．

一方でノンパラメトリック予測推論 (nonparametric predictive inference) [32] は Hill の仮定 $A_{(n)}$ [33, 34] に基づき，n 個のデータが観測された後の $n+1$ 番目の事象に対して予測を行う統計手法である．Hill の仮定 $A_{(n)}$ は対象としている事象に対して正確な確率を導くためには不十分であるが，それらの確率の上下限を与えることが可能である [35]．Coolen and Newby [36] は予防取替問題に対してノンパラメトリック予測推論を初めて用いた．Coolen and Coolen-Schrijner [37] は連続時間において状態監視可能な状況を想定した予防取替問題に対して同様のアプローチを適用している．Coolen and Yan [38] は打切りを含んだ故障データに対するノンパラメトリック予測推論を提案し，Coolen-Schrijner and Coolen [39] は n 個の故障データが得られた後，$n+1$ 番目の故障時間に対してノンパラメトリック予測推論を適用し，年齢取替問題を取り扱っている．更に Coolen-Schrijner and Coolen [40] は $n+1$ 番目及び $n+2$ 番目の故障時間を予測し最適予防取替年齢を適応的に求めている．また Coolen and Coolen-Maturi [41] は冗長化システムにおける複数の部品が同時に故障するような状況にノン

パラメトリック予測推論を適用している．

著者らは文献 [42] で文献 [23] と同様のセミマルコフモデルに対してノンパラメトリック予測推論を適用し，$n+1$ 番目及び $n+2$ 番目の障害時間を予測するとともに，適応的にシステムアベイラベイリティを最大化するソフトウェア若化スケジューリングを実施した．また文献 [43] では単位時間当たり運用費用を最小化する若化スケジュールを $n+2$ 番目までの障害時間のノンパラメトリック予測推論によって導出した．さらに，著者ら [44–46] は文献 [24] と同様のマルコフ再生過程によるモデルに対してもノンパラメトリック予測推論を適用した．文献 [44] では $n+1$ 番目の障害時間を予測し，文献 [45,46] では $n+1$ 番目，$n+2$ 番目の障害時間を予測し，システムアベイラビリティを最大化する若化スケジュールを導出した．

本論文では文献 [42] で提案された 2 ステップの適応的若化スケジューリングを 3 ステップの適応的若化スケジューリングへと拡張する．すなわち，n 個の障害発生時間データからノンパラメトリック予測推論によって $n+1$ 番目の障害時間を予測し，システムアベイラビリティの上下限を最大化する若化スケジュールを導出する．この予防若化スケジュールによって $n+1$ 番目の障害時間が打切られたならば，1 つの打切り時間を考慮して $n+2$ 番目の障害時間を予測し，$n+2$ 番目若化スケジュールを導出する．さらに $n+2$ 番目の障害時間が打切られたことによって得られた 2 つの打切り時間を考慮して $n+3$ 番目の障害時間をノンパラメトリック予測推論によって予測し，システムアベイラビリティの上下限を導出する．このシステムアベイラビリティを最大にするような最適若化スケジュールについて議論する．また，シミュレーション実験を通して本論文で提案する適応的 3 ステップ若化スケジュールの特徴について考察し，その有効性を検証する．

2.　セミマルコフモデル

本論文では，ソフトウェアシステムが以下の 4 状態で構成されるセミマルコフモデルについて考える．

状態 0：正常稼働状態

状態 1：劣化状態

状態 2：障害発生状態

状態 3：予防保全状態

ソフトウェアシステムは状態 0 の正常稼動状態からスタートし，メモリリーク量がある閾値を超えるなどして状態 1 の劣化状態に推移する．ここで状態 1 は正常稼動状態と比べて不安定でシステムが障害発生可能な状態である．状態 0 から状態 1 に推移する時間 Z の確

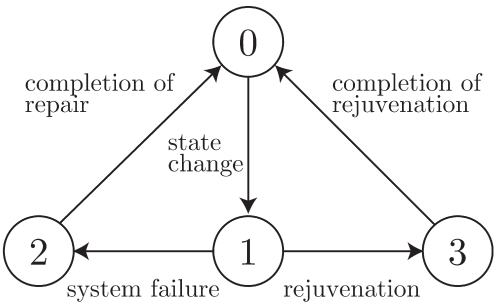

図２　モデルの状態推移図

率分布関数を $\Pr\{Z \leq t\} = F_0(t)$，平均を μ_0 (> 0) とする．システムが障害発生可能な状態から実際に障害が発生するまでの時間を非負で連続な確率変数 X とし，その確率分布関数を $\Pr\{X \leq t\} = F_f(t)$，信頼度関数を $S(t) = 1 - F_f(t)$，平均を λ_f (> 0) とする．一旦障害が発生すると，事後保全が直ちに開始される．事後保全とは，障害が発生した後に事後的にソフトウェアシステムを若化することを意味しており，本論文では修復という言葉によって代用する．修復に要する時間 Y も非負で連続な確率変数であり，その確率分布関数を $\Pr\{Y \leq t\} = F_a(t)$，平均を μ_a (> 0) とする．修復完了後は正常稼働状態である状態 0 へ戻る．

一方，システムが障害発生可能な状態に推移すると予防的なソフトウェア若化を行うかどうかの決定をするものとする．障害発生可能な状態から予防的な若化を行うまでの時間 T もまた非負で連続な確率変数であり，その確率分布関数を $F_r(t)$，平均を t_0 (> 0) とする．予防的若化にもオーバーヘッドが存在し，そのオーバーヘッド V に対する確率関数を $\Pr\{V \leq t\} = F_c(t)$，平均を μ_c $(0 < \mu_c < \mu_a)$ とする．修復の場合と同様に，予防的若化が完了すると，システムの障害発生率は正常稼働状態における初期状態まで復旧される．このモデルは状態 0 から再度状態 0 に戻るまでの時間間隔を周期にもつセミマルコフ過程であり，すべての状態が再生点 (regeneration points) となる．システムの状態推移図を図 2 に示す．特に障害発生可能な状態から予防的に若化を実施するまでの時間が t_0 の一定であるとすれば，確率分布関数 $F_r(t)$ を次のようなユニット関数

$$F_r(t) = U(t - t_0) = \begin{cases} 1 & \text{if } t \geq t_0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

に置き換えればよい．

Dohi ら [20] の結果を直接用いることによって，定常アベイラビリティは

$$A(t_0) = \Pr\{\text{software system is operative in the steady state}\}$$

$$= \frac{\mu_0 + \int_0^{t_0} S(t)dt}{\mu_0 + \mu_a[1 - S(t_0)] + \mu_c S(t_0) + \int_0^{t_0} S(t)dt} \tag{2}$$

のように求めることができる．以下では，式 (2) の $S(\cdot)$ に対して，理論分布を仮定することなくシステム障害発生時間データと打切り時間データからノンパラメトリック予測推論に基づいた予測を行い，式 (2) のシステムアベイラビリティを最大にする最適ソフトウェア若化スケジュールを適応的に導出することを考える．

3.　ノンパラメトリック予測推論

ノンパラメトリック予測推論は Coolen ら [32] によって提案された統計手法で，Hill [33, 34] の仮定 $A_{(n)}$ から導出されたものである．今，$X_j,\, j = 1, \cdots, n$ を交換可能な連続型確率変数とし，$x_{(j)}$ をその順序統計量とする．本論文でも文献 [40, 42] と同様に，Hill の仮定 [34] を導入する．

(A-1) タイは起こらない．つまり，$i \neq j$ に対して $x_{(i)} \neq x_{(j)}$.

(A-2) $x_{(j)},\ j = 1, \cdots, n$ が与えられるとき，次の実現値が区間 $I_j = (x_{(j)}, x_{(j+1)})$ となる確率は $1/(n+1), (j = 0, \cdots, n)$ である．ただし，$x_{(0)} = 0$ かつ $x_{(n+1)} = \infty$（あるいは X_j の上限が存在する）.

つまり，n 番目までの障害発生時間データ $(x_{(1)} < x_{(2)} < \cdots < x_{(n)})$ が観測されたとき，$n+1$ 番目の障害発生時間 X_{n+1} に対する予測確率は

$$\Pr\left\{X_{n+1} \in \left(x_{(j)},\ x_{(j+1)}\right)\right\} = \frac{1}{n+1}, \quad j = 0, \cdots, n \tag{3}$$

となる．

文献 [42] において X_{n+1} に対する最適ソフトウェア若化スケジュールは n 番目までの障害発生時間のいずれかと一致することが示された．以下では $n+1$ 番目の障害発生時間データの打切り時間を $x_{(k)},\, k = 1, \cdots, n$ と書くことにする．このとき $n+2$ 番目の障害発生時間 X_{n+2} に対する予測確率は，rc-$\left(A_{(n+1)}\right)$ の仮定 [38] を直接利用することにより

$$\Pr\left\{X_{n+2} \in \left(x_{(j)},\ x_{(j+1)}\right)\right\} = \begin{cases} \dfrac{1}{n+2}, & j = 0, \cdots, k-1, \\ \dfrac{n+2-k}{(n+2)(n+1-k)}, & j = k, \cdots, n+1 \end{cases} \tag{4}$$

となる．

さらに，X_{n+2} に対する最適ソフトウェア若化スケジュールは X_{n+1} に対するそれと一致することも示された．つまり，$x_{(k)}$ において２つの障害発生時間データが打切られることになる．したがって rc-$\left(A_{(n+1)}\right)$ の仮定より，$n+3$ 番目の障害発生時間 X_{n+3} に対する予測確率

$$\Pr\left\{X_{n+3} \in \left(x_{(j)},\ x_{(j+1)}\right)\right\} = \begin{cases} \dfrac{1}{n+3}, & j = 0, \cdots, k-1, \\ \dfrac{n+3-k}{(n+2)(n+1-k)}, & j = k, \cdots, n+1 \end{cases} \tag{5}$$

