Published in final edited form as:

Pflugers Arch. 2013 April ; 465(4): 497–507. doi:10.1007/s00424-012-1186-8.

MATLAB implementation of a dynamic clamp with bandwidth >125 KHz capable of generating I_{Na} at 37°C

Chris Clausen, Virginijus Valiunas, Peter R. Brink, and Ira S. Cohen

Department of Physiology and Biophysics and Institute for Molecular Cardiology, Health Sciences Center, Stony Brook University, Stony Brook, NY 11794-8661, U.S.A

Abstract

We describe the construction of a dynamic clamp with bandwidth >125 KHz that utilizes a high performance, yet low cost, standard home/office PC interfaced with a high-speed (16 bit) data acquisition module. High bandwidth is achieved by exploiting recently available software advances (code-generation technology, optimized real-time kernel). Dynamic-clamp programs are constructed using Simulink, a visual programming language. Blocks for computation of membrane currents are written in the high-level MATLAB language; no programming in C is required. The instrument can be used in single- or dual-cell configurations, with the capability to modify programs while experiments are in progress. We describe an algorithm for computing the fast transient Na^+ current (I_{Na}) in real time, and test its accuracy and stability using rate constants appropriate for 37°C. We then construct a program capable of supplying three currents to a cell preparation: I_{Na} , the hyperpolarizing-activated inward pacemaker current (I_f), and an inward-rectifier K^+ current (I_{K1}). The program corrects for the IR drop due to electrode current flow, and also records all voltages and currents. We tested this program on dual patch-clamped HEK293 cells where the dynamic clamp controls a current-clamp amplifier and a voltage-clamp amplifier controls membrane potential, and current-clamped HEK293 cells where the dynamic clamp produces spontaneous pacing behavior exhibiting Na^+ spikes in otherwise passive cells.

Keywords

dynamic clamp; whole-cell clamp; ionic currents; computer models; excitable cells

Introduction

It is difficult to characterize the role that individual membrane ionic currents play in a cell or a tissue, yet such an understanding is invaluable in deciphering the cause of tissue malfunction, as well as in developing a strategy for its treatment. In cardiac, neuronal and other excitable-cell preparations, a myriad of computer models have been created that mimic much of a particular cell's electrical activity [e.g., 8,11,29]. In *in silico* experiments, these models have found wide use in basic physiological studies assessing how individual ionic currents affect excitability, and in preclinical studies evaluating the efficacy and safety of putative new drugs and/or therapeutic interventions (e.g., gene therapy) [15]. Results from these experiments are often highly useful in the rational design of subsequent *in vitro* and *in*

Phone: +1-631-444-3042, FAX: +1-631-444-3432, Chris.Clausen@stonybrook.edu.

Ethical standards

This work complies with the current laws of the United States of America.

Conflict of interest

The authors declare that they have no conflict of interest.

vivo studies, where proposed experimental maneuvers can be verified in the real biology [17].

A major utility of computer models relies on their ability to describe accurately the currents measured experimentally. Often the equations describing the individual currents are derived from studies done in different mammalian species, each under possibly different nonphysiologic conditions (e.g., reduced concentrations, current-blocking drugs with dubious specificity, etc.). Furthermore, often ionic currents present within the cell have yet to be quantified, and thus are absent in a model. These limitations, due simply to an imperfect knowledge of the biology, may reduce the value of model predictions.

The dynamic clamp is a technology that can be used when a cell's ionic currents are incompletely characterized, yet when one or more individual currents of interest can be simulated with certainty and the ion carrying the current is not relevant to the effects of the current. It is a hybrid approach—one involving a computer model interacting directly with a biological preparation—that is used to study the effects of adding (or removing) these currents. The technology involves patching the cell with a current clamp, recording membrane potential (V_m) via an analog-to-digital converter (ADC), using the computer to compute the currents of interest, and subsequently supplying these currents (I_m) to the cell via a digital-to-analog converter (DAC) connected to the command current of the clamp amplifier. Although the utility of this technique involves the usual caveats regarding patch-clamp methodology (e.g., whole cell versus perforated patch), it nevertheless avoids the problem of not knowing quantitatively the characteristics of the inherent membrane currents of the preparation. This technology has great potential in furthering basic physiology, but perhaps even a higher utility involving drug discovery and/or design: given a new putative pharmaceutical agent and/or intervention, how specifically will it alter the physiological response in a given cell, and what alteration would be optimal?

The first dynamic-clamp study investigated the role that gap junctions play in the synchronization of electrical activity in embryonic chick aggregates [23]. This work was followed by studies investigating the role gap junctions play in synchronizing sinus-node myocytes [28], and in determining the change in firing rate when individual pancreatic β cells are coupled to form islets [13]. The technique has also been used to study the functional role of individual ion currents. Sun and Wang [25] showed that the transient outward K^+ current in canine ventricle, which is smaller in endocardium than in epicardium, was *not* sufficient to account for the longer action-potential duration of the endocardial myocytes, while Berecki et al. [2] demonstrated that a mutant form of the HERG K^+ channel (R56Q) that generated the arrhythmogenic long-QT syndrome *was* capable of producing a substantially longer ventricular action potential, a hallmark of the disease. In neurophysiology where the dynamic clamp has been more widely applied, it has led to improved functional understanding of the role that individual K^+ , Ca^{++} and mixed-ion voltage-dependent currents play in regulating excitability and determining firing rates [9], as well as providing a tool to simulate synaptic inputs allowing improved understanding of how they lead to an integrated output [22]. For additional descriptions and applications of the dynamic clamp, we direct the reader to a number of excellent reviews [7,9,19,21,24,30].

To implement the dynamic clamp, basically three different approaches are in use. One involves a MS Windows-based desktop PC running a modified version of the QuB electrophysiology software (see www.qub.buffalo.edu), a freely available package that provides a graphical user interface (GUI), coupled with a scripting language used to define membrane currents. The instrument attains a bandwidth of ~50 KHz [20], but even when using a high-speed multicore PC, ~3% of the time steps are *not* serviced in the 20- μ s time interval [20] thus producing an artifact called “jitter.” Jitter (defined as the inability to

supply I_m within a specified time interval) is typically only of concern when V_m is rapidly changing, and its effects have been studied quantitatively [3].

