US 20210381712A1

(19) United States
(12) Patent Application Publication
Harvey et al.

(10) Pub. No.: US 2021/0381712 A1
(43) Pub. Date: Dec. 9, 2021

(54) DETERMINING DEMAND CURVES FROM COMFORT CURVES

(71) Applicant: **PassiveLogic, Inc.**, Salt Lake City, UT (US)

(72) Inventors: **Troy Aaron Harvey**, Brighton, UT (US); **Jeremy David Fillingim**, Salt Lake City, UT (US)

(21) Appl. No.: **17/177,391**

(22) Filed: **Feb. 17, 2021**

Related U.S. Application Data

(60) Provisional application No. 62/704,976, filed on Jun. 5, 2020.

Publication Classification

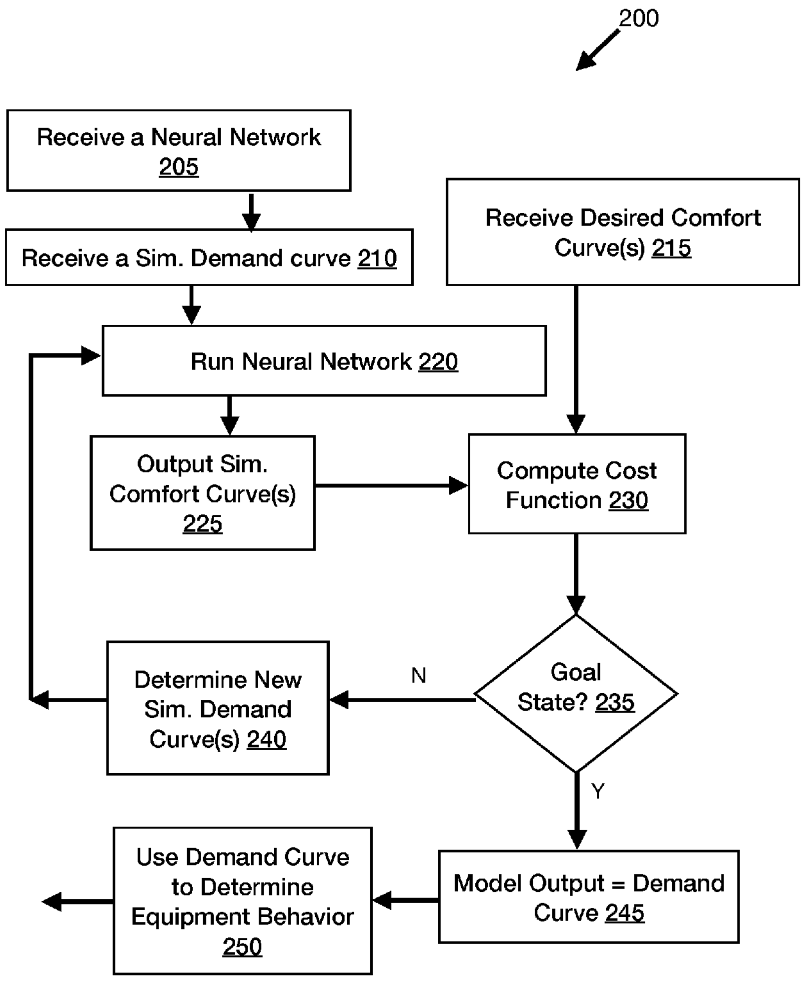
(51) **Int. Cl.**

F24F 11/65	(2006.01)
G06Q 10/06	(2006.01)
G06Q 50/06	(2006.01)
G06Q 50/16	(2006.01)
G05B 13/02	(2006.01)
G06N 3/04	(2006.01)
G06N 3/08	(2006.01)

(52) **U.S. Cl.**
CPC ***F24F 11/65*** (2018.01); ***G06Q 10/067*** (2013.01); ***G06Q 10/06313*** (2013.01); ***G06N 3/084*** (2013.01); ***G06Q 50/163*** (2013.01); ***G05B 13/027*** (2013.01); ***G06N 3/0472*** (2013.01); ***G06Q 50/06*** (2013.01)

(57) **ABSTRACT**

The amount of state over time (demand curves) that needs to be injected into a structure over time to achieve desired state values over time (desired comfort curves) at locations are determined by using a neural network that models the structure. Possibly random demand curves are fed into the neural network model at areas, such as the outside, state source locations (such as heaters), and are fed forward though the model, diffusing the state throughout the model. Comfort curves at chosen locations within the neural net representing physical locations are output. The comfort curves are compared with the desired comfort curves using cost function. Machine-learning methods are used to incrementally improve the demand curves until the output comfort curves are sufficiently close the desired state values.


200
Receive a Neural Network 205
Receive a Sim. Demand curve 210
Receive Desired Comfort Curve(s) 215
Run Neural Network 220
Output Sim. Comfort Curve(s) 225
Compute Cost Function 230
Goal State? 235
N
Y
Determine New Sim. Demand Curve(s) 240
Model Output = Demand Curve 245
Use Demand Curve to Determine Equipment Behavior 250

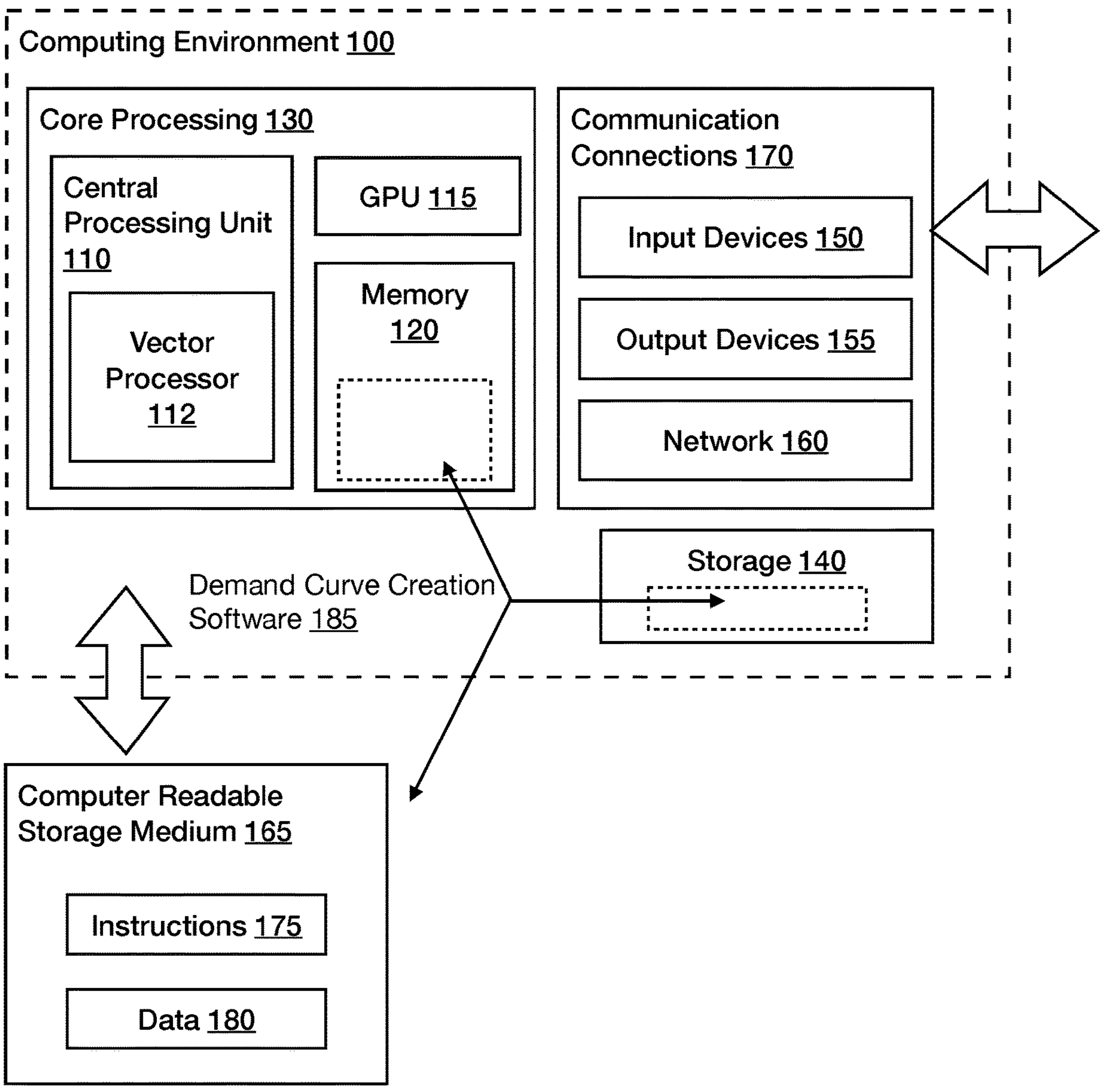

Computing Environment 100
Core Processing 130
Central Processing Unit 110
Vector Processor 112
GPU 115
Memory 120
Communication Connections 170
Input Devices 150
Output Devices 155
Network 160
Storage 140
Demand Curve Creation Software 185
Computer Readable Storage Medium 165
Instructions 175
Data 180

