(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년11월09일
(11) 등록번호 10-1917399
(24) 등록일자 2018년11월05일

(51) 국제특허분류(Int. Cl.)
G05B 13/04 (2006.01)
(21) 출원번호 10-2013-7011026
(22) 출원일자(국제) 2011년09월29일
심사청구일자 2016년09월29일
(85) 번역문제출일자 2013년04월29일
(65) 공개번호 10-2013-0124321
(43) 공개일자 2013년11월13일
(86) 국제출원번호 PCT/US2011/053974
(87) 국제공개번호 WO 2012/050970
국제공개일자 2012년04월19일
(30) 우선권주장
12/893,670 2010년09월29일 미국(US)
(56) 선행기술조사문헌
Pascal Gahinet 외 3. LMI Control Toolbox User's Guide., 1995.*
Musa A. Mammadov 외 1. A Nonsmooth Optimization Approach to H∞Synthesis. IEEE Conference on Decision and Control, and the European Control Conference 2005. 2005., pp. 6893-6898.*
US20060112382 A1
JP2008310601 A
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
더 매쓰웍스 인코포레이티드
미국 매사추세츠 내틱 애플 힐 드라이브 3 (우:01760-2098)
(72) 발명자
가히넷 파스칼
미국 매사추세츠주 01748 홉킨톤 웨지우드 드라이브 24
아프카리안 피에르
프랑스 톨루즈 에프-31400 아파트 에이25 루 두 독테우르 다스쿠 3
놀 도미니쿠스
프랑스 라우저빌 에프-31650 하메아우 드 도우멘 조우 29
(74) 대리인
김태홍

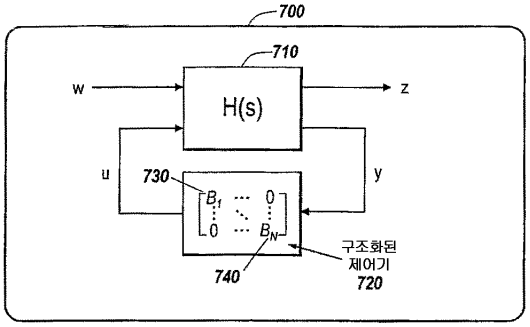
전체 청구항 수 : 총 24 항 심사관 : 이종환

(54) 발명의 명칭 **다중 입력 다중 출력 제어 (MIMO) 구조들을 제어하기 위한 상호작용식 시스템**

(57) 요 약

예시적인 실시예들은 사용자들이 다중 변수 피드백 제어 문제들을 상호작용식으로 포뮬레이팅하고 해결하도록 허용한다. 예를 들어, 사용자들은 복수의 제어 엘리먼트들이 하나 이상의 피드백 루프들에 걸쳐서 분산되고 제어 시스템의 전체 성능 및 강건성을 최적화하기 위해 공동으로 튜닝될 필요가 있는 문제들을 해결할 수 있다. 실시예들은 사용자들이 사용자에게 친숙한 포맷들에서 설계 요건들 및 목적들을 특정하도록 허용한다. 실시예들은 사용자에 의해 제공되는 설계 요건들 및/또는 목적들을 만족시키는 방식으로 제어 문제를 해결하도록 튜닝 가능한 파라미터들에 대해 동작할 수 있다.

대 표 도 - 도7



명 세 서

청구범위

청구항 1

컴퓨터 구현 방법에 있어서,

하나 이상의 파라미터를 각각이 포함하는 하나 이상의 튜닝 가능한 컴포넌트를 모델에서 식별하는 단계로서, 상기 식별은 컴퓨팅 디바이스에 의해 수행되는 것인, 단계;

상기 하나 이상의 튜닝 가능한 컴포넌트 중 적어도 하나를 포함하는 하나 이상의 피드백 루프를 식별하는 단계로서, 상기 하나 이상의 피드백 루프는 사용자 디바이스로부터 수신되는 제1 명령어에 기초하여 식별되는 것이고, 상기 하나 이상의 피드백 루프를 식별하는 것은 상기 컴퓨팅 디바이스에 의해 수행되는 것인, 단계;

표시를 위해 상기 제1 명령어에 기초하여 플롯을 제공하는 단계로서, 상기 표시를 위한 제공은 상기 컴퓨팅 디바이스에 의해 수행되는 것인, 단계;

상기 제공된 플롯에 기초하여, H∞ 제약의 형성에 연관된 제2 명령어를 수신하는 단계로서, 상기 제2 명령어는 상기 사용자 디바이스로부터 수신되고, 상기 제2 명령어를 수신하는 것은 상기 컴퓨팅 디바이스에 의해 수행되는 것인, 단계;

상기 제2 명령어 및 설계 요건을 수신하는 것에 기초하여, 상기 식별된 하나 이상의 피드백 루프에 연관된 정보를 표현하기 위한 H∞ 제약을 포뮬레이팅(formulate)하는 단계로서, 상기 H∞ 제약은 하나 이상의 전달 함수(transfer function)와 연관되고, 상기 포뮬레이팅은 상기 컴퓨팅 디바이스에 의해 수행되는 것인, 단계;

상기 하나 이상의 튜닝 가능한 컴포넌트를 파라미터화하는 단계로서, 상기 파라미터화는 상기 컴퓨팅 디바이스에 의해 수행되는 것인, 단계; 및

상기 하나 이상의 튜닝 가능한 컴포넌트의 파라미터화에 기초하여, 상기 H∞ 제약을 강제하기 위해 상기 하나 이상의 파라미터들을 튜닝함으로써 상기 식별된 하나 이상의 피드백 루프를 튜닝하기 위해 상기 하나 이상의 튜닝 가능한 컴포넌트의 상기 하나 이상의 파라미터와 상호작용하는 단계로서, 상기 상호작용은 상기 컴퓨팅 디바이스에 의해 수행되는 것인, 단계

를 포함하는, 컴퓨터 구현 방법.

청구항 2

제1항에 있어서,

상기 식별된 하나 이상의 피드백 루프를 튜닝하기 위해 상기 하나 이상의 튜닝 가능한 컴포넌트의 상기 하나 이상의 파라미터와 상호작용할 때, 상기 방법은,

MATLAB-호환 가능한 언어를 이용하여 상기 식별된 하나 이상의 피드백 루프를 튜닝하는 단계를 포함하는 것인, 컴퓨터 구현 방법.

청구항 3

제1항에 있어서,

그래픽 모델링 환경에 이용되는 블록 세트로부터 상기 하나 이상의 튜닝 가능한 컴포넌트를 선택하는 단계를 더 포함하는, 컴퓨터 구현 방법.

청구항 4

제1항에 있어서,

상기 식별된 하나 이상의 피드백 루프를 튜닝하기 위해 상기 하나 이상의 튜닝 가능한 컴포넌트의 상기 하나 이상의 파라미터와 상호작용할 때, 상기 방법은,

상기 식별된 하나 이상의 피드백 루프를 튜닝하는 것과 연관된 속도를 결정하는 단계로서, 상기 속도는 상호작용식 설계를 지원하도록 결정되는 것인, 단계; 및

상기 식별된 하나 이상의 피드백 루프를 튜닝하는데 상기 결정된 속도를 이용하는 단계를 더 포함하는, 컴퓨터 구현 방법.

청구항 5

삭제

청구항 6

삭제

청구항 7

프로세서 상에서 실행될 때 튜닝 가능한 컴포넌트를 정적으로 특정하고, 튜닝 가능한 컴포넌트를 동적으로 특정하며, 상기 튜닝 가능한 컴포넌트의 파라미터와 상호작용하기 위한 API를 구현하는 실행 가능한 명령어들을 보유하는 하나 이상의 비일시적 컴퓨터-판독 가능한 매체에 있어서,

상기 매체는,

컴포넌트의 미리 정의된 세트에 대한 파라미터화(parameterization)를 실현(embody)하는 미리 정의된 인터페이스를 식별하는 동작;

산술 연산(arithmetic operation)들의 세트를 구현하는 동작,

도우미 함수(helper function)들의 세트를 구현하는 동작,

상기 산술 연산들 및 상기 도우미 함수들을 이용하여 튜닝 가능한 컴포넌트를 동적으로 생성하는 동작으로서, 상기 동적으로 생성하는 동작은 또한, 기초 파라미터 컴포넌트와 고정된 계수 또는 고정된 컴포넌트를 조합하는 동작을 포함하는 것인, 상기 생성 동작; 및

상기 튜닝 가능한 컴포넌트의 파라미터적 모델(parametric model)을 생성하는 동작

을 위한 하나 이상의 실행 가능한 명령어들을 보유하며,

상기 파라미터적 모델은,

상기 튜닝 가능한 컴포넌트의 튜닝 가능한 파라미터를 설명(account for)하고,

사용자 입력이 상기 튜닝 가능한 파라미터와 상호작용하도록 허용하며,

상기 사용자 입력은,

상기 튜닝 가능한 파라미터를 초기화하고,

상기 튜닝 가능한 파라미터를 고정하거나,

상기 튜닝 가능한 파라미터 중 선택된 파라미터를 해제(free)하는 것인, 비일시적 컴퓨터-판독 가능한 매체.

청구항 8

제7항에 있어서,

상기 튜닝 가능한 컴포넌트는 MATLAB-호환 가능한 언어에서 구현되는 것인, 비일시적 컴퓨터-판독 가능한 매체.

청구항 9

제7항에 있어서,

상기 튜닝 가능한 컴포넌트는 블록 세트로부터 선택되고, 상기 블록 세트는 그래픽 모델링 환경과 연관되는 것인, 비일시적 컴퓨터-판독 가능한 매체.

청구항 10

프로세서 상에서 실행될 때, 임의의 피드백 제어 구조의 표준 형태를 구축하고 설계 요건의 H∞ 포뮬레이션에서 이용되는 점 대 점 전달 함수를 특정하기 위해 애플리케이션 프로그램 인터페이스(application program interface; API)를 구현하는 실행 가능한 명령어들을 보유하는 하나 이상의 비일시적 컴퓨터-판독 가능한 매체에 있어서,

상기 매체는,

사용자 입력 매커니즘과 상호작용하는 동작으로서, 상기 사용자 입력 매커니즘은 입력 신택스를 이용해서 산술 연산자 및 블록 도식화 연산(block diagramming operations)을 특정하고, 상기 입력 신택스는 사용자가,

선형 시간 불변 모델(linear time invariant model)에서 이용되는 선형 시간 불변 모델 컴포넌트를 입력하고;

튜닝 가능한 파라미터를 갖는 튜닝 가능한 컴포넌트를 기술하는 소프트웨어 기반 인터페이스를 입력하도록 허용하는 것인, 상기 상호작용 동작;

산술 연산자 및 블록 도식화 연산 중 하나 이상을 이용하여 상기 소프트웨어 기반 인터페이스와 상기 선형 시간 불변 모델 컴포넌트를 조합하는 동작으로서, 상기 블록 도식화 연산은,

직렬 연결,

병렬 연결, 또는

피드백 연결을 포함하는 것인, 상기 조합 동작; 및

상기 입력 신택스가,

상기 선형 시간 불변 모델 컴포넌트 및 상기 튜닝 가능한 컴포넌트를 포함하는 전체 제어 시스템의 표준 형태를 점증적으로(incrementally) 구성하고,

상기 표준 형태에 기초하여 파라미터적 모델을 생성하도록

허용하는, 상기 사용자 입력 매커니즘과의 상호작용 동작

을 위한 하나 이상의 실행가능 명령어들을 보유하며,

상기 파라미터적 모델은 최적화기에 대한 입력을 위해 적응되고, 설계 요건을 충족하도록 튜닝하기 위해 구성되는 것인, 비일시적 컴퓨터-판독 가능한 매체.

청구항 11

제10항에 있어서, 상기 하나 이상의 실행가능 명령어들은,

최적화기를 이용하여 상기 파라미터적 모델을 최적화하는 동작을 위한 명령어들을 더 포함하고,

상기 최적화하는 동작은, 상기 표준 형태 및 상기 튜닝 가능한 파라미터와 상호작용하여 H∞ 목적을 최소화하거나 또는 H∞ 제약을 강제하는 것인, 비일시적 컴퓨터-판독 가능한 매체.

청구항 12

제10항에 있어서, 상기 하나 이상의 실행가능 명령어들은,

상기 사용자 입력 매커니즘에 도우미(helper) 함수를 제공하는 동작을 위한 명령어들을 더 포함하고,

상기 사용자가 상기 입력 신택스를 이용하여 상기 API와 상호작용할 때 상기 도우미 함수는 상기 입력 매커니즘을 통해 상기 사용자에 액세스 가능한 것인, 비일시적 컴퓨터-판독 가능한 매체.

청구항 13

제10항에 있어서, 상기 하나 이상의 실행가능 명령어들은,

상기 입력 매커니즘과 호환 가능하고, 상기 입력 신택스에 따라 상호작용되는 함수를 제공하는 동작; 및

상기 제어 시스템에 질의하거나, 상기 제어 시스템을 분석하기 위해 상기 함수를 이용하는 동작

을 위한 명령어들을 더 포함하는, 비일시적 컴퓨터-판독 가능한 매체.

청구항 14

제10항에 있어서,

상기 표준 형태는 MATLAB-호환 가능한 언어를 이용하여 구성되는 것인, 비일시적 컴퓨터-판독 가능한 매체.

청구항 15

제10항에 있어서,

상기 튜닝 가능한 컴포넌트는 그래픽 모델링 환경에 이용되는 블록 세트로부터 선택되는 것인, 비일시적 컴퓨터-판독 가능한 매체.

청구항 16

삭제

청구항 17

제10항에 있어서,

상기 임의의 피드백 제어 구조는 이득 스케줄링을 이용하여 제어되는 것인, 비일시적 컴퓨터-판독 가능한 매체.

청구항 18

루프 형상화 요건을 H∞ 포뮬레이션(formulation) 내로 프로그램적으로(programmatically) 포뮬레이팅하기 위한 컴퓨터-구현 방법에 있어서,

개루프 이득(open-loop gain)을 위한 타겟 형상 또는 상기 타겟 형상에 대한 프록시(proxy)를 수신하는 단계; 및

소프트웨어 툴(tool)과 상호작용하는 단계

를 포함하고,

상기 소프트웨어 툴은,

상기 타겟 형상 또는 상기 프록시로부터 제어 구조 및 필터를 유도하고,

상기 유도된 제어 구조 및 필터로부터 표준 형태를 구성하고,

상기 유도된 제어 구조 및 필터로부터 H∞ 제약을 구성하며,

상기 표준 형태 및 상기 H∞ 제약은 제어 시스템에 대한 설계 요건을 포착하는 것인, 루프 형상화 요건을 H∞ 포뮬레이션 내로 프로그램적으로 포뮬레이팅하기 위한 컴퓨터-구현 방법.

청구항 19

제18항에 있어서,

상기 표준 형태는 MATLAB-호환 가능한 언어를 이용하여 구성되는 것인, 루프 형상화 요건을 H∞ 포뮬레이션 내로 프로그램적으로 포뮬레이팅하기 위한 컴퓨터-구현 방법.

청구항 20

제18항에 있어서,

상기 소프트웨어 툴은 Simulink-호환 가능한 환경 또는 Labview-호환 가능한 환경과 상호작용하는 것인, 루프 형상화 요건을 H∞ 포뮬레이션 내로 프로그램적으로 포뮬레이팅하기 위한 컴퓨터-구현 방법.

청구항 21

삭제

청구항 22

표준 형태의 구조를 이용하기 위한 컴퓨터-구현 방법에 있어서,

상기 표준 형태의 구조를 이용하는 것은 튜너 알고리즘(tuner algorithm)의 성능을 강화하고,

상기 방법은,

최적화 프로시저 동안 최적화기에 의해 공급되는 튜닝 가능한 파라미터의 값을 수신하는 단계;

구성 프로시저를 이용하여 상기 튜닝 가능한 파라미터 값을 위한 상기 표준 형태의 상태-공간 모델(state-space model)을 구성하는 단계로서, 상기 구성 단계는,

상기 튜닝 가능한 파라미터와 연관된 튜닝 가능한 컴포넌트를 구현하도록 소프트웨어 객체(object)를 이용하고,

각각의 소프트웨어 객체가, 상기 수신된 파라미터 값을 위한 상기 각각의 소프트웨어 객체의 상태-공간 표현의 제공을 책임지게 하고,

상기 튜닝 가능한 컴포넌트를 위한 애그리게이팅(aggregating)된 상태-공간 매트릭스(state-space matrix)를 생성하도록 상태-공간 매트릭스를 애그리게이팅하며,

상기 표준 형태의 원하는 폐루프 상태-공간 모델을 획득하기 위해 럼핑된 플랜트 모델(lumped plant model)의 상태-공간 표현과 상기 튜닝 가능한 컴포넌트의 애그리게이팅된 상태-공간 매트릭스를 조합함으로써

수행되는 것인, 상기 구성 단계;

후속 그래디언트 계산(gradient computations)을 가속화하기 위해 상기 구성 프로시저의 중간 결과를 캐싱(caching)하는 단계; 및

그래디언트 정보를 계산하는 단계

를 포함하고,

상기 그래디언트 정보의 계산은,

목적 및 제약을 구분하고,

상기 표준 형태의 상태-공간 모델을 구성하기 위해 이용된 동일한 소프트웨어 객체를 이용하고,

각각의 소프트웨어 객체가 상기 수신된 파라미터에 관하여 스칼라값 함수(scalar-valued function) - 상기 스칼라값 함수는 규칙을 적용한 부산물(by-product)임 - 의 그래디언트의 제공을 책임지게 하며;

상기 목적 및 제약의 전체 그래디언트로 각각의 튜닝 가능한 컴포넌트에 의해 공급되는 그래디언트 데이터를 애그리게이팅하도록 상기 캐싱된 중간 결과를 이용함으로써

수행되는 것인, 표준 형태의 구조를 이용하기 위한 컴퓨터-구현 방법.

청구항 23

제22항에 있어서,

상기 그래디언트 정보는 프로세서 대신 부가적인 계산을 요구함 없이 이용 가능하게 되는 것인, 표준 형태의 구조를 이용하기 위한 컴퓨터-구현 방법.

청구항 24

제22항에 있어서,

폐루프 시스템은 상기 튜닝 가능한 파라미터의 정해진 값에 대해 유도되는 것인, 표준 형태의 구조를 이용하기 위한 컴퓨터-구현 방법.

청구항 25

제22항에 있어서,

상기 목적 및 제약을 구분하기 위해 체인 규칙(chain rule)이 이용되는 것인, 표준 형태의 구조를 이용하기 위한 컴퓨터-구현 방법.

청구항 26

제22항에 있어서,

상기 튜너 알고리즘은 실시간으로 동작하는 것인, 표준 형태의 구조를 이용하기 위한 컴퓨터-구현 방법.

청구항 27

제26항에 있어서,

상기 튜너 알고리즘은, 하나 이상의 필드 프로그래밍 가능한 게이트 어레이(field programmable gate arrays; FPGA), 주문형 집적 회로(application specific integrated circuits; ASIC), 주문형 집적 프로세서(application specific integrated processors; ASIP), 프로그래밍 가능한 로직 디바이스(programmable logic devices; PLD), 다중-코어 디바이스, 그래픽 프로세싱 유닛(graphics processing units; GPU), 디지털 신호 프로세서(digital signal processors; DSP), 또는 코어에서 구현되는 것인, 표준 형태의 구조를 이용하기 위한 컴퓨터-구현 방법.

청구항 28

제26항에 있어서,

상기 튜너 알고리즘은 분산된 컴퓨팅 환경에서 구현되는 것인, 표준 형태의 구조를 이용하기 위한 컴퓨터-구현 방법.

발명의 설명

기 술 분 야

[0001] 본 발명 출원은 2010년 9월 29일 출원된 미국 가특허 출원 번호 제12/893670호를 우선권으로 주장하며, 상기 가출원의 내용은 인용에 의해 본원에 포함된다.

발명의 내용

해결하려는 과제

과제의 해결 수단

[0002] 본 발명은 임의의 피드백 제어 구조에서 설계 파라미터들을 튜닝(tuning)하기 위한 컴퓨터 구현 방법을 제공하며, 이 방법은 튜닝되는 하나 이상의 자유(free) 파라미터들을 포함하는 하나 이상의 튜닝 가능한 컴포넌트들을 식별하는 단계; 상기 하나 이상의 튜닝 가능한 컴포넌트들을 포함하는 하나 이상의 피드백 루프들을 식별하는 단계; 상기 임의의 피드백 제어 구조를 표준 형태 - 상기 표준 형태는, 상기 임의의 피드백 제어 구조에서 알려진 그리고 고정된 컴포넌트를 포함하는 럼핑된 선형 모델(lumped linear model); 및 상기 튜닝 가능한 컴포넌트들을 이용하여 형성되는 모음(collection)을 포함하고, 상기 모음은 블록 대각 방식(block diagonal fashion)으로 그룹핑됨 - 로 변환하는 단계; 설계 목적들 및 설계 요건들을 표현하기 위한 H∞ 목적들 또는 제약들 - 상기 H∞ 목적들 또는 제약들은 폐루프 시스템에서 하나 이상의 점 대 점 전달 함수(point-to-point transfer function)들과 관련됨 - 을 이용하는 단계; 상기 튜닝 가능한 컴포넌트들을 정적으로 또는 동적으로 파라미터화하는 단계; 상기 파라미터화에 기초하여 상기 튜닝 가능한 컴포넌트들의 자유 파라미터들과 상호작용하는 단계; 및 비평활(non-smooth) H∞ 최적화 알고리즘들의 클래스(class)로부터 선택된 일원(member)에 기초하는 튜너(tuner)를 이용하여 상기 피드백 제어 구조를 튜닝하는 단계를 포함하고, 상기 튜너는, 상기 표준 형태로 동작하고, 상기 튜닝 가능한 컴포넌트들이 블록 대각 형태에 있을 때 상기 튜닝 가능한 파라미터들 상에서 동작하고, 상기 파라미터들을 튜닝하여 상기 H∞ 목적들을 최소화하거나 상기 H∞ 제약들을 강제(enforce)하는

것이다.

[0003] 또한, 본 발명은 프로세서 상에서 실행되면 튜닝 가능한 컴포넌트들을 정적으로 특정하고, 튜닝 가능한 컴포넌트들을 동적으로 특정하고, 그리고 상기 튜닝 가능한 컴포넌트의 파라미터들과 상호작용하기 위한 API를 구현하는 실행 가능한 명령어들을 보유하는 하나 이상의 비일시적 컴퓨터-판독 가능한 매체들을 제공하며, 상기 매체들은, 컴포넌트들의 미리 정의된 세트에 대한 파라미터화(parameterization)들을 실현하는 미리 정의된 인터페이스들을 식별하고; 산술 연산들(arithmetic operations)의 세트를 구현하고, 도우미 함수들(helper functions)의 세트를 구현하고, 상기 산술 연산들 및 상기 도우미 함수들을 이용하여 튜닝 가능한 컴포넌트들을 동적으로 생성하고 - 상기 동적으로 생성하는 것은 기초 파라미터 컴포넌트들과 고정된 계수들 또는 고정된 컴포넌트들을 조합하는 것을 포함함 - ; 상기 튜닝 가능한 컴포넌트들의 파라미터적 모델(parametric model)을 생성하기 위한 하나 이상의 실행 가능한 명령어들을 보유하고, 상기 파라미터적 모델은, 상기 튜닝 가능한 컴포넌트들의 튜닝 가능한 파라미터들을 설명하고(accounting for), 그리고 사용자 입력들이 상기 튜닝 가능한 파라미터와 상호작용하도록 허용하고, 상기 사용자 입력들은, 상기 튜닝 가능한 파라미터들을 초기화하고, 상기 튜닝 가능한 파라미터들을 고정(fix)하거나, 또는 상기 튜닝 가능한 파라미터들의 선택된 파라미터를 해제(free)하는 것이다.

[0004] 또한, 본 발명은 프로세서 상에서 실행되면, 임의의 피드백 제어 구조의 표준 형태를 구축하기 위해 그리고 설계 요건들의 H∞ 포뮬레이션에서 이용되는 점 대 점 전달 함수들을 특정하기 위해 애플리케이션 프로그램 인터페이스(application program interface; API)를 구현하는 실행 가능한 명령어들을 보유하는 하나 이상의 비일시적인 컴퓨터-판독 가능한 매체들을 제공하며, 상기 매체들은, 입력 신택스(syntax) - 상기 입력 신택스는 사용자가, 선형 시간 불변 모델(linear time invariant model)들에서 이용되는 선형 시간 불변 모델 컴포넌트를 입력하게 하고; 튜닝 가능한 파라미터들을 갖는 튜닝 가능한 컴포넌트들을 기술하는 소프트웨어 기반 인터페이스들을 입력하게 함 - 를 이용하여 산술 연산자들 및 블록 도식화 연산(block diagramming operations)을 특정하는 사용자 입력 매커니즘과 상호작용하고; 산술 연산자들 및 블록 도식화 연산들 중 하나 이상을 이용하여 상기 소프트웨어 기반 인터페이스들과 상기 선형 시간 불변 모델 컴포넌트를 조합하고 - 상기 블록 도식화 연산들은, 직렬 연결, 병렬 연결, 또는 피드백 연결을 포함함 -; 상기 사용자 입력 매커니즘과 상호작용하기 위한 하나 이상의 실행 가능한 명령어들을 보유하고, 상기 상호작용은 상기 입력 신택스가, 상기 선형 시간 불변 모델 컴포넌트들 및 튜닝 가능한 컴포넌트들을 포함하는 전체 제어 시스템의 표준 형태를 점증적으로(incrementally) 구성하고, 그리고 상기 표준 형태에 기초하여 파라미터적 모델을 생성하도록 허용하고, 상기 파라미터적 모델은 최적화기에 대한 입력을 위해 적응되고, 상기 파라미터적 모델은 설계 요건들을 충족하기 위한 튜닝을 위해 구성되는 것이다.

[0005] 또한, 본 발명은 루프 형상화 요건들을 H∞ 포뮬레이션(formulation) 내로 프로그램적으로(programmatically) 포뮬레이팅하기 위한 컴퓨터-구현 방법을 제공하며, 이 방법은 개루프 이득(open-loop gain)을 위한 타겟 형상 또는 상기 타겟 형상에 대한 프록시(proxy)를 수신하는 단계; 및 소프트웨어 툴과 상호작용하는 단계를 포함하고, 상기 소프트웨어 툴은, 상기 타겟 루프 형상 또는 상기 프록시로부터 제어 구조 및 필터들을 유도하고, 상기 유도된 제어 구조 및 필터들로부터 표준 형태를 구성하고, 상기 유도된 제어 구조 및 필터들로부터 H∞ 제약을 구성하고, 상기 표준 형태 및 상기 H∞ 제약은 제어 시스템에 대한 설계 요건들을 포착하는 것이다.

