Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. Mario Ohlberger und Dr. Stephan Rave

19. Juni 2017

Übung zum Programmierpraktikum

Numerische Mehrskalenmethoden und Modellreduktion

Sommersemester 2017 — Blatt 4

Aufgabe 1 (RB-Projektion)

(a) Schreiben Sie eine Funktion `solve_reduced(d, basis, mu)`, welche zu einer gegebenen Diskretisierung `d` und einer reduzierten Basis `basis` die RB-Approximation der Lösung zum Parameter `mu` bestimmt. Verwenden Sie dabei die `apply`- und `apply2`-Methoden von `d.operator` und `d.rhs`, um das reduzierte Problem zu assemblieren.

(b) Berechnen Sie mittels `solve_reduced` reduzierte Lösungen der parametrischen Probleme von Blatt 1 für verschiedene Basisgrößen (mit oder ohne Greedy-Verfahren), und visualisieren Sie die reduzierte Lösung zusammen mit der hochdimensionalen Lösung sowie der Differenz der beiden Lösungen.

(c) Bestimmen Sie empirisch den maximalen Modellreduktionsfehler und plotten Sie diesen zusammen mit dem maximalen Fehler der orthogonalen Projektion auf den RB-Raum in Abhängigkeit von der Basisgröße.

Aufgabe 2 (Offline-Online-Zerlegung)

(a) Benutzen Sie `pymor.algorithms.projection.project`, um für das Problem von Blatt 1 Aufgabe 2c) einen RB-projizierten, affin-zerlegten Systemoperator und ein RB-projiziertes Rechte-Seite-Funktional zu erhalten. Wiederholen Sie hiermit die Experimente von Aufgabe 1. Messen Sie dabei zusätzlich die durchschnittlich benötigte Zeit zur Lösung des reduzierten Problems (ohne hochdimensionale Rekonstruktion), und plotten Sie diese gegen die Basisgröße.

Hinweise: Nutzen Sie `projected_rhs.as_vector()` um eine `VectorArray`-Darstellung der rechten Seite zu erhalten.

(b) Wiederholen Sie das Experiment für die übrigen beiden Probleme von Blatt 1 Aufgabe 2.

(c) Passen Sie die Implementierungen der Probleme von Blatt 1 Aufgabe 2 a) und b) so an, dass auch für diese eine Offline-Online-Zerlegung erreicht wird. Stellen Sie hierzu sicher, dass `mu` nur in den Koeffizienten einer `LincombFunction` auftritt.

Aufgabe 3 (Reduktoren und Fehlerschätzer)

(a) Vereinfachen Sie den Code aus Aufgabe 2, indem Sie `GenericRBReductor` verwenden, um eine RB-projizierte reduzierte Diskretisierung zu erhalten.

(b) Verwenden Sie `CoerciveRBReductor`, um zusätzliche einen Fehlerschätzer für den Modellreduktionsfehler zu assemblieren. Leiten Sie hierfür zunächst eine geeignete untere Schranke für die Koerzivitätskonstante des jeweils betrachteten Problems her. Plotten Sie die empirisch bestimmte minimale und maximale Effizienz des Fehlerschätzers gegen die Basisgröße.

Hinweise: Der Fehlerschätzer kann mit Hilfe der `estimate`*-Methode der reduzierten Diskretisierung ausgewertet werden.*

(c) Versuchen Sie im Quellcode von `pyMOR` die Funktionsweise von `CoerciveRBReductor.reduce()` nachzuvollziehen. Wie unterscheidet sich die Offline-Online-Zerlegung des Fehlerschätzers in `CoerciveRBReductor` von der in der Vorlesung behandelten Zerlegung?