Embedded MATLAB：從 MATLAB 至 Embeddable C 實作

Houman Zarrinkoub 博士
訊號處理產品行銷經理，The MathWorks

（本文爲簡介 Embedded MATLAB 系列兩篇文章中的第一篇文章，將著重於探討直接從 MATLAB 命令列產生有效率的可嵌入式 C 碼的程序。在第二篇文章中，則將探討將 Embedded MATLAB 代碼整合至 Simulink 模型的程序，此程序爲模型基礎設計工作流程中的一環。）

理想與實作間的鴻溝，是許多參與嵌入式開發的工程師們都曾面臨的相似問題。演算法開發者常因其程式語言功能、大型函式庫及彈性，而以 MATLAB® 建立演算法概念。

隨著設計朝向嵌入式實作演進，亦須加入真實世界的限制式，因此通常需要使用者手動將 MATLAB 轉譯成 C 碼。手動轉譯包括將方便的矩陣 MATLAB 語法重新撰寫成 C 實作，但此舉最後會得到多份以不同程式語言撰寫的相同演算法，在此階段，使用者要面對驗證如此多份演算法的工作，工作量便與完成多項設計重複相同。在此工作流程中，驗證修訂版本的成本極爲驚人，結果導致設計太快實體化，或偏離原本的規格。

最近，MATLAB 的創作者 The MathWorks 導入可直接處理此問題的新工具，這些工具可自動將定義良好的 MATLAB 語言子集合 Embedded MATLAB™，轉換成嵌入式 C 碼。此項技術可減少手動將 MATLAB 轉譯成 C 語言的開發與驗證成本，而 Embedded MATLAB 語言子集合支援 270 項以上的 MATLAB 運算子與函數，以及 90 項 Fixed-Point Toolbox 函式。

您可於 Embedded MATLAB 子集合中作業，維護一份設計複本（「真實設計」），並直接於 MATLAB 進行細化，以加入嵌入式實作需求。由於 Embedded MATLAB 演算法仍爲 MATLAB 代碼，因此設計重複工作將變得更爲簡單，而且您仍可保有 MATLAB 中的互動式偵錯與視覺化功能。此方法亦提供演算法領域專家與嵌入式軟體工程師通用的語言，並可共同瞭解設計意圖。您可從 Embedded MATLAB 代碼直接產生 C 碼，進而省去產生與驗證手寫 C 碼的成本。

在本文中，我們將先審視在設計過程初期階段中實用的 MATLAB 功能，接著並檢驗將 MATLAB「概念」代碼手動轉譯成 C 碼進行實作時，缺乏效率的情形。

之後，我們會提出其他工作流程，其中各項演算式的嵌入式細化（Elaboration）工作，均以 MATLAB 完成，而非 C 語言。我們會以一個範例來突顯使演算式相容於 Embedded MATLAB 子集合的所需步驟，並展示如何使用可自動將 Embedded MATLAB 代碼轉譯成嵌入式 C 碼的新工具。

下表比較了 MATLAB 與 C 碼於設計初始階段的相關功能。

MATLAB	C 碼
非類型化語言（使用前不須宣告變數資料類型與大小）	嚴格的類型語言（必須於使用前宣告變數資料類型與大小）
內建矩陣資料類型定義	未內建矩陣資料類型定義
內建複雜變數資料類型	無複雜變數資料類型
不須撰寫程式，即可變更變數大小	明確透過動態記憶體配置變更變數大小
廣泛支援以工具箱組織的線性代數及應用程式限定的程式庫。	不支援線性代數及應用程式限定的程式庫
豐富的內建視覺效果	不標準的視覺效果

將典型的 MATLAB 演算法轉換成可嵌入的 C 碼時，涉及納入數項實作需求的工作：

- *資料類型管理* – 資料類型必須於實作前決定。例如，在處理影像時，像素值常以 8 位元不帶正負號的整數表示；而在處理語音時，則以 16 位元帶正負號的整數表示每個樣本。在 MATLAB 中使用預設的 64 位元雙精度變數，並無法有效率地使用記憶體。
- *靜態記憶體配置* – 於執行階段執行時，MATLAB 可完美地處理變數大小動態變更的工作。然而，在嵌入式應用程式中，我們通常會避免動態記憶體配置，並於使用前靜態定義指定大小與資料類型的記憶體。
- *減少運算複雜度及記憶體使用量* – 嵌入式軟體工程師常花費許多時間對應高層級演算法，以利用目標硬體有限的記憶體與運算資源有效地作業，而這將導致必須依照目標處理器的指令集與資料表現方式調整設計。