が得られる．

式 (3)〜(5) より，$X_{n+1}, X_{n+2}, X_{n+3}$ に対する予測信頼度関数はそれぞれ

$$S_{X_{n+1}}\left(x_{(j)}\right) = \frac{n-j+1}{n+1}, \quad j = 0, \cdots, n+1, \tag{6}$$

$$S_{X_{n+2}}\left(x_{(j)}\right) = \begin{cases} \dfrac{n+2-j}{n+2}, & j = 0, \cdots, k, \\ \dfrac{(n+2-k)\,(n+1-j)}{(n+2)\,(n+1-k)}, & j = k+1, \cdots, n+1, \end{cases} \tag{7}$$

$$S_{X_{n+3}}\left(x_{(j)}\right) = \begin{cases} \dfrac{n+3-j}{n+3}, & j = 0, \cdots, k, \\ \dfrac{(n+3-k)\,(n+1-j)}{(n+3)\,(n+1-k)}, & j = k+1, \cdots, n+1 \end{cases} \tag{8}$$

となる．

式 (6)〜(8) において，信頼度関数は以前に得られた順序標本点でのみ定義されているが，各標本点間においては信頼度関数の上下限を定義することができる．信頼度関数の下限を $\underline{S}(x)$ と書くと，

$$\underline{S}_{X_{n+1}}(x) = S_{X_{n+1}}\left(x_{(j+1)}\right) = \frac{n-j}{n+1}, \quad x \in (x_{(j)},\ x_{(j+1)}),\ j = 0, \cdots, n, \tag{9}$$

$$\begin{aligned} \underline{S}_{X_{n+2}}(x) &= S_{X_{n+2}}\left(x_{(j+1)}\right) \\ &= \begin{cases} \dfrac{n+1-j}{n+2}, & x \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = 0, \cdots, k-1, \\ \dfrac{(n+2-k)\,(n-j)}{(n+2)\,(n+1-k)}, & x \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = k, \cdots, n, \end{cases} \end{aligned} \tag{10}$$

$$\begin{aligned} \underline{S}_{X_{n+3}}(x) &= S_{X_{n+3}}\left(x_{(j+1)}\right) \\ &= \begin{cases} \dfrac{n+2-j}{n+3}, & x \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = 0, \cdots, k-1, \\ \dfrac{(n+3-k)\,(n-j)}{(n+3)\,(n+1-k)}, & x \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = k, \cdots, n \end{cases} \end{aligned} \tag{11}$$

となる．同様に信頼度関数の上限を $\overline{S}(x)$ とすると，

$$\overline{S}_{X_{n+1}}(x) = S_{X_{n+1}}\left(x_{(j)}\right) = \frac{n-j+1}{n+1}, \quad x \in (x_{(j)},\ x_{(j+1)}),\ j = 0, \cdots, n, \tag{12}$$

$$\overline{S}_{X_{n+2}}(x) = S_{X_{n+2}}\left(x_{(j)}\right) = \begin{cases} \dfrac{n+2-j}{n+2}, & x \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = 0, \cdots, k, \\ \dfrac{(n+2-k)(n+1-j)}{(n+2)(n+1-k)}, & x \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = k+1, \cdots, n, \end{cases} \tag{13}$$

$$\overline{S}_{X_{n+3}}(x) = S_{X_{n+3}}\left(x_{(j)}\right) = \begin{cases} \dfrac{n+3-j}{n+3}, & x \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = 0, \cdots, k, \\ \dfrac{(n+3-k)(n+1-j)}{(n+3)(n+1-k)}, & x \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = k+1, \cdots, n \end{cases} \tag{14}$$

が得られる．

例えば，$n = 4$ 個の障害発生時間データが得られている状況で，X_{n+1} に対するノンパラメトリック予測推論に基づく信頼度関数の上下限は図 3（左）のようになる．また，さらに $x_{(2)}$ で 2 個の障害発生時間の打切りデータが得られている状況での X_{n+3} に対する信頼度関数の上下限は図 3（右）のようになる．図 3 から障害発生時間の打切りを観測することによって，信頼度関数上下限の差は，打切り時間 $x_{(2)}$ 以前では小さくなり，打切り時間以降では大きくなることが確認できる．

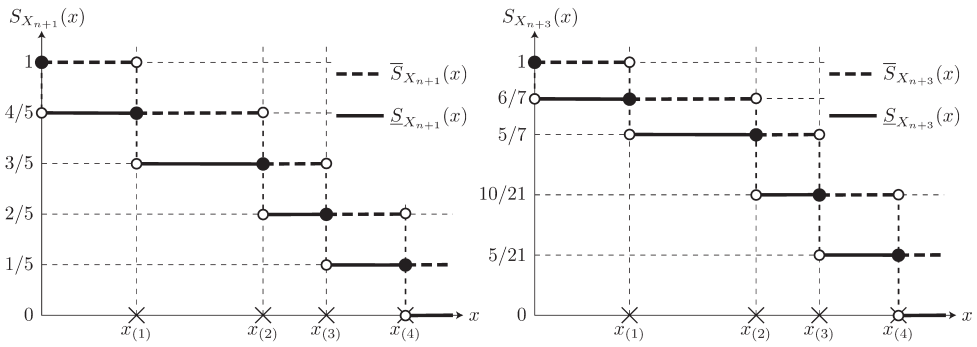

図 3　X_{n+1}（左）及び X_{n+3}（右）の予測信頼度関数

4.　適応的３ステップソフトウェア若化スケジューリング

文献 [42] では n 個のシステム障害発生データから $n+1$ 番目の障害に対してシステムアベイラビリティを最大にする最適ソフトウェア若化スケジュールをノンパラメトリック予測推論によって求め，さらに $n+1$ 番目の障害発生時刻データが，その予防若化によって打切られた場合に，n 個の障害発生データと１個の打切りデータから $n+2$ 番目の障害に対する最適若化スケジュールを求めた．本論文では $n+1$ および $n+2$ 番目の障害に対して得られたソフトウェア若化スケジュールを適用し，更に $n+3$ 番目の障害に対して適応的に最適ソフトウェア若化スケジュールを求めることを考える．

まず，$n+2$ 番目の障害に対して最適なソフトウェア若化時刻 T^*_{n+2} $(= T^*_{\mathrm{low},n+2})$ を設定したとき，T^*_{n+2} 以前に $n+2$ 番目の障害が発生したなら，新たに１つの障害発生時間データを得ることができる．すなわち障害を観測した場合，$n := n+1$ として，文献 [42] で得られた結果をそのまま利用すればよい．

一方で $n+2$ 番目のシステム障害に対する真の発生時刻が T^*_{n+2} 以降であった場合，つまり，T^*_{n+2} で障害が発生しなかった場合には，時刻 T^*_{n+2} でソフトウェア若化を予防的に実施するため，障害時間データは打切られることとなる．よって，時刻 T^*_{n+2} で打切られた打切り時間データも考慮し，$n+3$ 番目の障害時間を予測することで適応的にソフトウェア若化スケジュールを求めることを考える．なお，時刻 T^*_{n+2} でソフトウェア若化が予防的に実施された場合には，障害時間データには２個の打切り時間が含まれることとなる．ただし，文献 [42] で示された $T^*_{n+1} = T^*_{n+2} = x_{(k)}$ の結果から，それら２個の打切り時間は n 個の障害時間データのひとつと必ず一致する．なお，実際に打切られた時間を $x_{(k)}, k = 1, \cdots, n$ と書くことにする．すなわち，ここで利用可能な障害発生時間データは n 個の障害発生時間 $x_{(1)} < \cdots < x_{(n)}$ と打切り時間 $x_{(k)}$ であるが，$x_{(k)}$ において打切りが２回発生していることにも注意を要する．

以下では，式 (11)，(14) を利用してシステムアベイラビリティの上下限を導出する．更に $n+3$ 番目の障害発生時間 X_{n+3} に対して適応的なソフトウェア若化スケジュールを求めることを考える．

4.1　システムアベイラビリティの下限

ここでは，X_{n+3} に対してシステムアベイラビリティのノンパラメトリック予測推論に基づいた下限を式 (11) で定義される信頼度関数の下限を用いて導出し，このシステムアベイ

ラビリティの下限を最大にするようなソフトウェアの最適若化スケジュールの推定値が $x_{(j)}$, $j=1,\cdots,n$ のひとつとなることを示す.

X_{n+3} に対するシステムアベイラビリティの下限を $\underline{A}_{X_{n+3}}\left(x_{(j)}\right)$ で定義する. 式 (2) の $S(t_0)$ に式 (11) で定義される信頼度関数の下限を代入することで, システムアベイラビリティの下限に関する次の補助定理が得られる.