[0006] 또한, 본 발명은 표준 형태의 구조를 이용하기 위한 컴퓨터-구현 방법을 제공하며, 상기 표준 형태에 대한 구조를 이용하는 것은 튜너 알고리즘(tuner algorithm)의 성능을 강화하고, 상기 방법은, 최적화 프로시저 동안 최적화기에 의해 공급되는 튜닝 가능한 파라미터들의 값들을 수신하는 단계; 구성 프로시저를 이용하여 상기 튜닝 가능한 파라미터 값들에 대한 표준 형태의 상태-공간 모델(state-space model)을 구성하는 단계 - 상기 구성하는 단계는, 상기 튜닝 가능한 파라미터들과 연관된 튜닝 가능한 컴포넌트들을 구현하도록 소프트웨어 객체들을 이용함으로써, 각각의 소프트웨어 객체가 상기 수신된 파라미터 값들에 대한 상기 각각의 소프트웨어 객체의 상태-공간 표현의 제공을 책임지게 함으로써, 상기 튜닝 가능한 컴포넌트들에 대한 애그리게이팅된 상태-공간 매트릭스(aggregated state-space matrix)를 생성하도록 상태-공간 매트릭스들을 애그리게이팅(aggregating)함으로써, 그리고 상기 표준 형태의 원하는 폐루프 상태-공간 모델을 획득하기 위해 럼핑된 플랜트 모델(lumped plant model)의 상태-공간 표현과 상기 튜닝 가능한 컴포넌트들의 애그리게이팅된 상태-공간 매트릭스들을 조합함으로써 행해짐 -; 후속 그래디언트 계산들(gradient computations)을 가속화하기 위해 상기 구성 프로시저의 중간 결과들을 캐싱(caching)하는 단계; 및 그래디언트 정보를 계산하는 단계를 포함하고, 상기 그래디언트 정보를 계산하는 단계는, 상기 목적 및 제약들을 구분함으로써, 상기 표준 형태의 상태-공간 모델을 구성하기 위해 이용된 동일한 소프트웨어 객체들을 이용함으로써, 각각의 소프트웨어 객체가 상기 수신된 파라미터들에 관

하여 스칼라값 함수(scalar-valued function) - 상기 스칼라값 함수는 규칙을 적용한 부산물(by-product)임 - 의 그래디언트의 제공을 책임지게 되게 함으로써; 그리고 상기 목적 및 제약들의 전체 그래디언트들로 각각의 튜닝 가능한 컴포넌트에 의해 공급되는 그래디언트 데이터를 애그리게이팅하도록 상기 캐싱된 중간 결과들을 이용함으로써 행해지는 것이다.

[0007] 본 출원에 포함되고 본 출원의 부분을 구성하는 첨부 도면들은 본 발명의 하나 이상의 실시예들을 예시하며, 그 설명과 함께, 본 발명을 설명할 것이다.

도면의 간단한 설명

[0008] 도 1은 H∞ 합성 기법들을 이용하기 위한 종래의 구성을 예시하는 도면.

도 2는 본 발명의 실시예들 및 기법들을 구현하기 위한 예시적인 시스템을 예시하는 도면.

도 3은 본 발명의 양상들을 구현하는데 이용될 수 있는 모델링 환경의 예시적인 구현을 예시하는 도면.

도 4 내지 도 6은 본 발명의 예시적인 실시예들이 아키텍처들의 튜닝 컴포넌트들에 적용될 수 있는 예시적인 제어기 아키텍처를 예시하는 도면.

도 7은 시스템의 H∞ 합성 표현과 함께 파라미터들을 튜닝하기 위한 구조화된 제어기를 이용하는 본 발명의 예시적인 구현을 예시하는 도면.

도 8은 본 발명의 실시예와 함께 이용되는 피드백 루프에 대한 예시적인 원하는 루프 형상을 예시하는 도면.

도 9는 컴포넌트들을 튜닝하기 위해 예시적인 기법들이 적용될 수 있는 튜닝 가능한 엘리먼트들을 갖는 플랜트 및 제어기를 포함하는 예시적인 구성을 예시하는 도면.

도 10은 본 발명의 실시예를 이용하여 튜닝될 수 있는 다중 입력 다중 출력 제어 문제에 이용하기 위한 예시적인 원하는 루프 형상을 예시하는 도면.

도 11 내지 도 14는 본 발명의 실시예들과 함께 이용될 수 있는 커맨드들, 객체들, 및 출력들을 디스플레이하기 위한 예시적인 인터페이스들을 예시하는 도면.

도 15는 본 발명의 실시예를 이용하여 튜닝될 수 있는 튜닝 가능한 컴포넌트들을 갖는 예시적인 제어기를 예시하는 도면.

도 16a 및 도 16b는 본 발명의 실시예와 함께 이용될 수 있는 커맨드들 및/또는 출력들을 디스플레이하기 위한 예시적인 인터페이스를 예시하는 도면들.

도 17 및 도 18은 본 발명의 실시예를 이용하여 튜닝될 수 있는 튜닝 가능한 컴포넌트들을 포함하는 예시적인 오토파일롯 시스템을 예시하는 도면들.

도 19는 도 17 및 도 18의 오토파일롯 시스템과 관련된 입력들을 수신하고 및/또는 출력들을 디스플레이하는데 이용될 수 있는 예시적인 인터페이스를 예시하는 도면.

도 20은 도 17 및 도 18의 오토파일롯 시스템의 컴포넌트들을 튜닝하는 본 발명의 실시예와 함께 이용될 수 있는 예시적인 타겟 루프 형상을 예시하는 도면.

도 21 및 도 22는 도 17 및 도 18의 오토파일롯 시스템과 관련된 입력들을 수신하고 및/또는 출력들을 디스플레이하기 위한 예시적인 인터페이스들을 예시하는 도면들.

도 23은 도 17 및 도 18의 오토파일롯 시스템과 연관된 예시적인 단계적 응답을 예시하는 도면.

도 24는 도 17 및 도 18의 오토파일롯 시스템과 연관된 감도 함수의 이득을 평가하기 위해 이용될 수 있는 파일롯을 예시하는 도면.

도 25는 본 발명의 실시예들을 실시하기 위해 이용될 수 있는 예시적인 프로세싱을 예시하는 도면.

도 26은 본 발명의 실시예들을 구현하기 위해 이용될 수 있는 예시적인 아키텍처를 예시하는 도면.

도 27은 본 발명의 분산된 실시예를 구현하기 위한 예시적인 시스템을 예시하는 도면.

발명을 실시하기 위한 구체적인 내용

[0009] 다중 입력 다중 출력(MIMO) 제어기들을 설계하기 위한 종래의 접근법들은 H∞ 합성으로서 지칭되는 기법을 포함할 수 있다.

[0010] **종래의 H∞ 합성 기법**

[0011] 도 1은 표준 H∞ 합성 기법의 애플리케이션을 표현하는 구성을 예시한다. 도 1은 플랜트 H(s)(110) 및 제어기 C(s)(120)를 포함할 수 있다. 도 1에서, 표준 H∞ 합성은 입력 w(130)으로부터 출력 z(140)으로의 폐루프 피크 이득을 최소화하는 제어기 C(s)(120)를 컴퓨팅한다. 제어기 C(s)(120)는 블랙 박스(black box)로서 동작할 수 있는 럼핑된(lumped) H∞ 제어기를 표현할 수 있다(예를 들어, 사용자는 블랙 박스 내의 내부 표현들에 대한 준비된 액세스를 갖지 않을 수 있음). H(s)(110)는 제어기 C(s)(120)를 이용하여 제어될 수 있는 플랜트(plant)를 표현할 수 있다. 플랜트 H(s)(110)는 선형 플랜트 모델일 수 있다.

[0012] 종래의 H∞ 합성은 시스템(100)에 대한 폐루프 응답(소위 H∞ 놈(norm))의 피크 입력/출력 이득을 최소화하는데 이용될 수 있다. 사용자는 제어된 플랜트 H(s)에 대한 대역폭, 롤-오프(roll-off), 오버슛(overshoot) 및/또는 안정성 마진들(stability margins)과 같은 설계의 양상들 또는 목적들을 특정하는 제어 문제를 위한 설계 규격을 가질 수 있다. H∞ 합성 기반 프레임워크는 사용자가 규격들을 달성하는데 도움을 주도록 적용 가능하게 될 수 있지만, 몇몇 사용자들(이를테면, 엔지니어들)은 종래의 H∞ 기법들의 이용 시에 편안하고 능숙하다. 예를 들어, 사용자들은, 사용자가 친숙한 보통의 규격들을 종래의 H∞ 합성 기법들에 의해 요구되는 정규화된 폐루프 이득 제약(normalized closed-loop gain constraint)으로 변환하는 것이 지루하고, 비-직관적이며 및/또는 시간-소모적이라는 것을 알 수 있다. 또한, 종래의 H∞ 합성 툴들 및 기법들의 기술적 제약들은 통상적인 사용자들을 추가로 방해하고 및/또는 혼란스럽게 할 수 있다.

[0013] 종래의 H∞ 합성 기법들은 또한 이들이 사용자들의 통상적인 설계 작업 흐름들을 지원하지 않기 때문에 사용자들에게 바람직하지 않을 수 있다. 또한, 종래의 H∞ 합성 기법들은 제어기 C(s)(120)가 표현하는 시스템 구조의 내부 표현들에 사용자가 쉽게 액세스하는 것을 허용하지 않은 블랙 박스로서 제어기 C(s)(120)를 취급한다. 또한, 종래의 H∞ 합성 기법들은 별개의 입력 연결들을 결합하고 정해진 시스템에 대한 C(s)(120)를 생성할 때 이러한 연결들 사이에 제어 블록을 삽입한다. 이 접근법은 C(s)(120)가 유도되었던 시스템 구조를 표현하지 않는 C(s)(120)에 대한 구조를 발생시키며, 이는 통상적인 사용자가 종래의 H∞ 합성 기법으로 작동할 때 직면할 수 있는 어려움들을 추가로 악화시킨다.

[0014] 예를 들어, 사용자들은 실질적으로 실시간 동작을 제공하는 설계 툴들(즉, 툴들은 사용자를 과도하게 짜증나게 하거나 설계 툴과 사용자의 상호작용들을 불리하게 간섭하는 프로세싱 지연들에 직면하지 않고 사용자가 설계의 양상들을 상호작용식으로 설계, 수정 및 실행하는 것을 허용함)을 원할 수 있다. 종래의 H∞ 합성 기법들은 다수의 이유들로 상호작용 동작을 지원하지 않을 수 있다. 예를 들어, 종래의 기법들은 어떠한 특정한 구조 또는 순서 제약도 제어기 C(s)(120)에 가해지지 않을 때 리카티 식들(Riccati equations)의 쌍을 반복적으로 풀음(solving)으로써 최적의 제어기를 컴퓨팅하도록 시도할 수 있다. 식들의 쌍을 풀기 위한 이들 반복되는 시도들은 시간 소모적이고 계산적으로 고가일 수 있으며, 이는 종래의 H∞ 합성 기법들을 특히 상호작용식 설계 애플리케이션들에 대해 바람직하지 않게 하는 경향이 있다.

[0015] **예시적인 기법들 및 실시예들의 개요**

[0016] 예시적인 실시예들은 다중변수 피드백 제어 문제(multivariable feedback control problem)들을 포뮬레이팅(formulate)하고 해결하기 위해 사용자가 직관적인 사용자 인터페이스들을 이용하도록 허용하는 새로운 기법을 제공한다. 예를 들어, 예시적인 실시예들은 제어 시스템의 전체 성능 및 강건성(robustness)을 최적화기 위해 공동으로 튜닝될 필요가 있는 하나 이상의 피드백 루프들에 걸쳐서 분배된 복수의 제어 엘리먼트들을 갖는 제어 문제들을 사용자들이 다루도록 허용한다. 예시적인 실시예들은 확장 가능하며 실질적으로 임의의 수의 피드백 루프들 및/또는 임의의 수의 임의의 정도의 복잡도를 갖는 제어 문제들에 적용될 수 있다.

[0017] 예시적인 실시예들은 도 1과 유사하지만, 사용자가 블랙 박스 대신 화이트 박스(white box)로서 제어기 C(s)를 취급하도록 허용하는 시스템 표현을 이용할 수 있다. 예를 들어, 예시적인 실시예들은 각각의 블록 그 자체가

고정된 구조 및 복잡도를 갖는 블록 대각 구조(block diagonal structure)를 제어기 C(s)가 갖도록 허용한다. C(s)의 부가적인 복잡도는 분석되고 있는 시스템의 제어 아키텍처와 일치하는 방식으로 사용자가 제어기 C(s)를 표현하도록 허용한다. 예시적인 실시예들은 임의의 MIMO 제어 구조들을 자동으로 튜닝하기 위해 비-스무스(non-smooth) H∞ 최적화기들을 이용할 수 있다. 예시적인 기법들은 C(s)가 블록 대각 구조를 포함하는 H∞ 합성 표현의 정규 구조(canonical structure)에 대한 해결자들을 특화한다.

[0018] 예시적인 실시예들 및 기법들은 MIMO 튜닝 작업들을 수행하도록 시도할 때 종래의 기법들에 의해 제시되는 장애물들을 제거한다. 예를 들어, 예시적인 실시예들 및 기법들은 최적화기에 적합한 비용 함수 및 파라미터 벡터로의 제어 아키텍처 및 제어기 구조의 치환을 자동화한다. 예시적인 실시예들 및 기법들은 추가로 그래디언트(gradient)들이 저렴하게 컴퓨팅되도록 허용하고(반복 당 o(N), 여기서 N은 파라미터들의 수), 각각의 튜닝 가능한 블록 타입(이득 PID, 전달 함수, 상태-공간, 원(raw) 파라미터 등)이 그 자신의 기여분(contribution)을 그래디언트에 제공하는 객체-지향 방식으로 그래디언트들이 계산되도록 추가로 허용한다. 예시적인 실시예들은 사용자가 블록-지향 그래디언트들(block-wise gradients)을 컴퓨팅하는 것을 요구함 없이 그래디언트들을 이용하고 종래의 접근법들에서 행해지는 바와 같이 비용 함수의 전체 그래디언트를 계산하기 위해 이들을 함께 조합한다. 종래의 접근법들은 단순한 아키텍처를 제외하면 수동으로 고치기 어려울 수 있으며 본 발명의 양상들에서 이용된 기법들보다 훨씬(orders of magnitude) 덜 효율적일 수 있다.

[0019] 계산 하드웨어에서 실현될 때 본 발명의 예시적인 기법들은 사용자들이 상호작용식으로 MIMO 튜닝 작업들을 수행하도록 허용한다. 예를 들어, 예시적인 실시예들은 표준 개인용 컴퓨터(PC) 및 MATLAB 기술적 컴퓨팅 및 프로그래밍 환경과 같은 기술적 컴퓨팅 환경을 이용하여 1초 내지 30초 미만의 범위에 있는 통상적인 튜닝 시간들을 사용자가 경험하도록 허용한다. 실시예들은 프로세싱 시간들을 감소시키기 위해 다중 코어 또는 다른 타입들의 다중-프로세싱 디바이스들 또는 환경들에서 전개될 수 있다.

[0020] 여기서 기술되는 실시예들은 표현의 용이함을 위해 선형 제어 시스템들과 함께 논의될 것이지만, 본 발명의 실시예들은 비선형 제어 문제들을 해결하는데 이용될 수 있다. 예를 들어, 예시적인 기법들은 이득 스케줄링을 포함(그러나 이것으로 제한되지 않음)해서 비선형 제어 설계에 대한 접근법들을 지원할 수 있다.

[0021] **예시적인 시스템**

[0022] 도 2는 실시예를 실시하기 위한 예시적인 시스템(200)을 예시한다. 시스템(200)은 하나 이상의 엔티티들을 포함하는 모델을 구성하기 위해, 모델에 대한 PID 제어기를 구현하기 위해 및/또는 모델로부터 코드를 생성하기 위해, 예를 들어, 제어기에 대한 코드를 생성하기 위해 이용될 수 있다. 시스템(200)은 컴퓨터(205), 획득 로직(210), 운영 체제(215), 모델링 환경(220), 모델(230), 입력 디바이스(240), 디스플레이 디바이스(250), 모델 표현(260) 및 플랜트(270)를 포함할 수 있다. 도 2의 시스템은 예시적이며 시스템(200)의 다른 실시예들은 더 적은 디바이스들, 더 많은 디바이스들, 및/또는 도 2의 구성과 상이한 구성의 디바이스들을 포함할 수 있다.

[0023] 컴퓨터(205)는 프로세싱 동작들, 디스플레이 동작들, 통신 동작들 등을 수행하는 디바이스를 포함할 수 있다. 예를 들어, 컴퓨터(205)는 사용자 대신 프로세싱 활동들을 수행하고 및/또는 지원하는데 이용될 수 있는, 하나 이상의 프로세싱 또는 저장 디바이스들과 같은 로직을 포함할 수 있다. 컴퓨터(205)의 실시예들은 데스크톱 컴퓨터, 랩톱 컴퓨터, 클라이언트, 서버, 메인프레임, 개인 휴대 정보 단말(PDA), 웹-가능 셀룰러 전화, 스마트폰, 스마트 센서/엑추에이터, 또는 하나 이상의 활동들을 수행하고 및/또는 하나 이상의 결과들을 생성하도록 명령어들을 실행하는 다른 계산 또는 통신 디바이스를 포함할 수 있다.

[0024] 컴퓨터(205)는 추가로 다른 디바이스(도 2에서 도시되지 않음)에 데이터를 송신함으로써 또는 다른 디바이스로부터 데이터를 수신함으로써 통신 동작들을 수행할 수 있다. 데이터는 실질적으로 하나 이상의 네트워크들에서 및/또는 하나 이상의 다른 디바이스들과 함께 이용하도록 적응될 수 있는 임의의 포맷을 갖는 임의의 타입의 머신-판독 가능한 정보를 지칭할 수 있다. 데이터는 디지털 정보 또는 아날로그 정보를 포함할 수 있다. 데이터는 추가로 패킷화 및/또는 비-패킷화될 수 있다.

[0025] 획득 로직(210)은 컴퓨터(205) 외부의 디바이스들로부터 데이터를 획득할 수 있고 데이터가 컴퓨터(205)에 대해 이용 가능하게 할 수 있다. 예를 들어, 획득 로직(210)은 데이터가 컴퓨터(205)에 대해 이용 가능하게 하는데 이용되는 아날로그-디지털 변환기들, 디지털-아날로그 변환기들, 필터들, 멀티플렉서들 등을 포함할 수 있다.

컴퓨터(205)는 모델링 동작들, 제어기 설계 활동들 등을 수행하기 위해 획득된 데이터를 이용할 수 있다.

[0026] 운영 체제(215)는 컴퓨터(205)와 연관된 하드웨어 및/또는 소프트웨어 자원들을 관리할 수 있다. 예를 들어, 운영 체제(215)는 사용자 입력을 수신하고, 컴퓨팅 환경(205)을 동작시키고, 메모리를 할당하고 시스템 요청들을 우선순위화하는 등과 연관된 작업들을 관리할 수 있다. 일 실시예에서, 운영 체제(215)는 가상 운영 체제일 수 있다. 운영 체제(215)의 실시예들은, Linux, Mac OS, Microsoft Windows, Solaris, UNIX 등을 포함할 수 있다. 운영 체제(215)는 추가로 컴퓨터(205)에 의해 제공될 수 있는 가상 머신 상에서 실행될 수 있다.

[0027] 모델링 환경(220)은 사용자가 수학, 과학, 공학, 의학, 상업 등과 같은(그러나 이들로 제한되지 않음) 학문에 관련된 시뮬레이션 또는 모델링 작업들 수행하도록 허용하는 컴퓨팅 환경을 제공할 수 있다. 모델링 환경(220)은 사용자가 실행 가능한 시맨틱들(semantics)을 갖는 모델을 구성하도록 허용하기 위한 명령어들을 실행하는 하나 이상의 애플리케이션들을 지원할 수 있다. 예를 들어, 일 실시예에서, 모델링 환경(220)은 실행 가능한 의미론(semantics)을 갖는 자유-형태 모델들(예를 들어, 제 1의, 제 2의, 제 3의, 제 4의, 제 5의, 기타 등의 차수 모델들)을 사용자가 생성하도록 허용할 수 있다. 모델링 환경(220)은 시간-기반, 이벤트-기반 등의 모델링 활동들을 추가로 지원할 수 있다.

[0028] 모델(230)은 실행 가능한 텍스트 또는 그래픽 모델을 위한 정보를 포함할 수 있다. 예를 들어, 모델(240)은 시간-기반 모델들, 이벤트-기반 모델들, 상태 천이 모델들, 데이터 흐름 모델들, 컴포넌트 다이어그램들, 엔티티 흐름 다이어그램들, 식 기반 언어 다이어그램들 등일 수 있는 텍스트 모델들 또는 그래픽 모델들에 대한 정보를 포함할 수 있다. 모델(230)의 그래픽 실시예들은 동작들을 수행하기 위한 실행 가능한 코드를 표현하는 엔티티들(예를 들어, 블록들, 아이콘들 등)을 포함할 수 있다. 엔티티들에 대한 코드는 모델을 이용한 시뮬레이션을 수행하도록 실행될 수 있다. 하나의 엔티티에서 모델 내의 다른 엔티티로 데이터를 전달하기 위한 통로들을 표현하는 라인들을 이용하여 함께 연결될 수 있다.

[0029] 입력 디바이스(240)는 사용자 입력들을 수신할 수 있다. 예를 들어, 입력 디바이스(240)는 사용자 움직임 또는 동작을 컴퓨터(205)에 의해 해석될 수 있는 신호 또는 메시지로 변환할 수 있다. 입력 디바이스(240)는 키보드들, 포인팅 디바이스들, 바이오메트릭(biometric) 디바이스들, 가속도계, 마이크로폰들, 카메라들, 촉각 디바이스들 등을 포함(그러나 이들로 제한되지 않음)할 수 있다.

[0030] 디스플레이 디바이스(250)는 사용자에게 정보를 디스플레이할 수 있다. 디스플레이 디바이스(250)는 음극선관(CRT), 플라즈마 디스플레이 디바이스, 발광 다이오드(LED) 디스플레이 디바이스, 액정 디스플레이(LCD) 디바이스 등을 포함할 수 있다. 디스플레이 디바이스(250)의 실시예들은 원하는 경우 사용자 입력들을 수신(예를 들어, 터치 감지 스크린을 통해)하도록 구성될 수 있다. 일 실시예에서, 디스플레이 디바이스(250)는 하나 이상의 그래픽 사용자 인터페이스들(GUI들)을 사용자에게 디스플레이할 수 있다. GUI들은 모델(240) 및/또는 다른 타입들의 정보를 포함할 수 있다.

[0031] 모델 표현(260)은 모델(230)의 시각적 표현 및/또는 모델(230)에 의해 제공된 시각적 표현, 예를 들어, 플롯 윈도우(plot window)를 포함할 수 있다. 예를 들어, 모델 표현(260)이 사용자에게 디스플레이될 수 있고 라인들에 의해 연결된 다수의 엔티티들을 포함할 수 있다. 모델(230)이 실행될 때, 모델 표현(260)은 예를 들어, 모델을 통한 데이터의 흐름을 보여주도록 변할 수 있다.

[0032] 플랜트(270)는 데이터를 컴퓨터(205)에 제공하는 하나 이상의 디바이스들을 포함할 수 있다. 예를 들어, 플랜트(270)는 가속도계들, 열전대들(thermocouples), 광-전자(opto-electric) 트랜시버들, 스트레인 게이지(strain gauge)들 등과 같은 센서들을 이용하여 모니터링되는 엔진 시스템을 포함할 수 있다. 일 실시예에서, 획득 로직(210)은 아날로그 또는 디지털 형태로 플랜트(270)로부터 신호들을 수신할 수 있고, 컴퓨터(205)에서 이용하기에 적합한 형태로 신호들을 변환할 수 있다.

[0033] **예시적인 모델링 환경**

[0034] 도 3은 모델링 환경(220)의 예시적인 실시예를 예시한다. 모델링 환경(220)은 시뮬레이션 툴(310), 엔티티 라이브러리(320), 인터페이스 로직(330), 컴파일러(340), 제어기 로직(350), 최적화기(360), 시뮬레이션 엔진(370), 리포트 엔진(380) 및 코드 생성기(390)를 포함할 수 있다. 도 3에서 예시된 모델링 환경(220)의 실시예는 예시적이며, 모델링 환경(220)의 다른 실시예들이 본 발명의 사상으로부터 벗어남 벗이 더 많은 엔티티들 또

는 더 적은 엔티티들을 포함할 수 있다.

[0035] 시뮬레이션 툴(310)은 모델을 구축하기 위한 애플리케이션일 수 있다. 시뮬레이션 툴(310)은 실행 가능한 의미론을 갖는 텍스트 모델 또는 그래픽 모델을 구축하는데 이용될 수 있다. 그래픽 모델들의 경우에서, 시뮬레이션 툴(310)은 사용자들이 모델 엔티티들 및/또는 연결들을 생성, 수정, 진단, 삭제 등을 하도록 허용할 수 있다. 시뮬레이션 툴(310)은 사용자 입력들을 수신하고, 모델을 실행하고, 결과들을 디스플레이하고 코드를 생성하는 등을 위해 도 2 또는 도 3에서 예시된 다른 엔티티들과 상호작용할 수 있다. 시뮬레이션 툴(310)은 텍스트 모델들을 구성하거나 상호작용하기 위한 편집 윈도우 및/또는 그래픽 모델들을 생성하거나 상호작용하기 위한 GUI를 사용자에게 제공할 수 있다.

[0036] 엔티티 라이브러리(320)는 모델 표현(360)을 포함하는 디스플레이 윈도우를 사용자가 드래그 및 드롭할 수 있는 코드 모듈들 또는 엔티티들(예를 들어, 블록들/아이콘들)을 포함할 수 있다. 그래픽 모델들의 경우에, 사용자는 플랜트(370)와 같이 시스템의 그래픽 모델을 생성하기 위해 연결들을 이용하여 엔티티들을 추가로 결합할 수 있다.

[0037] 인터페이스 로직(330)은 모델링 환경(220)이 디바이스들(예를 들어, 플랜트(270), 타겟 환경 등) 또는 소프트웨어 모듈들(예를 들어, 함수, 애플리케이션 프로그램 인터페이스 등)에/로부터 데이터 및/또는 정보를 송신 또는 수신하도록 허용할 수 있다. 일 실시예에서, 인터페이스 로직(330)은 획득 로직(310)을 모델링 환경(220)과 인터페이스시킬 수 있다.

[0038] 컴파일러(340)는 모델을 실행 가능한 포맷으로 컴파일할 수 있다. 컴파일러(340)에 의해 생성된 컴파일된 코드는 모델링 결과를 생성하기 위해 컴퓨터(205) 상에서 실행될 수 있다. 일 실시예에서, 컴파일러(340)는 또한 모델과 연관된 에러들을 진단하기 위한 디버깅 성능들을 제공할 수 있다.

[0039] 제어기 로직(350)은 모델(330)에서 제어기들을 생성하고 구현하는데 이용될 수 있다. 예를 들어, 제어기 로직(350)은 모델 표현(260)에서 제어기들의 타입들을 표현하는 엔티티들에 대한 기능을 제공할 수 있다. 모델을 실행할 때, 제어기 로직(350)은 모델 표현(260)의 엔티티들과 상호작용함으로써 모델 상에서 제어 동작들을 수행할 수 있다. 일 실시예에서, 제어기 로직(350)은 모델 표현(360)에서 제어기들을 구현하는 제어 알고리즘들을 포함할 수 있다. 제어기 로직(350)의 실시예들은 자립형 또는 분산된 구현들로 동작하도록 구성될 수 있다.

