Whole-History Rating: A Bayesian Rating System for Players of Time-Varying Strength

Rémi Coulom

Whole-History Rating: A Bayesian Rating System for Players of Time-Varying Strength

Rémi Coulom

Université Charles de Gaulle, INRIA SEQUEL, CNRS GRAPPA, Lille, France

Abstract. Whole-History Rating (WHR) is a new method to estimate the time-varying strengths of players involved in paired comparisons. Like many variations of the Elo rating system, the whole-history approach is based on the dynamic Bradley-Terry model. But, instead of using incremental approximations, WHR directly computes the exact maximum a posteriori over the whole rating history of all players. This additional accuracy comes at a higher computational cost than traditional methods, but computation is still fast enough to be easily applied in real time to large-scale game servers (a new game is added in less than 0.001 second). Experiments demonstrate that, in comparison to Elo, Glicko, TrueSkill, and decayed-history algorithms, WHR produces better predictions.

1 Introduction

Institutions that organize competitive activities, such as sports or games, often rely on ratings systems. Rating systems provide an estimation of the strength of competitors. This strength estimation makes it possible to set up more balanced matches, motivate competitors by providing them with a measurement of their progress, and make predictions about the outcomes of future competitions.

Almost every institution designed its own rating system, so many algorithms exist. The following discussion summarizes the main kinds of rating systems.

Static Rating Systems. Static rating systems do not consider the variation in time of the strengths of players. They are appropriate for rating humans over a short period of time, or for rating computers. An effective method for a static rating system consists in using Bayesian inference with the Bradley-Terry model [1]. But static rating systems are not adapted to situations where players may make significant progress.

Incremental Rating Systems. Incremental rating systems, such as the FIDE rating system [4], Glicko [7], or TrueSkill [8] store a small amount of data for each player (one number indicating strength, and sometimes another indicating uncertainty). After each game, this data is updated for the participants in the game. The rating of the winner is increased, and the rating of the loser is decreased.

Incremental rating systems can handle players of time-varying strength, but do not make optimal use of data. For instance, if two players, A and B, enter the rating system at the same time and play many games against each other, and none against established opponents, then their relative strength will be correctly estimated, but not their strength with respect to the other players. If player A then plays against established opponents, and its rating changes, then the rating of player B should change too. But incremental rating systems would leave B's rating unchanged.

Decayed-history Rating Systems. In order to fix the deficiencies of incremental rating systems, a static rating algorithm may be applied, limited to recent games. This idea may be refined by giving a decaying weight to games, either exponential or linear[1]. With this decay, old games are progressively forgotten, which allows to measure the progress of players.

This decayed-history approach solves some problems of incremental rating systems, but also has some flaws. The main problem is that the decay of game weights generates a very fast increase in the uncertainty of player ratings. This is unlike incremental systems that assume that rating uncertainty grows like the square root of time. With the decayed-history approach, players who stop playing for a while may experience huge jumps in their ratings when they start playing again, and players who play very frequently may have the feeling that their rating is stuck. If players don't all play at the same frequency, there is no good way to tune the speed of the decay.

Accurate Bayesian Inference. Another approach that may be more theoretically sound than decayed history consists in using the same model as incremental algorithms, but with less approximations. The weakness of algorithms like Glicko and TrueSkill lies in the inaccuracies of representing the probability distribution with just one value and one variance for every player, ignoring covariance. Authors of incremental algorithms already proposed to correct inaccuracies by running several passes of the algorithm forward and backward in time [10, 5, 7, 2]. Edwards [3], with Edo ratings, proposed a method to directly estimate the maximum a posteriori on the exact model.

The WHR algorithm presented in this paper is similar in principle to Edo ratings, although the numerical method is different (Edo uses MM [9], whereas WHR uses the more efficient Newton's method). On his web site, Edwards wrote that "Edo ratings are not particularly suitable to the regular updating of current ratings". Experiments presented in this paper clearly indicate that he underestimated his idea: evaluating ratings of the past more accurately helps to evaluate current ratings: the prediction rate obtained with WHR outperforms decayed history and incremental algorithms. Also, the WHR algorithm allows to rapidly update ratings after the addition of one game, making it suitable for large-scale real-time estimation of ratings.

[1] This idea is often credited to Ken Thompson (for instance, by Sonas [14]), but I could not find a more precise reference

Paper Outline. Section 2 presents the dynamic Bradley-Terry model, Section 3 is the WHR algorithm, and Section 4 presents experiment results on data of the KGS Go server.

