第三章 跨學科優化

3.1 單目標最佳化

在單目標最佳化問題中，優化任務可以由單一純量值目標函數來表達

$$f(x_1, x_2, ..., x_k) \rightarrow \min \tag{3.1}$$

這通常是設計變量的隱函數。設計變量可以定義為連續變量，有下限和上限，或作為離散變量，它們假設數個離散值。在無約束的最佳化問題中，只有設計變量的邊界或值限制了優化空間。優化器在這些限制間搜索最小值的目標函數 $f(x)$。對於最大化問題，用戶定義的目標函數的符號在optiSlang中被反轉以得到最小化任務。

> **附註**
> 隱函數是一種在數學中常見的函數形式，與我們熟知的顯函數（像是 $y = f(x)$）不同。隱函數指的是那些不直接解出因變量（比如 y）作為自變量（比如 x）的函數，而是以包含兩者的等式來表達的函數。
>
> 例如，考慮方程式 $x^2 + y^2 = 1$。這是一個圓的方程，並不直接給出 y 作為 x 的函數，但 y 實際上依賴於 x。這就是一個隱函數的例子。這個方程式並不直接告訴我們 y 的值，但確實隱含著 y 和 x 之間的關係。

在工程問題中，通常需要滿足額外的限制，以便實現最優設計。這可以透過等式和不等式約束來實現

$$\begin{aligned} g_i(x_1, x_2, ..., x_k) &= 0, i = 1...m_e \\ h_j(x_1, x_2, ..., x_k) &\geq 0, j = 1...m_u \end{aligned} \tag{3.2}$$

這樣的限制可以被制定。約束函數可以只依賴於輸入變量，但也依賴於所有可用模型響應和任何數學組合。

在工業應用中處理等式約束非常困難。因此，不允許在optiSlang中明確地定義等式約束。然而，具有給定公差的等式約束可以容易地作為不等式約束來表達，這樣它們就可以被所有優化方法在optiSlang中處理。為了使優化器能夠準確地收斂到最優解，並且足夠準確地完成目標和約束條件，這是非常有用的，以使目標和約束函數處於相似的範圍。例如，這可以通過使用設計探索的結果來實現。

Figure 3.1: Recommended Flowchart for Single-Objective Optimization

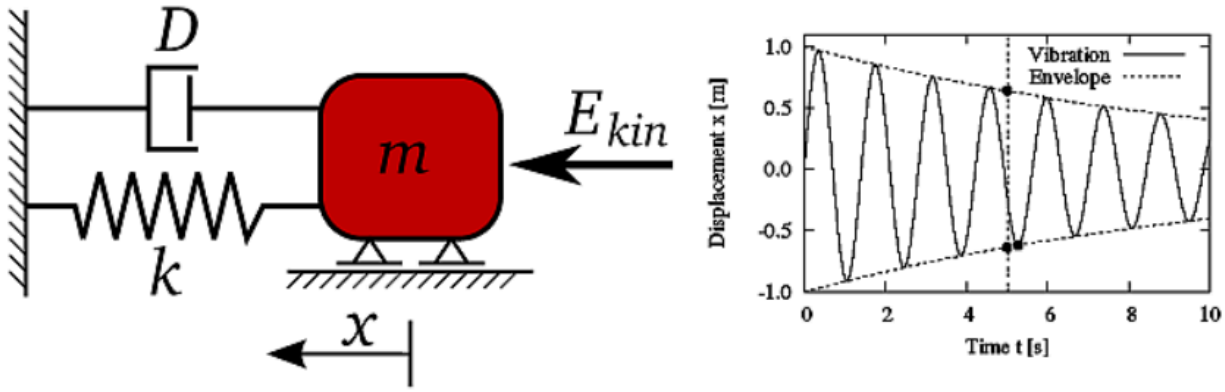
在圖3.1中：推薦的單目標優化流程圖如下：在定義設計變數和目標及約束函數後，透過敏感度分析來探索設計空間。獲得的變數敏感度分析可能有助於減少設計變數的數量。在敏感度分析中找到的最佳設計將作為後續優化程序的起點，最終確定一個最佳設計。

附帶實例：阻尼振子的優化

本章所述的優化方法原則將通過一個簡單的例子來演示。為此，我們將針對圖3.2所示的單自由度阻尼振子的最大振幅進行優化：系統屬性和振動行為-或（見第28頁）應最小化。

> **附註**
> 以汽車懸掛系統為例，當車輛行駛過崎嶇不平的道路時，它依靠彈簧系統來吸收嚴重的衝擊和振動。如果彈簧過於柔軟，它們可以有效地消減震動，但會導致車輛長時間上下波動，類似於在柔軟的床墊上跳躍，儘管舒適，但震盪持續時間較長。相對地，如果彈簧過於堅硬，它們雖然能快速地使車輛穩定，但這種快速的穩定性會伴隨著較大的震動，這就像在堅硬的板凳上跳動，震動感會很顯著且不舒適。
>
> 為了平衡吸震效果與迅速恢復穩定的需求，我們會選擇介於柔軟與堅硬之間的彈簧硬度，並搭配阻尼裝置。阻尼裝置能縮短震動持續的時間，幫助車輛在吸收衝擊後迅速返回穩定狀態。這種設計在汽車懸掛系統中極為關鍵，它不僅提高了駕乘舒適度，還有助於保護車輛不受惡劣道路條件的損害。

Figure 3.2: Damped Oscillator: System Properties and Oscillation Behavior



運動方程可以根據質量 m，彈簧剛度 k 和阻尼比 D 如下列公式所示：

$$\ddot{x} + 2D\omega_0\dot{x} + \omega_0^2 x = 0 \tag{3.3}$$

其中 $\omega_0 = \sqrt{k/m}$，是未阻尼的固有頻率。假設初始動能 E_{kin} 和初始位置 $x_0 = 0$，時間依賴的位移函數可以推導如下：

$$x(t) = e^{-D\omega_0 t}\sqrt{\frac{2E_{\text{kin}}}{m}}\frac{1}{w}\sin(\omega t) \tag{3.4}$$

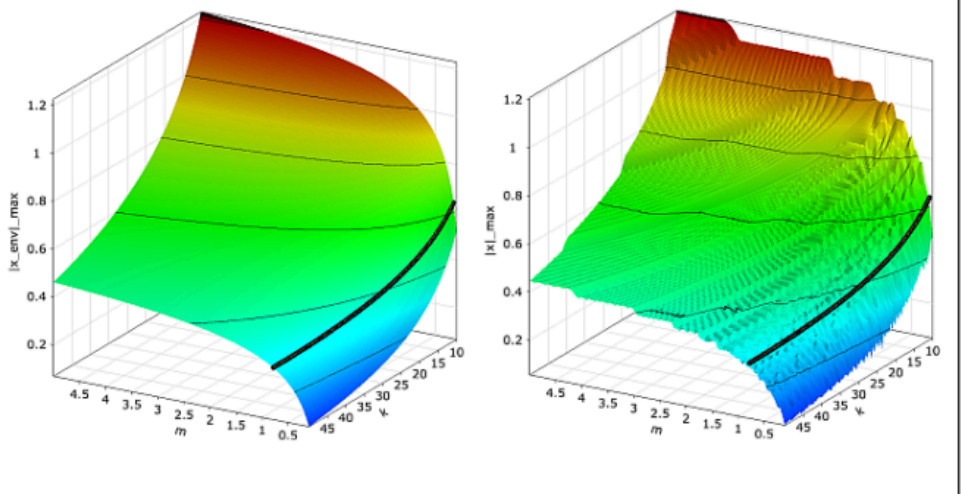
其中 $\omega_1 = \omega_0\sqrt{1-D^2}$ 是阻尼固有頻率。優化的目標是在假設 m 和 k 為設計參數，D 和 E_{kin} 為常數的情況下，最小化5秒後的最大振幅：

$$\begin{aligned} m &\in [0.1, 5.0kg], \quad D = 0.02 \\ k &\in [10, 50N/m], \quad E_{kin} = 10\text{Nm} \end{aligned} \tag{3.5}$$

作為優化約束，阻尼固有頻率應受限於 $\omega < 8$ 1/s。

如果從圖3.2所示的包絡線估計出5秒後的最大振幅：阻尼振盪器：系統特性與振盪行為（第28頁），那麼目標函數是關於設計參數的一個平滑且定義良好的函數，如圖3.3：阻尼振盪器的目標函數（第29頁）所示。