- *固定點支援* – 在嵌入式軟體或硬體中實作時，必須完全以固定點資料類型指定演算法。

工程師一般都會先將 MATLAB 演算法轉譯成 C 碼，以進行這些修改，因而創造出上述的設計鴻溝。並且，當軟體工程師在從 MATLAB 轉譯演算法時，往往可能將錯誤引入 C 碼或改變數值。若這些變更爲代碼最佳化的刻意結果，則演算法設計師可能必須於 MATLAB 演算法中重現這些變更，以保持等價（Equivalence）。然而，此程序不僅會增加無謂的工作，也可能也會發生錯誤，而以 Embedded MATLAB 爲基礎的嵌入式實作工作流程，便可處理這些問題。

Embedded MATLAB 代碼與許多 MATLAB 演算法不同，並非爲抽象的設計數學表示法，其中包含有效可嵌入式 C 實作所需的所有細節。任何遵守 Embedded MATLAB 子集合的 MATLAB 代碼，都可產生可嵌入式 C 碼；而確保符合 Embedded MATLAB 子集合的程序，即包括前述所討論的四項同樣的實作需求。採用 Embedded MATLAB 後，實作限制式將直接於 MATLAB 代碼中指定。

套用實作需求後，可讓您：

1) 以 MATLAB 維護您所設計的一份複本。
2) 直接於 MATLAB 合併內嵌式設計限制式，從概念形式將設計細化爲可準備實作的形式。
3) 使用 MATLAB 環境中可用的視覺化與分析功能反覆運算、測試及偵錯程式碼。
4) 驗證細化後設計的功能正確性。
5) 使用 Real-Time Workshop 以自動產生可嵌入式 C 碼。

如欲使 MATLAB 演算法相容於 Embedded MATLAB 子集合時，您可能必須：

- *設定變數的資料類型* – 您可於 MATLAB 函式本文中或編譯時期指派資料類型。在編譯時期進行指派會較爲方便，因爲單一 MATLAB 函式可產生具有不同資料類型、維度及複雜度的多重 C 函式變異。由於您在函式介面中指定了變數的資料類型及大小，因此將會自動推斷函式本文中所使用的變數資料類型及大小。

- *配合陣列大小變更，不動態資料配置* – 在 MATLAB 中，變數大小可於迴圈反覆運算之間變更，您可使用最大常數大小的緩衝區，並將常數大小緩衝區的定址配合嵌入式實作的陣列大小變更。

- *建立 Embedded MATLAB 相容函式* – 並非所有的工具箱函式都符合 Embedded MATLAB 子集合，這些函式的設計目的是爲了彈性搭配使用與達到數值準確性，而非用於嵌入式實作。工具箱函式可作爲您在使用 Embedded MATLAB 子集合開發功能等價之函式時的範本或參照，並符合有效進行嵌入式 C 碼實作所需的運算與記憶體限制式。之後，即可自動從 Embedded MATLAB 代碼產生所需的 C 碼。

- *從浮點轉換爲固定點* – 您可使用 Fixed Point Toolbox 中的資料紀錄與資料類型覆寫等工具，觀察 MATLAB 演算法中變數的動態範圍，這些工具有助於您將演算法轉換成 MATLAB 中最佳化的固定點表示法。由於此演算法在 MATLAB 內，因此您可直接將浮點與固定點的結果進行比較，以確保其差異在可接受的容差範圍內。

Embedded MATLAB 範例

以下爲此開發過程步驟的範例：

1) 確保其爲 Embedded MATLAB 相容且功能等價於參考代碼，並細化概念代碼，以加入嵌入式限制式。

2) 直接從 MATLAB 桌面產生 C 碼。

Embedded MATLAB 工作流程可套用至控制設計、通訊及訊號影像處理的各種工程問題。以此爲例，我們選擇已知爲適應性中數過濾轉變（Adaptive Median Filtering）的影像處理運算法，在許多 MATLAB 程式中，此典型實例經常出現，用於處理變動的變數大小。

MATLAB 函式如圖 1 所示，「adaptive_stats.m」依序擷取矩陣指定元素周圍較大的 region，以運算這些區域的統計資料（最小值、最大值及平均值），此函式會依據這些統計資料輸出矩陣篩選過的 Center Element。

圖 1 不相容的的統計函式

```
function Out = adaptive_stats(In)
% In : square matrix with odd number of rows/columns
% Out: Center element treated based on statistics
smax=size(In,1);
index=ceil(smax/2);
center=In(index,index);
% Initialize output
Out=center;
for s = 3:2:smax
    % Extract regions of successive sizes
    window_ind = -floor(s/2):floor(s/2);
    region = In(index+window_ind,ndex+window_ind);
    % Compute statistics
    rmin = min(region(:));
    rmax = max(region(:));
    rmed = median(region(:));
    % Conditionally replace center with median
    if rmed > rmin && rmed < rmax
        if center <= rmin || center >= rmax
            Out = rmed;
        end
        break;
    end
end
```