A computer running a general-purpose operating system (MS Windows, Mac OS, Linux) *cannot* completely prevent system processes from interrupting a dynamic-clamp program at random times, sometimes for many milliseconds. Thus, such an application is inherently prone to jitter. An approach that avoids jitter altogether involves use of a real-time operating system (RTOS) where the system is configured to *guarantee* that a running application will *not* be interrupted. The Real-Time eXperiment Interface (RTXI) [16] utilizes a modification to the Linux operating system that converts it to a RTOS (see www.rtai.org). RTXI is freely available (see www.rtxi.org) and provides GUIs, a library of computational modules, and hardware drivers for a number of different devices, that has seen extensive use in dynamic-clamp applications [4,6]. RTXI is fast (>50 KHz) and extremely flexible, but nevertheless can be cumbersome since significant working knowledge of Linux-based systems is required to install and maintain the package, and programming often requires proficiency in low-level languages (e.g., C or C++) when required capabilities are not already included in the GUI.

A third approach that is also jitter free involves the use of industrial hardware with embedded processors running proprietary RTOSs. Cambridge Electronic Design (see www.ced.co.uk) provides a commercial dynamic clamp (Power1401 mk 2) that utilizes an embedded processor. Their instrument is controlled by software (Signal) that runs on a standard PC and provides a GUI for data acquisition and programming the hardware. Another such embedded-processor instrument is implemented using LabVIEW-RT, available from National Instruments Corp. (see www.ni.com), where programming is done graphically (G code) by connecting “wires” between blocks on a diagram [14]. In addition, a GUI (G-clamp) is also available as a free download (see www.neurobio.pitt.edu). Compared to the RTXI dynamic clamps, the embedded-processor devices are significantly easier to program, but they are costly, programming flexibility is somewhat limited, and performance is limited by the slow performance of the embedded processor. Note that in the case of the LabVIEW-based system, this latter problem has recently been addressed by National Instruments (see Discussion).

The three approaches for implementing a dynamic clamp each work well and have seen extensive use. We chose a different approach that we hoped would be simpler to program, would exhibit increased performance, and could be implemented using inexpensive “off-the-shelf” hardware. It is based on MATLAB[®], a mathematical programming environment common to many laboratories (available from The Mathworks, see www.mathworks.com). A “toolbox” is available for industrial-process control that includes a proprietary RTOS kernel specifically designed for running hardware-in-the-loop applications. Moreover, the code runs on inexpensive, yet high speed, desktop PCs (avoiding the need for an embedded processor). Programming is done entirely in high-level languages: Simulink[®], a visual language (similar to G code) is used to construct a dynamic-clamp model via a graphical interface; computational elements are implemented as small functions written in the MATLAB language. We hoped to take advantage of recent software advances (introduced in 2007) where real-time performance is enhanced by automatically converting high-level Simulink and MATLAB code to optimized machine code that is downloaded to the PC running the dynamic clamp. Thus, our goal was to design a high-speed instrument that could be completely constructed, installed, programmed and maintained by electrophysiologists, who have reasonable but *not* expert experience in the use of laboratory computers.

Materials and methods

Electrophysiological measurements

Fig. 1a shows a block diagram of a test configuration to assess performance where a cell is doubly patched in voltage-clamp and current-clamp modes. Whole-cell patch recordings are performed using an Axopatch 200 amplifier (Axon Instruments, Foster City, CA), interfaced to a Windows-based PC (host) used for data acquisition and the generation of command-voltage pulse protocols (16-bit ADC, Digidata 1322A and pClamp9 software, Molecular Devices). The cell is patched with a second pipette attached to another Axopatch 200 amplifier operating in current-clamp mode, and interfaced to a second PC (target) running the dynamic-clamp program. Currents computed by the dynamic clamp are supplied as command currents to this amplifier.

Prior to applying these currents, we compensate for the access resistances of the voltage-clamp electrode (R_{eV}) and current-clamp electrode (R_{eC}). First, R_{eC} and cell capacitance (C_m) are measured and entered as parameters. The current clamp is subsequently set to supply zero current, and thus acts as an electrometer providing a direct measure of V_m . Next, the voltage clamp is adjusted to set the holding potential at -80 mV, and then to run a stepwise series of 60-ms depolarizing voltage steps (no attempts were made to correct for liquid junction potentials across the electrode). R_{eV} is then dialed into the voltage clamp's compensation circuitry and the percent electrode resistance compensation is adjusted such that measured V_m equals the step command voltage (after capacitive currents have decayed). Finally, voltage-clamp capacitance compensation is adjusted such that measured V_m follows the voltage command steps as rapidly, and with as little overshoot, as possible. Under these conditions when the current clamp supplies I_m , the potential it measures will *not* equal V_m because of the IR drop across the current-clamp electrode. The dynamic clamp corrects for this and computes V_m in software by adding $I_m R_{eC}$ to the voltage measured by the amplifier.

In another set of experiments, the dynamic-clamp is used in the more typical current-clamp configuration (Fig. 1b) where the cell is patched by a single amplifier in current-clamp mode. In this configuration, the compensation procedure for R_{eC} is different; V_m equals the electrode potential minus the IR drop across the electrode (e.g., $I_m R_{eC}$), and this correction is done in software by the dynamic-clamp program.

Implementation of the dynamic clamp

Host and target computers—The host computer is a Windows-based machine: the same computer that is used for data acquisition and the generation of voltage-clamp pulse protocols (Fig. 1a). This machine is used as the platform for software that controls the dynamic clamp. The dynamic-clamp program runs on the target computer, a machine that requires an ADC for recording V_m and a DAC for controlling I_m . The host machine downloads the dynamic-clamp program via an Ethernet connection to the target machine.