Figure 3.3: Objective Function of the Damped Oscillator



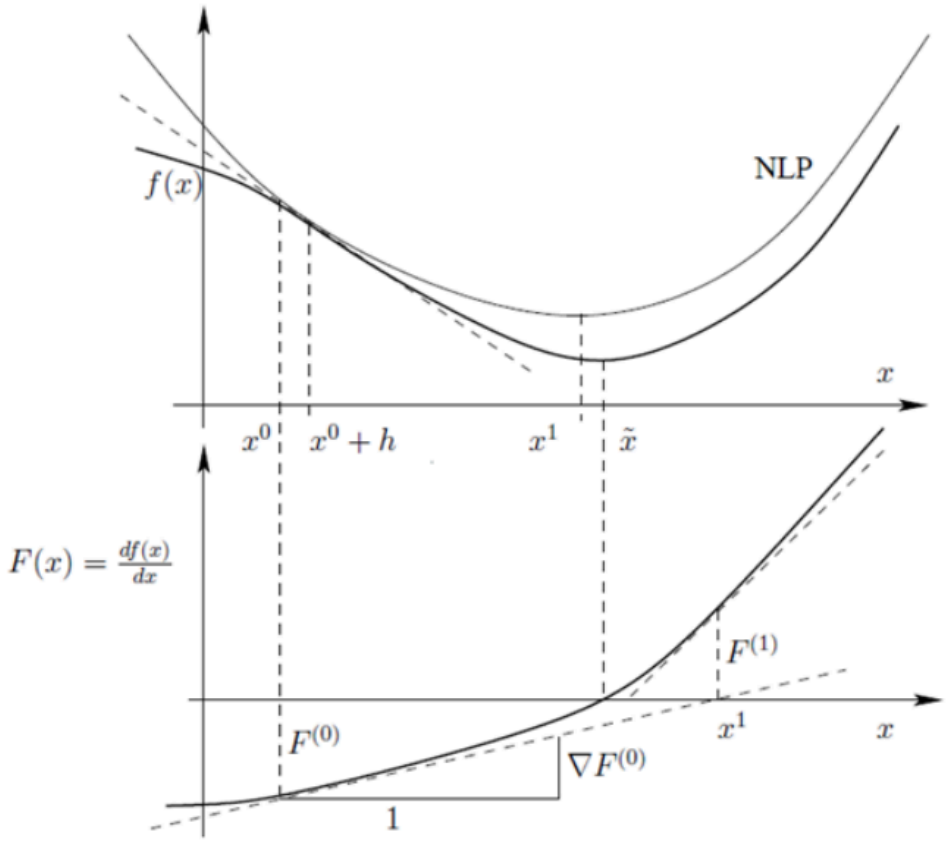
Objective function of the damped oscillator obtained by using an es- timate of the maximum amplitude from the envelope curve (left) and by using a coarse time discretization of the displacement curve (right).

約束條件在圖中由函數 w = 8 1/s 表示。如果使用較粗的時間間隔對方程式3.4（第28頁）進行時間離散化，得到的目標函數可能包含局部振盪，這可能會被解釋為求解器噪聲。為了展示在小求解器噪聲存在時不同優化器的行為，選擇了0.1秒的時間步進，這導致圖3.3中另外顯示的目標函數略帶噪聲：阻尼振盪器的目標函數（第29頁）。

3.1.1. 基於梯度的方法

基於梯度的優化方法使用目標函數的局部導數來尋找下一個局部最優。如圖3.4所示，通過尋找第一導數等於零的點，可以確定凸函數的最小值：基於梯度的方法使用二次近似的目標函數進行迭代搜索（第30頁）。

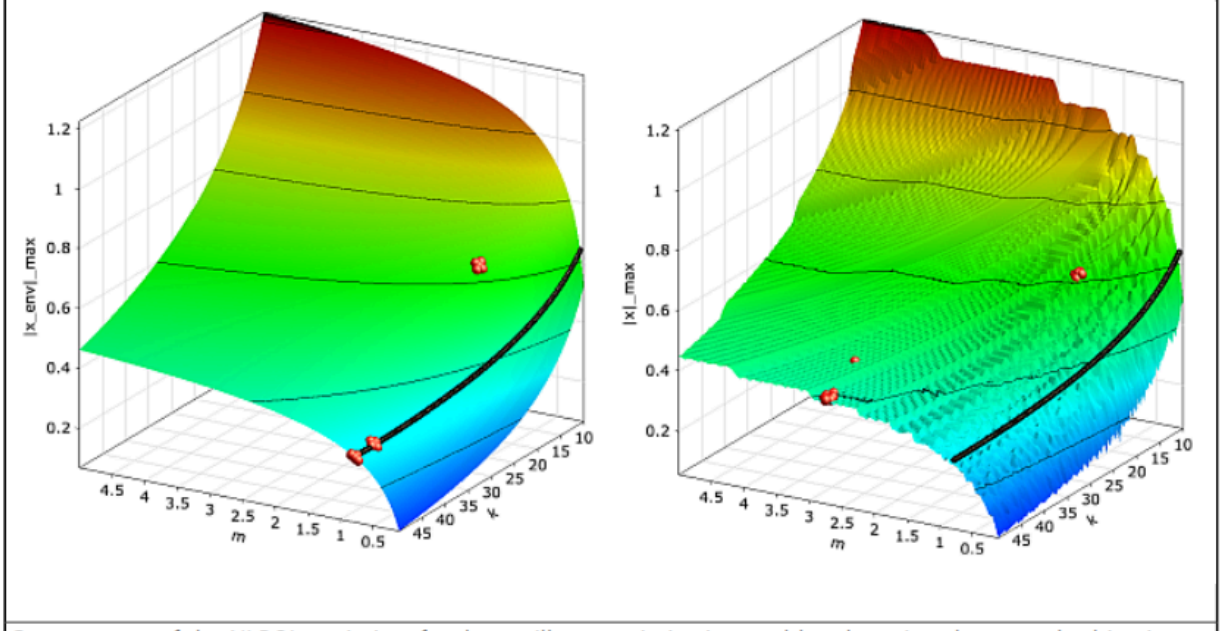
Figure 3.4: Iterative Search of a Gradient-Based Method Using a Quadratic Approximation of the Objective Function



如果已知第二導數，則可以用二階泰勒級數近似目標函數。這個過程稱為非線性規劃（Nonlinear Programming, NLP）。使用一階和二階導數的優化方法被稱為牛頓方法（Kelley 1999 (第89頁)）。如果將CAE求解器視為黑盒，則必須通過數值估計計算導數。由於二階導數需要大量的求解器調用，完整的牛頓方法不適用於複雜的優化問題。因此，開發了擬牛頓方法，其中從先前迭代步驟的一階導數近似二階導數。基於梯度的方法主要在於二階導數的估計方式以及如何考慮額外的約束方程式不同。關於不同方法的好概述可以在（Kelley 1999 (第89頁)）找到。

在optiSLang中，傾向於使用NLPQL方法（Nonlinear Programming by Quadratic Lagrangian）（Schittkowski 1986 (第90頁)）。目標和約束函數的一階導數是通過中心差分或使用指定的差分區間的單邊數值導數估計的。NLPQL方法具有使用二次拉格朗日的高效約束處理特性。從給定的起點開始，該方法搜索下一個局部最優，並在估計的梯度低於指定的容忍度時收斂。由於它是一種局部優化方法，建議使用全局敏感性分析的最佳設計作為起點，以找到全局最優。NLPQL在低維度（最多20個設計變量）非常高效。對於高維問題，計算數值導數越來越昂貴，其他方法更為高效。在模型響應噪聲存在的情況下，差分區間起著至關重要的作用。如果取得太小，求解器噪聲會嚴重扭曲估計的梯度，NLPQL會朝錯誤的方向運行。如果求解器噪聲不支配函數趨勢，增加差分區間可能會導致優化程序的良好收斂。

