METHODS AND APPARATUS FOR MIXER CHARACTERIZATION

FIELD OF THE DISCLOSURE

[0001] This disclosure relates generally to bioprocess systems and, more particularly, to methods and apparatus for mixer characterization.

BACKGROUND

[0002] Bioprocesses are used to produce medically and industrially critical products (e.g., therapeutics, biofuels, etc.) using biomanufacturing through optimization of natural and/or artificial biological systems to allow for large-scale production. Mixing during a biomanufacturing process is critical for successful production of homogenous batches of biologicals. Mixing equipment can be used for numerous operations, including homogenization, suspension, dispersion, and/or heat exchange. For example, mixing is regulated during medium and buffer preparation, cell culture, fermentation, and/or virus inactivation.

BRIEF DESCRIPTION OF THE DRAWINGS

[0003] FIG. 1 is an example diagram of an example environment for mixer characterization, including example mixer characterization circuitry used for identifying mixing time in accordance with the teachings of this disclosure.

[0004] FIG. 2 illustrates example predicted mixing time and experimental mixing time associated with a bioreactor.

[0005] FIG. 3 illustrates an example mixer with virtual probes positioned throughout the mixing unit as part of measurements taken for computational fluid dynamics analysis.

[0006] FIG. 4 illustrates example impeller designs associated with example mixing units.

[0007] FIG. 5 illustrates example probe positioning in a mixing unit for assessment of fluid dynamics and homogeneity over time.

[0008] FIG. 6 illustrates example tracer flow lines generated as part of assessing fluid distribution through a mixing unit.

[0009] FIG. 7 illustrates an example graphical representation of probe-based measurements at various locations throughout the mixing unit.

[0010] FIG. 8 shows an example graphical representation of mixing time curves indicating changes in mixing time based on power density readings for mixing units of various sizes.

[0011] FIG. 9 shows an example graphical representation of a change in power number based on changes in mixer unit volume.

[0012] FIG. 10 illustrates an example graphical representation of a comparison of mathematical modeling-based mixing times versus computations fluid dynamics-based mixing times for a smaller mixing unit.

[0013] FIG. 11 illustrates an example graphical representation of a comparison of mathematical modeling-based mixing times versus computations fluid dynamics-based mixing times for a larger mixing unit.

[0014] FIG. 12 is a block diagram representative of the mixer characterization circuitry that may be implemented in the example environment of FIG. 1.

[0015] FIG. 13 is a flowchart representative of example machine-readable instructions and/or example operations that may be executed, instantiated, and/or performed by example programmable circuitry to implement the example mixer characterization circuitry of FIG. 1.

[0016] FIG. 14 is a flowchart representative of example machine-readable instructions and/or example operations that may be executed, instantiated, and/or performed by example programmable circuitry to implement the example mixer characterization circuitry of FIG. 1 to identify power number and mixing time based on input parameters.

[0017] FIG. 15 is a flowchart representative of example machine-readable instructions and/or example operations that may be executed, instantiated, and/or performed by example programmable circuitry to implement the example mixer characterization circuitry of FIG. 1 to predict mixing time and feed location based on user-selected parameters.

[0018] FIG. 16 is a block diagram of an example processing platform including programmable circuitry structured to execute, instantiate, and/or perform the example machine readable instructions and/or perform the example operations of FIGS. 13-15 to implement the mixer characterization circuitry of FIG. 1.

[0019] FIG. 17 is a block diagram of an example implementation of the programmable circuitry of FIG. 16.

[0020] FIG. 18 is a block diagram of another example implementation of the programmable circuitry of FIG. 16.

[0021] FIG. 19 is a block diagram of an example software/firmware/instructions distribution platform (e.g., one or more servers) to distribute software, instructions, and/or firmware (e.g., corresponding to the example machine readable instructions of FIGS. 13, 14 and/or 15) to client devices associated with end users and/or consumers (e.g., for license, sale, and/or use), retailers (e.g., for sale, re-sale, license, and/or sub-license), and/or original equipment manufacturers (OEMs) (e.g., for inclusion in products to be distributed to, for example, retailers and/or to other end users such as direct buy customers).

[0022] In general, the same reference numbers will be used throughout the drawing(s) and accompanying written description to refer to the same or like parts. The figures are not to scale. Unless specifically stated otherwise, descriptors such as "first," "second," "third," etc., are used herein without imputing or otherwise indicating any meaning of priority, physical order, arrangement in a list, and/or ordering in any way, but are merely used as labels and/or arbitrary names to distinguish elements for ease of understanding the disclosed examples. In some examples, the descriptor "first" may be used to refer to an element in the detailed description, while the same element may be referred to in a claim with a different descriptor such as "second" or "third." In such instances, it should be understood that such descriptors are used merely for identifying those elements distinctly that might, for example, otherwise share a same name.

[0023] As used herein, the phrase "in communication," including variations thereof, encompasses direct communication and/or indirect communication through one or more intermediary components, and does not require direct physical (e.g., wired) communication and/or constant communication, but rather additionally includes selective communication at periodic intervals, scheduled intervals, aperiodic intervals, and/or one-time events.

[0024] As used herein, "programmable circuitry" is defined to include (i) one or more special purpose electrical circuits (e.g., an application specific circuit (ASIC)) structured to perform specific operation(s) and including one or more semiconductor-based logic devices (e.g., electrical hardware implemented by one or more transistors), and/or (ii) one or more general purpose semiconductor-based electrical circuits programmable with instructions to perform specific functions(s) and/or operation(s) and including one or more semiconductor-based logic devices (e.g., electrical hardware implemented by one or more transistors). Examples of programmable circuitry include programmable microprocessors such as Central Processor Units (CPUs) that may execute first instructions to perform one or more operations and/or functions, Field Programmable Gate Arrays (FPGAs) that may be programmed with second instructions to cause configuration and/or structuring of the FPGAs to instantiate one

or more operations and/or functions corresponding to the first instructions, Graphics Processor Units (GPUs) that may execute first instructions to perform one or more operations and/or functions, Digital Signal Processors (DSPs) that may execute first instructions to perform one or more operations and/or functions, XPUs, Network Processing Units (NPUs) one or more microcontrollers that may execute first instructions to perform one or more operations and/or functions and/or integrated circuits such as Application Specific Integrated Circuits (ASICs). For example, an XPU may be implemented by a heterogeneous computing system including multiple types of programmable circuitry (e.g., one or more FPGAs, one or more CPUs, one or more GPUs, one or more NPUs, one or more DSPs, etc., and/or any combination(s) thereof), and orchestration technology (e.g., application programming interface(s) (API(s)) that may assign computing task(s) to whichever one(s) of the multiple types of programmable circuitry is/are suited and available to perform the computing task(s).

[0025] As used herein integrated circuit/circuitry is defined as one or more semiconductor packages containing one or more circuit elements such as transistors, capacitors, inductors, resistors, current paths, diodes, etc. For example, an integrated circuit may be implemented as one or more of an ASIC, an FPGA, a chip, a microchip, programmable circuitry, a semiconductor substrate coupling multiple circuit elements, a system on chip (SoC), etc.

DETAILED DESCRIPTION

[0026] Bioprocess monitoring requires real-time, continuous measurement of process variables to ensure the stability, efficiency, and reproducibility of the process to provide a high-quality product. By measuring quality-related process variables that are necessary to maintain a narrow range of environmental conditions, consistent reproduction of the desired product can be achieved and documented. A variety of bioprocess instruments (also referred to herein as bioprocess units) are used during upstream processing (e.g., biomass expansion, media development and preparation, etc.) and downstream processing (e.g., product extraction and purification from the biomass, etc.), including bioreactors and mixers. For example, a bioreactor can be used to create a controlled environment for *in vitro* management of cells (e.g., cell proliferation, differentiation, etc.) during upstream processing. Bioreactors can include sensors directly interfacing, or used in conjunction with, the bioreactors to measure process variables, including oxygen and carbon dioxide concentration, biomass concentration, flow injection, and/or overall media composition. Mixing units (e.g., mixers) and/or any other type of mixing equipment is used for numerous operations such as

homogenization, suspension, dispersion and heat exchange during medium and buffer preparation, cell culture, fermentation, and/or virus inactivation. Mixing equipment needs to be scalable to satisfy requirements for low and large volumes as batch sizes and yields vary. Given the importance of mixers in bioprocessing, mixers should allow for efficiency, reproducibility, and scalability. As such, mixers should be optimized to handle various volumes and maintain uniformity during production.

[0027] For example, mixing permits the homogenization of fluids through the elimination of concentration gradients and temperature variations. In some examples, agitation can be introduced into a fluid system through the use of a friction-less/levitating impeller, a stirring shaft, a tumbling motion (non-rotating stirrer) or an oscillating disk. The fluid dynamics in the mixer can be classified as distribution, dispersion, and/or diffusion. To achieve effective fluid distribution, the impeller circulates fluid around the entire vessel in a set timeframe, such that the velocity of the fluid moved by the impeller distributes the fluid throughout all areas of the tank. The shape and speed of the impeller affects fluid distribution and turbulence generated based on the tank geometry. Separately, dispersion includes the process of eddy formation to create smaller flows from the bulk fluid flow, which facilitates rapid transfer of fluid content throughout the vessel. While turbulence facilitates fluid mixing at the macroscopic level, diffusion creates a concentration equilibrium at the molecular level.

[0028] The mixing process can be characterized using parameters such as mixing duration, homogenization time, quality, and/or power input (e.g., quantity of power introduced per unit of volume). For example, mixing duration can be determined based on data acquired in the same system under similar conditions and using a similar homogenization time. In some examples, impeller type, geometry, size, and/or orientation relative to the vessel can be used to determine mixing parameters. However, determination of mixing parameters for mixer type selection requires extensive testing and validation, as well as additional qualitative and quantitative testing to optimize mixing parameters for a given mixing process.

[0029] Methods and apparatus disclosed herein characterize bioprocess-based mixers to allow users to select and test mixers based on intended bioprocess applications. In examples disclosed herein, a mixer can be selected according to a plurality of impeller designs and/or operating parameters to identify a mixer that is most appropriate for particular processing needs. As such, mixer selection can be performed without the need for extensive qualitative and quantitative testing. In examples disclosed herein, a time to achieve homogeneity is determined and conveyed for selection of the mixer. In some examples,

addition of acids, bases, and/or salts can be sequenced and automated based on predetermined mixing time values. In examples disclosed herein, an identification is performed to determine a desired location to add feed during the mixing process. Likewise, methods and apparatus disclosed herein identify a mixing rate suitable for low sheer and/or to achieve uniform dispersion. In some examples, a total time for mixing the media powders is determined and presented, such that the need for intermittent visual checks is eliminated. In some examples, media transfer to a bioreactor is automated based on predetermined mixing time values. Furthermore, in examples disclosed herein, a user interface is generated to allow a user to view and select mixer-based information such as mixer size, working volume, and/or impeller speed. Additionally, methods and apparatus disclosed herein generate corresponding mixing times based on the user's mixer-based input parameters. In any and/or all of the examples disclosed herein, determined values are used for automated control of the mixer. For example, once a time to achieve homogeneity is determined, a selected mixer can be controlled such that mixing is performed based on the predetermined time. Similarly, the determined mixing rate (e.g., for low shear mixing, etc.) and/or location (e.g., for adding feed) can be automatically used by the selected mixer for a desired mixing application.

[0030] FIG. 1 is an example environment 100 for mixer characterization, including example mixer characterization circuitry used for identifying mixing time in accordance with the teachings of this disclosure. In the example of FIG. 1, the environment 100 includes mixer characterization circuitry 102, a user interface 103, a mixer 105, a network 110, a workstation 120, a user interface display 130, and a bioreactor 135.

[0031] The mixer characterization circuitry 102 performs mixer characterization to determine a mixing time associated with a given mixer type and/or size. For example, the mixer characterization circuitry 102 receives input parameters associated with a given mixer type (e.g., based on mixer volume, impeller design and/or geometry, impeller speed, etc.). The input parameters include mixer size, working volume, and/or impeller speed. In examples disclosed herein, the mixer characterization circuitry 102 performs modeling to identify a computational parameter (e.g., a shear, a power density, or a power number) associated with the input parameters and match a mixing time to the input parameters based on the power number, as described in more detail in connection with FIGS. 13-15. In examples disclosed herein, the mixer characterization circuitry 102 receives user-selected inputs associated with a given mixer (e.g., mixer 105), including mixer working volume and/or impeller speed (e.g., via the user interface 103). The mixer characterization circuitry 102 further predicts mixing time and feed location based on user-selected parameters by

determining the time required to reach homogeneity, the sequencing of additives based on mixing times, and/or the mixing speed used to maintain low shear and/or uniform dispersion, as described in more detail in connection with FIG. 15. In some examples, the mixer characterization circuitry 102 generates view(s) of velocity contour(s) and particle flow(s) and outputs (e.g., by displaying, transmitting, printing, storing, etc.) the view(s) to provide a visualization of the resulting mixing characteristics for the selected mixer type. In some examples, the mixer characterization circuitry 102 initiates the mixing using mixer 105 and subsequently initiates the automatic transfer of media to the bioreactor (e.g., bioreactor 135) once mixing is completed.