The target machine is a Pentium-based multicore desktop PC (see Online Resource 1). It was built using a Super Micro C2SBC-Q motherboard (Super Micro Computer, Inc., San Jose, CA) with a 3.16 GHz Intel Core 2 Duo processor (6 MiB L2 cache, 1333 MHz front-side bus) and 4 GiB of memory (800 MHz, dual-channel interleaved). Graphics and Ethernet support are included on the motherboard; the network connection is used to boot the RTOS kernel. The target machine does *not* require disks; a monitor and keyboard (for programming the BIOS) is desirable.

High bandwidth requires a data acquisition module with short latency. We chose a model PCIe-6251 (National Instruments, Austin, TX) that interfaces with the target computer via its PCI Express bus. The module has eight 16-bit differential ADC inputs with single-

channel latency of 0.8 μs and two 16-bit DAC outputs with latency of 0.35 μs . Modules with shorter latencies would *not* significantly improve the bandwidth (see Discussion). The data-acquisition module requires a special cable and terminal box (models SHC68-68-EPM and BNC-2110, National Instruments).

Mathworks software—The instrument is implemented entirely in MATLAB and Simulink (version R2011b). Additional required Mathworks products include MATLAB Coder™ and Simulink Coder™ that convert MATLAB and Simulink code to compilable C code, and xPC Target™ that provides ADC and DAC drivers and the RTOS kernel that manages the target computer. The purchase of Mathworks licenses provides the user with complete software support. A third-party compatible C compiler is also required; we used Microsoft Visual C++ Professional 2008 SP1.

Membrane currents—We chose to implement three membrane currents. For gating variables, we adhere to the Hodgkin-Huxley [10] formalism, namely for an arbitrary gating variable a , $da/dt = \alpha(1-a) - \beta a$, where α and β (ms^{-1}) are the forward and reverse voltage-dependent rate constants. The simplicity of this approach has seen widespread use in *in silico* studies.

An inward rectifying K^+ current (I_{K1}) that normally sets the cell's resting potential was characterized in voltage-clamp studies of canine myocytes (data not shown), and was found to be fit well by the equation

$$I_{\text{K1}} = \frac{G_{\text{K1}}(V_{\text{m}} - E_{\text{K}})}{1 + \exp[0.0810(V_{\text{m}} - E_{\text{K}} + 1.4425)]}, \quad (1)$$

where G_{K1} (nS/pF) is channel conductance and E_{K} (mV) is the K^+ reversal potential. This channel activates instantaneously.

A pacemaker current (I_{f} , also referred to as I_{h} in neurons [12]) is an inward current that activates at hyperpolarized potentials and is described by

$$I_{\text{f}} = \bar{G}_{\text{f}} y (V_{\text{m}} - E_{\text{f}}), \quad (2)$$

where $\bar{G}_{\text{f}}$ (nS/pF) is the maximal channel conductance, E_{f} (mV) is the channel's reversal potential, and y is the activation gating variable. Equations describing the forward and reverse rate constants for y , derived from voltage-clamp studies in canine Purkinje myocytes [5], can be found in Online Resource 1.

We included a fast transient Na^+ current (I_{Na}) given by

$$I_{\text{Na}} = \bar{G}_{\text{Na}} m^3 h j (V_{\text{m}} - E_{\text{Na}}), \quad (3)$$

where $\bar{G}_{\text{Na}}$ (nS/pF) is the maximum channel conductance, E_{Na} (mV) is the Na^+ equilibrium potential, m is the activation gating variable, h is the fast inactivation gating variable and j is the slow inactivation (or recovery from inactivation) gating variable. Eq. 3 is from a canine ventricular model [18]. Equations describing the rate constants for m , h and j , can be found in Online Resource 1.

Programming and operation—Details involved in constructing the dynamic-clamp program are found in Online Resource 1, which details additional equations, numerical methods, program listings, and specific program settings. Our test implementation is shown as the Simulink model (a visual computer program) in Fig. 2.

Signals and constants in Fig. 2 are double-precision floating-point quantities with physical units: amplifier gains of the ADC and DAC result in a dynamic range of ± 200 mV and 5 V (± 10 nA), respectively. During an experiment, the model (Fig. 2) is typically displayed on the host computer. After specifying model parameters (e.g., run duration, typically 10–30 s), entering “Ctrl-B” builds the model on the host machine and downloads it to the target machine (typically taking <10 s). The dynamic clamp is started by clicking on a “start” button above the displayed diagram. Upon completion, simple `MATLAB` commands upload the recorded data to the host computer. Changes to model parameters can be made at any time, and the dynamic clamp can be restarted without having to rebuild the program. Alternatively, the dynamic-clamp can be controlled by commands from the `MATLAB` workspace providing the ability to run complex experimental protocols automatically via `MATLAB` scripts.

Cell preparation

Experiments were performed on HEK293 cells grown in Dulbecco’s modified Eagle medium supplemented with 10% FBS, 2 mmol/L L-glutamine, 100 U/mL penicillin, 100 μ g/mL streptomycin and 0.5 mg/mL G418 (geneticin). Production, characterization and identification of these cells and culture conditions have been described previously [27]. Experiments were carried out on cells cultured for 1–3 days. Glass cover slips with adherent cells were transferred to an experimental chamber mounted on the stage of an inverted microscope. The chamber was perfused at room temperature ($\sim 20^\circ\text{C}$) with bath solution containing (in mmol/L): 140 NaCl, 1 MgCl_2 , 5 KCl, 2 CaCl_2 , 5 HEPES (pH 7.4) and 10 glucose. Patch pipettes were filled with solution containing (in mmol/L): 120 K-aspartate, 10 NaCl, 3 MgATP , 5 HEPES (pH 7.2) and 10 EGTA (pCa ~ 8). Filled patch pipettes typically had a resistance of 1–2 M Ω .