Figure 3.5: Convergence of the NLPQL Optimizer



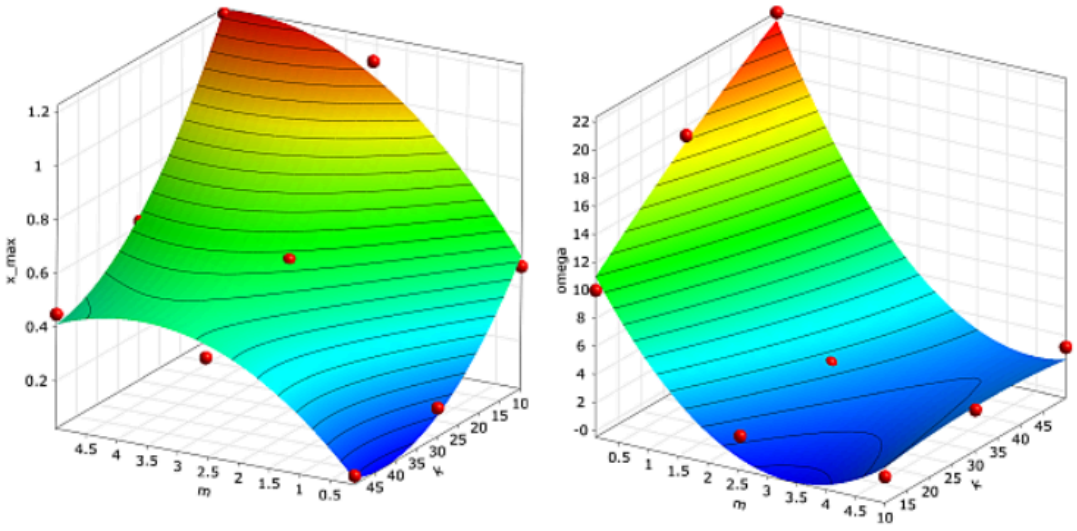
Convergence of the NLPQL optimizer for the oscillator optimization problem by using the smooth objective function based on the envelope estimate (left) and by using the noisy objective function from the time discretization (right).

在圖3.5：NLPQL優化器的收斂（第31頁），展示了NLPQL方法在阻尼振子優化方面的收斂情況：在第一次研究中，考慮了從包絡線估計得到的平滑目標函數。在這種情況下，優化器在三次迭代步驟中收斂到真正的最優解（m = 0.78公斤，k = 50.0牛頓/米）。如果用相對較小的差分間隔（設計空間的1%）來研究嘈雜的目標函數，優化器會走錯方向，無法收斂到真正的最優解。然而，通過增加差分間隔，可以為此例實現對真正最優解的收斂。

3.1.2 響應面方法

響應面方法通過數學替代函數來取代模型反應。在此基礎上，使用近似模型而非耗時的求解器調用來解決優化問題。經典流程採用實驗設計方案，該方案由求解器評估。基於這些設計，通常使用多項式函數進行近似。然而，多項式的度數往往無法代表求解器模型的非線性，而近似質量低下往往導致設計建議不可用。因此，不建議僅使用全局多項式響應面方法來尋找最佳設計。

Figure 3.6: Global Polynomial Approximation



Global polynomial approximation of the maximum amplitude (left, R^2_{adj} = 98%) and the damped eigen-frequency (right, R^2_{adj} = 96%) of the optimized oscillator using a quadratic basis and a full factorial design scheme.

為了展示全局多項式響應面近似結合經典實驗設計方案的弱點，使用二次多項式近似阻尼振子的最大振幅和阻尼固有頻率。支撐點是通過三級全因子設計生成的。儘管通過調整確定係數表示近似質量非常好，但在近似上找到的最優解（m = 3.31公斤，k = 50.0牛頓/米）與真正的

最優解相去甚遠。

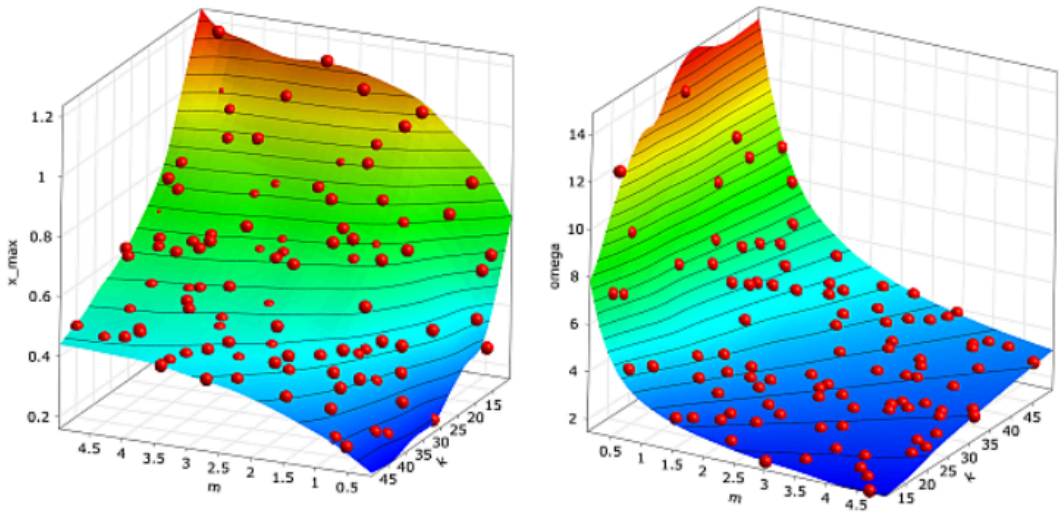
使用最佳預測的元模型進行優化

由於最佳預測的元模型在找到一個最佳變量子空間和每個被調查模型響應的最佳近似模型方面的強大能力，強烈建議使用MOP近似作為第一步優化，而不是全局多項式模型。如在預測係數（第19頁）中解釋的那樣，預測係數比確定係數提供了一個更可靠的近似質量估計。如果在目標或約束函數中使用的某個模型響應的質量低，優化過程可能找不到有用的優化設計。為了檢查在MOP上找到的最佳設計的質量，optiSLang提供了對最佳設計的驗證，其中求解器輸出和真實目標和約束值是通過單一求解器調用計算的。通常，基於MOP的優化找到的最佳設計比前一靈敏度分析的最佳設計更好。如果是這樣，這個設計可以用作進一步局部搜索的起始設計。

> **附註**
> Metamodel of Optimal Prognosis（MOP）不是指一個特定的數學模型，而是指一個經過調整和優化以後的模型，這個模型能夠在給定的數據範圍內預測或估計結果。這種模型可以基於不同的數學方法建立，比如多項式（polynomial）、克里金（kriging）方法，或是更複雜的如神經網絡等。
>
> MOP的目標是找到一個最佳的近似模型，這個模型可以在不完全知道真實物理過程的情況下，根據可用的數據來預測系統的行為。這種模型特別適用於複雜的工程問題，其中真實系統的直接模擬可能非常耗時或者計算上不可行。

在許多約束條件的存在下，如果近似質量不是完美的，基於MOP的最佳設計往往會違反一個或多個這些條件。這個問題可以通過調整相應約束條件的限制來解決，以推動在近似上運行的優化器回到可行區域。在每次調整約束條件後，應驗證最佳設計，以檢查是否可能滿足關鍵約束。
由於求解器噪聲被MOP近似平滑處理，這個過程比基於梯度的方法更穩定。**此外，可以通過在近似函數上使用全局優化器，例如進化算法（EA），來解決非凸優化問題**。然而，通常MOP使用均勻分布的設計來近似模型，這可能導致在最優點附近的局部近似質量不足。因此，使用全局近似模型的優化應被理解為一個低成本的預優化步驟。

Figure 3.7: Metamodel of Optimal Prognosis



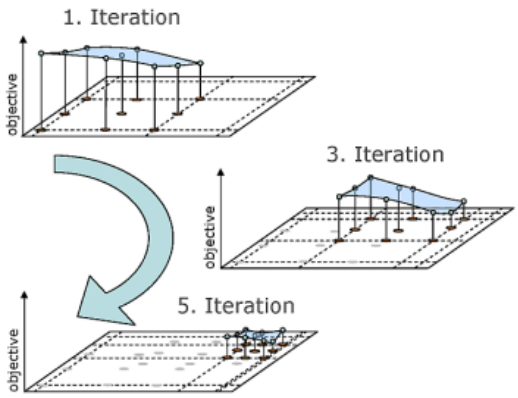
Approximation of the maximum amplitude (left, *CoP* = 99%) and of the damped eigen-frequency (right, *CoP* = 98%) by using the Metamodel of Optimal Prognosis with 100 Latin Hypercube samples

