比例強度ソフトウェア信頼性モデルに対するカーネル回帰

林　坂　弘一郎

概 要

非同次ポアソン過程 (NHPP) モデルに代表される多くのソフトウェア信頼性モデルは，ソフトウェア開発工程のテスト工程において検出されるフォールト数データからソフトウェアの品質特性の評価を行う．本論文では検出フォールト数だけでなく，テスト労力などのメトリクスデータも利用した比例強度モデルに対してカーネル回帰を適用する．これにより，パラメータ推定が容易になり，NHPP モデルと比較してモデルの適合度および予測精度が向上することを示す．

キーワード：ソフトウェア信頼性，ソフトウェアメトリクス，比例強度モデル，非同次ポアソン過程，カーネル回帰，ノンパラメトリック推定，クロスバリデーション

1. まえがき

ソフトウェアの開発工程にはウォータフォールモデル，プロトタイプモデル，スパイラルモデルの他に，エクストリームプログラミングに代表されるアジャイル開発手法などがある．今日においても最も利用されているソフトウェア開発工程はウォータフォールモデルである．これは，要求定義，外部設計，内部設計，プログラミング，テスト，運用・保守の行程を上流工程から下流工程へと滝が流れるように次々と進めていく開発モデルである．ウォータフォールモデルでは各開発行程の成果物に対して充分なレビューを実施し，その行程での欠陥や誤りを可能な限り除去することが重要である．しかしながら，レビューで検出できなかった欠陥や誤りは各工程の成果物に埋め込まれ，次の行程へと進んでいくこととなる [1,2].

テストの目的はシステムが要求される性能や機能を満たしているかどうかを検証することであり，正しく動作することを確認するよりも，より多くのエラーを検出することにある．テスト工程においてエラーを発見・除去し，より高い信頼性をもつ高品質なソフトウェアを出荷することが求められている．

テストの進捗状況を把握したり，テストが終了した時点におけるソフトウェアの信頼性を把握したりするためにソフトウェア信頼性モデルが利用されている．多くのソフトウェア信頼性モデルは確率モデルによって記述されており，テスト工程において個々のフォールトが発見される確率が時間と共にどのように変化するかについてはこれまでに様々なモデルが提案されている．特に代表的なモデルは非同次ポアソン過程 (Non-Homogeneous Poisson Process; NHPP) モデルである．Goel and Okumoto [3] は指数形 NHPP モデルを提案し，その後，Yamada ら [4] の遅延 S 字形 NHPP モデルや，多くの NHPP モデル [5–8] が提案された．NHPP モデルは一定のテスト環境下におけるフォールト検出過程を記述したモデルである．しかしながら，現実のテスト工程では，そのテスト工程中にテストチーム人員数に増減があったり，テストの負荷が変更されるなど，必ずしもテスト環境が一定であるとは限らない．例えば，ソフトウェアの納期が近づきつつあるにもかかわらず，テスト項目の消化が不充分であれば，テストの人員を増員して集中的にテストを実施することも考えられる．しかし，このような場合に将来のフォールト検出の振る舞いを予測するためのモデルとして NHPP モデルは不十分である．

Yamada ら [9–12] はテスト環境を記述するテスト労力関数に基づいた NHPP モデルであるテスト労力依存型モデルを提案した．これはテスト工程において投入されるテスト労力をワイブル関数によって表現することで，テスト環境の変化が NHPP モデルに取り入れられている．しかしながら，テスト労力依存型モデルでは，テスト労力関数によって将来のテスト労力量までもが規定されてしまう．したがって，テスト労力の動的な変化には対応できない．

一方で，Kapur ら [13] はテスト環境の変化をチェンジポイントモデルとして表現し，Dharmasena ら [14] はカーネル密度推定 [15] に基づいた局所的多項式回帰モデルを提案した．Dharmasena らは局所的多項式回帰モデルが NHPP と比較して高い適合度を有していることを示したが，このモデルによって得られた累積フォールト検出数にはその振る舞いが非減少関数にならないという問題点がある．

Rinsaka ら [16] や Shibata ら [17] は動的に変化するテスト環境をメトリクスデータとして有機的に利用したソフトウェア信頼性モデルを提案した．これらのモデルにおいて考えられているメトリクスデータは d $(d = 1, 2, \cdots)$ 次元の時間依存共変量である．Rinsaka ら [16] の比例強度モデルや Shibata ら [17] のマルチファクターモデルでは，テスト環境の動的な変化を考慮することで，NHPP モデルやテスト労力依存型モデルと比較してソフトウェアフォールトデータに対する適合度が高くなることが示された．Shibata ら [18] は比例強度モデルのパラメータを推定するためのソフトウェアツールを開発した．しかしながら，比例強度モデルはパラメータ推定にいくつかの問題点をもっている．つまり，推定時にパラメータの非負

制約違反が多発したり，大域的な最適化が難しいという問題点である．Okamura ら [19] はマルチファクターモデル [17] に対する EM アルゴリズムを示し，パラメータ推定に関する問題点を改善した．

本論文では比例強度モデル [16] に対してカーネル回帰 [20,21] に基づいたノンパラメトリック推定を導入し，比例強度モデルのパラメータ推定に関する問題点を改善することを考える．具体的には，比例強度モデルのベースライン強度関数と各期に検出されたフォールト数との比率の自然対数に対してカーネル回帰を適用する．これによって，比例強度モデルで問題となっていた比例強度に関するパラメータ推定がノンパラメトリックに行うことができ，最尤推定における推定パラメータ数を NHPP モデルと同水準まで引き下げることが可能となる．さらに，NHPP モデルと比較して提案モデルの適合度および予測精度が高くなることも示す．また，NHPP モデルやテスト労力依存型モデルでは将来のテスト計画に関する感度分析が不可能であったが，本論文で提案するモデルではこれが可能となることについても述べる．

本論文の構成は次の通りである．**2.** では代表的なソフトウェア信頼性モデルについて概観し，その特徴について述べる．**3.** ではカーネル回帰の手順とパラメータの決定方法について述べる．**4.** ではカーネル回帰を比例強度モデルに適用するためのアルゴリズムを示し，**5.** では，実際のデータを用いて NHPP モデルと提案モデルの適合度および予測精度を比較することで，提案モデルの有効性を検証する．

2.　ソフトウェア信頼性モデル

ここでは，Goel and Okumoto [3] および Yamada ら [4] が提案した代表的な NHPP モデル，Yamada ら [9–11] のテスト労力依存型モデル，および Rinsaka ら [16] が提案した比例強度モデルについて概観し，その特徴を述べる．

2.1　NHPP モデル

NHPP モデルは時間区間 $(0,t]$ に検出されるソフトウェアフォールトを数える確率変数 $\{N(t),\ t \geq 0\}$ を導入し，これに対して非同次ポアソン過程 (NHPP) を仮定するモデルである．NHPP モデルは以下の性質を満たす [1,2,5].