Numerical simulations

In some cases, dynamic-clamp performance was compared to “exact” solutions of the membrane current equations. These simulations were done in `MATLAB`; the term “exact” means that differential equations were solved to an accuracy of six significant digits using routines supplied in the `MATLAB` library.

Results

Dynamic clamp bandwidth, latency, and model complexity

Tests were done to ascertain the maximum bandwidth of the dynamic clamp. Ten 10-s runs were made at a bandwidth of 125 KHz (8- μ s time step) yielding 1.25×10^7 total samples where task execution time (TET) was logged by the RTOS. TET is the measured time needed to execute each time step of the model, and has a resolution of one CPU clock tic (0.316 ns). Variability among timing measurements was found *not* to be normally distributed (long TET values were rare events); median values are reported.

The first test was done using a “hardware-only” model consisting of the output of the ADC linked directly to the input of the DAC (a 5 Hz sine wave was supplied to the ADC that spanned its dynamic range). TET ranged from 4.55 μ s to 6.56 μ s with a median of 4.59 μ s. Maximum TET equates to a maximum *hardware-limited* bandwidth (no floating-point computations) of 152 KHz. For the model of Fig. 2 where I_{K1} , I_f , I_{Na} , electrode-resistance compensation, and data logging were computed in real time, TET ranged from 4.93 μ s to 6.97 μ s, with a median of 4.98 μ s. Maximum TET here equates to a maximum *working* bandwidth of 143 KHz. Using maximum TET values to compute bandwidths ensures that at these bandwidths, the dynamic clamp is performing reliably (i.e., no jitter).

Latency is defined as the time between sampling of V_m by the ADC and the time when I_m appears on the output of the DAC. TET provides a lower-bound estimate of latency (it does not include the 0.35- μ s settling time of the DAC), since in the “hardware only” mode, the time (4.59 μ s) was not significantly reduced with removal of the DAC (4.57 μ s). Fig. 2 TET was found to be remarkably constant with >92% of the measured times occurring within the narrow window of 4.98 ± 0.05 μ s.

We next measured the time needed to perform the floating-point computations, the elapsed time between measuring V_m by the ADC and supplying the current to the DAC (a 5 Hz ± 100 mV sine wave was supplied to the ADC). Computational time ranged from 0.343 μ s to 0.935 μ s with a median of 0.346 μ s. Subtracting the maximum TET (6.97 μ s) from the sampling interval (8 μ s) yields 1.03 μ s that can be used for additional computations. Thus, a model nearly four times the complexity of Fig. 2 should still run reliably at 125 KHz.

Numerical accuracy of I_{Na} algorithm

Details of the numerical methods used in computing the membrane currents can be found in Online Resource 1. The fastest kinetics are observed in the computation of the activation gating variable (m) of I_{Na} , which (at 37°C) typically exhibits time constants <50 μ s. In computer simulations shown in Fig. 3, we assess the accuracy of the algorithm used for computing I_{Na} ($\bar{G}_{Na} = 1$ nS/pF, $E_{Na} = +60$ mV) in the dynamic-clamp software running at 125 KHz. The currents result from instantaneously stepping from a holding potential of -80 mV to three depolarized potentials (0 mV, +20 mV and +40 mV), with increasingly rapid m -activation kinetics ($\tau_m = 65, 46$ and 36 μ s, respectively). Blue curves are results computed by the dynamic-clamp algorithm, and red curves are the exact solutions. The dynamic-clamp peak currents lag the exact solutions by no more than 7.9 μ s. Peak-current amplitudes are slightly less than the exact peaks with a worst-case error of 1.1%. This shows that the real-time computation of I_{Na} at a sampling interval of 8 μ s is being done accurately. Note that these tests are under worst-case conditions, namely instantaneous stepping to depolarized potentials. In current-clamp studies, changes in V_m occur more gradually owing to charging C_m .

Accuracy of entire dynamic-clamp numerical algorithm

We simulated the case of a cell exhibiting unusually rapid spikes with upstroke velocities ($\dot{V}_{max}$) exceeding 1000 V/s, significantly higher than that expected experimentally. When $G_{K1} = 4$ nS/pF ($E_K = -90$ mV), $\bar{G}_f = 2$ nS/pF ($E_f = -30$ mV), and $\bar{G}_{Na} = 300$ nS/pF ($E_{Na} = +60$ mV), exact solution of the equations (kinetics appropriate for 37°C) produced spontaneous pacemaker activity with rapid Na^+ spikes where steady-state $\dot{V}_{max}$ was 1130 V/s. This simulation was compared with results computed with the dynamic-clamp algorithm (Fig. 2) clocked at 8 μ s, and where $dV_m/dt = -I_m/C_m$ was integrated using Huen's method (see Online Resource 1). The voltage trace computed by the dynamic-clamp algorithm was visually identical to the exact solution. Thus, the currents computed by the dynamic-clamp algorithm running at 125 KHz are accurate. Note that Fig. 6 that shows similar experimental results recorded from a HEK293 cell.

Operation of the dynamic clamp in voltage-clamped cells

To assess dynamic-clamp performance in actual cells, we performed experiments in doubly patched HEK293 cells (see Fig. 1a) where the voltage clamp was used to control V_m and the dynamic clamp responded by producing simulated I_{Na} . Experiments were performed at room temperature ($\sim 20^\circ$ C), where the kinetics of the gating-variable rate constants for I_{Na} were reduced by a factor of 8 (see Online Resource 1). The parameters used in computing I_{Na} are $\bar{G}_{Na} = 5$ nS/pF and $E_{Na} = +99$ mV.