為了展示在MOP近似上進行優化的好處，通過用100個拉丁超立方樣本建立MOP來研究阻尼振子。MOP的近似函數顯示在圖3.7：最佳預測的元模型（第33頁）。該圖表明了在預測係數方面的優異近似質量。通過使用MOP近似函數，最優解被確定得非常接近真正的最優解（m = 0.77公斤，k = 50.0牛頓/米）。在獲得的最優點處，近似的阻尼固有頻率為ω = 8.00 1/s，但求解器驗證的真實固有頻率為ω = 8.04 1/s，這意味著約束條件略有違反。在這種情況下，降低最大允許固有頻率將迫使優化器留在可行區域。

適應性響應面方法

為了提高最優點附近的近似質量，適應性方法非常有效。optiSLang提供了一種基於多項式的局部適應性響應面方法（ARSM）。ARSM程序從單一起始設計開始，並建立一個以起始設計為中心點的初始實驗設計（DoE）方案。對於線性或二次多項式模型，相應的D-最優設計方案被優先作為DoE方案。基於對模型響應的近似，尋找DoE方案參數範圍內的最優設計。在下一次迭代步驟中，圍繞這個最優設計建立一個新的DoE方案。根據當前和上一次迭代步驟中最優設計的距離，DoE方案會被移動、縮小或擴展。這個過程原則上在圖3.8中展示。關於適應過程的更多細節可以在（Etman等人，1996年，第89頁）和（Stander和Graig，2002年，第90頁）中找到。

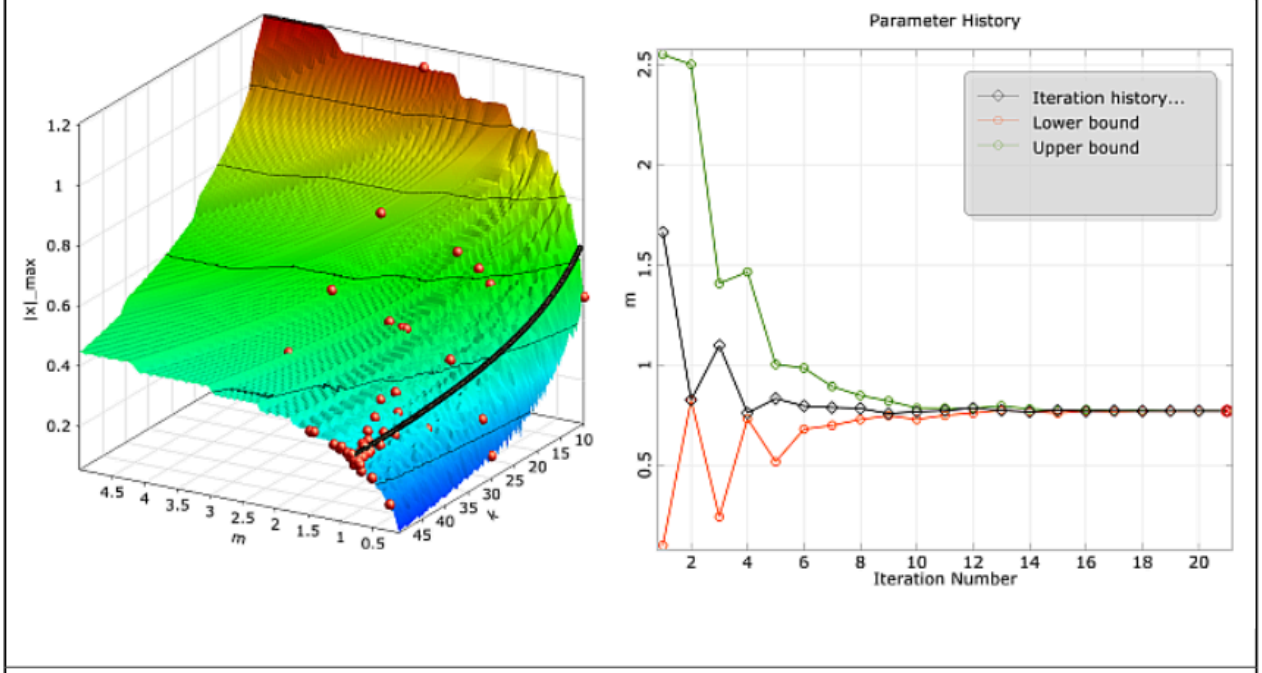
Figure 3.8: Adaptation of the Polynomial Approximation Scheme Inside the Adaptive Response Surface Method



當DoE縮小到最小尺寸，或者兩個迭代步驟之間最優設計位置及其目標值的變化低於指定容忍度時，ARSM算法收斂。由於DoE方案使用比多項式近似所需的設計多50%，求解器噪聲被平滑，少量失敗的設計對優化器不構成問題。由於其對高達20個變量的效率以及對求解器噪聲的魯棒性，ARSM是低維單目標優化問題的首選方法。建議使用前一靈敏度分析的最優設計作為ARSM的起始設計。對於強烈局部化搜索，初始DoE方案的起始範圍應該縮小。

在圖3.9：適應性響應面方法的收斂（第35頁），通過分析嘈雜的目標函數，展示了ARSM優化器對振子問題的收斂情況。

Figure 3.9: Convergence of the Adaptive Response Surface Method



Convergence of the Adaptive Response Surface Method for the damped oscillator by using the noisy objective function: designs used for the adaptation (left) and modification of the local DoE bounds during the iteration (right).

使用50%設計空間的起始範圍和線性多項式基礎，優化器在幾個迭代步驟內運行於真正最優解的區域。在20次迭代步驟後，算法在m = 0.77公斤，k = 49.3牛頓/米處收斂，滿足約束條件。這個例子闡明了ARSM優化器對於嘈雜模型響應展示了穩定的收斂行為，與基於梯度的方法相比。

3.1.3. 基於群體方法

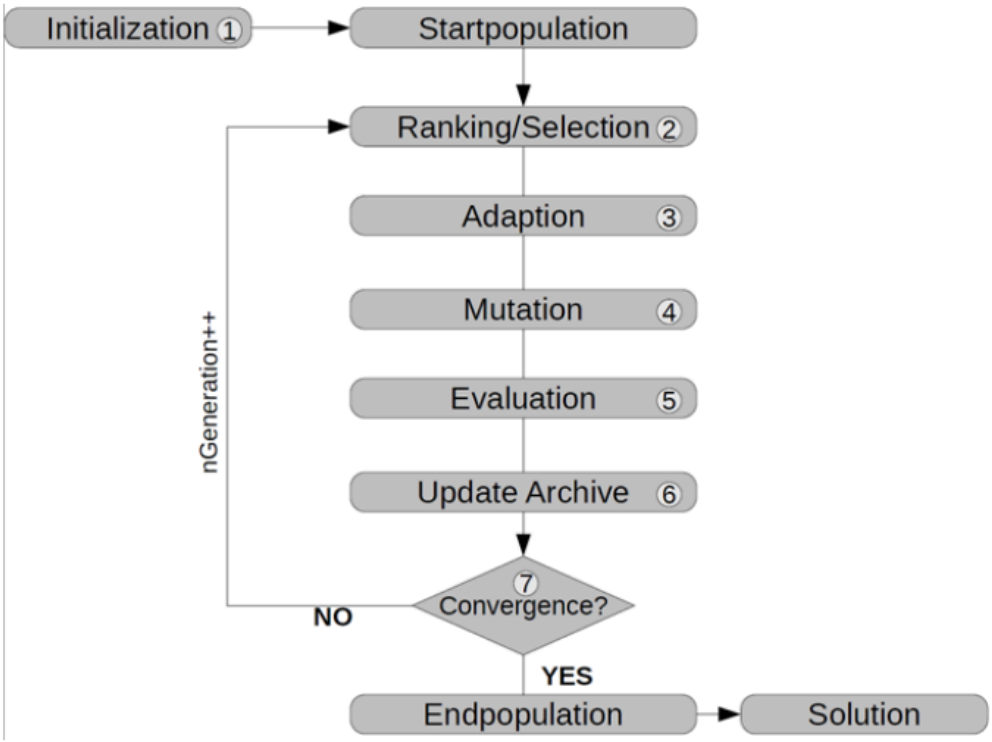
基於群體的優化方法模仿自然過程，如生物進化或群體智慧。基於“適者生存”原則，一群人工個體在可能解決方案的設計空間中搜索，以找到優化問題解決方案的更好近似。