[0032] The user interface 103, associated with the workstation 120, generates the user interface display 130. The user interface display 130 includes example velocity contours 140, an example mixer size selection 145, an example working volume selection 150, an example impeller speed selection 155, an example tracer flow line display 160, an example power number indicator 165, an example mixing time indicator 170, and an example calculation selector 175. In some examples, the user interface 103 displays the velocity contours 140 and/or the tracer flow line display 160 to allow the user to visualize the mixing process in real-time and/or to present a visual representation of fluid dispersion at specific time intervals during the duration of the mixing process. In the example of FIG. 1, the user can select the mixer type using the mixer size selection 145 (e.g., XDM-200, etc.), the working volume using the working volume selection 150 (e.g., 200 L), and the impeller speed using the impeller speed selection 155 (e.g., 20 revolutions per minute, rpm). In the example of FIG. 1, the mixer characterization circuitry 102 determines the power number and the mixing time associated with the mixer input features (e.g., mixer size, working volume, and impeller speed), and the user interface 103 displays the power number using the power number indicator 165 and the mixing time using the mixing time indicator 170. For example, the user can select the calculation selector 175 to display the identified power number and mixing time results based on the user-provided inputs.

[0033] The mixer 105 can be any type of mixer used in bioprocess systems (e.g., a jacketed mixer, a single-wall mixer, etc.). In the example of FIG. 1, the mixer 105 is a unit that is part of upstream processing that can also include media storage tanks and/or media exchangers. In some examples, the mixer 105 can be used in downstream processes that focus on optimizations to extract and maximize final product yields (e.g., filtration, mixing, purification based on chromatography, etc.). The mixer 105 can be used during biomanufacturing for buffer and media preparations or other mixing needs (e.g., blending,

stirring, suspending, dissolving, etc.). In examples disclosed herein, the mixer 105 can automatically adjust mixing based on mixing values determined by the mixer characterization circuitry 102 of FIG. 1. For example, the mixing rate and/or location of feed addition can be adjusted automatically based on outputs received from the mixer characterization circuitry 102, as described in more detail below. As such, calculations performed by the mixer characterization circuitry 102 can be used to directly control and/or automatically adjust the performance of the mixer 105. The network 110 is a wired or wireless network used for communication among the mixer 105, workstation 120, and/or bioreactor 135. In some examples, the network 110 is a wired network (e.g., PROFIBUS, EthernetIP, CAN, RS-485, etc.). In some examples, the network 110 allows for communication between a control system and field devices (e.g., bioprocess units). The network 110 can be implemented using any suitable wireless network(s) including, for example, one or more data buses, one or more Local Area Networks (LANs), one or more wireless LANs, one or more cellular networks, the Internet, etc. As used herein, the phrase "in communication," including variances thereof, encompasses direct communication and/or indirect communication through one or more intermediary components and does not require direct physical (e.g., wired) communication and/or constant communication, but rather additionally includes selective communication at periodic or aperiodic intervals, as well as one-time events.

[0034] The workstation 120 permits a user to operate and/or monitor the bioprocess units, including the mixer 105 and/or the bioreactor 135. In some examples, the workstation 120 is communicatively coupled to a controller, a communication interface, a data logger, and/or a data storage (not shown) via a bus or local area network (LAN) (e.g., an Area Control Network (ACN)). The LAN can be implemented using any desired communication medium and protocol. For example, the LAN can be based on a hardware or wireless Ethernet communication protocol. However, any other suitable wired or wireless communication medium and protocol could be used. The workstation 120 can be configured to perform operations associated with one or more information technology applications, user-interactive applications (e.g., via the user interface 103), and/or communication applications. For example, the workstation 120 can be configured to perform operations associated with process control-related applications and communication applications that enable the workstation 120 to communicate with other devices or systems using any desired communication media (e.g., wireless, hardwired, etc.) and protocols (e.g., HTTP, SOAP, etc.). In the example of FIG. 1, the workstation 120 can be used to receive input from the

mixer characterization circuitry 102 and/or provide input to the mixer characterization circuitry 102 via the user interface 103.

[0035] The bioreactor 135 can receive mixed fluid from the mixer 105 (e.g., fluid transfer 137 from the mixer 105 to the bioreactor 135). In some examples, the mixer characterization circuitry 102 identifies a time at which to initiate automatic transfer of media to the bioreactor 135 once mixing is completed. For example, the bioreactor 135 can be used to create a controlled environment for *in vitro* management of cells (e.g., cell proliferation, differentiation, etc.) during upstream processing. Bioreactors can include sensors directly interfacing, or used in conjunction with, the bioreactors to measure process variables, including oxygen and carbon dioxide concentration, biomass concentration, flow injection, and/or overall media composition. For example, the bioreactor 135 can be used for expansion (e.g., growth of CHO cells, bacteria, yeast, etc.) to permit biological reactions under controlled conditions for a variety of purposes, including the production of pharmaceuticals, vaccines, antibodies, and biofuel. Such bioreactors can be used in any domain of industrial biotechnology requiring large scale production, providing the necessary biological, biochemical, and biomechanical conditions for synthesis of desired products.

[0036] FIG. 2 illustrates example predicted mixing time and experimental mixing time 200 associated with a bioreactor. In examples disclosed herein, any other type of unit (e.g., mixer, etc.) can be used to identify and compare predicted and experimental mixing times. In the example of FIG. 2, a total of 18 data points were tested in physical experiments using bioreactor parameters to identify experimental T95 mixing times. For example, a comparison of the experimental mixing times 215 to predicted mixing times 210 is shown based on output predictions from the mixer characterization circuitry 102 and the identified experimental data. In the example of FIG. 2, the mixer characterization circuitry 102 identifies a predicted mixing time, as described in connection with FIGS. 13-15. For example, the mixer characterization circuitry 102 determines the predicted mixing time based on bioreactor parameters such as bioreactor volume (e.g., 40 Liters (L), 120 L, 200 L), average power to volume ratio (e.g., 30 Watts per meter cubed (W/m^3), 90 W/m^3, 150 W/m^3), and/or direction of impeller rotation (e.g., up, down). In the example of FIG. 2, a T95 mixing time is identified corresponding to when a measured value (e.g., conductivity, glucose, osmolality, etc.) reaches 95% of a final stable value. In some examples, the T95 mixing times (e.g., T95 mixing times for predicted versus experimental data 205) can be calculated based on an analytic reaching 95% of the final recorded stable value determined from data collected during the last 10 minutes of mixing.

[0037] FIG. 3 illustrates an example mixer 300 with virtual probes (e.g., P1, P2, P3, P4) positioned throughout the mixing unit as part of measurements taken for computational fluid dynamics analysis. In the example of FIG. 3, the mixer 300 has a volume of 2400 L and includes two down-pumping impellers 305 positioned on opposite sides of the bottom region of the mixer 300, where each impeller has a speed of 300 revolutions per minute (rpm). In the example of FIG. 3, a solution of sodium chloride (e.g., 5L NaCl) is added to a region of the mixer. In examples disclosed herein, addition of acids, bases, and/or salts can be sequenced and automated based on predetermined mixing time values determined by the mixer characterization circuitry 102 of FIG. 1, as described in more detail in connection with FIG. 15. In examples disclosed herein, an identification is performed to determine a desired location for adding feed during the mixing process. In the example of FIG. 3, the mixer characterization circuitry 102 identifies the mixing time associated with mixing the added sodium chloride solution. For example, an experimental T98 value (e.g., when a measured value reaches 98% of the final stable value) is 71.4 seconds, whereas the mixer characterization circuitry 102 generates a predicted T98 value of 75 seconds. For example, the virtual probes (e.g., P1, P2, P3, P4) positioned within the mixer 300 can be used to obtain experimental values (e.g., T98 values) to determine the experimental mixing time. As such, the mixer characterization circuitry 102 accurately predicts the mixing time based on user-provided parameters (e.g., such as the type of mixing unit and mixer volume), without the need for testing using virtual probes, as shown in connection with FIG. 1.

[0038] FIG. 4 illustrates example impeller designs 400 associated with various mixing units (e.g., a fist impeller design 425 and a second impeller design 450). For example, impeller design directly affects the mixing time. Factors such as impeller type, geometry, size, and/or orientation relative to the vessel are considered to determine how the impeller affects the mixing time. As shown in connection with FIG. 4, the impeller shape and diameter (e.g., a first diameter 465 associated with the first impeller design 425 (e.g., 200-300 millimeter diameter), a second diameter 480 associated with the second impeller design 450 (e.g., 100-200 mm diameter), etc.) can vary depending on the type of mixing unit. To achieve effective fluid distribution, the impeller circulates fluid around the entire vessel in a set timeframe, such that the velocity of the fluid moved by the impeller distributes the fluid throughout all areas of the tank. The shape and speed of the impeller affects fluid distribution and turbulence generated based on the tank geometry. For example, smaller diameter impellers would not generate the same level of fluid distribution as larger diameter impellers in a bioreactor of the same size. As such, identification of mixing times using various

impeller designs can be used to determine how impeller design affects the mixing time when a user identifies a certain mixer type as part user-based inputs into the user interface 103 of FIG. 1.

[0039] FIG. 5 illustrates example probe positioning 500 in a mixer for assessment of fluid dynamics and homogeneity over time. As described in connection with FIG. 4, virtual probes (e.g., P1 - P11) are positioned throughout the mixer unit to track the mixing progress. For example, probes P1, P4, P7, P10, and P11 are positioned in a first cross-section 505, probes P2, P5, and P8 are positioned in a second cross-section 510, and probes P3, P6, and P9 are positioned in a third cross-section 515. In the example of FIG. 5, the first and third cross-sections 505, 515 are located in areas around impeller 520, while the second cross-section 510 is located at a cross-section of the impeller 520 positioned within the mixer. As such, the virtual probes can be used to gather fluid dynamics-associated data during the mixing process (e.g., to identify T95 and/or T98 mixing times). In examples disclosed herein, the volume of the mixer (e.g., 44 L, 100 L) and the agitation (e.g., 42 rpm, 50 rpm) results in an experimental mixing time of 25.8 seconds and 37.8 seconds, respectively, with a corresponding predicted T95 mixing time of 26.16 seconds and 40.44 seconds (e.g., identified using the mixer characterization circuitry 102 of FIG. 1), resulting in deviations of 1.4% and 6.9%, respectively.

[0040] FIG. 6 illustrates example tracer flow lines 600 generated as part of assessing fluid distribution through a mixer. In the example of FIG. 6, the illustrated tracer flow lines 605, 610, 615, 620, 625, 630 are identified for a variety of mixer volumes (e.g., ranging from 52 L to 1000 L). The mixer volumes range in different sizes, allowing the mixer characterization circuitry 102 to match the tracer flow lines 605, 610, 615, 620, 625, 630 to a given mixer based on user-provided input (e.g., to generate the tracer flow line display 160 of FIG. 1). For example, the tracer flow lines track fluid velocity and circulation patterns within the mixer, showing a visual representation of the mixing progress at various time intervals of the mixing process. As such, the tracer flow lines can serve as an indication of the mixing progress and type of fluid distribution within the mixer.

[0041] FIG. 7 illustrates an example graphical representation 700 of probe-based measurements at various locations throughout the mixer. In the example of FIG. 7, a concentration percentage 710 of a solvent is identified over time 715 based on input from probes 720 (e.g., probes P1 - P27). The probe concentration graph 700 can be used to determine mixing times for a particular mixing vessel and the mixing time(s) can be modeled

to match a set of input parameters provided by the user (e.g., using the user interface display 130), as described in more detail in connection with FIG. 14.