[0040] 최적화기(360)는 모델에 대한 코드, 파라미터들, 성능(예를 들어, 실행 속도) 등을 최적화할 수 있다. 예를 들어, 최적화기(360)는 코드가 최적화되지 않은 경우 실행되는 코드보다 코드가 더 적은 메모리를 점유하고, 코드가 보다 효율적으로 실행하고, 코드가 더 빨리 실행하게 하도록 코드를 최적화할 수 있다. 최적화기(360)는 예를 들어, 제어기에 대한 파라미터들을 최적화하기 위해 제어기 로직(350)에 대한 최적화들을 또한 수행할 수 있다. 일 실시예에서, 최적화기(360)는 컴파일러(340), 제어기 로직(350), 코드 생성기(390) 등과 함께 동작할 수 있거나 이들 내로 통합될 수 있다. 최적화기(360)의 실시예들은 예를 들어, 최적화기(360)가 동작하는 데이터를 수신하기 위해 다른 객체 지향 소프트웨어와 상호작용하는 소프트웨어 객체들을 통해 구현될 수 있다.

[0041] 시뮬레이션 엔진(370)은 시스템을 시뮬레이션하기 위해 모델을 실행하기 위한 동작들을 수행할 수 있다. 시뮬레이션 엔진(370)은 사용자 선호도들 또는 시스템 선호도들에 기초하여 자립 또는 원격 시뮬레이션들을 수행하도록 구성될 수 있다.

[0042] 리포트 엔진(380)은 모델링 환경(220)의 정보에 기초하여 리포트를 생성할 수 있다. 예를 들어, 리포트 엔진(380)은 제어기가 설계 규격들을 만족하는지를 표시하는 리포트, 제어기가 적절한 방식으로 동작하는지를 표시하는 리포트, 모델이 적절히 컴파일하는지를 표시하는 리포트 등을 생성할 수 있다. 리포트 엔진(380)의 실시예들은 디스플레이 디바이스(250) 상에 디스플레이를 위한 전자 포맷, 하드카피(hardcopy) 포맷, 및/또는 저장 디바이스에 저장을 위해 적응된 포맷의 리포트들을 생성할 수 있다.

[0043] 코드 생성기(390)는 모델로부터 코드를 생성할 수 있다. 일 실시예에서, 코드 생성기(390)는 제 1 포맷의 코드를 수신할 수 있고, 제 1 포맷으로부터 제 2 포맷으로 코드를 변환할 수 있다. 일 실시예에서, 코드 생성기(390)는 모델의 적어도 일부로부터 소스 코드, 어셈블리 언어 코드, 이진 코드, 인터페이스 정보, 구성 정보, 성능 정보, 작업 정보 등을 생성할 수 있다. 예를 들어, 코드 생성기(390)는 모델로부터 C, C++, SystemC, Java, 구조화된 텍스트(Structured Text)등의 코드를 생성할 수 있다.

[0044] 코드 생성기(390)의 실시예들은 그래픽 모델(예를 들어, 시스템 모델링 언어(System Modeling Language; SysML), 확장 가능한 마크업 언어(Extensible Markup Language; XML), 실시간 및 임베딩된 시스템들의 모델링

및 분석(Modeling and Analysis of Real Time and Embedded Systems; MARTE), 하드웨어 기술 언어(Hardware Description Language; HDL), 자동 오픈 시스템 아키텍처(Automotive Open System Architecture; AUTOSAR) 등) 중 일부 또는 모두 다로부터 단일화된 모델링 언어(Unified Modeling Language; UML) 기반 표현들 및/또는 확장들을 추가로 생성할 수 있다. 일 실시예에서, 최적화기(370)는 파라미터(예를 들어, 메모리 이용, 실행 속도, 다중-프로세싱 등)에 따라 최적화되는 코드를 생성하도록 코드 생성기(390)와 상호작용할 수 있다. 본 발명의 원리들과 일치하는 모델링 환경들의 실시예들은 검증 컴포넌트, 확인 컴포넌트들 등과 같은 컴포넌트들을 추가로 포함할 수 있다.

[0045] 본 발명의 실시예들은 다중변수 피드백 제어 문제들을 포뮬레이팅하고 해결하기 위해 그리고 실질적으로 임의의 순서 및/또는 지연의 비-선형 모델에서 이용하기 위한 제어기들을 설계하기 위해 이용될 수 있다. 실시예들은 비-선형 모델의 적어도 일부를 표현할 수 있는 선형 시간 불변 모델들(linear time invariant models)을 생성하기 위해 정확한 선형화 기법들을 이용하도록 구성될 수 있다.

[0046] **예시적인 제어 아키텍처들**

[0047] 예로서, 본 발명의 실시예들은 다른 것들 중에서도, 실질적으로 임의의 순서로 배열되는 제어기 블록들을 포함할 수 있는 복수의 컴포넌트들 및 하나 이상의 피드백 루프들을 갖는 제어 아키텍처들에 적용될 수 있다.

[0048] 도 4는 본 발명의 실시예들이 적용될 수 있는 예시적인 제어 아키텍처, 즉 높은 받음각 모드(high angle of attack mode)로 F-14에서 이용된 오토파일롯을 예시한다. 오토파일롯은 GUI(400)에서 디스플레이될 수 있고, 이득들 및 시상수(time constant)들을 포함하는 8개의 튜닝 가능한 파라미터들을 포함할 수 있다.

[0049] 도 5는 증류탑(distillation column)에서 이용되는 제어기를 위한 예시적인 아키텍처를 예시한다. 도 5의 제어기는 GUI(500)를 통해 사용자에게 디스플레이될 수 있고, 튜닝을 요구하는 2x2 이득 매트릭스 및 4개의 비례하는 통합(PI) 이득들을 포함할 수 있다.

[0050] 도 6은 풍력 터빈(wind turbine)을 위한 피치 및 요(yaw) 제어를 위한 예시적인 아키텍처를 예시한다. 도 6의 아키텍처는 GUI(600)를 통해 사용자에게 디스플레이될 수 있고 튜닝을 요구하는 2개의 이득들 및 3개의 PI 제어기들을 포함할 수 있다. 도 6에서 예시되는 바와 같이, 본 발명의 실시예들과 함께 이용되는 GUI들은 윈도우들, 패인들(panes)과 같은 복수의 인터페이스들을 포함할 수 있다.

[0051] 본 발명의 예시적인 실시예들은 또 다른 타입들의 아키텍처들과 그리고 도 4 내지 6에서 및 본 즉시 출원의 다른 곳에서 예시되는 아키텍처들보다 더 복잡하거나 덜 복잡한 아키텍처들과 함께 이용될 수 있다.

[0052] 여기서 기재되는 예시적인 실시예들 및/또는 기법들은 구조들이 단일의 일반적인 표현들로 감소되도록 허용함으로써 임의의 제어 구조들의 효율적인 튜닝을 허용한다.

[0053] **예시적인 정규 구조**

[0054] 도 7은 변환된 임의의 제어 구조들을 표현하는데 이용될 수 있는 예시적인 정규 구조를 예시한다.

[0055] 도 7을 참조하면, 시스템(700)은 인터페이스를 통해 사용자에게 디스플레이될 수 있고, 제어 시스템의 고정된 컴포넌트들을 단일의 럼핑된(lumped) 모델 내로 조합하는 선형 모델일 수 있는 H(s)(710)를 포함할 수 있다. 본 발명의 일 실시예에서, H(s)(710)는 시스템의 모든 고정된 제어 컴포넌트들을 표현할 수 있다. 시스템(700)은 추가로 튜닝될 엘리먼트들을 포함할 수 있는 제어기(720)를 포함할 수 있다. 예를 들어, 예시적인 실시예에서, 제어기(720)는 튜닝될 제어 엘리먼트들을 표현하는 하나 이상의 블록들(B_1(730) 내지 B_N(740))을 포함하는 구조화된 제어기일 수 있다. 예로서, 제어기(720)가 단일 블록을 포함할 때, 그 블록은 B_1으로서 지칭되고, 3개의 블록들을 갖는 제어기(720)는 블록들(B_1, B_2 및 B_3)을 포함할 것이다. 튜닝될 제어 엘리먼트들은 설계 시에 변경될 수 있다. 예를 들어, 튜닝될 엘리먼트들은 이득, 동적인 엘리먼트들(예를 들어, 전달 함수들, 상태-공간 모델들 등) 및/또는 플랜트 또는 제어기의 설계 파라미터들을 포함할 수 있다.

[0056] 제어기(720)의 예시적인 실시예들은 블록들(B_1 내지 B_N)의 블록-대각 애그리게이트(block-diagonal aggregate)를 포함할 수 있고 구조화된 제어기(720)로서 지칭될 수 있다. 구조화된 제어기들(720)은 반복되는 블록들을 포함할 수 있다(즉, B_2와 같은 특정한 블록은 대각을 따라 여러 번 나타날 수 있음).

[0057] 시스템(700)은 다음과 같이 주파수 도메인에서 대응하는 식들을 이용하여 표현될 수 있다:

수학식 1

$$\begin{cases} \begin{bmatrix} z \\ y \end{bmatrix} = H(s) \begin{bmatrix} w \\ u \end{bmatrix} \\ u = \begin{bmatrix} B_1(s) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & B_N(s) \end{bmatrix} y \end{cases}$$

[0058]

[0059] 식 1에서, H(s) 및 튜닝된 블록들(B_1 내지 B_N)은 신호들(u 및 y을 통해 피드백 방식으로 상호작용한다. 식 1의 정규 구조는 3개의 특성들을 갖는다:

[0060] (1) B_1, ..., B_N이 블록으로서 나타나는 신호-흐름 다이어그램들에서 블록들(B_1, ..., B_N)을 잔여 다이어그램으로부터 분리하고 블록-대각 구조를 생성하기 위해 그들의 입력들 및 출력들을 붙임으로써 변환될 수 있다.

수학식 2

$$C(s) = \begin{bmatrix} B_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & B_N \end{bmatrix}$$

[0061]

[0062] (2) 그의 계수들이 몇몇 파라미터들(p_1, ..., p_N)의 유리 함수인 전달 함수 또는 상태-공간은 블록들(B_1, ..., B_N)이 그 형태를 이루는 정규 형태(canonical form)로 변환될 수 있다.

[0063]

수학식 3

$$B_j = \begin{bmatrix} p_j & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & p_j \end{bmatrix}$$

[0064]

[0065] (3) 이 정규 표현은 직렬, 병렬, 및 피드백 상호연결 하에서 마무리되며, 이는 정규 구조의 임의의 상호연결이 더 큰 블록 리스트를 갖는 다른 정규 구조를 산출한다는 것을 의미한다.

[0066] 위의 특성들 (1) 내지 (3)은 본 발명의 원리들과 일치하는 도 7의 정규 구조를 튜닝하기 위한 임의의 방법이 임의의 순서 및 구조의 선형 제어 엘리먼트들을 갖는 가상의 임의의 제어 구조에 응용 가능하다는 것을 보장한다.

[0067] **튜닝 가능한 블록들을 파라미터화하기 위한 예시적인 기법**

[0068] 예시적인 실시예들은 최적화 문제로서 튜닝 작업을 포뮬레이팅하기 위해 시스템의 튜닝 가능한 엘리먼트들을 파라미터화하도록 구성될 수 있다. 예를 들어, 일 실시예에서, 시스템의 모든 튜닝 가능한 엘리먼트들이 파라미터화될 수 있다. 표 1은 비례 적분 미분(proportional integral derivative; PID) 블록, 고정-순서 전달 함수 블록, 및 고정-순서 상태-공간 블록과 같이 몇몇 공통 제어 엘리먼트들의 파라미터화를 요약한다.

표 1

[0069]

튜닝 가능한 제어 엘리먼트	파라미터화
이득 블록	이득 매트릭스의 각각의 엔트리는 파라미터임
PID 블록 $B(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{1 + T_f s}$	Kp, Ki, Kd 및 Tf는 파라미터들임
고정된-차수 전달 함수 블록 $B(s) = \frac{b_m s^m + \cdots + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_0}$	분자 및 분모 계수들은 파라미터들임
고정된-차수 채널-공간 블록 $B(s)\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$	A,B,C,D 매트릭스들의 엔트리들은 파라미터들임, 디폴트에 의해, 우리는 A는 3-대각(tri-diagonal)으로 제한하며, 이는 완전히 일반적이며 필요한 파라미터들의 수를 감소시킴

[0070] 선택된 제어 엘리먼트들의 파라미터화

[0071] 예시적인 실시예들은 기본 파라미터들을 수반하는 표현들을 작성함으로써 제어 엘리먼트들의 커스텀 파라미터화들(custom parameterizations)을 사용자가 생성하도록 허용하기 위해 소프트웨어-기반 툴들을 제공 및/또는 이용하도록 추가로 구성될 수 있다. 예를 들어, 본 발명의 실시예에서 이용된 예시적인 소프트웨어 툴은 다음과 같은(그러나 이들로 제한되지 않음) 부가적인 구조들을 갖는 튜닝 가능한 블록들을 사용자가 생성하도록 허용할 수 있다:

[0072] 실수 값 a에 의해 파라미터화된 저역-통과 필터 $a/(s+a)$

[0073] 관찰기 형태(observer form)의 상태-공간 제어기 :

수학식 4

$$\begin{cases} \dot{x} = Ax + Bu + L(y - Cx - Dy) \\ u = -Kx \end{cases}$$

[0074]

[0075] 수학식 4에서, 파라미터들은 이득 매트릭스들(K 및 L)이다.

[0076] **다목적 지원을 위한 예시적인 기법**

[0077] 예로서, 2개의 H∞ 제약들이 제공될 수 있는데, 즉

[0078] $$\|T_1\|_\infty < 1, \quad \|T_2\|_\infty < 1$$

[0079] 이며, 이는 2개의 별개의 폐루프 전달 함수를 수반한다.

수학식 5

$$\begin{cases} T_1 = F(H_1, C) \\ T_2 = F(H_2, C) \end{cases}$$

[0080]

[0081] 여기서 F(.,.)는 도 7에서 도시된 선형 분수 변환(linear fractional transformation; LFT) 상호연결을 나타내고,

[0082] 수학식 2

$$C(s) = \begin{bmatrix} B_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & B_N \end{bmatrix}$$

[0083]

[0084] 은 튜닝될 구조화된 보상자(structured compensator)를 나타낸다.

[0085] 2개의 H∞ 제약들($\|T_1\|_\infty < 1, \quad \|T_2\|_\infty < 1$)은 애그리게이트 전달 함수

수학식 6

[0086] $$T(s) = F(\begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix}, \begin{bmatrix} C & 0 \\ 0 & C \end{bmatrix})$$

[0087] $$\|T\|_\infty < 1$$

[0088] 내로 조합될 수 있다.

[0089] C(s)가 전체-차수 다중 입력 다중 출력(full-order multiple input multiple output; MIMO) 제어기일 때, 종래의 H∞ 합성 기법들은 결과적인 제어기의 $\begin{bmatrix} C & 0 \\ 0 & C \end{bmatrix}$ 블록-대각 구조로 인한 결과적인 문제를 해결할 수 없다.

[0090] 대조적으로, 본 발명의 예시적인 실시예들은 블록 반복은 물론 $\begin{bmatrix} C & 0 \\ 0 & C \end{bmatrix}$ 블록-대각 구조를 지원한다. 그러므로 본 발명의 예시적인 실시예들은 다목적 문제와 같이 종래의 H∞ 합성 기법이 실패한 문제들을 해결하기 위해 이용될 수 있다:

[0091] $\|T_1\|_\infty < 1$ 및 $\|T_2\|_\infty < 1$ 가 되도록 $C(s)$ 를 튜닝하거나, 또는 동등하게는, $\|T\|_\infty < 1$가 되도록 $C(s)$ 를 튜닝하는 것은 단목적 문제를 해결하는 것과 차이가 없다. 실제적인 관점에서, 예시적인 실시예들은 사용자가 각각의 루프 전달 함수를 독립적으로 제한하도록 허용하여서 사용자는 모든 요건들을 포착하는 단일의 MIMO 제약에 따라야(come up with) 하는 것은 아니다. 이는 다수의 사용자들에게 바람직한 특징인데, 그 이유는 모든 요건들을 포착하는 단일 MIMO 제약에 따르는 것은 통상적으로 사용자가 문제들 제시하는 방식과 간섭하는 교차 성분(cross-term)들로 인해 사용자들에 대해 도전적인 작업이기 때문이다.

[0092] **루프 형상화 요건들을 자동으로 포뮬레이팅하기 위한 예시적인 기법**

[0093] 루프 형상화(Loop shaping)는 피드백 제어 시스템들을 튜닝하기 위한 인기있는 주파수-도메인 기법이다. 예로서, 단일 입력 단일 출력(SISO) 피드백 루프에 대해, 루프 형상화 프로시저는 다음으로 구성될 수 있다:

[0094] 1. 개방-루프 응답 L(s)에 대한 원하는 이득 프로파일의 견지에서 (주파수의 함수로서 이득) 설계 요건들을 표현. 예를 들어, 이득은 양호한 트래킹 및 외란 제거(disturbance rejection)를 위해 저 주파수에서 하이(>1)이어야 하고, 이득은 노이즈 및 모델링 에러에 대한 둔감성(insensitivity)을 위해 높은 주파수에서 로우(<1)이어야 한다. 천이 주파수(이득 = 1)는 크로스오버 주파수(ω_c)라 불리며 제어 시스템의 응답의 속도에 직접 영향을 준다.

[0095] 2. 폐루프 안정성 및 충분한 안정성 마진들을 유지하면서 원하는 루프 형상에 접근하도록 보상자의 튜닝 가능한 파라미터들을 조정.

[0096] 다중-루프 제어 시스템들에 있어서, 이러한 프로시저는 아래에서 논의되는 바와 같은 몇몇 조정들을 통해 MIMO 개루프 응답에 적용될 수 있다.

[0097] 바로 위 단계(1)는 보통 주파수-도메인 기법들과 친숙한 제어 엔지니어들에 의해 잘 이해된다. 적분 동작을 갖는 피드백 루프에 대한 원하는 루프 형상은 도 8에서 예시된 플롯(800)에 의해 표현된다. 일 실시예에서, 플롯(800)은 디스플레이 디바이스(250)를 이용하여 GUI를 통해 사용자에게 디스플레이될 수 있다.

[0098] 도 8을 참조하면, 0dB 크로스오버 주파수는 ω_c =10(ω_c 는 도 8에서 805라 불림)이며, 이는 약 1/10 = 0.1 초의 응답 시간에 대응한다. 도 8에서, 응답 시간은 피드백 루프가 변화들에 얼마나 빨리 반응하는지를 지칭한다. ω_c 의 증가 또는 감소는 각각 응답을 빠르게 하거나 느리게 할 수 있다. 도 8의 루프 형상은 w < 10(영역(810))에 대해 높은 이득 및 w > 10(영역(820))에 대해 낮은 이득을 갖는다.

[0099] 예시적인 실시예들 및 기법들은 원하는 루프 형상에 또는 그 근처에 도달하도록 시스템에서 보상기들의 튜닝 가능한 파라미터들을 조정하는데 적합하다. 본 발명의 실시예들과 함께 이용되는 기법들은 H∞ 놈 제약으로 루프 형상화 목적(즉, 특정한 프로파일에 정합하도록 개루프 이득을 형상화)를 포뮬레이팅하는데 유리할 수 있다. H∞ 놈 제약 내에서 루프 형상화 목적를 포뮬레이팅하기 위한 몇 개의 알려진 기법들이 존재하지만, 이러한 치환(translation)은 종종 강건한 제어 이론 및 H∞ 합성에 훈련되지 않은 엔지니어들을 곤란하게 한다. 예시적인 실시예들은 종래의 접근법들을 이용할 때 엔지니어들(사용자들)이 직면하는 어려움들을 제거하도록 리포뮬레이션 단계(reformulation step)를 단순화하는 기법을 이용한다. 예를 들어, 실시예들은 사용자로부터의 부담(burden)들을 컴퓨팅하고 및/또는 구성을 제거하도록 리포뮬레이션 단계를 자동화할 수 있다.

[0100] 도 9는 본 발명의 양상들을 구현하기 위해 이용될 수 있는 시스템(900)을 예시한다. 시스템(900)은 입력 신호(r), 출력 신호(y), 제어기 C(s)(910), 및 플랜트 G(s)(920)와 함께 SISO 및 MIMO 피드백 루프를 포함할 수 있다. 시스템(900)은 신호들(u 및 n)을 추가로 포함한다. 도 9에서, C(s)(910)는 하나 이상의 튜닝 가능한 블록들을 포함할 수 있고 신호들(u 및 y)은 동일한 수의 엔트리들(즉, 동일한 수의 제어들 및 측정들)을 포함하는 것으로 가정될 수 있다.

[0101] 도 9에서, 개루프 전달 함수는 다음과 같이 표현될 수 있고:

수학식 7

[0102] $$L = GC$$

[0103] 폐루프 전달 함수는 다음과 같고:

수학식 8

[0104] $$\begin{bmatrix} y \\ e \end{bmatrix} = \begin{bmatrix} T & -T & GS \\ S & -S & GS \end{bmatrix} \begin{bmatrix} r \\ n \\ d \end{bmatrix}$$

[0105] 표기로는,

수학식 9

$$S = (I + L)^{-1}, T = LS$$

[0106]

[0107] 타겟 루프 형상 p(s) 및 외란 제거 팩터 β가 주어지면, 다음을 고려한다:

수학식 10

$$X(s) = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix} \begin{bmatrix} T & -T & GS \\ S & -S & GS \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/\rho & 0 \\ 0 & 0 & 1/\beta \end{bmatrix}$$

[0108]

[0109] H∞ 제약

[0110] $\|X\|_\infty < 1$ 는 다음을 보장한다는 것이 검증될 수 있다:

수학식 11

$$\begin{cases} \|S(j\omega)\| < \min\left(1, \frac{1}{\rho(\omega)}\right) \\ \|T(j\omega)\| < \min\left(1, \rho(\omega)\right) \\ \|GS(j\omega)\| < \beta \min\left(1, \frac{1}{\rho(\omega)}\right) \end{cases}$$

[0111]

[0112] . $S \approx 1/L$ 및 $T \approx 1$ 이며, 여기서 루프 이득 $\|L(j\omega)\|$ 은 1보다 크고

[0113] . $T \approx L$ 및 $S \approx 1$ 이며, 여기서 루프 이득은 1보다 작기 때문에, 이는

[0114] 다음과 등가이다:

수학식 12

$$\begin{cases} \|L(j\omega)\| > \rho(\omega) \text{ 여기서 } \|L(j\omega)\| > 1 \\ \|L(j\omega)\| < \rho(\omega) \text{ 여기서 } \|L(j\omega)\| < 1 \\ \|S(j\omega)\| < 1 \\ \|T(j\omega)\| < 1 \\ \|GS(j\omega)\| < \beta \min\left(1, \frac{1}{\rho(\omega)}\right) \end{cases}$$

[0115]

[0116] 수학식 12의 최초의 2개의 제약들은 루프 이득을 크로스오버 주파수로부터 충분히 크게 또는 아주 적게 멀어지게 함으로써 원하는 루프 형상을 강제한다. 예를 들어, 크로스오버 주파수로부터 멀리 있는 상위 값 및 하위 값과 같은 루프 이득에 대한 원하는 값들을 포함할 수 있다. 수학식 12의 제 3 및 제 4 제약들은 S 및 T의 이득을 크로스오버에 가까운 1에 근접하게 유지하도록 동작할 수 있으며, 이는 충분한 오버슛(overshoot) 및 안정성 마진들을 보장하는데 중요하다. 마지막으로, 도 12의 마지막 제약은 사용자들이 β를 감소시킴으로써 외란 제거 특성들을 개선하게 한다.

[0117] 도 9에 관한 위의 논의를 요약하면, H∞ 제약

[0118] $$\|X\|_\infty < 1$$

[0119] 가 원하는 루프 형상을 임프레싱(impress)하고, 충분한 마진들을 제공하고 β를 통해 외란 제거를 최적화한다는 것이 예시되었다. 이는 루프 형상화 목적들(loop shaping objectives)을 본 발명의 예시적인 기법들과 함께 이용하기에 적합한 H∞ 제약으로 치환하기 위한 체계적인 방식을 제공한다. 예를 들어, 예시적인 실시예들 및 기법들은 다음과 같이 사용자들로부터의 이어지는 기본 정보만을 이용하여 동작할 수 있는 소프트웨어 툴들의 생성을 허용한다:

[0120] · 타겟 루프 형상 p(s)(일 실시예에서 이득만이 중요할 수 있다)

[0121] · 튜닝 가능한 블록들

[0122] · 폐루프 시스템의 블록도에서 신호들(r, d, y)의 위치

[0123] 이 정보로부터, 본 발명의 양상들을 구현하는 소프트웨어 툴들은 자동으로 G 및 C를 추출하고, X(s)를 구성하고 본 발명의 양상들과 일치하는 구조화된 H∞ 합성 기법들을 이용하여 블록 파라미터들을 튜닝할 수 있다. 이들 소프트웨어 툴들 및 기법들은 원하는 크로스오버 주파수(ω_c)로부터 충분한 루프 형상을 자동으로 생성함으로써 그리고 적분 동작이 필요한지 여부를 결정함으로써 타겟 루프 형상 p(s)에 대한 요구를 추가로 제거할 수 있다.

[0124] 본 발명의 소프트웨어 구현 양상들이 MIMO 문제들에 또한 적용될 수 있다. 예를 들어, 예시적인 기법들은 각각이 특정한 접합(joint)을 제어하는 4개의 별개의 피드백 루프들을 갖는 로봇 암(robot arm)의 제어에 적용될 수 있다. MIMO 경우에서, 정제(refinement)는 SISO 문제들에 이용되는 루프 형상과 상이할 수 있는 루프 형상을 이용하는 것으로 구성될 있다.

[0125] 도 10은 본 발명의 양상들에 따라 MIMO 문제들을 해결하는데 이용될 수 있는 루프 형상을 예시한다. 도 10에서, 트래이스(1010)는 대략 0dB 정도의 이득을 갖는 평탄부(1015)를 포함한다. 평탄부(1015)는 크로스오버 주파수들의 범위를 허용한다. 크로스오버 주파수의 범위의 허용은 결합된 다중-루프 제어 시스템들에서, 모든 루프들이 동일한 주파수에서 0dB를 교차하는 것이 가능하지 않을 수 있다는 사실을 설명하는데 도움을 줄 수 있다.

[0126] **본 발명의 양상들을 구현하기 위한 예시적인 소프트웨어-기반 툴들**

[0127] 본 발명의 예시적인 실시예들은 여기서 기술된 기법들을 실시하기 위해 소프트웨어로 구현될 수 있다. 본 발명의 소프트웨어 구현 실시예들은 통상적인 사용자들에게 쉽게 이해되는 용어들을 이용하고 사용자들이 포맷들에 정보를 입력하도록 허용하는 그래픽 사용자 인터페이스들(GUI들)과 같은 인터페이스들을 사용자에게 제공한다. 소프트웨어 구현 실시예들은 제어 시스템들의 빠른 설계 및/또는 분석을 지원하는 알고리즘들 및 프로세싱 기법들을 추가로 이용한다. 사실상, 본 발명의 실시예들은 제어 시스템들의 효율적인 설계 및 분석을 장려하는 작업 흐름들을 사용자가 활용하도록 허용하는 상호작용식 제어 설계 및 분석 애플리케이션들을 지원하도록 구성될 수 있다.