2 The Dynamic Bradley-Terry Model

This section briefly presents the dynamic Bradley-Terry model [6] that is the basis for the WHR system.

2.1 Notations

- Player number: $i \in \{1, \ldots, N\}$, integer index
- Elo rating of player i at time t: $R_i(t)$, real number.
- γ rating of player i at time t: $\gamma_i(t)$, defined by $\gamma_i(t) = 10^{\frac{R_i(t)}{400}}$.
- Natural rating of player i at time t: $r_i(t) = \ln \gamma_i(t) = R_i(t) \frac{\ln 10}{400}$.

Elo ratings are familiar to chess players, but are on a rather arbitrary and inconvenient scale. Natural ratings will be used most of the time in this paper, because they make calculations easier.

2.2 Bradley-Terry Model

The Bradley-Terry model for paired comparisons gives the probability of winning a game as a function of ratings:

$$P(\text{player } i \text{ beats player } j \text{ at time } t) = \frac{\gamma_i(t)}{\gamma_i(t) + \gamma_j(t)} \ .$$

The Bradley-Terry model may be generalized to handle draws, advantage of playing first, teams, and multi-player games [9]. In this paper, only the simple formulation will be used, but it would be straightforward to generalize the WHR algorithm to those more complex situations.

2.3 Bayesian Inference

The principle of Bayesian Inference consists in computing a probability distribution over player ratings ($\boldsymbol{\gamma}$) from the observation of game results ($\boldsymbol{G}$) by inverting the model thanks to Bayes formula:

$$p(\boldsymbol{\gamma}|\boldsymbol{G}) = \frac{P(\boldsymbol{G}|\boldsymbol{\gamma})p(\boldsymbol{\gamma})}{P(\boldsymbol{G})} \ .$$

In this formula, $p(\boldsymbol{\gamma})$ is a prior distribution over $\boldsymbol{\gamma}$ (lower-case p is used for probability densities), and $P(\boldsymbol{G})$ is a normalizing constant. $P(\boldsymbol{G}|\boldsymbol{\gamma})$ is the Bradley-Terry model described in the previous section. $p(\boldsymbol{\gamma}|\boldsymbol{G})$ is called the posterior distribution of $\boldsymbol{\gamma}$. The value of γ that maximizes $p(\boldsymbol{\gamma}|\boldsymbol{G})$ is the maximum a posteriori, and may be used as an estimation of the strengths of players, derived from the observation of their game results.

2.4 Prior

In the dynamic Bradley-Terry model, the prior has two roles. First, a prior probability distribution over the range of ratings is applied. This way, the rating of a player with 100% wins does not go to infinity. Also, a prior controls the variation of the ratings in time, to avoid huge jumps in ratings.

In the dynamic Bradley-Terry model, the prior that controls the variation of ratings in time is a Wiener process:

$$r_i(t_2) - r_i(t_1) \sim \mathcal{N}\big(0, |t_2 - t_1|w^2\big) \ .$$

w is a parameter of the model, that indicates the variability of ratings in time. The extreme case of $w = 0$ would mean static ratings.

Some realizations of a Wiener process are plotted on Figure 1. The Wiener process is a model for Brownian motion, and can be simulated by adding an independent normally-distributed random value at each time step. This means that the variance increases linearly with time, so the confidence interval grows like the square root of time. A Wiener process is said to be memoryless (or Markovian), that is to say, if $t_1 < t_2 < t_3$,

$$p\big(r(t_3)\big|r(t_1), r(t_2)\big) = p\big(r(t_3)\big|r(t_2)\big) \ ,$$
$$p\big(r(t_1)\big|r(t_2), r(t_3)\big) = p\big(r(t_1)\big|r(t_2)\big) \ .$$

Fig. 1. Three realizations of a Wiener process, with $r(0) = 0$. The dashed line indicates the 95% confidence interval for $p(r(t)|r(0) = 0)$.

3 Algorithm

The WHR algorithm consists in computing, for each player, the $\gamma(t)$ function that maximizes $p(\boldsymbol{\gamma}|\boldsymbol{G})$. Once this maximum a posteriori has been computed, the variance around this maximum is also estimated, which is a way to estimate rating uncertainty.