[0042] FIG. 8 shows an example first graphical representation 800 and example second graphical representation 850 of mixing time 805 curves indicating changes in mixing time 810 based on power density readings 815 (P/V) for mixers of various sizes. In the example of FIG. 8, mixing times 810 (e.g., T95 mixing time) can be determined based on the average power to volume ratio (e.g., 30 W/m^3, 90 W/m^3, 150 W/m^3) for a variety of mixer volumes (e.g., 52 L, 122 L, 200 L). For example, as the average power to volume ratio increases, the mixing time decreases. This same trend is also observed for larger mixer volumes (e.g., 300 L, 610 L, 1000 L) in the second graphical representation 850. In some examples, the mixer characterization circuitry 102 identifies a mixing time for a particular mixer based on the power density (P/V) readings. For example, the mixer characterization circuitry 102 determines the mixing time 810 for a mixer with a volume of 52 L, 122 L, and/or 200 L.

[0043] FIG. 9 shows an example first graphical representation 900 and an example second graphical representation 950 of a change in power number based on changes in mixer unit volume. In the example of FIG. 9, the power number is identified for a first mixer type (e.g., 200 L) and a second mixer type (e.g., 1000 L), as represented using the first graphical representation 900 and the second graphical representation 950 (e.g., power number versus volume 905). For example, a power number 910 is determined at different mixing solution volumes 915 of the mixer, with a power number equation 920 determined based on the resulting linear and/or non-linear relationship (e.g., using a Grenville –Nienow equation, etc.). In examples disclosed herein, the power number can be determined based on parameters such as impeller rotational speed (in rotations per second), blend time to 95% reduction in variance, impeller diameter, and/or tank diameter. In the example of FIG. 9, the power number increases at a faster rate for a mixer unit with a smaller overall volume (e.g., 200 L) (e.g., as shown in the first graphical representation 900) as compared to a mixer unit with a larger overall volume (e.g., 1000 L) (e.g., as shown in the second graphical representation 950). In examples disclosed herein, the mixer characterization circuitry 102 performs modeling to identify the power number(s) associated with the input parameters and match a mixing time to the input parameters based on the power number, as described in more detail in connection with FIGS. 13-15.

[0044] FIG. 10 illustrates an example first graphical representation 1050 of a comparison of mathematical modeling-based mixing times 1055 versus computational fluid

dynamics (CFD)-based mixing times 1060 for a smaller mixer 1000. In the example of FIG. 10, the modeling-based mixing times 1055 are matched to the CFD-based mixing times 1060 to verify the accuracy of the mathematical model based on the power number calculations described in connection with FIG. 9 (e.g., with a total deviation of +/- 10% between the model mixing time and the CFD-based mixing time for a smaller unit). For example, the mixer characterization circuitry 102 confirms that the math model mixing time(s) match the CFD-based mixing times during modeling and/or determination of the power number used to determine the mixing time for a particular mixer.

[0045] FIG. 11 illustrates an example second graphical representation 1150 of a corresponding comparison of mathematical modeling-based mixing times 1055 versus CFD-based mixing times 1060 for a larger mixer 1100. In the examples of FIGS. 10 and 11, the accuracy of the model improves with smaller mixer unit volumes (e.g., with a total deviation of +/- 20% between the model mixing time and the CFD-based mixing time for a larger unit). In some examples, the mixer characterization circuitry 102 compares the math model mixing time(s) to the CFD-based mixing time(s) for mixers of various volumes to determine whether there are any deviations between the mixing times.

[0046] FIG. 12 is a block diagram of an example implementation of the mixer characterization circuitry 102 of FIG. 1. The mixer characterization circuitry 102 of FIG. 1 may be instantiated (e.g., creating an instance of, bring into being for any length of time, materialize, implement, etc.) by programmable circuitry such as a Central Processing Unit (CPU) executing first instructions. Additionally or alternatively, the mixer characterization circuitry 102 of FIG. 1 may be instantiated (e.g., creating an instance of, bring into being for any length of time, materialize, implement, etc.) by (i) an Application Specific Integrated Circuit (ASIC) and/or (ii) a Field Programmable Gate Array (FPGA) structured and/or configured in response to execution of second instructions to perform operations corresponding to the first instructions. It should be understood that some or all of the circuitry of FIG. 12 may, thus, be instantiated at the same or different times. Some or all of the circuitry of FIG. 12 may be instantiated, for example, in one or more threads executing concurrently on hardware and/or in series on hardware. Moreover, in some examples, some or all of the circuitry of FIG. 12 may be implemented by microprocessor circuitry executing instructions and/or FPGA circuitry performing operations to implement one or more virtual machines and/or containers.

[0047] In the example of FIG. 12, the mixer characterization circuitry 102 includes example input receiver circuitry 1202, example power number identifier circuitry 1204,

example mixing time identifier circuitry 1206, example feed placement identifier circuitry 1208, example homogeneity identifier circuitry 1210, example bioreactor transfer optimizer circuitry 1212, and/or example data storage 1216. In the example of FIG. 12, the input receiver circuitry 1202, the power number identifier circuitry 1204, the mixing time identifier circuitry 1206, the feed placement identifier circuitry 1208, the homogeneity identifier circuitry 1210, the bioreactor transfer optimizer circuitry 1212, and/or the data storage 1216 are in communication via an example bus 1220.

[0048] The input receiver circuitry 1202 receives input parameters provided by the user. For example, the input receiver circuitry 1202 receives reference input parameters such as mixer size, working volume, and/or impeller rotational speed. In some examples, the input receiver circuitry 1202 receives input such as blend time to 95% reduction in variance, impeller diameter, and/or tank diameter. For example, the user can enter mixer-based input parameters using the user interface display 130 of FIG. 1, as previously described. In some examples, the input receiver circuitry 1202 receives parameters that are used for developing a model that identifies mixing time based on the type of selected mixer (e.g., using an identified power number). In some examples, the input receiver circuitry 1202 identifies data (e.g., velocity contours, tracer flow lines, etc.) obtained during testing of various mixers (e.g., using virtual probes positioned throughout the mixer). In some examples, the input receiver circuitry 1202 receives steady state torque analysis data for each mixer volume. For example, input data associated with each mixer provides information about mixer parameters that are used to further determine the associated mixing speed that facilitates the mixing of various bioprocessing-based fluids (e.g., mixing regulation for medium and buffer preparation, cell culture, fermentation, and/or virus inactivation).

[0049] The power number identifier circuitry 1204 identifies the power number used to estimate impeller speeds for a desired power density (P/V) ratio. For example, the power number identifier circuitry 1204 identifies the power number based on factors such as impeller rotational speed (in rotations per second), blend time to 95% reduction in variance, impeller diameter, and/or tank diameter. In some examples, the power number identifier circuitry 1204 uses the Grenville-Nienow equation to determine the power number based on the mixer unit input parameters. As previously shown, the power number equation can be derived from the relationship between the power number and mixer unit volume, as described in connection with FIG. 9. As such, the power number identifier circuitry 1204 identifies the power number when a user enters information about the type of mixer being used (e.g., mixer size and type), as shown in connection with the user interface display 130 of FIG. 1. In

examples disclosed herein, the power number identifier circuitry 1204 performs modeling to identify the power number and match mixing time to input parameters based on a characterization of the mixing time and power number for different mixer type(s). For example, the power number identifier circuitry 1204 models the power number and mixing time parameter(s) using computational fluid dynamics, as described in connection with FIGS. 6-8. For example, identification of the power number provides a direct connection between a given mixer and the identified mixing time associated with that mixer, allowing for an organized and well-planned approach to bioprocessing based on the mixer selection.

[0050] The mixing time identifier circuitry 1206 identifies the mixing time based on a power number generated by the power number identifier circuitry 1204, as shown in connection with the user interface display 130 of FIG. 1. For example, the mixing time identifier circuitry 1206 determines the T95 mixing time (e.g., calculated based on an analytic reaching 95% of the final recorded stable value) or the T98 mixing time (e.g., calculated based on an analytic reaching 98% of the final recorded stable value) using the generated power number. In some examples, the mixing time identifier circuitry 1206 identifies deviations in the mixing time determined using the computational fluid dynamics (CFD) model and a mixing time determined using the mathematical model (e.g., deviations obtained for mixers of various capacities). For example, identification of the mixing time provides the time needed for homogenization of fluids through the elimination of concentration gradients and temperature variations, improving the efficiency and predictability of this bioprocessing step.

[0051] The feed placement identifier circuitry 1208 determines sequencing of feed addition to mixing media based on identified mixing time values. For example, the feed placement identifier circuitry 1208 sequences and/or automates the addition of acids, bases, and/or salts based on predetermined mixing time values determined using the mixing time identifier circuitry 1206. In examples disclosed herein, the feed placement identifier circuitry 1208 identifies a desired location for adding feed during the mixing process (e.g., based on particle properties). For example, as described in connection with FIG. 3, the mixing time identifier circuitry 1206 identifies the mixing time associated with mixing an added sodium chloride solution. For example, the feed placement identifier circuitry 1208 identifies a location for adding the feed to avoid dead spots, where small amounts of ingredients are not readily incorporated into the feed. In some examples, the feed placement identifier circuitry 1208 identifies multiple points of feed application to promote adequate dispersion. For example, the specific location of the feed placement improves the overall mixing process by

taking into account factors such as the time required to reach homogeneity, the sequencing of additives based on mixing times, and/or the mixing speed used to maintain low shear and/or uniform dispersion.

[0052] The homogeneity identifier circuitry 1210 identifies the time to achieve complete dispersion of proteins and/or aggregates at a given mixing speed. For example, the homogeneity identifier circuitry 1210 determines homogeneity times for various mixer units based on virtual probe positioning within the mixers, as shown in connection with FIG. 5. In some examples, the homogeneity identifier circuitry 1210 identifies the time to achieve dispersion based on an assessment of fluid dynamics captured using the virtual probes. In some examples, the homogeneity identifier circuitry 1210 receives user-based input indicating the type and/or amount of feed that will be added during the mixing process. In some examples, the homogeneity identifier circuitry 1210 maintains low shear for proteins before chromatography by determining revolutions per minute to achieve low shear and uniform dispersion. For example, the homogeneity identifier circuitry 1210 determines whether low shear needs to be maintained based on a user's determination that chromatography will be performed (e.g., when using cell aggregate suspensions and/or proteins). In examples disclosed herein, the homogeneity identifier circuitry 1210 determines the time required to achieve a homogenous solution. Contrary to existing methods, where dead zones are not captured (e.g., homogeneity is measured at sensor locations only and regions of slow mixing are not captured by virtual probes), methods and apparatus disclosed herein allow for an automatic identification of the time required to achieve a homogenous solution, which introduces improved efficiency and reliability. For example, compared to the use of virtual probes, slow mixing regions are considered in the time needed to achieve homogeneity, allowing users to plan bioprocess activities in a more efficient manner. Likewise, current methods rely on pH meters and/or conductivity probes. Mixing is checked by taking samples offline and performing a visual check for haziness and/or using turbidity sensors. Poor mixing can result in clumps of non-mixed media powders that may only be revealed after draining the solution (e.g., resulting from uneven mixing). In examples disclosed herein, intermittent visual checks are eliminated through the use of a user display interface that displays the time required to achieve homogeneity and/or uniform dispersion (e.g., time required for particle suspension).

[0053] The bioreactor transfer optimizer circuitry 1212 identifies time points at which media from the mixer can be transferred to the bioreactor. For example, the bioreactor transfer optimizer circuitry 1212 automates the media transfer based on predetermined

mixing time values identified using the mixing time identifier circuitry 1206. As shown in the example of FIG. 1, media from the mixer 105 can be transferred to the bioreactor 135 once the mixing process is complete. The bioreactor transfer optimizer circuitry 1212 identifies the bioreactor unit to transfer the mixing media to and automates the transfer process based on an identification of bioreactor 135 location. For example, transfer of the media directly from the mixer to the bioreactor based on a determined mixing time completion improves the efficiency of the transfer process during upstream processing (e.g., initiating biological reactions under controlled conditions for a variety of purposes, including the production of pharmaceuticals, vaccines, antibodies, and biofuel).

[0054] The data storage 1216 can be used to store any information associated with the input receiver circuitry 1202, the power number identifier circuitry 1204, the mixing time identifier circuitry 1206, the feed placement identifier circuitry 1208, the homogeneity identifier circuitry 1210, and/or the bioreactor transfer optimizer circuitry 1212. The example data storage 1216 of the illustrated example of FIG. 12 can be implemented by any memory, storage device and/or storage disc for storing data such as flash memory, magnetic media, optical media, etc. Furthermore, the data stored in the example data storage 1216 can be in any data format such as binary data, comma delimited data, tab delimited data, structured query language (SQL) structures, image data, etc.

