Computation of ECG signal features
using MCMC modelling in software and
FPGA reconfigurable hardware

Timothy Bodisco¹, Jason D’Netto², Neil Kelson², Jasmine Banks¹
and Ross Hayward¹

¹ Science & Engineering Faculty, Queensland University Of Technology, Australia
   {timothy.bodisco,j.banks,r.hayward}@qut.edu.au
² High Performance Computing & Research Support, Queensland University Of
   Technology, Australia
   {j.dnetto,n.kelson}@qut.edu.au

January 31, 2014

Abstract

Computational optimisation of clinically important electrocardiogram (ECG) signal features, within a single heart beat, using a Markov-chain Monte Carlo method is undertaken. A detailed, efficient data-driven software implementation of an MCMC algorithm has been shown. Initially software parallelisation is explored and has been shown that despite the large amount of model parameter inter-dependency that parallelisation is possible. Also, an initial reconfigurable hardware approach is explored for future applicability to real-time computation on a portable ECG device, under continuous extended use.

1 Introduction

Electrocardiogram (ECG) signal analysis involves identification of heart rate and other clinically relevant features, such as: the QRS complex, T wave, and ST segment, which characterise peak values and time intervals within a single heart beat. An important scenario is where the subject’s ECG signal is continuously monitored (e.g. in a home or workplace setting) for at least 24 hours via a portable battery powered recording device. Detailed real time signal processing is not usually performed by the device but rather via data download and computer analysis after completion of the recording period. Such recording devices are referred to as Holter monitors [7].
Both time and power efficient computation is needed in order to realise a portable ECG monitor suitable for continuous extended use and real time signal processing with the capability of detecting a range of clinically relevant features beyond the usual heart rate and, less commonly, QRS complex detection. Additionally, a relatively high sampling frequency is needed to ensure sufficient signal quality. Currently, there does not appear to be a portable ECG monitor that matches these requirements. Many commercially available monitors run at an extremely low sampling frequency, such as 250 Hz, which has been recommended as a minimum for the detection of the ECG waveform QRS complex [8]. However, this low sampling frequency may preclude accurate detection of other features. Monitors that attempt to obtain additional clinically significant ECG signal characteristics typically operate at higher sampling frequency of 1 kHz and peak voltages of 6 to 8 mV [7]. However, the higher frequency devices on the market are portable devices suited for emergencies and are not suitable for constant use [3], and/or perform data recording only [11, 8]. A 1 kHz and low power device with a 24 hour battery life has been developed [12], but performs limited signal analysis using only amplitude and slope. Further, it has been shown [10] that the detection method used in this device, while using very little power, suffers greatly under noisy conditions and is insufficient for accurate characteristic detection.

In this work, we focus on a Markov-chain Monte Carlo (MCMC) based approach recently proposed by the Authors [2] which compares favourably with existing algorithms (e.g. [9, 10]) in modelling a typical healthy ECG signal trace, in the presence of extreme noise and some variability. Presented here is a work-in-progress toward the generation of a hardware based ECG monitor. This paper shows how to efficiently implement the MCMC model in software; power-efficient Field Programmable Gate Array (FPGA) reconfigurable hardware is also explored to determine the future applicability of this approach for real time computation on a portable battery powered device, operating for continuous periods of one or more days.

2 MCMC modelling

The ECG signal is modelled here as a Normal distribution with a time-varying mean and some precision (inverse of the variance) where the time-varying mean is a generic mathematical representation of the ECG signal from a typical adult heart beat. Feature identification is achieved by modelling the signal as a continuous, piecewise defined function over a data window $[0, T]$ that encapsulates the
beat of interest, and resolving change-point parameters in the model, denoted $\delta_i$, and the parameters associated with the amplitudes of the features, denoted $\alpha_i$. Owing to space limitations, the full details of the model and the standard computation are omitted; however, this information can be found in a corresponding paper by the Authors [2]. Briefly, Equation 1 illustrates the model:

