Feedback Guided Dynamic Integral Partition;  
A New Convergence Case

Sabin Tabirca  
Tatiana Tabirca  
University College Cork, Department of Computer Science, BCRI  
College Road, Cork, Ireland

Lawrence Tianruo Yang  
St. Francis Xavier University, Department of Computer Science  
P.O.Box 5000, Antigonish, B2G 2W5, Canada

Abstract: In this article we introduce a new case for an integral partition method called Feedback Guided Dynamic Integral Partition (FGDIP). The method generates iteratively a sequence of integral bounds by re-balancing the previous integral partition to achieve a better one. The convergence of the bounds is proven when the initial function does not vary too much. Experimental results show that the proposed method FGDIP achieves better performance than the classical Newton’s method.

Key Words: Integral Partition, Convergence, Balance.

1 Introduction

Consider a function $w : [a, b] \rightarrow (0, \infty)$ which is continuous. The problem to investigate is to find an approximation to the unique partition $a = x_0 < x_1 < \ldots < x_p = b$ which satisfies

$$\int_{x_{j-1}}^{x_j} w(x) dx = \frac{1}{p} \int_{a}^{b} w(x) dx, \quad j = 1, 2, \ldots, p. \quad (1)$$

This partition splits the integral $\int_{a}^{b} f(x) dx$ in $p$ equal parts or sub-integrals. It is clear that the above equation can be simply rewritten as

$$\int_{a}^{x_j} w(x) dx = \frac{j}{p} \int_{a}^{b} w(x) dx, \quad j = 1, 2, \ldots, p. \quad (2)$$

If $W : [a, b] \rightarrow [0, \infty)$ is defined by $W(x) = \int_{a}^{x} w(x) dx$ then Equation (2) can be rewritten as:

$$W(x_j) = \frac{j}{p} W(b), \quad j = 1, 2, \ldots, p. \quad (3)$$

This equation reduces the integral partition problem to the problem of solving $p$ non-linear equations $W(x_j) = \frac{j}{p} W(b)$ where the function $W(x)$ is differentiable and bijective. Several classical techniques can be used to solve these equations including the bisection or Newton’s methods (see [4] or [8] for more details).

Feedback Guided Dynamic Integral Partition (FGDIP) is a recent method that is different to these classical methods ([7]). The method finds the solution iteratively and it approximates the optimal bounds $\{x_j, j = 0, 1, \ldots, p\}$ over a sequence of steps $t = 0, 1, \ldots, \text{niter}$. For each $t$, the previous step information represented by the approximated bounds and their integral values is used to guide the new iteration bounds to a better integral partition. From this point of view the method is similar to Feedback Guided Dynamic Loop Scheduling (FGDLS) which is a scheduling technique from parallel computation [1]. Tabirca et.al [7] introduced this method and proved that it converges when an auxiliary function $\Phi_j(u, v)$ is a contraction on $u$.

The FGDIP method is important and it can be used to solve several theoretical and practical problems. The FGDIP output is what numerical analysts call ”equidistribution” of $w$. The equidistribution appears in various algorithms for solving ordinary differential equa-
tions [2]. Moreover, the integral partition over N dimensional regions (instead of intervals) is becoming more and more popular being important in solving partial differential equations [3]. The FGDIP method also has a practical application to parallel loop scheduling as illustrated in [6]. If the iteration workloads are known and given by the continuous function \( w : [0, n_j] \to (0, \infty) \) then we can consider the total workload of the parallel loop as being the integral \( \int_0^n w(x)dx \). In this case the FGDIP method can provide a block scheduling \( \{l_j, h_j, j = 1, \ldots, p \} \) that achieves a perfect load balance very close to \( \int_{l_j}^{h_j} w(x)dx \approx \frac{1}{p} \int_0^n w(x)dx \).

2 The FGDIP Method

The FGDIP method works with the bounds \( a = x_0^t < x_1^t < \ldots < x_p^t = b \) for each iteration \( t \geq 0 \). The initial bounds \( a = x_0^0 < x_1^0 < \ldots < x_p^0 = b \) for \( t = 0 \) can be arbitrarily or randomly chosen. The method calculates the new bounds \( \{x_j^t, j = 0, 1, \ldots, p \} \) from the bounds \( \{x_j^t, j = 0, 1, \ldots, p \} \) by balancing the integral partition of the step \( t \): the FGDIP method uses the feedback information represented by the bounds and integral partition at the step \( t \) to guide the calculation of the new bounds at the step \( t + 1 \).