[0055] In some examples, the apparatus includes means for receiving input. For example, the means for receiving input may be implemented by input receiver circuitry 1202. In some examples, the input receiver circuitry 1202 may be instantiated by programmable circuitry such as the example programmable circuitry 1612 of FIG. 16. For instance, the input receiver circuitry 1202 may be instantiated by the example microprocessor 1700 of FIG. 17 executing machine executable instructions such as those implemented by at least block 1305 of FIG. 13. In some examples, the input receiver circuitry 1202 may be instantiated by hardware logic circuitry, which may be implemented by an ASIC, XPU, or the FPGA circuitry 1800 of FIG. 18 structured to perform operations corresponding to the machine readable instructions. Additionally or alternatively, the input receiver circuitry 1202 may be instantiated by any other combination of hardware, software, and/or firmware. For example, the input receiver circuitry 1202 may be implemented by at least one or more hardware circuits (e.g., processor circuitry, discrete and/or integrated analog and/or digital circuitry, an FPGA, an ASIC, an XPU, a comparator, an operational-amplifier (op-amp), a logic circuit, etc.) structured to execute some or all of the machine readable instructions and/or to perform

some or all of the operations corresponding to the machine readable instructions without executing software or firmware, but other structures are likewise appropriate.

[0056] In some examples, the apparatus includes means for identifying a power number. For example, the means for identifying a power number may be implemented by power number identifier circuitry 1204. In some examples, the power number identifier circuitry 1204 may be instantiated by programmable circuitry such as the example programmable circuitry 1612 of FIG. 16. For instance, the power number identifier circuitry 1204 may be instantiated by the example microprocessor 1700 of FIG. 17 executing machine executable instructions such as those implemented by at least block 1310 of FIG. 13. In some examples, the power number identifier circuitry 1204 may be instantiated by hardware logic circuitry, which may be implemented by an ASIC, XPU, or the FPGA circuitry 1800 of FIG. 18 structured to perform operations corresponding to the machine readable instructions. Additionally or alternatively, the power number identifier circuitry 1204 may be instantiated by any other combination of hardware, software, and/or firmware. For example, the power number identifier circuitry 1204 may be implemented by at least one or more hardware circuits (e.g., processor circuitry, discrete and/or integrated analog and/or digital circuitry, an FPGA, an ASIC, an XPU, a comparator, an operational-amplifier (op-amp), a logic circuit, etc.) structured to execute some or all of the machine readable instructions and/or to perform some or all of the operations corresponding to the machine readable instructions without executing software or firmware, but other structures are likewise appropriate.

[0057] In some examples, the apparatus includes means for identifying a mixing time. For example, the means for identifying a mixing time may be implemented by mixing time identifier circuitry 1206. In some examples, the mixing time identifier circuitry 1206 may be instantiated by programmable circuitry such as the example programmable circuitry 1612 of FIG. 16. For instance, the mixing time identifier circuitry 1206 may be instantiated by the example microprocessor 1700 of FIG. 17 executing machine executable instructions such as those implemented by at least block 1320 of FIG. 13. In some examples, the mixing time identifier circuitry 1206 may be instantiated by hardware logic circuitry, which may be implemented by an ASIC, XPU, or the FPGA circuitry 1800 of FIG. 18 structured to perform operations corresponding to the machine readable instructions. Additionally or alternatively, the mixing time identifier circuitry 1206 may be instantiated by any other combination of hardware, software, and/or firmware. For example, the mixing time identifier circuitry 1206 may be implemented by at least one or more hardware circuits (e.g., processor circuitry, discrete and/or integrated analog and/or digital circuitry, an FPGA, an ASIC, an XPU, a

comparator, an operational-amplifier (op-amp), a logic circuit, etc.) structured to execute some or all of the machine readable instructions and/or to perform some or all of the operations corresponding to the machine readable instructions without executing software or firmware, but other structures are likewise appropriate.

[0058] In some examples, the apparatus includes means for identifying a feed placement. For example, the means for identifying a feed placement may be implemented by feed placement identifier circuitry 1208. In some examples, the feed placement identifier circuitry 1208 may be instantiated by programmable circuitry such as the example programmable circuitry 1612 of FIG. 16. For instance, the feed placement identifier circuitry 1208 may be instantiated by the example microprocessor 1700 of FIG. 17 executing machine executable instructions such as those implemented by at least block 1525 of FIG. 15. In some examples, the feed placement identifier circuitry 1208 may be instantiated by hardware logic circuitry, which may be implemented by an ASIC, XPU, or the FPGA circuitry 1800 of FIG. 18 structured to perform operations corresponding to the machine readable instructions. Additionally or alternatively, the feed placement identifier circuitry 1208 may be instantiated by any other combination of hardware, software, and/or firmware. For example, the feed placement identifier circuitry 1208 may be implemented by at least one or more hardware circuits (e.g., processor circuitry, discrete and/or integrated analog and/or digital circuitry, an FPGA, an ASIC, an XPU, a comparator, an operational-amplifier (op-amp), a logic circuit, etc.) structured to execute some or all of the machine readable instructions and/or to perform some or all of the operations corresponding to the machine readable instructions without executing software or firmware, but other structures are likewise appropriate.

[0059] In some examples, the apparatus includes means for identifying homogeneity. For example, the means for identifying homogeneity may be implemented by homogeneity identifier circuitry 1210. In some examples, the homogeneity identifier circuitry 1210 may be instantiated by programmable circuitry such as the example programmable circuitry 1612 of FIG. 16. For instance, the homogeneity identifier circuitry 1210 may be instantiated by the example microprocessor 1700 of FIG. 17 executing machine executable instructions such as those implemented by at least block 1510 of FIG. 15. In some examples, the homogeneity identifier circuitry 1210 may be instantiated by hardware logic circuitry, which may be implemented by an ASIC, XPU, or the FPGA circuitry 1800 of FIG. 18 structured to perform operations corresponding to the machine readable instructions. Additionally or alternatively, the homogeneity identifier circuitry 1210 may be instantiated by any other combination of hardware, software, and/or firmware. For example, the homogeneity identifier circuitry 1210

may be implemented by at least one or more hardware circuits (e.g., processor circuitry, discrete and/or integrated analog and/or digital circuitry, an FPGA, an ASIC, an XPU, a comparator, an operational-amplifier (op-amp), a logic circuit, etc.) structured to execute some or all of the machine readable instructions and/or to perform some or all of the operations corresponding to the machine readable instructions without executing software or firmware, but other structures are likewise appropriate.

[0060] In some examples, the apparatus includes means for bioreactor transfer. For example, the means for bioreactor transfer may be implemented by bioreactor transfer optimizer circuitry 1212. In some examples, the bioreactor transfer optimizer circuitry 1212 may be instantiated by programmable circuitry such as the example programmable circuitry 1612 of FIG. 16. For instance, the bioreactor transfer optimizer circuitry 1212 may be instantiated by the example microprocessor 1700 of FIG. 17 executing machine executable instructions such as those implemented by at least block 1540 of FIG. 15. In some examples, the bioreactor transfer optimizer circuitry 1212 may be instantiated by hardware logic circuitry, which may be implemented by an ASIC, XPU, or the FPGA circuitry 1800 of FIG. 18 structured to perform operations corresponding to the machine readable instructions. Additionally or alternatively, the bioreactor transfer optimizer circuitry 1212 may be instantiated by any other combination of hardware, software, and/or firmware. For example, the bioreactor transfer optimizer circuitry 1212 may be implemented by at least one or more hardware circuits (e.g., processor circuitry, discrete and/or integrated analog and/or digital circuitry, an FPGA, an ASIC, an XPU, a comparator, an operational-amplifier (op-amp), a logic circuit, etc.) structured to execute some or all of the machine readable instructions and/or to perform some or all of the operations corresponding to the machine readable instructions without executing software or firmware, but other structures are likewise appropriate.

[0061] While an example implementation of the mixer characterization circuitry 102 of FIG. 1 is illustrated in FIG. 12, one or more of the elements, processes and/or devices illustrated in FIG. 12 may be combined, divided, re-arranged, omitted, eliminated and/or implemented in any other way. Further, the example input receiver circuitry 1202, the example power number identifier circuitry 1204, the example mixing time identifier circuiry 1206, the example feed placement identifier circuitry 1208, the example homogeneity identifier circuiry 1210, the example bioreactor transfer optimizer circuitry 1212, and/or, more generally, the example mixer characterization circuitry 102 of FIG. 1 may be implemented by hardware, software, firmware and/or any combination of hardware, software

and/or firmware. Thus, for example, any of the example input receiver circuitry 1202, the example power number identifier circuitry 1204, the example mixing time identifier circuiry 1206, the example feed placement identifier circuitry 1208, the example homogeneity identifier circuiry 1210, the example bioreactor transfer optimizer circuitry 1212, and/or, more generally, the example mixer characterization circuitry 102 of FIG. 1 could be implemented by programmable circuitry in combination with machine readable instructions (e.g., firmware or software), processor circuitry, analog circuit(s), digital circuit(s), logic circuit(s), programmable processor(s), programmable microcontroller(s), graphics processing unit(s) (GPU(s)), digital signal processor(s) (DSP(s), ASIC(s)), programmable logic device(s) (PLD(s)), and/or field programmable logic device(s) (FPLD(s)) such as FPGAs. Further still, the mixer characterization circuitry 102 of FIG. 1 may include one or more elements, processes, and/or devices in addition to, or instead of, those illustrated in FIG. 12, and/or may include more than one of any or all of the illustrated elements, processes and devices.

[0062] Flowcharts representative of example machine readable instructions, which may be executed by programmable circuitry to implement and/or instantiate the mixer characterization circuitry 102 of FIG. 1 and/or representative of example operations which may be performed by programmable circuitry to implement and/or instantiate the mixer characterization circuitry 102 of FIG. 1, are shown in FIGS. 13-15. The machine readable instructions may be one or more executable programs or portion(s) of one or more executable programs for execution by programmable circuitry, such as the programmable circuitry 1612 shown in the example processor platform 1600 discussed below in connection with FIG. 16 and/or may be one or more function(s) or portion(s) of functions to be performed by the example programmable circuitry (e.g., an FPGA) discussed below in connection with FIGS. 17 and/or 18. In some examples, the machine readable instructions cause an operation, a task, etc., to be carried out and/or performed in an automated manner in the real world. As used herein, "automated" means without human involvement.

[0063] The program may be embodied in instructions (e.g., software and/or firmware) stored on one or more non-transitory computer readable and/or machine readable storage medium such as cache memory, a magnetic-storage device or disk (e.g., a floppy disk, a Hard Disk Drive (HDD), etc.), an optical-storage device or disk (e.g., a Blu-ray disk, a Compact Disk (CD), a Digital Versatile Disk (DVD), etc.), a Redundant Array of Independent Disks (RAID), a register, ROM, a solid-state drive (SSD), SSD memory, non-volatile memory (e.g., electrically erasable programmable read-only memory (EEPROM), flash memory, etc.), volatile memory (e.g., Random Access Memory (RAM) of any type, etc.), and/or any other

storage device or storage disk. The instructions of the non-transitory computer readable and/or machine readable medium may program and/or be executed by programmable circuitry located in one or more hardware devices, but the entire program and/or parts thereof could alternatively be executed and/or instantiated by one or more hardware devices other than the programmable circuitry and/or embodied in dedicated hardware. The machine readable instructions may be distributed across multiple hardware devices and/or executed by two or more hardware devices (e.g., a server and a client hardware device). For example, the client hardware device may be implemented by an endpoint client hardware device (e.g., a hardware device associated with a human and/or machine user) or an intermediate client hardware device gateway (e.g., a radio access network (RAN)) that may facilitate communication between a server and an endpoint client hardware device. Similarly, the non-transitory computer readable storage medium may include one or more mediums. Further, although the example program is described with reference to the flowcharts illustrated in FIGS. 13-15, many other methods of implementing the example mixer characterization circuitry 102 of FIG. 1 may alternatively be used. For example, the order of execution of the blocks of the flowchart(s) may be changed, and/or some of the blocks described may be changed, eliminated, or combined. Additionally or alternatively, any or all of the blocks of the flow chart may be implemented by one or more hardware circuits (e.g., processor circuitry, discrete and/or integrated analog and/or digital circuitry, an FPGA, an ASIC, a comparator, an operational-amplifier (op-amp), a logic circuit, etc.) structured to perform the corresponding operation without executing software or firmware. The programmable circuitry may be distributed in different network locations and/or local to one or more hardware devices (e.g., a single-core processor (e.g., a single core CPU), a multi-core processor (e.g., a multi-core CPU, an XPU, etc.)). For example, the programmable circuitry may be a CPU and/or an FPGA located in the same package (e.g., the same integrated circuit (IC) package or in two or more separate housings), one or more processors in a single machine, multiple processors distributed across multiple servers of a server rack, multiple processors distributed across one or more server racks, etc., and/or any combination(s) thereof.

