BioSim: An Integrated Simulation of an Advanced Life Support System for Intelligent Control Research

David Kortenkamp and Scott Bell
Metrica Inc. and S&K Technologies
NASA Johnson Space Center/ER2, Houston TX 77058
kortenkamp@jsc.nasa.gov and scott@traclabs.com

Keywords Human space flight, simulation, integrated control, advanced life support

Abstract

This paper describes a simulation of an advanced life support system. An advanced life support system is one in which many resources are reused or regenerated. Such systems will be necessary for long duration human missions, such as those to Mars, where resupply opportunities are limited. Advanced life support systems are tightly integrated and have low margins. Thus, they require complex control strategies to maintain balance and to optimize system resources. The simulation presented here provides a platform for testing and development of autonomous control strategies for advanced life support systems.

1. Introduction

Advanced life support systems require complex control strategies that can maintain system balance with small margins and minimal buffers. These control strategies must deal with continuous and discrete processes and with the dynamical interactions between sub-systems such as air revitalization, water recovery, food production, solid waste processing and the crew. The goal of autonomously controlling an advanced life support system challenges many areas of research, including distributed data management and control, sensor interpretation, planning and scheduling, modeling and simulation, and validation and verification of autonomous control systems. These various research areas must be integrated into a coherent control architecture that can react to continuous changes and still plan long-range activities.

This paper describes a simulation of a typical advanced life support system. The goal of the simulation is to provide a development platform that allows autonomous control researchers to test out and compare their approaches on a standard, verified set of models. We start with a description of advanced life support systems and then introduce the simulation and its implementation.

1.1 The Domain

Advanced life support (ALS) provides basic life-sustaining functions inside spacecraft or on planetary surfaces. ALS includes controlling pressure, temperature and humidity, providing usable water and breathable air, supplying acceptable food, and managing wastes. A typical advanced life support system consists of the following integrated components [from JSC Advanced Life Support Requirements Document available at http://advlifesupport.jsc.nasa.gov]:

- Air: stores and maintains vehicle/habitat atmospheric gases including pressure control, overall composition, and trace constituents.
- Biomass: produces, stores and provides raw agricultural products to the Food Subsystem (next