(a) History of a player (b) optimizing one by one (c) Newton's method

Fig. 2. Newton's method applied to the history of one player. (a) The rating history of a player who has played 4 games is defined by 4 ratings r_1, r_2, r_3, and r_4, at the four times of the four games. (b) Two ratings that are close in time, such as r_2 and r_3, are strongly correlated. So, methods such as MM [9] that optimize parameters one by one, are very inefficient. (c) Since the optimized function is very similar to a quadratic form, Newton's method is extremely efficient.

3.1 Optimization Method

The first step of the WHR algorithm consists in computing the maximum a posteriori for all the $\gamma_i(t)$. Since γ_i are functions of time, this is an infinite-dimensional optimization problem. But it is easy to reduce it to a finite-dimensional problem, since knowing the values of γ at the times of games is enough. Rating estimation between two consecutive games may be done with interpolation formulas provided in Appendix C.

Figure 2 illustrates how optimization is performed with Newton's method. Since the Wiener process is Markovian, the Hessian matrix $\left(\frac{\partial^2 \log p}{\partial \boldsymbol{r}^2}\right)$ is tridiagonal, so Newton's method has a cost linear in the number of ratings. Formally, Newton's method consists in updating the rating vector $\boldsymbol{r}$ of one player (the vector of ratings at times when that player played a game) according to this formula:

$$\boldsymbol{r} \leftarrow \boldsymbol{r} - \left(\frac{\partial^2 \log p}{\partial \boldsymbol{r}^2}\right)^{-1} \frac{\partial \log p}{\partial \boldsymbol{r}}$$

The complete details of how to perform this update are provided in Appendices A and B. Since p is very similar to a Gaussian, $\log p$ is very similar to a quadratic form, so only one iteration of Newton's method is enough to get a very good value of $\boldsymbol{r}$. The overall optimization algorithm consists in applying this Newton update to every player in turn, and iterate until convergence.

3.2 Incremental Updates

The global optimization algorithm described in the previous section may take a few minutes to converge on a big database of games (see Section 4). So, it may be too slow to restart the algorithm from scratch in order to estimate new ratings when one new game is added to the database.

In order to let WHR work in real time when one new game is added, a simple solution consists in keeping the rating estimations obtained before the addition, and applying Newton's method once to every player of this game. This is orders of magnitude faster, although a little less accurate. If computation time allows,

more iterations of Newton's method may be applied from time to time. For instance, in experiments of incremental WHR described in the next section, one iteration was run on every player every 1000 games.

3.3 Estimating Rating Uncertainty

Once the maximum a posteriori has been found, rating uncertainty may be estimated. Since the posterior probability is similar to a Gaussian, its covariance matrix may be approximated by the opposite of the inverse of the Hessian.

In practice it is not possible to compute the whole covariance matrix for all the parameters at the same time. But rating uncertainty of one player can be estimated by considering the Hessian of the ratings of this player only, assuming opponents have a fixed rating equal to the maximum a posteriori. The detailed formulas for this operation can be found in Appendix B.2.

This numerical algorithm is extremely similar to a technique developed by physicists to estimate functions from noisy observations [11, 12].

4 Experiments in the Game of Go

4.1 Speed of Convergence

The WHR algorithm was tested on the database of games of the KGS Go server. This database contains all rated games since 2000 and until October, 2007. It contains 213,426 players, and 10.8 million games. Computations were performed on an Intel Core2 Duo at 2.4 GHz (using only one thread), and took about 7 minutes for 200 iterations. The prior was one virtual win and one virtual loss against a virtual player of rating zero, on the day of the first game. The Wiener process had a variance of $w^2 = 60$ Elo^2 per day.

Figure 3 shows the speed of converge of the log-likelihood. Figure 4 shows the result of player Crazy Stone, who made a lot of progress in year 2007. This figure shows that rating uncertainty increases slowly during long periods of inactivity.

4.2 Prediction Ability

The prediction ability of WHR was compared experimentally with the basic Elo algorithm [4], TrueSkill [8], Glicko [7], Bayeselo [1], and decayed history. Bayeselo may be considered as a special case of WHR, with $w^2 = 0$, or a special case of decayed history, with an infinitely long decay. Decayed history was implemented with an exponential decay, that is to say each game was weighted with a coefficient $e^{(t-t_0)/\tau}$, where τ is a parameter of the algorithm.

Bayeselo, WHR and decayed history all used the same scheme for incremental update of ratings: after each addition of a game, the ratings of the two participants were optimized with one step of Newton's method. Before each prediction of the outcome of a game, the ratings of the two participants were optimized, too. Every 1000 games, the ratings of all participants were optimized, one by one.