FIG. 1

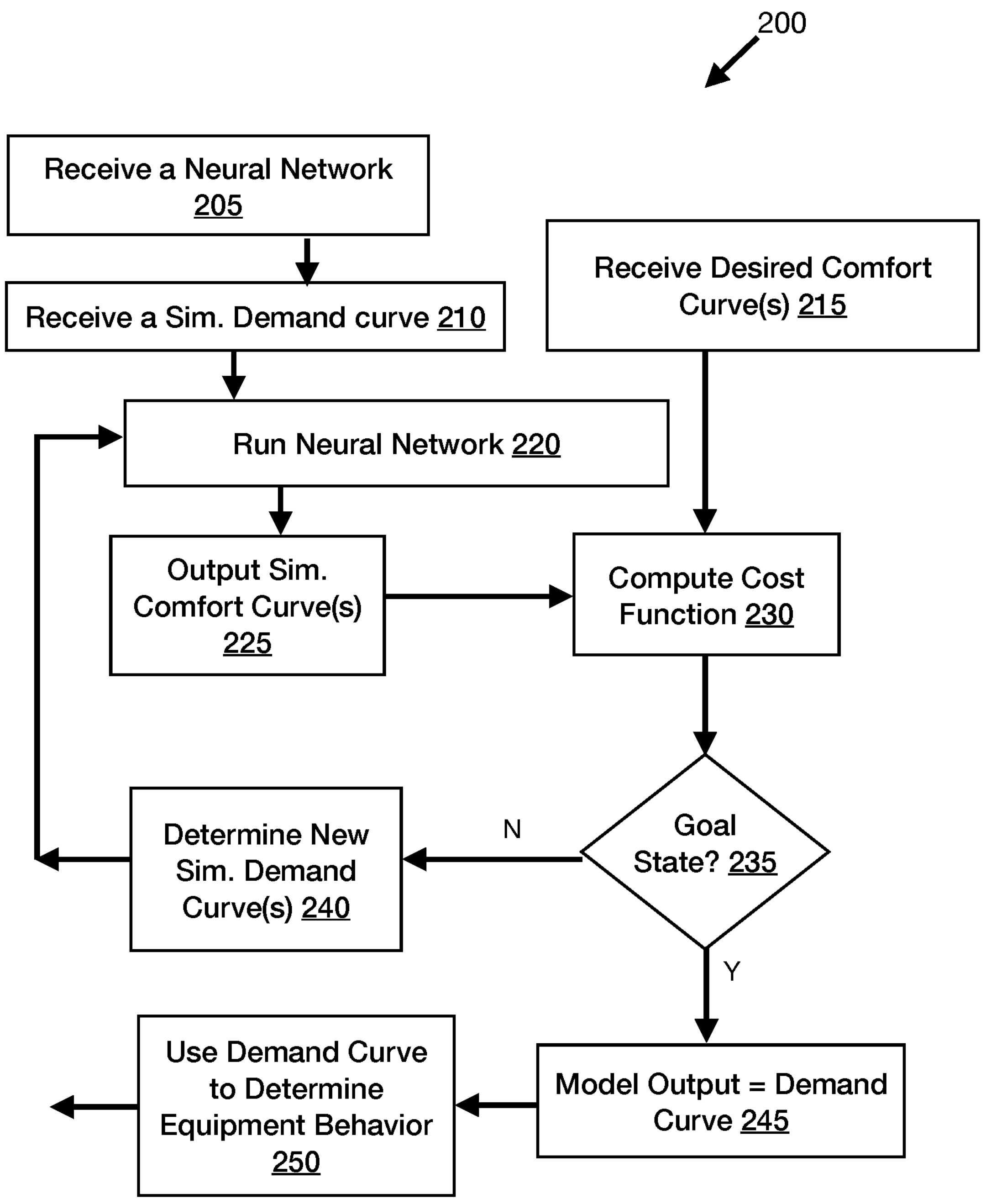

200
Receive a Neural Network 205
Receive a Sim. Demand curve 210
Receive Desired Comfort Curve(s) 215
Run Neural Network 220
Output Sim. Comfort Curve(s) 225
Compute Cost Function 230
Goal State? 235
N
Determine New Sim. Demand Curve(s) 240
Y
Model Output = Demand Curve 245
Use Demand Curve to Determine Equipment Behavior 250

FIG. 2

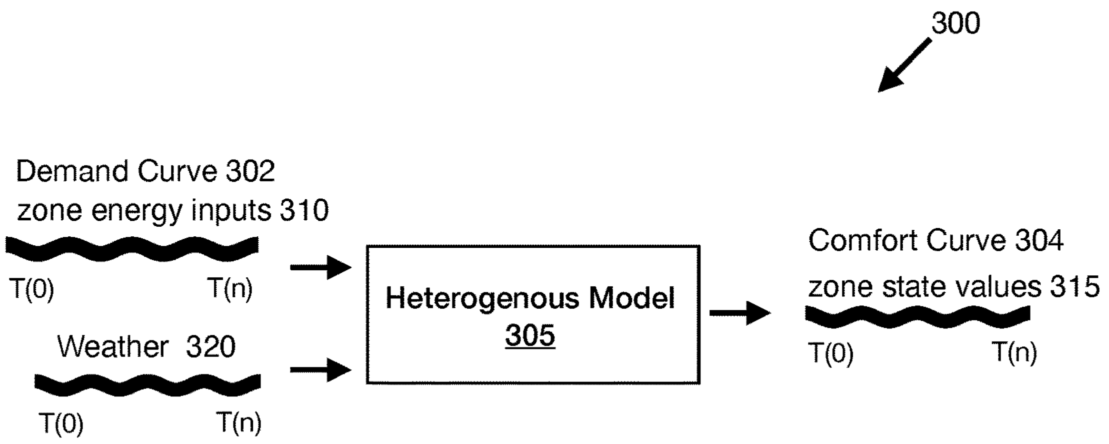

300
Demand Curve 302
zone energy inputs 310
T(0)
T(n)
Weather 320
T(0)
T(n)
Heterogenous Model 305
Comfort Curve 304
zone state values 315
T(0)
T(n)

FIG. 3

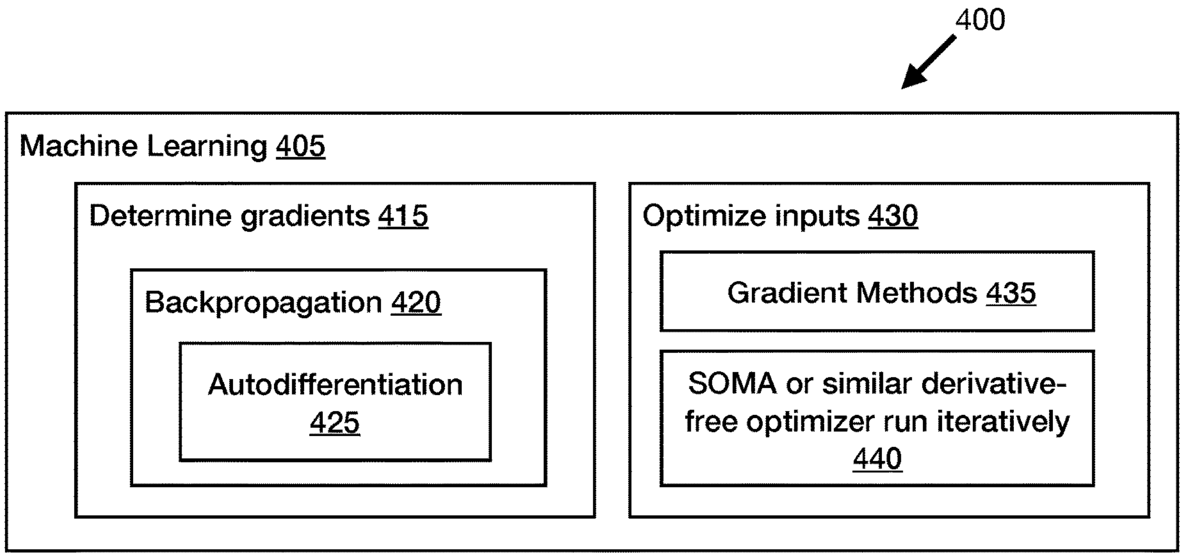

400
Machine Learning 405
Determine gradients 415
Backpropagation 420
Autodifferentiation 425
Optimize inputs 430
Gradient Methods 435
SOMA or similar derivative-free optimizer run iteratively 440