[0064] The machine readable instructions described herein may be stored in one or more of a compressed format, an encrypted format, a fragmented format, a compiled format, an executable format, a packaged format, etc. Machine readable instructions as described herein may be stored as data (e.g., computer-readable data, machine-readable data, one or more bits (e.g., one or more computer-readable bits, one or more machine-readable bits, etc.),

a bitstream (e.g., a computer-readable bitstream, a machine-readable bitstream, etc.), etc.) or a data structure (e.g., as portion(s) of instructions, code, representations of code, etc.) that may be utilized to create, manufacture, and/or produce machine executable instructions. For example, the machine readable instructions may be fragmented and stored on one or more storage devices, disks and/or computing devices (e.g., servers) located at the same or different locations of a network or collection of networks (e.g., in the cloud, in edge devices, etc.). The machine readable instructions may require one or more of installation, modification, adaptation, updating, combining, supplementing, configuring, decryption, decompression, unpacking, distribution, reassignment, compilation, etc., in order to make them directly readable, interpretable, and/or executable by a computing device and/or other machine. For example, the machine readable instructions may be stored in multiple parts, which are individually compressed, encrypted, and/or stored on separate computing devices, wherein the parts when decrypted, decompressed, and/or combined form a set of computer-executable and/or machine executable instructions that implement one or more functions and/or operations that may together form a program such as that described herein.

[0065] In another example, the machine readable instructions may be stored in a state in which they may be read by programmable circuitry, but require addition of a library (e.g., a dynamic link library (DLL)), a software development kit (SDK), an application programming interface (API), etc., in order to execute the machine-readable instructions on a particular computing device or other device. In another example, the machine readable instructions may need to be configured (e.g., settings stored, data input, network addresses recorded, etc.) before the machine readable instructions and/or the corresponding program(s) can be executed in whole or in part. Thus, machine readable, computer readable and/or machine readable media, as used herein, may include instructions and/or program(s) regardless of the particular format or state of the machine readable instructions and/or program(s).

[0066] The machine readable instructions described herein can be represented by any past, present, or future instruction language, scripting language, programming language, etc. For example, the machine readable instructions may be represented using any of the following languages: C, C++, Java, C#, Perl, Python, JavaScript, HyperText Markup Language (HTML), Structured Query Language (SQL), Swift, etc.

[0067] As mentioned above, the example operations of FIGS. 13-15 may be implemented using executable instructions (e.g., computer readable and/or machine readable instructions) stored on one or more non-transitory computer readable and/or machine readable media. As used herein, the terms non-transitory computer readable medium, non-

transitory computer readable storage medium, non-transitory machine readable medium, and/or non-transitory machine readable storage medium are expressly defined to include any type of computer readable storage device and/or storage disk and to exclude propagating signals and to exclude transmission media. Examples of such non-transitory computer readable medium, non-transitory computer readable storage medium, non-transitory machine readable medium, and/or non-transitory machine readable storage medium include optical storage devices, magnetic storage devices, an HDD, a flash memory, a read-only memory (ROM), a CD, a DVD, a cache, a RAM of any type, a register, and/or any other storage device or storage disk in which information is stored for any duration (e.g., for extended time periods, permanently, for brief instances, for temporarily buffering, and/or for caching of the information). As used herein, the terms "non-transitory computer readable storage device" and "non-transitory machine readable storage device" are defined to include any physical (mechanical, magnetic and/or electrical) hardware to retain information for a time period, but to exclude propagating signals and to exclude transmission media. Examples of non-transitory computer readable storage devices and/ or non-transitory machine readable storage devices include random access memory of any type, read only memory of any type, solid state memory, flash memory, optical discs, magnetic disks, disk drives, and/or redundant array of independent disks (RAID) systems. As used herein, the term "device" refers to physical structure such as mechanical and/or electrical equipment, hardware, and/or circuitry that may or may not be configured by computer readable instructions, machine readable instructions, etc., and/or manufactured to execute computer-readable instructions, machine-readable instructions, etc.

[0068] "Including" and "comprising" (and all forms and tenses thereof) are used herein to be open ended terms. Thus, whenever a claim employs any form of "include" or "comprise" (e.g., comprises, includes, comprising, including, having, etc.) as a preamble or within a claim recitation of any kind, it is to be understood that additional elements, terms, etc., may be present without falling outside the scope of the corresponding claim or recitation. As used herein, when the phrase "at least" is used as the transition term in, for example, a preamble of a claim, it is open-ended in the same manner as the term "comprising" and "including" are open ended. The term "and/or" when used, for example, in a form such as A, B, and/or C refers to any combination or subset of A, B, C such as (1) A alone, (2) B alone, (3) C alone, (4) A with B, (5) A with C, (6) B with C, or (7) A with B and with C. As used herein in the context of describing structures, components, items, objects and/or things, the phrase "at least one of A and B" is intended to refer to implementations

including any of (1) at least one A, (2) at least one B, or (3) at least one A and at least one B. Similarly, as used herein in the context of describing structures, components, items, objects and/or things, the phrase "at least one of A or B" is intended to refer to implementations including any of (1) at least one A, (2) at least one B, or (3) at least one A and at least one B. As used herein in the context of describing the performance or execution of processes, instructions, actions, activities and/or steps, the phrase "at least one of A and B" is intended to refer to implementations including any of (1) at least one A, (2) at least one B, or (3) at least one A and at least one B. Similarly, as used herein in the context of describing the performance or execution of processes, instructions, actions, activities and/or steps, the phrase "at least one of A or B" is intended to refer to implementations including any of (1) at least one A, (2) at least one B, or (3) at least one A and at least one B.

[0069] As used herein, singular references (e.g., "a", "an", "first", "second", etc.) do not exclude a plurality. The term "a" or "an" object, as used herein, refers to one or more of that object. The terms "a" (or "an"), "one or more", and "at least one" are used interchangeably herein. Furthermore, although individually listed, a plurality of means, elements, or actions may be implemented by, e.g., the same entity or object. Additionally, although individual features may be included in different examples or claims, these may possibly be combined, and the inclusion in different examples or claims does not imply that a combination of features is not feasible and/or advantageous.

[0070] FIG. 13 is a flowchart representative of example machine readable instructions and/or example operations 1300 that may be executed, instantiated, and/or performed by programmable circuitry to implement the example mixer characterization circuitry 102 of FIG. 1. The machine readable instructions and/or the operations 1300 of FIG. 13 begin at block 1305, at which the input receiver circuitry 1202 receives input parameters such as mixer size, working volume, and/or impeller speed. In some examples, the input receiver circuitry 1202 receives the impeller speed information from virtual probes positioned within the mixing vessel(s), as shown in connection with FIG. 5. Once the input receiver circuitry 1202 receives the input parameters, the power number identifier circuitry 1204 performs modeling to identify the power number and match the mixing time to the input parameters, at block 1310. For example, the power number identifier circuitry 1204 characterizes the mixing time and the power number for different mixer type(s), as described in more detail in connection with FIG. 14. Once the power number identifier circuitry 1204 matches input parameter(s) to specific mixing times, the input receiver circuitry 1202 receives user-selected

inputs via the user interface display 130 of FIG. 1 (e.g., mixer size, working volume, impeller speed, etc.), at block 1315.

[0071] In some examples, the input receiver circuitry 1202 receives inputs identifying a type of feed to add to the mixer and/or whether to initiate a transfer to a bioreactor after the mixing process is completed. The mixing time identifier circuitry 1206 proceeds to predict the mixing time based on the user-selected parameters, while the feed placement identifier circuitry 1208 identifies desired feed locations based on the type and/or quantity of feed, at block 1320, as described in more detail in connection with FIG. 14. In some examples, the input receiver circuitry 1202 generates view(s) of velocity contour(s) and particle flow(s), at block 1322. For example, as shown in connection with FIG. 1, the input receiver circuitry 1202 generates real-time views and/or time interval-based views of the velocity contour(s) and particle flow(s) before and/or during the mixing process, allowing the user to visualize the fluid dynamics associated with a particular mixer (e.g., via the user interface display 130 of FIG. 1). Once the mixing time identifier circuitry 1206 identifies and displays the mixing time to the user (e.g., via the user interface display 130), the mixing time identifier circuitry 1206 initiates the mixing process, at block 1325. In some examples, the bioreactor transfer optimizer circuitry 1212 determines whether to initiate an automatic transfer of media from the mixer (e.g., mixer 105 of FIG. 1) to the bioreactor (e.g., bioreactor 135 of FIG. 1) once the mixing process is completed, at block 1330. For example, if the user indicates that an automatic transfer of the media should occur, the bioreactor transfer optimizer circuitry 1212 completes the media transfer to the bioreactor, at block 1335. For example, input parameters received at the user interface display 130 can vary in the working volume and impeller speed, based on the mixer selected. When the working volume is larger (e.g., 200L), the mixing time increases and the corresponding velocity contours and particles flows are displayed for the larger working volume mixer (e.g., tracer flow line 605 of FIG. 6). Subsequent transfer to the bioreactor can require a longer time, and will depend on factors such as whether feed is added to the mixer during the mixing process. For a smaller working volume (e.g., 52L), the mixing time will decrease, and the corresponding velocity contours and particles flows will be generated to match the lower-volume mixer type (e.g., tracer flow line 625 of FIG. 6).

[0072] FIG. 14 is a flowchart representative of example machine readable instructions and/or example operations 1310 that may be executed, instantiated, and/or performed by programmable circuitry to implement the example mixer characterization circuitry 102 of FIG. 1 to perform modeling to identify the power number and match mixing time to input parameters, in accordance with teachings disclosed herein. In the example of FIG. 14, the

input receiver circuitry 1202 identifies virtual probe(s) positioned within the mixer unit(s) to track fluid dispersion, at block 1405. For example, the virtual probes (e.g., P1, P2, P3, P4) positioned within the mixer 300 of FIG. 3 can be used to obtain experimental values (e.g., T98 values) to determine the experimental mixing time. Use of the virtual probes allows for accurate predictions of the mixing time based on the input parameters. Based on this data, the mixing time identifier circuitry 1210 determines experimental values associated with mixing times for a variety of mixer(s), as shown in connection with FIGS. 6-7. In the example of FIG. 14, the power number identifier circuitry 1204 determines the power number based on input parameters (e.g., input power, media density, impeller speed, and/or impeller diameter), at block 1408. The mixing time identifier circuitry 1210 proceeds to characterize the mixing time and power number for various mixer type(s), at block 1410. For example, the mixing times are matched to specific mixer units of various volumes (e.g., 52 L, 200 L, etc.). The power number identifier circuitry 1204 models the power number and mixing time parameter(s) using computational fluid dynamics (CFD), at block 1415. For example, CFD-based studies can be used to predict mixing performance of a mixer in solid-liquid, liquid-liquid, high viscosity mixing and/or hard to resolve solid-liquid mixing. As such, a digital version of mixers with various impeller designs and/or operation parameters is obtained using methods and apparatus disclosed herein. The resulting data is incorporated into a user interface (e.g., user interface display 130 of FIG. 1) to allow users to determine optimum mixer-based designs without extended wet lab work. For example, using a 200 L mixer will result in a different mixing time than using a 52 L mixer, and the mixing time is determined after input parameters are provided, without the need for further experimentation and/or validation. Additionally, the power number identifier circuitry 1204 fits the generated data to confirm mixer parameters by evaluating the total deviations between determined model-based mixing times and experimental mixing times identified using virtual probes, at block 1420 (e.g., as shown in connection with FIGS. 10-11). In some examples, the power number identifier circuitry 1204 post-processes CFD-based data and uses the Grenville-Nienow equation to fit the data to predict the mixing time and power number. For example, the coefficients of the Grenville-Nienow equation are fitted to the generated CFD data. As such, inputs of a mixer size, volume, and/or agitation speed can be received at the user interface display 130 of FIG. 1 (e.g., from a user, another program, system, etc.) for predictions of the power number and mixing time.