補助定理 4.1. X_{n+3} に対するシステムアベイラビリティの下限は式 (15)〜(18) によって与えられる.

$$\underline{A}_{X_{n+3}}\left(x_{(j)}\right)=\frac{(n+3)\mu_0+\sum_{l=1}^{j}x_{(l)}+(n+2-j)x_{(j)}}{(n+3)\mu_0+j\mu_a+(n+3-j)\mu_c+\sum_{l=1}^{j}x_{(l)}+(n+2-j)x_{(j)}},$$
$$t_0=x_{(j)},\ j=1,\cdots,k, \tag{15}$$

$$\underline{A}_{X_{n+3}}\left(x_{(j)}\right)=\frac{\Phi_{31}\left(x_{(j)}\right)}{\Psi_{31}\left(x_{(j)}\right)},\quad t_0=x_{(j)},\ j=k+1,\cdots,n+1, \tag{16}$$

$$\underline{A}_{X_{n+3}}\left(t_0\right)=\frac{(n+3)\,\mu_0+\sum_{l=1}^{j}x_{(l)}+(n+2-j)\,t_0}{(n+3)\,\mu_0+(1+j)\,\mu_a+(n+2-j)\,\mu_c+\sum_{l=1}^{j}x_{(l)}+(n+2-j)\,t_0},$$
$$t_0\in\left(x_{(j)},\ x_{(j+1)}\right),\ j=0,\cdots,k-1, \tag{17}$$

$$\underline{A}_{X_{n+3}}\left(t_0\right)=\frac{\Phi_{32}\left(t_0\right)}{\Psi_{32}\left(t_0\right)},\quad t_0\in\left(x_{(j)},\ x_{(j+1)}\right),\ j=k,\cdots,n. \tag{18}$$

ここで, $\Phi_{31}\left(x_{(j)}\right)$, $\Psi_{31}\left(x_{(j)}\right)$, $\Phi_{32}\left(t_0\right)$, $\Psi_{32}\left(t_0\right)$ はそれぞれ

$$\Phi_{31}\left(x_{(j)}\right)=(n+3)(n+1-k)\mu_0+(n+3-k)\sum_{l=1}^{j}x_{(l)}-2\sum_{l=1}^{k-1}x_{(l)}$$
$$+(n+3-k)(n-j)x_{(j)}, \tag{19}$$

$$\Psi_{31}\left(x_{(j)}\right)=(n+3)(n+1-k)\mu_0+(jn+3j-2k-jk)\mu_a+(n+3-k)(n+1-j)\mu_c$$
$$+(n+3-k)\sum_{l=1}^{j}x_{(l)}-2\sum_{l=1}^{k-1}x_{(l)}+(n+3-k)(n-j)x_{(j)}, \tag{20}$$

$$\Phi_{32}\left(t_0\right)=(n+3)(n+1-k)\mu_0+(n+3-k)\sum_{l=1}^{j}x_{(l)}-2\sum_{l=1}^{k-1}x_{(l)}$$
$$+(n+3-k)(n-j)t_0, \tag{21}$$

$$\Psi_{32}(t_0) = (n+3)(n+1-k)\mu_0 + (n+3-3k+jn+3j-jk)\mu_a + (n+3-k)(n-j)\mu_c$$
$$+ (n+3-k)\sum_{l=1}^{j} x_{(l)} - 2\sum_{l=1}^{k-1} x_{(l)} + (n+3-k)(n-j)t_0 \tag{22}$$

である．

証明 はじめに，$t_0 = x_{(j)},\ j = 1, \cdots, k$ とする．このとき

$$\begin{aligned}\int_0^{t_0} \underline{S}_{X_{n+3}}(x)dx &= \int_0^{x_{(j)}} \underline{S}_{X_{n+3}}(x)dx \\ &= \sum_{l=0}^{j-1} \frac{n+2-l}{n+3}\left(x_{(l+1)} - x_{(l)}\right) \\ &= \frac{1}{n+3}\left[(n+2-j)x_{(j)} + \sum_{l=1}^{j} x_{(l)}\right]\end{aligned} \tag{23}$$

となる．一方で，$t_0 = x_{(j)},\ j = k+1, \cdots, n+1$ とすると

$$\begin{aligned}&\int_0^{t_0} \underline{S}_{X_{n+3}}(x)dx \\ &= \int_0^{x_{(k)}} \underline{S}_{X_{n+3}}(x)dx + \int_{x_{(k)}}^{x_{(j)}} \underline{S}_{X_{n+3}}(x)dx \\ &= \frac{1}{n+3}\left[(n+2-j)x_{(j)} + \sum_{l=1}^{k} x_{(l)}\right] + \sum_{l=k}^{j-1} \frac{(n+3-k)(n-l)}{(n+3)(n+1-k)}\left(x_{(l+1)} - x_{(l)}\right) \\ &= \frac{1}{(n+3)(n+1-k)}\left[(n+3-k)\sum_{l=1}^{j} x_{(l)} - 2\sum_{l=1}^{k-1} x_{(l)} + (n+3-k)(n-j)x_{(j)}\right]\end{aligned} \tag{24}$$

が得られる．式 (23) および式 (24) を式 (2) にそれぞれ代入すれば，式 (15), (16) が得られる．

次に，$t_0 \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = 0, \cdots, k-1$ について

$$\begin{aligned}\int_0^{t_0} \underline{S}_{X_{n+3}}(x)dx &= \int_0^{x_{(j)}} \underline{S}_{X_{n+3}}(x)dx + \int_{x_{(j)}}^{t_0} \underline{S}_{X_{n+3}}(x)dx \\ &= \frac{1}{n+3}\left[(n+2-j)x_{(j)} + \sum_{l=1}^{j} x_{(l)}\right] + \frac{n+2-j}{n+3}\left(t_0 - x_{(j)}\right) \\ &= \frac{1}{n+3}\left[\sum_{l=1}^{j} x_{(l)} + (n+2-j)t_0\right]\end{aligned} \tag{25}$$

が得られ，同様に $t_0 \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = k, \cdots, n$ に対しては

$$\int_0^{t_0} \underline{S}_{X_{n+3}}(x)dx$$

$$
\begin{aligned}
&= \int_0^{x_{(k)}} \underline{S}_{X_{n+3}}(x)dx + \int_{x_{(k)}}^{x_{(j)}} \underline{S}_{X_{n+3}}(x)dx + \int_{x_{(j)}}^{t_0} \underline{S}_{X_{n+3}}(x)dx \\
&= \frac{1}{n+3}\left[(n+2-k)x_{(k)} + \sum_{l=1}^{k} x_{(l)}\right] \\
&\quad + \frac{n+3-k}{(n+3)(n+1-k)}\left[(n+1-j)x_{(j)} - (n-k)x_{(k)} + \sum_{l=k+1}^{j-1} x_{(l)}\right] \\
&\quad + \frac{(n+3-k)(n-j)}{(n+3)(n+1-k)}\left(t_0 - x_{(j)}\right) \\
&= \frac{1}{(n+3)(n+1-k)}\left[(n+3-k)\sum_{l=1}^{j} x_{(l)} - 2\sum_{l=1}^{k-1} x_{(l)} + (n+3-k)(n-j)t_0\right] \quad (26)
\end{aligned}
$$

が得られる．式 (25), (26) を式 (2) にそれぞれ代入すれば，式 (17), (18) が得られる． □

X_{n+3} に対してソフトウェア若化を実施しない場合，つまり $t_0 \to \infty$ のとき，ノンパラメトリック予測推論によるシステムアベイラビリティの下限は，

$$
\underline{A}_{X_{n+3}}\left(x_{(n+1)}\right) = \frac{\mu_0 + \frac{1}{(n+3)(n+1-k)}\left[(n+3-k)\sum_{l=1}^{n} x_{(l)} - 2\sum_{l=1}^{k-1} x_{(l)}\right]}{\mu_0 + \mu_a + \frac{1}{(n+3)(n+1-k)}\left[(n+3-k)\sum_{l=1}^{n} x_{(l)} - 2\sum_{l=1}^{k-1} x_{(l)}\right]} \quad (27)
$$

となる．式 (11) より，$t_0 > x_{(n)}$ を満たす t_0 に対して，$\underline{S}_{X_{n+3}}(t_0) = 0$ となる．したがって，$t_0 > x_{(n)}$ に対し，システムアベイラビリティの下限は一定である．式 (16), (18) より，$t_0 > x_{(n)}$ において $\underline{S}_{X_{n+3}}(t_0) < \underline{S}_{X_{n+3}}\left(x_{(n)}\right)$ であることから，$t_0 > x_{(n)}$ が最適なソフトウェア若化スケジュールにはなり得ない．

補助定理 4.2. $t_0 \in \left(x_{(j)},\ x_{(j+1)}\right)$, $j = 0, \cdots, n-1$ に対して $\underline{A}_{X_{n+3}}(\cdot)$ は連続かつ単調増加関数である．更に，$\underline{A}_{X_{n+3}}(\cdot)$ は $x_{(j)}$, $j = 1, \cdots, n$ において左連続であり，$x_{(j)}$ が局所解となる．

証明 $t_0 \in \left(x_{(j)},\ x_{(j+1)}\right)$, $j = 0, \cdots, n-1$ において $\underline{S}_{X_{n+3}}(\cdot)$ は連続であるので，$\underline{A}_{X_{n+3}}(\cdot)$ も連続関数である．また明らかに，$\underline{A}_{X_{n+3}}(\cdot)$ は $t_0 \in \left(x_{(j)},\ x_{(j+1)}\right)$, $j = 0, \cdots, n-1$ において単調増加関数である．さらに，$j = 0, \cdots, k-1$ に対して，

$$
\begin{aligned}
&\lim_{\epsilon \to 0} \underline{A}_{X_{n+3}}\left(x_{(j)} + \epsilon\right) \\
&= \frac{(n+3)\mu_0 + \sum_{l=1}^{j} x_{(l)} + (n+2-j)x_{(j)}}{(n+3)\mu_0 + (1+j)\mu_a + (n+2-j)\mu_c + \sum_{l=1}^{j} x_{(j)} + (n+2-j)x_{(j)}} \\
&< \underline{A}_{X_{n+3}}\left(x_{(j)}\right), \quad (28)
\end{aligned}
$$

$j=k,\cdots,n$ に対して，

$$\lim_{\epsilon\to 0}\underline{A}_{X_{n+3}}\left(x_{(j)}+\epsilon\right)=\frac{\lim_{\epsilon\to 0}\Phi_{32}\left(x_{(j)}+\epsilon\right)}{\lim_{\epsilon\to 0}\Psi_{32}\left(x_{(j)}+\epsilon\right)}<\underline{A}_{X_{n+3}}\left(x_{(j)}\right) \tag{29}$$

となる．ただし，

$$\begin{aligned}\lim_{\epsilon\to 0}\Phi_{32}\left(x_{(j)}+\epsilon\right)&=(n+3)(n+1-k)\mu_0+(n+3-k)\sum_{l=1}^{j}x_{(l)}-2\sum_{l=1}^{k-1}x_{(l)}\\&\quad+(n+3-k)(n-j)x_{(j)},\end{aligned} \tag{30}$$

$$\begin{aligned}&\lim_{\epsilon\to 0}\Psi_{32}\left(x_{(j)}+\epsilon\right)\\&=(n+3)(n+1-k)\mu_0+(n+3-3k+jn+3j-jk)\mu_a+(n+3-k)(n-j)\mu_c\\&\quad+(n+3-k)\sum_{l=1}^{j}x_{(l)}-2\sum_{l=1}^{k-1}x_{(l)}+(n+3-k)(n-j)x_{(j)}\end{aligned} \tag{31}$$