Fig. 4 shows results from a representative cell ($C_m = 24$ pF). The cell was initially clamped to -80 mV, at which point the dynamic clamp was started. It ran continuously during the subsequent voltage-clamp pulse protocol: a series of 360-ms cycles where V_m was held at -80 mV for 8 ms, followed by a depolarized step for 80 ms, and then returned to -80 mV for 272 ms. Step potentials ranged from -40 mV to $+60$ mV. Observe the electrode potential (Fig. 4, middle), where the potentials are “contaminated” by I_{Na} being supplied by the dynamic clamp. The agreement between command potential (Fig. 4, upper) and V_m (Fig. 4, lower) shows that correction for the IR drop across the current-clamp electrode ($R_{eC} = 7.0$ M Ω) is being done accurately, and also provides experimental verification that voltage-clamp electrode resistance compensation (done by amplifier circuitry) is being done effectively.

Fig. 5 (blue traces) shows I_{Na} recorded by the dynamic clamp at the respective step potentials shown in Fig. 4. The records show the expected characteristics of I_{Na} . Namely, time constant and peak current both decrease as one steps to successively more depolarizing potentials, owing to more rapid activation (m) kinetics and potentials becoming closer to E_{Na} , respectively. Furthermore, the measured dynamic-clamp currents are those that are expected, as shown by agreement with the red traces in Fig. 5 where I_{Na} is computed *exactly* using the measured V_m values. The small differences between the measured and exact traces of Fig. 5 are not unexpected (see Discussion).

Operation of the dynamic clamp in current-clamp mode

In these experiments we used the setup seen in Fig. 1b, the more usual whole-cell patched configuration in current-clamp mode. HEK293 cells typically exhibit resting potentials between -30 mV and -60 mV, and virtually no time-dependent currents at membrane potentials between -100 mV and $+80$ mV. The blue trace in Fig. 6 shows representative results from a HEK293 cell where I_{Na} was gradually increased over the course of 10 s. Although the cell preparation was at $\sim 20^\circ\text{C}$, I_{Na} was computed using rate constants appropriate for 37°C .

Prior to starting the dynamic clamp, the resting potential was -57.6 mV, cell conductance was 0.83 nS and cell capacitance was 22 pF. $\bar{G}_{Na}$ was programmed to increase linearly from 0 nS/pF to 100 nS/pF ($E_{Na} = +60$ mV) over the course of 10 s by using a ramp function (not shown in Fig. 2). At time $t = 0$, the dynamic clamp was started with $G_{K1} = 4$ nS/pF ($E_K = -90$ mV) and $\bar{G}_f = 2$ nS/pF ($E_f = -30$ mV). One initially observes an immediate hyperpolarization caused by I_{K1} , followed by a slower depolarization due to the activation of I_f that attains threshold for activation of I_{Na} . One subsequently observes spontaneous pacing (cycle length ~ 0.45 s), with increasing rapid depolarizations reaching peak potentials approaching E_{Na} at $t = 10$ s (when $\bar{G}_{Na} = 100$ nS/pF). We simulated the HEK293-cell response using a mathematical model where all equations were solved exactly. The model contained the three currents plus a leak current $I_{leak} = G_{leak}(V_m - E_{leak})$, where we assumed that G_{leak} was the cell's resting conductance (0.038 nS/pF) and E_{leak} equaled its resting potential (-57.6 mV) *prior* to starting the dynamic clamp. The results of this simulation are shown in Fig. 6 (red) plotted with the measured cell response (blue). The agreement between the HEK293-cell response and the model simulation is remarkable, considering that all native currents in the HEK293 cell are lumped into I_{leak} , that E_{leak} is an assumed parameter, and that the simulation is unaffected by bandwidth limitations of the current-clamp electronics.

We computed $\dot{V}_{max}$ of the measured spikes in Fig. 6 (blue) to assess the dynamic clamp's ability to follow these rapid depolarizations, by computing the difference in successive V_m measurements divided by the sampling interval (8 μ s). We found that $\dot{V}_{max}$ increased

linearly as a function of $\bar{G}_{Na}$ ($\dot{V}_{max} = 9.57 \cdot \bar{G}_{Na} - 64.9$, $r^2 = 0.996$) reaching a maximum of >825 V/s.

Discussion

Ease of programming

Programming the dynamic clamp involved constructing the Simulink model seen in Fig. 2 (i.e., dragging and dropping blocks onto the diagram and making interconnections by moving a cursor from outputs of blocks to the inputs of other blocks). There are a number of additional blocks (not shown in Fig. 2) for performing arithmetic, generating pulses and time-varying signals (e.g., the ramp function used in Fig. 6 to vary $\bar{G}_{Na}$), storing data, etc. The software provides a library of blocks representing different hardware for measuring (ADC) and outputting (DAC) analog voltages; it includes drivers for 16 different hardware manufacturers. The only other programming involves writing computational functions in the `MATLAB` language to implement the different currents of interest (a total of 84 lines of code, see Online Resource 1). `MATLAB` code is a high-level language that frees one from having to declare data types, allocate storage, etc. It includes a library of transcendental and special functions, and is matrix-based and includes LAPACK (see www.netlib.org/lapack) for performing linear algebra and eigensystem analysis. The functions are first tested and debugged in `MATLAB`, and are then included unchanged as blocks with the Simulink diagram (Fig. 2). For efficiency, the Simulink diagram and `MATLAB` code utilize so-called code generation technology whereby they are converted *automatically* to C code, which is compiled by an optimizing compiler and linked prior to downloading to the target computer. This ensures that the code executes as rapidly as possible using the native instruction set of the target processor.

In Fig. 2, the dynamic clamp is programmed to supply I_{K1} , I_f and I_{Na} . The actual currents are irrelevant, as any current that can be described mathematically could have been used. In addition, the dynamic clamp can be programmed for experiments in multiple cells. In Valiunas et al. [26], we programmed the instrument in a dual-cell configuration where membrane potentials were recorded from two current-clamped cells, with separate command currents supplied to each. The cells were coupled by “virtual” gap junctions, where the difference in the cells’ membrane potentials resulted in electrotonic current flow being supplied by the two clamps. This was done by modifying Fig. 2, a process requiring less than an hour of effort.