[0128] 예를 들어, 예시적인 실시예들이 여기서 기술된 기법들을 구현하기 위해 하나 이상의 소프트웨어 툴들을 이용할 때, 소프트웨어 툴들은 다음과 같은(그러나 이들로 제한되지 않음) 서비스들을 사용자에게 제공할 수 있다:

[0129] 1. 제어 엘리먼트들의 구조를 특정하고 엘리먼트들의 파라미터화(parameterization)를 자동으로 생성;

[0130] 2. 커스텀 엘리먼트들 및 파라미터화들을 생성;

[0131] 3. 도 7의 대응하는 정규 구조에 정해진 제어 아키텍처를 맵핑;

[0132] 4. 단순 설계 요건들을 H∞ 제약들로 치환;

[0133] 5. 근본적인 구조화된 H∞ 합성 문제를 해결함으로써 제어 엘리먼트들을 자동으로 튜닝;

[0134] 6. 원 제어 엘리먼트의 견지에서 결과들을 제시; 및

[0135] 7. 보통의 선형 분석 툴들을 이용한 빈틈없는 통합(tight integration)을 통해 설계의 확인을 용이하게 함

[0136] 제시의 명확성을 위해, 예시적인 실시예들 및 기법들은 MathWorks의 제어 시스템 툴박스와 같은 MATLAB 언어 호환 가능한 툴박스들 및/또는 MATLAB 프로그래밍 언어와 관련하여 기술된다. 본 발명의 실시예들 및/또는 실시예들과 연관된 기법들은 본 발명의 사상으로부터 벗어남 없이 다른 소프트웨어 패키지들 및/또는 애플리케이션들 등을 이용하여 다른 프로그래밍 환경들에서 구현될 수 있다. 예를 들어, 실시예들 및/또는 기법들은 MATLAB-호환 가능한 제품들/언어들, Octave; Python; Comsol Script; National Instruments로부터의 MATRIXx; Wolfram Research, Inc.로부터의 Mathematica; Mathsoft Engineering & Education Inc.로부터의 Mathcad; Maplesoft로부터의 Maple; Imagine That Inc.로부터의 Extend ; The French Institution for Research in Computer Science and Control(INRIA)로부터의 Scilab; Cadence로부터의 Virtuoso; Dynasim로부터의 Modelica 또는 Dymola; C/C++ 라이브러리들 등을 포함(그러나 이들로 제한되지 않음)하는 애플리케이션들을 이용하여 구현될 수 있다.

[0137] 몇몇 실시예들에서, 본 발명의 기법들을 구현하기 위한 환경은 C++, C, 포트란, 파스칼 등과 같은 종래의 프로그래밍 언어로 코드를 개발하도록 사용자에게 요구하는 환경과 같은 다른 타입의 컴퓨팅 환경에서 작업들이 수행되는 경우보다 더욱 효율적으로 수학, 과학, 공학, 의학, 상업 등과 같은 학문들에 관련된 작업들을 사용자들이 수행하도록 허용하는 컴퓨팅 환경을 제공하는 하드웨어 또는 하드웨어/소프트웨어 기반 로직을 포함할 수 있다.

[0138] 일 구현에서, 환경은 관련 기술의 당업자들에게 친숙한 수학적 표기들로 문제들 및/또는 해결책들을 표현하는데 이용될 수 있는 동적으로 타이핑된(typed) 언어를 포함할 수 있다. 예를 들어, 환경은 어레이가 디멘션닝(dimensioning)을 요구하지 않을 수 있는 기본 엘리먼트로서 어레이를 이용할 수 있다. 이들 어레이들은 연산들이 어레이의 값들과 같은 값들의 전체 세트에 적용될 수 있다는 점에서 어레이 프로그래밍을 지원하는데 이용될 수 있다. 어레이 프로그래밍은 프로그래머가 사고하게 하는 고-레벨 프로그래밍 기법 또는 모델로서 어레이 기반 연산들이 취급되도록 허용하고 개별 비-어레이의 명시적 루프들로 재분류할 필요 없이 데이터의 전체 애그리게이션들 상에서 연산(즉, 스칼라 연산)할 수 있다.

[0139] 환경은 추가로 데이터 분석, 데이터 가상화, 애플리케이션 개발, 시뮬레이션, 모델링, 알고리즘 개발 등을 위해 이용될 수 있는 매트릭스 및/또는 백터 포뮬레이션(formulation)을 수행하도록 추가로 적응될 수 있다. 이들 매트릭스 및/또는 벡터 포뮬레이션들은 통계, 재무, 영상 프로세싱, 신호 프로세싱, 제어 설계, 생명 과학(life sciences), 교육, 자세한 이벤트 분석 및/또는 설계, 상태 기반 분석 및/또는 설계 등과 같은 다수의 영역들에서 이용될 수 있다.

[0140] 원할 때, 본 발명의 양상들과 일치하는 실시예들 및 기법들은 The MathWorks, Inc.에 의한 Simulink® 모델링 환경, Stateflow®환경, the SimEvents™ 환경; Simulink/Stateflow/SimEvents-호환 가능한 제품들; Visual Solutions에 의한 VisSIM; National Instruments에 의한 LabView®; Dynasim에 의한 Dymola; Measurement Computing에 의한 SoftWIRE; DALSA Coreco에 의한 WiT; Agilent에 의한 VEE Pro 또는 System Vue; PPT Vision에 의한 Vision Program Manager; Khoral Research에 의한 Khoros; Gedae, Inc.에 의한 Gedae; INRIA로부터의 Scicos; Cadence로부터의 Virtuoso; IBM로부터의 Rational Rose; Telelogic으로부터의 Rhopsody 또는 Tau; 버클리의 캘리포니아 대학으로부터의 Ptolemy; 또는 Unified Modeling Language(UML) 또는 SysML 환경의 양상들과 같은(이들로 제한되지 않음) 제품들을 이용하여 그래픽-기반 환경에서 구현될 수 있다.

[0141] **튜닝 가능한 블록들을 특정하기 위한 예시적인 기법들**

[0142] 본 발명의 실시예들은 사용자가 시스템 내의 튜닝 가능한 블록들을 특정하도록 허용할 수 있으며, 여기서 튜닝 가능한 블록들이 최적의 성능 및 강건성(robustness)을 위해 튜닝되어야 하는 제어 시스템의 컴포넌트들이다. 예를 들어, 제어 시스템의 튜닝 가능한 컴포넌트들은 본 발명의 실시예에서 도 7의 정규 구조에서 블록들(B_1, ... , B_N)에 대응할 수 있다.

[0143] 예시적인 실시예들은 이득들, PID들, 전달 함수들 및 상태-공간 모델들과 같은 엘리먼트들을 제어하기 위해 객체들을 이용할 수 있다. 객체들은 추가로 튜닝 가능한 파라미터들을 설명하고 사용자가 파라미터들을 초기화, 세팅(예를 들어, 고정), 또는 해제(free)하도록 허용하는데 이용될 수 있다.

[0144] 예로서, 사용자는 2개의 입력들 및 하나의 출력을 갖는 튜닝 가능한 이득을 생성하기 위해 텍스트 사용자 인터페이스와 같은 인터페이스와 상호작용할 수 있다. 모델링 환경(220)은 인터페이스(1100)를 생성할 수 있다. 예를 들어, 인터페이스(1100)는 디스플레이 디바이스(250)를 통해 사용자에게 디스플레이되는 텍스트 편집 윈도우일 수 있다. 도 11을 참조하면, 예시적인 실시예에서, 사용자는 프롬프트(예를 들어, ">>")에서 커맨드(1105)를 입력할 수 있다.

[0145] 커맨드(1105)에 응답하여, 모델링 환경(220)은 인터페이스(1100)를 통해 사용자에게 디스플레이될 수 있는 객체 G(1110)를 반환할 수 있다. 일 실시예에서, 모델링 환경(220)은 모델링 환경(220)과 호환 가능한 다른 타입들의 객체들을 디스플레이하는데 이용되는 포맷과 유사한 포맷으로 객체 G(1110)를 디스플레이하도록 구성될 수 있다. 예를 들어, 객체 G(1110)는 수치 값보단 오히려 튜닝 가능한 파라미터들의 설명을 포함할 수 있는 "Gain" 특성을 제외하면 제어 시스템 툴박스의 보통의 선형 시간 불변(linear time invariant; LTI) 객체들에 유사할 수 있다. 예를 들어, 그리고 도 12를 참조하면, 사용자는 커맨드(1205)를 입력할 수 있고 모델링 환경(220)은 G.Gain에 대한 값들을 포함할 수 있는 응답(1210)을 반환할 수 있다.

[0146] 도 12를 계속 참조하면, 응답(1201)은 G.Gain에 대한 디폴트 값들을 포함할 수 있다. 예시적인 실시예에서, 디폴트 값들은 사용자가 디폴트 값으로부터 다른 값으로 하나 이상의 값들을 변경할 때까지 모델링 환경(220)에서 이용되는 현재의 값들일 수 있다. 예를 들어, 이들의 디폴트/현재 "Value"은 응답(1210)에서 [0 0]이다. 사용자는 커맨드들(1215 및 1220)을 입력함으로써 이득 매트릭스의 최초의 엔트리를 1로 세팅할 수 있다. 예를 들어, 사용자는 다음을 입력할 수 있다.

[0147]
$$>> G.Gain.Value(1) = 1$$
$$>> G.Gain.Free(1) = false$$

[0148] 사용자는 이어서 G.Gain을 입력할 수 있고 모델링 환경(220)은 출력(1225)을 반환할 수 있다. 도 12에서, 출력(1225)은 이러한 새로운 제약(즉, 디폴트 세팅으로부터 사용자 특정 세팅으로 "Value"을 변경)을 반영하는 이득 매트릭스의 파라미터화를 디스플레이할 수 있다.

[0149] 예시적인 실시예들은 인터페이스(1300) 내에서 도 13에 도시된 커맨드들(1305, 1310, 및 1315)을 각각 이용하여 튜닝 가능한 PID 블록, 전달 함수 블록 또는 상태-공간 블록과 같은 다른 타입들의 블록들을 사용자들이 생성하도록 허용할 수 있다.

[0150] 예시적인 실시예들은 통상적인 또는 주류의 요구들을 해결할 수 있는 미리-정의된 블록들을 사용자에게 제공할 수 있다. 몇몇 사용자들에 대해, 미리-정의된 블록들은 그들의 요구들을 만족시키기에 충분할 수 있다. 그러나 다른 사용자들은 미리-정의된 블록들의 구조에 비해 더 많은 구조를 갖는 블록과 같이 보다 정교한 및/또는 강건한 제어 엘리먼트들에 대한 요구들과 같은 상이한 요구들을 가질 수 있다.

[0151] 예시적인 실시예들은 저역-통과 필터 블록들, 노치(notch) 필터 블록들, 관찰자 형태의 상태-공간 모델들 등과 같은 미리-정의된 블록들과 상이한 블록들을 요구하는 사용자들의 요구들을 지원하도록 적응될 수 있다. 예를 들어, 실시예들은 미리-정의된 블록들과 상이한 블록들의 스위트(suite)를 제공하도록 구성될 수 있지만; 이 모음들은 특정한 사용자들의 요구를 만족시키기 위해 매우 많은 수의 블록들을 포함할 필요가 있을 수 있다. 본

발명의 실시예에서, 사용자들이 단순한 대수(algebra)를 이용하여 그 자신의 파라미터화를 구성하도록 허용하는 프레임워크가 사용자들에게 제공된다. 이 프레임워크는 사용자들의 모든 요구들을 커버하도록 시도하는 블록들의 큰 스위트를 유지할 필요성을 제거한다.

[0152] 일 실시예에서, 프레임워크는 다음을 포함할 수 있다:

[0153] 1. 파라미터의 기본 표기를 실현하는 "파라미터" 객체; 및

[0154] 2. 도 7의 정규 구조로 파라미터들 상의 보통의 연산들을 자동으로 변환하는 컴퓨터 대수(computer algebra).

[0155] 예로서, 저역 통과 필터를 고려한다:

수학식 13

[0156] $$F(s) = \frac{a}{s+a} = \frac{a/s}{1+a/s}$$

[0157] 여기서, alpha는 실제 파라미터이다. 수학식 13은 제 1 차수 전달 함수이지만, 위에서 소개된 전달-함수 블록(도 13, 커맨드(1310))은 동일한 계수 a가 분자 및 분모 둘 다에서 나타난다는 제약(즉, DC 이득은 1로 제한됨)을 표현할 수 없다. 예시적인 실시예에서, 사용자는 다음과 같이, 커맨드들(1320)(도 13)을 입력함으로써 이 제어 엘리먼트를 특정할 수 있다:

[0158]
```
>> s = tf('s');
>> a = realp('a',2);
>> F = feedback(a/s,1)
```

[0159] 커맨드들(1320)이 모델링 환경(220)에 의해 프로세싱되고, @genss 객체 "F"가 반환되며, 여기서 반환된 객체는 단독 블록으로서 "a"로 도 7의 정규 구조를 실현한다. 사용자는 커맨드(1325)를 인터페이스(1300)에 타이핑할 수 있고 모델링 환경(220)은 다음과 같이 응답(1330)을 반환할 수 있다:

[0160]
```
ans =
    a: [1x1 realp]
```

[0161] 본 발명의 실시예들은 "a"가 보통의 이중 값이었던 경우와 동일한 F를 생성하기 위한 신택스(syntax)를 이용할 수 있다. 이 특징은 사용자가 파라미터들과 상호작용하기 위해 새로운 API를 학습해야 할 필요성을 제거한다. 예로서, 제어 시스템 툴박스의 보통의 LTI 객체들을 조작하기 위한 신택스는 파라미터들 및 파라미터적 모델들(parametric models)을 핸들링하도록 끊임없이 확장된다. 본 발명의 실시예들은 원할 때 파라미터들과 상호작용하기 위해 대안의 신택스들을 또한 지원할 수 있다. 예를 들어, 파라미터적 저역-통과 필터 F(s)를 특정하기 위한 대안적인 신택스는 다음과 같이 커맨드(1335)(도 13)를 통해 표현될 수 있다:

```
>> a = realp('a',2);
>> F = tf(a,[1 a])
```

[0162]

[0163] 커맨드들(1335)의 대안적인 신택스는 또한 다음과 같이, 커맨드들(1340)을 통해 표현되는, 제어 시스템 툴박스에서 보통의 전달 함수를 생성하기 위한 신택스와 유사(즉, 필적)하다:

```
>> a = 2;
>> F = tf(a,[1 a])
```

[0164]

[0165] 커맨드들(1335 또는 1340)의 입력은 인터페이스(1300)를 통해 결과적인 전달 함수를 디스플레이하는 응답(1345)을 반환할 수 있다.

[0166] 본 발명의 실시예들은 위의 예와 비교해서 보다 복잡한 본질의 제어 문제들을 해결하는데 이용될 수 있다. 예로서, 관찰자 형태의 상태-공간 제어기는 다음과 같이 제공될 수 있다.

수학식 14

$$\begin{cases} \dot{x} = Ax + Bu + L(y - Cx - Dy) \\ u = -Kx \end{cases}$$

[0167]

[0168] 수학식 14에서, A, B, C, D는 알려진 매트릭스들이며, 튜닝 가능한 변수들은 이득 매트릭스들 K(상태-피드백 이득) 및 L(관찰자 이득)이다. 제어기들의 구조화된 상태-공간 파라미터화는 예를 들어, 이득 스케줄링에 유용하다. 예시 목적들을 위해, 본 발명의 실시예를 이용할 때, 2개의 측정들(y), 하나의 제어 신호(u) 및 3개의 상태들(x)이 존재한다고 가정될 수 있다. 사용자는 다음과 같은 커맨드(1410)(도 14)를 타이핑함으로써 이러한 튜닝 가능한 블록을 생성할 수 있다:

```
>> K = realp('K',zeros(1,3));
>> L = realp('L',zeros(3,2));
>> OBC = ss(A-B*K-L*C+L*D*K,L,-K,0)
```

[0169]

[0170] 사용자는 이어서 다음과 같이 커맨드(1420)를 타이핑할 수 있다:

```
>> OBC.Blocks
```

[0171]

[0172] 모델링 환경(220)은 커맨드를 프로세싱할 수 있고 인터페이스(1400)를 통해 다음과 같은 응답(1430)을 디스플레이할 수 있다:

ans =

K: [1x3 realp]
L: [3x2 realp]

[0173]

[0174] 도 13과 함께 논의되는 예에서와 마찬가지로, 도 14와 함께 논의되는 예는 모델링 환경(220)에서 보통의 상태-공간 모델들을 생성하기 위한 신택스와 유사한(즉, 실질적으로 반영하는(mirror)) 신택스를 이용한다. 실시예에서, 도 14의 커맨드들의 입력 결과는 구조 보상기를 갖는, 도 7의 정규 구조를 캡슐화하는 @genss 모델이다:

수학식 15

$$C(s) = \begin{bmatrix} K & 0 & 0 & 0 & 0 & 0 \\ 0 & K & 0 & 0 & 0 & 0 \\ 0 & 0 & K & 0 & 0 & 0 \\ 0 & 0 & 0 & L & 0 & 0 \\ 0 & 0 & 0 & 0 & L & 0 \\ 0 & 0 & 0 & 0 & 0 & L \end{bmatrix}$$

[0175]

[0176] 도 7을 재차 참조하면, 도 7의 정규 구조에 대한 2개의 조각들 즉 (1) 상호연결 모델 H(s) 및 (2) 튜닝 가능한 블록들(B_1, ... , B_N)이 존재한다. 도 8 내지 도 14 및 수반하는 텍스트는 튜닝 가능한 블록들(선행 문장에서 아이템(2))을 특정하기 위한 툴들 및 기법들을 논의한다. 상호연결 모델 H(s)를 구축하기 위한 툴들 및 기법들은 아래에서 논의된다.

[0177] **정규 구조를 유도하기 위한 예시적인 기법**

[0178] 예시적인 실시예들 및 기법들은 2개 이상의 접근법들을 지원하는데 이용될 수 있다. 예를 들어, 실시예들 및 기법들은 모델링 환경(예를 들어, Simulink 모델링 환경, Simulink-호환 가능한 환경 등과 같은 그래픽 모델링 환경 등) 내에 존재하는 툴들을 이용하여 H(s)를 자동으로 추출하는 것을 지원할 수 있다. 예를 들어, H(s)는 Simulink 모델링 환경에서 기존의 선형화 툴들을 이용하여 자동으로 추출될 수 있다. 그리고 (2) H(s)는 텍스트 컴퓨팅 및/또는 모델링 환경(예를 들어, MATLAB 프로그래밍 환경, MATLAB-호환 가능한 환경 등)과 연관된 표준 커맨드들을 이용하여 자동으로 구축될 수 있다. 예를 들어, H(s)는 MATLAB 프로그래밍 환경과 상호작용하는 사용자에 의해 블록도들을 구축하기 위한 커맨드들을 이용하여 구축될 수 있다.

[0179] 예시적인 환경들은 그래픽 모델링 환경들과 같은 그래픽-기반 환경들에서 튜닝 가능한 블록들을 특정하는데 이용될 수 있다. 예로서, 사용자는 제어 시스템들에서 이용하기 위한 튜닝 가능한 블록들을 특정하기 위해 Simulink 모델링 환경과 상호작용할 수 있다.

[0180] Simulink 환경에서, H(s)는 (1) 튜닝될 블록들(B_1, ..., B_N)(도 7) 및 (2) 외부 I/O 신호들(w 및 z)(도 7)의 지식에 의해 완전히 결정될 수 있다. 튜닝 가능한 블록들은 블록도에서 Simulink 블록들로서 나타날 때, 이 정보는 다음과 같이 쉽게 제공될 수 있다:

[0181] 1. 블록들(B_1, ..., B_N) 각각에 대한 블록 경로들의 리스트; 및

[0182] 2. w 및 z로 구성된 신호들이 블록도에 입력되고 블록들을 나가는 곳을 특정하는 선형 입력 및 출력 지점들.

[0183] 정보가 특정되면, LINLFT 같은 커맨드가 LTI 모델로서 H(s)를 계산하고 반환한다. 위의 프로시저는 선형 및 비선형 모델들에 적용될 수 있다. 프로시저가 비선형 모델들에 적용될 때, 제어 아키텍처의 선형화된 모델이 생성된다.

[0184] MATLAB 환경과 같은 텍스트 프로그래밍 환경에서 H(s)의 유도는 튜닝 가능한 블록들(예를 들어, 도 11 내지 14 및 수반되는 텍스트)의 사양에 대해 위에서 논의된 컴퓨터 대수에 의존할 수 있다. 예시적인 실시예들은 도 7에서 도시된 파라미터적 LFT 모델들의 파라미터 객체들, 파라미터적 블록들 및 타입을 또한 핸들링하기 위해 보통의 LTI 모델들(예를 들어, Math Works 제어 시스템 툴박스와 함께 행해질 수 있는 바와 같이)을 조합하기 위해 사용자가 통상적인 연산을 확장하는 것을 가능하게 한다. 몇몇 통상적인 연산들의 예들은,

[0185] · 보통의 대수 : 더하기, 빼기, 곱하기, 나누기, 반전시키기 등; 및

[0186] · 상호연결 연산들: 입력 또는 출력 연계(concatenation). 부가(append), 병렬, 직렬, 피드백, 리프트(lft), 연결 등

[0187] 을 포함(그러나 이들로 제한되지 않음)할 수 있다.

[0188] 예시적인 실시예들은 (도 7에서 도시된 바와 같은 튜닝 가능한 블록들의 블록-대각 배열을 갖는 LTI 모델 또는 매트릭스의 LFT 상호연결로서 정의된 바와 같이) 정적 및 동적 선형 분수 변환(linear fractional transform; LFT) 모델들의 관념들(notion) 및 객체-지향 프레임워크를 이용한다. 예시적인 실시예들은 이중 어레이들, LTI 모델들, 파라미터들, 튜닝 가능한 블록들, 및 LFT 모델들을 사용자가 조합하도록 허용하는 객체 시스템(object system)을 생성할 수 있다. 예를 들어, LFT 모델들은 가장 일반적인 모델들이며, 이중 어레이, 상태-공간 모델, 주파수 응답 데이터 모델과 같이 H의 3개의 상이한 수치 표현들에 각각 대응하는 @genmat, @genss, 및 @genfrd와 같은 3개의 타입들로 이루어질 수 있다. 결과적인 객체 시스템은 정지(close)되고, 실질적으로 튜닝 가능한(파라미터적) 엘리먼트들과 LTI 모델들의 임의의 조합은 도 7의 정규 구조를 갖는 @genss 또는 @genfrd 모델을 생성할 것이다.

[0189] 도 15는 본 발명의 예시적인 실시예들에 의해 제공되는 객체 기반 프레임워크를 이용하여 상호작용될 수 있는 실행 가능한 그래픽 모델(1500)을 예시한다. 도 15는 복수의 실행 가능한 블록들을 포함할 수 있는 모델(1500)을 포함할 수 있다. 스코프(scope) 블록(1505) 및 증류탑(distillation column)(플랜트) 블록(1510)(이하 플랜트 블록(1510))과 같은 모델(1500)의 몇몇 블록들은 튜닝 가능하지 않을 수 있다. 대조적으로 이득 블록(1515) 및 PI 제어기 블록들(1520 및 1525)과 같은 모델(1500)의 다른 블록들은 튜닝 가능할 수 있다. 모델링 환경(220)의 실시예들은 제 1 타입의 가시적 식별자(예를 들어, 라인 폭, 쉐이딩(shading), 컬러, 필(fill) 패턴, 텍스트 패턴 등)를 이용하여 튜닝 가능한 블록들을 나타내고 제 2 타입의 가시적 식별자를 이용하여 비-튜닝 가능한 블록들을 표시하도록 구성될 수 있다.

[0190] 도 15를 계속 참조하면, 플랜트 블록(1510)의 LTI 모델(G)은 모델(1500)에서 제공될 수 있고, 사용자는 r로부터 y로 폐루프 전달의 LFT 모델을 구성하기 위해 모델(1500)과 상호작용할 수 있다. 예를 들어, 사용자는 다음과 같은 커맨드들(1605 및 1610)(도 16a)을 입력할 수 있다:

```
% 튜닝 가능한 블록을 특정
>> DM = ltiblock.gain('Decoupler',eye(2));
>> PI_L = ltiblock.pid('PI_L','pi');
>> PI_V = ltiblock.pid('PI_V','pi');
```

[0191]

[0192] 및

```
% 폐-루프 전달 T를 유도
>> C = append(PI_L,PI_V) * DM;
>> T = feedback(G * C , eye(2));
```

[0193]

[0194]

[0195] 모델링 환경(220)은 커맨드들(1605 및 1610)을 프로세싱할 수 있고 @genss 모델 T를 생성할 수 있다. 사용자는 모델링 환경(220)과 상호작용할 수 있고, 다음과 같은 커맨드(1615)를 입력함으로써 (도 7에서와 같이) 모델 T를 그의 부분(H 및 B)으로 분해할 수 있다:

[0196]
```
>> [H,B] = getLFTModel(T)
```

[0197] 모델링 환경(220)은 커맨드(1615)를 프로세싱할 수 있고 다음과 같은 응답(1620)을 반환할 수 있다:

```
a =
        x1       x2
   x1  -0.01333        0
   x2         0  -0.01333

b =
       u1  u2  u3  u4  u5  u6
   x1   0   0   2   0   0   0
   x2   0   0   0   2   0   0

c =
            x1      x2
   yD   0.5853   -0.576
   yB   0.7213  -0.7307
   ?   -0.5853    0.576
   ?   -0.7213   0.7307
   ?   -0.5853    0.576
   ?   -0.7213   0.7307

d =
       u1  u2  u3  u4  u5  u6
   yD   0   0   0   0   0   0
   yB   0   0   0   0   0   0
   ?    1   0   0   0   1   0
   ?    0   1   0   0   0   1
   ?    1   0   0   0   0   0
   ?    0   1   0   0   0   0
```

[0198]

[0199] 연속적 시간-시간 모델.

[0200]
```
B =

    [1x1 ltiblock.pid ]
    [1x1 ltiblock.pid ]
    [2x2 ltiblock.gain]
```

[0201] **튜닝 가능한 파라미터들을 최적화하기 위한 예시적인 기법**

[0202] 일부 가중된 폐루프 모델 T(s)의 견지에서 H∞ 제약

[0203] $$\|T(s)\|_\infty < 1$$

[0204] 로서 설계 요건들을 포뮬레이팅하기 위해 모델(1500) 및 모델링 환경(2200)과 사용자가 상호작용하면, 사용자는 다음과 같은 커맨드를 입력함으로써 최적화기를 시동할 수 있다:

[0205]
```
>> T = hinfstruct(T0,options)
```

[0206] 여기서 T0는 튜닝되지 않은(un-tuned) 폐루프 모델이고, 옵션들은 최적하기를 위해 세팅된 몇몇 옵션이다. 모델링 환경(220)은 커맨드를 프로세싱할 수 있고 튜닝 가능한 파라미터들이 최적화기에 의해 발견된 특정 값들로 세팅되는 튜닝된 폐루프 모델 T를 반환할 수 있다. 예를 들어, 일 실시예에서, 모든 튜닝 가능한 파라미터들은 최적화기(360)에 의해 결정된 바와 같은 최상의 값들로 세팅될 수 있다.