In order to formalise this process we define a function, called Piecewise Constant Approximation (PCA), \( \hat{w}^t : [a, b] \to (0, \infty) \) by

\[
\hat{w}^t(x) = \frac{1}{x_j^t - x_{j-1}^t} \int_{x_{j-1}^t}^{x_j^t} w(x)dx, \quad x \in [x_{j-1}^t, x_j^t]
\]

(4)

and \( \hat{w}^t(b) = w(b) \). Obviously, this function takes the constant value \( \hat{w}_j^t \) on each interval \( [x_{j-1}^t, x_j^t] \) (see Figure 1), which makes possible to equipartition it. Therefore, we cannot find the equidistribution of the initial function \( f \) but we can solve the equidistribution values for a series of simple functions \( \hat{w}^t \).

The bounds \( \{x_j^{t+1}, j = 0, 1, \ldots, p \} \) are calculated as follows:

\[
\int_a^{x_j^{t+1}} \hat{w}^t(x)dx = \frac{j}{p} \int_a^b \hat{w}^t(x)dx, \quad j = 0, \ldots, p.
\]

(5)

This equation can be further simplified to

\[
\int_a^{x_j^{t+1}} \hat{w}^t(x)dx = j \bar{W}, \quad j = 0, 1, \ldots, p; \quad \forall t > 0
\]

(6)

where

\[
\frac{1}{p} \int_a^b \hat{w}^t(x)dx = \frac{1}{p} \int_a^b w(x)dx := \bar{W}
\]

is an invariant of the FGDIP algorithm [7].

The bounds \( \{x_j^t, j = 0, 1, \ldots, p \} \) give the following integral partition \( T_j^t = \int_{x_j^t}^{x_{j+1}^t} w(x)dx, \quad j = 1, 2, \ldots, p \) which we aim to balance it further.

Let \( pos \) be the index such that \( x_p^t \leq x_j^{t+1} < x_{pos}^t+1 \), which is equivalent to

\[
\int_a^{x_p^t} \hat{w}^t(x)dx \leq j \cdot \bar{W} < \int_a^{x_{pos}^t+1} \hat{w}^t(x)dx.
\]

(7)

Based on \( \int_a^{x_j^t} \hat{w}^t(x)dx = \int_a^b w(x)dx = \sum_{i=1}^j T_i^t \), Equation (7) becomes:

\[
\sum_{i=1}^{pos} T_i^t \leq j \bar{W} < \sum_{i=1}^{pos+1} T_i^t.
\]

(8)

This means that \( pos \) specifies the position of \( j \bar{W} \) in the monotonically increasing sequence \( S_j = \sum_{i=1}^j T_i^t, \quad j = 1, 2, \ldots, p \) and it can be found using a binary search.

[7] proposes a complete equation for \( x_j^{t+1} \) which is given below

\[
x_j^{t+1} = x_p^t + \left( j \bar{W} - S_{pos} \right) \frac{x_{pos}^t - x_p^t}{T_{pos+1}^t},
\]

(9)

and which it is the basis of the algorithm given in Figure 2.

In this algorithm the initial bounds \( \{x_j^0, j = 0, 1, \ldots, p \} \) are uniformly distributed over the interval \([a, b]\). For the step \( t \) the algorithm calculates the integrals \( T_j^t = \int_{x_j^t}^{x_{j+1}^t} w(x)dx, \quad j = 1, \ldots, p \) and checks whether they all
Inputs:
- \( p \) - the number of integrals to partition
- \( w : [a, b] \to (0, \infty) \) - the function to be partitioned
- \( ERR \) - the tolerance of the computation

Output:
- \( \{x^*_j, j = 0, 1, \ldots, p\} \) the partition bounds.

procedure FGDIP \((p, a, b, w, x^*)\)
begin
  Calculate \( \bar{W} \leftarrow \frac{1}{p} \int_a^b w(x) dx \);
  Initialise \( x^*_0 = a + j \cdot (b - a)/p; j = 0, 1, \ldots, p; \)
  for \( t = 0, 1, \ldots, n \) do
    Calculate \( T^*_j = \int_{x^*_t}^{x^*_{t+1}} w(x) dx, j = 1, \ldots, p; \)
    if \( |T^*_j - \bar{W}| < ERR, j = 1, \ldots, p \) then
      \( x^*_j \leftarrow x^*_j, j = 0, 1, \ldots, p; \)
      return;
    end if
    Compute \( S_j = \sum_{i=1}^{j} T^*_i, j = 1, 2, \ldots, p; \)
    \( x^*_{p+1} \leftarrow a; \)
    for \( j = 1, \ldots, p - 1, \)
      \( pos \leftarrow \text{search}(p, S_j, \bar{W}); \)
      \( x^*_{p+1} \leftarrow x^*_pos + (\bar{W} - S_{pos}) \frac{x^*_{pos+1} - x^*_pos}{x^*_pos + 1}; \)
    end for
  end for
end