RTOS characteristics

Much of the performance of the dynamic clamp can be attributed to the xPC target kernel. The kernel forms a small system that runs entirely in Intel supervisor (kernel) mode. It is optimized to eliminate all non-essential processes and has direct access to the hardware (e.g., ADC, DAC, internal timer), thereby avoiding the overhead of operating-system calls. The kernel is small (<4 MiB), thus much (if not all) of it resides in the CPU cache memory (6 MiB), which eliminates the need to access main memory that otherwise would degrade performance. The dynamic-clamp code also runs in supervisor mode, and (being small) typically it too resides in the cache.

All code is compiled using a flat (32-bit) memory model, where the program has full access to the PC’s memory (up to 2 GiB), providing the ability to record large amounts of data (>255 -million floating-point numbers for a 2-GiB machine). These data are uploaded to the host computer following an experimental run, a process involving execution of simple `MATLAB` commands and/or scripts issued from the host computer.

Bandwidth considerations

We found that the bulk of the time involves communication with the ADC and DAC, devices that reside on the PCI bus. Input/output instructions that communicate with the PCI bus are relatively slow (0.5–1 μ s); for each time interval, these instructions are executed to start the ADC, poll it to ascertain when data are available, and to start the DAC. Although the ADC and DAC have short hardware latencies, even shorter latencies would *not* be expected to result in a significant increase in performance; performance is limited instead by the communication times between the CPU and the PCI bus. Since the desired bandwidth (125 KHz) approaches the hardware-limited bandwidth (152 KHz), this requires special considerations when running different models.

Under conditions where kinetics of channel gating is sufficiently slow (e.g., time constants greater than ca. 100 μ s), the dynamic clamp is run in interrupt mode. A timer is programmed to interrupt the processor after each time step, at which point the ADC is sampled and current is computed and outputted to the DAC. The kernel uses any remaining time to support other concurrent tasks, e.g., display of graphical results on the target's monitor, communication with the host computer (for monitoring of signals by the host, changing program parameters "on the fly," or aborting the running program). Interrupt mode works well for bandwidths $\leq$ 50 KHz, as tested by a model involving four currents that depend on two recorded potentials and two outputted currents in a dual whole-cell current-clamp configuration [26]. At higher bandwidths (typically >70 KHz), interrupt mode sometimes fails producing overruns where the kernel *detects* that it has missed a time point. By default an overrun halts the running program, but alternatively the program can be allowed to run following a detected overrun, thus permitting real-time corrective action.

Interrupt mode entails an overhead of ~ 8 μ s, the time required for communication with an interrupt controller, the saving of processor registers, CPU hardware latency, reloading of cache memory, etc. This overhead is essentially independent of processor speed; a faster clocked CPU will *not* significantly reduce it. Polling mode is an alternative more primitive way to synchronize the program to a given clock rate, but it avoids the 8- μ s interrupt overhead altogether; the processor essentially runs in a tight loop checking a real-time clock for the start of the next time interval. The clock resides on the CPU itself and can be queried in one instruction cycle (it increments at the CPU clock frequency). Once the start of the next time interval is detected, the ADC is started, the program waits for its completion, the currents are computed and results are outputted to the DAC. The kernel then returns to polling the clock awaiting the start of the next interval. Thus in polling mode, timing does *not* involve the use of an external timer.

Running in polling mode does, however, come with some limitations. Since interrupts are disabled, other concurrent target tasks halt while the dynamic clamp is in operation. Screen updates of the target's monitor and network communication between the target and host computers are temporarily suspended, and an experimental run cannot be stopped prematurely. All data, however, are logged by the target and can be subsequently uploaded to the host when network communications is restored. Changing model parameters "on the fly" via commands from the host computer is prevented during an experimental run, but one can vary a parameter as a function of time by adding blocks to the Simulink model (e.g., in Fig. 6 $\bar{G}_{Na}$ was increased linearly with time over the 10-s duration of the model by using a ramp function).

Floating-point arithmetic

For the model shown in Fig. 2, computing the currents requires ~ 138 floating-point computations (including 14 evaluations of e^x) to compute three currents involving the

integration of four differential equations. At the 8- μ s sampling interval, we estimate that ~548 floating-point computations could be performed in the allotted time, equating to a model with ~10 currents, each described by their respective gating variables. Of course, this bandwidth is *only* required when I_{Na} (at 37°C) is included in the model. At slower bandwidths (< 50 KHz) that are adequate for most other experimental conditions *not* involving I_{Na} , the computational power of the processor is sufficient to construct a dynamic clamp with a model consisting of far more ionic currents, each requiring the integration of their associated gating variables. Moreover, the use of MATLAB'S matrix arithmetic capabilities should greatly facilitate efficient computation of complex channel kinetics, e.g., complex Markov models [20], where solutions can involve eigensystem analysis.

Tests in HEK293 cells

The first test of the complete instrumentation involved using HEK293 cells doubly patched where a voltage clamp controlled membrane potential and the dynamic clamp supplied membrane current. In these room-temperature experiments, the kinetics of I_{Na} were slowed to that expected at ~20°C, but this was done only because it was unreasonable to expect the voltage clamp to be capable of controlling membrane potential with 37°C kinetics. The dynamic clamp produced I_{Na} (Fig. 5) consistent with what was expected when the cell's membrane potential was stepped to depolarized potentials. Small differences were noted between measured currents (blue) and the currents computed exactly (red)—the measured peak currents were slightly less (<5%) than that computed currents—but this is not surprising. For example, tabulated values for V_m were only recorded starting 5 ms preceding the pulse and ending 5 ms following the pulse; when integrating the exact solution during the inter-pulse interval, V_m was estimated by linear interpolation. Thus, we conclude that this complex experimental setup performed well, showing that the dynamic clamp could be seamlessly added to an existing setup, where it produced expected and accurate results.