[0207] 사용자는 하나 이상의 튜닝 가능한 블록들에 대한 튜닝된 값들에 액세스하고자 할 수 있다. 예를 들어, 일 실시예에서, 사용자는 다음과 같은 커맨드를 입력함으로써 각각의 튜닝 가능한 블록의 튜닝된 값들에 액세스할 수 있다:

[0208]
```
>> T.Blocks
```

[0209] 특정한 상황들에서, C0는 하나 이상의 튜닝 가능한 블록들을 포함하는 보상기로 구성될 수 있다. 이 상황들에서, 사용자는 본 발명의 예시적인 실시예를 이용하여 C0에 대한 LFT 모델에 튜닝된 파라미터 값들을 "푸시(push)"할 수 있다. 예를 들어, 사용자는 다음과 같은 커맨드를 타이핑할 수 있다:

[0210]
```
>> C = replaceBlock(C0,T.Blocks)
```

[0211] 모델링 환경(220)은 커맨드를 프로세싱할 수 있고 T에서의 그들의 튜닝된 값들로 C0에서의 관련 파라미터를 자동으로 대체할 수 있다. 예시적인 실시예들은 Simulink, 또는 Simulink-호환 가능한 모델과 같은 모델에서 블록 파라미터들이 튜닝의 능률화(streamlining)를 용이하게 하기 위해 도우미 함수들(helper functions)과 같은

함수를 포함할 수 있다. 예로서 그리고 Simulink을 참조하면,

[0212] · LINLFT는 튜닝을 위해 선택된 Simulink 블록들에 기초하여 올바른 튜닝 가능한 블록 객체들(ltiblock.gain, ltiblock.tf,...)을 자동으로 선택하고 구성하도록 확장될 수 있고,

[0213] · 도우미 함수는 hinfstruct에 의해 계산된 튜닝된 파라미터 값들을 Simulink 블록으로 역으로(back) 푸시하도록 제공될 수 있다.

[0214] 도우미 함수들은 문서화된 함수들이 동작할 때 문서화된 함수들에 의해 이용되는, 비문서화된 함수들과 같은 함수들을 포함할 수 있다.

[0215] **결과를 확인하기 위한 예시적인 기법**

[0216] 예시적인 실시예들은 튜닝 가능한 블록들 및 LFT 모델들을 보통의 LTI 모델들의 피어들(peers)로서 취급할 수 있다. 이를 행하는 것은 예시적인 실시예들이 STEP 또는 BODE같은 표준 커맨드들을 이용하여 튜닝된 제어 시스템을 직접 분석하도록 허용할 수 있다. 예를 들어, 사용자는 다음과 같은 커맨드를 타이핑할 수 있다:

[0217] $>> \text{bode}(T)$ % T의 플롯들 Bode 응답

[0218] 모델링 환경(220)은 커맨드를 프로세싱할 수 있고 디스플레이 디바이스를 통해 하나 이상의 Bode 플롯들(plots)을 사용자에게 디스플레이할 수 있다.

[0219] 예시적인 실시예들은 표준 tf(), ss(), zpk(), frd(), pid() 커맨드들을 이용하여 사용자가 튜닝 가능한 엘리먼트들을 보통의 LTI 모델로 변환하도록 추가로 허용할 수 있다. 예를 들어, F가 T를 구성하는 튜닝 가능한 블록들 중 하나인 경우, 사용자는 다음과 같은 커맨드를 이용하여 F를 전달 함수로 변환할 수 있다:

[0220] $>> \text{Ftuned} = \text{tf}(\text{T.Blocks.F})$ % 전달 함수로서 F의 튜닝된 값을 획득

[0221] 위에서 기술된 커맨드들 및 기법들은 사용자가 종래의 툴들과 상호작용할 때 이용 가능하지 않은 강건한 튜닝 성능들(robust tuning capabilities)을 사용자에게 제공한다. 이 기법들/커맨드들은 텍스트 또는 그래픽 모델링 환경 오버레이를 더 복잡하게 하거나 사용자들에 대해 혼란스럽게 하지 않고 강건한 성능들을 추가로 제공한다.

[0222] **본 발명의 실시예를 실시하기 위한 예시적인 작업흐름**

[0223] 도 17은 F14 제트 전투기에 이용되는 아날로그 오토파일롯의 예시적인 모델(1700)을 예시한다. 본 발명의 예시적인 모델들 및 기법들은 F14에 대한 아날로그 오토파일롯 시스템을 튜닝하기 위해 모델(1700)에 적용될 수 있다. 예를 들어, 실시예들 및 기법들은 Simulink 모델링 환경을 이용하여 F14의 세로축에 대한 오토파일롯을 튜닝할 수 있다. 모델(1700)은 표준 캐스캐이드-루프 오토파일롯(standard cascade-loop autopilot)을 구현할 수 있는 제어기 블록(1705)을 포함할 수 있다.

[0224] 도 17을 참조하면, 모델(1700)은 피치 레이트(q)를 명령하는 내부 루프(1710) 및 받음각(angle of attack) alpha를 명령하는 외부 루프(1715)를 포함할 수 있다.

[0225] 도 18은 제어기 블록(1705)을 더 상세히 예시한다. 도 18에서 알 수 있는 바와 같이, 윈도우(1705-A)는 도 17

의 제어기 블록(1705)에 포함된 컴포넌트들 및 연결들을 포함한다. 도 18에서, 오토파일롯은 고정된 구조를 가지며 튜닝될 필요가 있을 수 있는 이득 및 프리필터 블록들로 구성된다. 예를 들어, 튜닝 가능한 이득 블록들(1820, 1825 및 1830)은 삼각형 형상과 쉐이딩(shading)을 갖고 도시되며, 여기서 쉐이딩은 이득 블록이 튜닝 가능하다는 것을 사용자에게 표시할 수 있다. 프리필터 블록들(1805, 1810 및 1815)은 형상이 직사각형일 수 있고, 쉐이딩될 수 있으며, 여기서 쉐이딩은 프리필터 블록이 튜닝 가능하다는 것을 표시할 수 있다. 도 18은 튜닝될 수 있는 쉐이딩된 PI 보상기(1840)를 더 포함할 수 있다. 본 발명의 실시예들은 튜닝 가능한 이득 블록들을 표시하기 위해 제 1 컬러 및/또는 쉐이딩 밀도(shading density) 및/또는 튜닝 가능한 프리필터 블록 및/또는 PI 보상기 블록들을 표시하기 위해 제 1 컬러 및/또는 쉐이딩 밀도를 이용할 수 있다.

[0226] 예시적인 실시예들 및 기법들은 원하는 성능 및 강건성 레벨을 달성하기 위해 사용자가 선택된 제어 엘리먼트들을 자동으로 그리고 공동으로 튜닝하도록 허용한다.

[0227] 도 19를 참조하면, 사용자는 도 17 및 도 18의 오토파일롯에 대한 파라미터들을 정의하기 위해 다음과 같은 커맨드들(1905)을 입력할 수 있다:

[0228]
```
s = tf('s');
pKa = realp('Ka',0);
pKq = realp('Kq',0);
pKf = realp('Kf',0);
pKi = realp('Ki',1);
pTal = realp('Tal',1);
pW1 = realp('W1',0);
pW2 = realp('W2',0);
```

[0229] 도 19에서, 사용자는 실제 스칼라 파라미터들로서 모든 파라미터들을 정의하도록 선택할 수 있다. 커맨드(1905)에서, 사용자는 파라미터들 대부분을 0으로 초기화하도록 결정할 수 있다.

[0230] 사용자는 커맨드(1905)의 파라미터들을 수반하는 표현들로서 alpha 및 피치 프리필터들 및 적분기를 생성하기 위해 모델링 환경(220)과 상호작용할 수 있다. 예를 들어, 사용자는 적분기, alpha 및 피치 프리필터를 생성하기 위해 다음과 같은 커맨드(1910)를 입력할 수 있다:

[0231]
```
Integrator = pKi/s;                % Ki/s
AlphaSensor = tf(1,[pTal 1]);      % 1/(Tal*s+1)
PitchFilter = tf([1 pW1],[1 pW2]); % (s+W1)/(s+W2)
```

[0232] **목적를 트래킹하기 위한 예시적인 기법**

[0233] 사용자는 도 17 및 도 18의 오토파일롯 모델과 상호작용할 때 H∞ 제약으로서 트래킹 목적(tracking objective)를 포뮬레이팅하고자 할 수 있다. 원하는 대역폭은 1 rad/s(radian/second)일 수 있고, 사용자는 받음각 alpha가 오버슛을 거의 갖지 않는 스틱 입력(stick input)(파일롯 요구)을 트래킹하는 것을 원할 수 있다. 예시적인 실시예들 및 기법들은 외부 루프(alpha)(1715)(도 17)에 대한 개루프 형상의 견지에서 사용자의 요건들을 포뮬레이팅하는데 이용될 수 있다.

[0234] 사용자는 다음과 같은 커맨드들(1915)을 입력함으로써 실제적인 타겟 루프 형상이라고 여겨지는 것을 정의할 수 있다:

```
wc = 1;   % 타겟 크로스오버
LS = (1+0.01*s/wc)/(0.01+s/wc);
bodemag(LS,{1e-3,1e3}), grid, title('Target loop shape')
```

[0235]

[0236] 모델링 환경(220)은 커맨드(1915)를 프로세싱하고 GUI(2000)를 통해 도 20에서 도시된 바와 같은 Bode 플롯을 디스플레이할 수 있다. GUI(2000)는 대략 1 rad/s(0dB 크로스오버) 정도의 대역폭 및 1%(즉, w=0에서 40dB 이득)를 초과할 가능성이 적은 정상-상태 에러를 보장하는 루프 형상을 디스플레이한다. 사용자는 그가 실제적으로 w=0에서 더 많은 이득을 기대할 수 없다는 것을 깨달을 수 있는데, 그 이유는 외부 루프가 적분기(즉, 적분기는 도 17의 피치 레이트 루프에 있음)를 포함하지 않기 때문이다.

[0237] alpha로의 폐루프 모델 맵핑(r, n, d)(여기서 r은 alpha 커맨드이고, n은 alpha 상의 측정 노이즈이고, d는 액추에이터에서 입력되는 외란임)은 본 명세서에서 위에서 논의된 예시적인 루프 형상화 기법들이 이용될 수 있도록 유도될 필요가 있을 수 있다. 예시적인 실시예에서, 사용자는 (1) 모델의 튜닝된 블록들을 나열함으로써, (2) 신호들(r, n, d, alpha)을 선형화 입력/출력들(I/O들)로서 표기함으로써, 및 (3) (r, n, d)로부터 y=alpha로 폐루프 전달의 LFT 모델을 추출하기 위해 LINLFT 커맨드를 이용함으로써 폐루프 모델을 유도할 수 있다. 예로서, 사용자는 도 21에 예시되는 코드를 생성할 수 있다. 예를 들어, 사용자는 텍스트를 인터페이스(2100)에 입력함으로써 코드 부분(2105 및 2110)을 생성할 수 있다:

```
% 튜닝된 블록들
TunedBlocks = {...
    'f14/Controller/Alpha-sensor Low-pass Filter',...
    'f14/Controller/Pitch Rate Lead Filter',...
    'f14/Controller/Ka',...
    'f14/Controller/Kq',...
    'f14/Controller/Kf',...
    'f14/Controller/Integrator'};
%  도 4의 구조화된 보상기들
C = blkdiag(AlphaSensor,PitchFilter,pKa,pKq,pKf,Integrator);

linios = [...
    linio('f14/Controller/Stick Prefilter',1,'in') ; ... % r
    linio('f14/Aircraft Dynamics Model',4,'outin') ; ... % alpha,n
    linio('f14/Controller',1,'in') ]; % d
H = linlft('f14',linios,TunedBlocks);
% 폐-루프 전달  [r;n;d] -> y=alpha
Ta0 = lft(H,C);
```

[0238]

[0239]

[0240] 여기서, 사용자는 다음을 입력함으로써 T10의 출력으로서 부가될 수 있는 에러 신호 e = r - y를 요구할 수 있다:

```
T10 = [0 0 0;1 0 0] + [1;-1] * Ta0;
```

[0241]

[0242] 사용자는 본 명세서에서 위에서 논의된 루프 형상화 가중치들을 부가함으로써 H∞ 제약으로서 포뮬레이션을 완료할 수 있다. 예를 들어, 사용자는 다음과 같은 커맨드(2205)를 인터페이스(220)(도 22)에 입력할 수 있다:

[0243]
```
beta = 3;
Win = blkdiag(1,1/LS,beta);
Wout = blkdiag(1,LS);
T10 = Wout * T10 * Win;
T10.InputName = {'r','n','d'};
T10.OutputName = {'y','e'};
```

[0244] 대안적인 실시예에서, 루프 형상화 가중치들은 사용자 입력을 요구하지 않고 모델링 환경(2200)에 의해 프로그램적으로(programmatically) 부가될 수 있다.

[0245] 도 21 및 도 22와 연관된 입력들이 모델링 환경(220)으로 입력될 때, 도 20의 루프 형상화 목적는 H∞ 제약

[0246] $$\|T_1(s)\|_\infty < 1$$

[0247] 에 의해 포착될 수 있다.

[0248] **강건성 목적를 식별하기 위한 예시적인 기법**

[0249] 도 17의 캐스캐이드 루프 구조는 2개의 측정들(alpha 및 q) 및 하나의 제어(엘레베이터 편차(elevator deflection) delta)를 갖는 MIMO 제어 구조이다. 몇몇 예들에서, 사용자는 각각의 SISO 루프의 안정성 마진들을 평가하도록 시도할 수 있지만, 이러한 접근법은 충분한 MIMO 마진들을 보장하는데 항상 충분한 것은 아니다. 본 발명의 실시예들에 의해 지원되는 보다 신뢰할 수 있고 정확한 접근법은 감도 함수

수학식 16

[0250] $$S = (I + GK)^{-1}$$

[0251] 의 피크 이득이 1에 근접하게 유지되는 것을 보장한다. 수학식(16)에서, G 는 delta를 (alpha, q)에 맵핑하고 K는 (alpha, q)로부터 delta를 생성하는 애그리게이트 제어기이다. 즉, 제 2 요건은 다음과 같다:

수학식 18

[0252] $$\|T_2(s)\|_\infty < 1 \text{ with } T_2 = S$$

[0253] T2의 폐루프 모델은 플랜트의 출력(alpha, q)에서 측정되고 이전에 논의된 바와 같이 LINLFT를 이용한다. 예를 들어, 사용자는 다음과 같은 커맨드(2210)(예를 들어, 도 22)를 입력할 수 있다:

```
linios = [...
    linio('f14/Aircraft Dynamics Model',4,'inout') ; ... % alpha
    linio('f14/Aircraft Dynamics Model',3,'inout') ];   % q
H = linlft('f14',linios,TunedBlocks);
S0 = lft(H,C);
T20 = S0;
```

[0254]

[0255] **예시적인 튜닝 기법**

[0256] T2의 폐루프 모델이 유도되면, 사용자는 제어기 파라미터들을 튜닝하고자 할 수 있다. 예시적인 실시예는 단일 제약 내로 트래킹 및 강건성에 대한 H∞ 제약들을 조합하는 것을 용이하게 한다. 예를 들어, 단일의 제약은 사용자가 다음을 정의할 때 표현될 수 있다:

```
T0 = blkdiag(T10,T20);
```

[0257]

[0258] 모델링 환경(220)은 최적화기(360)를 이용하여 5개의 최적화들을 실행하도록 HINFSTRUCT 솔버(solver)에 지시할 수 있다. 예를 들어, 최적화기(360)는 로컬 최소치에서 조급한 종결의 위험을 완화하기 위해 5개의 상이한 시작 지점들로부터 5개의 최적화들을 실행할 수 있다. 최적화기(360)는 높은 이득 설계를 방지하기 위해 폐루프 폴들의 크기를 50으로 추가로 제한할 수 있다. 사용자는 다음과 같은 커맨드를 입력함으로써 위의 최적화를 수행하도록 최적화기(360)를 구성할 수 있다:

```
Options = hinfstructOptions('Display','final','RandomStart',4,'SpecRadius',50);
```

[0259]

[0260] 사용자는 다음과 같은 커맨드를 입력함으로써 특정한 최적화를 실행할 수 있다:

```
T = hinfstruct(T0,Options);
```

[0261]

[0262] 모델링 환경(220)은 커맨드들이 실행될 때 튜닝된 목적(T)를 반환할 수 있다.

[0263] **해결책을 확인하기 위한 예시적인 기법**

[0264] 사용자는 제어기 설계를 확인하고자 할 수 있고 예시적인 실시예들은 사용자가 상호작용 방식으로 확인들을 수행하도록 허용한다. 일 실시예에서, 설계는 폐루프 전달들(Ta0(alpha 루프에 대한) 및 S(감도 함수))에 튜닝된 파라미터 값들을 푸시함으로써 확인될 수 있다. 사용자는 다음과 같이 2개의 커맨드들을 입력할 수 있다:

[0265]
```
Ta = replaceBlock(Ta0,T.Blocks);
S = replaceBlock(S0,T.Blocks);
```

[0266] 모델링 환경(220)은 커맨드들을 프로세싱할 수 있고 alpha 루프의 폐루프 응답을 시뮬레이션할 수 있다.

[0267] 도 23은 모델(1700)의 alpha 루프에 대한 폐루프 응답을 보여주는 예시적인 플롯(2300)을 예시한다. 예를 들어, 사용자는 플롯(2300)이 디스플레이 디바이스(250)를 통해 디스플레이되게 하기 위해 다음과 같은 커맨드를 입력할 수 있다:

[0268]
```
step(Ta(1,1))  % alpha에 대한 alpha 커맨드
```

[0269] 예시적인 실시예들은 S의 피크 이득이 0dB에 근접하다는 것을 사용자가 검증하도록 추가로 허용할 수 있다. 예를 들어, 사용자는 다음을 타이핑할 수 있다:

[0270]
```
sigma(S)  % 감도 함수의 피크 이득
```

[0271] 모델링 환경(220)은 커맨드를 프로세싱할 수 있고 디스플레이 디바이스(250)를 통해 도 24의 플롯(2400)을 디스플레이할 수 있다.

[0272] 예시적인 실시예들은 모델링 환경(220)과 상호작용함으로써 사용자가 튜닝된 파라미터들의 값(예를 들어, Kq)에 액세스하도록 추가로 허용한다. 예를 들어, 사용자는 다음을 타이핑할 수 있고:

[0273]
```
T.Blocks.Kq.Value  % 이득의 튜닝된 값
```

[0274] 모델링 환경(220)은 커맨드가 프로세싱될 때 응답을 생성할 수 있다. 예를 들어, 다음이 사용자에게 디스플레이될 수 있다:

[0275]
```
ans =

    4.3486e-001
```

[0276] **예시적인 프로세싱**

[0277] 도 25는 본 발명의 하나 이상의 실시예들을 실시하기 위한 예시적인 프로세싱을 예시한다. 사용자는 하나 이상의 튜닝 가능한 컴포넌트들을 포함하는 모델을 생성할 수 있다(동작 2505). 예를 들어, 사용자는 그래픽 모델과 상호작용할 수 있고 신호들을 표현하는 라인들에 의해 연결되는 다수의 컴포넌트들을 포함하는 자유로운 형태의 그래픽 모델을 구축할 수 있다. 모델은 하나 이상의 피드백 루프들을 포함할 수 있고 플랜트로서 표현될

수 있는 고정된 컴포넌트들 및 제어기를 통해 표현될 수 있는 하나 이상의 튜닝 가능한 컴포넌트들을 포함할 수 있다. 사용자는 고정된 및 튜닝 가능한 모델 컴포넌트들을 생성하기 위한 커맨드들을 사용자가 입력하는 텍스트 환경을 이용하여 모델을 추가로 생성할 수 있다. 일 실시예에서, 동작(2505)은 본 발명의 실시예가 기존의 모델을 수신할 때와 같이 선택적일 수 있다.

[0278] 모델이 식별될 때, 모델과 연관된 설계 요건들이 수신될 수 있다(동작(2510)). 예를 들어, 사용자는 모델에 대한 개방 루프 응답을 위해 원하는 이득 프로파일을 특정할 수 있고, 구조화된 제어기 내로 그룹핑될 수 있는 모델에서의 튜닝 가능한 블록들을 특정할 수 있고 및/또는 모델에 대한 폐 루프 표현에 대한 입력, 출력 및 외란의 위치들을 식별할 수 있다. 사용자는 대안적으로 모델에 대한 설계 요건들을 특정하는데 이용될 수 있는 텍스트 환경에서 커맨드와 상호작용할 수 있다. 다른 실시예들에서, 설계 요건들은 데이터 저장소로부터 판독되고, API를 통해 프로그램적으로 수신되는 등이 될 수 있다.

[0279] 설계 요건들은 본 발명의 예시적인 실시예를 이용하여 H∞ 제약 내로 포뮬레이팅될 수 있다(동작(2515)). 예를 들어, 본 발명의 실시예는 모델의 컴포넌트들을 튜닝하는데 이용하기 위해 H∞ 제약 내로 사용자 입력 디바이스를 통해 수신된 설계 요건들(예를 들어, 루프 형상 목적)을 자동으로 변환할 수 있다. 컴퓨터(205)는 H∞ 합성 기법들을 이용하여 프로세싱 동작들을 수행할 수 있고, 플랜트, 제어기 및 전달 함수를 추출할 수 있다(동작(2520)). 일 실시예에서, 제어기는 블록 대각 구조로 배열되는 튜닝 가능한 컴포넌트들의 튜닝 가능한 파라미터들을 포함할 수 있다. 예를 들어, H∞ 제약 내로의 설계 요건들의 변환은 도 7에서 도시된 바와 같은 정규 구조로 제어 아키텍처를 맵핑하는 것을 포함할 수 있다.

[0280] 변환 이후에, 플랜트, 제어기 및 전달 함수는 모델의 파라미터들을 튜닝하는데 이용될 수 있는 최적화기와 호환 가능한 포맷이 될 수 있다. 호환 가능한 포맷은 최적화기에 대한 입력일 수 있고, 최적화기는 구조화된 제어기와 연관된 컴포넌트들을 튜닝할 수 있다(동작(2525)). 일 실시예에서, 최적화기는 튜닝되지 않은 컴포넌트들을 수신할 수 있다.

[0281] 최적화기는 튜닝된 컴포넌트들을 포함하는 결과를 생성할 수 있다(동작(2530)). 예를 들어, 튜닝된 컴포넌트들은 모델에 대한 안정성 목적를 만족시키면서 동작(2510)에서 수신된 설계 요건들을 만족시킬 수 있다.

[0282] 최적화기에 의해 생성되는 결과는, 그 결과가 원하는 방식으로 모델이 동작하도록 허용할 것임을 보장하도록 확인될 수 있다(동작(2535)). 예를 들어, 결과의 확인은 오버슛 마진(overshoot margin)이 초과되지 않고, 원하는 응답 레이트가 달성되고, 안정성 마진이 유지되고 원하는 루프 이득이 달성되고, 원하는 외란 제거가 달성되는 것을 보장할 수 있다.

[0283] **예시적인 아키텍처**

[0284] 도 26은 도 2의 컴퓨터(205)를 구현하는데 이용될 수 있는 예시적인 컴퓨터 아키텍처를 예시한다. 도 26은 컴퓨터(205)에 대응하는 엔티티의 예시적인 다이어그램이다. 예시되는 바와 같이, 엔티티는 버스(2610), 프로세싱 로직(2620), 메인 메모리(2630), 판독-전용 메모리(ROM)(2640), 저장 디바이스(2650), 입력 디바이스(2660), 출력 디바이스(2670), 및/또는 통신 인터페이스(2680)를 포함할 수 있다. 버스(2610)는 엔티티의 컴포넌트들 사이에서 통신을 허용하는 경로를 포함할 수 있다.

[0285] 프로세싱 로직(2620)은 프로세서, 마이크로프로세서, 또는 다른 타입의 프로세싱 로직(예를 들어, 필드 프로그래밍 가능한 게이트 어레이(field programmable gate array; FPGA), 그래픽 프로세싱 유닛(graphics processing unit; GPU), 디지털 신호 프로세서(digital signal processor; DSP), 주문형 집적 회로(application specific integrated circuit; ASIC), 프로그래밍 가능한 로직 디바이스(programmable logic device; PLD) 등)을 포함할 수 있다. 구현을 위해, 프로세싱 로직(2620)은 단일 코어 프로세서 또는 다중-코어 프로세서를 포함할 수 있다. 다른 구현에서, 프로세싱 로직(2620)은 단일 프로세싱 디바이스 또는 프로세싱 클러스터 또는 컴퓨팅 그리드와 같은 프로세싱 디바이스들의 그룹을 포함할 수 있다. 또 다른 구현에서, 프로세싱 로직(2620)은 서로에 관해 로컬, 또는 원격일 수 있는 다수의 프로세서들을 포함할 수 있고 프로세싱 동안 하나 이상의 스레드들을 이용할 수 있다.

[0286] 메인 메모리(2630)는 랜덤 액세스 메모리(RAM) 또는 프로세싱 로직(2620)에 의한 실행을 위해 정보 및 명령어들을 저장할 수 있는 다른 타입의 동적 저장 디바이스를 포함할 수 있다. ROM(2640)은 ROM 디바이스 또는 프로세

싱 로직(2620)에 의한 이용을 위해 정적 정보 및/또는 명령어들을 저장할 수 있는 다른 타입의 정적 저장 디바이스를 포함할 수 있다. 저장 디바이스(2650)는 자기, 고체 상태, 및/또는 광학 레코딩 매체 및 그의 대응하는 드라이브, 또는 프로세싱 로직(2620)에 의한 이용을 위해 정보 및/또는 명령어들을 저장할 수 있는 다른 타입의 정적 저장 디바이스를 포함할 수 있다.

[0287] 입력 디바이스(2660)는 키보드, 마우스, 펜, 터치패드, 가속도계, 마이크로폰, 음성 인식, 카메라, 바이오메트릭 메커니즘들(biometric mechanisms) 등과 같이 운용자가 정보를 엔티티에 입력하도록 허용하는 로직을 포함할 수 있다. 일 실시예에서, 입력 디바이스(2660)는 입력 디바이스(240)에 대응할 수 있다.

[0288] 출력 디바이스(2670)는 디스플레이, 프린터, 스피커, 촉각 인터페이스(haptic interface) 등을 포함해서 정보를 운용자에게 출력하는 메커니즘을 포함할 수 있다. 통신 인터페이스(2680)는 엔티티가 다른 디바이스들 및/또는 시스템들과 통신하는 것을 가능하게 하는 임의의 트랜시버-유형 로직을 포함할 수 있다. 예를 들어, 통신 인터페이스(2680)는 네트워크를 통해 다른 디바이스 또는 시스템과 통신하기 위한 메커니즘들을 포함할 수 있다.