(i) $N(0) = 0$
(ii) $\{N(t),\ t \geq 0\}$ は独立増分をもつ
(iii) $\Pr\{N(t+\Delta t) - N(t) \geq 2\} = o(\Delta t)$
(iv) $\Pr\{N(t+\Delta t) - N(t) = 1\} = \lambda(t)\Delta t + o(\Delta t)$

ここで，$\lambda(t)$ は強度関数である．また，$o(\Delta t)$ は高位の無限小を意味し，

$$\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0 \tag{1}$$

である．このとき，$N(t)$ は平均値関数 $H(t)$ の非同次ポアソン過程

$$\Pr\{N(t) = n\} = \frac{\{H(t)\}^n}{n!} \exp[-H(t)], \quad (n = 0, 1, 2, \cdots) \tag{2}$$

に従う．ただし，

$$H(t) = \mathrm{E}[N(t)] = \int_0^t \lambda(x) dx, \quad (t \geq 0) \tag{3}$$

である．

これまでに数多くの NHPP モデルが提案されているが，特に代表的な 2 種類のモデルは Goel and Okumoto [3] が提案した指数形ソフトウェア信頼性モデル

$$\lambda(t) = ab \exp(-bt) \tag{4}$$

$$H(t) = a\{1 - \exp(-bt)\} \tag{5}$$

および， Yamada ら [4] が提案した遅延 S 字形ソフトウェア信頼性モデル

$$\lambda(t) = ab^2 t \exp(-bt) \tag{6}$$

$$H(t) = a[1 - (1 + bt) \exp(-bt)] \tag{7}$$

である．なお，パラメータ a $(a > 0)$ はソフトウェアのテスト開始前に潜在する総期待フォールト数を意味し，

$$H(\infty) = \lim_{t \to \infty} H(t) = a \tag{8}$$

である．また，パラメータ b $(b > 0)$ は 1 個あたりのフォールト発見率を意味している．

NHPP モデルのパラメータ推定には一般的に最尤法が用いられる．一定のテスト時間間隔 $(0,\ t_k]$ において検出された総フォールト数を y_k $(k = 1, 2, \cdots, n)$ とする．NHPP モデルのパラメータ $\boldsymbol{\theta} = (a,\ b)$ に対して，平均値関数を $H(t; \boldsymbol{\theta})$ と書く．このとき，n 組の観測データ $(t_k,\ y_k)$ $(k = 1, 2, \cdots, n)$ に対する NHPP モデルの尤度関数は，

$$\begin{aligned} L(\boldsymbol{\theta}) &= \Pr\{N(t_1) = y_1,\ N(t_2) = y_2,\ \cdots,\ N(t_n) = y_n\} \\ &= \exp[-H(t_n; \boldsymbol{\theta})] \prod_{k=1}^{n} \frac{\{H(t_k; \boldsymbol{\theta}) - H(t_{k-1}; \boldsymbol{\theta})\}^{(y_k - y_{k-1})}}{(y_k - y_{k-1})!} \end{aligned} \tag{9}$$

となる．ただし，$t_0 = 0$，$y_0 = 0$ とする．式 (9) から，対数尤度関数は

$$LLF(\boldsymbol{\theta}) = \sum_{k=1}^{n} (y_k - y_{k-1}) \ln[H(t_k; \boldsymbol{\theta}) - H(t_{k-1}; \boldsymbol{\theta})] - H(t_n; \boldsymbol{\theta})$$

$$-\sum_{k=1}^{n}\ln[(y_k-y_{k-1})!] \tag{10}$$

となる．したがって，式 (10) を最大化する最尤推定値 $\hat{\boldsymbol{\theta}}=(\hat{a},\ \hat{b})$ は

$$\frac{\partial LLF(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}=\mathbf{0} \tag{11}$$

を数値的に解くことで得られる．なお，$\mathbf{0}=(0,0)$ である．

NHPP モデルでは，テスト期間中にテスト環境が変化しないことが仮定されている．したがって，例えばテスト行程の終盤にもかかわらず未だ大半のテストケースが消化されていないなどの理由でテスト人員を増員して集中的にテストを実施したとしても，NHPP モデルではこのようなテスト環境の変化を考慮することは出来ない．

2.2　テスト労力依存型モデル

ソフトウェア開発のテスト工程で投入される工数やテストケース数などの資源はテスト労力と呼ばれる．Yamada ら [9–11] は，テスト時刻 t におけるテスト労力の投入量をテスト労力関数 $w(t;\boldsymbol{\beta})$ として，時間区間 $(0,\ t]$ において投入される総テスト労力量を

$$W(t,\boldsymbol{\beta})=\int_0^t w(x;\boldsymbol{\beta})dx \tag{12}$$

で記述し，平均値関数を

$$H_t(t;\boldsymbol{\theta},\boldsymbol{\beta})=a\left(1-\exp\left[-r\cdot W(t;\boldsymbol{\beta})\right]\right) \tag{13}$$

とするテスト労力依存型ソフトウェア信頼性モデルを提案した．ただし，a は総期待フォールト数，r は単位テスト労力あたりのフォールト発見率を意味する．Yamada ら [11, 12] はテスト労力関数にワイブル曲線

$$w(t;\boldsymbol{\beta}) = \alpha\delta m t^{m-1}\exp\left(-\delta t^m\right),\quad (\alpha>0,\ \delta>0,\ m>0) \tag{14}$$

$$W(t;\boldsymbol{\beta}) = \alpha\left[1-\exp\left(-\delta t^m\right)\right] \tag{15}$$

を提案した．ここで，$\boldsymbol{\beta}=(\alpha,\delta,m)$ はテスト労力関数を特徴付けるパラメータであり，α はテストに必要な総テスト労力量，δ は尺度パラメータ，m は形状パラメータである．

テスト労力依存型モデルのパラメータ推定には最小二乗法と最尤法の 2 種類の推定手法が利用される．すなわち，時間区間 $(t_{k-1},\ t_k]$ において投入されたテスト労力量 w_k $(k=1,2,\cdots,n)$ から最小二乗法によってテスト労力関数のパラメータ $\boldsymbol{\beta}=(\alpha,\delta,m)$ を推定する．

その後，フォールト検出数データ y_k $(k = 1, 2, \cdots, n)$ から最尤法によって平均値関数のパラメータ $\boldsymbol{\theta} = (a,\ r)$ が推定される．

テスト労力依存型モデルでは，テスト労力の投入パターンを式 (14) のテスト労力関数で表現することにより，NHPP モデルでは考慮できなかったテスト環境の変化をモデルに取り入れることが可能となる．しかしながら，テスト労力関数を推定した時点で，その後のテスト労力投入量がテスト労力関数によって規定されてしまうこととなる．このため，将来に集中的にテストを実施した場合にフォールトの検出傾向がどのように変化するかを調査するなどの感度分析をテスト労力依存型モデルでは行うことが出来ない．

2.3　比例強度モデル

比例強度モデル [16] はソフトウェアテスト工程のテスト環境を特徴付けるメトリクスデータ（テスト時間，投入テスト労力量等）に基づきソフトウェアの信頼性を評価する．時間区間 $(t_{k-1},\ t_k]$ $(k = 1, 2, \cdots, n)$ において投入された d $(d = 1, 2, \cdots)$ 種類のテスト労力量であるメトリクスデータ $\boldsymbol{x}_k = (x_{k1}, \cdots, x_{kd})^{\mathrm{T}}$ が観測可能であるとする．ここで，$\boldsymbol{x}_k$ は時間依存共変量である．メトリクスデータを考慮した強度関数を

$$\lambda_x\left(t_k; \boldsymbol{\theta}, \boldsymbol{\beta} | \boldsymbol{x}_k\right) = \lambda_0(t_k; \boldsymbol{\theta}) g(\boldsymbol{x}_k; \boldsymbol{\beta}) \tag{16}$$