FIG. 4

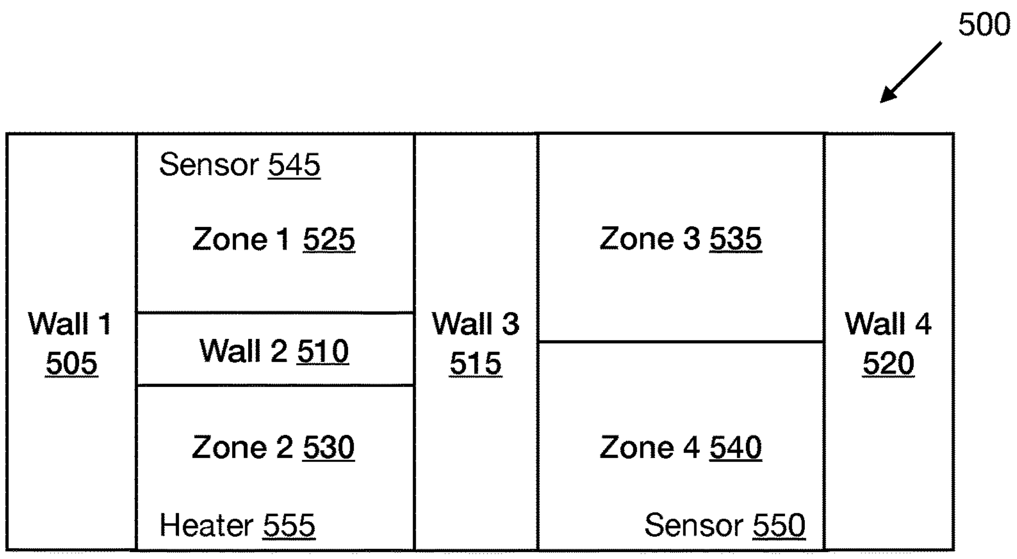

500
Sensor 545
Zone 1 525
Wall 1 505
Wall 2 510
Wall 3 515
Zone 3 535
Wall 4 520
Zone 2 530
Zone 4 540
Heater 555
Sensor 550

FIG. 5

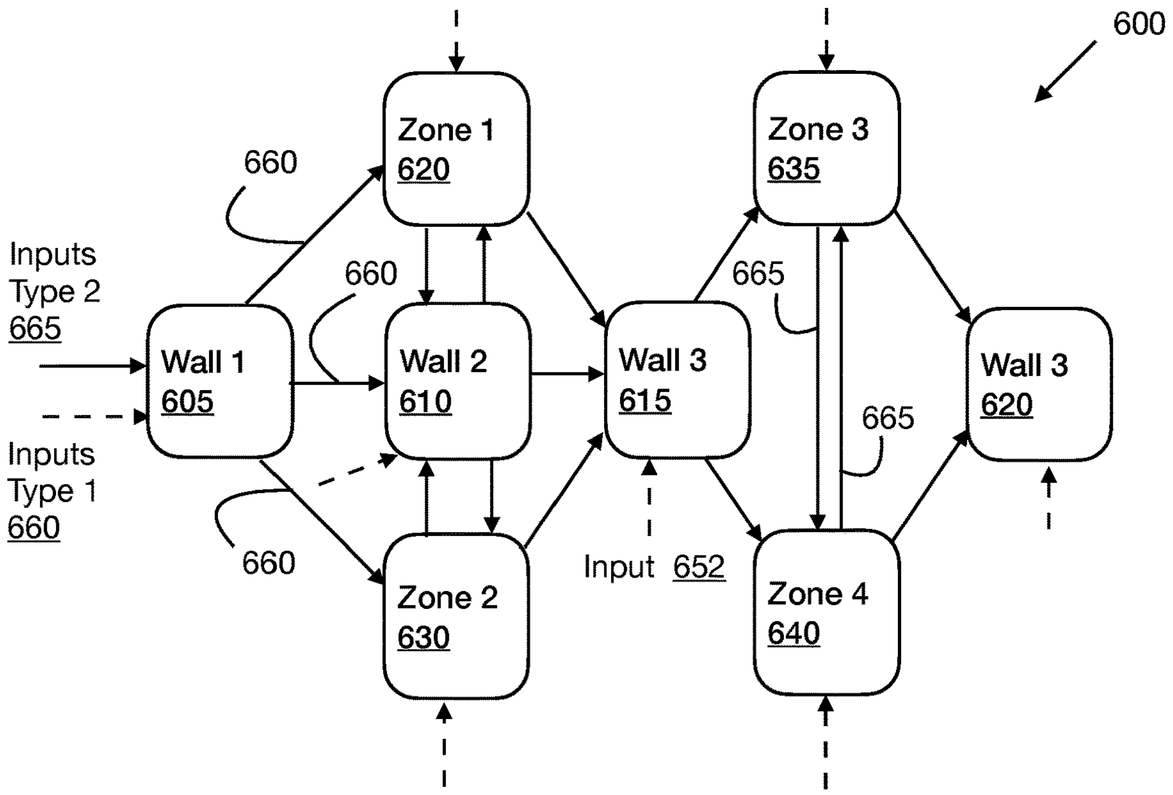

600
660
Zone 1 620
Zone 3 635
Inputs Type 2 665
660
665
Wall 1 605
Wall 2 610
Wall 3 615
Wall 3 620
665
Inputs Type 1 660
660
Input 652
Zone 2 630
Zone 4 640

FIG. 6

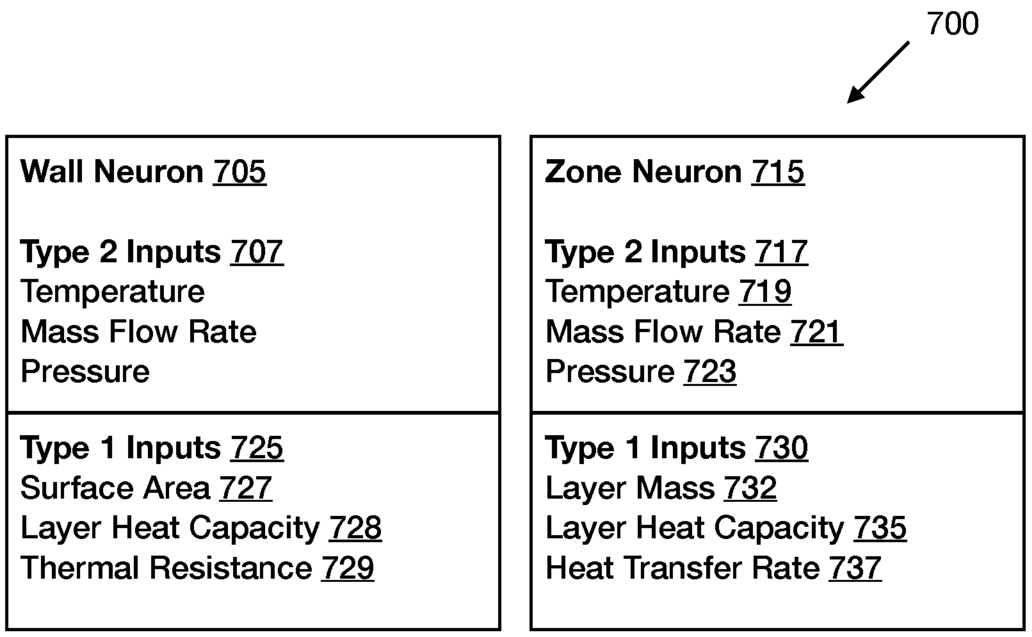

700
Wall Neuron 705
Type 2 Inputs 707
Temperature
Mass Flow Rate
Pressure
Type 1 Inputs 725
Surface Area 727
Layer Heat Capacity 728
Thermal Resistance 729
Zone Neuron 715
Type 2 Inputs 717
Temperature 719
Mass Flow Rate 721
Pressure 723
Type 1 Inputs 730
Layer Mass 732
Layer Heat Capacity 735
Heat Transfer Rate 737

FIG. 7

DETERMINING DEMAND CURVES FROM COMFORT CURVES

RELATED APPLICATION

[0001] The present application hereby incorporates by reference the entirety of, and claims priority to, U.S. provisional patent application Ser. No. 62/704,976 filed 5 Jun. 2020.

[0002] The present application hereby incorporates by reference U.S. utility patent application Ser. No. 17/009,713, filed Sep. 1, 2020.

FIELD

[0003] The present disclosure relates to neural network methods for creating demand curves from comfort curves. More specifically the present disclosure relates to receiving a time curve of desired state values and outputting a time curve of energy amounts that may be input into a structure to achieve the desired state values.