所有類型的群體基算法都遵循特定流程。首先是隨機初始化（見圖3.10：群體基優化算法的通用流程圖（第36頁），步驟1）一個包含μ個個體的群體，這些個體代表給定優化問題的可能解。

初始化後，實際的迭代循環開始，每個循環代表一代。為了創建新一代，將選擇最適合的設計（步驟2）。步驟3中提到的適應可以通過不同方式完成，例如交叉或群體移動。之後可以應用突變（步驟4）。

所有個體都會被評估（步驟5），通過基於目標值和可能的約束違反來賦予它們一個適應度值。因此，高適應度分數意味著對問題的良好適應，並與小目標值相對應。評估每個設計後，存儲好的解決方案的存檔將被更新以進行下一次選擇步驟（步驟6）。世代計數器增加，迭代繼續，直到滿足停止標準（步驟7）。

Figure 3.10: General Flowchart of Population Based Optimization Algorithms



選擇合適的起始群體可以顯著提高群體基方法的效率。如果在初始靈敏度分析後進行優化，強烈建議使用靈敏度分析的最佳設計作為起始群體。為了保持優化的全球特性，選擇覆蓋設計空間一定範圍的設計是有用的。

群體基算法在廣泛的應用中都是推薦使用的。它們通常是解決數學上條件不佳的問題的最後手段，因為在設計變量數量大或失敗設計數量大的情況下，它們的表現可能仍然良好。然而，與其他優化方法相比，收斂行為可能較慢。在所有基於梯度的優化或響應面近似失敗的情況下，如果變量或約束數量大，如果設計變量是離散或二進制的，如果響應是離散的，或者如果用戶不了解優化問題，都建議使用群體基算法。

進化算法

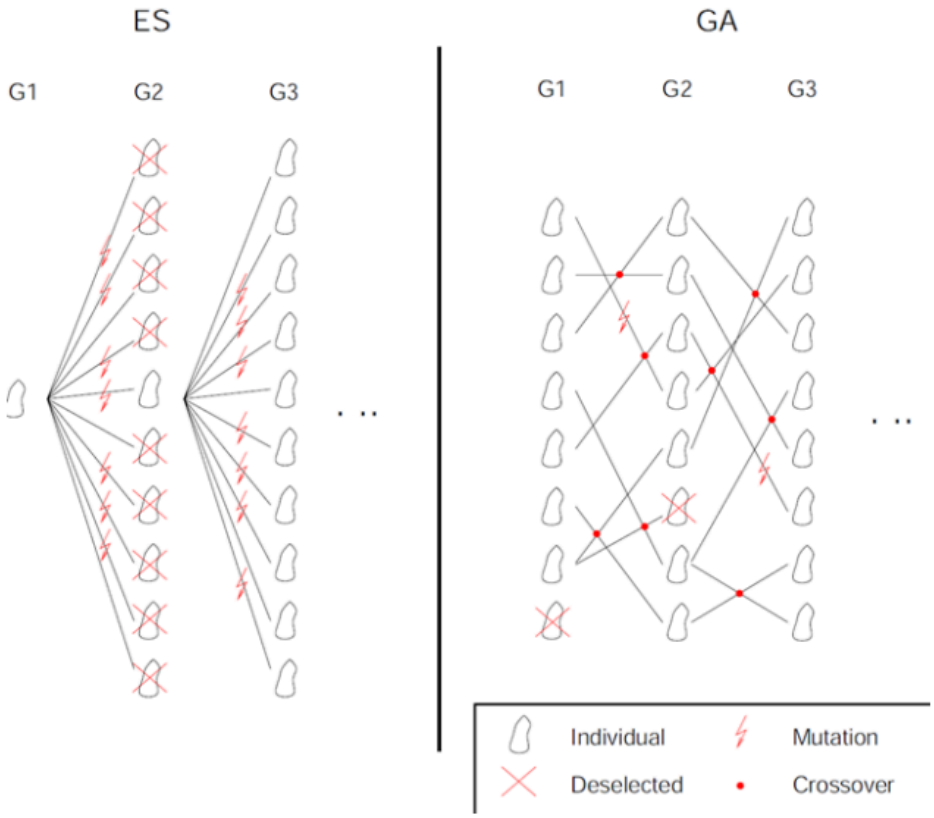
進化算法（EA）是模仿自然生物進化過程的隨機搜索方法。在過去幾十年中，基於三個主要類別實現了許多EA變體：遺傳算法（GA）（Holland 1975 (第89頁)）、進化策略（ES）（Rechenberg 1964 (第90頁)）和進化編程（EP）（Fogel等人，1966 (第89頁)）。這些算法最初是為了解決沒有梯度信息可用的優化問題而開發的，如二進制或離散搜索空間，儘管它們也可以應用於具有連續變量的問題。在optiSLang中有靈活實現的遺傳算法和進化策略可用，允許在純（強）GA和純（強）ES之間進行調整。

在隨機或手動初始化包含μ個個體的群體後，實際的迭代循環開始，每個循環代表一代g。根據其適應度或排名，選擇確定了λ個個體進行繁殖。提供不同的隨機選擇操作符，以調整選擇壓力。親代選擇設定隨機搜索過程的方向。通過對選定個體應用交叉和突變操作符來引入變異。這些新個體被稱為新生兒或後代。最後，對結果後代個體進行評估，並使用替換方案形成一個新的μ個體群體。

兩種主要變體之間的主要區別，遺傳算法（GA）和進化策略（ES），是向群體引入變異的方式。使用交叉操作符對基因進行重組代表遺傳算法中的主要變異，而對於進化策略，（適應性）突變將變異引入到群體中（見圖3.11：進化策略與遺傳算法對比（第38頁））。

交叉操作符是一種重組方法，其中兩個親代個體產生兩個後代，通過在染色體之間共享信息。其目的是獲得具有更好特性的個體（利用）並保持群體的多樣性（探索）。交叉被視為遺傳算法中的主要搜索操作符。

Figure 3.11: Evolution Strategies Versus Genetic Algorithms



突變向後代染色體的基因引入隨機變異。每個基因都會以指定的概率或突變率進行選擇。實數值突變基於每個基因的正態分佈函數，以基因的值作為其平均值。突變是進化策略（ES）的主要搜索操作符，但也可以在重組後應用（見圖3.10：群體基優化算法的通用流程圖（第36頁））。

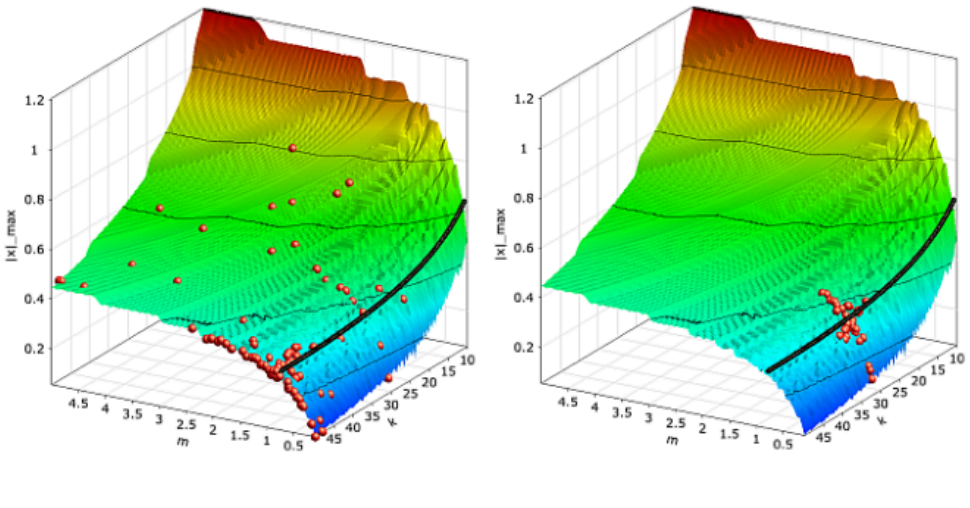
突變率和相對於變量範圍定義的標準差是突變的控制參數，可以保持不變或在算法運行過程中進行修改。optiSLang提供恆定和適應性突變程序。更多細節可以在（Bäck 1996 (第89頁)）和（Riedel等人，2005 (第90頁)）中找到。