Figure 2: The FGDIP Algorithm.

satisfy \( |T^*_j - \bar{W}| < ERR, j = 1, \ldots, p \). In this case we have
\[
\int_{x^*_0}^{x^*_1} w(x) dx \simeq \ldots \simeq \int_{x^*_{p-1}}^{x^*_p} w(x) dx \simeq \bar{W}
\]
so that the bounds \( \{x^*_j, j = 0, 1, \ldots, p\} \) achieve a good integral partition and we can finish the computation. If there is at least one integral so that \( |T^*_j - \bar{W}| \geq ERR \) then we find the new bounds \( \{x^*_{j+1}, j = 0, 1, \ldots, p\} \) by re-balancing the integrals. To achieve this, we find the position \( pos \) of \( \bar{W} \) in the ordered array \( \{S_j, j = 1, 2, \ldots, p\} \) by a binary search and then we can apply Equation (9) to find \( x^*_j \). This process involves the computation of \( p \) integrals and \( p \) binary searches.

3 Convergence when \( \sup w(u) < 2 \cdot \inf w(u) \)

In this section, we present a proof for the convergence of FGDIP when the initial function \( w : [a, b] \to (0, \infty) \) satisfies the equation

\[
\sup_{u \in [a, b]} w(u) < 2 \cdot \inf_{u \in [a, b]} w(u). \tag{10}
\]

Theorem 1 Assuming that \( w(x), x \in [a, b] \) is continuous then the partition sequence \( \{x^*_j\}_{j>0} \) satisfies the following inequalities:

\[
|x^*_j - x^*_j| \leq \sup_{v \in [a, b]} \left| 1 - \frac{w(u)}{w(v)} \right| |x^*_j - x^*_j|. \tag{11}
\]

Proof Let the index \( j \in [1, p - 1] \).
Let \( u(j) \) be the index \( pos \) from the FGDIP procedure which satisfies

\[
x^*_u(j) \leq x^*_u(j+1).
\]

Recall that the sequence \( \{x^*_j\}_{j>0} \) satisfies Equation (9). This equation can be rewritten as:

\[
(x^*_u(j) - x^*_u(j)) \frac{\int_{x^*_u(j)}^{x^*_u(j+1)} w(x) dx}{(x^*_u(j+1) - x^*_u(j))} = \frac{w(c_1)}{(x^*_j - x^*_u(j))}.
\]

Given that \( w(x), x \in [a, b] \) is continuous, by the mean value theorem for integrals, we find that there \( \exists c_1 \in (x^*_u(j), x^*_u(j+1)) \) such that

\[
\int_{x^*_u(j)}^{x^*_u(j+1)} w(x) dx = w(c_1)
\]

and there \( \exists c_2 \in (x^*_u(j), x^*_j) \) such that

\[
\int_{x^*_u(j)}^{x^*_j} w(x) dx = w(c_2).
\]

Thus we have

\[
(x^*_u(j) - x^*_u(j))w(c_1) = (x^*_j - x^*_u(j))w(c_2) \iff (x^*_u(j) - x^*_j)w(c_1) = (x^*_j - x^*_u(j))w(c_2) \iff (x^*_u(j) - x^*_j)w(c_1) = (x^*_j - x^*_u(j))(w(c_2) - w(c_1)) \iff
\]

3
\[ x_{j+1}^t - x_j^t = (x_j^t - x_{u(j)}^t) \left( \frac{w(c_2) - w(c_1)}{w(c_1)} \right). \]

Hence
\[
\left| x_{j+1}^t - x_j^t \right| = \left| x_j^t - x_{u(j)}^t \right| \left| \frac{w(c_2) - w(c_1)}{w(c_1)} \right| = 1 - \frac{w(c_2)}{w(c_1)} \cdot \left| x_{u(j)}^t - x_j^t \right| \leq 1 - \frac{w(v)}{w(v)} \left| x_{u(j)}^t - x_j^t \right|.
\]

Now, let us remark that, since \( u(j) \) is the index of the bound \( x_{u(j)}^t \) which is closest to \( x_j^t \) we have that
\[
\left| x_{u(j)}^t - x_j^t \right| \leq \left| x_j^t - x_j^t \right| \]
which proves the theorem.