BACKGROUND

[0004] Current building automation systems rely on current state of the building and simple requirements to determine building behavior. For example, a building may know that at 7:00 am it should be at 70°. At 7:00 am it will then check what the current temperature is and turn on the heater to warm the building up to the desired temperature or turn on the air conditioner to cool it down. Not only is this simplistic, but it leaves buildings unable to adapt to predictable events, such as when weather reports predict changes; when the number of people in the building is known to change at a certain time, and so on. Trying to model buildings quickly runs into problems, as even simple buildings are very complex in terms of the current controllers that are used to manage the systems in buildings. Proportional-Integral-Derivative controllers (PID controllers)—originally designed for ship steering in 1922—are widely used to control HVAC and other systems in building, but fit very poorly into creating models that have more than a single setpoint. To model a room heterogeneously, you would need roughly 50 PID controllers; why so many? The walls are made of multiple materials that transfer state differently, and there are four walls, typically; the ceiling and floor are made of different levels of materials, forces act on the outside of the walls; there are heat sources, such as people and lights in the room, all of which together make up the building.

SUMMARY

[0005] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description section. This summary does not identify required or essential features of the claimed subject matter.

[0006] In an embodiment, a method of determining a demand curve implemented by one or more computers is disclosed, comprising: receiving a neural network of a plurality of controlled building zones; receiving a desired comfort curve for at least one of the plurality of controlled building zones; performing a machine learning process to run the heterogenous model using a simulated demand curve as input and receiving a simulated comfort curve as output; computing a cost function using the simulated comfort curve and the desired comfort curve; using the cost function to determine a new simulated demand curve; iteratively performing the using and computing steps until a goal state is reached; and determining that the new simulated demand curve is the demand curve upon the goal state being reached.

[0007] In an embodiment, the new simulated demand curve is a time series of zone energy inputs and the simulated comfort curve is a time series of zone state values.

[0008] In an embodiment, computing the cost function further comprises determining difference between the desired comfort curve and the simulated comfort curve.

[0009] In an embodiment, performing the machine learning process further comprises performing automatic differentiation recursively through the neural network producing a new simulated demand curve.

[0010] In an embodiment, the goal state comprises the cost function being minimized, the model running for a specific time, or the model running a specific number of cycles.

[0011] In an embodiment, the neural network comprises multiple activation functions.

[0012] In an embodiment, performing the machine process comprises computing a gradient of the neural network by taking a reverse gradient of the cost function result forward using automatic differentiation.

[0013] In an embodiment, performing the machine learning process further comprises taking a gradient of the neural network backward using automatic differentiation.

[0014] In an embodiment, the neural network is a heterogeneous neural network.

[0015] In an embodiment, any neuron may be an output neuron.

[0016] In an embodiment, a demand curve creation system is disclosed, the system comprising: at least one processor, a memory in operable communication with the processor and demand curve creation code residing in memory which comprises: receiving a heterogenous neural network of a plurality of controlled building zones; receiving a ground truth comfort state zone curve for at least one of the plurality of controlled building zones; performing a machine learning process to run the heterogenous neural network using a simulated demand curve as input and receiving a simulated comfort curve as output; computing a cost function using the simulated comfort state zone curve and the ground truth comfort curve; using the cost function to determine a new simulated demand curve; iteratively performing the using and computing steps until a goal state is reached; and determining that the current simulated demand curve is the demand curve upon the goal state being reached.

[0017] In an embodiment, the machine learning process comprises a backward pass that computes the gradient with respect to the cost function, and then uses an optimizer to update the demand curves.

[0018] In an embodiment, the backward pass that computes the gradient uses automatic differentiation.

[0019] In an embodiment, the optimizer uses stochastic gradient descent or mini batch gradient descent to minimize the cost function.

[0020] In an embodiment, the neural network is a heterogenous neural network.

[0021] In an embodiment, the heterogenous neural network comprises neurons, any of which may be inputs.

[0022] In an embodiment, any of the neurons may be outputs.

[0023] In an embodiment, a computer-readable storage medium configured with executable instructions to perform a method for creation of a demand curve upon receipt of a comfort curve is disclosed, the method comprising: receiving a ground truth comfort curve for at least one of the plurality of controlled building zones; performing a machine learning process to run a heterogenous neural network using a simulated demand curve as input and receiving a simulated comfort curve as output; computing a cost function using the simulated comfort state zone curve and the ground truth comfort curve; using the cost function to determine a new simulated demand curve; iteratively performing the using and computing steps until a goal state is reached; and determining that the current simulated demand curve is the demand curve upon the goal state being reached.

[0024] In an embodiment, the machine learning process comprises using automatic differentiation to perform back-propagation.

[0025] In an embodiment, the machine learning process comprises using a gradient descent method to perform incremental optimization of the simulated demand curve

[0026] Additional features and advantages will become apparent from the following detailed description of illustrated embodiments, which proceeds with reference to accompanying drawings.

BRIEF DESCRIPTION OF THE FIGURES

[0027] FIG. **1** depicts a computing system in conjunction with which described embodiments can be implemented.

[0028] FIG. **2** is a flow diagram showing an exemplary embodiment of a method to determine demand curves from comfort curves.

[0029] FIG. **3** is a functional block diagram showing an exemplary embodiment of the input and output of a model with which described embodiments can be implemented.

[0030] FIG. **4** is a functional block diagram showing different machine learning functions with which described embodiments can be implemented.

[0031] FIG. **5** depicts a physical system whose behavior can be determined by using a neural network.

[0032] FIG. **6** depicts a simplified neural network that may be used to model behaviors of the physical system of FIG. **3**.

[0033] FIG. **7** is a block diagram describing the nature of exemplary neurons.

[0034] Corresponding reference characters indicate corresponding components throughout the several views of the drawings. Skilled artisans will appreciate that elements in the FIGURES are illustrated for simplicity and clarity and have not necessarily been drawn to scale. For example, the dimensions of some of the elements in the figures may be exaggerated relative to other elements to help to improve understanding of various embodiments. Also, common but well-understood elements that are useful or necessary in a commercially feasible embodiment are often not depicted in order to facilitate a less obstructed view of these various embodiments.

DETAILED DESCRIPTION

[0035] Disclosed below are representative embodiments of methods, computer-readable media, and systems having particular applicability to systems and methods for building neural networks that describe physical structures. Described embodiments implement one or more of the described technologies.

[0036] Various alternatives to the implementations described herein are possible. For example, embodiments described with reference to flowchart diagrams can be altered, such as, for example, by changing the ordering of stages shown in the flowcharts, or by repeating or omitting certain stages.

[0037] Embodiments comprise using a heterogeneous neural model of a defined space that models the various materials in the space as connected nodes, and uses machine learning techniques to train the model. “Defined Space” should be understood broadly—it can be a building, several buildings, buildings and grounds around it, a defined outside space, such as a garden or an irrigated field, etc. A portion of a building may be used as well. For example, a floor of a building may be used, a random section of a building, a room in a building, etc. This may be a space that currently exists, or may be a space that exists only as a design. Other choices are possible as well.

[0038] The defined space may be divided into zones. Each zone may have a different set of requirements for state values during a time period. For example, for the state “temperature,” a user Chris may like their office at 72° from 8 am-5 pm, while a user Avery may prefer their office at 77° from 6 am-4 pm. These preferences can be turned into comfort curves, which are a chronological (time-based) state curve. Chris’s office comfort curve may be 68° from Midnight to 8 am, 72° from 8 am to 5 pm, then 68° from 5 pm to midnight. The comfort curves (for a designated space, such as Chris’s office), are then used to calculate demand curves, which are the amount of state that may be input into the associated zones to achieve the state desired over time. For Chris’s office, that is the amount of heat (or cold) that may be pumped into their office for the 24 hour time period covered by the comfort curve. These zones are controlled by some sort of equipment, allowing their state to be changed. Such zones may be referred to as controlled building zones.

[0039] As a brief overview, in an illustrative embodiment, we have the comfort curve(s) we want zones (e.g., areas) to conform to, such as Chris’s office, as described above, and we wish to find the amount of state necessary over time to meet the temperature (e.g., state) indicated by the comfort curve. We call the amount of state over time a demand curve. To determine comfort curves, we use demand curves as input into a heterogenous model, such as a heterogenous neural network that represents the zones within a structure. We then run the model forward with the demand curves as input to determine comfort curve output for that demand curve. That is, when we pump such an amount of state into a structure, the structure, in turn, has some amount of state over time. This can be thought of as running a furnace from time T to time T+20, and from time T+120 to time T+145 in a structure with two zones. The state propagates through both zones in the neural network which includes the walls, the air, etc. The model outputs the state from time T to time T+240 in both zones, giving us two comfort curves, e.g., what temperature the two zones were from time T to time T+240. We then check the comfort curve output with the desired comfort curve using a cost function, and then machine learning curves are used to tune the input values to create a new demand curve. In some embodiments, a gradient of the cost function is calculated through backpropagate to the input, and then optimized by, e.g., a type of gradient descent, etc., giving us a new demand curve to try. This is repeated until a goal state is reached. The last demand curve run is the demand curve that is then used to determine state needed over time in one or more spaces.