$$s(t) = y \sim N(\mu(t) + DC, \tau)$$

$$\mu(t) = \begin{cases} 
\alpha_1 \cos \left( \frac{\pi}{\delta_1} t \right) + \beta & 0 \leq t < \delta_1 \\
-\alpha_2 \cos \left( \frac{\pi}{\delta_2-\delta_1} (t - \delta_1) \right) + \beta - \alpha_1 + \alpha_2 & \delta_1 \leq t < \delta_2 \\
-\alpha_3 \cos \left( \frac{\pi}{\delta_3-\delta_2} (t - \delta_2) \right) + \beta - \alpha_1 + 2\alpha_2 + \alpha_3 & \delta_2 \leq t < \delta_3 \\
\beta - \alpha_1 + 2\alpha_2 + 2\alpha_3 \cos \left( \frac{\pi}{\delta_4-\delta_3} (t - \delta_3) \right) + \beta - \alpha_1 + 2\alpha_2 + 2\alpha_3 & \delta_3 \leq t < \delta_4 \\
\vdots \\
-\alpha_{12} \cos \left( \frac{\pi}{T-\delta_{17}} (t - \delta_{17}) \right) + \alpha_{12} - 2\alpha_{11} & \delta_{17} \leq t < T 
\end{cases}$$

where other symbols include the time $t$, the DC voltage offset $DC$, and an initial offset $\beta$ for matching of the signal and model at the start of the data window.

Feature identification is achieved by resolving the model parameters in an iterative process using Markov-chain Monte Carlo (MCMC). This process works by proposing candidate values for each model parameter in a random walk and determining the probability of whether the candidate parameter is more or less likely than the current parameter [5]. For each parameter the candidate value is selected from a Normal distribution with the current parameter as the mean and some deviation. The probability of acceptance, $A$, is governed by Equation 2.

$$A = \min \left\{ 1, \frac{R(\theta^*)}{R(\theta^{m-1})} \frac{p(\theta^*|\cdot)}{p(\theta^{m-1}|\cdot)} \right\}$$

where, $m$ denotes the $m^{th}$ MCMC iteration and $\cdot$ denotes the full set of model parameters omitting the parameter of interest denoted as $\theta$. For computational efficiency the proposal ratio, $\frac{R(\theta^*)}{R(\theta^{m-1})}$, is assumed to be unity and

$$p(\theta|\cdot) \propto p(y|\theta; \cdot) p(\theta)$$

As $p(y|\theta; \cdot)$ represents the probability of the data given the model parameters, it is reliant on the data as a whole ($p(\theta)$ is the prior distribution for the parameter $\theta$). Therefore, the computation of the the probability of acceptance, $A$, is done with Equation 3.

$$A = \min \left( 1, \frac{\prod p(y_i|\mu^*_i, \tau)}{\prod p(y_i|\mu_i, \tau)} \frac{p(\theta^*)}{p(\theta)} \right)$$

3
From a computational stand-point, Equation 3 is problematic as the product of a series of probabilities will always tend to zero. This is easily overcome by exploiting the features of the logarithmic function, i.e.

\[
\log\left(\prod p(y_i|\mu_i^*, \tau)\prod p(y_i|\mu_i, \tau)\right) = \log\left(\prod p(y_i|\mu_i^*, \tau)\right) - \log\left(\prod p(y_i|\mu_i, \tau)\right)
\]

\[
= \sum \log(p(y_i|\mu_i^*, \tau)) - \sum \log(p(y_i|\mu_i, \tau))
\]

therefore,

\[
\prod p(y_i|\mu_i^*, \tau) = \exp\left(\sum \log(p(y_i|\mu_i^*, \tau)) - \sum \log(p(y_i|\mu_i, \tau))\right).
\]

In the format shown in Equation 4, the computation can be minimised by saving the value of the sum of log probabilities from the previously successful (candidate value accepted) iteration. Therefore, at each iteration only the sum of log probabilities using the candidate value to generate \(\mu^*\) need to be generated. This modelling approach is shown graphically in Figure 1.

3 Software design

A flowchart of the model algorithm suitable for software implementation is shown in Figure 1. There is an outer FOR loop for the number of iterations, inside this loop there is a sequential FOR loop for the calculation of each of the model parameters. After creating the new candidate values for the \(\delta_i\) parameters updates involve boundary checking—each \(\delta_i\) parameters is constrained, i.e. \(\delta_{i-1} < \delta_i < \delta_{i+1}\). With each the new candidate value (from model parameters: \(\delta_i, \alpha_i, \beta \) or \(DC\)), a new \(\mu\) is calculated (denoted \(\mu^*\)), and the sum of new log probabilities is determined, Equation 4 and therefore the probability of acceptance, \(A\), is determined, Equation 3. For the model parameter \(\tau\) only the sum of log probabilities is needed to be determined as changing \(\tau\) does not effect \(\mu\). As \(A\) is the probability of acceptance, if \(A\) is less than 1 then acceptance is determined by comparing to a random number sampled from a Uniform distribution \([0,1]\) (i.e. if \(A \geq \text{Unif}[0,1]\) accept the candidate parameter). The candidate value becomes the new value of the model parameter if the acceptance test passes.