と定義する．ここで，$\lambda_0(t_k; \boldsymbol{\theta})$ (> 0) はベースライン強度関数，$\boldsymbol{\theta} = (a, b)$ はベースライン強度関数のパラメータである．また，$g(\boldsymbol{x}_k; \boldsymbol{\beta})$ は共変量関数，$\boldsymbol{\beta} = (\beta_1, \cdots, \beta_d)^{\mathrm{T}}$ は係数ベクトルであり，$\beta_j \geq 0$ $(j = 1, 2, \cdots, d)$ と仮定する．

Rinsaka ら [16] は共変量関数に

$$g(\boldsymbol{x}_k; \boldsymbol{\beta}) = \exp(\boldsymbol{x}_k^{\mathrm{T}} \boldsymbol{\beta}) \tag{17}$$

を仮定した．このとき，比例強度モデルの平均値関数 $H_p(t_k; \boldsymbol{\theta}, \boldsymbol{\beta})$ は次式のようになる．

$$\begin{aligned} H_p(t_k; \boldsymbol{\theta}, \boldsymbol{\beta}) &= \sum_{i=1}^{k} \exp(\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{\beta}) \int_{t_{i-1}}^{t_i} \lambda_0(u; \boldsymbol{\theta}) du \\ &= \sum_{i=1}^{k} \exp(\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{\beta}) \left[H_0(t_i; \boldsymbol{\theta}) - H_0(t_{i-1}; \boldsymbol{\theta})\right] \end{aligned} \tag{18}$$

ここで，$H_0(t_i; \boldsymbol{\theta})$ はベースライン平均値関数であり，

$$H_0(t_i; \boldsymbol{\theta}) = \int_0^{t_i} \lambda_0(u; \boldsymbol{\theta}) du \tag{19}$$

である．

式 (18) より，比例強度モデルでは時間区間 $(t_{k-1},\ t_k]$ において新たに検出されるフォールト数 $y_k - y_{k-1}$ の期待値は，その時間区間に投入されたメトリクス $\boldsymbol{x}_k$ に比例する．また，すべての $j\ (=1,2,\cdots,d)$ に対して $\beta_j = 0$ のとき，式 (18) の比例強度モデルは NHPP モデルに帰着することに注意を要する．

比例強度モデルでは時間依存共変量 $\boldsymbol{x}_k$ を利用しているため，テスト労力量が関数的に決定されるテスト労力依存型モデルと異なり，テスト環境の動的な変化に柔軟に対応できる．これは，将来のテスト環境データを式 (18) に入力すれば，フォールト検出の傾向が予測できる，つまり，NHPP モデルやテスト労力依存型モデルが出来なかったテスト環境に関する感度分析が可能であることを意味している．しかし，これまでのテスト環境を示すメトリクスデータと比較して極めて大きなメトリクスデータを入力して将来の予測を行った場合，式 (17) に示す比例強度が指数的に大きくなる結果，式 (18) のモデルが過剰に反応する可能性があることに注意を要する．

Rinsaka ら [16] は式 (10) を変形した対数尤度関数

$$\begin{aligned} LLF(\boldsymbol{\theta},\boldsymbol{\beta}) \quad &= \quad \sum_{k=1}^{n}(y_k - y_{k-1})\ln[H_p(t_k;\boldsymbol{\theta},\boldsymbol{\beta}) - H_p(t_{k-1};\boldsymbol{\theta},\boldsymbol{\beta})] \\ &\quad - H_p(t_n;\boldsymbol{\theta},\boldsymbol{\beta}) - \sum_{k=1}^{n}\ln[(y_k - y_{k-1})!] \end{aligned} \tag{20}$$

を最大にする最尤推定値 $(\boldsymbol{\theta},\ \boldsymbol{\beta})$ を数値的に求めた．Shibata ら [18] は式 (20) に基づいて比例強度モデルのパラメータを推定するためのソフトウェアツール (PISRAT) を公開した．この PISRAT では準ニュートン法 [22] とネルダー・ミード法 [23] の最適化アルゴリズムが利用できるが，パラメータ推定時に問題を生じることがある．それはすなわち，パラメータの非負制約違反が多発する，局所解が多数存在し，パラメータ推定時に設定した初期値に収束値が大きく依存するため，大域的な最適化が難しい，などの問題である．したがって，本論文では共変量関数をカーネル回帰 [20,21] によってノンパラメトリックに推定することで，これらの問題点を改善することを考える．

3.　カーネル回帰

いま，2 つの入力 $\boldsymbol{x} = (x_1,\cdots,x_d)^{\mathrm{T}}$ および $\boldsymbol{x}' = (x'_1,\cdots,x'_d)^{\mathrm{T}}$ に対して，カーネル関数

$$k(\boldsymbol{x},\boldsymbol{x}') = \exp\left(-\beta\left\|\boldsymbol{x}-\boldsymbol{x}'\right\|^2\right) \tag{21}$$

を考える．ただし，$\boldsymbol{z} = (z_1, \cdots, z_d)^{\mathrm{T}}$ に対して $\|\boldsymbol{z}\|^2 = \sum_{m=1}^{d} z_m^2$ とする．また，$\beta\ (>0)$ はカーネルパラメータと呼ばれ，学習データに応じて適当に決めておくパラメータである．学習データであるサンプル $\boldsymbol{x}^{(j)}\ (j = 1, 2, \cdots, n)$ が得られるとき，カーネル回帰では与えられた $\boldsymbol{x}$ に対して

$$y = \sum_{j=1}^{n} \alpha_j k\left(\boldsymbol{x}^{(j)}, \boldsymbol{x}\right) \tag{22}$$

なる関数を当てはめる．すなわち，与えられた $\boldsymbol{x}$ に対して，各サンプル $\boldsymbol{x}^{(j)}$ との距離 $k\left(\boldsymbol{x}^{(j)}, \boldsymbol{x}\right)$ をひとつの成分と見て，それらを重み α_j で足しあわせる．カーネルパラメータ β が大きいときには，各サンプルに対するカーネル関数一つひとつがより尖った形状となり，得られる解はサンプル間において大きく振動する傾向がある．一方で β を小さくすれば，より幅広いカーネル関数を足しあわせることになるため，得られる解は滑らかになる．

式 (22) の重み $\boldsymbol{\alpha} = (\alpha_1,\ \alpha_2, \cdots, \alpha_n)^{\mathrm{T}}$ は，二乗誤差

$$r_k(y, \boldsymbol{x}, \boldsymbol{\alpha}) = \left(y - \sum_{j=1}^{n} \alpha_j k\left(\boldsymbol{x}^{(j)}, \boldsymbol{x}\right)\right)^2 \tag{23}$$