[0040] "Optimize" means to improve, not necessarily to perfect. For example, it may be possible to make further improvements in a value or an algorithm which has been optimized.

[0041] "Determine" means to get a good idea of, not necessarily to achieve the exact value. For example, it may be possible to make further improvements in a value or algorithm which has already been determined.

[0042] A "cost function," generally, compares the output of a simulation model with the ground truth—a time curve that represents the answer the model is attempting to match. This gives us the cost—the difference between simulated truth curve values and the expected values (the ground truth). The cost function may use a least squares function, a Mean Error (ME), Mean Squared Error (MSE), Mean Absolute Error (MAE), a Categorical Cross Entropy Cost Function, a Binary Cross Entropy Cost Function, and so on, to arrive at the answer. In some implementations, the cost function is a loss function. In some implementations, the cost function is a threshold, which may be a single number that indicates the simulated truth curve is close enough to the ground truth. In other implementations, the cost function may be a slope. The slope may also indicate that the simulated truth curve and the ground truth are of sufficient closeness. When a cost function is used, it may be time variant. It also may be linked to factors such as user preference, or changes in the physical model. The cost function applied to the simulation engine may comprise models of any one or more of the following: energy use, primary energy use, energy monetary cost, human comfort, the safety of building or building contents, the durability of building or building contents, microorganism growth potential, system equipment durability, system equipment longevity, environmental impact, and/or energy use CO2 potential. The cost function may utilize a discount function based on discounted future value of a cost. In some embodiments, the discount function may devalue future energy as compared to current energy such that future uncertainty is accounted for, to ensure optimized operation over time. The discount function may devalue the future cost function of the control regimes, based on the accuracy or probability of the predicted weather data and/or on the value of the energy source on a utility pricing schedule, or the like.

[0043] A "goal state" may read in a cost (a value from a cost function) and determine if that cost meets criteria such that a goal has been reached. Such criteria may be the cost reaching a certain value, being higher or lower than a certain value, being between two values, etc. A goal state may also look at the time spent running the simulation model overall, if a specific running time has been reached, the neural network running a specific number of iterations, and so on.

[0044] A machine learning process is one of a variety of computer algorithms that improve automatically through experience. Common machine learning processes are Linear Regression, Logistic Regression, Decision Tree, Support Vector Machine (SVM), Naive Bayes, K-Nearest Neighbors (kNN), K-Means Clustering, Random Forest, Backpropagation with optimization, etc.

[0045] An "optimization method" may include Gradient Descent, stochastic gradient descent, min-batch gradient descent, methods based on Newton's method, inversions of the Hessian using conjugate gradient techniques, Evolutionary computation such as Swarm Intelligence, Bee Colony optimization; SOMA, and Particle Swarm, etc. Non-linear optimization techniques, and other methods known by those of skill in the art may also be used.

[0046] In some machine learning techniques, Backpropagation may be performed by automatic differentiation, or by a different method to determine partial derivatives.

[0047] A "state" as used herein may be Air Temperature, Radiant Temperature, Atmospheric Pressure, Sound Pressure, Occupancy Amount, Indoor Air Quality, CO2 concentration, Light Intensity, or another state that can be measured and controlled.

[0048] Some structures comprise multiple zones (such as rooms or specific areas monitored by a sensor). Each separate zone may be modeled by its own neural model. The collection of neural models can comprise the heterogenous model of the structure. In such a multiple zone model, when zones share a surface, such as (in a building implementation), a wall, a floor, or a ceiling, the outside neuron of one neural model may be used as the inner neuron of the next. Some zones may overlap with other zones, while some zones do not. The structure may be covered in zones, or some locations within a structure may have no explicit zone. Defined spaces may be defined into multiple subsystems. Any of these portioned defined spaces may be used as the subsystems.

[0049] FIG. **1** illustrates a generalized example of a suitable computing environment **100** in which described embodiments may be implemented. The computing environment **100** is not intended to suggest any limitation as to scope of use or functionality of the disclosure, as the present disclosure may be implemented in diverse general-purpose or special-purpose computing environments.

[0050] With reference to FIG. **1**, the core processing is indicated by the core processing **130** box. The computing environment **100** includes at least one central processing unit **110** and memory **120**. The central processing unit **110** executes computer-executable instructions and may be a real or a virtual processor. It may also comprise a vector processor **112**, which allows same-length neuron strings to be processed rapidly. In a multi-processing system, multiple processing units execute computer-executable instructions to increase processing power and as such the vector processor **112**, GPU **115**, and CPU can be running simultaneously. The memory **120** may be volatile memory (e.g., registers, cache, RAM), non-volatile memory (e.g., ROM, EEPROM, flash memory, etc.), or some combination of the two. The memory **120** stores software **185** implementing the described methods of using a neural net to derive a demand curve from a comfort curve.

[0051] A computing environment may have additional features. For example, the computing environment **100** includes storage **140**, one or more input devices **150**, one or more output devices **155**, one or more network connections (e.g., wired, wireless, etc.) **160** as well as other communication connections **170**. An interconnection mechanism (not shown) such as a bus, controller, or network interconnects the components of the computing environment **100**. Typically, operating system software (not shown) provides an operating environment for other software executing in the computing environment **100**, and coordinates activities of the components of the computing environment **100**. The computing system may also be distributed; running portions of the software **185** on different CPUs.

[0052] The storage **140** may be removable or non-removable, and includes magnetic disks, magnetic tapes or cassettes, CD-ROMs, CD-RWs, DVDs, flash drives, or any other medium which can be used to store information and which can be accessed within the computing environment **100**. The storage **140** stores instructions for the software, such a demand curve creation software **185** to implement methods of neuron discretization and creation.

[0053] The input device(s) **150** may be a device that allows a user or another device to communicate with the computing environment **100**, such as a touch input device such as a keyboard, video camera, a microphone, mouse, pen, or trackball, and a scanning device, touchscreen, or another device that provides input to the computing environment **100**. For audio, the input device(s) **150** may be a sound card or similar device that accepts audio input in analog or digital form, or a CD-ROM reader that provides audio samples to the computing environment. The output device(s) **155** may be a display, printer, speaker, CD-writer, or another device that provides output from the computing environment **100**.

[0054] The communication connection(s) **170** enable communication over a communication medium to another computing entity. The communication medium conveys information such as computer-executable instructions, compressed graphics information, or other data in a modulated data signal. Communication connections **170** may comprise input devices **150**, output devices **155**, and input/output devices that allows a client device to communicate with another device over network **160**. A communication device may include one or more wireless transceivers for performing wireless communication and/or one or more communication ports for performing wired communication. These connections may include network connections, which may be a wired or wireless network such as the Internet, an intranet, a LAN, a WAN, a cellular network or another type of network. It will be understood that network **160** may be a combination of multiple different kinds of wired or wireless networks. The network **160** may be a distributed network, with multiple computers, which might be building controllers, acting in tandem. A computing connection **170** may be a portable communications device such as a wireless handheld device, a cell phone device, and so on.

[0055] Computer-readable media are any available non-transient tangible media that can be accessed within a computing environment. By way of example, and not limitation, with the computing environment **100**, computer-readable media include memory **120**, storage **140**, communication media, and combinations of any of the above. Computer readable storage media **165** which may be used to store computer readable media comprises instructions **175** and data **180**. Data Sources may be computing devices, such as general hardware platform servers configured to receive and transmit information over the communications connections **170**. The computing environment **100** may be an electrical controller that is directly connected to various resources, such as HVAC resources, and which has CPU **110**, a GPU **115**, Memory, **120**, input devices **150**, communication connections **170**, and/or other features shown in the computing environment **100**. The computing environment **100** may be a series of distributed computers. These distributed computers may comprise a series of connected electrical controllers.

[0056] Although the operations of some of the disclosed methods are described in a particular, sequential order for convenient presentation, it should be understood that this manner of description encompasses rearrangement, unless a particular ordering is required by specific language set forth below. For example, operations described sequentially can be rearranged or performed concurrently. Moreover, for the sake of simplicity, the attached figures may not show the various ways in which the disclosed methods, apparatus, and systems can be used in conjunction with other methods, apparatus, and systems. Additionally, the description sometimes uses terms like "determine," "build," and "identify" to describe the disclosed technology. These terms are high-level abstractions of the actual operations that are performed. The actual operations that correspond to these terms will vary depending on the particular implementation and are readily discernible by one of ordinary skill in the art.

[0057] Further, data produced from any of the disclosed methods can be created, updated, or stored on tangible computer-readable media (e.g., tangible computer-readable media, such as one or more CDs, volatile memory components (such as DRAM or SRAM), or nonvolatile memory components (such as hard drives) using a variety of different data structures or formats. Such data can be created or updated at a local computer or over a network (e.g., by a server computer), or stored and accessed in a cloud computing environment.