最後的操作符，存檔更新方案，指定了如何從當前群體（μ）和生成的後代個體（λ）中形成下一代群體。從選擇池中選擇下一代最好的μ個個體。通過調整存檔大小（α），可以提供不同策略來構成選擇池。使用空存檔的（μ，λ）-策略導致每個世代步驟都完全更換群體，個體的最大壽命只有一代。這種方法用於經典遺傳算法。（μ + λ）-策略防止存檔中的個體被更差適應度的後代替換。這種策略推薦用於進化策略。（α（μ）+ λ）-策略允許在兩個極端案例之間調整存檔更新方案。

optiSLang提供無參數約束處理方法，考慮群體中所有個體，這些個體根據其目標值和約束違反進行比較。不可行解決方案根據其約束違反排名，其適應度通過添加最差可行解決方案的適應度到排名值來修改。這個過程不需要額外的懲罰參數。此外，它可以處理多重約束而不需要約束值的縮放。即使所有個體都違反約束，該方法也可以引導搜索朝向約束空間的可行區域。

在optiSLang中，進化算法提供了預定的全局和局部搜索。全局搜索從隨機或手動選擇的起始群體開始，適合在整個設計空間探測新的可能解決方案。局部搜索嘗試改善單一設計而不發現新區域。在圖3.12：進化算法的收斂（第39頁），兩種搜索策略進行了比較。

Figure 3.12: Convergence of the Evolutionary Algorithm



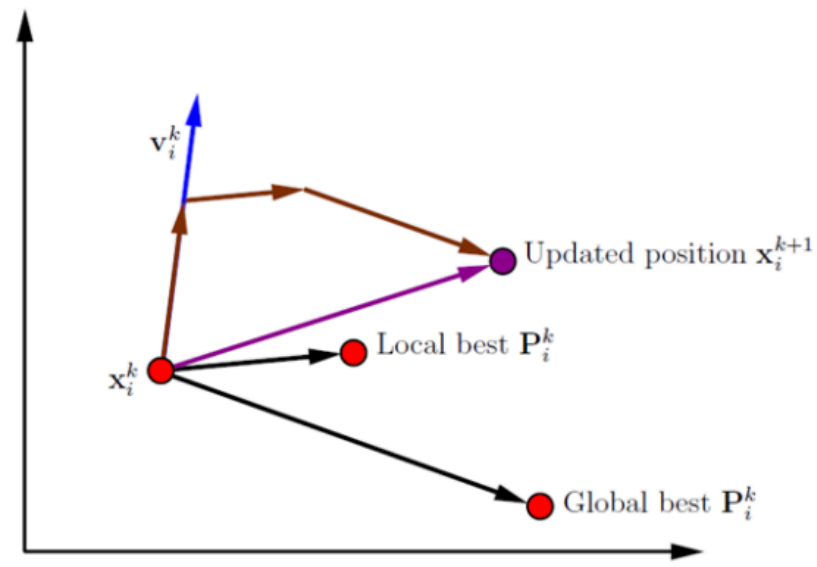
Convergence of the evolutionary algorithm with global search (left) and local search (right) for the damped oscillator with noisy objective function.

該圖顯示，從隨機起始群體開始的全局搜索在整個設計空間生成設計，而使用局部搜索方法生成的設計則接近起始設計。由於在進化算法中應用了隨機搜索策略，優化器可能並不總是非常接近全局最優解。例如，使用全局搜索為阻尼振子找到的最優解（m = 0.81公斤，k = 50.0牛頓/米）並不如ARSM優化器的結果好。因此，**通常使用進化算法來檢測全局最優解的區域，然後用其他優化方法進行進一步改進是有用的。**

粒子群優化

粒子群優化（PSO）是一種受自然啟發的優化方法，由（Kennedy和Eberhart 1995（第89頁））開發，模仿群體的社會行為。有關優先位置的信息將傳遞給群體中的其他個體，使它們移動到之前最佳位置的方向。在自然界中，一個個體可以是一隻鳥、蜜蜂或魚。關於這類算法的修改和改進的概述可以在（Engelbrecht 2005（第89頁））和（Poli等人，2007（第90頁））中找到。

Figure 3.13: Particle Swarm Optimization



Update of a particle position x_i^k in the Particle Swarm Optimization using a combination of the old velocity v_i^k, the direction to the local best position P_i^k and the direction to the global best position P_g^k.

PSO從初始化一個大小為μ的群體開始，每個個體代表優化問題的一個可能解決方案。群體智慧受到代表個人和全球行為的兩個主要組件的影響。這意味著每個個體記住其個人最佳解決方案，並且還會受到群體最佳解決方案的影響。每個個體將改變其位置xi，朝向其個人最佳找到的位置Pi和全球最佳找到的位置Pg。

$$
\begin{aligned}
v_i^{k+1} &= wv_i^k + c_p R_1 \times (P_i^k - x_i^k) + c_g R_2 \times (P_g^k - x_i^k) \\
x_i^{k+1} &= x_i^k + v_i^{k+1}
\end{aligned}
\tag{3.6}
$$

影響群體速度和擴散的有三個重要參數。慣性權重w是上一次迭代步驟k的速度的縮放因子。個人加速係數cp是第二項的縮放因子，這也被稱為認知組件。第三項，被稱為社會組件，由全球加速係數cg縮放。R1和R2是從[0, 1]中均勻選擇的兩個隨機數字向量。

適當的係數cp、cg和w的選擇對PSO的收斂行為非常重要。optiSLang提供了兩種預定義的搜索策略，每個係數有默認值。如果用戶對優化空間有初步信息，並且已經找到了預優化設計，則建議使用局部搜索策略。在這種情況下，群體運動在所有世代中較慢且不太強烈。在預定義的全球搜索策略中，群體運動在開始時非常強烈，並通過減少先前速度和認知組件的權重以及增加社會組件的權重，在優化過程中被阻尼。這樣，在前幾次迭代中達到全球探索，並且在最後幾次迭代中可能進行開發。

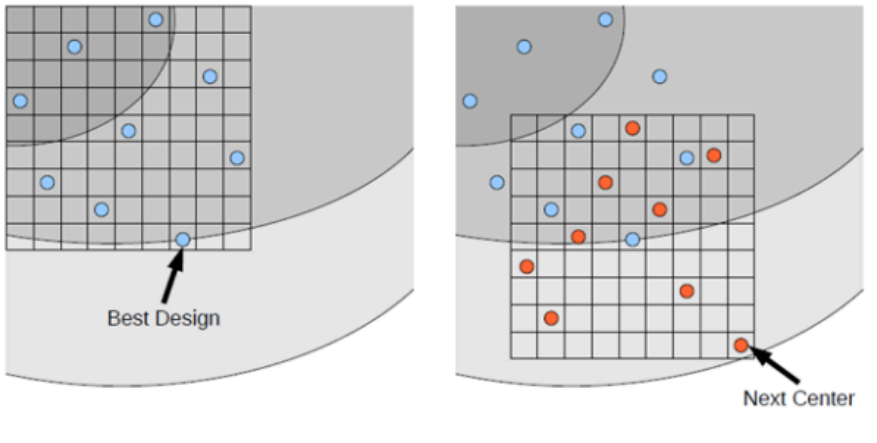
為了允許更複雜的搜索，optiSLang啟用了適用於進化算法的相同突變方法。要執行經典PSO搜索，必須取消選擇突變操作符。全球最佳解決方案記錄在存檔中，並在每個迭代步驟結束時更新。在存在約束條件的情況下，通過選擇具有最小目標值的可行解決方案來找到全球最佳解決方案。如果第一個群體中只有不可行的解決方案，則全球最佳將被選為違反約束最少的個體。

PSO與進化算法類似，用於具有大量失敗設計的優化問題。與進化算法相比，如果調查了離散設計變量和許多約束條件，則效率較低。對於具有連續設計變量的優化任務，PSO通常非常接近全球最優解。與進化算法類似，選擇合格的起始群體，例如從靈敏度分析中，可能顯著提高算法的效率。

隨機設計改進

隨機設計改進（Stochastic Design Improvement ,SDI）是一種局部的、單目標優化程序，它通過使用簡單的隨機方法改進提出的設計，而無需對設計空間中的潛在關係有廣泛的了解。基於一個初始起始設計，通過在給定範圍內均勻分佈的拉丁超立方抽樣生成大小為μ的起始群體。在每個迭代步驟中，最佳設計被作為下一次迭代抽樣的新中心點，如圖3.14所示：隨機設計改進（第42頁）。抽樣方案的範圍在每個迭代步驟中進行調整。根據優化問題，整個群體可能會移動到更好的區域，在每一步實現改進。使用在進化算法中應用的無參數約束處理方法，比較一次迭代中的所有個體關於目標和約束。如果達到了最大迭代次數，或者達到了與初始設計相比指定的改進（獲得改進），或者在兩次迭代步驟之間觀察到了指定的性能惡化，則算法收斂。

Figure 3.14: Stochastic Design Improvement



Update scheme of the Stochastic Design Improvement approach by generating a random sampling around the best design of a previous iteration.