步驟 1：驗證是否相容 Embedded MATLAB 子集合

爲確定此函式是否與 Embedded MATLAB 子集合相容，我們建議您使用 EMLMEX 命令。在編譯 Embedded MATLAB 代碼時，EMLMEX 會檢查所有潛在的 MATLAB 語法違規事項、自動產生說明問題，並連結有問題代碼行列的編譯報告。

套用至此函數的命令語法爲：

```
>> emlmex adaptive_stats.m -eg {In}
```

此範例選項（接著–eg 分隔符號）可提供執行編譯時期指派的直覺式方法。依據前述設定變數資料類型的相容性工作，此方法可藉由指定函式介面的範例，設定 Embedded MATLAB 函式內的變數資料類型。EMLMEX 會使用 MATLAB 工作區中列於儲存格陣列並接著–eg 選項的變數資料類型與大小，並於編譯時期指派給函式輸入，然後推斷函式中所有變數資料類型與大小。

編譯報告會顯示，由於作爲變數 *s* 的五個變數（window_ind、region、rmin、rmax、rmed）變更大小採用迴圈內不同的數值，因此原始的函式並不相容，且此函式不相容的部分已於圖 1 中明顯標示。而這項問題違反了前述第二項配合陣列大小變更不具動態資料配置的相容性工作。若手動將此函式轉譯成 C，將會造成此五個變數非預期的動態記憶體配置。

解決方法爲引入以常數最大大小矩陣之 regions of interest 運作的新函式。圖 2 爲經過修改的函式「adaptive_stats_roi.m」，除了不含大小變更違規事項並可呼叫新函式（實作新的 region-of-interest 函式）外，其他完全相同。新函式呼叫標示於圖 2 中。

圖 2 相容函式

```
function Out = adaptive_stats_roi(In)
% In : square matrix with odd number of rows/columns
% Out: Center element kept treated based on statistics
smax=size(In,1);
index=ceil(smax/2);
center=In(index,index);
% Initialize output
Out=center;
for s = 3:2:smax
    % Compute Region-of-interest statistics
    [rmin,rmax,rmed]= roi_stats(In,smax,s);
    % Conditionally remove noise
    if rmed > rmin && rmed < rmax
        if center <= rmin || center >= rmax
            Out = rmed;
```

```
        end
        break;
    end
end
```

請注意，在依以下規則撰寫「roi_stats.m」新函式時，我們亦將某些以實作爲導向的最佳化納入 MATLAB 代碼：

- 移除多個函式呼叫的額外負擔（Overhead）。
- 移除每項統計多餘的 for 迴圈，以降低複雜度。
- 索引相同排序陣列的不同元素，以取得三項統計資料。

許多支援的 MATLAB 建構函式、運算子及函式，均用作爲 Embedded MATLAB 子集合的一部分。請注意，「mysort」爲 MATLAB 函式，用以排序陣列的次要部分。

```
Function [rmin,rmax,rmed] = roi_stats(in,smax,s)
first = ceil(smax/2)-floor(s/2);
last  = ceil(smax/2)+floor(s/2);
% Initialize large array for sorting
v = ones(smax*smax,1);
count = 1;
for i = first:last
    for j = first:last
        v(count) = in(i,j);
        count = count+1;
    end
end
% Sort the sub-matrix only
n = count-1;
v = mysort(v,n);
% Compute statistics on sub-matrix
rmed = v(ceil(n/2));
rmin = v(1);
rmax = v(n);
```