が最小になるように決定すればよい．このために，次のようなカーネル行列 K を定義する．

$$K = \begin{pmatrix} k\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}\right) & k\left(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}\right) & \cdots & k\left(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(1)}\right) \\ k\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right) & k\left(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(2)}\right) & \cdots & k\left(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(2)}\right) \\ \vdots & \vdots & \ddots & \vdots \\ k\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(n)}\right) & k\left(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(n)}\right) & \cdots & k\left(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n)}\right) \end{pmatrix} \tag{24}$$

このとき，二乗誤差の総和は

$$R_k(\boldsymbol{\alpha}) = (\boldsymbol{y} - K\boldsymbol{\alpha})^{\mathrm{T}} (\boldsymbol{y} - K\boldsymbol{\alpha}) \tag{25}$$

となる．ここで，$\boldsymbol{y} = (y_1, \cdots, y_n)^{\mathrm{T}}$ である．式 (25) を最小にする解は

$$\boldsymbol{\alpha} = K^{-1}\boldsymbol{y} \tag{26}$$

によって求められる．このとき，得られる関数は各サンプル点を誤差なく通る解になるが，過学習によりグラフは非常に不安定な振動を示す．したがって，これに関数の表現能力を抑えるための正則化項を加えた

$$R_{k,\lambda}(\boldsymbol{\alpha}) = (\boldsymbol{y} - K\boldsymbol{\alpha})^{\mathrm{T}} (\boldsymbol{y} - K\alpha) + \lambda\boldsymbol{\alpha}^{\mathrm{T}} K\boldsymbol{\alpha} \tag{27}$$

を最小にする $\boldsymbol{\alpha}$ を利用する．ただし，λ (>0) は正則化パラメータである．式 (27) を最小にする $\boldsymbol{\alpha}$ は，I_n を n 次の単位行列とすると

$$\boldsymbol{\alpha} = (K + \lambda I_n)^{-1} \boldsymbol{y} \tag{28}$$

によって求められる．λ が小さい時にはサンプルにより適合した解が得られるが，全体の構造を推測する能力，すなわち，汎化能力が小さくなる．一方で，λ を適度に大きくすれば，サンプルに当てはめる効果が多少弱められるが，汎化能力をもつ解が得られることになる．なお，λ を過度に大きくしてしまえば，式 (27) において，$\boldsymbol{\alpha}^T K \boldsymbol{\alpha}$ を最小化する $\boldsymbol{\alpha} = \mathbf{0}$ が解となるので，サンプルに当てはめる能力も汎化能力も持たないこととなる．

本論文では，カーネルパラメータ β，および正則化パラメータ λ を n-fold クロスバリデーション（CV = Cross validation; 交差確認法）[21] によって決定する．つまり，一つのサンプルをテストデータとして除いておき，残りの $n-1$ 個のデータで学習し，1 つのテストデータとの誤差を求める．同様にすべてのデータがテストデータとなるように繰り返して誤差を求め，それらの平均誤差を CV 誤差とする．この CV 誤差が最小となるような β と λ を採用する．

4.　比例強度モデルに対するカーネル回帰

本論文では比例強度モデル [16] に基づいた次のような平均値関数を考える．

$$H_p(t_k; \boldsymbol{\theta}) = \sum_{i=1}^{k} g(\boldsymbol{x}_i) \int_{t_{i-1}}^{t_i} \lambda_0(u; \boldsymbol{\theta}) du \tag{29}$$

式 (29) において比例強度に相当する $g(\boldsymbol{x}_i)$ をカーネル回帰によってノンパラメトリックに推定する．具体的には以下の手順で平均値関数を推定する．

Step 1: フォールト検出個数データ $\boldsymbol{y}$ から NHPP のパラメータ $\boldsymbol{\theta}$ を最尤法で推定する．

Step 2: 次式により，ベースライン強度関数 $\hat{\lambda}_0(t_i)$ と実際の検出フォールト数との比率 π_i $(i = 1, 2, \cdots, n)$ を求める．

$$\pi_i = \begin{cases} \dfrac{y_i - y_{i-1}}{\hat{\lambda}_0(t_i)} & \text{if } y_i - y_{i-1} > 0 \\ \dfrac{\varepsilon}{\hat{\lambda}_0(t_i)}, \quad (\varepsilon > 0) & \text{if } y_i - y_{i-1} = 0 \end{cases} \tag{30}$$

Step 3: 共変量データ $\boldsymbol{x}_i$ 及び比率の自然対数 $\ln(\pi_i)$ から比例強度の自然対数 $\ln(\hat{g}(\boldsymbol{x}_i))$ をカーネル回帰によりノンパラメトリックに推定する．

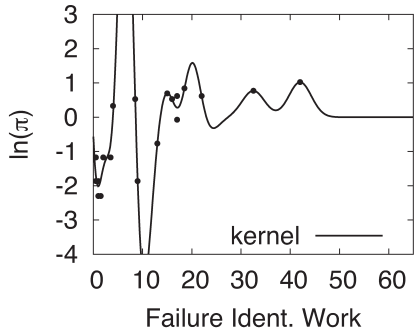

図 1: カーネル回帰による対数比例強度の近似

Step 4: 推定された比例強度 $\hat{g}(\boldsymbol{x}_i) = \exp(\ln(\hat{g}(\boldsymbol{x}_i)))$ を用いて，式 (29) により平均値関数を求める．

なお，式 (30) において $y_k - y_{k-1}$ は，テスト期間 $(t_{k-1}, t_k]$ において新たに検出されたフォールト数を意味し，検出フォールト数が 0 の場合には式 (30) の分子を ε とした．これは，Step 3 において $\ln(\pi_i)$ を計算する必要があり，この対数の真数条件を満足させるためのものである．また，π_i に対して直接的にカーネル回帰を適用することも可能であるが，この場合，推定された回帰曲線が部分的に負の値をとる可能性があり，最終的な累積フォールト数がその部分において減少するという問題が発生する．このため，本論文では，$\ln(\pi_i)$ に対してカーネル回帰を適用することとした．この結果，$\ln(\pi_i)$ の回帰曲線 $\ln(\hat{g}(\boldsymbol{x}_i))$ が一時的に負の値になったとしても，$\hat{g}(\boldsymbol{x}_i) = \exp(\ln(\hat{g}(\boldsymbol{x}_i))) > 0$ の条件は満たされるので，最終的な累積フォールト数の振る舞いは非減少関数になることが保証される．

次に，Musa [24] のフォールトデータを利用して，提案手法の推定手順を示す．ここで利用するデータは 21 週間のソフトウェアテストにおいて 136 個のフォールトを検出したものである．また，メトリクスデータとして Execution time (CPU hr), Failure identification work (Person hr), および Computer time failure identification (CPU hr) の 3 種類が利用可能であるが，ここでは Failure identification work を採用する．なお，**5.** の数値例においては 3 種類すべてのメトリクスデータを利用する．

まず，ベースライン平均値関数に指数形の NHPP モデル [3] を仮定し，21 週間のテストにおいて検出されたフォールト個数データから最尤法によって NHPP のパラメータ $\boldsymbol{\theta}$ を推定した．この結果，最尤推定値として $\hat{\boldsymbol{\theta}} = (\hat{a},\ \hat{b}) = (1.5395\mathrm{E}{+}06,\ 4.2051\mathrm{E}{-}06)$ が得られた．