である．一方で

$$\lim_{\epsilon\to 0}\underline{A}_{X_{n+3}}\left(x_{(j)}-\epsilon\right)=\underline{A}_{X_{n+3}}\left(x_{(j)}\right) \tag{32}$$

であるので，$\underline{A}_{X_{n+3}}(\cdot)$ は $x_{(j)}, j=1,\cdots,n$ において左連続であり，$x_{(j)}$ が局所解となる．□

補助定理 4.2. より，次の定理が得られる．

定理 4.3. システムアベイラビリティの下限 $\underline{A}_{X_{n+3}}(\cdot)$ の最大値は $x_{(j)}, j=1,\cdots,n$ のいずれかの点で与えられる．

4.2　システムアベイラビリティの上限

次に，X_{n+3} に対してシステムアベイラビリティのノンパラメトリック予測推論による上限を式 (14) で与えられる信頼度関数の上限を用いて導出する．また．このシステムアベイラビリティの上限を最大にするソフトウェア最適若化スケジュールの推定値が $x^{-}_{(j)}, j=1,\cdots,n$ となることを示す．ただし，$x^{-}_{(j)}$ は $x_{(j)}$ の直前を意味する．

今，X_{n+3} に対してシステムアベイラビリティのノンパラメトリック予測推論に基づいた上限を $\overline{A}_{X_{n+3}}(\cdot)$ と書くこととする．式 (2) に式 (14) を代入することで次の補助定理を得る．

補助定理 4.4. X_{n+3} に対するシステムアベイラビリティの上限は式 (33)〜(37) によって与えられる．

$$\overline{A}_{X_{n+3}}\left(x_{(j)}\right)=\frac{(n+3)\,\mu_0+(n+3-j)\,x_{(j)}+\sum_{l=1}^{j}x_{(l)}}{(n+3)\,\mu_0+j\mu_a+(n+3-j)\,\mu_c+(n+3-j)\,x_{(j)}+\sum_{l=1}^{j}x_{(l)}},$$

$$t_0 = x_{(j)},\ j = 1, \cdots, k, \tag{33}$$

$$\overline{A}_{X_{n+3}}\left(x_{(k+1)}\right) = \frac{\Phi_{33}\left(x_{(k+1)}\right)}{\Psi_{33}\left(x_{(k+1)}\right)}, \quad t_0 = x_{(k+1)}, \tag{34}$$

$$\overline{A}_{X_{n+3}}\left(x_{(j)}\right) = \frac{\Phi_{34}\left(x_{(j)}\right)}{\Psi_{34}\left(x_{(j)}\right)}, \quad t_0 = x_{(j)},\ j = k+2, \cdots, n+1, \tag{35}$$

$$\overline{A}_{X_{n+3}}\left(t_0\right) = \frac{\Phi_{35}\left(t_0\right)}{\Psi_{35}\left(t_0\right)}, \quad t_0 \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = 0, \cdots, k, \tag{36}$$

$$\overline{A}_{X_{n+3}}\left(t_0\right) = \frac{\Phi_{36}\left(t_0\right)}{\Psi_{36}\left(t_0\right)}, \quad t_0 \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = k+1, \cdots, n. \tag{37}$$

ただし，

$$\Phi_{33}\left(x_{(k+1)}\right) = (n+3)(n+1-k)\mu_0 + (n+1-k)\sum_{l=1}^{k+1} x_{(l)} + (n+1-k)(n+2-k)x_{(k+1)}, \tag{38}$$

$$\begin{aligned}\Psi_{33}\left(x_{(k+1)}\right) = {} & (n+3)(n+1-k)\mu_0 + \left(n+3+kn-k^2\right)\mu_a + (n+3-k)(n-k)\mu_c \\ & + (n+1-k)\sum_{l=1}^{k+1} x_{(l)} + (n+1-k)(n+2-k)x_{(k+1)},\end{aligned} \tag{39}$$

$$\begin{aligned}\Phi_{34}\left(x_{(j)}\right) = {} & (n+3)(n+1-k)\mu_0 + (n+3-k)\sum_{l=1}^{j} x_{(l)} - 2\sum_{l=1}^{k} x_{(l)} \\ & + (n+3-k)(n+1-j)x_{(j)},\end{aligned} \tag{40}$$

$$\begin{aligned}\Psi_{34}\left(x_{(j)}\right) = {} & (n+3)(n+1-k)\mu_0 + (jn+3j-2k-jk)\mu_a + (n+3-k)(n+1-j)\mu_c \\ & + (n+3-k)\sum_{l=1}^{j} x_{(l)} - 2\sum_{l=1}^{k} x_{(l)} + (n+3-k)(n+1-j)x_{(j)},\end{aligned} \tag{41}$$

$$\Phi_{35}\left(t_0\right) = (n+3)\mu_0 + \sum_{l=1}^{j} x_{(l)} + (n+3-j)t_0, \tag{42}$$

$$\Psi_{35}\left(t_0\right) = (n+3)\mu_0 + j\mu_a + (n+3-j)\mu_c + \sum_{l=1}^{j} x_{(l)} + (n+3-j)t_0, \tag{43}$$

$$\Phi_{36}(t_0) = (n+3)(n+1-k)\mu_0 + (n+3-k)\sum_{l=1}^{j} x_{(l)} - 2\sum_{l=1}^{k} x_{(l)} + (n+3-k)(n+1-j)t_0, \tag{44}$$

$$\Psi_{36}(t_0) = (n+3)(n+1-k)\mu_0 + (jn+3j-2k-jk)\mu_a + (n+3-k)(n+1-j)\mu_c + (n+3-k)\sum_{l=1}^{j} x_{(l)} - 2\sum_{l=1}^{k} x_{(l)} + (n+3-k)(n+1-j)t_0. \tag{45}$$

証明 はじめに $t_0 = x_{(j)},\ j = 1, \cdots, k+1$ とする．このとき

$$\begin{aligned}\int_0^{t_0} \overline{S}_{X_{n+3}}(x)dx &= \int_0^{x_{(j)}} \overline{S}_{X_{n+3}}(x)dx \\ &= \sum_{l=0}^{j-1} \frac{n+3-l}{n+3}\left(x_{(l+1)} - x_{(l)}\right) \\ &= \frac{1}{n+3}\left[(n+3-j)x_{(j)} + \sum_{l=1}^{j} x_{(l)}\right]\end{aligned} \tag{46}$$

となる．一方で，$t_0 = x_{(j)},\ j = k+2, \cdots, n$ とすると

$$\begin{aligned}&\int_0^{t_0} \overline{S}_{X_{n+3}}(x)dx \\ &= \int_0^{x_{(k+1)}} \overline{S}_{X_{n+3}}(x)dx + \int_{x_{(k+1)}}^{x_{(j)}} \overline{S}_{X_{n+3}}(x)dx \\ &= \frac{1}{n+3}\left\{[n+3-(k+1)]x_{(k+1)} + \sum_{l=1}^{k+1} x_{(l)}\right\} + \sum_{l=k+1}^{j-1} \frac{(n+3-k)(n+1-l)}{(n+3)(n+1-k)}\left(x_{(l+1)} - x_{(l)}\right) \\ &= \frac{1}{(n+3)(n+1-k)}\left[(n+3-k)\sum_{l=1}^{j} x_{(l)} - 2\sum_{l=1}^{k} x_{(l)} + (n+3-k)(n+1-j)x_{(j)}\right]\end{aligned} \tag{47}$$

が得られる．式 (8), (46) を式 (2) に代入すれば式 (33), (34) が得られる．さらに，式 (2), (8), (47) より式 (35) が得られる．

次に $t_0 \in \left(x_{(j)},\ x_{(j+1)}\right),\ j = 0, \cdots, k$ について

$$\begin{aligned}\int_0^{t_0} \overline{S}_{X_{n+3}}(x)dx &= \int_0^{x_{(j)}} \overline{S}_{X_{n+3}}(x)dx + \int_{x_{(j)}}^{t_0} \overline{S}_{X_{n+3}}(x)dx \\ &= \frac{1}{n+3}\left[(n+3-j)x_{(j)} + \sum_{l=1}^{j} x_{(l)}\right] + \frac{n+3-j}{n+3}\left(t_0 - x_{(j)}\right) \\ &= \frac{1}{n+3}\left[\sum_{l=1}^{j} x_{(l)} + (n+3-j)t_0\right]\end{aligned} \tag{48}$$

が得られ，同様に $t_0 \in \left(x_{(j)},\ x_{(j+1)}\right)$, $j = k+1, \cdots, n$ については

$$
\begin{aligned}
&\int_0^{t_0} \overline{S}_{X_{n+3}}(x)dx \\
&= \int_0^{x_{(j)}} \overline{S}_{X_{n+3}}(x)dx + \int_{x_{(j)}}^{t_0} \overline{S}_{X_{n+3}}(x)dx \\
&= \frac{1}{(n+3)(n+1-k)} \left[(n+3-k) \sum_{l=1}^{j} x_{(l)} - 2 \sum_{l=1}^{k} x_{(l)} + (n+3-k)(n+1-j) x_{(j)} \right] \\
&\quad + \frac{(n+3-k)(n+1-j)}{(n+3)(n+1-k)} \left(t_0 - x_{(j)} \right) \\
&= \frac{1}{(n+3)(n+1-k)} \left[(n+3-k) \sum_{l=1}^{j} x_{(l)} - 2 \sum_{l=1}^{k} x_{(l)} + (n+3-k)(n+1-j) t_0 \right] \quad (49)
\end{aligned}
$$