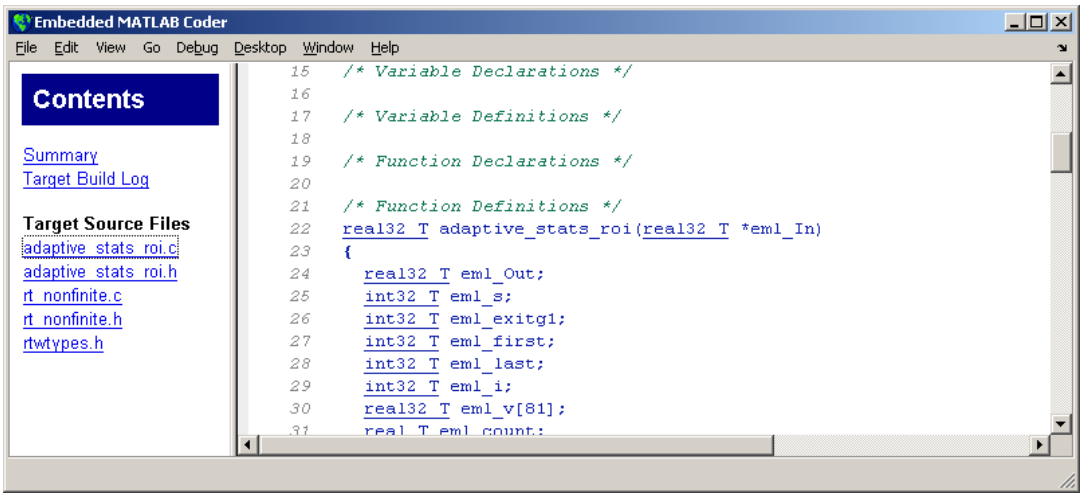
透過 EMLMEX 命令的執行，在觀察到無錯誤訊息後，我們即可確定更新過的函式「adaptive_stats_roi.m」現在相容於 Embedded MATLAB，並可準備自動產生 C 碼。請注意，您亦可使用 MATLAB M-Lint 檢查程式識別您演算法的相容性問題。

步驟 2：從 Embedded MATLAB 相容函式自動產生 C 碼

EMLC 爲命令列工具，可自動將您相容的 MATLAB 代碼自動轉譯爲 C 原始程式碼；此外，您可使用以下命令產生相容函式「adaptive_stats_roi.m」的代碼。

```
>> emlc adaptive_stats_roi.m -eg {In} -c -report
```

–report 選項可讓您建立包含 C 原始程式檔與標頭檔（自動由此 MATLAB 函式產生）之超連結的 HTML 報告。



您可輕鬆整合 Embedded MATLAB 工作流程中現有的 C 碼。您可能已開發 C 函式的程式庫，並且想要在您的程式中重新使用這些函式，而 Embedded MATLAB 提供如「eml.ceval」等的 C 介面函式，可讓您從 Embedded MATLAB 函式內部呼叫外部 C 函式。例如，在我們的 MALAB 函式「oi_stats.m」中，我們可以不要呼叫 MATLAB 函式「mysort」，轉而呼叫宣告爲以下項目的現有 C 函式「c_sort」：

```
Bool c_sort(float *U, float *Y, int n, int m);
```

此項取代法涉及使用以下「eml.ceval」函式，將輸入與輸出引數依傳值或參照傳遞給 C 函式。

```
% Sort the sub-matrix only
sorted=eml.ceval('c_sort',eml.rref(v(1)),eml.wref(v(1)),count,int(numel(v)));
```

摘要

使用 Embedded MATLAB 語言子集合產生的代碼，目的即在於縮短理想與實作之間的鴻溝，此方法導入了全新的工作流程，其中演算法設計與細化工作是在單一語言與環境下進行。透過 MATLAB 的自動 C 碼產生功能，您便可維護您所設計的單一來源，使追蹤重複作業變得更爲簡單；此外，演算法設計師與軟體工程師更可協力工作，以迅速產生最佳實作。

（如需瞭解更詳盡的資訊，請造訪 MathWorks 網站上的 Embedded MATLAB 網頁：www.mathworks.com/products/featured/embeddedmatlab/ 。
網頁中包含：示範教學、客戶推薦、技術細節及 MathWorks 相關產品連結。如需透過詳細的示範教學學習如何使用 Embedded MATLAB，請觀看其中一段 Webinars featuring Embedded MATLAB，例如 www.mathworks.com/company/events/archived_webinars.html 上的「Algorithm Design and Code Generation with Embedded MATLAB」。
並請造訪 MATLAB CENTRAL www.mathworks.com/matlabcentral，以關鍵字「Embedded MATLAB」進行搜尋，查看使用者所提供的 Embedded MATLAB 範例，包括用於 Webinar 示範教學的 MATLAB 檔案。）

簡歷

Houman Zarrinkoub 博士於 2001 年加入 The MathWorks，擔任訊號處理團隊的資深主管，該團隊負責動態影片及影像處理模塊組（Video and Image Processing Blockset）。Houman Zarrinkoub 博士目前爲訊號處理產品行銷經理，負責訊號處理工具箱。在加入 The MathWorks 前，Houman Zarrinkoub 博士曾於 Nortel Networks 公司擔任無線語音處理軟體工程師長達六年的時間，並於 1994 年取得 McGill University 的電機工程學士（BSEE），之後再於加拿大的 Institut National de la Recherche Scientifique, Université du Quebec 取得電機工程碩士（MSEE）與博士學位。