Fig. 3. Speed of optimization. One point is plotted every 10 iterations. Log-likelihood is the difference between the current log-likelihood and the initial log-likelihood, when all ratings were set to zero.

Fig. 4. Whole-History Rating estimation of player CrazyStone. Dots indicate days when games were played.

The method to compare algorithms consisted in measuring their prediction rates over a database of games. The prediction rate is the proportion of games whose most likely winner was predicted correctly (when two players have the same rating, this counts as 0.5 correct prediction). Parameters of the algorithm were first tuned to optimize prediction rate over a training database, then prediction rate was measured on a different test database.

The training set was made of the 726,648 rated even games with komi 6.5 played on KGS between 2000-11-07 and 2005-05-20. The test set consisted of the 2,331,757 rated even games with komi 6.5 played between 2005-05-21 and 2007-10-01. The time resolution of the database is one day, so if a player played several games in one day, they were considered as simultaneous.

Results are summarized in Table 1. It shows that WHR significantly outperforms the other algorithms. Algorithms that remember all the game results outperform the fast incremental methods.

Table 1. Prediction performance of some rating algorithms. Prior = 1 means one virtual win and one virtual loss against a player of rating zero. 95% confidence of superiority is obtained with a difference in prediction rate of 0.163% in the training set, and 0.091% in the test set. Since the same data was used to measure the performances of all algorithms, there is some variance reduction, so differences may be more significant. Time was measured on the training set.

Algorithm	Time	Training	Test	Optimal parameters
Elo	0.41 s	56.001%	55.121%	$k = 20$
Glicko	0.73 s	56.184%	55.522%	$\sigma_0 = 150$ Elo, $w^2 = 20$ Elo^2/day
TrueSkill	0.40 s	56.212%	55.536%	$\beta^2 = 1$, $\sigma_0^2 = 0.5$, $w^2 = 0.000975$/game
Bayeselo	88.66 s	56.216%	55.671%	prior = 1
Decayed history	89.86 s	56.260%	55.698%	prior = 1, $\tau = 400$ days
WHR	252.00 s	56.356%	55.793%	prior = 1.2, $w^2 = 14$ Elo^2/day

Performance on the test set is inferior to performance on the training set. This probably cannot be explained by overfitting alone. Because these two sets of games correspond to different periods of time, they don't have the same statistical properties. It may be that the KGS rating system improved, and since matches are automatically balanced by the server, recent games are more balanced, so they are more difficult to predict.

A remarkable aspect of these results is that parameters that optimize prediction rate give a very low variance to the Wiener process. The static Bayeselo algorithm even outperformed incremental algorithms on the test set. This is surprising, because many players on KGS are beginners, and made very big progress during the 2.5 years of the test set. The high number of experienced players probably outweighed beginners. This is, by the way, an important inaccuracy in the dynamic Bradley Terry model: it does not take those different abilities to make progress into consideration.

5 Conclusion

WHR is a new rating algorithm that directly computes player ratings as a function of time. It is computationally more costly than incremental and decayed-history algorithms, but more accurate, and fast enough to be applied in real time to large-scale game servers.

A research direction would be to apply WHR to more data sets, and compare it empirically to more alternative rating algorithms, such as Edo, and TrueSkill Through Time (TTT). It is likely that WHR would not outperform Edo and TTT in terms of prediction rate, since their models are almost identical. But the main difference may be in computation time. Previous experiments with Edo and TTT were run with a time resolution of one year, whereas WHR operates with a time resolution of one day. Such a short time period between ratings induces a very strong correlation between parameters, which Newton's method may handle more efficiently than MM or approximate message passing.

Another research direction would be to improve the model. An efficient application of WHR to Go data would require some refinements of the dynamic Bradley-Terry model, that the KGS rating algorithm [13] already has. In particular, it should be able to

- take handicap, komi, and time control into consideration,
- deal with outliers,
- handle the fact that beginners make faster progress than experts.

Acknowledgments

I thank William Shubert for providing KGS data. I am also very grateful to the reviewers, whose remarks helped to improve this paper a lot.