\( \star \)

Since, \( w : [a, b] \rightarrow (0, \infty) \) is a continuous function, the function \( W : [a, b] \times [a, b] \rightarrow (0, \infty) \) defined by
\[
W(u, v) = 1 - \frac{w(u)}{w(v)}
\]
is continuous on the compact set \([a, b] \times [a, b] \). The supremum \( \sup_{(u,v) \in [a,b] \times [a,b]} W(u,v) \) exists because the function \( W \) achieves its upper bound on the compact set.

**Theorem 2** The following equivalence holds
\[
\sup_{u,v \in [a,b]} \left| 1 - \frac{w(u)}{w(v)} \right| < 1 \iff \sup_{u \in [a,b]} w(u) < 2 \inf_{u \in [a,b]} w(u). \tag{12}
\]

**Proof** the following chain of equivalencies gives a proof of the theorem.
\[
\begin{align*}
\sup_{u,v \in [a,b]} \left| 1 - \frac{w(u)}{w(v)} \right| < 1 & \iff \left| 1 - \frac{w(u)}{w(v)} \right| < 1, \ \forall u, v \in [a, b] \iff \\
-1 < 1 - \frac{w(u)}{w(v)} < 1, \ \forall u, v \in [a, b] & \iff \\
0 < \frac{w(u)}{w(v)} < 2, \ \forall u, v \in [a, b] & \iff \\
w(u) < 2w(v), \ \forall u, v \in [a, b] & \iff \\
\sup_{u \in [a,b]} w(u) < 2 \inf_{u \in [a,b]} w(u). \tag{12}
\end{align*}
\]

\( \star \)

Provided that the initial function satisfies Equation (10) then we find that
\[
q := \sup_{v,u \in [a,b]} \left| 1 - \frac{w(u)}{w(v)} \right| < 1
\]
so that the sequence satisfies
\[
\left| x_{j+1}^t - x_j^t \right| \leq q \left| x_j^t - x_j^t \right|, \ j = 1, 2, \ldots, p - 1.
\]

If we repeatedly apply this result, we obtain that
\[
\left| x_j^t - x_j^t \right| \leq q^{t-1} \left| x_j^t - x_j^t \right|, \ j = 1, 2, \ldots, p - 1.
\]

Since \( q < 1 \) it follows that the sequence \( \{x_j^t \} \) is convergent to \( x_j^t \).

\( \star \)

Equation (12) provides a simple condition for the convergence of FGDIP. It states that, the maximum of the initial function should be less than two times the minimum of it; this is equivalent to requiring that the initial function does not have large variations across the interval of study.

**4 Experimental Results**

Some numerical experiments have been conducted to observe how the FGDIP method balances the integral partition. For that we have used a Pentium III machine with the following technical specifications: Processor Intel Centrino Duo Processor T2600 (2.16GHz, 2MB L2 Cache, 667MHz FSB) with 2GB 667MHz DDR2 SDRAM memory. The algorithms have been implemented in C++ as language of choice.

The method is compared with the classical Newton method. For this method we generate the bounds
\[
y_j^{t+1} = y_j^t - \frac{W(y_j^t) - W(y_j^t)}{w(y_j^t)}, \ t \geq 0, \tag{13}
\]
which converge to \( x_j^t \) for all \( j = 0, 1, \ldots, p \), see [5]. To calculate the \( p - 1 \) bounds \( \{y_j^t, j = 1, 2, \ldots, p - 1\} \) we have to calculate \( p - 1 \) integrals for each step \( t \geq 0 \). Therefore, we can say that the methods calculate the same number of integrals and they both use an iterative process to approximate the values.