次に，ベースライン強度関数 $\hat{\lambda}_0(t_i)$ と実際の検出フォールト数との比率 π_i を求め，横軸を Failure identification work，縦軸を $\ln(\pi_i)$ としてプロットする．なお，Failure identification work と $\ln(\pi_i)$ の相関係数は 0.848 となり強い相関があることが確認できる．さらに，クロスバリデーションによって得られたカーネル回帰に関するパラメータの最適解は $(\beta^*,\ \lambda^*) = (9.6416\mathrm{E}-02,\ 4.9138\mathrm{E}-04)$ となった．これらパラメータを用いてカーネル回帰を行うと，図 1 に示す解が得られた．

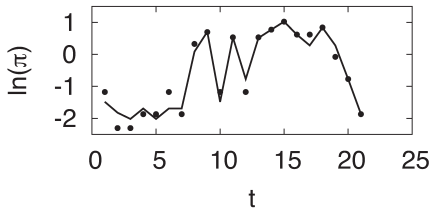

図 2: 横軸をテスト時刻とした対数比例強度

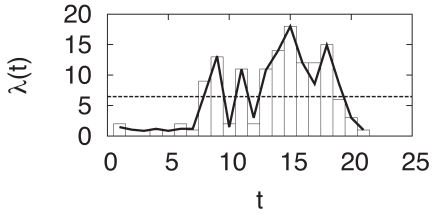

図 3: カーネル回帰によって得られた強度関数

図 1 の横軸をテスト時刻 t_k としてプロットしたものが図 2 であり，式 (16) によって求めた強度関数を図 3 に示す．カーネル回帰を用いてテスト時間区間に投入されるテスト労力量を考慮することで，図 3 に示すとおり，提案手法によって得られる各期の期待フォールト検出数は NHPP モデルと比較してデータにより適合することがわかる．この結果，図 4 に示す平均値関数が，従来の NHPP モデルと比較してより正確に表現されていることも確認できる．

また，図 1 の回帰曲線を利用することにより，Failure identification work 投入量の増減が，フォールト検出の振る舞いにどの程度の影響を与えるかを確認することが可能である．これは，テスト期間中において将来のフォールト検出の振る舞いを予測する際に利用できる．ただし，本論文で提案した手法においては，これまでに与えられた投入量に比較的近い範囲で

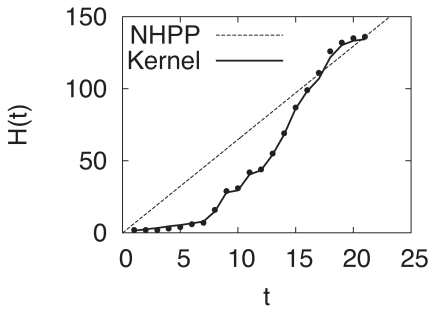

図 4: カーネル回帰によって得られた平均値関数

予測が可能であり，これまでの投入量から大幅に離れた範囲（例えば図 1 においてはおおむね 50 以上）では比例強度の自然対数が 0 となる．つまり比例強度が 1 となるため，予測発見個数はベースライン強度関数に一致することとなる．これは，仮にメトリクスの投入量を倍増させても，モデルが過剰な反応を示さないことを意味している．これは，予測において過剰な反応を示す傾向がある比例強度モデル [16] と大きく異なる特徴である．

5.　数値例

ここでは実際のリアルタイム制御システムの開発工程で実施されたテストにおいて観測された 4 種類のソフトウェアフォールト検出データ [24] を用いて提案手法の有効性を検証する．これらのデータにはテスト期間中の各週において検出されたフォールト数のほかに，次の 3 種類のメトリクスデータがある．

1. Execution Time (ET)
2. Failure Identification work (FI)
3. Computer Time-failure identification (CT)

表 1 にはフォールトデータの概要を示す．これら 4 種類のデータを用いて，NHPP モデルと提案モデルの過去のデータに対する適合性評価，および将来の未知のデータに対する予測評価を実施する．なお，提案モデルにおいては上記 3 種類のメトリクスデータをすべて利用する．また，ベースライン強度関数には指数形 (Exponential) モデル [3]

$$\lambda_0(t;a,b) = ab\exp(-bt)$$

および，S 字形 (S-Shape) モデル [4]

$$\lambda_0(t;a,b) = ab^2 t\exp(-bt)$$

を仮定する．

表 1: フォールトデータの概要

data set	week	faults	ET (CPU hr)	FI (person hr)	CT (CPU hr)
DS-1	21	136	38.0	227	100.3
DS-2	17	54	32.8	296	357.1
DS-3	14	38	21.5	137.1	30.3
DS-4	16	53	18.5	111.3	49.0

表 2: パラメータ推定値と平均二乗誤差

Case	$\hat{a}$	$\hat{b}$	MSE (NHPP)	MSE (Kernel)
DS-1 (Exponential)	1539472	4.205E−06	534.51	193.46
DS-1 (S-Shape)	859.34	0.0337	84.82	5.33
DS-2 (Exponential)	129.34	0.0318	25.51	28.86
DS-2 (S-Shape)	63.31	0.1999	10.01	0.06
DS-3 (Exponential)	48.84	0.1075	3.22	0.02
DS-3 (S-Shape)	40.07	0.3360	9.90	28.38
DS-4 (Exponential)	58.30	0.1499	12.01	0.60
DS-4 (S-Shape)	53.94	0.3744	8.64	0.92

5.1　適合性評価

表 2 には NHPP モデルのパラメータ推定値と NHPP 及び提案モデルの平均二乗誤差 (MSE) を示す．なお，Exponential は指数形 NHPP モデルを，S-Shape は遅延 S 字形 NHPP モデルを意味する．また図 5 から図 8 には各フォールトデータに対して NHPP モデルと提案モデルの平均値関数および強度関数の振る舞いを示す．

図 5 および表 2 より，NHPP モデルは DS-1 に対して指数形，S 字形で共に適合度が低く，実際のフォールト数との誤差が大きくなっている．これに対して提案モデルは S 字形モデルがフォールト数のデータによく適合していることが読み取れる．DS-2 に対しては図 6 より，

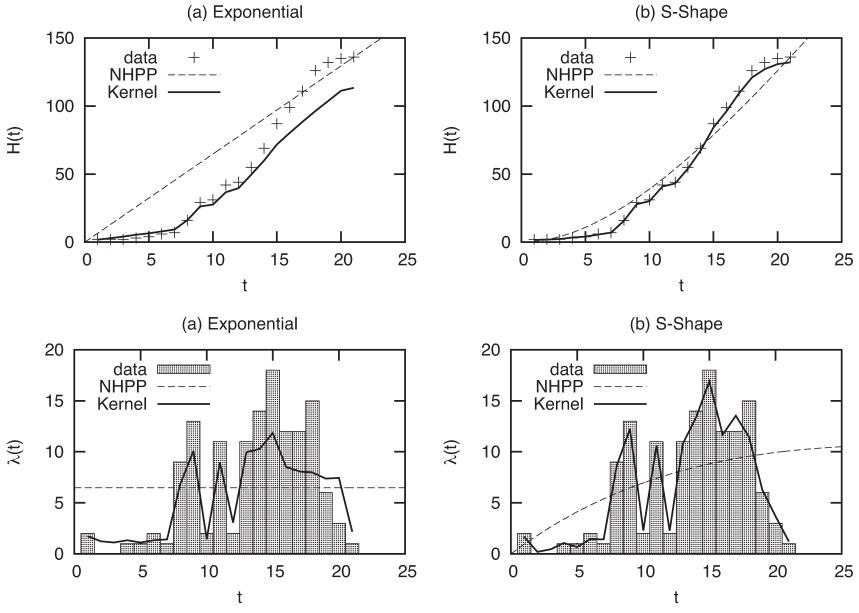

図 5: 平均値関数と強度関数の振る舞い (DS-1)