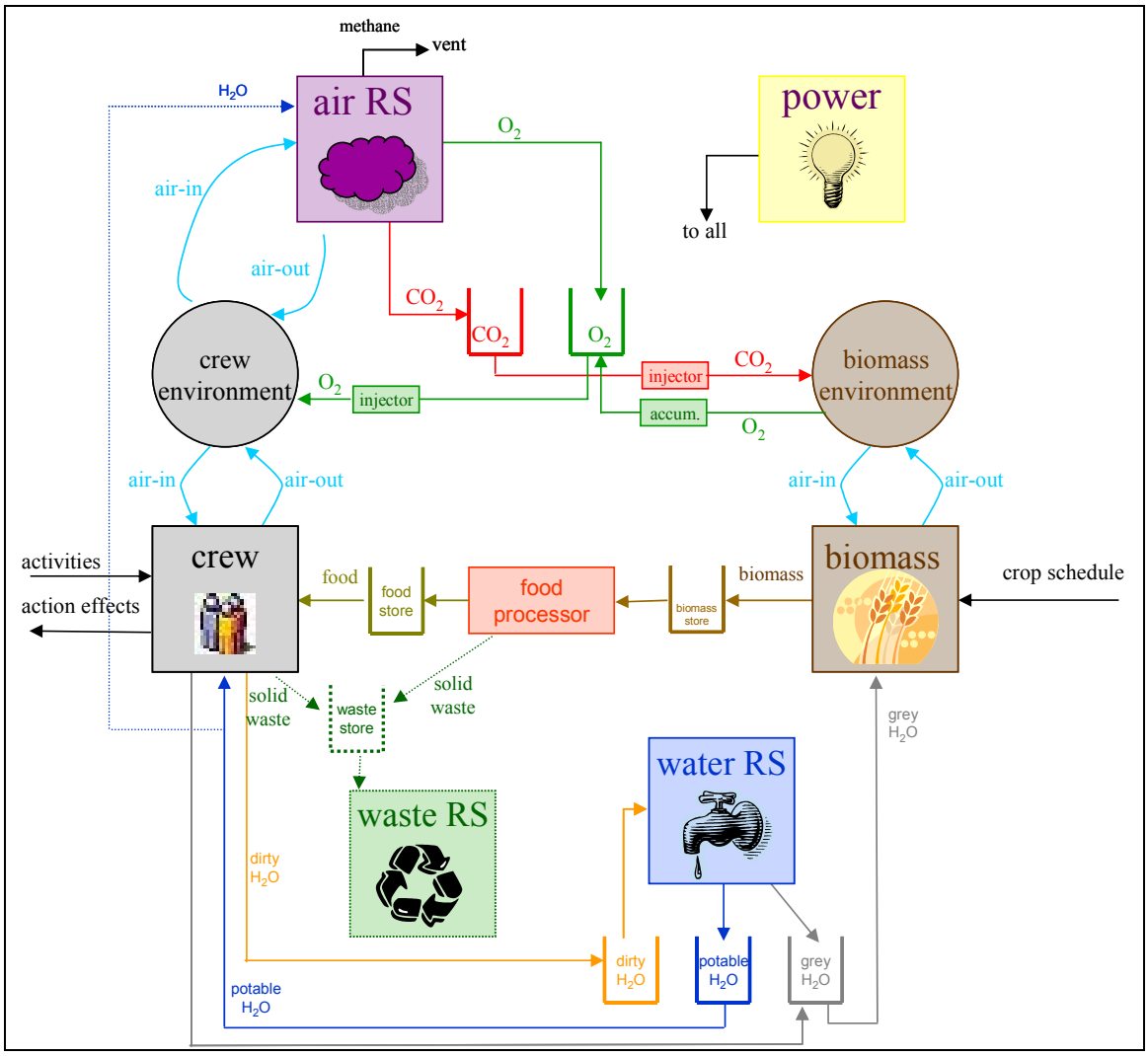

Figure 1 – BioSim modules and their interactions

bullet) while recycling air and water.

- Food: receives harvested agricultural products from Biomass Subsystem (above), stabilizes them and stores raw and stabilized agricultural products, food ingredients and pre-packaged food and beverage items. Transforms raw agricultural products into a ready-to-eat form via food processing and meal preparation operations.
- Thermal: responsible for maintaining vehicle/habitat temperature and humidity within appropriate bounds and for rejecting collected waste heat.
- Waste: collects and conditions waste material from anywhere in the vehicle/habitat, including packaging, human wastes, inedible biomass, etc. May also sterilize and store the waste, or reclaim life support commodities.
- Water: collects wastewater from all possible sources, recovers and transports potable water and stores. It then provides that water at the appropriate purity for crew consumption and hygiene.

Advanced life support systems differ from those

used currently on the space shuttle and space station in that they are designed to regenerate and reuse consumables to minimize mass and re-supply. This makes them essential for future long duration missions.

1.2. Simulation Overview

Figure 1 is an overview of our simulation, which consists of several life support modules that produce and consume resources. The simulation is a discrete event simulation with a fixed time step of one hour. Each module contains a model of that particular process. The modules take inputs (the resources consumed) and produce outputs (the resources produced) each time step based on the models. The resources are passed from module to module at the beginning of each one hour time step. Each and every arrow in Figure 1 (except for the transfer of air between the environments and the crew and biomass modules) represents a controllable parameter of the simulation. That is, at each time step an external control program can set the flows of each arrow in the simulation. In addition, many of the internal parameters of the modules are also settable from an external control program. An application programming interface (API) allows access to all simulation parameters. The goal is to provide a development platform for autonomous control researchers.

2. The Simulation

The simulation consists of modules that implement the various components of an advanced life support system. These components include the crew (with an associated crew environment, i.e., the air they breath), the biomass (i.e., crops) production system with it's own environment, the food processing system, the water recovery system, the air revitalization system and the power production system. We currently do not model either the solid waste recovery system or the thermal control system. In this section we will describe each of these modules in detail.