[0073] FIG. 15 is a flowchart representative of example machine readable instructions and/or example operations 1320 that may be executed, instantiated, and/or performed by

programmable circuitry to implement the example mixer characterization circuitry 102 of FIG. 1 to perform modeling predict mixing time and feed location based on user-selected parameters, in accordance with the teachings disclosed herein. In the example of FIG. 15, the homogeneity identifier circuitry 1210 determines whether to identify the time needed to achieve complete dispersion of proteins and/or aggregates at any impeller speed, at block 1505. For example, the homogeneity identifier circuitry 1210 determines the time needed to achieve homogeneity (e.g., uniform distribution of particles in the mixture fluid), at block 1510. The homogeneity identifier circuitry 1210 displays the resulting time required for complete dispersion using the user interface display 130 of FIG. 1. For example, homogeneity eliminates concentration gradients and/or temperature variations as the velocity of the fluid moved by the impeller distributes the fluid throughout all areas of the tank. In some examples, the user specifies the feed to add to the mixer during the mixing process (e.g., addition of acids, bases, and/or salts). As such, the feed placement identifier circuitry 1208 determines the availability and/or type of feed to add to the mixer, at block 1515. The feed placement identifier circuitry 1208 proceeds to determine sequencing of the acid, base, and/or salt addition based on identified mixing time value(s), at block 1520. Furthermore, the feed placement identifier circuitry 1208 identifies the location(s) for adding the feed, at block 1525. For example, feed can be added to a mixer and/or bioreactor using feed lines made of silicone (e.g., for liquid addition), providing control over the mixing fluid composition. In some examples, the homogeneity identifier circuitry 1210 determines whether low shear needs to be maintained for a particular mixture (e.g., mixture containing proteins), at block 1530. For example, low shear stress can be tolerated by proteins, where as high shear provokes protein damage. CFD analysis can be used to identify favorable mixing limits for biological systems that are sensitive to shear stress. In particular, animal cell lines are sensitive to shear stress in bioprocess-based production and can be easily ruptured when exposed to intense physical forces. As such, identification of whether low shear is needed for a particular feed addition is critical to ensuring successful bioprocess completion. In some examples, the homogeneity identifier circuitry 1210 identifies the type of fluid in the mixer based on a user-initiated entry using the user interface display 130 of FIG. 1. When low shear is required, the homogeneity identifier circuitry 1210 determines the speed (e.g., revolutions per minute, rpm) to maintain low shear and achieve uniform dispersion, at block 1535. If low shear is not needed, the bioreactor transfer optimizer circuitry 1212 estimates the time needed prior to automatic transfer of the media to the bioreactor (e.g., based on mixing time values), at block 1540. In examples disclosed herein, mixing time and power number prediction is

generated using input parameters associated with a given mixer. Unlike known methods, which require experimentation to determine an exact mixing time associated with a particular mixer, methods and apparatus disclosed herein allow for the instantaneous generation of the mixing time for a particular mixer (e.g., based on mixer volume and impeller speed). Furthermore, the mixing process can be regulated based on the type of feed that is added to the mixer (e.g., acid, base, etc.), while addition of cells to the mixer results in regulation of shear forces to maintain cell viability during the mixing process.

[0074] FIG. 16 is a block diagram of an example programmable circuitry platform 1600 structured to execute and/or instantiate the example machine-readable instructions and/or the example operations of FIGS. 13-15 to implement the example mixer characterization circuitry 102 of FIG. 1. The programmable circuitry platform 1600 can be, for example, a server, a personal computer, a workstation, a self-learning machine (e.g., a neural network), a mobile device (e.g., a cell phone, a smart phone, a tablet such as an iPad™), a personal digital assistant (PDA), an Internet appliance, a DVD player, a CD player, a digital video recorder, a Blu-ray player, a gaming console, a personal video recorder, a set top box, a headset (e.g., an augmented reality (AR) headset, a virtual reality (VR) headset, etc.) or other wearable device, or any other type of computing and/or electronic device.

[0075] The programmable circuitry platform 1600 of the illustrated example includes programmable circuitry 1612. The programmable circuitry 1612 of the illustrated example is hardware. For example, the programmable circuitry 1612 can be implemented by one or more integrated circuits, logic circuits, FPGAs microprocessors, CPUs, GPUs, DSPs, and/or microcontrollers from any desired family or manufacturer. The programmable circuitry 1612 may be implemented by one or more semiconductor based (e.g., silicon based) devices. In this example, the processor circuitry 1612 implements the input receiver circuitry 1202, the power number identifier circuitry 1204, mixing time identifier circuitry 1206, the feed placement identifier circuitry 1208, the homogeneity identifier circuitry 1210, and the bioreactor transfer optimizer circuitry 1212.

[0076] The programmable circuitry 1612 of the illustrated example includes a local memory 1613 (e.g., a cache, registers, etc.). The programmable circuitry 1612 of the illustrated example is in communication with a main memory including a volatile memory 1614 and a non-volatile memory 1616 by a bus 1618. The volatile memory 1614 may be implemented by Synchronous Dynamic Random Access Memory (SDRAM), Dynamic Random Access Memory (DRAM), RAMBUS® Dynamic Random Access Memory

(RDRAM®), and/or any other type of RAM device. The non-volatile memory 1616 may be implemented by flash memory and/or any other desired type of memory device. Access to the main memory 1614, 1616 of the illustrated example is controlled by a memory controller 1617. In some examples, the memory controller 1617 may be implemented by one or more integrated circuits, logic circuits, microcontrollers from any desired family or manufacturer, or any other type of circuitry to manage the flow of data going to and from the main memory 1614, 1616.

[0077] The programmable circuitry platform 1600 of the illustrated example also includes interface circuitry 1620. The interface circuitry 1620 may be implemented by hardware in accordance with any type of interface standard, such as an Ethernet interface, a universal serial bus (USB) interface, a Bluetooth® interface, a near field communication (NFC) interface, a Peripheral Component Interconnect (PCI) interface, and/or a Peripheral Component Interconnect Express (PCIe) interface.

[0078] In the illustrated example, one or more input devices 1622 are connected to the interface circuitry 1620. The input device(s) 1622 permit(s) a user (e.g., a human user, a machine user, etc.) to enter data and/or commands into the programmable circuitry 1612. The input device(s) 1622 can be implemented by, for example, an audio sensor, a microphone, a camera (still or video), a keyboard, a button, a mouse, a touchscreen, a track-pad, a trackball, an isopoint device, and/or a voice recognition system.

[0079] One or more output devices 1624 are also connected to the interface circuitry 1620 of the illustrated example. The output devices 1624 can be implemented, for example, by display devices (e.g., a light emitting diode (LED), an organic light emitting diode (OLED), a liquid crystal display (LCD), a cathode ray tube (CRT) display, an in-place switching (IPS) display, a touchscreen, etc.), a tactile output device, a printer, and/or speaker. The interface circuitry 1620 of the illustrated example, thus, typically includes a graphics driver card, a graphics driver chip, and/or graphics processor circuitry such as a GPU.

[0080] The interface circuitry 1620 of the illustrated example also includes a communication device such as a transmitter, a receiver, a transceiver, a modem, a residential gateway, a wireless access point, and/or a network interface to facilitate exchange of data with external machines (e.g., computing devices of any kind) by a network 1626. The communication can be by, for example, an Ethernet connection, a digital subscriber line (DSL) connection, a telephone line connection, a coaxial cable system, a satellite system, a line-of-site wireless system, a cellular telephone system, an optical connection, etc.

[0081] The programmable circuitry platform 1600 of the illustrated example also includes one or more mass storage devices 1628 to store software and/or data. Examples of such mass storage devices 1628 include magnetic storage devices (e.g., floppy disk, drives, HDDs, etc.), optical storage devices (e.g., Blu-ray disks, CDs, DVDs, etc.), RAID systems, and/or solid-state storage discs or devices such as flash memory devices and/or SSDs.

[0082] The machine executable instructions 1632, which may be implemented by the machine readable instructions of FIGS. 13, 14 and/or 15, may be stored in the mass storage device 1628, in the volatile memory 1614, in the non-volatile memory 1616, and/or on at least one non-transitory computer readable storage medium such as a CD or DVD which may be removable.

[0083] FIG. 17 is a block diagram of an example implementation of the programmable circuitry 1612 of FIG. 16. In this example, the programmable circuitry 1612 of FIG. 16 is implemented by a microprocessor 1700. For example, the microprocessor 1700 may be a general purpose microprocessor (e.g., general purpose microprocessor circuitry). The microprocessor 1700 executes some or all of the machine readable instructions of the flowchart of FIGS. 13, 14 and/or 15 to effectively instantiate the circuitry of FIG. 12 logic circuits to perform the operations corresponding to those machine readable instructions. In some such examples, the circuitry of FIG. 12 is instantiated by the hardware circuits of the microprocessor 1700 in combination with the instructions. For example, the microprocessor 1700 may implement multi-core hardware circuitry such as a CPU, a DSP, a GPU, an XPU, etc. Although it may include any number of example cores 1702 (e.g., 1 core), the microprocessor 1700 of this example is a multi-core semiconductor device including N cores. The cores 1702 of the microprocessor 1700 may operate independently or may cooperate to execute machine readable instructions. For example, machine code corresponding to a firmware program, an embedded software program, or a software program may be executed by one of the cores 1702 or may be executed by multiple ones of the cores 1702 at the same or different times. In some examples, the machine code corresponding to the firmware program, the embedded software program, or the software program is split into threads and executed in parallel by two or more of the cores 1702. The software program may correspond to a portion or all of the machine readable instructions and/or operations represented by the flowcharts of FIGS. 13, 14 and/or 15.

[0084] The cores 1702 may communicate by a first example bus 1704. In some examples, the first bus 1704 may implement a communication bus to effectuate communication associated with one(s) of the cores 1702. For example, the first bus 1704 may

implement at least one of an Inter-Integrated Circuit (I2C) bus, a Serial Peripheral Interface (SPI) bus, a PCI bus, or a PCIe bus. Additionally or alternatively, the first bus 1704 may implement any other type of computing or electrical bus. The cores 1702 may obtain data, instructions, and/or signals from one or more external devices by example interface circuitry 1706. The cores 1702 may output data, instructions, and/or signals to the one or more external devices by the interface circuitry 1706. Although the cores 1702 of this example include example local memory 1720 (e.g., Level 1 (L1) cache that may be split into an L1 data cache and an L1 instruction cache), the microprocessor 1700 also includes example shared memory 1710 that may be shared by the cores (e.g., Level 2 (L2_ cache)) for high-speed access to data and/or instructions. Data and/or instructions may be transferred (e.g., shared) by writing to and/or reading from the shared memory 1710. The local memory 1720 of each of the cores 1702 and the shared memory 1710 may be part of a hierarchy of storage devices including multiple levels of cache memory and the main memory (e.g., the main memory 1614, 1616 of FIG. 16). Typically, higher levels of memory in the hierarchy exhibit lower access time and have smaller storage capacity than lower levels of memory. Changes in the various levels of the cache hierarchy are managed (e.g., coordinated) by a cache coherency policy.

[0085] Each core 1702 may be referred to as a CPU, DSP, GPU, etc., or any other type of hardware circuitry. Each core 1702 includes control unit circuitry 1714, arithmetic and logic (AL) circuitry (sometimes referred to as an ALU) 1716, a plurality of registers 1718, the L1 cache 1720, and a second example bus 1722. Other structures may be present. For example, each core 1702 may include vector unit circuitry, single instruction multiple data (SIMD) unit circuitry, load/store unit (LSU) circuitry, branch/jump unit circuitry, floating-point unit (FPU) circuitry, etc. The control unit circuitry 1714 includes semiconductor-based circuits structured to control (e.g., coordinate) data movement within the corresponding core 1702. The AL circuitry 1716 includes semiconductor-based circuits structured to perform one or more mathematic and/or logic operations on the data within the corresponding core 1702. The AL circuitry 1716 of some examples performs integer-based operations. In other examples, the AL circuitry 1716 also performs floating-point operations. In yet other examples, the AL circuitry 1716 may include first AL circuitry that performs integer-based operations and second AL circuitry that performs floating point operations. In some examples, the AL circuitry 1716 may be referred to as an Arithmetic Logic Unit (ALU).

[0086] The registers 1718 are semiconductor-based structures to store data and/or instructions such as results of one or more of the operations performed by the AL circuitry 1716 of the corresponding core 1702. For example, the registers 1718 may include vector register(s), SIMD register(s), general purpose register(s), flag register(s), segment register(s), machine specific register(s), instruction pointer register(s), control register(s), debug register(s), memory management register(s), machine check register(s), etc. The registers 1718 may be arranged in a bank as shown in FIG. 17. Alternatively, the registers 1718 may be organized in any other arrangement, format, or structure including distributed throughout the core 1702 to shorten access time. The second bus 1722 may be implemented by at least one of an I2C bus, a SPI bus, a PCI bus, or a PCIe bus.