[0058] FIG. **2** illustrates a method **200** that that determines a demand curve from a comfort curve using a heterogenous neural network. The operations of method **200** presented below are intended to be illustrative. In some embodiments, method **200** may be accomplished with one or more additional operations not described, and/or without one or more of the operations discussed. Additionally, the order in which the operations of method **200** are illustrated in FIG. **2** and described below is not intended to be limiting.

[0059] In some embodiments, method **200** may be implemented in one or more processing devices (e.g., a digital or analog processor, or a combination of both; a series of computer controllers each with at least one processor networked together, and/or other mechanisms for electronically processing information etc.) The one or more processing devices may include one or more devices executing some or all of the operations of method **200** in response to instructions stored electronically on an electronic storage medium. The one or more processing devices may include one or more devices configured through hardware, firmware, and/or software to be specifically designed for execution of one or more of the operations of method **200**.

[0060] At operation **205**, a neural network model **205** is received. The neural network model may have been stored in memory, and so may be received from the processing device that the model is being run on. In some implementations, the neural network model may be stored within a distributed system, and received from more than one processors within the distributed system, etc.

[0061] In some embodiments described herein, in a heterogenous neural network, the fundamentals of physics are utilized to model single components or pieces of equipment on a one-to-one basis with neural net neurons. In some embodiments, some neurons use physics equations as activation functions. Different types of neurons may have different equations for their activation functions, such that a neural network may have multiple activation functions within its neurons. When multiple components are linked to each other in a schematic diagram, a neural net is created that models the components as neurons. The values between the objects flow between the neurons as weights of connected edges. These neural networks model not only the real complexities of systems but also their emergent behavior and the system semantics. Therefore, it bypasses two major steps of the conventional AI modeling approaches: determining the shape of the neural net, and training the neural net from scratch.

[0062] As the neurons are arranged in order of an actual system (or set of equations) and because the neurons themselves comprise an equation or a series of equations that describe the function of their associated object, and certain relationships between them are determined by their location in the neural net. Therefore, a huge portion of training is no longer necessary, as the neural net itself comprises location information, behavior information, and interaction information between the different objects represented by the neurons. Further, the values held by neurons in the neural net at given times represent real-world behavior of the objects so represented. The neural net is no longer a black box but itself contains important information. This neural net structure also provides much deeper information about the systems and objects being described. Since the neural network is physics- and location-based, unlike the conventional AI structures, it is not limited to a specific model, but can run multiple models for the system that the neural network represents without requiring separate creation or training.

[0063] In some embodiments, the neural network that is described herein chooses the location of the neurons to tell you something about the physical nature of the system. The neurons are arranged in a way that references the locations of actual objects in the real work. The neural network also places actual equations that can be used to determine object behavior into the activation function of the neuron. The weights that move between neurons are equation variables. Different neurons may have unrelated activation functions, depending on the nature of the model being represented. In an exemplary embodiment, each activation function in a neural network may be different.

[0064] As an exemplary embodiment, a pump could be represented in a neural network as a series of network neurons, some that represent efficiency, energy consumption, pressure, etc. The neurons will be placed such that one set of weights (variables) feeds into the next neuron (e.g., with an equation as its activation function) that uses those weights (variables). Now, two previous required steps, shaping the neural net and training the model may already be performed, at least to a large part. Using embodiments discussed here the neural net model need not be trained on information that is already known. In some embodiments, the individual neurons represent physical representations. These individual neurons may hold parameter values that help define the physical representation. As such, when the neural net is run, the parameters helping define the physical representation can be tweaked to more accurately represent the given physical representation.

[0065] This has the effect of pre-training the model with a qualitative set of guarantees, as the physics equations that describe objects being modeled are true, which saves having to find training sets and using huge amounts of computational time to run the training sets through the models to train them. A model does not need to be trained with information about the world that is already known. With objects connected in the neural net like they are connected in the real world, emergent behavior arises in the model that, in certain cases, maps to the real world. This model behavior that is uncovered is often otherwise too computationally complex to determine. Further, the neurons represent actual objects, not just black boxes. The behavior of the neurons themselves can be examined to determine behavior of the object, and can also be used to refine the understanding of the object behavior. One example of heterogenous models is described in U.S. patent application Ser. No. 17/143,796, filed on Jan. 7, 2021, which is incorporated herein in its entirety by reference.

[0066] At operation **210**, a simulated demand curve is received. Initially, the values of the demand curve may be random, may be a demand curve from another similar model run, etc. As a brief overview, in an illustrative embodiment, we have the comfort curve(s) we want zones (e.g., areas) to conform to, such as Chris's office, as described above, and we wish to find the demand curve (amount of state needed over time) necessary for the structure modeled by the heterogenous model amount of state necessary over time to meet the state (e.g., temperature) indicated by the comfort curve. To do so, we use simulated demand curves as input in the model, run the model which outputs the simulated comfort curve for the given demand curve.

[0067] FIG. **3** is a functional block diagram showing an exemplary embodiment of the input and output of a model. With reference to FIG. **3**, this entails using a demand curve **302** (e.g., an amount of state over time, e.g., zone energy inputs **310**) as input into the heterogenous model **305**. In some embodiments, specifically, the demand curves **302** are injected into a defined space as described by the demand curve at times T(0) through T(n); in the case of a heterogenous model modeling multiple zones, multiple demand curves are injected in corresponding zones over time. In some embodiments, outside influences, such as weather **320**, are also used as inputs. In some cases, if the model is optimized using backpropagation, the backpropagation may not be backpropagated to such outside influence inputs, as, e.g., the weather inputs are not optimized, and so are not changed.

[0068] At operation **220**, the neural network is run. Running the model may entail feedforward—running the demand curve state though the model to the outputs over time T(0)-T(n), capturing internal state values within neurons over the same time T(0)-T(n). These internal state values (such as a temperature variable in a neuron that model's Chris's office) may be the outputs that define the simulated comfort curves **225**. At operation **225**, simulated comfort curve(s) are output. In other embodiments, the comfort curve is output **225** successively in timesteps during the model run, or other methods are used. The first time the heterogenous model is run for a given set of comfort curves, a demand curve **210** may be supplied. This initial demand curve may be determined randomly, or another method may be used, such as a demand curve stored previously that was used as the solution to a similar comfort curve problem.

[0069] At operation **215**, the desired comfort curve(s) are received. These are the curves that describe the state the structure being modeled is to be in; e.g., the temperature Chris's office should be for some time period. These may also be called ground truth comfort curves. Ground truth is the information provided by direct evidence or is the desired output from the neural network.

[0070] At operation **230**, a cost function is computed using the time series of desired comfort curve(s) and the model output—a simulated comfort curve. The cost function measures the difference between the time series of desired comfort curve(s) (e.g., the desired temperature of Chris's office over 24 hours) and the comfort curve(s) output **304** from the neural network **220**. Details of the cost function are described elsewhere, such as with reference to FIG. **7**.

[0071] At operation **235** a goal state is checked to determine if a stopping state has been reached. The goal state may be that the cost from the cost function is within a certain value, that the program has run for a given time, that the model has run for a given number of iterations, that the model has run for a given time, that a threshold value has been reached, such as the cost function should be equal or lower than the threshold value, or a different criterion may be used. If the goal state has not been reached, then a new set of inputs needs to be determined that are incrementally closer to an eventual answer—a lowest (or highest or otherwise determined) value for the cost function, as described elsewhere. This can be thought of as iteratively executing the running neural network **220**, outputting the simulated comfort curve **225**, and computing the cost function **230**, and determining a new simulated demand curve **240** until the goal state **235** is reached.

[0072] At operation **240** new simulated demand curve(s) are determined for the next run of the neural network. This may be performed using the cost function; by using machine learning algorithms, etc. In some embodiments, backpropagation is used to determine a gradient of the cost function in relation to the various values in the neural network (such as the inputs shown in FIG. **7**) is determined and then used to backpropagate that gradient through the neural network to obtain an updated set of demand curves.

[0073] If the goal state **235** has determined that a stopping state been reached, then the demand curve that was used for the last heterogenous model run is set as the solved demand curve; that is, the demand curve that will meet the requirements for the desired comfort curve, within some range. The input that is given to the model is used as the eventual output.

[0074] In some implementations, once the demand curve has been determined **245**, it can then be used to determine equipment behavior **250** for controlled equipment in the building (i.e., when the equipment should turn on and off/hold intermediate values), which can then be used to control equipment in a building. Controlling equipment in such a predetermined fashion can save greatly on energy costs, as well as more accurately controlling state in a defined space for the people and objects therein.