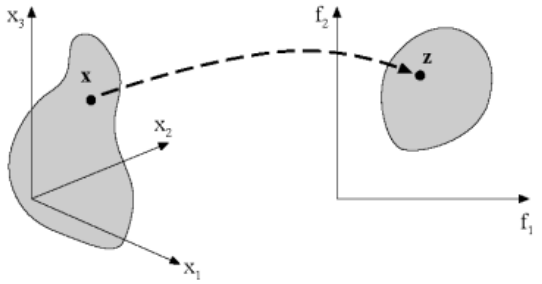
由於其純隨機方法，SDI非常魯棒，適用於大量失敗設計、離散和連續參數，以及大量設計變量和約束條件。然而，它旨在改進初始設計，而非找到全局最優解。因此，與進化算法和粒子群優化相比，其效率可能顯著較低。

3.2. 多目標優化

大多數現實世界的優化問題包括多個目標和大量的設計變量。這導致了多目標優化問題的一般化公式：

$$\begin{aligned} &\text{最小化：} f_m(\mathbf{x}), \quad m = 1, 2, ..., M \\ &\text{受限於：} g_j(\mathbf{x}) \geq 0, \quad j = 1, 2, ..., J \\ &h_k(\mathbf{x}) = 0, \quad k = 1, 2, ..., K \\ &\mathbf{x}_i^{(L)} \leq \mathbf{x}_i \leq \mathbf{x}_i^{(U)}, \quad i = 1, 2, ..., n \end{aligned} \tag{3.7}$$

Figure 3.15: Design Space (Left) and Objective Space (Right)



其中 $\mathbf{x} = (x_1, x_2, ..., x_n)^T$ 為設計變量向量，且不等式 $g_j(\mathbf{x})$ 和等式 $h_k(\mathbf{x})$ 約束。所有符合約束和變量界限的解構成可行的n維設計空間。每個解 $\mathbf{x}$ 被指定為一個向量 $f(\mathbf{x}) = \mathbf{z} = (z_1, z_2, ..., z_M)^T$ 描述一個M維目標空間中的點。這是與單目標優化問題的主要差異，在單目標優化問題中只有一個目標。設計空間和目標空間之間的映射關係在圖3.15中示出：設計空間（左）和目標空間（右）（第43頁）。

多種程序已被開發來解決多目標優化問題（參見Branke等人，2008年，第89頁）。它們被分類為非互動式和互動式方法。在非互動式方法中，多目標優化問題被轉化為單一目標問題。這可以通過選擇一個偏好的目標函數並將剩餘的目標作為約束來完成，例如：

$$\begin{aligned} \tilde{f}(\mathbf{x}) &= f_i(\mathbf{x}) \\ f_{j\neq i}(\mathbf{x}) &\leq f_j^{\text{limit}} \end{aligned} \tag{3.8}$$

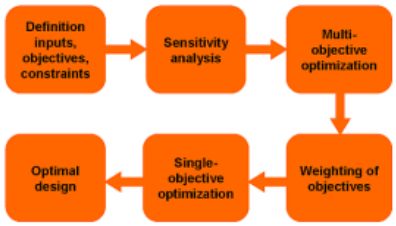
另一種可能性是通過使用個別權重 w_i 將所有目標結合在一個函數中：

$$\tilde{F}(\mathbf{x}) = \sum_{i=1}^{M} w_i f_i(\mathbf{x}) \tag{3.9}$$

這兩種方法都需要對與每個目標相關的優化潛力有很好的了解，尤其是對不同目標的重要性有清晰的了解。

在設計過程的早期階段，這樣的信息通常很少，且事先無法將多目標任務轉化為單一目標任務。在這些情況下，互動式方法是更實用的。在optiSLang中，帕雷托優化作為最受歡迎的互動式方法之一。

Figure 3.16: Recommended Flowchart for Multi-Objective Optimization



在圖3.16：多目標優化的推薦流程圖（第44頁）中，顯示了推薦的優化流程：最初確定輸入、可能的目標和約束函數。靈敏度分析用來檢測不重要的參數和約束，並在此之後進行了價值衝突分析，以確定在相互衝突的目標中優化的潛力，並導出適合進行單一目標優化的策略。最終，優化程序確定了一個最佳設計。

3.2.1. 帕雷托優化

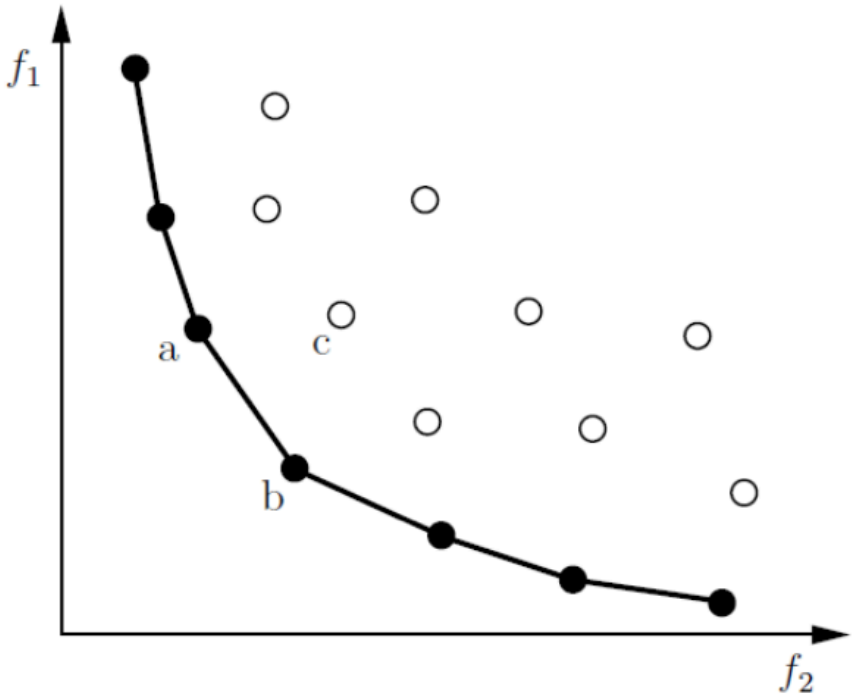
解決多目標優化問題需要一個考慮不同目標的標準。一個可能的標準是帕雷托支配性，這是為兩個決策向量a和b在設計空間X中制定的，如下所述：

- 如果解決方案a ∈ X在所有目標中至少在一個目標上是可行且優於或等於解決方案b，則稱解決方案a支配解決方案b。
- 如果解決方案a ∈ X在所有目標上既不支配也不被解決方案b支配，則解決方案a對於解決方案b是無差異的。

使用支配標準允許在不同決策向量之間建立部分秩序，如圖3.17：帕雷托（第45頁）所示。一個解決方案是帕雷托最優的，如果沒有其他決策向量可以改進一個目標而不至少在另一個目標上造成惡化。換句話說，一個解決方案是帕雷托最優的，如果它不被任何其他解決方案支配。帕雷托這個術語是以維爾弗雷多·帕雷托命名的，他是一位使用這一概念於經濟效率研究中的意大利經濟學家。

如果所有目標同等重要，且事先沒有設定偏好，一個解決方案的支配性是確定它是否優於其他解決方案的唯一方法。因此，非支配子集構成了帕雷托集。在目標空間中相應的點稱為帕雷托前沿。

Figure 3.17: Pareto



Pareto dominating (filled circles) and dominated designs (unfilled circles) including the resulting Pareto frontier of two conflicting objectives. (*a* and *b* dominate *c* but are indifferent to each other)