[0087] Each core 1702 and/or, more generally, the microprocessor 1700 may include additional and/or alternate structures to those shown and described above. For example, one or more clock circuits, one or more power supplies, one or more power gates, one or more cache home agents (CHAs), one or more converged/common mesh stops (CMSs), one or more shifters (e.g., barrel shifter(s)) and/or other circuitry may be present. The microprocessor 1700 is a semiconductor device fabricated to include many transistors interconnected to implement the structures described above in one or more integrated circuits (ICs) contained in one or more packages.

[0088] The microprocessor 1700 may include and/or cooperate with one or more accelerators (e.g., acceleration circuitry, hardware accelerators, etc.). In some examples, accelerators are implemented by logic circuitry to perform certain tasks more quickly and/or efficiently than can be done by a general-purpose processor. Examples of accelerators include ASICs and FPGAs such as those discussed herein. A GPU, DSP and/or other programmable device can also be an accelerator. Accelerators may be on-board the microprocessor 1700, in the same chip package as the microprocessor 1700 and/or in one or more separate packages from the microprocessor 1700.

[0089] FIG. 18 is a block diagram of another example implementation of the programmable circuitry of FIG. 16. In this example, the programmable circuitry 1612 is implemented by FPGA circuitry 1800. For example, the FPGA circuitry 1800 may be implemented by an FPGA. The FPGA circuitry 1800 can be used, for example, to perform operations that could otherwise be performed by the example microprocessor 1700 of FIG. 17 executing corresponding machine readable instructions. However, once configured, the FPGA circuitry 1800 instantiates the operations and/or functions corresponding to the machine readable instructions in hardware and, thus, can often execute the

operations/functions faster than they could be performed by a general-purpose microprocessor executing the corresponding software.

[0090] More specifically, in contrast to the microprocessor 1700 of FIG. 17 described above (which is a general purpose device that may be programmed to execute some or all of the machine readable instructions represented by the flowcharts of FIGS. 13, 14 and/or 15 but whose interconnections and logic circuitry are fixed once fabricated), the FPGA circuitry 1800 of the example of FIG. 18 includes interconnections and logic circuitry that may be configured, structured, programmed, and/or interconnected in different ways after fabrication to instantiate, for example, some or all of the operations/functions corresponding to the machine readable instructions represented by the flowcharts of FIGS. 13, 14 and/or 15. In particular, the FPGA 1800 may be thought of as an array of logic gates, interconnections, and switches. The switches can be programmed to change how the logic gates are interconnected by the interconnections, effectively forming one or more dedicated logic circuits (unless and until the FPGA circuitry 1800 is reprogrammed). The configured logic circuits enable the logic gates to cooperate in different ways to perform different operations on data received by input circuitry. Those operations may correspond to some or all of the instructions (e.g., the software and/or firmware) represented by the flowcharts of FIGS. 13, 14 and/or 15. As such, the FPGA circuitry 1800 may be configured and/or structured to effectively instantiate some or all of the operations/functions corresponding to the machine readable instructions of the flowcharts of FIGS. 13, 14 and/or 15 as dedicated logic circuits to perform the operations/functions corresponding to those software instructions in a dedicated manner analogous to an ASIC. Therefore, the FPGA circuitry 1800 may perform the operations/functions corresponding to the some or all of the machine readable instructions of FIGS. 13, 14 and/or 15 faster than the general-purpose microprocessor can execute the same.

[0091] In the example of FIG. 18, the FPGA circuitry 1800 is configured and/or structured in response to being programmed (and/or reprogrammed one or more times) based on a binary file. In some examples, the binary file may be compiled and/or generated based on instructions in a hardware description language (HDL) such as Lucid, Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL), or Verilog. For example, a user (e.g., a human user, a machine user, etc.) may write code or a program corresponding to one or more operations/functions in an HDL; the code/program may be translated into a low-level language as needed; and the code/program (e.g., the code/program in the low-level language) may be converted (e.g., by a compiler, a software application, etc.) into the binary file. In some examples, the FPGA circuitry 1800 of FIG. 18 may access

and/or load the binary file to cause the FPGA circuitry 1800 of FIG. 18 to be configured and/or structured to perform the one or more operations/functions. For example, the binary file may be implemented by a bit stream (e.g., one or more computer-readable bits, one or more machine-readable bits, etc.), data (e.g., computer-readable data, machine-readable data, etc.), and/or machine-readable instructions accessible to the FPGA circuitry 1800 of FIG. 18 to cause configuration and/or structuring of the FPGA circuitry 1800 of FIG. 18, or portion(s) thereof.

[0092] In some examples, the binary file is compiled, generated, transformed, and/or otherwise output from a uniform software platform utilized to program FPGAs. For example, the uniform software platform may translate first instructions (e.g., code or a program) that correspond to one or more operations/functions in a high-level language (e.g., C, C++, Python, etc.) into second instructions that correspond to the one or more operations/functions in an HDL. In some such examples, the binary file is compiled, generated, and/or otherwise output from the uniform software platform based on the second instructions. In some examples, the FPGA circuitry 1800 of FIG. 18 may access and/or load the binary file to cause the FPGA circuitry 1800 of FIG. 18 to be configured and/or structured to perform the one or more operations/functions. For example, the binary file may be implemented by a bit stream (e.g., one or more computer-readable bits, one or more machine-readable bits, etc.), data (e.g., computer-readable data, machine-readable data, etc.), and/or machine-readable instructions accessible to the FPGA circuitry 1800 of FIG. 18 to cause configuration and/or structuring of the FPGA circuitry 1800 of FIG. 18, or portion(s) thereof.

[0093] The FPGA circuitry 1800 of FIG. 18, includes example input/output (I/O) circuitry 1802 to obtain and/or output data to/from example configuration circuitry 1804 and/or external hardware 1806. For example, the configuration circuitry 1804 may be implemented by interface circuitry that may obtain a binary file, which may be implemented by a bit stream, data, and/or machine-readable instructions, to configure the FPGA circuitry 1800, or portion(s) thereof. In some such examples, the configuration circuitry 1804 may obtain the binary file from a user, a machine (e.g., hardware circuitry (e.g., programmable or dedicated circuitry) that may implement an Artificial Intelligence/Machine Learning (AI/ML) model to generate the binary file), etc., and/or any combination(s) thereof). In some examples, the external hardware 1806 may be implemented by external hardware circuitry. For example, the external hardware 1806 may be implemented by the microprocessor 1700 of FIG. 17.

[0094] The FPGA circuitry 1800 also includes an array of example logic gate circuitry 1808, a plurality of example configurable interconnections 1810, and example storage circuitry 1812. The logic gate circuitry 1808 and the configurable interconnections 1810 are configurable to instantiate one or more operations/functions that may correspond to at least some of the machine readable instructions of FIGS. 13, 14, and/or 15 and/or other desired operations. The logic gate circuitry 1808 shown in FIG. 18 is fabricated in blocks or groups. Each block includes semiconductor-based electrical structures that may be configured into logic circuits. In some examples, the electrical structures include logic gates (e.g., And gates, Or gates, Nor gates, etc.) that provide basic building blocks for logic circuits. Electrically controllable switches (e.g., transistors) are present within each of the logic gate circuitry 1808 to enable configuration of the electrical structures and/or the logic gates to form circuits to perform desired operations/functions. The logic gate circuitry 1808 may include other electrical structures such as look-up tables (LUTs), registers (e.g., flip-flops or latches), multiplexers, etc.

[0095] The configurable interconnections 1810 of the illustrated example are conductive pathways, traces, vias, or the like that may include electrically controllable switches (e.g., transistors) whose state can be changed by programming (e.g., using an HDL instruction language) to activate or deactivate one or more connections between one or more of the logic gate circuitry 1808 to program desired logic circuits.

[0096] The storage circuitry 1812 of the illustrated example is structured to store result(s) of the one or more of the operations performed by corresponding logic gates. The storage circuitry 1812 may be implemented by registers or the like. In the illustrated example, the storage circuitry 1812 is distributed amongst the logic gate circuitry 1808 to facilitate access and increase execution speed.

[0097] The example FPGA circuitry 1800 of FIG. 18 also includes example dedicated operations circuitry 1814. In this example, the dedicated operations circuitry 1814 includes special purpose circuitry 1816 that may be invoked to implement commonly used functions to avoid the need to program those functions in the field. Examples of such special purpose circuitry 1816 include memory (e.g., DRAM) controller circuitry, PCIe controller circuitry, clock circuitry, transceiver circuitry, memory, and multiplier-accumulator circuitry. Other types of special purpose circuitry may be present. In some examples, the FPGA circuitry 1800 may also include example general purpose programmable circuitry 1818 such as an example CPU 1820 and/or an example DSP 1822. Other general purpose programmable

circuitry 1818 may additionally or alternatively be present such as a GPU, an XPU, etc., that can be programmed to perform other operations.

[0098] Although FIGS. 17 and 18 illustrate two example implementations of the programmable circuitry 1612 of FIG. 16, many other approaches are contemplated. For example, FPGA circuitry may include an on-board CPU, such as one or more of the example CPU 1820 of FIG. 18. Therefore, the programmable circuitry 1612 of FIG. 16 may additionally be implemented by combining at least the example microprocessor 1700 of FIG. 17 and the example FPGA circuitry 1800 of FIG. 18. In some such hybrid examples, one or more cores 1802 of FIG. 18 may execute a first portion of the machine readable instructions represented by the flowchart(s) of FIGS. 13-15 to perform first operation(s)/function(s), the FPGA circuitry 1800 of FIG. 18 may be configured and/or structured to perform second operation(s)/function(s) corresponding to a second portion of the machine readable instructions represented by the flowcharts of FIGS. 13-15, and/or an ASIC may be configured and/or structured to perform third operation(s)/function(s) corresponding to a third portion of the machine readable instructions represented by the flowcharts of FIGS. 13-15.

[0099] It should be understood that some or all of the circuitry of FIG. 12 may, thus, be instantiated at the same or different times. For example, same and/or different portion(s) of the microprocessor 1700 of FIG. 17 may be programmed to execute portion(s) of machine-readable instructions at the same and/or different times. In some examples, same and/or different portion(s) of the FPGA circuitry 1800 of FIG. 18 may be configured and/or structured to perform operations/functions corresponding to portion(s) of machine-readable instructions at the same and/or different times.

[00100] In some examples, some or all of the circuitry of FIG. 12 may be instantiated, for example, in one or more threads executing concurrently and/or in series. For example, the microprocessor 1700 of FIG. 17 may execute machine readable instructions in one or more threads executing concurrently and/or in series. In some examples, the FPGA circuitry 1800 of FIG. 18 may be configured and/or structured to carry out operations/functions concurrently and/or in series. Moreover, in some examples, some or all of the circuitry of FIG. 12 may be implemented within one or more virtual machines and/or containers executing on the microprocessor 1700 of FIG. 17.

[00101] In some examples, the programmable circuitry 1612 of FIG. 16 may be in one or more packages. For example, the microprocessor 1700 of FIG. 17 and/or the FPGA circuitry 1800 of FIG. 18 may be in one or more packages. In some examples, an XPU may be implemented by the programmable circuitry 1612 of FIG. 16 which may be in one or more

packages. For example, the XPU may include a CPU (e.g., the microprocessor 1700 of FIG. 17, the CPU 1820 of FIG. 18, etc.) in one package, a DSP (e.g., the DSP 1822 of FIG. 18) in another package, a GPU in yet another package, and an FPGA (e.g., the FPGA circuitry 1800 of FIG. 18) in still yet another package.