が得られる．式 (8), (48) を式 (2) に代入すれば，式 (36) が得られ，式 (8), (49) を代入すれば，式 (37) が得られる． □

X_{n+3} に対してソフトウェア若化を行わない場合，ノンパラメトリック予測推論に基づいたシステムアベイラビリティの上限は，$x_{(n+1)} = \infty$ であることから

$$
\overline{A}_{X_{n+3}}\left(x_{(n+1)}\right) = \frac{\Phi_{34}\left(x_{(n+1)}\right)}{\Psi_{34}\left(x_{(n+1)}\right)} = 1 \quad (50)
$$

となる．ただし，

$$
\begin{aligned}
\Phi_{34}\left(x_{(n+1)}\right) &= \Psi_{34}\left(x_{(n+1)}\right) \\
&= (n+3)(n+1-k)\mu_0 + \left(n^2 + 4n - nk - 3k + 3\right)\mu_a \\
&\quad + (n+3-k) \sum_{l=1}^{n+1} x_{(l)} - 2 \sum_{l=1}^{k} x_{(l)} \quad (51)
\end{aligned}
$$

である．したがって，以下ではまず $(0,\ x_{(n)}]$ の区間について考える．

補助定理 4.5. $t_0 \in \left(x_{(j)},\ x_{(j+1)}\right)$, $j = 0, \cdots, n$ に対して，$\overline{A}_{X_{n+3}}(\cdot)$ は連続かつ単調増加関数である．更に $\overline{A}_{X_{n+3}}(\cdot)$ は $x_{(j)}$, $j = 1, \cdots, n$ において右連続であり，$x^{-}_{(j)}$ が局所解となる．

証明 $\overline{A}_{X_{n+3}}(\cdot)$ が $t_0 \in \left(x_{(j)},\ x_{(j+1)}\right)$ に関して単調増加であることの証明は補助定理 4.1. の証明と同様である．一方で，$j = 0, \cdots, n-1$ に対して

$$
\lim_{\epsilon \to 0} \overline{A}_{X_{n+3}}\left(x_{(j)} + \epsilon\right) = \overline{A}_{X_{n+3}}\left(x_{(j)}\right) \quad (52)
$$

である．また，$j=1,\cdots,k$ に対して

$$\begin{aligned}&\lim_{\epsilon\to 0}\overline{A}_{X_{n+3}}\left(x_{(j)}-\epsilon\right)\\&=\frac{(n+3)\mu_0+\sum_{l=1}^{j-1}x_{(l)}+(n+4-j)x_{(j)}}{(n+3)\mu_0+(j-1)\mu_a+(n+4-j)\mu_c+\sum_{l=1}^{j-1}x_{(l)}+(n+4-j)x_{(j)}}\\&>\overline{A}_{X_{n+3}}(x_{(j)}),\end{aligned}\tag{53}$$

$j=k+1,\cdots,n$ に対しても

$$\begin{aligned}\lim_{\epsilon\to 0}\overline{A}_{X_{n+3}}\left(x_{(j)}-\epsilon\right)&=\frac{\lim_{\epsilon\to 0}\Phi_{36}\left(x_{(j)}-\epsilon\right)}{\lim_{\epsilon\to 0}\Psi_{36}\left(x_{(j)}-\epsilon\right)}\\&>\overline{A}_{X_{n+3}}\left(x_{(j)}\right).\end{aligned}\tag{54}$$

ただし，

$$\begin{aligned}\lim_{\epsilon\to 0}\Phi_{36}\left(x_{(j)}-\epsilon\right)&=(n+3)(n+1-k)\mu_0+(n+3-k)\sum_{l=1}^{j-1}x_{(l)}-2\sum_{l=1}^{k}x_{(l)}\\&\quad+(n+3-k)(n+2-j)x_{(j)},\end{aligned}\tag{55}$$

$$\begin{aligned}\lim_{\epsilon\to 0}\Psi_{36}\left(x_{(j)}-\epsilon\right)&=(n+3)(n+1-k)\mu_0+(jn-n+3j-3-k-jk)\mu_a\\&\quad+(n+3-k)(n+2-j)\mu_c+(n+3-k)\sum_{l=1}^{j-1}x_{(l)}\\&\quad-2\sum_{l=1}^{k}x_{(l)}+(n+3-k)(n+2-j)x_{(j)}\end{aligned}\tag{56}$$

である．したがって，$\overline{A}_{X_{n+3}}(\cdot)$ は $x_{(j)}, j=1,\cdots,n$ において右連続であり，$x^-_{(j)}, j=1,\cdots,n$ が局所解となる．□

補助定理 4.5. より，次の定理が得られる．

定理 4.6. $(0,\ x_{(n)}]$ におけるシステムアベイラビリティの上限 $\overline{A}_{X_{n+3}}(\cdot)$ の最大値は $x^-_{(j)}$, $j=1,\cdots,n$ のいずれかの点で与えられる．

次に，領域を $(0,\ \infty]$ に拡大して考える．もし，$x_{(n+1)}$ の上限 r が既知であると仮定することができるのであれば，定理 4.6. で与えられる $x^-_{(j^*)}$ における最大アベイラビリティ $\overline{A}_{X_{n+3}}(x^-_{(j^*)})$ と $\overline{A}_{X_{n+3}}(r^-)$ を比較すればよい．つまり，$\overline{A}_{X_{n+3}}(x^-_{(j^*)})\geq\overline{A}_{X_{n+3}}(r^-)$ なら $x^-_{(j^*)}$ でソフトウェア若化を行うことが最適となり，$\overline{A}_{X_{n+3}}(x^-_{(j^*)})<\overline{A}_{X_{n+3}}(r^-)$ ならソフトウェア若化を行

わないことが最適となる．一方で，r が未知である場合，$\overline{A}_{X_{n+3}}(x^{-}_{(j^*)}) = \overline{A}_{X_{n+3}}(r^{-})$ を満たす r を臨界値 r^* とし，X_{n+3} の上限として想定可能な r について $r \leq r^*$ なら定理 4.6. で得られた $x^{-}_{(j^*)}$ でソフトウェア若化を実施し，$r > r^*$ ならソフトウェア若化を実施しなければよい．

4.3　数値例

ここでは本論文で提案した適応的 3 ステップ若化方策の単純な数値例により，最適ソフトウェア若化スケジュールと最大システムアベイラビリティの振る舞いを示す．今，3 ステップまで $(s = 1, 2, 3)$ の各若化方策に対して，$T^{*}_{\mathrm{low},n+s} = \mathrm{argmax}\underline{A}_{X_{n+s}}(t_0)$, $T^{-*}_{\mathrm{up},n+s} = \mathrm{argmax}\overline{A}_{X_{n+s}}(t_0)$ と書くことにする．障害発生時間分布は未知であるが，$n = 8$ 個の障害発生時間データの順序標本 (825, 1127, 1598, 2195, 2574, 3737, 4589, 5054) が得られているものとする．その他のパラメータは $\mu_0 = 240.00$，$\mu_a = 0.50$，$\mu_c = 0.16$ と設定する．表 1 には障害発生時間とシステムアベイラビリティの上下限 $\overline{A}_{X_{n+s}}(x^{-}_{(j)})$, $\underline{A}_{X_{n+s}}(x_{(j)})$ を示す．

まず，X_{n+1} に対して，システムアベイラビリティの上下限をそれぞれ最大にするソフトウェア若化スケジュールは $T^{-*}_{\mathrm{up},n+1} = T^{*}_{\mathrm{low},n+1} = x_{(6)} = 3737$ となり，最大システムアベイラビリティは $\overline{A}_{X_{n+1}}(3737^{-}) = 0.999877$, $\underline{A}_{X_{n+1}}(3737) = 0.999840$ と推定される．なお，システムアベイラビリティの上限に関する臨界値は $r^* = 9827.8$ となる．よって，X_{n+1} に対する最適ソフトウェア若化スケジュールは $T^{*}_{n+1} = 3737$ となる．

次に，$n+1$ 番目の（真の）障害発生時間は 3737 より大きく，ソフトウェア若化が実施されたために，$n+1$ 番目の障害発生時間は $T^{*}_{n+1} = 3737$ において打切られたとする．表 1 より，X_{n+2} に対してシステムアベイラビリティの上下限を最大にするソフトウェア若化スケジュールはそれぞれ $T^{-*}_{\mathrm{up},n+2} = x_{(6)} = 4589$ および $T^{*}_{\mathrm{low},n+2} = x_{(7)} = 3737$ となり，最大システムアベイラビリティは $\overline{A}_{X_{n+2}}(4589^{-}) = 0.999889$, $\underline{A}_{X_{n+2}}(3737) = 0.999858$ と推定される．なお，**5.** のシミュレーション実験において示すが，$T^{-*}_{\mathrm{up},n+s}$ と $T^{*}_{\mathrm{low},n+s}$ は多くの場合において一致する．異なった場合も，理論的な最大システムアベイラビリティとの絶対誤差平均の観点から，システムアベイラビリティの下限を最大にする $T^{*}_{\mathrm{low},n+s}$ を最適なソフトウェア若化スケジュールとして採用すればよい．したがって，X_{n+2} に対する最適ソフトウェア若化スケジュールも $T^{*}_{n+2} = 3737$ となる．

さらに，X_{n+2} についてもソフトウェア若化が実施され，$n+2$ 番目の障害発生時間も $T^{*}_{n+2} = 3737$ で打切られたとする．図 4 には X_{n+3} に対するシステムアベイラビリティ上下限の振る舞いを示す．図 4 および表 1 に示すとおり，X_{n+3} に対してシステムアベイラビリティの上下限を最大にする若化スケジュールもまた $T^{-*}_{\mathrm{up},n+3} = x_{(6)} = 4589$ および