[0289] 도 26에서 도시되는 엔티티는 메인 메모리(2630)와 같은 컴퓨터-판독 가능한 저장 매체에 저장되는 소프트웨어 명령어들을 실행하는 프로세싱 로직(2620)에 응답하여 특정한 동작들을 수행할 수 있다. 컴퓨터-판독 가능한 저장 매체는 물리적(예를 들어, 유형의(tangible)) 또는 로컬 메모리 디바이스로서 정의될 수 있다. 소프트웨어 명령어들은 통신 인터페이스(2680)를 통해 저장 디바이스(2650)와 같은 다른 컴퓨터-판독 가능한 저장 매체로부터 또는 다른 디바이스로부터 메인 메모리(2630)내로 판독될 수 있다. 메인 메모리(2630)에 포함되는 소프트웨어 명령어들은 소프트웨어 명령어들이 프로세싱 로직 상에서 실행될 때 여기서 기술되는 기법들을 프로세싱 로직(2620)이 수행하게 할 수 있다. 대안적으로, 하드와이어 회로(hardwired circuitry)는 여기서 기술된 기법들을 구현하기 위해 소프트웨어 명령어들과 조합하여, 또는 그 대신에 이용될 수 있다. 따라서, 여기서 기술되는 구현들은 하드웨어 회로 및 소프트웨어의 임의의 특정한 조합으로 제한되지 않는다.

[0290] 도 26이 엔티티의 예시적인 컴포넌트들을 도시하지만, 다른 구현들에서, 엔티티는 도 26에서 도시된 것보다 적은, 또는 상이한, 또는 부가적인 컴포넌트들을 포함할 수 있다. 또 다른 구현들에서, 엔티티의 하나 이상의 컴포넌트들은 엔티티의 하나 이상의 다른 컴포넌트들에 의해 수행되는 것으로서 기술되는 하나 이상의 작업들을 수행할 수 있다.

[0291] **예시적인 분배된 실시예**

[0292] 분배되는 실시예들은 2개 이상의 프로세싱 자원들을 이용하여 프로세싱을 수행할 수 있다. 예를 들어, 실시예들은 단일 프로세싱 디바이스의 2개 이상의 코어들을 이용한 프로세싱, 단일의 인클로저 내에 설치된 다수의 프로세싱 디바이스를 통한 분배 프로세싱, 및/또는 네트워크에 의해 연결된 다수의 타입들의 프로세싱 로직을 통한 분배 프로세싱을 수행할 수 있다.

[0293] 도 27은 분배된 컴퓨팅 환경을 이용하여 클라이언트 디바이스(예를 들어, 컴퓨터(205)) 대신 비-선형 모델들에 대한 제어기들의 상호작용식 설계를 지원할 수 있는 예시적인 시스템이다. 시스템(2700)은 컴퓨터(205), 네트워크(2730), 서비스 제공자(2740), 원격 데이터베이스(2750) 및 클러스터(2760)를 포함할 수 있다. 도 27의 구현은 예시적이며 본 발명의 다른 분배된 구현들은 보다 많은 디바이스들 및/또는 엔티티들, 더 적은 디바이스들 및/또는 엔티티들, 및/또는 도 27의 예시적인 구성과 상이한 구성들의 디바이스들/엔티티들을 포함할 수 있다.

[0294] 컴퓨터(205)는 그래픽 사용자 인터페이스(GUI)(2710) 및 모델링 환경(220)을 포함할 수 있다. GUI(2710)는 사용자가 컴퓨터(205) 및/또는 원격 디바이스들(예를 들어, 서비스 제공자(2740))과 상호작용하도록 허용하는 인터페이스를 포함할 수 있다. 예시적인 실시예에서, GUI(2710)는 도 4 내지 도 6 및 도 15 내지 도 18의 인터페이스들과 유사할 수 있다.

[0295] 네트워크(2730)는 데이터(예를 들어, 패킷 데이터 또는 비-패킷 데이터)를 전달할 수 있는 임의의 네트워크를 포함할 수 있다. 네트워크(2730)의 구현들은 로컬 영역 네트워크들(local area networks; LAN들), 대도시 영역 네트워크들(metropolitan area networks; MAN들), 및/또는 인터넷 프로토콜(Internet protocol; IP), 비동기식 전달 모드(asynchronous transfer mode; ATM), 동기식 광학 네트워크(synchronous optical network; SONET), 사용자 데이터그램 프로토콜(user datagram protocol; UDP), IEEE 802.10 등과 같은 실제의 임의의 네트워크

프로토콜을 이용하여 동작할 수 있는, 인터넷과 같은 광역 네트워크들(wide area networks; WAN들)을 포함할 수 있다.

[0296] 네트워크(2730)는 라우터들, 스위치들, 방화벽들 및/또는 서버들(도시되지 않음)과 같은 네트워크 디바이스들을 포함할 수 있다. 네트워크(2730)는 유선 도체들 및/또는 광섬유들을 이용하는 하드와이어 네트워크일 수 있고 및/또는 자유-공간 옵티컬(free-space optical), 라디오 주파수(radio frequency; RF), 및/또는 음향 전송 경로들을 이용한 무선 네트워크일 수 있다. 구현에서, 네트워크(2730)는 인터넷과 같은 실질적으로 공개 공용 네트워크일 수 있다. 다른 구현에서, 네트워크(2730)는 기업체 가상 네트워크와 같은 보다 제한된 네트워크일 수 있다. 네트워크들 및/또는 여기서 기술된 네트워크 상에서 동작하는 디바이스들의 구현들은 임의의 특정한 데이터 타입, 프로토콜, 아키텍처/구성 등으로 제한되지 않는다. 예를 들어, 일 실시예에서, 네트워크(2730)는 퀀텀-호환 가능한 네트워킹 프로토콜들을 이용하는 퀀텀 네트워크(quantum network)일 수 있다.

[0297] 서비스 제공자(2740)는 서비스를 다른 디바이스가 이용 가능하게 하는 디바이스를 포함할 수 있다. 예를 들어, 서비스 제공자(2740)는 서버 및/또는 다른 디바이스들을 이용하여 하나 이상의 서비스들을 목적지(destination)에 제공하는 엔티티를 포함할 수 있다. 서비스들은 동작을 수행하기 위해 목적지에 의해 실행되는 명령어들을 포함할 수 있다. 대안적으로, 서비스는 목적지를 위해 동작을 수행하도록 목적지 대신 실행되는 명령어들을 포함할 수 있다.

[0298] 예를 위해, 컴퓨터(205)와 같이 하나 이상의 웹-기반 서비스들을 목적지에 제공하는 웹 서버를 서비스 제공자가 동작시킨다는 것을 가정한다. 웹-기반 서비스들은 서비스 제공자에 의해 동작되는 하드웨어를 이용하여 컴퓨터(205)가 전기적 및/또는 머신적 시스템들의 분배된 시뮬레이션을 수행하도록 허용할 수 있다. 예를 들어, 컴퓨터(205)의 이용자는 서비스 제공자의 하드웨어를 이용하여 시스템 모델들에 대한 PID 제어기들을 상호작용식으로 설계하도록 허용될 수 있다. 구현에서, 고객(사용자)은 구독 기반으로 서비스들을 수신할 수 있다. 구독은 월별 구독, 이용당 요금, 서비스 제공자(2740)와 고객 간에 교환된 정보에 양에 기초한 요금, 고객에 의해 이용된 프로세서 사이클들의 수에 기초한 요금, 고객에 의해 이용된 프로세서들의 수에 기초한 요금과 같은 어레인지먼트(arrangement)를 포함할 수 있다.

[0299] 원격 데이터베이스(2750)는 컴퓨터(205)와 같은 다른 디바이스들에 의한 이용을 위해 머신-판독 가능한 정보를 저장하는 디바이스를 포함할 수 있다. 일 실시예에서, 원격 데이터베이스(2750)는 시스템 모델들, 제어기들 등에 관한 정보를 포함하는 데이터 구조들을 저장하는 저장 디바이스들(예를 들어, 하드 디스크들, 광학 디스크들, 고체-상태 저장 디바이스들 등)의 어레이 또는 그리드를 포함할 수 있다.

[0300] 클러스터(2760)는 원격 프로세싱(예를 들어, 분배된 프로세싱, 병렬 프로세싱 등)을 수행하는데 이용될 수 있는 실행의 유닛들(2770)과 같은 프로세싱 디바이스들의 그룹을 포함할 수 있다. 실행의 유닛들(2770)은 컴퓨터(205)와 같이 요청 디바이스 대신 프로세싱 동작을 수행하는 하드웨어 및/또는 하드웨어/소프트웨어 기반 디바이스들을 포함할 수 있다. 일 실시예에서, 실행의 유닛들(2770)은 부분적인 결과를 각각 실행할 수 있고 부분적인 결과들은 모델에 대한 전체 결과내로 조합될 수 있다.

[0301] **예시적인 구현들**

[0302] 실시예는 컴퓨터 구현 방법 또는 비일시적인 컴퓨터 판독 가능한 매체들 상에 상주하는 머신 실행 가능한 명령어들을 포함할 수 있다. 방법 및/또는 명령어들은 컴퓨터 구현 방법이 자립형, 예를 들어, 로컬 머신, 또는 분배된 환경(예를 들어, 컴퓨팅 디바이스들의 클러스터 또는 그리드를 이용함)을 이용하여 실시될 수 있는 임의의 피드백 제어 구조에서 설계 파라미터들을 튜닝하기 위해 구현될 수 있다.

[0303] 예를 들어, 컴퓨터-구현 방법으로서 실시될 때, 방법은 튜닝되는 하나 이상의 자유 파라미터들을 갖는 하나 이상의 튜닝 가능한 컴포넌트들을 식별하는 것을 포함할 수 있다. 튜닝 가능한 컴포넌트들은 임의의 피드백 제어 구조의 텍스트 또는 그래픽 모델의 부분일 수 있다. 모델은 추가로 하나 이상의 튜닝 가능한 컴포넌트들을 포함하는 하나 이상의 피드백 루프들을 식별하는 것을 포함할 수 있다.

[0304] 방법은 추가로 튜닝 가능한 컴포넌트들을 이용하여 형성된 모음(collection) 및 임의의 피드백 제어 구조에서 알려진 그리고 고정된 컴포넌트들을 포함하는 럼핑된 선형 모델을 포함하는 표준 형태로 임의의 피드백 제어 구조를 변환하는 것을 포함할 수 있다. 튜닝 가능한 컴포넌트들로부터 형성되는 모음은 블록 대각 방식으로 그룹

핑되거나, 배열될 수 있다. 방법에서, 블록 대각 배열의 컴포넌트들은 블록 대각 배열의 개별 파라미터들 또는 파라미터들에서 액세스될 수 있으며, 이는 사용자에게 가시적이 되고, 알고리즘에 대해 이용 가능하게 되고 판독되거나 기록 등이 될 수 있다.

[0305] 방법은 설계 목적들 및/또는 설계 요건들을 표현하기 위해 H∞ 목적들 또는 제약들을 이용할 수 있다. 방법의 구현에서, H∞ 목적들 또는 제약들은 페루프 시스템에서 하나 이상의 점 대 점 전달 함수들와 관련된다. 방법은 추가로 튜닝 가능한 컴포넌트들을 파라미터화할 수 있다. 예를 들어, 방법은 튜닝 가능한 컴포넌트들을 통계적으로 파라미터화할 수 있거나, 또는 튜닝 가능한 컴포넌트들을 동적으로 파라미터화할 수 있다. 튜닝 가능한 컴포넌트들이 파라미터화되면, 방법은 이 파라미터화에 기초하여 튜닝 가능한 컴포넌트들의 자유 파라미터들(free parameters)과 상호작용할 수 있다.

[0306] 방법은 튜너(tuner)를 이용하여 피드백 제어 구조를 튜닝할 수 있다. 예를 들어, 일 구현에서, 방법은 비-스무스(non-smooth) H∞ 최적화 알고리즘들의 클래스 내에 있는 튜너를 이용하여 피드백 제어 구조를 튜닝할 수 있다. 튜너는 표준 형태로 동작할 수 있고 튜닝 가능한 컴포넌트들이 블록 대각 형태로 있을 때 튜닝 가능한 파라미터들 상에서 동작할 수 있다. 튜너는 추가로 H∞ 목적들을 최소화하고 및/또는 H∞ 제약들을 강제하도록 파라미터들을 튜닝할 수 있다.

[0307] 컴퓨터 구현 방법은 MATLAB-호환 가능한 환경과 같은 기술적 컴퓨팅 환경에서 구현될 수 있다. 방법은 추가로 Simulink-호환 가능한 환경 또는 Labview-호환 가능한 환경과 같은 텍스트 또는 그래픽 환경에서 구현될 수 있다. 구현들은 추가로 예를 들어, 튜닝 가능한 컴포넌트들, 고정된 컴포넌트들, 연결들, 알고리즘들 등을 포함할 수 있는 블록 세트들과 상호작용할 수 있다.

[0308] 컴퓨터 구현 방법은 추가로 임의의 피드백 제어 구조들의 상호작용식 설계 및 튜닝을 지원하는 방식으로 구현될 수 있다. 예를 들어, 사용자의 실시간 상호작용식 설계 및 튜닝 활동들을 지원하는 방법이 구현될 수 있다. 방법의 구현들은 FPGA들, ASIC들, ASIP들, PLD들, GPU들, DSP들, 다중-코어 디바이스들, 분배된 컴퓨팅 자원들 등을 이용하여 구현될 수 있다.

[0309] 방법의 구현들은 추가로 사용자 대신 다목적 또는 다요건 설계 작업들을 단순화하기 위해 복수의 점 대 점 페루프 전달 함수들 상의 별개의 목적들 또는 제약들을 사용자 또는 디바이스가 특정하도록 허용할 수 있다. 방법의 구현들은 사용자 또는 사용자들의 그룹에 의해 이용되는 종래의 작업흐름들을 지원함으로써 이를 행할 수 있다. 방법은 점 대 점 전달 함수들에 설계 요건들을 적용하는 것을 포함하도록 확대될 수 있으며, 여기서 상기 적용은 다목적 및 다요건 설계 작업들의 포뮬레이션을 용이하게 한다. 이 방법은 추가로 원할 때 객체 지향 프레임워크에서 전개될 수 있다.

[0310] 본 발명의 실시예들은 추가로 컴퓨터-구현 방법 동작들 및/또는 비일시적인 컴퓨터-판독 가능한 매체들 상에 상주하는 실행 가능한 명령어들을 포함할 수 있다. 예를 들어, 매체들/실행 가능한 명령어들로서 구현될 때, 매체들은 실행되면 본 발명의 실시예들 또는 구현을 실시하는 명령어들을 저장할 수 있다. 일 구현에서, 실행된 명령어들은 사용자 대신 본 발명의 기법 또는 방법을 수행할 수 있다. 일 구현에서, 매체들은 프로세서 상에서 실행되면 튜닝 가능한 컴포넌트들을 정적으로 특정하고, 튜닝 가능한 컴포넌트들을 동적으로 특정하고, 및/또는 튜닝 가능한 컴포넌트들의 파라미터들과 상호작용하기 위한 API를 구현하는 실행 가능한 명령어들을 보유할 수 있다. 일 구현에서, API는 객체-지향일 수 있다.

[0311] 실행 가능한 명령어들은 실행되면 컴포넌트들의 미리 정의된 세트에 대한 파라미터화들을 실현하는 미리 정의된 인터페이스들을 식별할 수 있다. 명령어들은 추가로 도우미 함수(helper function)들의 세트를 구현하고 및/또는 산술 연산들의 세트를 구현할 수 있다. 실행되는 명령어들은 추가로 산술 연산들 및 도우미 함수들을 이용하여 튜닝 가능한 컴포넌트들을 동적으로 생성할 수 있다. 일 구현에서, 동적으로 생성하는 것은 조합 기초 파라미터 컴포넌트들 및 고정된 계수들 또는 고정된 컴포넌트들을 포함할 수 있다.

[0312] 실행되는 명령어들은 추가로 튜닝 가능한 컴포넌트들의 파라미터적 모델을 생성할 수 있다. 일 구현에서, 파라미터적 모델은 튜닝 가능한 컴포넌트의 튜닝 가능한 파라미터들을 설명할 수 있고 사용자가 튜닝 가능한 파라미터들과 상호작용하도록 허용할 수 있다. 예를 들어, 사용자 입력들은 튜닝 가능한 파라미터들을 초기화하고 튜닝 가능한 파라미터를 고정(fix)시키고, 또는 튜닝 가능한 파라미터들 중 선택된 파라미터들을 해제할 수 있다.

[0313] 본 발명의 다른 실시예들은 하나 이상의 비일시적인 컴퓨터-판독 가능한 매체들 상에 상주하는 실행 가능한 명령어들을 통해 또는 컴퓨터-구현 방법으로서 구현될 수 있다. 실시예는 임의의 피드백 제어 구조들의 표준 형

태를 구축하기 위해 그리고 설계 요건들의 H∞ 포뮬레이션에 이용되는 점 대 점 전달 함수들을 특정하기 위해 애플리케이션 프로그램 인터페이스(application program interface; API)를 구현, 전개, 동작 등을 하게 할 수 있다. API의 실시예들은 원할 때 객체-기반이 될 수 있다.

[0314] 실시예들은 입력 신택스를 이용한 블록 도식화 연산들 및/또는 산술 연산들을 특정하는 사용자 입력 매커니즘과 상호작용할 수 있다. 예를 들어, 사용자 신택스는 사용자가 PID 제어기들과 같은 다른 타입들의 제어 구조들과 상호작용하기 위해 신택스를 이용한다는 점에서 사용자에게 친숙한 신택스일 수 있다. 입력 신택스는 사용자가 선형 시간 불변 모델 컴포넌트(linear time invariant model component)를 입력하도록 허용할 수 있으며, 여기서 선형 시간 불변 모델 컴포넌트는 선형 시간 불변 모델들에서 그리고 튜닝 가능한 파라미터들을 갖는 튜닝 가능한 컴포넌트들을 기술하는 입력 소프트웨어 기반 인터페이스들로서 이용된다.

[0315] 실시예들은 산술 연산자들 중 하나 이상 및 블록 도식화 연산들을 이용하여 소프트웨어 기반 인터페이스들과 선형 시간 불변 모델 컴포넌트를 조합할 수 있다. 블록 도식화 연산들의 예들은 직렬 연결, 병렬 연결, 또는 피드백 연결을 포함(그러나 이들로 제한되지 않음)할 수 있다. 실시예는 사용자 입력 매커니즘과 상호작용할 수 있고 이 상호작용은 입력 신택스가 전체 제어 시스템의 표준 형태를 점증적으로(incrementally) 구성하도록 허용할 수 있다. 표준 형태는 선형 시간 불변 모델 컴포넌트들 및 튜닝 가능한 컴포넌트들을 포함할 수 있다. 입력 신택스는 추가로 표준 형태에 기초하여 파라미터적 모델을 생성하는 것을 용이하게 하는데 이용될 수 있다. 파라미터적 모델은 최적화기와 호환 가능한 형태일 수 있으며, 여기서 호환 가능한 형태는 파라미터적 모델이 최적화기에 입력되도록 허용한다. 파라미터적 모델은 설계 요건들이 만족되도록 허용하는 방식으로 그것이 튜닝되도록 허용하는 형태로 셋업될 수 있다.

[0316] 실시예들은 추가로 H∞ 목적들을 최소화하고 및/또는 H∞ 제약들을 강제하도록 튜닝 가능한 파라미터들과 그리고 표준 형태와 상호작용하는 최적화기를 이용하여 파라미터적 모델을 최적화하는 것을 지원하도록 구성될 수 있다. 실시예들은 추가로 사용자 입력 메커니즘에 도우미 함수들을 제공하도록 구성될 수 있다. 실시예들은 사용자가 입력 신택스를 이용하여 API와 상호작용할 때 도우미 함수들이 입력 매커니즘을 통해 사용자에게 액세스될 수 있도록 허용할 수 있다. 여전히 추가로, 실시예는 입력 매커니즘과 함께 동작할 수 있고 입력 매커니즘과 호환 가능한 기능들을 제공하도록 구성될 수 있다. 기능들은 제어 시스템에 질의하도록 그리고/또는 제어 시스템을 분석하도록 이용될 수 있다.

[0317] 본 발명의 실시예는 루프 형상화 요건들을 H∞ 포뮬레이션내로 프로그램적으로 포뮬레이팅하기 위해 비일시적인 매체들 상에 상주할 수 있는 컴퓨터-구현 방법 및/또는 컴퓨터-실행 가능한 명령어들을 포함할 수 있다. 실시예는 방법으로서 실시될 때 자립형 또는 분배된 디바이스들을 이용하여 구현될 수 있고 하나 이상의 사용자들 대신 상호작용식(예를 들어, 실시간) 동작을 지원하도록 구성될 수 있다. 방법은 개루프 이득을 위한 타겟 형상, 또는 타겟 형상을 위한 프록시를 수신하는 것을 포함할 수 있다. 예를 들어, 타겟 루프 형상 또는 프록시는 사용자 대신 입력 매커니즘(예를 들어, 키보드, GUI 등)로부터 수신될 수 있거나, 또는 예를 들어, 컴퓨터-판독 가능한 저장 매체로부터 프로그램적으로 검색될 수 있다.

[0318] 방법은 타겟 루프 형상 또는 프록시로부터 제어 구조 및 필터들을 유도하는 소프트웨어 툴과 상호작용하는 것을 포함할 수 있다. 소프트웨어 툴은 추가로 유도된 제어 구조 및 필터들로부터 표준 형태를 구성하고 유도된 제어 구조 및 필터들로부터 H∞ 제약을 구성할 수 있다. 방법에서, 표준 형태 및 H∞ 제약은 제어 시스템에 대한 설계 요건들을 포착할 수 있다.

[0319] 부가적인 실시예는 표준 형태의 구조를 이용하기 위해 컴퓨터-판독 가능한 매체들 상에 상주하는 실행 가능한 명령어들 및/또는 컴퓨터-구현 방법을 포함할 수 있다. 표준 형태에 대한 구조를 이용하는 것은 사용자 대신 상호작용식 작업흐름들을 지원할 수 있는 튜너 알고리즘의 성능을 강화할 수 있다. 실시예는 최적화 프로시저 동안 최적화기에 의해 공급되는 튜닝 가능한 파라미터들의 값들을 수신할 수 있다. 최적화 프로시저들은 최적화 활동들을 수행하기 위한 머신 구현 기법들(machine implemented techniques)을 포함할 수 있다.

[0320] 실시예는 추가로 구성 프로시저/기법을 이용하여, 튜닝 가능한 파라미터들과 연관된 튜닝 가능한 컴포넌트들을 구현하기 위해 소프트웨어 객체들을 이용함으로써, 수신된 파라미터 값들에 대한 그의 상태-공간 표현의 제공을 책임지는 각각의 소프트웨어 객체를 형성함으로써, 튜닝 가능한 컴포넌트들에 대한 애그리게이팅된 상태-공간 매트릭스(aggregated state-space matrix)를 생성하기 위해 상태-공간 매트릭스들을 애그리게이팅함으로써, 그리고/또는 표준 형태의 원하는 폐루프 상태-공간 모델을 획득하기 위해 럼핑된 플랜트 모델의 상태-공간 표현과 튜닝 가능한 컴포넌트들의 애그리게이팅된 상태-공간 매트릭스들을 조합함으로써 튜닝 가능한 파라미터 값들에 대한 표준 형태의 상태-공간 모델을 구성할 수 있다.

[0321] 실시예는 후속적인 그래디언트 계산(gradient computation)들을 가속화하기 위해 구성 프로시저의 중간 결과들을 캐싱(caching)할 수 있다. 그래디언트 정보는 체인 규칙(chain rule)과 같은 규칙을 적용함으로써, 목적 및 제약들을 구분함으로써, 표준 형태의 상태-공간 모델을 구성하는데 이용되었던 동일한 소프트웨어 객체들을 이용함으로써, 수신된 파라미터들에 관해 스칼라값 함수(scalar-valued function)(스칼라값 함수는 규칙의 적용에 의한 부산물임)의 그래디언트의 제공을 책임지는 각각의 소프트웨어 객체를 형성함으로써, 및/또는 목적 및 제약들의 전체 그래디언트들 내로 각각의 튜닝 가능한 컴포넌트들에 의해 공급된 그래디언트 데이터를 효율적으로 애그리게이팅하기 위해 캐싱된 중간 결과들을 이용함으로써 계산될 수 있다.

[0322] 실시예는 프로세서 대신 부가적인 계산들을 요구하지 않고 그래디언트 정보를 이용 가능하게 할 수 있다. 예를 들어, 그래디언트 정보는 그래디언트 정보를 이용 가능하게 하지 않고 초래된 계산 비용에 비교하면 부가되지 않은 계산 비용으로 이용 가능하게 될 수 있다. 실시예는 추가로 실질적으로 계산 비용 없이 튜닝 가능한 파라미터들의 정해진 값에 대한 폐루프 시스템을 유도할 수 있는데, 예를 들어, 비용은 비용을 고려하는 것이 시스템 설계, 모델링 시간들 등과 연관된 계산 예산들에 악영향을 주지 않다는 점에서 무시할 정도다(negligible).

[0323] **결론**

[0324] 구현들은 사용자에게 친숙한 특성들을 이용하여 사용자가 시스템 모델들에 대한 제어기들을 상호작용식으로 설계하도록 허용할 수 있다.

[0325] 본 발명의 예시적인 실시예들의 앞선 설명은 예시 및 설명을 제공하지만, 소모적(exhaustive)이거나 개시된 바로 그 형태로 본 발명을 제한하도록 의도되는 것은 아니다. 수정들 및 변동들은 위의 교시에 비추어서 가능하거나 본 발명의 실시로부터 획득될 수 있다. 예를 들어, 일련의 동작들이 도 25에 관하여 기술되었지만, 동작들의 순서는 본 발명의 원리와 일치하는 다른 구현들에서 수정될 수 있다. 추가로, 비-의존적 동작들이 병렬로 수행될 수 있다.

[0326] 또한, 본 발명의 원리들과 일치하는 구현들은 본 발명의 사상으로부터 벗어남 없이 도면들에서 예시되고 명세서에서 기술되는 것들 이외의 다른 디바이스들 및 구성들을 이용하여 구현될 수 있다. 디바이스들 및/또는 컴포넌트들은 특정한 전개들 및/또는 응용들에 의존하여 도 2, 도 3, 도 26 또는 도 27의 구현들로부터 제거되거나 및/또는 부가될 수 있다. 또한, 개시되는 구현들은 임의의 특정한 하드웨어 조합으로 제한되지 않을 수 있다.

[0327] 또한, 본 발명의 특정한 부분들은 하나 이상의 기능들을 수행하는 "로직"으로서 구현될 수 있다. 이 로직은 하드와이어 로직과 같은 하드웨어, 주문형 집적 회로, 필드 프로그래밍 가능한 게이트 어레이, 마이크로프로세서, 또는 하드웨어 및 소프트웨어의 조합을 포함할 수 있다. 본 발명의 설명에서 이용되지 않은 엘리먼트, 동작 또는 명령어들은 그러한 것으로 명시적으로 기술되지 않으면 본 발명에 필수적이거나 임계적인 것으로서 해석돼선 안 된다. 또한, 여기서 이용된 바와 같이, 단수 표현은 하나 이상의 아이템들을 포함하도록 의도된다. 단지 하나의 아이템만이 의도되는 경우 용어 "하나" 또는 유사한 언어가 이용된다. 추가로, 여기서 이용된 바와 같은 구문 "~에 기초하는"은 달리 명시적으로 언급되지 않으면 "~에 적어도 부분적으로 기초하는"을 의미하도록 의도된다.