The **crew** module is based on models provided in [4]. The number, gender, age and weight of the crew are settable as input parameters with a default mixed-gender crew of four. As the crew cycles through activities (sleep, exercise, recreation, etc.) they consume O_2, food and water and produce CO_2, dirty water and solid waste. The amount of resources consumed and produced varies by crew members and their activities. The crew's activities can be adjusted by external controllers or a default set of activities can be used. The crew module has an associated crew environment that contains the air they breathe. The initial size and gas composition (percentages of O_2, CO_2 and inert gases) are adjustable and the default is 1.54893 x 10^6 liters (from [6]) with a gas composition equivalent to sea level air. As the crew performs activities and breathes they change the gas mixture of the crew environment.

The **biomass** module contains crops. The types of crops planted and amounts of each type are adjustable as input parameters. The amounts of the different crops can also be adjusted during a simulation run. As crops grow they consume CO_2, potable or grey water and light. They produce O_2 while they grow and produce biomass when they are harvested. The production and consumption of resources are modeled according to [3] and [JSC Advanced Life Support Program Baseline Values and Assumptions Document available at http://advlifesupport.jsc.nasa.gov]. Currently only wheat crops are modeled, but several other crops will be added shortly. The biomass

module has its own environment that contains the air they "breath." We keep the biomass environment and crew environment separate because the ideal gas composition for growing plants is different from the safe gas composition for people [1]. The simulation is modular and can be initialized with a single environment for crew and crops or with multiple environments for different crops –– the default is two environments. Harvesting and planting of crops is currently done automatically with the default that the same type of crop is planted as was harvested. However, different crop scheduling can be done during simulation runs by an external controller. Crops are planted and harvested in shelves that contain 8.2 square meters of growing area and have their own lights. The lighting and water available to the crops is adjustable. We are working on linking crew activities to planting and harvesting of crops.

The **food processing** module is currently very simple. It takes in harvested biomass and energy and produces edible biomass and solid waste. The edible biomass passes to the crew module to be used as food. Food processing is currently automatic, but we are working on linking crew activities to food production.

The **air revitalization** module scrubs CO_2 from the crew environment and also produces O_2. It takes air in from the crew environment at a rate that is adjustable by external controllers. It returns air with a reduced amount of CO_2 back into the crew environment. The CO_2 removed from the air is put into a CO_2 store, where it is available to be injected into the biomass environment when necessary. This module also produces O_2 from water which is placed in the O_2 store where it is available to be injected into the crew environment when necessary. Methane is produced as a by-product, which could be used for rocket fuel, but is currently simply vented. A separate O_2 accumulator takes O_2 out of the biomass environment and places it into the O_2 store. A control challenge is to maintain an optimal gas mixture in the crew and biomass environments while minimizing energy use by the accumulator and air revitalization module and while minimizing store sizes. The capacities of the stores are settable at initialization. The air revitalization module is based on a recently completed test at NASA Johnson Space Center [5].

The **water recovery** module consumes dirty water from the crew and produces potable and grey water (i.e., water that can be used for washing but not drinking). The water recovery module consists of four subsystems that process the water. The biological water processing (BWP) subsystem is required to remove organic compounds. Then the water passes to a reverse osmosis (RO) subsystem, which makes 85% of the water passing through it grey. The last 15% is passed to the air evaporation subsystem (AES), which recovers all of the remaining water. These two streams of grey water are passed through a post-processing subsystem (PPS) to create potable water. An external controller can turn on or off various subsystems. For example, all water can pass through the AES at a higher energy cost. We based our water recovery module on a recently completed test at NASA Johnson Space Center [2].

The **power production** module supplies electricity to all of the other modules. There are two models for this module. One simulates a nuclear-style power system that supplies a continuous, fixed amount indefinitely. A second simulates a solar-style power system that supplies a varying amount of power indefinitely. An external control program can set the amount of power going to each module up to the

amount of power available.