表１　８個の障害発生時間データ及びシステムアベイラビリティの上下限

	$x_{(1)}$	$x_{(2)}$	$x_{(3)}$	$x_{(4)}$	$x_{(5)}$	$x_{(6)}$	$x_{(7)}$	$x_{(8)}$
$x_{(j)}$	825	1127	1598	2195	2574	3737	4589	5054
$\overline{A}_{X_{n+1}}(x_{(j)}^{-})$	0.999850	0.999852	0.999861	0.999870	0.999865	0.999877	0.999876	0.999868
$\underline{A}_{X_{n+1}}(x_{(j)})$	0.999797	0.999805	0.999820	0.999832	0.999828	0.999840	0.999837	0.999826
$\overline{A}_{X_{n+2}}(x_{(j)}^{-})$	0.999850	0.999855	0.999867	0.999877	0.999875	0.999888	0.999889	0.999880
$\underline{A}_{X_{n+2}}(x_{(j)})$	0.999803	0.999814	0.999831	0.999845	0.999843	0.999858	0.999854	0.999841
$\overline{A}_{X_{n+3}}(x_{(j)}^{-})$	0.999850	0.999858	0.999871	0.999883	0.999882	0.999896	0.999899	0.999889
$\underline{A}_{X_{n+3}}(x_{(j)})$	0.999807	0.999821	0.999840	0.999855	0.999855	0.999872	0.999866	0.999852

$T^{*}_{\mathrm{low},n+3} = x_{(7)} = 3737$ となった．また，最大アベイラビリティは $\overline{A}_{X_{n+3}}(4589^{-}) = 0.999899$, $\underline{A}_{X_{n+3}}(3737) = 0.999872$ と推定され，X_{n+2} に対するそれよりも上昇していることが確認できる．後のシミュレーション実験の結果から，障害発生時間データが打切られた場合には，X_{n+2} および X_{n+3} に対してシステムアベイラビリティの下限を考えたソフトウェア若化スケジュールは，X_{n+1} の最適若化スケジュールから変化しないことが認められた．また，理論的な最大アベイラビリティとの絶対誤差平均の観点から，X_{n+1} の場合と同様に，X_{n+2}, X_{n+3} に対してもシステムアベイラビリティの下限を最大にする $T^{*}_{\mathrm{low},n+s}$ を最適なソフトウェア若化スケジュールとして採用すれば良い．

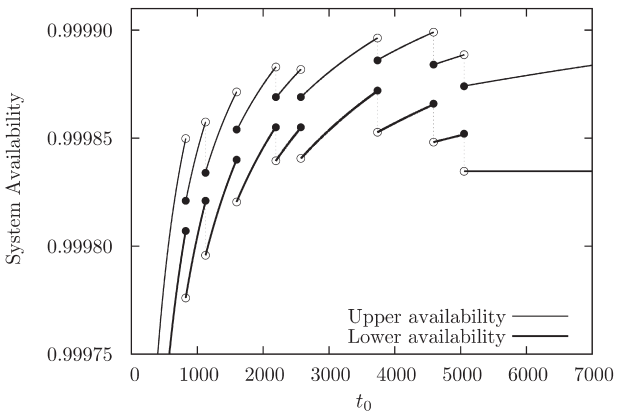

図４　X_{n+3} に対するシステムアベイラビリティ上下限の振る舞い

表 2　ケース設定及び理論的な最適ソフトウェア若化スケジュールと最大アベイラビリティ

Case	γ	θ	λ_f	SD[X]	μ_0	μ_a	μ_c	t_0^*	$A(t_0^*)$
Case I	1.5	2215.46	2000.00	1357.93	240.00	0.50	0.16	1816.62	0.999792
Case II	2.0	2256.76	2000.00	1045.45	240.00	0.50	0.16	1359.67	0.999819
Case III	4.0	2206.53	2000.00	561.09	240.00	0.50	0.16	1316.40	0.999869

5.　シミュレーション実験

ここでは，本論文で提案したノンパラメトリック予測推論に基づいた X_{n+1}, X_{n+2} および X_{n+3} に対する適応的ソフトウェア若化スケジュールの推定に関するシミュレーション実験を通して，提案手法で得られた適応的予防若化スケジュールの推定値と理論的な最適解との比較を行い提案手法の有効性を検証する．以下では，障害発生時間 X がワイブル分布

$$F_f(t) = 1 - e^{-(t/\theta)^\gamma}, \quad \gamma > 0,\ \theta > 0 \tag{57}$$

に従うものと仮定する．ここで，γ，θ はそれぞれ，形状パラメータ，尺度パラメータである．表 2 には 3 種類のケース設定及び障害発生時間分布が既知である場合の理論的な最適ソフトウェア若化スケジュールと最大アベイラビリティを示す．表 2 から，各ケースにおいて平均障害発生時間は同じであるが，γ が大きくなるに従って，障害発生時間の標準偏差 SD[X] は小さくなり，同時に最適なソフトウェア若化スケジュールは短く，最大アベイラビリティは大きくなることが確認できる．

表 3 および 4 には各ケースの $n = 10$, $n = 100$, $n = 200$ についてそれぞれ 10,000 回のシミュレーション実験によって得られた下限，上限の結果を示す．X_{n+1}, X_{n+2}, X_{n+3} に対して，下限，上限ともに最適若化スケジュールの平均値は全ての場合で理論的な最適若化スケジュール t_0^* よりも大きくなっていることが確認できる．また中央値についてもほぼ全てで大きく推定さている．なお，例外は下限 Case I, $n = 200$ の $T^*_{\mathrm{low,n+2}}$, $T^*_{\mathrm{low,n+3}}$ および上限 Case I, $n = 200$ の $T^*_{\mathrm{up,n+3}}$ のみである．得られた障害時間データ数が大きいほど，すなわち n が大きいほど平均値と中央値は漸近的に理論的な最適解へ近づいていくとともに，推定値の標準誤差も小さくなっていく様子が確認できる．また，表 3, 4 からはワイブル分布の形状パラメータが大きくなるほど，つまり障害時間発生時間の標準偏差が小さくなるほど，最適若化スケジュールの推定値の標準偏差も小さくなることがわかる．

表３　シミュレーション実験結果（下限）

	$T^*_{\mathrm{low},n+1}$	$A_{\mathrm{low},n+1}$	$T^*_{\mathrm{low},n+2}$	$A_{\mathrm{low},n+2}$	$T^*_{\mathrm{low},n+3}$	$A_{\mathrm{low},n+3}$
Case I : $n = 10$ $(t_0^* = 1816.62,\ A(t_0^*) = 0.999792)$						
Mean	2376.1	0.99979650	2240.2	0.99979694	2183.6	0.99980157
Median	1977.6	0.99980200	1877.0	0.99980200	1825.3	0.99980600
SD	1448.8	4.696E-05	1373.4	4.523E-05	1355.9	4.425E-05
Case I : $n = 100$						
Mean	1957.3	0.99979774	1941.7	0.99979772	1934.6	0.99979837
Median	1835.4	0.99979800	1822.4	0.99979800	1817.1	0.99979900
SD	672.2	1.553E-05	664.1	1.546E-05	657.6	1.542E-05
Case I : $n = 200$						
Mean	1901.9	0.99979603	1892.1	0.99979601	1889.2	0.99979636
Median	1822.4	0.99979600	1815.8	0.99979600	1812.6	0.99979700
SD	510.7	1.111E-05	506.2	1.108E-05	503.8	1.107E-05
Case II : $n = 10$ $(t_0^* = 1359.67,\ A(t_0^*) = 0.999819)$						
Mean	1723.6	0.99981793	1642.3	0.99981886	1603.7	0.99982343
Median	1590.8	0.99982200	1517.8	0.99982300	1494.0	0.99982700
SD	707.1	3.746E-05	671.9	3.593E-05	637.6	3.506E-05
Case II : $n = 100$						
Mean	1432.0	0.99982289	1423.4	0.99982288	1420.2	0.99982344
Median	1404.4	0.99982300	1397.6	0.99982300	1393.7	0.99982400
SD	295.3	1.280E-05	292.4	1.275E-05	291.3	1.267E-05
Case II : $n = 200$						
Mean	1400.6	0.99982185	1396.4	0.99982184	1395.5	0.99982211
Median	1382.7	0.99982200	1377.4	0.99982200	1376.8	0.99982200
SD	229.6	9.235E-06	228.4	9.217E-06	227.9	9.195E-06
Case III : $n = 10$ $(t_0^* = 1316.40,\ A(t_0^*) = 0.999869)$						
Mean	1507.9	0.99985882	1473.9	0.99986025	1465.9	0.99986343
Median	1493.7	0.99986100	1460.7	0.99986200	1453.5	0.99986500
SD	287.0	2.129E-05	278.6	2.040E-05	275.7	1.966E-05
Case III : $n = 100$						
Mean	1351.5	0.99987013	1347.5	0.99987013	1347.1	0.99987037
Median	1346.6	0.99987000	1342.8	0.99987000	1342.1	0.99987100
SD	138.2	7.309E-06	138.0	7.285E-06	137.9	7.248E-06
Case III : $n = 200$						
Mean	1337.2	0.99987004	1335.8	0.99987004	1335.6	0.99987017
Median	1334.3	0.99987000	1332.9	0.99987000	1332.5	0.99987000
SD	110.7	5.281E-06	110.6	5.272E-06	110.5	5.253E-06