References

1. Rémi Coulom. Bayeselo. http://remi.coulom.free.fr/Bayesian-Elo/, 2005.
2. Pierre Dangauthier, Ralf Herbrich, Tom Minka, and Thore Graepel. TrueSkill through time: Revisiting the history of chess. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, Vancouver, Canada, 2007. MIT Press.
3. Rod Edwards. Edo historical chess ratings. http://members.shaw.ca/edo1/, 2004.
4. Arpad E. Elo. *The Rating of Chessplayers, Past and Present.* Arco Publishing, New York, 1978.
5. Ludwig Fahrmeir and Gerhard Tutz. Dynamic stochastic models for time-dependent ordered paired comparison systems. *Journal of the American Statistical Association*, 89(428):1438–1449, December 1994.
6. Mark E. Glickman. *Paired Comparison Model with Time-Varying Parameters.* PhD thesis, Harvard University, Cambridge, Massachusetts, 1993.
7. Mark E. Glickman. Parameter estimation in large dynamic paired comparison experiments. *Applied Statistics*, 48:377–394, 1999.
8. Ralf Herbrich and Thore Graepel. TrueSkill$^{\text{TM}}$: A Bayesian skill rating system. Technical Report MSR-TR-2006-80, Microsoft Research, 2006.
9. David R. Hunter. MM algorithms for generalized Bradley-Terry models. *The Annals of Statistics*, 32(1):384–406, 2004.
10. Leonhard Knorr-Held. Dynamic rating of sports teams. *The Statistician*, 49(2):261–276, 2000.
11. George B. Rybicki and David G. Hummer. An accelerated lambda iteration method for multilevel radiative transfer. *Astronomy and Astrophysics*, 245(1):171–181, May 1991.
12. Georges B. Rybicki and William H. Press. Interpolation, realization, and reconstruction of noisy, irregularly sampled data. *The Astrophysical Journal*, 398:169–176, October 1992.
13. William M. Shubert. Details of the KGS rank system. http://www.gokgs.com/help/rmath.html, 2007.
14. Jeff Sonas. Chessmetrics. http://db.chessmetrics.com/CM2/Formulas.asp, 2005.

A Gradient and Hessian Matrix for One Player

A.1 Terms of the Bradley-Terry model

The result of one game G_j may be written as:

$$P(G_j) = \frac{A_{ij}\gamma_i + B_{ij}}{C_{ij}\gamma_i + D_{ij}} \ ,$$

where A_{ij}, B_{ij}, C_{ij}, and D_{ij} are constants that do not depend on γ_i.

$W(i)$ is the set of games that i won, and $L(i)$ the set of games that i lost. $r = \ln \gamma$, so $dr = \frac{d\gamma}{\gamma}$.

$$\ln P = \sum_{j \in W(i)} \ln(A_{ij}\gamma_i) + \sum_{j \in L(i)} \ln(B_{ij}) - \sum_j \ln(C_{ij}\gamma_i + D_{ij})$$

$$\frac{\partial \ln P}{\partial r_i} = |W(i)| - \gamma_i \sum_j \frac{C_{ij}}{C_{ij}\gamma_i + D_{ij}}$$

$$\begin{aligned}\frac{\partial^2 \ln P}{\partial r_i^2} &= -\gamma_i \left(\sum_j \frac{-C_{ij}^2 \gamma_i}{(C_{ij}\gamma_i + D_{ij})^2} + \frac{C_{ij}}{C_{ij}\gamma_i + D_{ij}} \right) \\ &= -\gamma_i \sum_j \frac{C_{ij} D_{ij}}{(C_{ij}\gamma_i + D_{ij})^2}\end{aligned}$$

A.2 Terms of the Wiener prior

These terms are added to the Hessian for every pair of consecutive ratings, with $\sigma^2 = |t_2 - t_1| w^2$.

$$\begin{aligned}\frac{\partial \ln p}{\partial r_i(t_1)} &= -\frac{r_i(t_1) - r_i(t_2)}{\sigma^2} \\ \frac{\partial^2 \ln p}{\partial r_i(t_1)^2} &= -\frac{1}{\sigma^2} \\ \frac{\partial^2 \ln p}{\partial r_i(t_1) \partial r_i(t_2)} &= \frac{1}{\sigma^2}\end{aligned}$$

B LU Decomposition of the Hessian matrix

$$H = (h_{ij}) = LU = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ a_2 & 1 & 0 & \dots & 0 \\ 0 & a_3 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & a_n & 1 \end{pmatrix} \begin{pmatrix} d_1 & b_1 & 0 & \dots & 0 \\ 0 & d_2 & b_2 & \dots & 0 \\ 0 & 0 & d_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & b_{n-1} \\ 0 & 0 & 0 & \dots & d_n \end{pmatrix}$$