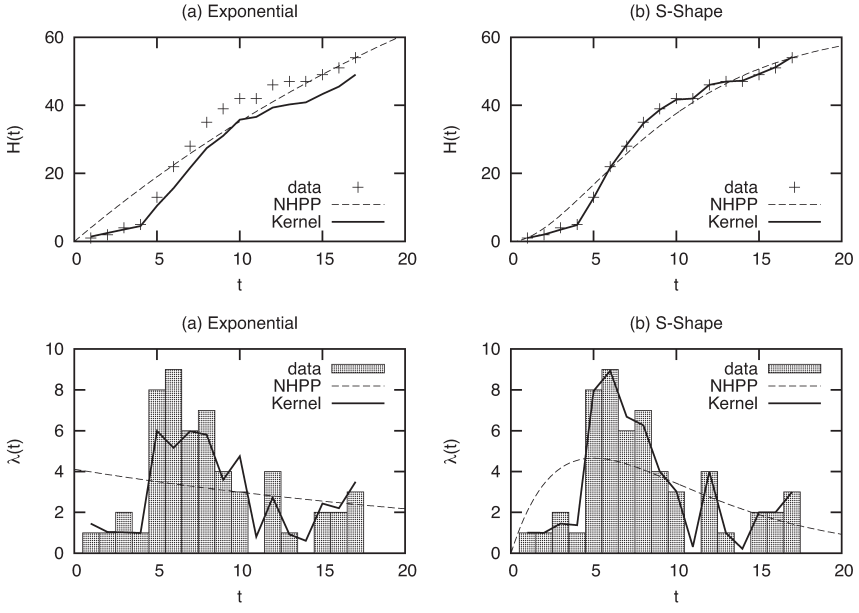

図 6: 平均値関数と強度関数の振る舞い (DS-2)

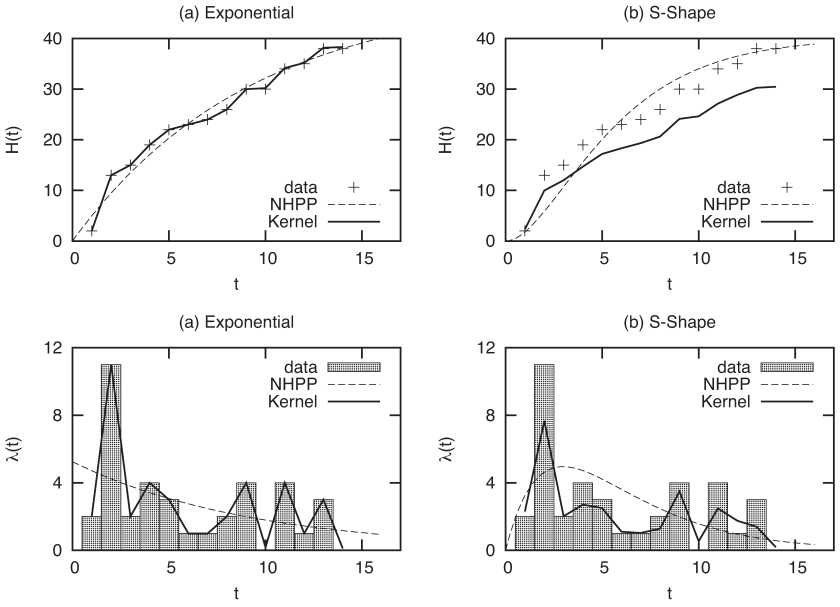

図 7: 平均値関数と強度関数の振る舞い (DS-3)

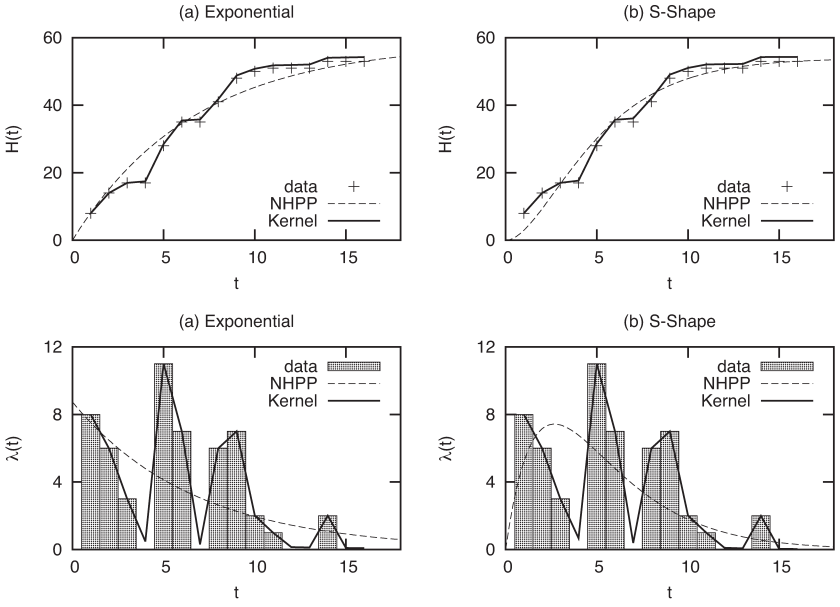

図 8: 平均値関数と強度関数の振る舞い (DS-4)

NHPP モデルは指数形モデルと比較して S 字形モデルがより適合していることがわかる．一方で提案モデルは S 字形モデルの適合度が更に高いことがわかる．図 7 より，DS-3 に対しては，NHPP モデル，提案モデル共に指数形モデルが S 字形モデルと比較してデータにより適合していることがわかる．表 2 より，指数形モデルに基づいた提案モデルは MSE が特に小さく，フォールト発見の不規則な挙動を正確に表現できていることが読み取れる．また，図 8 および表 2 より，NHPP モデルでは S 字形モデルの適合度が高いが，提案モデルでは指数形モデルの適合度が高いことが読み取れる．DS-4 に対しても提案モデルは NHPP モデルと比較してより高い適合度を示していることがわかる．

したがって，各データに対してベースラインのモデルを適切に選択することが出来れば，提案モデルは NHPP よりもより高い推定精度を有していることが示された．なお，表 2 および図 6, 7 より，提案モデルは DS-2 の指数形モデルと DS-3 の S 字形モデルにおいて，誤差が大きくなっている．これは提案モデルが各週で新たに検出されたフォールト数のデータに適合させようとすることに起因する．すなわち，例えば図 6 において，提案モデルの指数形モデルは 5, 6, 8 週目にフォールト数を少なく見積もっている．これにより，累積フォールト数の推定値がテスト期間最後まで少なく見積もられているからである．