3.1 Parallelism opportunities

The \(\delta_i\) parameter updates can not be parallelised because \(\delta_i\) is dependent on \(\delta_{i-1}\) and \(\delta_{i+1}\). However, most of the \(\alpha\) parameters are dependent on only a small
subset of the model parameters, and the nature of these dependencies allow for three parallel streams for parameter updating, as shown in Table 1. Each stream will require their own copy of the parameters while updating a given parameter, and synchronise the parameter they updated after the update is completed.
Table 1: Parameter Update Streams

| α1 | α5 | α11 |
| β  | α6 | α4  |
| α2 | α7 | α10 |
| α12| α8 | α3  |
| τ  | α9 | DC  |

3.2 Bottlenecks

For the software implementation, the computational bottlenecks in the algorithm include performing mathematical operations of cosine, $e^x$, $x^y$, log $x$, and $\Gamma(x)$ [6] within the parameter updates. The rest of the software implementation involves relatively less costly addition, subtraction, multiplication, division, branching, assignment, loops, Boolean logic and reading/writing files.

4 FPGA design

A FPGA comprises reconfigurable hardware capable of performing multiple asynchronous operations [4] which can be leveraged to obtain accelerated computation of functions with many operations. For the present initial study, an initial FPGA hardware design informed by the MCMC model and the software design implementation is attempted. To this end, an examination of the computational aspects of the MCMC model and software implementation is useful, and some observations can be made in this regard. Only one $\delta_i$ variable is calculated at once, as $\delta_i$ is dependent on $\delta_{i-1}$ and $\delta_{i+1}$, where $\delta_0 = 0$ and $\delta_{18} = T$. Each $\delta_i$ equation can be considered of the form $\alpha_{multiply} \cos \left( \frac{\pi}{\delta_{high} - \delta_{low}} (t - \delta_{low}) \right) + \alpha_{addition}$, where each $\cos \left( \frac{\pi}{\delta_{high} - \delta_{low}} (t - \delta_{low}) \right)$ can be treated separately and input with the appropriate $\delta_{high}$ and $\delta_{low}$ values depending on the value of $t$ to produce the desired output. Most of the $\delta_i$ equations have different $\alpha_{addition}$ combinations and $\alpha_{multiply}$ is a single variable except for $\delta_4$ and $\delta_{13}$. Also, for $\delta_6$, $\delta_7$, $\delta_8$, $\delta_{15}$, $\delta_{16}$ and $\delta_{17}$, $\alpha_{addition} = \alpha_{multiply}$ with the appropriate $\alpha$ value used depending on the value of $t$.

Figure 2 shows a simplified diagram of the proposed method to create the model mean for any given parameter $\alpha$, $\beta$ and $DC$ values from Equation 1. The $\delta$ bus is a bus of all the current $\delta$ values. The $\delta_{select}$ bus is the binary encoding for which equation is to be active, and which $\delta$ values are $\delta_{high}$ and $\delta_{low}$. 

6
Figure 2: A simplified diagram of the VHDL for (a) the model equation and (b) the $\delta_7$ and $\delta_{16}$ equations.

The enable bus uses the $\delta_{select}$ bus to switch on only one $\delta$ equation at a time. For single clock tick computation, cosine values can be drawn from a look-up table. The clock is used to synchronise the $\delta$ functions and the cosine block. The cosine look-up table functions on the inverted clock to ensure the required value is present at the positive edge of the clock for use by the $\delta$ equation blocks.

Given the similarities between the lines of the piecewise function, Equation 1 in the FPGA approach it is possible to significantly improve the computation time by creating modules. This work would, therefore, look at initially creating modules specifically aimed at accelerating the $e^x$, $x^y$, $\log x$, and $\Gamma(x)$ functions [1].

5 Conclusions

The computation of a data-driven MCMC algorithm has been shown in this paper. Additionally, the optimisation for a software implementation of an MCMC algorithm for feature detection of ECG signals has been given. It has been shown that there are excellent opportunities to parallelise the MCMC algorithm, despite the significant dependency between the model parameters. As a work-in-progress toward a stand alone personal monitor, the future hardware implementation of
the feature detection algorithm is also discussed and some implementation optimisations and strategies have been shown. Therefore, the feasibility of using this approach in reconfigurable hardware has been confirmed.

References