Algorithm ($i \geq 2$):

$$\begin{aligned} d_1 &= h_{11} \\ b_1 &= h_{12} \\ a_i &= h_{ii-1}/d_{i-1} \\ d_i &= h_{ii} - a_i b_{i-1} \\ b_i &= h_{ii+1} \end{aligned}$$

B.1 Computing $H^{-1}G$

The problem is to find vector X so that $LUX = G$. The first step of the algorithm consists in finding Y so that $LY = G$ ($i \geq 2$):

$$\begin{aligned} y_1 &= g_1 \\ y_i &= g_i - a_i y_{i-1} \end{aligned}$$

Then, find X, so that $UX = Y$ ($i < n$):

$$\begin{aligned} x_n &= y_n/d_n \\ x_i &= (y_i - b_i x_{i+1})/d_i \end{aligned}$$

Since $h_{ii+1} > 0$ and $h_{ii} < -(h_{ii-1} + h_{ii+1})$, $-1 < a_i < 0$ and $d_i < -h_{ii+1}$, so this algorithm has no division by zero. In order to ensure numerical stability, 0.001 is subtracted to all diagonal elements of H.

B.2 Computing Diagonal and Sub-diagonal Terms of H^{-1}

The covariance matrix for the full history of one player is approximated by

$$\Sigma = -H^{-1} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{12} & \sigma_2^2 & \dots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1n} & \sigma_{2n} & \dots & \sigma_n^2 \end{pmatrix}$$

In order to compute the confidence envelope around rating histories, only the diagonal and sub-diagonal terms of the reverse of H are needed. This can be done in cost linear in n [11]. The trick consists in doing the UL decomposition as well as the LU decomposition:

$$H = (h_{ij}) = U'L' \begin{pmatrix} 1 & a'_1 & 0 & \dots & 0 \\ 0 & 1 & a'_2 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & a'_{n-1} \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} d'_1 & 0 & 0 & \dots & 0 \\ b'_2 & d'_2 & 0 & \dots & 0 \\ 0 & b'_3 & d'_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & b'_n & d'_n \end{pmatrix}$$

Algorithm ($i < n$):

$$\begin{aligned} d'_n &= h_{nn} \\ b'_n &= h_{nn-1} \\ a'_i &= h_{ii+1}/d'_{i+1} \\ d'_i &= h_{ii} - a'_i b'_{i+1} \\ b'_i &= h_{ii-1} \end{aligned}$$

In order to solve $HX = G$, first solve $U'Y' = G$ ($i < n$):

$$\begin{aligned} y'_n &= g_n \\ y'_i &= g_i - a'_i y'_{i+1} \end{aligned}$$

Then, solve $L'X = Y'$ ($i > 1$):

$$\begin{aligned} x_1 &= y'_1/d'_1 \\ x_i &= (y'_i - b'_i x_{i-1})/d'_i \end{aligned}$$

The i-th term of the diagonal of the inverse of H, σ_i^2, is x_i, computed with $g_j = \delta_{ij}$. It can be computed by using the two decompositions at the same time:

$$\begin{aligned} x_i &= (y_i - b_i x_{i+1})/d_i \\ x_{i+1} &= (y'_{i+1} - b'_{i+1} x_i)/d'_{i+1} \end{aligned}$$

Since $y_i = 1$, and $y'_{i+1} = 0$, we get ($i < n$):

$$\begin{aligned} \sigma_i^2 &= -x_i = d'_{i+1}/(b_i b'_{i+1} - d_i d'_{i+1}) \\ \sigma_n^2 &= -1/d_n \end{aligned}$$

Sub-diagonal elements may be computed with

$$\sigma_{ii-1} = -a_i \sigma_i^2$$

C Interpolation Formulas

The mean, μ, and variance, σ^2, of a Wiener process at time t may be interpolated between two times t_1 and t_2, assuming that the means at t_1 and t_2 are μ_1 and μ_2, and the covariance matrix is $\Sigma = \left(\begin{smallmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{smallmatrix}\right)$, with the following formulas:

$$\mu = \frac{\mu_1(t_2 - t) + \mu_2(t - t_1)}{t_2 - t_1}$$

$$\sigma^2 = \frac{(t_2 - t)(t - t_1)}{t_2 - t_1} w^2 + \frac{(t_2 - t)^2 \sigma_1^2 + 2(t_2 - t)(t - t_1)\sigma_{12} + (t - t_1)^2 \sigma_2^2}{(t_2 - t_1)^2}$$