**Example 1.** The initial function \( w : [0,10] \rightarrow (0, \infty) \) is defined by \( w(x) = 2 + \frac{1}{2} \cdot \sin \frac{2\pi}{x} \). Remark that
\[ \sup_{u \in [0,10]} W^*(u) = \frac{5}{2} < 2 \cdot \inf_{u \in [0,10]} W^*(u) = 3, \]

therefore Equation (12) holds. The initial bounds are given by \[ (0, 2.5, 5, 7.5, 10). \]
The bounds for the first 5 iterations are presented below.

\[
\begin{align*}
\text{t=1:} & \quad 0 & 2.5 & 5 & 7.5 & 10 \\
\text{t=2:} & \quad 0 & 2.156 & 3.406 & 5.539 & 10 \\
\text{t=3:} & \quad 0 & 2.166 & 3.738 & 6.069 & 10 \\
\text{t=4:} & \quad 0 & 2.172 & 3.939 & 6.352 & 10 \\
\text{t=5:} & \quad 0 & 2.176 & 4.059 & 6.530 & 10 \\
\end{align*}
\]

and they generate the following integral partition

\[
\begin{align*}
\text{t=1:} & \quad 5.795 & 5.795 & 4.204 & 4.204 \\
\text{t=2:} & \quad 4.938 & 3.099 & 4.586 & 7.374 \\
\text{t=3:} & \quad 4.962 & 3.867 & 4.726 & 6.443 \\
\text{t=4:} & \quad 4.979 & 4.320 & 4.726 & 5.973 \\
\text{t=5:} & \quad 4.989 & 4.587 & 4.735 & 5.688 \\
\end{align*}
\]

and their variation is presented in Figure 3. Moreover, the equidistribution for \( t = 20 \) gives the following bounds \((0, 2.181, 4.247, 6.972, 10) \) which generate the integral partition of \((4.998, 4.997, 4.998, 5.007) \).

However, after 20 iterations, the Newton method produces the following bounds \((0, 2.180, 4.177, 6.733, 10) \) which generates the following distribution of the integrals \((4.997, 4.845, 4.788, 5.368) \). It means that the Newton’s method needs to calculate more iterations in order to generate the bounds with similar accuracy.

**Example 2.** Unfortunately, we cannot say that the FGDIP algorithm always converges. If we consider the function

\[ W^* : [0,10] \to (0, \infty), \quad W^*(x) = 200 + 100 \cdot \sin \frac{2\pi x}{10}, \]

then we find that

\[ \sup_{u \in [0,b]} W^*(u) = 3 > 2, \]

which means that Equation (10) is not true. If the initial bounds \( x^1 = (0, 2.5, 5, 7.5, 10) \) are considered, then we find the bound sequence \((x^t)_{t>0} \) is periodic. The periodicity starts from \( t = 106 \) and the period length is 104. The following bounds illustrate this property (see Figure 4).

**Figure 3:** Integral Partition for \( W^*(x) = 2 + \frac{1}{2} \sin \frac{2\pi x}{10} \).

**Figure 4:** Integral Partition for \( W^*(x) = 200 + 100 \cdot \sin \frac{2\pi x}{10} \).

The second experiment examines the overall performances of the FGDIP method against Newton’s method. The same function as in Example 1 is analysed but the integral is now partitioned into 100 sub-integrals. We observe the execution times in milliseconds for the error tolerance equal to \( 10^{-2}, 10^{-4}, 10^{-8}, 10^{-12} \) (see Table 1). It is clear that Newton’s method is less efficient since it has to calculate more iterations.

Similar experiments have been carried out for various types of polynomial and trigonometrical functions and they all have proven that the FGDIP method is
Table 1: Execution Times: FGDIP vs. Newton.

<table>
<thead>
<tr>
<th></th>
<th>$10^{-2}$</th>
<th>$10^{-1}$</th>
<th>$10^{-8}$</th>
<th>$10^{-12}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FGDIP</td>
<td>134.4</td>
<td>371.2</td>
<td>713.3</td>
<td>1176.2</td>
</tr>
<tr>
<td>Newton</td>
<td>131.7</td>
<td>421.3</td>
<td>951.5</td>
<td>1591.3</td>
</tr>
</tbody>
</table>

more efficient than the Newton method. Moreover, the FGDIP method has converged to the optimal bounds for some cases of trigonometrical functions which the Newton method did not converge.

5 Conclusions

This article has presented a new convergence case for the FGDIP method. We have proven that the bounds tend to the optimal bounds when the initial function does not vary too much. Some experiments have illustrated the convergence as well as the non-convergence. Finally, practical experiments have also shown that the method works better than the classical Newton’s method.

References:


Acknowledgement The second author acknowledges the financial support from Centre for Unified Computing at University College Cork, Ireland.