表4　シミュレーション実験結果（上限）

	$T^{-*}_{\mathrm{up},n+1}$	$A_{\mathrm{up},n+1}$	$T^{-*}_{\mathrm{up},n+2}$	$A_{\mathrm{up},n+2}$	$T^{-*}_{\mathrm{up},n+3}$	$A_{\mathrm{up},n+3}$
Case I : $n=10$ $(t_0^*=1816.62,\ A(t_0^*)=0.999792)$						
Mean	2673.9	0.99983865	2649.2	0.99983580	2621.8	0.99983701
Median	2103.7	0.99984300	2052.1	0.99984000	2017.2	0.99984100
SD	1733.4	3.679E-05	1746.8	3.610E-05	1750.6	3.579E-05
Case I : $n=100$						
Mean	2007.9	0.99980220	1993.2	0.99980212	1985.1	0.99980271
Median	1842.0	0.99980300	1828.8	0.99980300	1823.9	0.99980300
SD	843.7	1.514E-05	840.4	1.509E-05	828.7	1.505E-05
Case I : $n=200$						
Mean	1911.0	0.99979824	1900.3	0.99979822	1897.3	0.99979855
Median	1825.0	0.99979800	1816.8	0.99979800	1814.6	0.99979900
SD	532.1	1.098E-05	526.2	1.096E-05	521.4	1.094E-05
Case II : $n=10$ $(t_0^*=1359.67,\ A(t_0^*)=0.999819)$						
Mean	1724.5	0.99985440	1677.7	0.99985242	1664.5	0.99985400
Median	1539.5	0.99985700	1500.2	0.99985500	1502.9	0.99985700
SD	827.0	3.066E-05	807.1	2.991E-05	780.5	2.955E-05
Case II : $n=100$						
Mean	1422.0	0.99982687	1419.2	0.99982682	1417.6	0.99982733
Median	1393.8	0.99982700	1392.2	0.99982700	1390.9	0.99982800
SD	295.5	1.259E-05	293.0	1.254E-05	291.5	1.247E-05
Case II : $n=200$						
Mean	1396.3	0.99982384	1394.5	0.99982383	1394.2	0.99982410
Median	1377.8	0.99982400	1376.1	0.99982400	1375.6	0.99982400
SD	230.0	9.163E-06	228.3	9.143E-06	227.8	9.121E-06
Case III : $n=10$ $(t_0^*=1316.40,\ A(t_0^*)=0.999869)$						
Mean	1469.2	0.99988891	1456.3	0.99988789	1463.3	0.99988872
Median	1454.9	0.99989100	1442.9	0.99988900	1451.0	0.99989000
SD	288.1	1.781E-05	280.1	1.727E-05	276.6	1.676E-05
Case III : $n=100$						
Mean	1345.7	0.99987343	1344.7	0.99987339	1346.2	0.99987361
Median	1340.6	0.99987400	1339.9	0.99987400	1341.2	0.99987400
SD	138.0	7.192E-06	138.0	7.161E-06	137.6	7.123E-06
Case III : $n=200$						
Mean	1334.4	0.99987171	1334.5	0.99987170	1335.3	0.99987181
Median	1331.2	0.99987200	1331.6	0.99987200	1332.1	0.99987200
SD	111.0	5.232E-06	110.8	5.224E-06	110.4	5.214E-06

さらに表 3, 4 の各ケースについて，X_{n+1}, X_{n+2}, X_{n+3} の比較を行うと，全てのケースにおいて X_{n+1} から X_{n+3} へと適応的に若化スケジューリングを行っていくことで，下限と上限に対する最適若化スケジュール推定値の平均値が t_0^* に近づくことが確認できる．また中央値についてもおおむね同じ傾向があることが読み取れる．ただし，例外は下限，上限の Case I, $n = 200$ の X_{n+2} から X_{n+3} および上限 Case III の $n = 10,\ 100,\ 200$ での X_{n+2} から X_{n+3} である．さらに表 3 および 4 のアベイラビリティ推定値に関して，下限については Case III でデータ数が少ない $n = 10$ の時のみ例外があるが，それ以外の下限について，また上限の全ての場合についてノンパラメトリック予測推論によるアベイラビリティの推定値が理論的な最大アベイラビリティ $A(t_0^*)$ よりも大きく推定される傾向にあることが読み取れる．

次に表 5 には 10,000 回のシミュレーション実験において得られたアベイラビリティの下限，上限をそれぞれ考えた場合の予防若化スケジュール推定値が一致した回数を示す．表 5 よりこれら 2 つの推定値は多くの場合に一致するため，この一致した結果を最適な適応的若化スケジュールとして利用できることがわかる．

一方でこれら 2 つの推定値が異なった場合の指針については表 6, 7 に示す最適若化スケジュールとアベイラビリティそれぞれの絶対誤差平均の考察から与えることができる．表 6 において例えば $\Gamma_{\mathrm{low},n+1}$ は X_{n+1} に対する下限に基づいた最適若化スケジュールの推定値と理論的な最適若化スケジュール t_0^* との絶対誤差平均を意味する．表 6 より，例えば Case I の $n = 200$ に対する $\Gamma_{\mathrm{low},n+1} = 390.2838$ と $\Gamma_{\mathrm{up},n+1} = 400.8446$ のように下限の絶対誤差平均を対応する上限のそれと比較すると，おおむね下限の誤差が小さくなることが読み取れる．

またアベイラビリティ下限の推定値と理論的な最大アベイラビリティ $A(t_0^*)$ の絶対誤差平均を $\Delta_{\mathrm{low},n+1}$ のように表す．表 7 より，例えば Case I, $n = 10$ の $\Delta_{\mathrm{low},n+3} = 3.620\text{E-}05$ と $\Delta_{\mathrm{up},n+3} = 5.029\text{E-}05$ など，上限と下限のアベイラビリティ絶対誤差平均を比較すると，全ての場合において下限の誤差が小さくなっていることがわかる．したがって，本研究で提案したノンパラメトリック予測推論による下限と上限の適応的最適若化スケジュールは多くの場合一致するが，それらが異なった場合には，アベイラビリティ推定値の誤差の観点から下限の若化スケジュールを採用すれば良いことがわかる．

さらに，表 6 においては X_{n+1} から X_{n+3} へと適応的に若化スケジューリングを行うことで最適な若化スケジュールの誤差が小さくなることも読み取れる．一方で表 7 より，アベイラビリティの誤差は X_{n+1} から X_{n+2} へと適応的にスケジューリングすることで小さくなるが，X_{n+2} から X_{n+3} のステップでは大きくなることがわかる．これは X_{n+1} や X_{n+2} に対してソフトウェアの予防若化が実施されることによる障害時間データの打切りが発生し，図 3 に示したとおり，打切り時間以降で予測信頼度関数の上下限の差が大きくなる．この影響によって

表 5　アベイラビリティ上下限に関して若化スケジュールが一致した回数

	$T^{*}_{\mathrm{low},n+1}=T^{-*}_{\mathrm{up},n+1}$			$T^{*}_{\mathrm{low},n+2}=T^{-*}_{\mathrm{up},n+2}$			$T^{*}_{\mathrm{low},n+3}=T^{-*}_{\mathrm{up},n+3}$		
n	10	100	200	10	100	200	10	100	200
Case I	8326	9603	9820	8234	9702	9867	8129	9668	9859
Case II	8902	9689	9807	9177	9867	9924	9293	9900	9953
Case III	9142	9650	9759	9599	9831	9886	9727	9939	9971

表 6　最適若化スケジュール推定値の絶対誤差平均

	$\Gamma_{\mathrm{low},n+1}$	$\Gamma_{\mathrm{low},n+2}$	$\Gamma_{\mathrm{low},n+3}$	$\Gamma_{\mathrm{up},n+1}$	$\Gamma_{\mathrm{up},n+2}$	$\Gamma_{\mathrm{up},n+3}$
Case I						
$n=10$	1031.5736	957.4009	927.1174	1323.9232	1319.0489	1301.2495
$n=100$	490.6130	485.6303	480.3622	544.4960	537.0916	530.5784
$n=200$	390.2838	387.2293	385.7741	400.8446	395.7181	393.6200
Case II						
$n=10$	555.2083	507.3969	482.9939	593.2947	561.4355	544.8581
$n=100$	236.4491	233.7691	232.4838	235.3160	233.6476	232.2714
$n=200$	183.6854	182.2044	181.7656	183.6299	182.0158	181.5977
Case III						
$n=10$	271.0886	251.1247	246.6099	255.9369	245.8521	246.5535
$n=100$	112.9219	112.2345	112.0809	111.8999	111.8026	111.7768
$n=200$	89.8739	89.5843	89.4644	89.6982	89.5958	89.3550

ノンパラメトリック予測推論によるアベイラビリティの下限と上限の差も図 4 のように打切り時刻以降で大きくなるためであると推察される．

6.　むすび

本論文ではエージング現象が観測されるようなソフトウェアシステムに対する予防若化方策をセミマルコフモデルで記述し，障害発生時間データからノンパラメトリック予測推論によって適応的な 3 ステップの最適予防若化スケジュールを導出した．具体的には n 個の障害発生時間データが得られている状態から $n+1, n+2, n+3$ 番目の障害発生時間の確率分布をノンパ

表７　システムアベイラビリティ上下限の絶対誤差平均

	$\Delta_{\mathrm{low},n+1}$	$\Delta_{\mathrm{low},n+2}$	$\Delta_{\mathrm{low},n+3}$	$\Delta_{\mathrm{up},n+1}$	$\Delta_{\mathrm{up},n+2}$	$\Delta_{\mathrm{up},n+3}$
Case I						
$n = 10$	3.727E-05	3.591E-05	3.620E-05	5.212E-05	4.958E-05	5.029E-05
$n = 100$	1.343E-05	1.337E-05	1.355E-05	1.496E-05	1.488E-05	1.518E-05
$n = 200$	9.508E-06	9.474E-06	9.577E-06	1.022E-05	1.019E-05	1.033E-05
Case II						
$n = 10$	2.937E-05	2.817E-05	2.813E-05	4.047E-05	3.863E-05	3.964E-05
$n = 100$	1.082E-05	1.077E-05	1.086E-05	1.214E-05	1.207E-05	1.227E-05
$n = 200$	7.730E-06	7.715E-06	7.771E-06	8.344E-06	8.323E-06	8.423E-06
Case III						
$n = 10$	1.804E-05	1.701E-05	1.573E-05	2.305E-05	2.201E-05	2.239E-05
$n = 100$	5.930E-06	5.908E-06	5.912E-06	6.879E-06	6.843E-06	6.916E-06
$n = 200$	4.291E-06	4.286E-06	4.293E-06	4.739E-06	4.731E-06	4.768E-06

ラメトリック予測推論によって求め，予測信頼度関数の上下限を定式化，システムアベイラビリティを最大にするソフトウェア若化スケジュールを適応的に導出した．

シミュレーション実験によって，$n+1$ 番目から $n+3$ 番目の障害へと適応的に若化スケジューリングを進めていくことで，本論文で提案した最適ソフトウェア若化スケジュールの推定値は理論的な最適解へと近づき，その平均誤差が小さくなることが示された．

参考文献

[1] M. Grottke, L. Lie, K. V. Vaidyanathan, and K. S. Trivedi, Analysis of software aging in a web server, *IEEE Transactions on Reliability*, **55**(3), 411–420, (2006).