5.2　予測評価

次にテスト期間中の任意の時点において，将来のフォールト検出の振る舞いを予測することを考え，その予測評価を行う．このために $n-5$ 週間分のデータを学習データとして，これを得た時点で NHPP モデルと提案モデルによってソフトウェアの信頼性を評価し，その後の 5 週間分のデータをテストデータとして予測二乗誤差 (PSE) を求めた．その結果を表 3 に示す．なお，提案モデルにおいて将来のフォールト検出の振る舞いを予測するためには，将来投入されるメトリクスデータが必要となる．このため，本論文では将来のメトリクスデータが既知であると仮定した．

表 3 および図 12 より，DS-4 に対してのみ，提案モデルよりも NHPP モデルがより高い予測精度を持っていることがわかる．これは前述の通り，提案モデルが 5,6,8,9 週目で検出フォールト数を少なく見積もっており，ここでの誤差が予測精度にも影響していることがわかる．一方で DS-1, DS-2, DS-3 に対しては，いずれの場合も提案モデルが NHPP モデルと比較してより高い予測精度を持っていることがわかる．

以上の結果，本論文で提案したカーネル回帰による比例強度モデルは，適合度および予測精度に関して従来の NHPP モデルと比較してより高い精度を持っていることが示された．

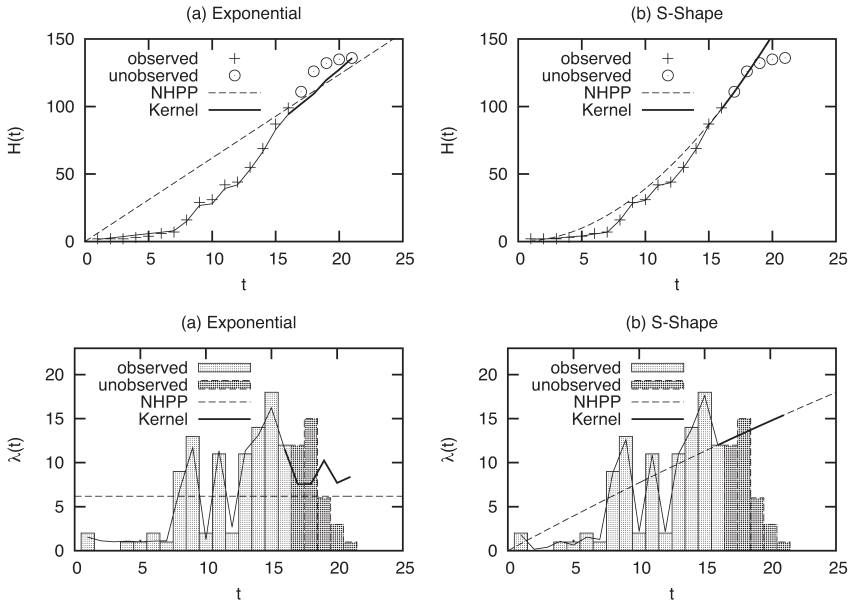

図 9: 予測平均値関数と強度関数の振る舞い (DS-1)

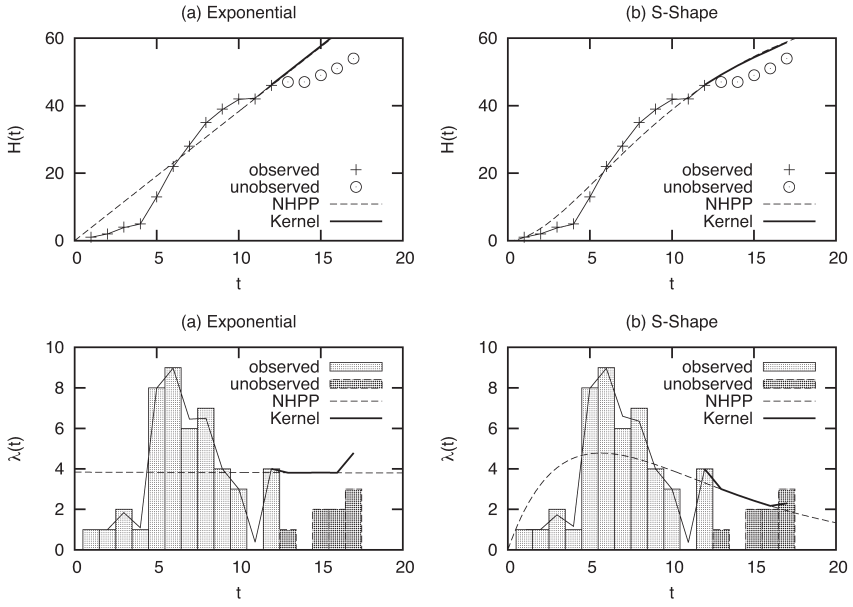

図 10: 予測平均値関数と強度関数の振る舞い (DS-2)

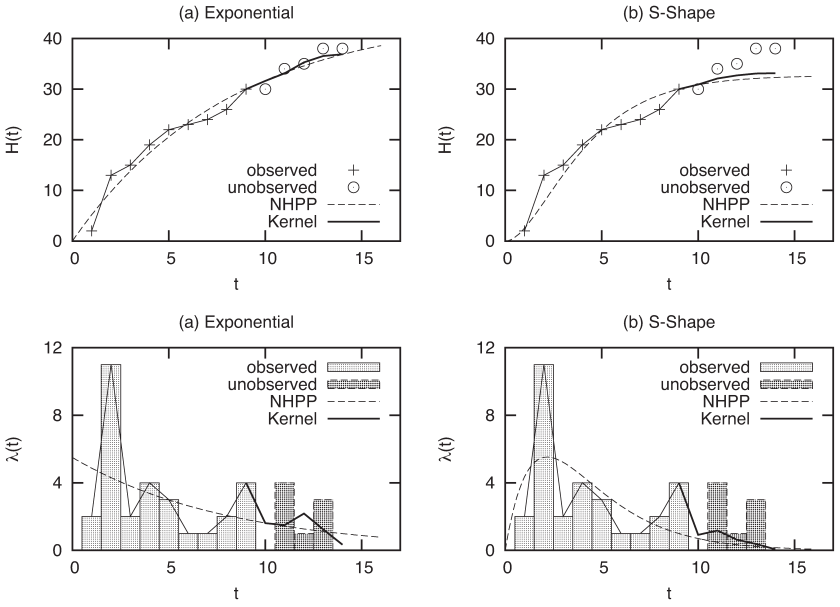

図 11: 予測平均値関数と強度関数の振る舞い (DS-3)