[0328] 여기서 이용된 표제들 및 부제들은 명세서를 서브섹션(subsection)들로 분할함으로써 독자를 돕고자 하는 것이다. 이들 표제들 및 부제들은 본 발명의 특징들을 정의하는 것으로서 또는 본 발명의 범위를 제한하는 것으로서 해석되지 않는다.

부호의 설명

[0329]

205 컴퓨터	210 획득 로직
215 운영체제	220 모델링 환경
230 모델	240 입력 디바이스(들)
250 디스플레이 디바이스	260 모델 표현
270 플랜트	310 시뮬레이션 툴

320 엔티티 라이브러리 330 인터페이스 로직

340 컴파일러 350 제어 로직

360 최적화기 370 시뮬레이션 엔진

380 리포트 엔진 390 코드 생성기

720 구조화된 제어기 2620 프로세싱 로직

2630 메모리 2650 저장 디바이스

2660 입력 디바이스 2670 출력 디바이스

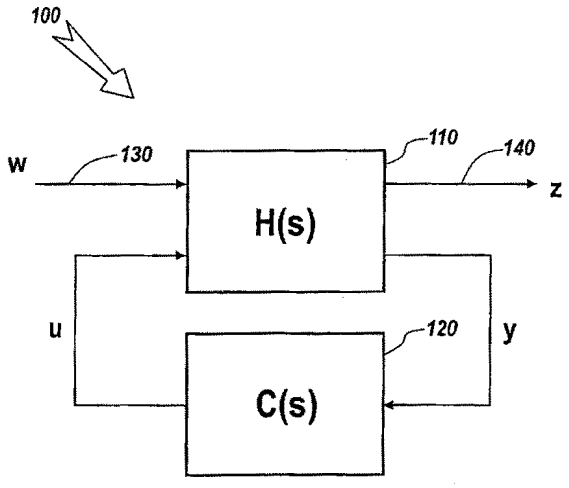
2680 통신 인터페이스 2720 컴퓨팅 환경

2730 네트워크 2740 서비스 제공자

2750 원격 데이터베이스 2760 클러스터

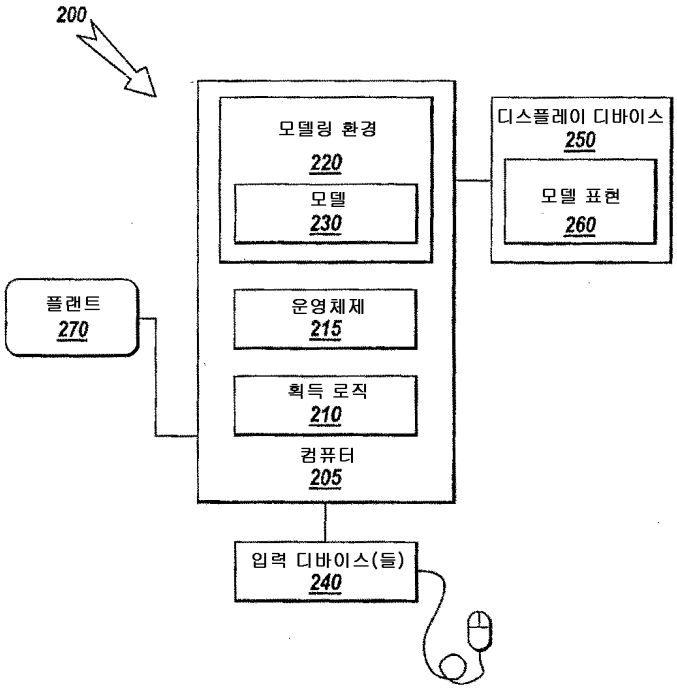
도면

도면1

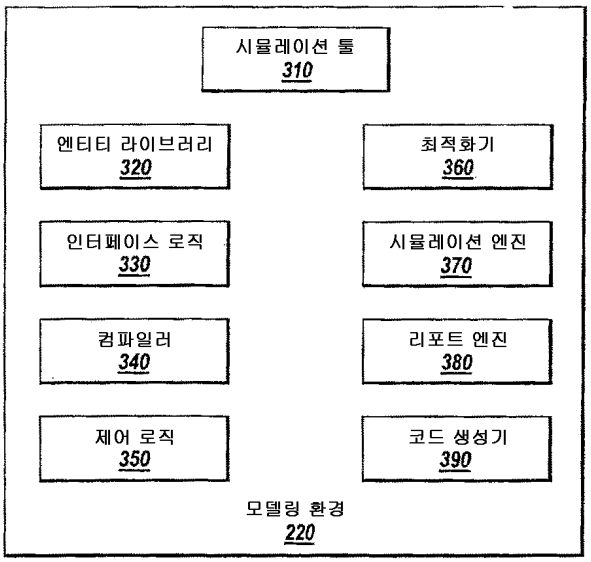


(종래 기술)

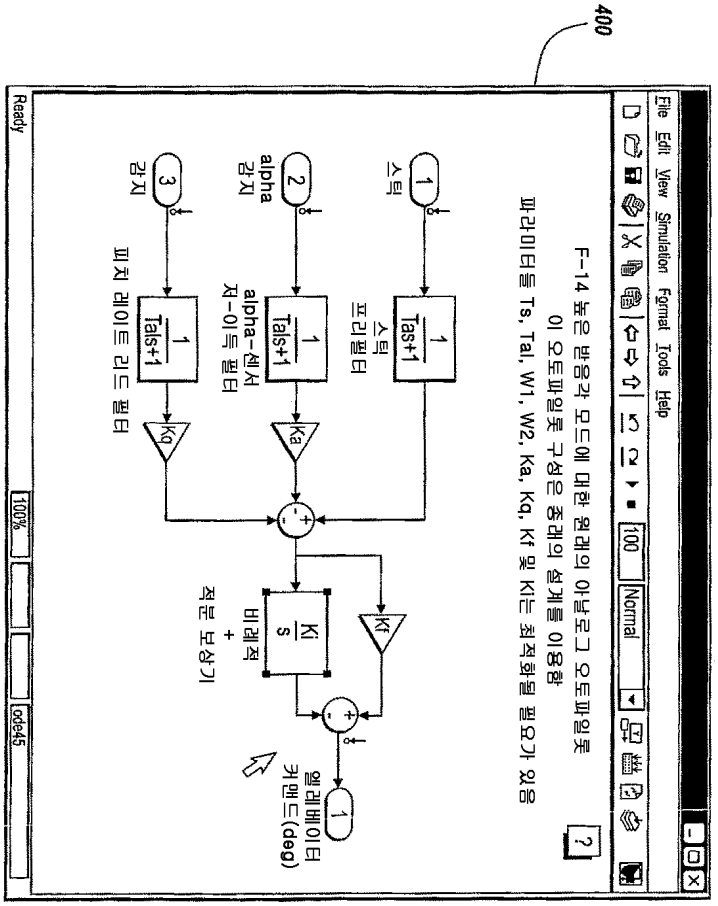
도면2


200
모델링 환경
220
모델
230
디스플레이 디바이스
250
모델 표현
260
플랜트
270
운영체제
215
획득 로직
210
컴퓨터
205
입력 디바이스(들)
240

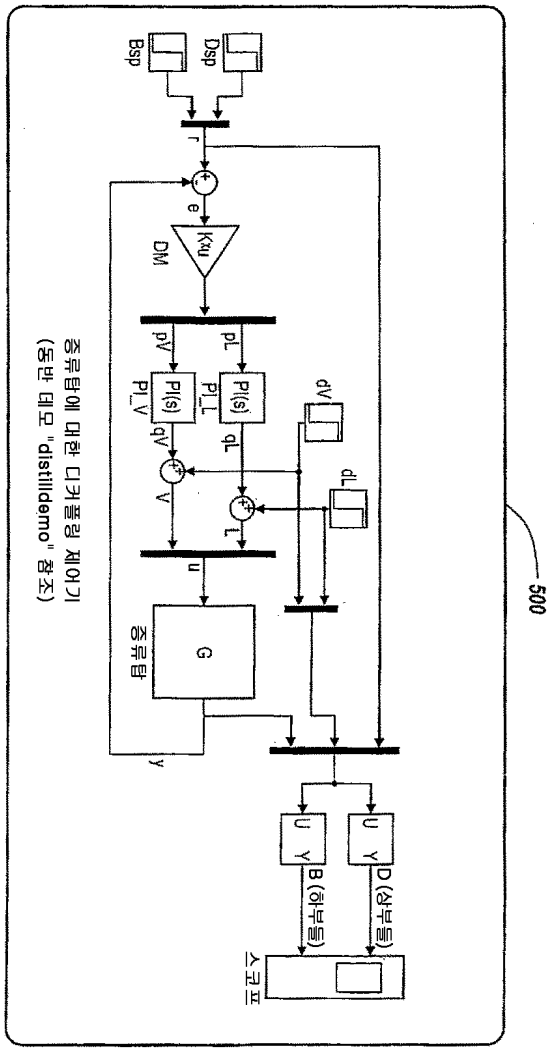
도면3


시뮬레이션 툴
310
엔티티 라이브러리
320
최적화기
360
인터페이스 로직
330
시뮬레이션 엔진
370
컴파일러
340
리포트 엔진
380
제어 로직
350
코드 생성기
390
모델링 환경
220

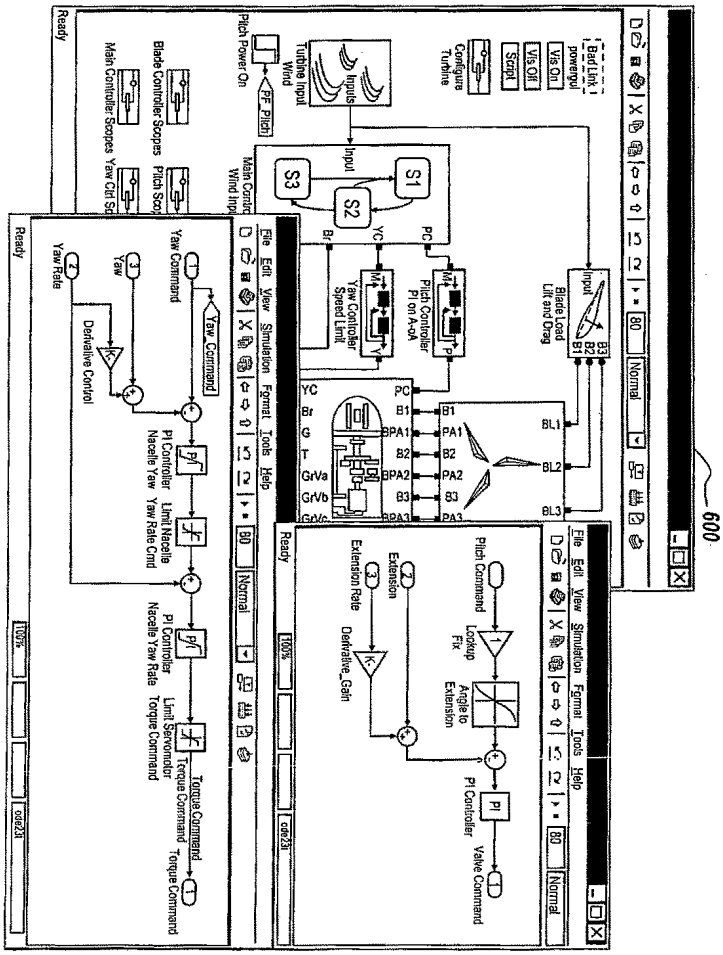
도면4


400
File Edit View Simulation Format Tools Help
100
Normal
F-14 높은 받음각 모드에 대한 원래의 아날로그 오토파일롯
이 오토파일롯 구성은 종래의 설계를 이용함
파라미터들 Ts, Tal, W1, W2, Ka, Kq, Kf 및 Ki는 최적화될 필요가 있음
1
스틱
2
alpha
감지
3
감지
1/(Tas+1)
스틱
프리필터
1/(Tals+1)
alpha-센서
자-이득 필터
1/(Tals+1)
피치 레이트 리드 필터
Ka
Kq
Kf
Ki/s
비례적
+
적분 보상기
엘레베이터
커맨드(deg)
Ready
100%
ode45

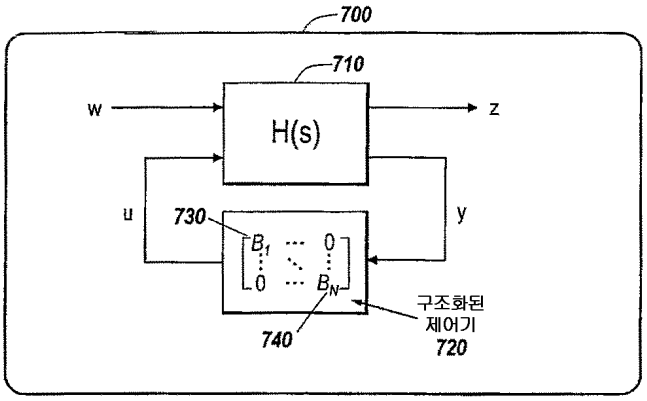
도면5


증류탑에 대한 디커플링 제어기
(동반 데모 "distilldemo" 참조)
Bsp
Dsp
r
e
Kxu
DM
pV
pL
PI(s)
PI_V
PI_L
qV
qL
V
L
dV
dL
u
증류탑
G
y
U
Y
B (하부플)
D (상부플)
스코프
500

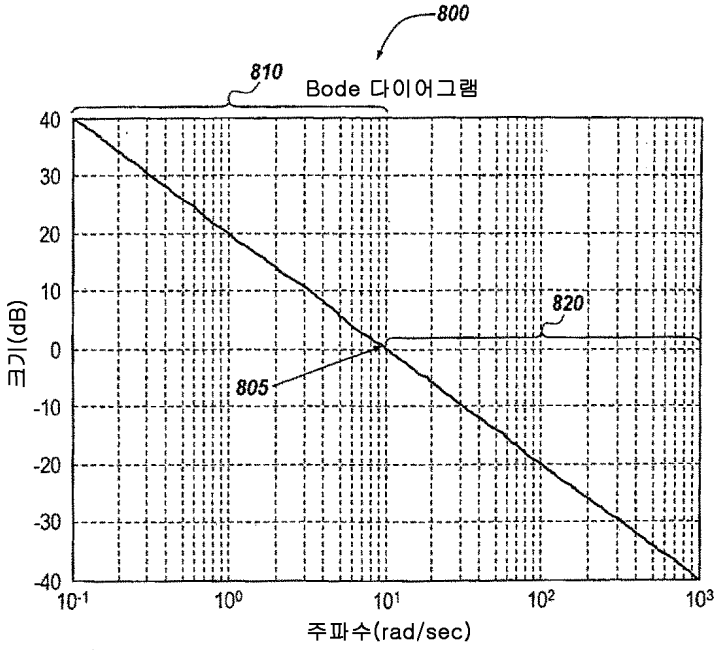
도면6


600
Ready
Pitch Power On
RF_Pitch
Turbine Input Wind
Inputs
Configure Turbine
Bad Link ! powerpu
Vis On
Vis Off
Script
Blade Controller Scopes
Pitch Scop
Main Controller Scopes
Yaw Ctrl Sc
Main Controller Wind Inpu
Input
S1
S2
S3
Br
YC
PC
Yaw Controller Speed Limit
Pitch Controller PI on A-oA
Blade Load Lift and Drag
B3
B2
B1
BL1
BL2
BL3
BPA1
BPA2
B3
PA1
PA2
PA3
GrVa
GrVb
GrVc
File Edit View Simulation Format Tools Help
Normal
Yaw Rate
Yaw
Yaw Command
Derivative Control
Yaw_Command
PI Controller Nacelle Yaw
Limit Nacelle Yaw Rate Cmd
PI Controller Nacelle Yaw Rate
Limit Servomotor Torque Command
Torque Command
Extension Rate
Extension
Derivative_Gain
Pitch Command
Lookup Fix
Angle to Extension
PI Controller
Valve Command
100%
ode23t

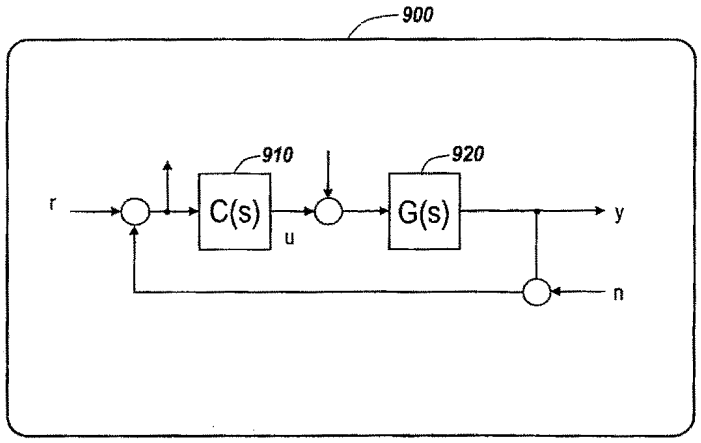
도면7


700
710
w
H(s)
z
u
730
y
B_1
B_N
0
구조화된 제어기
720
740

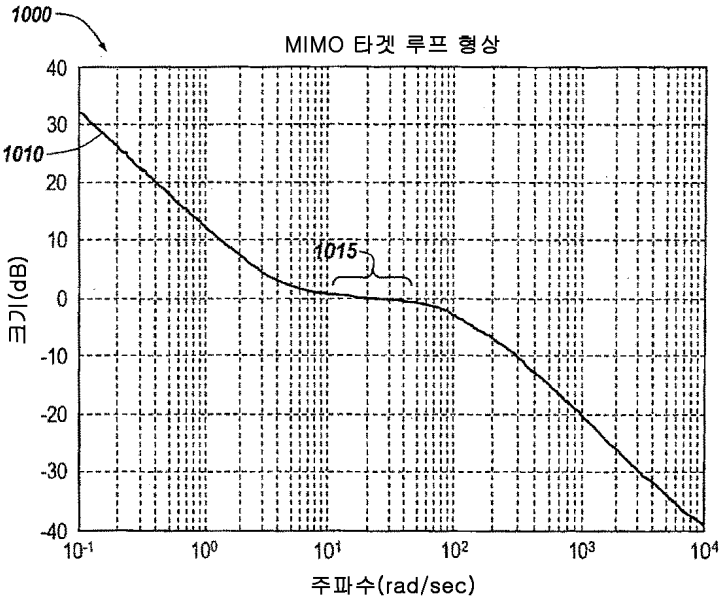
도면8


800
810
Bode 다이어그램
820
805
크기(dB)
주파수(rad/sec)

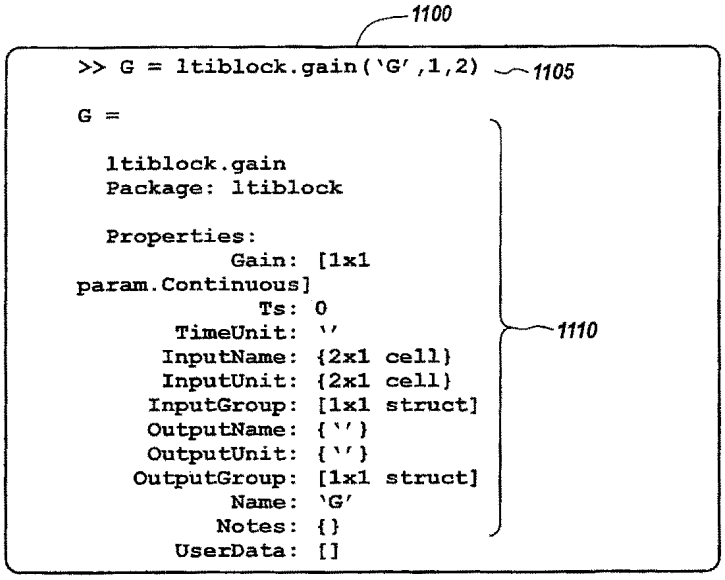
도면9


900
910
920
C(s)
G(s)
r
u
y
n

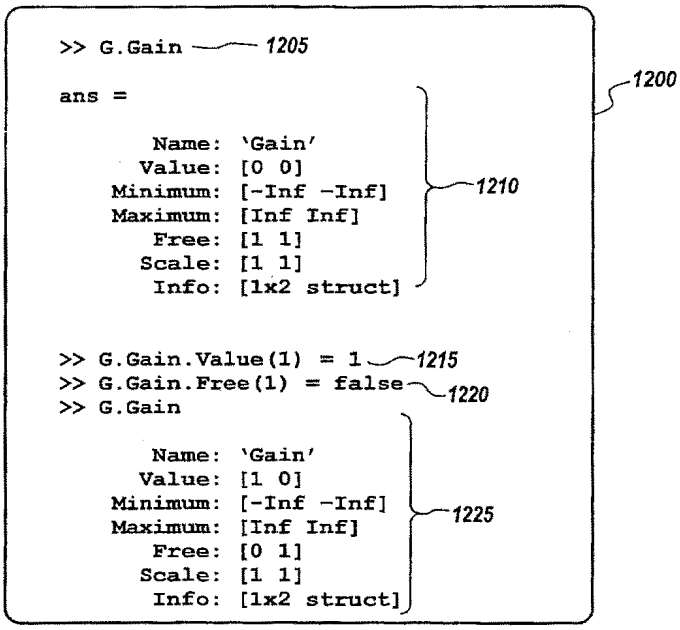
도면10


1000
MIMO 타겟 루프 형상
1010
1015
크기(dB)
40
30
20
10
0
-10
-20
-30
-40
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
주파수(rad/sec)

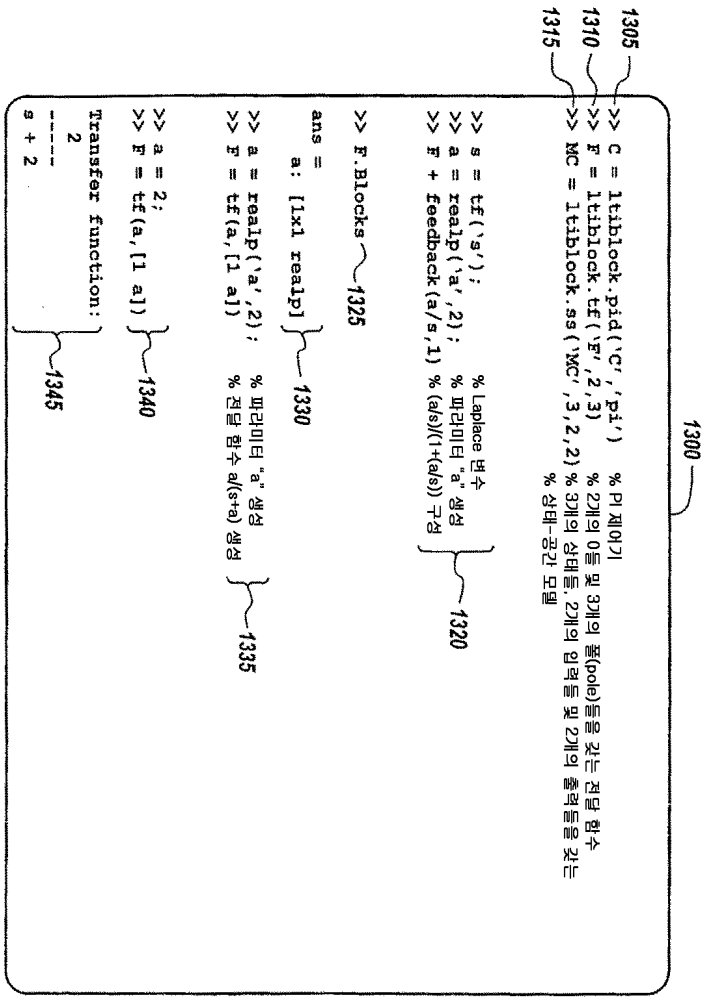
도면11


1100
>> G = ltiblock.gain('G',1,2)
1105
G =
ltiblock.gain
Package: ltiblock
Properties:
Gain: [1x1 param.Continuous]
Ts: 0
TimeUnit: ''
InputName: {2x1 cell}
InputUnit: {2x1 cell}
InputGroup: [1x1 struct]
OutputName: {''}
OutputUnit: {''}
OutputGroup: [1x1 struct]
Name: 'G'
Notes: {}
UserData: []
1110

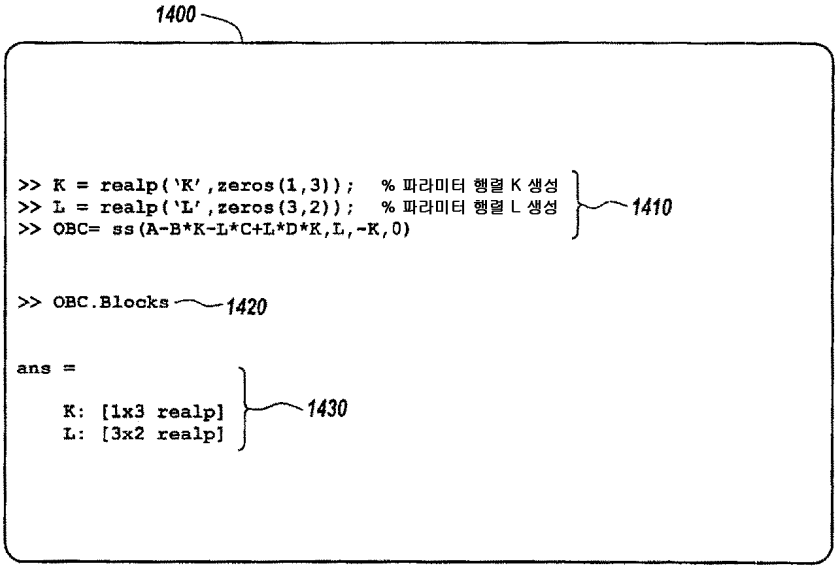
도면12


>> G.Gain
1205
ans =
Name: 'Gain'
Value: [0 0]
Minimum: [-Inf -Inf]
Maximum: [Inf Inf]
Free: [1 1]
Scale: [1 1]
Info: [1x2 struct]
1210
1200
>> G.Gain.Value(1) = 1
1215
>> G.Gain.Free(1) = false
1220
>> G.Gain
Name: 'Gain'
Value: [1 0]
Minimum: [-Inf -Inf]
Maximum: [Inf Inf]
Free: [0 1]
Scale: [1 1]
Info: [1x2 struct]
1225