[2] J. Gray, Why do computers stop and what can be done about it? *Technical Report 85.7, PN87614*, Tandem Computers, (1985).

[3] M. Grottke and K. S. Trivedi, A classification of software faults, *Supplemental Proceedings of Sixteenth International IEEE Symposium on Software Reliability Engineering (ISSRE-2005)*, 4.19–4.20, IEEE CS Press, (2005).

[4] E. S. Raymond, *The New Hacker's Dictionary*, MIT Press, (1991).

[5] E. Adams, Optimizing preventive service of the software products, *IBM Journal of*

Research & Development, **28**(1), 2–14, (1984).

[6] E. Marshall, Fatal error: How Patriot overlooked a Scud, *Science*, **255**(5050), 1347, (1992).

[7] A. Avritzer, and E. J. Weyuker, Monitoring smoothly degrading systems for increased dependability, *Empirical Software Engineering*, **2**(1), 59–77, (1997).

[8] V. Castelli, R. E. Harper, P. Heidelberger, S. W. Hunter, K. S. Trivedi, K. V. Vaidyanathan, and W. P. Zeggert, Proactive management of software aging, *IBM Journal of Research & Development*, **45**(2), 311–332, (2001).

[9] W. Yurcik, and D. Doss, Achieving fault-tolerant software with rejuvenation and reconfiguration, *IEEE Software*, **18**(4), 48–52, (2001).

[10] M. Shereshevsky, J. Crowell, B. Cukic, V. Gandikota, and Y. Liu, Software aging and multifractality of memory resources, *Proceedings of International Conference on Dependable Systems and Networks (DSN-2003)*, 721–730, IEEE CS Press, (2003).

[11] R. Matias and P. J. F. Filho, An experimental study on software aging and rejuvenation in web servers, *Proceedings of Annual International Computer Software and Applications Conference (COMPSAC-2006)*, 189-196, IEEE CS Press, (2006).

[12] M. Grottke and K. S. Trivedi, Fighting bugs: Remove, retry replicate, and rejuvenate, *IEEE Computer*, **40**, 107–109, (2007).

[13] M. Grottke, R. Matias and K. S. Trivedi, The fundamentals of software aging, *Proceedings of 18th International Symposium on Software Reliability Engineering (ISSRE-2008)*, 1–6, IEEE CS Press, (2008).

[14] A. Macedo, T. Ferreira, R. Matias, The mechanics of memory-related software aging, *Proceedings of Second International Workshop on Software Aging and Rejuvenation (WoSAR-2010)*, 1–5, IEEE CS Press, (2010).

[15] R. Matias, P. A. Barbetta, K. S. Trivedi and P. J. F. Filho, Accelerated Degradation Tests Applied to Software Aging Experiments, *IEEE Transactions on Reliability* **59**(1), 102–114, (2010).

[16] F. Machida, V. F. Nicola and K. S. Trivedi, Job completion time on a virtualized server subject to software aging and rejuvenation, *Proceedings of Third International Workshop on Software Aging and Rejuvenation (WoSAR-2011)*, 44–49, IEEE CS Press, (2011).

[17] T. Yoshimura, H. Yamada and K. Kono, Can Linux be rejuvenated without reboots?

Proceedings of Third International Workshop on Software Aging and Rejuvenation (WoSAR-2011), 50–55, IEEE CS Press, (2011).

[18]「プログラムに不具合」日航，システム障害で，日経産業新聞, 6 月 10 日, 13 ページ, (2014).

[19] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, Software rejuvenation: analysis, module and applications, *Proceedings of 25th International Symposium on Fault Tolerant Computing (FTC-1995)*, 381–390, IEEE CS Press, (1995).

[20] T. Dohi, K. Goševa-Popstojanova, and K. S. Trivedi, Estimating software rejuvenation schedule in high assurance systems, *Computer Journal*, **44**(6), 473–485, (2001).

[21] T. Dohi, K. Iwamoto, H. Okamura, and N. Kaio, Discrete availability models to rejuvenate a telecommunication billing application, *IEICE Transactions on Communications (B)*, **E86-B**(10), 2931–2939, (2003).

[22] K. Vaidyanathan and K. S. Trivedi, A comprehensive model for software rejuvenation, *IEEE Transactions on Dependable and Secure Computing*, **2**(2), 124–137, (2005).

[23] K. Rinsaka and T. Dohi, Estimating the optimal software rejuvenation schedule with small sample data, *Quality Technology and Quantitative Management Journal*, **6**(1), 55–65, (2009).

[24] K. Rinsaka and T. Dohi, A faster algorithm for periodic preventive rejuvenation schedule maximizing system availability, in M. Malek, M. Reitenspiess and A. Moorsel (Eds.), *Service availability: 4th International Service Availability Symposium, LNCS 4526*, 94–109, Springer, (2007).

[25] J. Zhao, Y. B. Wang, G. R. Ning, K. S. Trivedi, R. Matias and K.-Y. Cai, A comprehensive approach to optimal software rejuvenation, *Performance Evaluation*, **70**(11), 917–933, (2013).

[26] A. Pfening, S. Garg, A. Puliafito, M. Telek and K. S. Trivedi, Optimal rejuvenation for tolerating software failures *Performance Evaluation*, **27&28**, 491–506, (1996).

[27] H. Eto and T. Dohi, Optimality of control-limit type of software rejuvenation policy, *Proceedings of International Conference on Parallel and Distributed Systems (ICPADS'05)*, 483–487, IEEE CS Press, (2005).

[28] H. Eto and T. Dohi, Analysis of a service degradation model with preventive rejuvenation, in D. Penkler, M. Reitenspiess and F. Tam (Eds.), *Service availability: Third International Service Availability Symposium, LNCS 4328*, 17–29, Springer, (2006).

[29] H. Eto and T. Dohi, Determining the optimal software rejuvenation schedule via semi-Markov decision process, *Journal of Computer Science*, **2**(6) 528–535, (2006).

[30] H. Eto, T. Dohi and J. Ma, Simulation-based optimization approach for software cost model with rejuvenation, in C. Rong, M. G. Jaatun, F. E. Sandnes, L T. Yang and J. Ma (Eds.), *Autonomic and Trusted Computing: Fifth International Conference, ATC 2008, LNCS 5060*, 206–2180, Springer (2008).

[31] H. Okamura and T. Dohi, Dynamic software rejuvenation policies in a transaction-based system under Markovian arrival processes, *Performance Evaluation*, **70**(3) 197–211, (2013).

[32] F. P. A. Coolen, P. Coolen-Schrijner, and K. J. Yan, Nonparametric predictive inference in reliability, *Reliability Engineering and System Safety*, **78**(2), 185–193, (2002).

[33] B. M. Hill, Posterior distribution of percentiles: Bayes' theorem for sampling from a population, *Journal of the American Statistical Association*, **63**(322), 677–691, (1968).

[34] B. M. Hill, Parametric models for A_n: splitting processes and mixtures, *Journal of Royal Statistical Society B*, **55**(2), 423–433, (1993).

[35] F. P. A. Coolen, On the use of imprecise probabilities in reliability, *Quality and Reliability Engineering International*, **20**(3), 193–202, (2004).

[36] F. P. A. Coolen and M. J. Newby, Guidelines for corrective replacement based on low stochastic structure assumption, *Quality and Reliability Engineering International*, **13**(4), 177–182, (1997).

[37] F. P. A. Coolen and P. Coolen-Schrijner, Condition monitoring: a new perspective, *Journal of the Operational Research Society*, **51**(3), 311–319, (2000).

[38] F. P. A. Coolen and K. J. Yan, Nonparametric predictive inference with right-censored data, *Journal of Statistical Planning and Inference*, **126**(1), 25–54, (2004).

[39] P. Coolen-Schrijner and F. P. A. Coolen, Non-parametric predictive inference for age replacement with a renewal argument, *Quality and Reliability Engineering International*, **20**(3), 203–215, (2004).

[40] P. Coolen-Schrijner and F. P. A. Coolen, Adaptive age replacement strategies based on nonparametric predictive inference, *Journal of the Operational Research Society*, **55**(12), 1281–1297, (2004).

[41] F. P. A. Coolen and T. Coolen-Maturi, Predictive inference for system reliability after common-cause components failures, *Reliability Engineering and System Safety*, **135**, 27–33, (2015).

[42] 林坂弘一郎, 土肥正, ソフトウェアの最適若化スケジューリングに対するノンパラメトリック予測推論：適応的スケジューリング, 神戸学院大学経営学論集, **4**(1), 31–53, (2007).

[43] K. Rinsaka and T. Dohi, Adaptive software rejuvenation schedule based on non-parametric predictive inference: An expected cost model, *Advanced Reliability Modeling III – Proceedings of The 2008 Asian International Workshop on Advanced Reliability Modeling (AIWARM-2008)*, 591–598, McGraw-Hill, (2008).

[44] K. Rinsaka and T. Dohi, Non-parametric predictive inference of preventive rejuvenation schedule in operational software systems, *Proceedings of 18th International Symposium on Software Reliability Engineering (ISSRE-2007)*, 247–256, IEEE CS Press, (2007).

[45] K. Rinsaka and T. Dohi, Non-parametric predictive inference of adaptive software rejuvenation schedule, *Proceedings of 18th International Symposium on Software Reliability Engineering (ISSRE-2008)*, 1–6, IEEE CS Press, (2008).

[46] K. Rinsaka and T. Dohi, Toward high assurance software systems with adaptive fault management, *Software Quality Journal (Online First Articles)*, 21 pages, (2014).