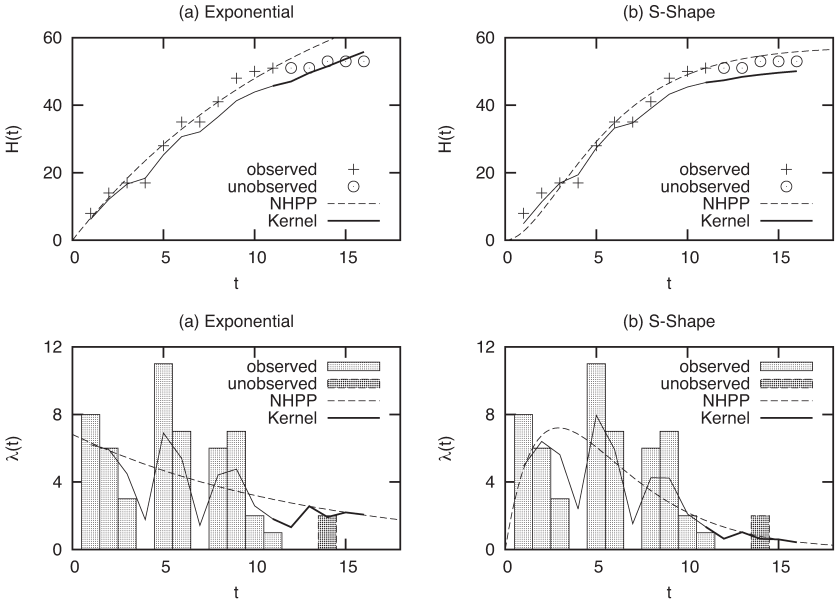

図 12: 予測平均値関数と強度関数の振る舞い (DS-4)

表 3: 予測平均二乗誤差

Case	MSE (NHPP)	MSE (Kernel)	PSE (NHPP)	PSE (Kernel)
DS-1 (Exponential)	5.78E+02	5.39E+00	125.07	113.89
DS-1 (S-Shape)	4.81E+01	2.47E−01	258.32	281.42
DS-2 (Exponential)	2.48E+01	2.83E−02	68.72	78.18
DS-2 (S-Shape)	1.04E+01	4.03E−02	23.92	21.76
DS-3 (Exponential)	3.83E+00	4.19E−10	2.00	1.39
DS-3 (S-Shape)	6.12E+00	1.66E−05	16.93	11.47
DS-4 (Exponential)	6.69E+00	1.57E+01	50.29	5.64
DS-4 (S-Shape)	1.12E+01	8.29E+00	5.06	11.10

6. むすび

ソフトウェアのテスト工程では種々のテスト労力が観測可能であり，本論文ではこれらテスト労力をメトリクスデータとして有機的に活用した比例強度ソフトウェア信頼性モデルを考え，これにカーネル回帰を適用した．具体的には，最尤推定時の推定パラメータ数を従来の NHPP モデルと同数にまで減少させ，比例強度モデルの比例定数に対応する項をカーネル回帰によってノンパラメトリックに推定した．これによって，比例強度モデルで問題となっていたパラメータ推定が容易になった．

また，数値例では実際のフォールトデータを利用して提案モデルと適合度と予測精度を検証した．これによって，提案モデルは従来の NHPP モデルと比較してより高い適合度，および予測精度を有していることが示された．

本論文では，ベースライン強度関数のパラメータを最尤法で推定した後に，カーネル回帰を適用したが，これらの手順を繰り返し，同時最適化を行うことは今後の課題である．また，数値例において 3 種類のメトリクスデータを利用したが，最適なメトリクスの組合せを決定する問題，すなわち変数選択や，ベースライン強度関数のモデル選択の基準を示すことも必要である．さらに，将来のメトリクスデータが未知である場合に対して，将来の信頼性予測評価を行うための手法を確立することも今後の課題である．

参考文献

[1] 山田茂，ソフトウェア信頼性モデル：基礎と応用，日科技連 (1994).

[2] 山田茂，ソフトウェア信頼性の基礎：モデリングアプローチ，共立出版 (2011).

[3] A.L. Goel and K. Okumoto, “Time-dependent error-detection rate model for software reliability and other performance measures,” *IEEE Trans. Reliability*, vol.R-28, no.3, pp.206–211 (1979).

[4] S. Yamada, M. Ohba and S. Osaki, “S-shaped reliability growth modeling for software error detection,” *IEEE Trans. Reliability*, vol.R-32, no.5, 475–478 (1983).

[5] J.D. Musa, A. Iannino and K. Okumoto, *Software Reliability Measurement, Prediction, Application*, McGraw-Hill, New York (1987).

[6] M. Lyu (ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill, New York (1996).

[7] H. Pham, *Software Reliability*, Springer-Verlag, London (2000).

[8] H. Pham, *System Software Reliability*, Springer-Verlag, London (2010).

[9] S. Yamada, H. Ohtera and H. Narihisa, “Software reliability growth models with testing-effort,” *IEEE Trans. Reliability*, vol.R-35, no.1, pp.19–23 (1986).

[10] 山田茂，“テスト労力の投入量を考慮したソフトウェア信頼度成長モデルと適合性比較,” 電子情報通信学会論文誌，vol.J70-D, no.12, pp.2438–2445 (1987).

[11] S. Yamada, H. Ohtera and H. Narihisa, “A testing-effort dependent reliability model for computer programs,” *Trans. IECE Japan*, vol.E69, no.11, pp.1217–1224 (1986).

[12] S. Yamada, J. Hishitani and S. Osaki, “Software reliability growth with a Weibull testing-effort: A model & application,” *IEEE Trans. Reliability*, vol.R-42, no.1, pp.100–106 (1993).

[13] P.K. Kapur, V.B. Singh, A. Sameer and V.S.S. Yadavalli, “Software reliability growth model with change-point and effort control using a power function of the testing time,” *International Journal of Production Research*, vol.40, no.3, pp.771–787 (2008).

[14] L.S. Dharmasena, P. Zeephongsekul and C.L. Jayasinghe, “Software reliability growth models based on local polynomial modeling with kernel smoothing,” *Proceedings of IEEE International Symposium on Software Reliability Engineering,* pp.220–229, IEEE Computer Society Press (2011).

[15] B.W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London (1986).

[16] K. Rinsaka, K. Shibata and T. Dohi, “Proportional intensity-based software reliability modeling with time-dependent metrics,” *Proceedings of IEEE International Computer Software and Applications Conference*, pp.369–376, IEEE Computer Society Press (2006).

[17] K. Shibata, K. Rinsaka and T. Dohi, “Metrics-based software reliability models using non-homogeneous Poisson processes,” *Proceedings of The 17th IEEE International Symposium on Software Reliability Engineering*, pp.52–61, IEEE Computer Society Press (2006).

[18] K. Shibata, K. Rinsaka and T. Dohi, “PISRAT: Proportional intensity-based software reliability assessment tool,” *Proceedings of 13th Pacific Rim International Symposium on Dependable Computing*, pp.43–52, IEEE Computer Society Press (2007).

[19] H. Okamura, Y. Etani and T. Dohi, “A multi-factor software reliability model based on logistic regression,” *Proceedings of IEEE 21st International Symposium on Software Reliability Engineering*, pp.31–40, IEEE Computer Society Press (2010).

[20] J. Shaw-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press (2004).

[21] 赤穂昭太郎，カーネル多変量解析：非線形データ解析の新しい展開，岩波書店 (2008).

[22] 茨木俊秀，最適化の数学，共立出版 (2011).

[23] J.A. Nelder and R. Mead, “A simplex method for function minimization,” *Computer Journal*, vol.7, pp.308–315 (1965).

[24] J.D. Musa, *Software Reliability Data, Technical Report, Data and Analysis Center for Software*, Rome Air Development Center (1979).
(http://www.thedacs.com/get_pdf/DACS-2248.pdf)