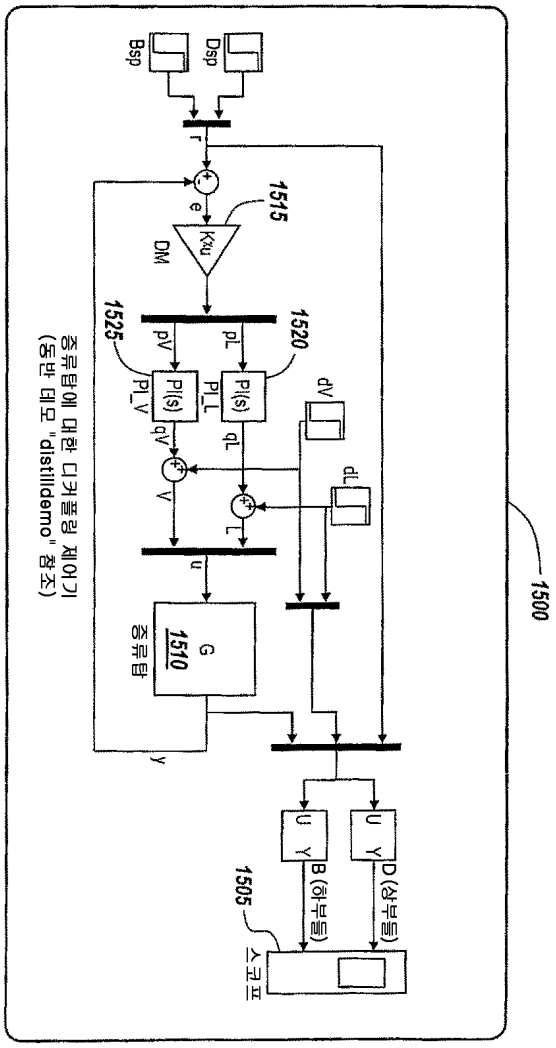
도면13


1305
1310
1315
>> C = ltiblock.pid('C','pi') % PI 제어기
>> F = ltiblock.tf('F',2,3) % 2개의 0들 및 3개의 폴(pole)들을 갖는 전달 함수
>> MC = ltiblock.ss('MC',3,2,2) % 3개의 상태들, 2개의 입력들 및 2개의 출력들을 갖는 % 상태-공간 모델
1300
>> s = tf('s'); % Laplace 변수
>> a = realp('a',2); % 파라미터 "a" 생성
>> F + feedback(a/s,1) % (a/s)/(1+(a/s)) 구성
1320
>> F.Blocks
1325
ans =
a: [1x1 realp]
1330
>> a = realp('a',2); % 파라미터 "a" 생성
>> F = tf(a,[1 a]) % 전달 함수 a/(s+a) 생성
1335
>> a = 2;
>> F = tf(a,[1 a])
1340
Transfer function:
2
-----
s + 2
1345

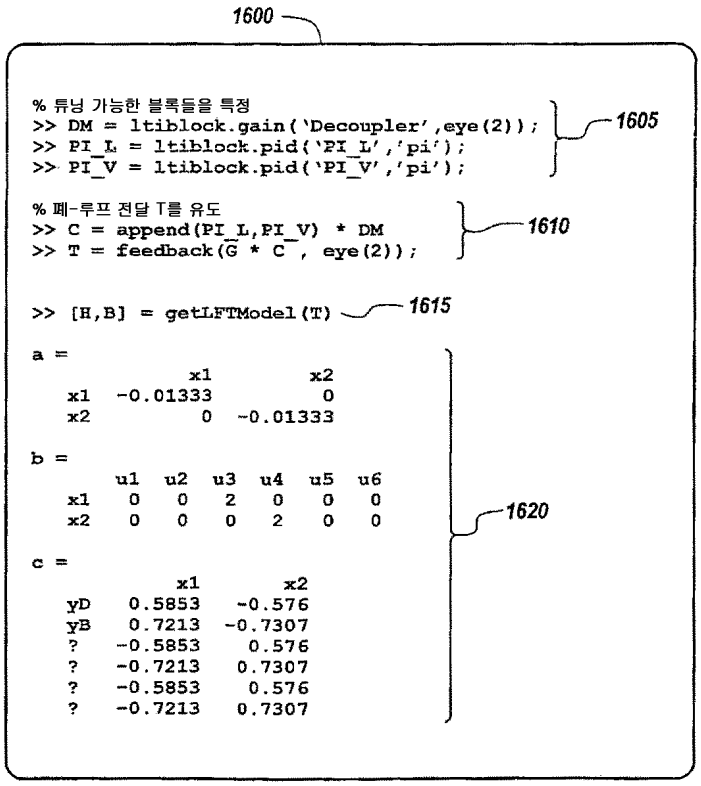
도면14


1400
>> K = realp('K',zeros(1,3)); % 파라미터 행렬 K 생성
>> L = realp('L',zeros(3,2)); % 파라미터 행렬 L 생성
>> OBC= ss(A-B*K-L*C+L*D*K,L,-K,0)
1410
>> OBC.Blocks
1420
ans =
K: [1x3 realp]
L: [3x2 realp]
1430

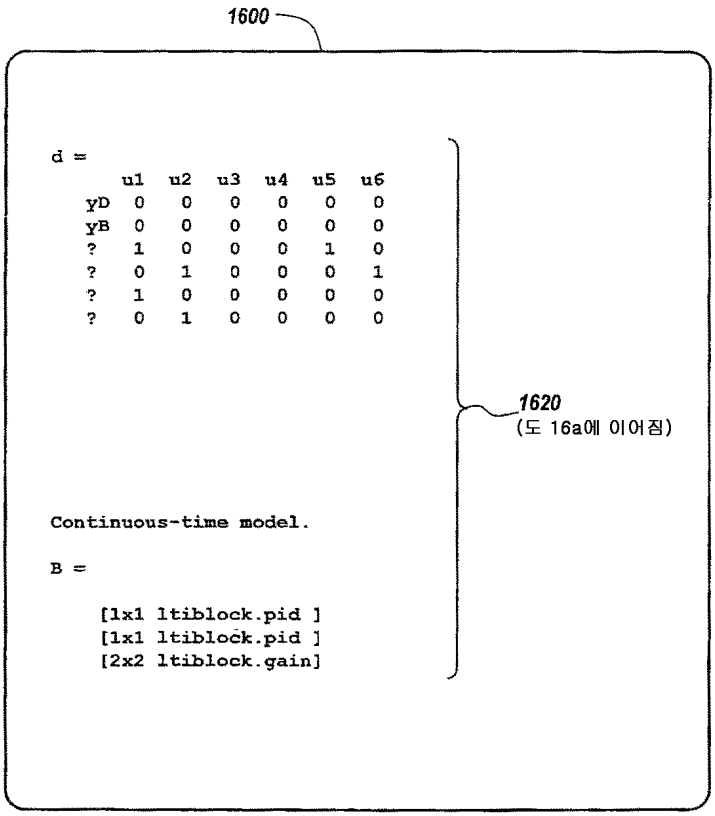
도면15


중류탑에 대한 디커플링 제어기
(동반 데모 "distildemo" 참조)
Bsp
Dsp
r
e
DM
Kxu
1515
1525
1520
pV
pL
PI(s)
PI_V
PI_L
dV
dL
qV
qL
V
L
u
G
1510
증류탑
y
U
Y
B(하부들)
D(상부들)
1505
스코프
1500

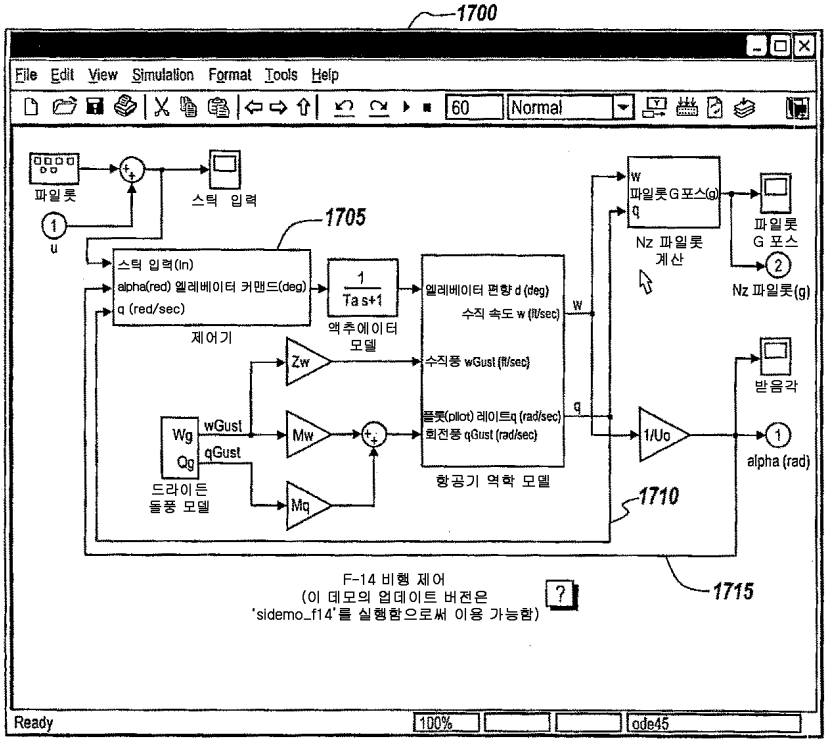
도면16a


1600
% 튜닝 가능한 블록들을 특정
>> DM = ltiblock.gain('Decoupler',eye(2));
>> PI_L = ltiblock.pid('PI_L','pi');
>> PI_V = ltiblock.pid('PI_V','pi');
1605
% 폐-루프 전달 T를 유도
>> C = append(PI_L,PI_V) * DM
>> T = feedback(G * C , eye(2));
1610
>> [H,B] = getLFTModel(T)
1615
a =
x1 x2
x1 -0.01333 0
x2 0 -0.01333
b =
u1 u2 u3 u4 u5 u6
x1 0 0 2 0 0 0
x2 0 0 0 2 0 0
c =
x1 x2
yD 0.5853 -0.576
yB 0.7213 -0.7307
? -0.5853 0.576
? -0.7213 0.7307
? -0.5853 0.576
? -0.7213 0.7307
1620

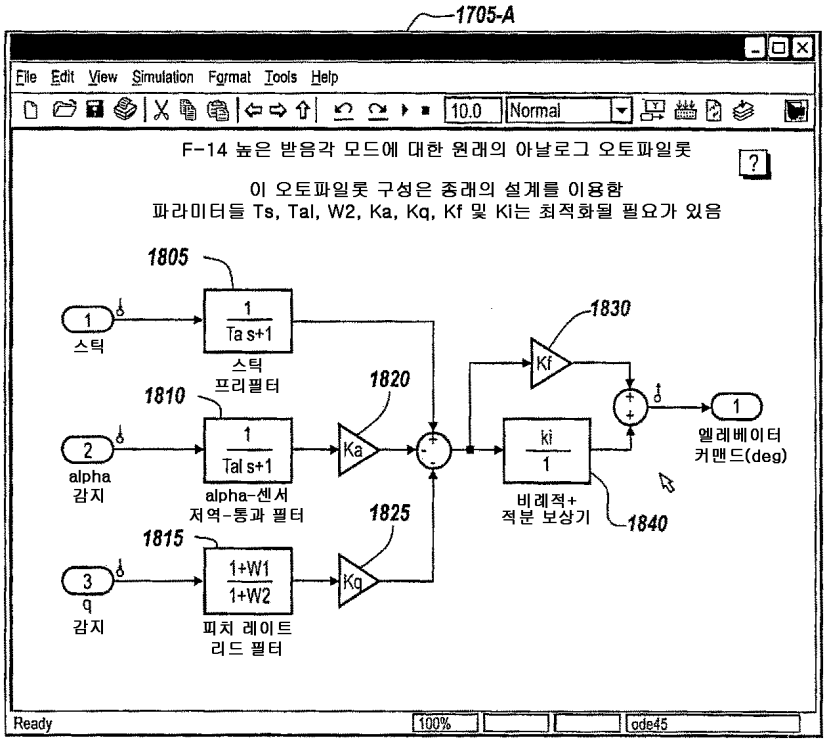
도면16b


1600
d =
u1 u2 u3 u4 u5 u6
yD 0 0 0 0 0 0
yB 0 0 0 0 0 0
? 1 0 0 0 1 0
? 0 1 0 0 0 1
? 1 0 0 0 0 0
? 0 1 0 0 0 0
1620
(도 16a에 이어짐)
Continuous-time model.
B =
[1x1 ltiblock.pid]
[1x1 ltiblock.pid]
[2x2 ltiblock.gain]

도면17


1700
File Edit View Simulation Format Tools Help
60
Normal
파일롯
u
스틱 입력
1705
스틱 입력(in)
alpha(red) 엘레베이터 커맨드(deg)
q (red/sec)
제어기
1
Ta s+1
액추에이터 모델
엘레베이터 편향 d (deg)
수직 속도 w (ft/sec)
수직풍 wGust (ft/sec)
롤릿(pilot) 레이트q (rad/sec)
회전풍 qGust (rad/sec)
항공기 역학 모델
w
q
Zw
Mw
Mq
Wg
Qg
wGust
qGust
드라이든 돌풍 모델
w
파일롯G포스(g)
q
Nz 파일롯 계산
파일롯 G 포스
Nz 파일롯(g)
받음각
1/Uo
alpha (rad)
1710
1715
F-14 비행 제어
(이 데모의 업데이트 버전은
'sidemo_f14'를 실행함으로써 이용 가능함)
Ready
100%
ode45

도면18

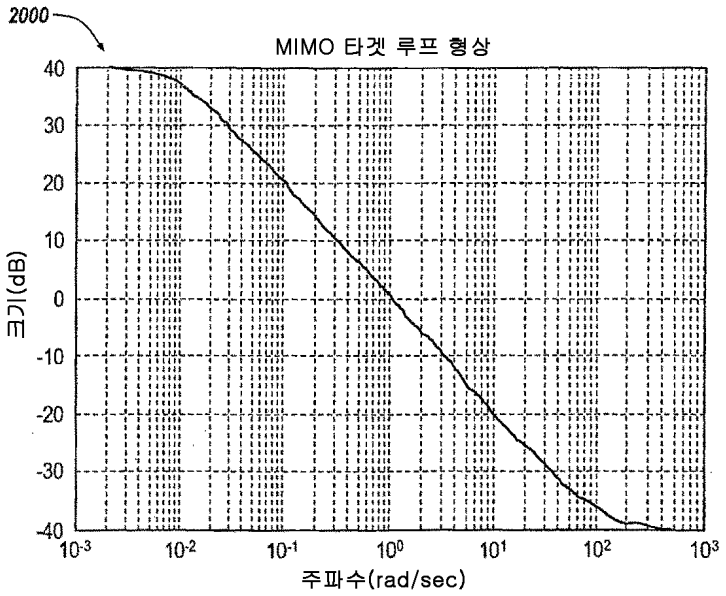

1705-A
File Edit View Simulation Format Tools Help
10.0
Normal
F-14 높은 받음각 모드에 대한 원래의 아날로그 오토파일롯
이 오토파일롯 구성은 종래의 설계를 이용함
파라미터들 Ts, Tal, W2, Ka, Kq, Kf 및 Ki는 최적화될 필요가 있음
1805
1
스틱
1
Ta s+1
스틱 프리필터
1810
2
alpha 감지
1
Tal s+1
alpha-센서 저역-통과 필터
1820
Ka
1815
3
q 감지
1+W1
1+W2
피치 레이트 리드 필터
1825
Kq
1830
Kf
ki
1
비례적+ 적분 보상기
1840
1
엘레베이터 커맨드(deg)
Ready
100%
ode45

도면19

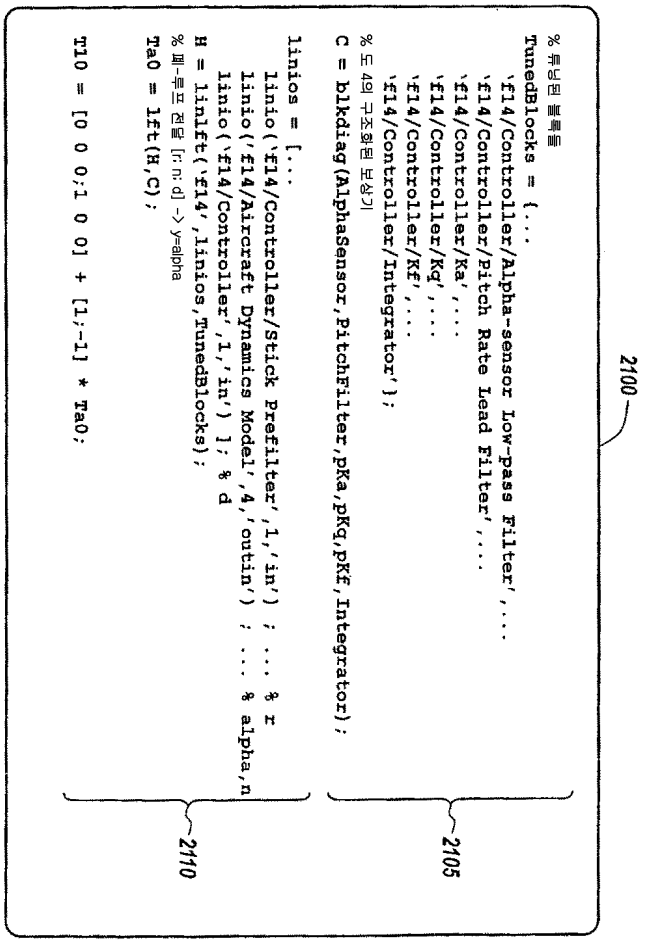
```
s = tf('s')
pKa = realp('Ka',0);
pKq = realp('Kq',0);
pKf = realp('Kf',0);
pKi = realp('ki',1);
pTal = realp('Tal',1);
pW1 = realp('W1',0);
pW2 = realp('W2',0);

Integrator = pKi/s;                    % Ki/s
AlphaSensor = tf(1,[pTal 1]);          % 1/(Tal*s+1)
PitchFilter = tf([1 pW1],[1 pW2]);     % (s+W1)/(s+W2)

wc = 1;     % 타겟 크로스오버
LS = (1+0.01*s/wc)/(0.01+s/wc);
bodemag(LS,{1e-3,1e3}), grid, title('Target loop shape')
```

1905 1910 1915

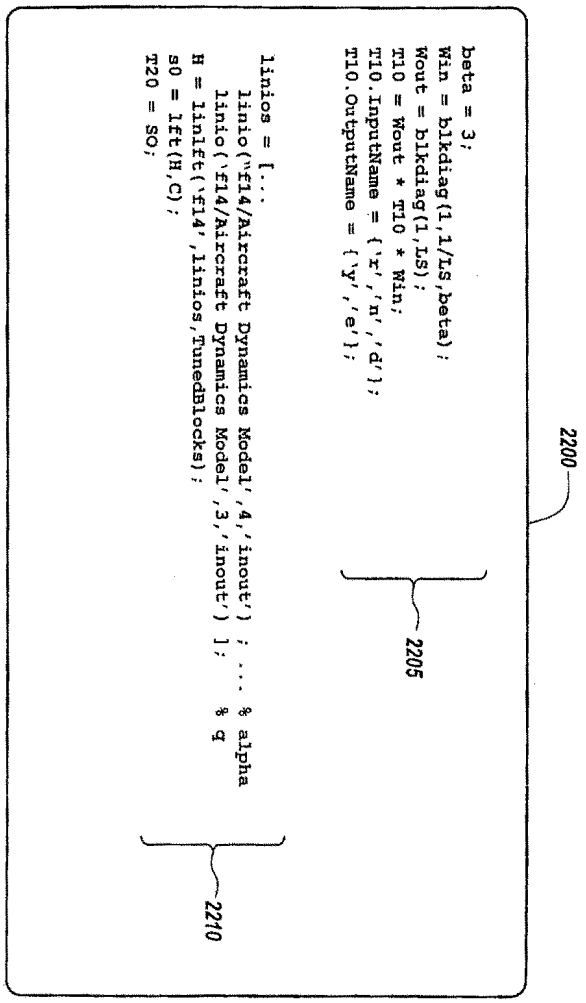
도면20



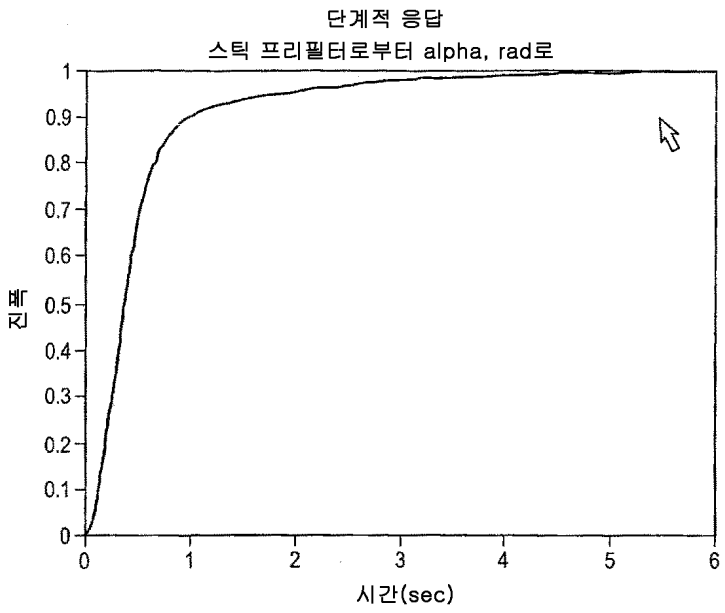
도면21


% 튜닝된 블록들
TunedBlocks = {...
'f14/Controller/Alpha-sensor Low-pass Filter',...
'f14/Controller/Pitch Rate Lead Filter',...
'f14/Controller/Ka',....
'f14/Controller/Kq',....
'f14/Controller/Kf',....
'f14/Controller/Integrator'};
% 도 4의 구조화된 보상기
C = blkdiag(AlphaSensor,PitchFilter,pKa,pKq,pKf,Integrator);
linios = [...
linio('f14/Controller/Stick Prefilter',1,'in') ; ... % r
linio('f14/Aircraft Dynamics Model',4,'outin') ; ... % alpha,n
linio('f14/Controller',1,'in')]; % d
H = linlft('f14',linios,TunedBlocks);
% 폐-루프 전달 [r; n; d] -> y=alpha
Ta0 = lft(H,C);
T10 = [0 0 0;1 0 0] + [1;-1] * Ta0;
2100
2105
2110

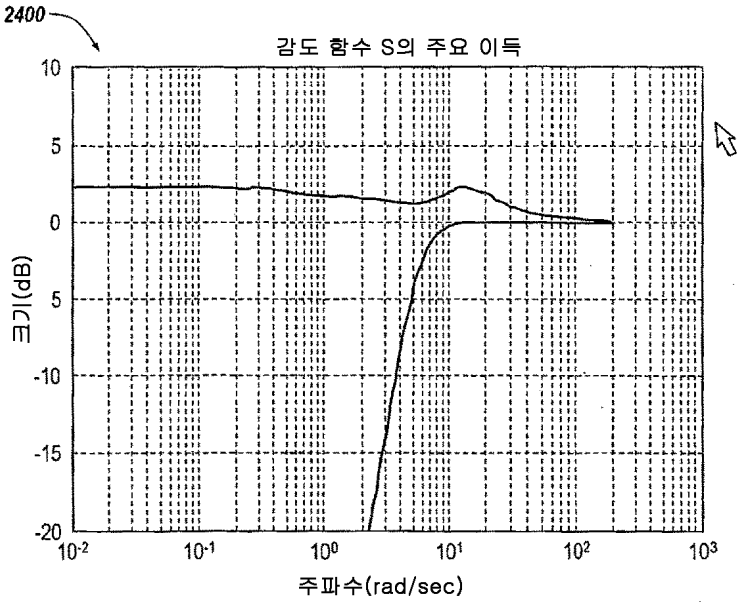
도면22


beta = 3;
Win = blkdiag(1,1/LS,beta);
Wout = blkdiag(1,LS);
T10 = Wout * T10 * Win;
T10.InputName = {'x','n','d'};
T10.OutputName = {'y','e'};
2205
2200
linios = [...
linio('f14/Aircraft Dynamics Model',4,'inout') ; ... % alpha
linio('f14/Aircraft Dynamics Model',3,'inout')]; % q
H = linlft('f14',linios,TunedBlocks);
s0 = lft(H,C);
T20 = S0;
2210

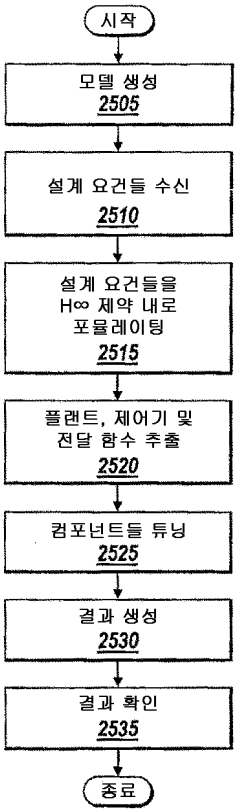
도면23


단계적 응답
스틱 프리필터로부터 alpha, rad로
진폭
시간(sec)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
1
2
3
4
5
6

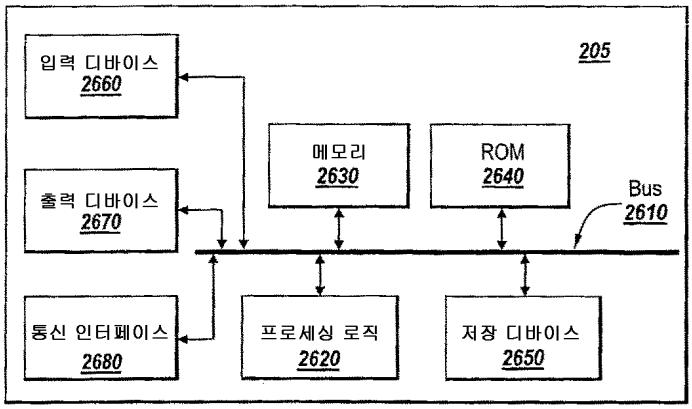
도면24


2400
감도 함수 S의 주요 이득
10
5
0
5
-10
-15
-20
크기(dB)
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
주파수(rad/sec)

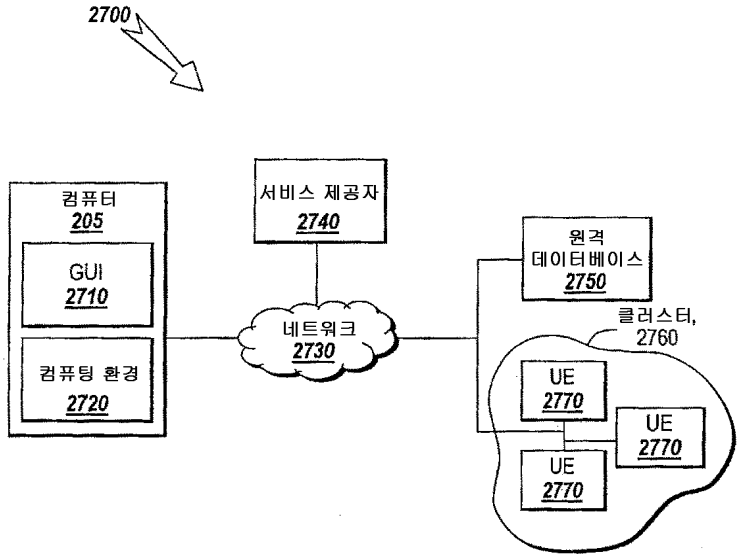
도면25


시작
모델 생성
2505
설계 요건들 수신
2510
설계 요건들을
H∞ 제약 내로
포뮬레이팅
2515
플랜트, 제어기 및
전달 함수 추출
2520
컴포넌트들 튜닝
2525
결과 생성
2530
결과 확인
2535
종료

도면26


205
입력 디바이스
2660
메모리
2630
ROM
2640
출력 디바이스
2670
Bus
2610
통신 인터페이스
2680
프로세싱 로직
2620
저장 디바이스
2650

도면27


2700
컴퓨터
205
GUI
2710
컴퓨팅 환경
2720
서비스 제공자
2740
네트워크
2730
원격
데이터베이스
2750
클러스터,
2760
UE
2770
UE
2770
UE
2770