The more typical configuration of a dynamic clamp was used when we performed experiments in HEK293 cells patched by a single current-clamp amplifier. Although the experiments were done at ~20°C, the kinetics for I_{Na} were appropriate for 37°C. The spontaneous pacing observed (Fig. 6) was in surprisingly good agreement with that expected based on a mathematical model. In addition, we found that as $\bar{G}_{Na}$ increased, $\dot{V}_{max}$ increased linearly to values >825 V/s. This range of responsiveness is important for investigators performing experiments in mammalian myocytes, where $\dot{V}_{max}$ rarely exceeds 500 V/s [4]. Namely, the dynamic clamp is capable of supplying inward Na^+ current in mammalian preparations under *all* experimental conditions.

Conclusions

Our description of a dynamic clamp constructed using MATLAB software appears to exhibit the fastest reported performance to date (up to 143 KHz) with sufficient additional capacity to construct models several times more complex than Fig. 2. Our intention is not to promote a superior implementation, but rather to present an alternative based on recent advancements, notably ease of construction and programming, code-generation technology producing optimized machine code, and an optimized RTOS that exploits the architecture (e.g., polling mode, use of cache memory) of modern high-performance off-the-shelf components. Finally, it has not escaped our attention that as of 2009, National Instruments has enhanced LabVIEW-RT. Their software now runs on low-cost desktop PC's, and they have implemented code-generation technology that converts their G-code and computational blocks to optimized machine code. We suspect that users of the popular G-clamp GUI [14] might observe performance benchmarks similar to those reported here.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

Mr. Robert Butz provided valuable hardware advice and built the dynamic clamp hardware. Dr. Michael R. Rosen provided critical comments on the manuscript. The xPC Target development team at Mathworks graciously responded to detailed technical questions regarding their xPC target software. Supported by National Institutes of General Medical Sciences grant GM-088181, National Heart, Lung and Blood Institute grants HL-111401, HL-094410 and HL-111649, and New York Stem Cell Science grant CO24344.

References

1. Berecki G, Wilders R, de Jonge B, van Ginneken ACG, Verkerk AO. Re-Evaluation of the Action Potential Upstroke Velocity as a Measure of the Na⁺ Current in Cardiac Myocytes at Physiological Conditions. *PLoS ONE*. 2010; 5(12):e15772. [PubMed: 21217835]
2. Berecki G, Zegers JG, Verkerk AO, Bhuiyan ZA, de Jonge B, Veldkamp MP, Wilders R, van Ginneken ACG. HERG channel (dys)function revealed by dynamic action potential clamp technique. *Biophys. J.* 2005; 88:566–578. [PubMed: 15475579]
3. Bettencourt JC, Lillis KP, Stupin LR, White JA. Effects of imperfect dynamic clamp: Computational and experimental results. *J. Neurosci. Meth.* 2008; 169:282–289.
4. Butera RJ, Wilson CG, DelNegro CA, Smith JC. A methodology for achieving high-speed rates for artificial conductance injection in electrically excitable biological cells. *IEEE Trans. Biomed. Eng.* 2001; 48:1560–1470.
5. Clausen, C.; Rosen, MR.; Cohen, IS. Synthesis of the cardiac purkinje-fiber action potential using a computer model. In: Zaza, A., editor. *An Introduction to Cardiac Electrophysiology*. Harwood Academic Publishers: Amsterdam; 2000. p. 199-216.
6. Dorval AD, Christini DJ, White JA. Real-time LINUX dynamic clamp: a fast and flexible way to construct virtual ion channels in living cells. *Ann. Biomed. Eng.* 2001; 29:897–907. [PubMed: 11764320]
7. Economo MH, Fernandez FR, White JA. Dynamic Clamp: Alteration of Response Properties and Creation of Virtual Realities in Neurophysiology. *J. Neurosci.* 2010; 30:2407–2413. [PubMed: 20164323]
8. Fridlyand LE, Tamarina N, Philioson LH. Modeling of Ca²⁺ flux in pancreatic β -cells: role of the plasma membrane and intracellular stores. *Am. J. Physiol. Endocrinol. Metab.* 2003; 285:E138–E154. [PubMed: 12644446]
9. Goaillard J-M, Marder E. Dynamic Clamp Analysis of Cardiac, Endocrine, and Neural Function. *Physiology*. 2006; 21:197–207. [PubMed: 16714478]
10. Hodgkin AL, Huxley AF. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* 1952; 117:500–544. [PubMed: 12991237]
11. Hund TJ, Rudy Y. Rate Dependence and Regulation of Action Potential and Calcium Transient in a Canine Cardiac Ventricular Cell Model. *Circulation*. 2004; 110:3168–3174. [PubMed: 15505083]
12. Kaupp UB, Seifert R. Molecular diversity of pacemaker ion channels. *Annu. Rev. Physiol.* 2001; 63:235–257. [PubMed: 11181956]
13. Kinard TA, de Vries G, Sherman A, Satin LS. Modulation of the bursting properties of single mouse pancreatic beta-cells by artificial conductances. *Biophys J.* 1999; 76:1423–1435. [PubMed: 10049324]
14. Kullmann PHM, Wheeler DW, Beacom J, Horn JP. Implementation of a fast 16-bit dynamic clamp using LabVIEW-RT. *J. Neurophysiol.* 2004; 91:542–554. [PubMed: 14507986]
15. Lau DH, Clausen C, Sosunov EA, Shlapakova IN, Anyukhovskiy EP, Danilo P Jr, Rosen TS, Kelly C, Duffy HS, Szabolcs MJ, Chen M, Robinson RB, Lu J, Kumari S, Cohen IS, Rosen MR. Epicardial border zone overexpression of skeletal muscle sodium channel SkM1 normalizes