[00102] A block diagram illustrating an example software distribution platform 1905 to distribute software such as the example machine readable instructions 1632 of FIG. 16 to other hardware devices (e.g., hardware devices owned and/or operated by third parties from the owner and/or operator of the software distribution platform) is illustrated in FIG. 19. The example software distribution platform 1905 may be implemented by any computer server, data facility, cloud service, etc., capable of storing and transmitting software to other computing devices. The third parties may be customers of the entity owning and/or operating the software distribution platform 1905. For example, the entity that owns and/or operates the software distribution platform 1905 may be a developer, a seller, and/or a licensor of software such as the example machine readable instructions 1632 of FIG. 16. The third parties may be consumers, users, retailers, OEMs, etc., who purchase and/or license the software for use and/or re-sale and/or sub-licensing. In the illustrated example, the software distribution platform 1905 includes one or more servers and one or more storage devices. The storage devices store the machine readable instructions 1632, which may correspond to the example machine readable instructions of FIGS. 13-15, as described above. The one or more servers of the example software distribution platform 1905 are in communication with an example network 1910, which may correspond to any one or more of the Internet and/or any of the example networks described above. In some examples, the one or more servers are responsive to requests to transmit the software to a requesting party as part of a commercial transaction. Payment for the delivery, sale, and/or license of the software may be handled by the one or more servers of the software distribution platform and/or by a third party payment entity. The servers enable purchasers and/or licensors to download the machine readable instructions 1632 from the software distribution platform 1905. For example, the software, which may correspond to the example machine readable instructions of FIGS. 13-15, may be downloaded to the example programmable circuitry platform 1600, which is to execute the machine readable instructions 1632 to implement the mixer characterization circuitry 102. In some examples, one or more servers of the software distribution platform 1905 periodically offer, transmit, and/or force updates to the software (e.g., the example machine readable instructions 1632 of FIG. 16) to ensure improvements, patches, updates, etc., are distributed and applied to the software at the end user devices.

Although referred to as software above, the distributed "software" could alternatively be firmware.

[00103] From the foregoing, it will be appreciated that example systems, methods, apparatus, and articles of manufacture have been disclosed that permit identification of a mixing time based on mixer unit input parameters. As such, mixer selection can be performed without the need for extensive qualitative and quantitative testing, given that modeling of the mixer-based parameters provides the anticipated mixing times without additional required inputs. In examples disclosed herein, users are informed about the time required for achieving homogeneity. In some examples, addition of acids, bases, and/or salts can be sequenced and automated based on predetermined mixing time values. In examples disclosed herein, an identification is performed to determine a desired location for adding feed during the mixing process. In some examples, media transfer to a bioreactor is automated based on predetermined mixing time values. In particular, a user interface is generated to allow a user to view and select mixer-based information such as mixer size, working volume, and/or impeller speed. Based on the selected mixer-based information, the mixing time is identified instantaneously, improving the efficiency of the bioprocessing steps associated with mixing of the bioprocess fluids.

[00104] Example methods, apparatus, systems, and articles of manufacture for efficient execution of convolutional neural networks for compressed video sequences are disclosed herein. Further examples and combinations thereof include the following:

[00105] Example 1 includes an apparatus for characterization of a mixer, the apparatus comprising interface circuitry, machine readable instructions, and programmable circuitry to at least one of instantiate or execute the machine readable instructions to identify a computational parameter to match a first set of input parameters to a first mixing time of the mixer, determine, based on the computational parameter, a second mixing time of the mixer based on a second set of input parameters, generate, based on the second mixing time, at least one of a time-associated recommendation or a location-associated recommendation, the time-associated recommendation or the location-associated recommendation including at least one of a time to reach homogeneity, a sequence of feed addition to the mixer, a location of the feed addition to the mixer, or a time for media transfer from the mixer to a bioreactor, and control at least one of the sequence of feed addition to the mixer or the location of the feed addition to the mixer.

[00106] Example 2 includes the apparatus of example 1, wherein the input parameters include at least one of a size of the mixer, a working volume of the mixer, or an impeller speed of the mixer.

[00107] Example 3 includes the apparatus of example 1, wherein the computational parameter is a shear, a power density, or a power number.

[00108] Example 4 includes the apparatus of example 3, wherein the programmable circuitry is to model the power number based on computational fluid dynamics and mathematical modeling.

[00109] Example 5 includes the apparatus of example 1, wherein the programmable circuitry is to automatically select a mixing speed to maintain low shear based on the time-associated recommendation or the location-associated recommendation.

[00110] Example 6 includes the apparatus of example 1, wherein the sequence of feed addition to the mixer includes addition of at least one of an acid, a base, or a salt, the sequence of feed addition to occur automatically based on the time-associated recommendation or the location-associated recommendation.

[00111] Example 7 includes the apparatus of example 1, wherein the programmable circuitry is to identify, using a virtual probe, the mixing time based on a type of mixing, including at least one of a liquid mixing, a solid-liquid mixing, a solid dispersion mixing, or a suspension-based mixing.

[00112] Example 8 includes a method for characterization of a mixer, the method comprising identifying a computational parameter to match a first set of input parameters to a first mixing time of the mixer, determining, based on the computational parameter, a second mixing time of the mixer based on a second set of input parameters, generating, based on the second mixing time, at least one of a time-associated recommendation or a location-associated recommendation, the time-associated recommendation or the location-associated recommendation including at least one of a time to reach homogeneity, a sequence of feed addition to the mixer, a location of the feed addition to the mixer, or a time for media transfer from the mixer to a bioreactor, and controlling at least one of the sequence of feed addition to the mixer or the location of the feed addition to the mixer.

[00113] Example 9 includes the method of example 8, wherein the input parameters include at least one of a size of the mixer, a working volume of the mixer, or an impeller speed of the mixer.

[00114] Example 10 includes the method of example 8, wherein the computational parameter is a shear, a power density, or a power number.

[00115] Example 11 includes the method of example 10, further including modeling the power number based on computational fluid dynamics and mathematical modeling.

[00116] Example 12 includes the method of example 8, further including automatically selecting a mixing speed to maintain low shear based on the time-associated recommendation or the location-associated recommendation.

[00117] Example 13 includes the method of example 8, wherein the sequence of feed addition to the mixer includes addition of at least one of an acid, a base, or a salt, the sequence of feed addition to occur automatically based on the time-associated recommendation or the location-associated recommendation.

[00118] Example 14 includes the method of example 8, further including identifying, using a virtual probe, the mixing time based on a type of mixing, including at least one of a liquid mixing, a solid-liquid mixing, a solid dispersion mixing, or a suspension-based mixing.

[00119] Example 15 includes a non-transitory machine readable storage medium comprising instructions to cause programmable circuitry to at least identify a computational parameter to match a first set of input parameters to a first mixing time of a mixer, determine, based on the computational parameter, a second mixing time of the mixer based on a second set of input parameters, generate, based on the second mixing time, at least one of a time-associated recommendation or a location-associated recommendation, the time-associated recommendation or the location-associated recommendation including at least one of a time to reach homogeneity, a sequence of feed addition to the mixer, a location of the feed addition to the mixer, or a time for media transfer from the mixer to a bioreactor, and control at least one of the sequence of feed addition to the mixer or the location of the feed addition to the mixer.

[00120] Example 16 includes the non-transitory machine readable storage medium of example 15, wherein the input parameters include at least one of a size of the mixer, a working volume of the mixer, or an impeller speed of the mixer.

[00121] Example 17 includes the non-transitory machine readable storage medium as defined in example 15, wherein the computational parameter is a shear, a power density, or a power number.

[00122] Example 18 includes the non-transitory machine readable storage medium as defined in example 17, wherein the instructions are to cause the programmable circuitry to model the power number based on computational fluid dynamics and mathematical modeling.

[00123] Example 19 includes the non-transitory machine readable storage medium as defined in example 15, wherein the instructions are to cause the programmable circuitry to automatically select a mixing speed to maintain low shear based on the time-associated recommendation or the location-associated recommendation.

[00124] Example 20 includes the non-transitory machine readable storage medium as defined in example 15, wherein the sequence of feed addition to the mixer includes addition of at least one of an acid, a base, or a salt, the sequence of feed addition to occur automatically based on the time-associated recommendation or the location-associated recommendation.

[00125] Although certain example methods, apparatus and articles of manufacture have been described herein, the scope of coverage of this patent is not limited thereto. On the contrary, this patent covers all methods, apparatus and articles of manufacture fairly falling within the scope of the claims of this patent.

What is claimed is:

1. An apparatus for characterization of a mixer, the apparatus comprising:

interface circuitry;

machine readable instructions; and

programmable circuitry to at least one of instantiate or execute the machine readable instructions to:

identify a computational parameter to match a first set of input parameters to a first mixing time of the mixer;

determine, based on the computational parameter, a second mixing time of the mixer based on a second set of input parameters;

generate, based on the second mixing time, at least one of a time-associated recommendation or a location-associated recommendation, the time-associated recommendation or the location-associated recommendation including at least one of a time to reach homogeneity, a sequence of feed addition to the mixer, a location of the feed addition to the mixer, or a time for media transfer from the mixer to a bioreactor; and

control at least one of the sequence of feed addition to the mixer or the location of the feed addition to the mixer.

2. The apparatus of claim 1, wherein the input parameters include at least one of a size of the mixer, a working volume of the mixer, or an impeller speed of the mixer.

3. The apparatus of claim 1, wherein the computational parameter is a shear, a power density, or a power number.

4. The apparatus of claim 3, wherein the programmable circuitry is to model the power number based on computational fluid dynamics and mathematical modeling.

5. The apparatus of claim 1, wherein the programmable circuitry is to automatically select a mixing speed to maintain low shear based on the time-associated recommendation or the location-associated recommendation.

6. The apparatus of claim 1, wherein the sequence of feed addition to the mixer includes addition of at least one of an acid, a base, or a salt, the sequence of feed addition to occur automatically based on the time-associated recommendation or the location-associated recommendation.

7. The apparatus of claim 1, wherein the programmable circuitry is to identify, using a virtual probe, the mixing time based on a type of mixing, including at least one of a liquid mixing, a solid-liquid mixing, a solid dispersion mixing, or a suspension-based mixing.

8. A method for characterization of a mixer, the method comprising:

identifying a computational parameter to match a first set of input parameters to a first mixing time of the mixer;

determining, based on the computational parameter, a second mixing time of the mixer based on a second set of input parameters;

generating, based on the second mixing time, at least one of a time-associated recommendation or a location-associated recommendation, the time-associated recommendation or the location-associated recommendation including at least one of a time to reach homogeneity, a sequence of feed addition to the mixer, a location of the feed addition to the mixer, or a time for media transfer from the mixer to a bioreactor; and

controlling at least one of the sequence of feed addition to the mixer or the location of the feed addition to the mixer.

9. The method of claim 8, wherein the input parameters include at least one of a size of the mixer, a working volume of the mixer, or an impeller speed of the mixer.

10. The method of claim 8, wherein the computational parameter is a shear, a power density, or a power number.

11. The method of claim 10, further including modeling the power number based on computational fluid dynamics and mathematical modeling.

12. The method of claim 8, further including automatically selecting a mixing speed to maintain low shear based on the time-associated recommendation or the location-associated recommendation.

13. The method of claim 8, wherein the sequence of feed addition to the mixer includes addition of at least one of an acid, a base, or a salt, the sequence of feed addition to occur automatically based on the time-associated recommendation or the location-associated recommendation.

14. The method of claim 8, further including identifying, using a virtual probe, the mixing time based on a type of mixing, including at least one of a liquid mixing, a solid-liquid mixing, a solid dispersion mixing, or a suspension-based mixing.

15. A non-transitory machine readable storage medium comprising instructions to cause programmable circuitry to at least:

identify a computational parameter to match a first set of input parameters to a first mixing time of a mixer;

determine, based on the computational parameter, a second mixing time of the mixer based on a second set of input parameters;

generate, based on the second mixing time, at least one of a time-associated recommendation or a location-associated recommendation, the time-associated recommendation or the location-associated recommendation including at least one of a time to reach homogeneity, a sequence of feed addition to the mixer, a location of the feed addition to the mixer, or a time for media transfer from the mixer to a bioreactor; and

control at least one of the sequence of feed addition to the mixer or the location of the feed addition to the mixer.

16. The non-transitory machine readable storage medium of claim 15, wherein the input parameters include at least one of a size of the mixer, a working volume of the mixer, or an impeller speed of the mixer.

17. The non-transitory machine readable storage medium as defined in claim 15, wherein the computational parameter is a shear, a power density, or a power number.

18. The non-transitory machine readable storage medium as defined in claim 17, wherein the instructions are to cause the programmable circuitry to model the power number based on computational fluid dynamics and mathematical modeling.

19. The non-transitory machine readable storage medium as defined in claim 15, wherein the instructions are to cause the programmable circuitry to automatically select a mixing speed to maintain low shear based on the time-associated recommendation or the location-associated recommendation.

20. The non-transitory machine readable storage medium as defined in claim 15, wherein the sequence of feed addition to the mixer includes addition of at least one of an acid, a base, or a salt, the sequence of feed addition to occur automatically based on the time-associated recommendation or the location-associated recommendation.