[0075] FIG. **3** depicts some potential inputs and outputs of the heterogenous model. The end result we want is a demand curve **302**. To create the demand curves **302** we use a demand curve **302** as input into the model, the model then outputs a comfort curve **304**. This comfort curve can be thought of as a time series of zone state values **315**, where a zone is a location whose state values wish to be measured. For example, the zone state values may be the simulated temperature taken in a zone for each model time step, here, from T(0) to T(n). There does not need to be a direct way to measure the corresponding location in a space that is to be measured. We iteratively modify the demand curve using machine learning techniques to successively more closely match the comfort curve. The initial demand curve **302** may be chosen randomly from among feasible values, or may be chosen by some other method. The demand curve **302** zone energy inputs **310** feeds state into the model at each model step. The state may be fed in at various locations, such as at neurons that represent HVAC equipment, etc. The state then diffuses through the model through the time steps. The model outputs a comfort curve of the state that has diffused at specific locations, such as at Chris's office—the comfort curve may be a time curve of the temperature at a neuron representing Chris's office. This comfort curve is then compared with the desired state over the model running time. The desired state may be the desired temperature in Chris's office over the time that the model was run for. This pattern iterates until the modeled comfort curve is close enough to the desired comfort curve. Then, the last demand curve used becomes the solution, that is, the amount of state needed to be injected into areas in the model over time. Very often the neural network will be comprised of many neurons that represent zones. The state values of these neuron zones affect each other. For example, a room with a heater in it will warm up the room next door, for example. Therefore, the calculations in a neural network that model such a system are far from straightforward.

[0076] A heterogenous model **305** may take as input a demand curve **302** (e.g., an amount of state over time **310**, in this example, from T(0) to T(n), with the state value fluctuating during that time). These comfort curves **215** can be areas whose state needs can be quantified, such as the amount of humidity that should be in an area, how loud a sound should be, the amount of CO2 in a space allowed, temperature, etc. FIG. **3** only explicitly shows one set of demand curves **302** and a weather curve **320** as input, but many heterogenous model runs will be run for multiple zones, and will include multiple demand curve **302**/zone energy inputs **310**.

[0077] In some implementations, other values that may affect the state of a designated space during the time the model is being run may also be used as input into the neural network **205**. In an exemplary embodiment, the desired comfort curves are for a known period of time. In such instances, weather **320** affecting the building (such as one or more of the temperatures, wind speed, rainfall, etc. in a weather forecast) can be used as a further input into the heterogenous model expressed as weather value curve(s) for the same time period, in this case, T(0) to T(n).

[0078] FIG. **4** is a functional block diagram **400** showing different machine learning functions. At operation **240**, new demand curves are determined. Ways of determining new demand curves **240** are shown with reference to FIG. **4**. These demand curves may be determined by using machine learning **405**. These machine learning **405** techniques may comprise determining gradients **415** of the various variables within the neural network with respect to the cost function. This will provide a space which allows one to incrementally optimize the inputs **430** using the gradients. This shows which way to step to minimize the cost function with respect to the inputs. In some embodiments, gradients of the internal variables with respect to the cost function are determined **415**. With reference to FIG. **7**, in some embodiments, the internal parameters, e.g., **707**-**737** of each neuron have their partial derivatives calculated. Different neurons may have different parameters. For example, a neuron modeling a pump may have parameters such as density, shaft speed, volume flow ratio, hydraulic power, etc. If the derivatives are differentiable, then backpropagation **420** can be used to determine the partial derivatives. Backpropagation finds the derivative of the error (given by the cost function) for the parameters in the neural network, that is, backpropagation computes the gradient of the cost function with respect to the parameters within the network.

[0079] Backpropagation **420** calculates the derivative between the cost function and parameters by using the chain rule from the last neurons calculated during the feedforward propagation (a backward pass), through the internal neurons, to the first neurons calculated. In some embodiments, backpropagation will be performed by automatic differentiation **425**. According to Wikipedia, "automatic differentiation is accomplished by augmenting the algebra of real numbers and obtaining a new arithmetic. An additional component is added to every number to represent the derivative of a function at the number, and all arithmetic operators are extended for the augmented algebra." Other methods may be used to determine the parameter partial derivatives. These include Particle Swarm and SOMA ((Self-Organizing Migrating Algorithm), etc. The backpropagation may work on a negative gradient of the cost function, as the negative gradient points in the direction of smaller values.

[0080] After the partial derivatives are determined, the demand curve **302** is optimized to lower the value of the cost function with respect to the inputs. This process is repeated incrementally. Many different optimizers may be used, which can be roughly grouped into 1) gradient descent methods **435** and 2) other methods **440**. Among the gradient descent methods **435** are standard gradient descent, stochastic gradient descent, and mini-batch gradient descent. Among the other methods **440** are Momentum, Adagrad, AdaDelta, ADAM (adaptive movement estimation), and so on.

[0081] Once a new demand curve is determined, the heterogenous model **305** is run again. If the goal state is reached **235**, then the last demand curve that was determined **145** is determined to be the demand curve that satisfies the original comfort curve requirements. Once the demand curve has been determined, it can be used to determine how much state needs to be input into different zones in a structure at what times to meet the comfort curve needs. This method can save as much as 30% of energy costs over adjusting the state when the need arises. If the goal state has not been reached, then the determine new simulation demand curve step **240**, the run neural network step **220**, the output simulation comfort curve step **225**, and compute cost function state **230** are iteratively performed, within incrementally optimizes the comfort curve until the goal state **235** is reached.

[0082] FIG. **5** depicts a physical system **500** whose behavior can be determined by using a neural network, which may be a heterogenous neural network. A portion of a structure **500** is shown which comprises a Wall **1 505**. This Wall **1 505** is connected to a room which comprises Zone **1 525**. This zone also comprises a sensor **545** which can determine state of the zone. Wall **2 510** is between Zone **1 525** and Zone **2 530**. Zone **2** does not have a sensor. Wall **3 515** is between the two zones **1 525** and **2 530** and the two zones Zone **3 535** and Zone **4 540**. Zone **3** and Zone **4** do not have a wall between them. Zone **4** has a sensor **550** that can determine state in Zone **4**. Zones **3 535** and Zone **4 540** are bounded on the right side by Wall **4 520**. Zone **2 530** has a heater **555**, which disseminates heat over the entire structure. The zones **1-4** are controlled building zones, as their state (in this case heat) can be controlled by the heater **555**.

[0083] FIG. **6** depicts a simplified heterogenous neural network **600** that may be used to model behaviors of the simplified physical system of FIG. **5**. In some embodiments, areas of the structure are represented by neurons that are connected with respect to the location of the represented physical structure. The neurons are not put in layers, as in other types of neural networks. Further, rather than being required to determine what shape the neural network should be to best fit the problem at hand, the neural network configuration is, in some embodiments, determined by a physical layout; that is, the neurons are arranged topologically similar to a physical structure that the neural net is simulating.

[0084] For example, Wall **1 505** is represented by neuron **605**. This neuron **605** is connected by edges **660** to neurons representing Zone **1 620**, Wall **2 610**, and Zone **2 630**. This mirrors the physical connections between Wall **1 505**, Zone **1 525**, Wall **2 510**, and Zone **2 530**. Similarly, the neurons for Zone **1 620**, Wall **2 610**, and Zone **2 630** are connected by edges to the neuron representing Wall **3 615**. The neuron representing Wall **3 515** is connected by edges to the neurons representing Zone **3 635** and Zone **4 640**. Those two neurons **635**, **640** are connected by edges to the neuron representing Wall **3 620**. Even though only one edge is seen going from one neuron to another neuron for clarity in this specific figure, a neuron may have multiple edges leading to another neuron, as will be discussed later. Neurons may have edges that reference each other. For example, edge **660** may be two-way.

[0085] In some implementations, the edges have inputs that are adjusted by activation functions within neurons. Some inputs may be considered temporary properties that are associated with the physical system, such as temperature. In such a case, a temperature input represented in a neural network **600** may represent temperature in the corresponding location in the physical system **500**, such that a temperature input in Neuron Zone **1 620** can represent the temperature at the sensor **545** in Zone **1 525**. In this way, the body of the neural net is not a black box, but rather contains information that is meaningful (in this case, a neuron input represents a temperature within a structure) and that can be used.

[0086] In some implementations, inputs may enter and exit from various places in the neural network, not just from an input and an output layer. This can be seen with inputs of type **1**, which are the dashed lines entering each neuron. Inputs of type **2** are the straight lines. In the illustrative example, each neuron has at least one input. For purposes of clarity not all inputs are included. Of those that are, inputs of type **2** are marked with a straight line, where inputs of type **1** are marked with a dashed line. Input **665** is associated with the neuron that represents Wall **1 605**, while input **652** is associated with the neuron that represents Wall **3 615**. Signals, (or weights) passed from edge to edge, and transformed by the activation functions, can travel not just from one layer to the layer in a lock-step fashion, but can travel back and forth between layers, such as signals that travel along edges from Zone **1 620** to Wall **2 610**, and from there to Zone **2 630**. Further, there may be multiple inputs into a single neuron, and multiple outputs from a single neuron.