對於多目標優化，可以制定以下要求：

- 找一組接近帕雷托最優解的解決方案（收斂）。
- 尋找足夠多樣化的解決方案以代表整個帕雷托前沿（多樣性）。

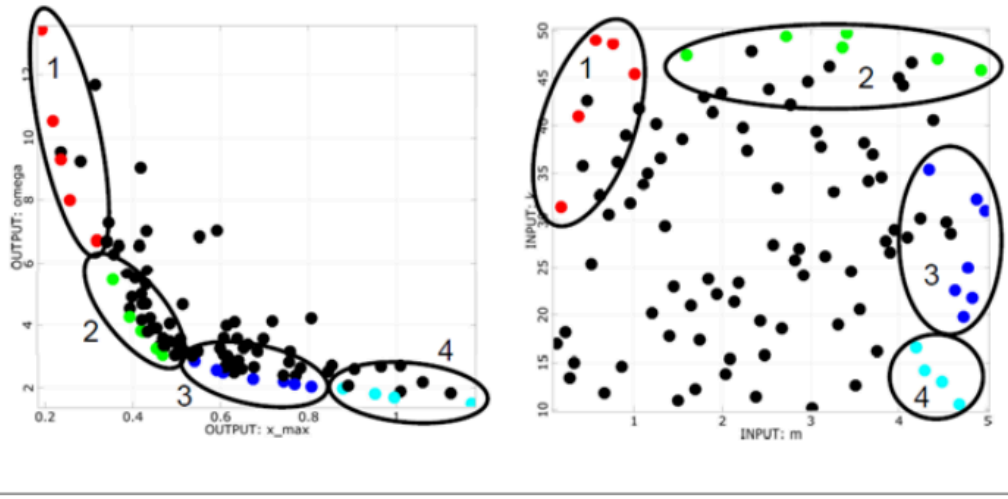
儘管優化產生了一組帕雷托最優解，用戶對於獲得唯一一個解決方案感興趣。必須引入額外的偏好，這需要對問題有全面的了解，以選擇滿足要求的解決方案。

衝突目標的分析

為了獲得不同的折衷解決方案，必須存在衝突的目標。前期靈敏度分析的結果應用於檢查關於可能衝突的目標函數。

在振子的例子中，阻尼固有頻率作為第二個目標函數，應最小化。用於靈敏度分析和設計探索的樣本的蟻丘圖顯示在圖3.18：衝突目標（第46頁）。圖表顯示最大振幅的最小化與阻尼固有頻率的最小化相衝突。在蟻丘圖中可以識別所有帕雷托支配設計。進一步的信息可以通過將目標空間中的帕雷托最優設計細分為在設計空間中有特定區域的群組來獲得。圖表比如指出群組2的設計在目標空間中高度集中，但在設計空間中廣泛分佈。此外，所有帕雷托最優設計都接近設計空間邊界。

Figure 3.18: Conflicting Objectives



Conflicting objectives of the damped oscillator (maximum amplitude vs. eigen-frequency) including cluster analysis of the Pareto optimal designs in the objective space (left) and in the design space (right).

多目標群體基方法

在過去的二十年中，進化算法已被確立用於解決多目標優化問題。平行搜索一組帕雷托最優解是這種方法的主要優勢。optiSLang的實現基於強度帕雷托進化算法2（SPEA2）（Zitzler等人，2001年，第90頁），其特點是以下原則：

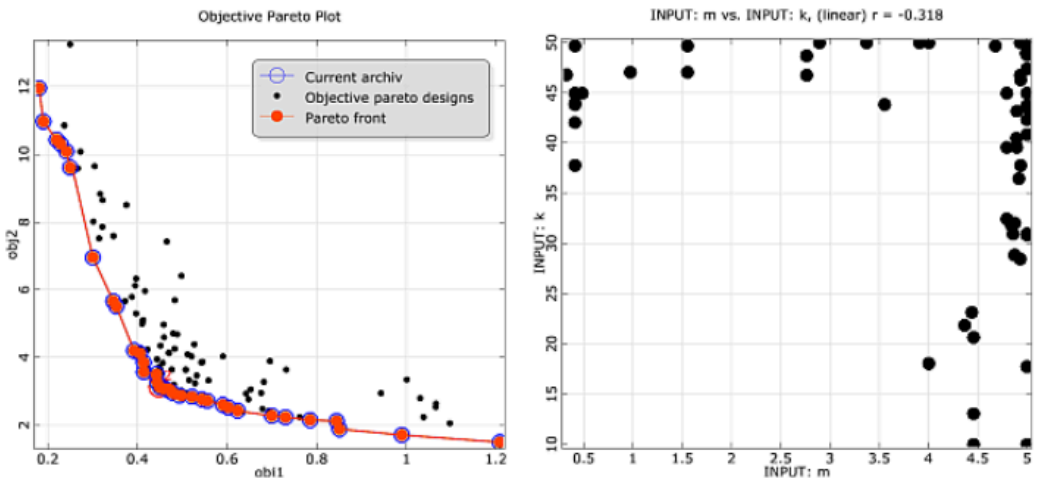
- 通過使用非支配個體的存檔應用精英策略。
- 適應度分配基於支配標準。
- 通過密度估計實現保持群體多樣性。

算法從隨機或手動初始化第一代群體大小為μ開始，接著對所有個體進行評估。對於第一代，從群體中選出λ（父母數量）個個體進行繁殖。評估所有新生後，存檔個體和後代都被賦予新的適應度值。α最佳解構成下一代的存檔。如果達到了最大代數或存檔停滯，則優化停止。

粒子群優化算法也可以應用於多目標優化問題。正如單目標PSO所提及的，optiSLang選擇最適合的設計作為全球領導者。對於下一個迭代步驟，群體將朝這個方向移動。在多目標優化中，適應度評估與單目標方法不同。

與單目標自然啟發優化方法相比，多目標EA和PSO的適應度分配總是考慮整個群體，因為標準是基於個體間的比較。適應度分配方法是一種基於支配的排名，它考慮到一個解決方案被多少個體支配以及一個解決方案支配多少個體。為了保持帕雷托前沿的多樣性，向適應度分配程序引入了額外的標準。個體與其第k近鄰的距離作為密度的估計器。意圖是偏好目標空間中人群較少的區域的解決方案，如邊界解決方案。

Figure 3.19: Pareto Frontier



Pareto frontier for the damped oscillator (left) and Pareto optimal designs plotted in the design space (right) obtained by evolutionary algorithms.

在圖3.19：帕雷托前沿（第47頁）中，顯示了多目標進化算法對阻尼振子例子的結果帕雷托前沿。帕雷托前沿與圖3.18：衝突目標（第46

頁）中的帕雷托最優設計相符。帕雷托前沿現在可以用來判斷不同優化目標的重要性，並選擇合適的最優設計。選擇合適的起始群體而不是純隨機生成，尤其對於高維優化問題，可以顯著提高多目標優化方法的效率。應使用靈敏度分析或先前優化運行的選定設計來實現此目的。

3.3. 一鍵優化

一鍵優化（One-Click Optimization, OCO）方法是由Ansys開發的一種混合型和動態優化方法。它有效地結合了直接（高保真模型）和MOP（低保真模型）輔助搜索策略。OCO是一個通用優化器，**能夠自動且反覆選擇optiSLang提供的最合適優化方法**。

OCO考慮了optiSLang中可用的允許優化方法，例如NLPLQ、單純形法、進化算法等。它通過使用一個動態選擇方案來解決大多數選擇方案的不足，該方案從探索設計空間開始，以研究響應的統計特徵。然後引入了一個稱為成功因子的內部選擇標準來比較優化算法方案。這個成功因子考慮了評估次數和在目標和可行性方面的預估改進。

OCO最創新的方面是它的動態和適應性。實際上，它允許根據算法的行為來決定是連續使用不同的算法，還是使用一個算法。例如，如果一個算法表現良好，OCO就會繼續使用它。相反，如果另一個算法看起來更有前途，OCO會動態切換到預期更有希望的算法。

這種動態切換系統可以幫助避免局部最小值、任意變化和手動選擇帶來的額外負擔。

默認情況下，OCO公開了一個主要設置來控制評估的最大次數。選擇使用代理模型來構建MOP，以及允許的優化方法列表是預先定義的。高級設置和專家設置可用於自定義OCO。