- activation, preserves conduction, and suppresses ventricular arrhythmia. *Circulation*. 2009; 119:19–27. [PubMed: 19103989]
16. Lin RJ, Bettencourt J, Wha IJ, Christini DJ, Butera RJ. Real-time experiment interface for biological control applications. *Conf Proc IEEE Eng Med Biol Soc*. 2010; 2010:4160–4163. [PubMed: 21096883]
 17. Lu Z, Wu C-YC, Jiang Y-P, Ballou LM, Clausen C, Cohen IS, Lin RZ. Suppression of Phosphoinositide 3-Kinase Signaling and Alteration of Multiple Ion Currents in Drug-Induced Long QT Syndrome. *Sci Transl Med*. 2012; 4 131ra50.
 18. Luo CH, Rudy Y. A dynamic model of the cardiac ventricular action potential. I. Simulations of ionic currents and concentration changes. *Circ. Res*. 1994; 74:1071–1096. [PubMed: 7514509]
 19. Marder E, Goaillard J-M. Variability, compensation and homeostasis in neuron and network function. *Nature Reviews Neuroscience*. 2006; 7:563–574.
 20. Milesu LS, Yamanishi T, Ptak K, Mogri MZ, Smith JC. Real-time kinetic modeling of voltage-gated ion channels using dynamic clamp. *Biophys. J*. 2008; 95:66–87. [PubMed: 18375511]
 21. Prinz AA, Abbott LF, Marder E. The dynamic clamp comes of age. *Trends in Neurosciences*. 2004; 27:218–224. [PubMed: 15046881]
 22. Robinson HPC, Kawai N. Injection of digitally synthesized synaptic conductance transients to measure the integrative properties of neurons. *J. Neurosci. Meth*. 1993; 49:157–165.
 23. Scott, S. Stimulatio Simulations of Young Yet Cultured Beating Hearts (PhD thesis). Buffalo, NY: State University of New York; 1979.
 24. Sharp AA, O'Neil MB, Abbott LF, Marder E. Dynamic clamp: computer generated conductances in real neurons. *J. Neurophysiol*. 1993; 69:992–995. [PubMed: 8463821]
 25. Sun X, Wang HS. Role of the transient outward current (I_{to}) in shaping canine ventricular action potential: a dynamic clamp study. *J. Physiol*. 2005; 564:411–419. [PubMed: 15649977]
 26. Valiunas, V.; Clausen, C.; Butz, R.; Valiuniene, L.; Brink, PR.; Cohen, IS. Using the dynamic clamp and coupled cells to study pacemaker activity; *Proceedings of 2011 Keystone Symposia, Molecular Cardiology: Disease Mechanisms and Experimental Therapeutics*; February 22–27, 2011; Keystone, Colorado. 2011. p. 168
 27. Valiunas V, Kanaporis G, Valiuniene L, Gordon C, Wang HZ, Li L, Robinson RB, Rosen MR, Cohen IS, Brink PR. Coupling an HCN2-expressing cell to a myocyte creates a twocell pacing unit. *J. Physiol*. 2009; 587:5211–5226. [PubMed: 19736302]
 28. Verheijck EE, Wilders R, Joyner RW, Golod DA, Jumar R, Jongsma HJ, Bouman LN, van Ginneken ACG. Pacemaker synchronization of electrically coupled rabbit sinoatrial node cells. *J. Gen. Physiol*. 1998; 111:95–112. [PubMed: 9417138]
 29. Wang HS, McKinnon D. Potassium currents in rat prevertebral and paravertebral sympathetic neurones: control of firing properties. *J. Physiol*. 1995; 485:319–335. [PubMed: 7666361]
 30. Wilders R. Dynamic clamp: a powerful tool in cardiac electrophysiology. *J. Physiol*. 2006; 576.2:349–359. [PubMed: 16873403]

a: Test setup

b: Normal setup

Fig. 1.

HOST is a Windows-based PC used to perform voltage-clamp experiments. TARGET is a desktop PC that runs the dynamic-clamp program. Communication between HOST and TARGET (TCP/IP) uses a dedicated Ethernet connection. A: Test setup where the cell is doubly patched in voltage- and current-clamp modes. B: Normal setup where the cell is patched with a single electrode in current-clamp mode.

Fig. 2.

Simulink diagram forming the dynamic-clamp program that computes I_{K1} , I_f and I_{Na} . Small blocks on the left are model parameters. Blocks labeled “IK1,” “If” and “INa” are MATLAB functions that compute the currents. I_f and I_{Na} involve integration of gating variables requiring knowledge of time (clock symbol). Small circles denote summing; triangles denote gain. “Analog Input” supplies electrode potential, where blocks below it correct the potential for the IR drop across the electrode. “Analog Output” generates a voltage that is cabled to the current-clamp command input. The vertical bar in the upper right denotes a multiplexer, and provides a means of saving data in target-machine memory with subsequent uploading to the host machine. Changes to parameter values or the structure of the diagram can be done during an experiment.

Fig. 3.

Computer simulation of I_{Na} resulting from stepping V_m from a holding potential of -80 mV to three depolarized potentials (inset). Red: exact solution to the equations. Blue: solution by the dynamic-clamp algorithm running at an $8\text{-}\mu\text{s}$ sampling rate.

Fig. 4. Membrane potentials in a doubly patched HEK293 cell under voltage-clamp conditions (see Fig. 1a). V_{command} : voltage-clamp command potentials (-80 mV holding potential stepping in 20 -mV increments from -40 mV to $+60$ mV). V_{elec} : dynamic-clamp electrode potential. V_m : V_{elec} corrected for electrode IR drop due to the current being supplied by the dynamic clamp.

Fig. 5. Currents in a doubly patched HEK293 cell resulting from the pulse protocol seen in Fig. 4. Blue: currents recorded by the dynamic clamp. Red: currents predicted by a mathematical model that solves the equations exactly using V_m shown in Fig. 4 (see text for details).

Fig. 6.

Membrane potential in a HEK293 cell in current-clamp mode (see Fig. 1b). The dynamic clamp was programmed to add three currents: I_{K1} , I_f and I_{Na} . G_{K1} and $\bar{G}_f$ were set to constant values (see text), but $\bar{G}_{Na}$ was ramped from 0 to 100 nS/pF over 10 s. Blue: measured cellular response. Red: response predicted by a mathematical model.