For example, a system that represents a building may have several inputs that represent different states, such as temperature, humidity, atmospheric pressure, wind, dew point, time of day, time of year, etc. These inputs may be time curves that define the state over time. A system may have different inputs for different neurons. In some embodiments, inputs may be time curves, which defines the state at a particular time, over a period of time.

[0087] In some implementations, outputs are not found in a traditional output layer, but rather are values within a neuron within the neural network. These may be located in multiple neurons. These outputs for a run may be time curves. For example, the neuron associated with Zone **1 620** may have a temperature value that can be looked at each timestep of a model run, creating temperature time curves that represent the temperature of the corresponding physical Zone **1 525**.

[0088] In some embodiments, activation functions in a neuron transform the weights on the upstream edges, and then send none, some, or all of the transformed weights to the next neuron(s). Not every activation function transforms every weight. Some activation functions may not transform any weights. In some embodiments, each neuron may have a different activation function. In some embodiments, some neurons may have similar functions.

[0089] A heterogenous neural network may have many inputs and outputs. Each output may have its own comfort curve associated with it, so a heterogenous neural network may input many demand curves **302** and output many comfort curves **304**.

[0090] FIG. **7** is a block diagram **700** describing possible inputs and outputs of neurons. Neural networks described herein may not have traditional input and output layers. Rather, neurons may have internal values that can be captured as output. Similarly, a wide variety of neurons, even those deep within a neural net can be used for input. For example, Chris's office may be in Zone **4 540**. This zone may be represented by a neuron **640** that is somewhere in the middle of a neural network **600**. A zone neuron **715** may have an activation function that is comprised of several equations that model state moving through the space. The space itself may have inputs associated with it, e.g., Layer Mass **732**, Layer Heat Capacity **735**, and Heat Transfer Rate **737**, to name a few. For the purposes of this disclosure, we are calling these type **1** inputs **725**, **730**. The neuron may also have temporary values that flow through the neural network, that may be changed by the neuron's activation function. These type **2** inputs **707**, **717** may be qualities such as Temperature **719**, Mass Flow Rate **721**, Pressure **723**, etc. Different neurons may have different values. For example a Wall Neuron **705** may have Type **1** inputs **725** such as Surface Area **727**, Layer Heat Capacity **728**, and Thermal Resistance **729**, as well as Type **2** inputs **707**. A comfort curve output **304** of the neural network **600** may comprise a value gathered from among the variables in a neuron. For example, Chris's office may be zone **4 540**, which is represented by the neuron Zone **4 640**. The output of the heterogenous model **305** may be a time series of the zone neuron temperature.

[0091] A cost function can be calculated using these internal neural net values. A cost function (also sometimes called a loss function) is a performance metric on how well the neural network is reaching its goal of generating outputs as close as possible to the desired values. To create the cost function we determine the values we want from inside the neural network, retrieve them, then make a vector with the desired values; viz: a cost C=(y,0) where y=desired values, and 0=network prediction values. These desired values are sometimes called the "ground truth." With reference to FIG. **5**, Zone **1 525** has a sensor **545** which can record state within the zone. Similarly, Zone **4 540** has a sensor **550** which can also record state values. In some embodiments, desired values may be synthetic, that is, they are the values that are hoped to be reached. In some embodiments, the desired values may be derived from actual measurements.

[0092] Continuing the example from FIG. **5**, there are two sensors that gather sensor data. The desired values are time series of the actual temperatures from the sensors. The network prediction values are not determined from a specific output layer of the neural network, as the data we want is held within neurons within the network. The zone neurons **715** in our sample model hold a temperature value **719**. The network prediction values to be used for the cost function are, in this case, the values (temperature **719**) within the neuron **620** that corresponds to Zone **525** (where we have data from sensor **545**) and the values (temperature **719**) within the neuron **640** that correspond to Zone **4 540**, with sensor **550**.

[0093] When the model is run, a record of the temperature values from those zones can be accumulated from time t0 to tn; and neuron **640** which may be another internal temperature from time t0 to tn or a different value. These are our network prediction values. In the instant example, the desired values are data from the sensors **545** and **550**. Once the we have the network prediction values and the desired values, we can calculate the cost function, which quantifies the error between what the model predicts and the desired values and presents it as a value, a vector, or something else.

[0094] The networks described herein may be heterogenous neural networks. Heterogenous neural networks comprise neural networks that have neurons with different activation functions. These neurons may comprise virtual replicas of actual or theoretical physical locations. The activation functions of the neurons may comprise multiple equations that describe state moving through a location associated with the neuron. In some embodiments, heterogenous neural networks also have neurons that comprise multiple variables that hold values that are meaningful outside of the neural network itself. For example, a value, such as a temperature value (e.g., **719**) may be held within a neuron (e.g., **640**) which can be associated with an actual location (e.g., **540**).

[0095] In view of the many possible embodiments to which the principles of the disclosed invention may be applied, it should be recognized that the illustrated embodiments are only preferred examples of the invention and should not be taken as limiting the scope of the invention. Rather, the scope of the invention is defined by the following claims. We therefore claim as our invention all that comes within the scope and spirit of these claims.

We claim:

1. A method of determining a demand curve implemented by one or more computers comprising:

receiving a neural network of a plurality of controlled building zones;

receiving a desired comfort curve for at least one of the plurality of controlled building zones;

performing a machine learning process to run the neural network using a simulated demand curve as input and receiving a simulated comfort curve as output;
computing a cost function using the simulated comfort curve and the desired comfort curve;
using the cost function to determine a new simulated demand curve;
iteratively executing the performing, computing, and using steps until a goal state is reached; and
determining that the new simulated demand curve is the demand curve upon the goal state being reached.

2. The method of claim **1**, wherein the simulated demand curve is a time series of zone energy inputs and the simulated comfort curve is a time series of zone state values.

3. The method of claim **2**, wherein computing the cost function further comprises determining difference between the desired comfort curve and the simulated comfort curve.

4. The method of claim **2**, wherein performing the machine learning process further comprises performing automatic differentiation backward through the neural network producing a new simulated demand curve.

5. The method of claim **1**, wherein the goal state comprises the cost function being minimized, the neural network running for a specific time, or the neural network running a specific number of iterations.

6. The method of claim **5**, wherein the neural network comprises multiple activation functions within its neurons.

7. The method of claim **6**, wherein performing the machine learning process comprises computing a gradient of the neural network by using automatic differentiation.

8. The method of claim **7**, wherein performing the machine learning process further comprises optimizing the simulated demand curve using gradient descent, stochastic gradient descent, or mini-batch gradient descent.

9. The method of claim **1**, wherein the neural network is a heterogeneous neural network.

10. The method of claim **9**, wherein any neuron may be an output neuron.

11. A demand curve creation system, the system comprising: a processor, a memory in operable communication with the processor, and demand curve creation code residing in memory which comprises: receiving a neural network of a plurality of controlled building zones;
receiving a ground truth comfort curve for at least one of the plurality of controlled building zones;
performing a machine learning process to run the neural network using a simulated demand curve as input and receiving a simulated comfort curve as output;
computing a cost function using the simulated comfort curve and the ground truth comfort curve;
using the cost function to determine a new simulated demand curve;
iteratively executing the performing the machine learning process, computing the cost function, and using the cost function steps until a goal state is reached; and
determining that the simulated demand curve is the demand curve upon the goal state being reached.

12. The demand curve creation system of claim **11**, wherein the machine learning process comprises using backpropagation that computes a cost function gradient for values in the neural network, and then uses an optimizer to update the demand curve.

13. The demand curve creation system of claim **12**, wherein the backpropagation that computes a cost function gradient uses automatic differentiation.

14. The demand curve creation system of claim **12**, wherein the optimizer uses stochastic gradient descent or mini-batch gradient descent to minimize the cost function.

15. The demand curve creation system of claim **11**, wherein the neural network is a heterogenous neural network.

16. The demand curve creation system of claim **15**, wherein the heterogenous neural network comprises neurons, any of which may be inputs.

17. The demand curve creation system of claim **16**, wherein any of the neurons may be outputs.

18. A computer-readable storage medium configured with executable instructions to perform a method for creation of a demand curve upon receipt of a comfort curve, the method comprising:
receiving a ground truth comfort curve for at least one of a plurality of controlled building zones;
performing a machine learning process to run a heterogenous neural network using a simulated demand curve as input and receiving a simulated comfort curve as output;
computing a cost function using the simulated comfort curve and the ground truth comfort curve;
using the cost function to determine a new simulated demand curve;
iteratively executing the performing the machine learning process, computing the cost function, and using the cost function steps until a goal state is reached; and
determining that the simulated demand curve is the demand curve upon the goal state being reached.

19. The computer-readable storage medium of claim **18**, wherein the machine learning process comprises using automatic differentiation to perform backpropagation.

20. The computer-readable storage medium of claim **19**, wherein the machine learning process comprises using a gradient descent method to perform incremental optimization of the simulated demand curve.

* * * * *