2.1 Malfunctions and stochastic processes

When evaluating a control system for life support systems it is not enough to look at how it performs in nominal situations. Life support systems will malfunction and it is important for the control system to identify and respond to these malfunctions and continue the mission safely. We have implemented malfunctions in each module and provided an API to introduce those malfunctions at any time in the simulation. Each module can have malfunctions introduced of varying degrees of severity and temporal length. For simplicity, the malfunctions have been broken into two categories based on temporal length: permanent and temporary; and three subcategories of severity: low, medium and high. These malfunctions are interpreted differently by each module. For example, a temporary but severe malfunction in the potable water store would be a large water leak. A permanent but low severity malfunction in the power production module would be the loss of a small part of a solar array.

Each module can experience multiple malfunctions at the same time and the control system must detect them, schedule the crew to repair them (if repairable), and monitor to make sure the repairs went accordingly. Scheduling the crew to do repairs can be done automatically (i.e., the crew schedule is changed by the simulation) through the malfunction API or by directly manipulating the crew schedule through the crew module API. Permanent malfunctions are non-repairable and require the control system to reallocate resources to continue the mission. A permanent malfunction with the water recovery system, for example, might cause a decrease in potable water. The control system could react by lowering available water to the plants to provide enough water to the crew.

Stochastic models have been added to the simulation to provide perturbations in the data returned to external controllers. These perturbations are currently modeled using a Gaussian function with different parameters for each module. The variance of the Gaussian function can be decreased or increased using an API call and can be set to zero to make the simulation deterministic.

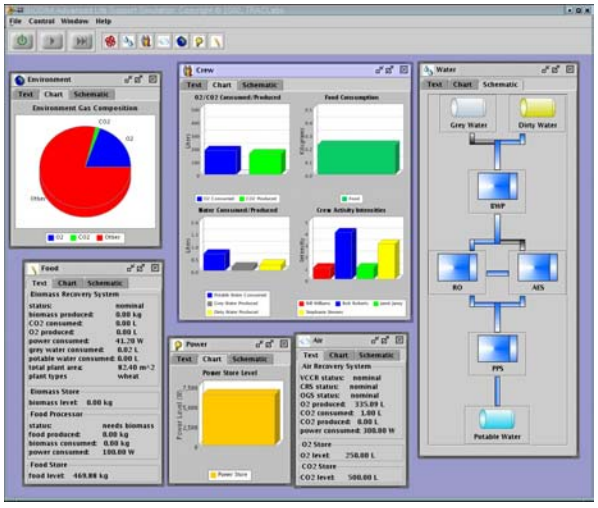

Figure 2 – BioSim's graphical user interface

2.2 User interaction

There are two methods for user interaction with the simulation. We have implemented a Java graphical user interface (GUI) that displays data from the simulation and allows limited commanding of the simulation. Each module has three views of its inner processes and resource levels: a text view, a chart view, and a schematic view. The text view simply has name/value pairs that directly describe the state of the module. A graph view displays relevant aspects of the module using graphs and charts. The schematic view provides a layout of the module's subsystems and

their interactions. The GUI exists to give the user a quick view of what the simulation is doing and give a modicum of control.

The second method for controlling the simulation is through an application programmers interface (API). The API defines all of the methods for setting the initial configuration of the simulation, running the simulation, gathering data from the simulation and sending control commands to the simulation. The API uses CORBA (see next subsection), which allows an external controller to be written in any language that supports CORBA and to be run on a separate computer from the simulation. The next section (controlling the simulation) provides more details on the API.

2.3 Implementation Details

The simulation is written entirely in Java for its portability and ease of use. We have tested the simulation using Unix, Windows and Macintosh operating systems. Using IBM's latest virtual machine for Java we achieve approximately 200 simulation hours per second on a single desktop PC. The modules interact with each other using the Common Object Request Broker Architecture (CORBA). CORBA also allows the simulation to run completely distributed and in parallel, so each module can be run on separate and remote machines to increase the speed of the simulation. Furthermore, CORBA also lets any language implementing an Object Request Broker (ORB) to communicate with the simulation. Most modern programming languages implement an ORB, including Java, C++, C, Lisp, etc. Thus, an autonomous control researcher can write an ALS controller in the language and platform of their choice and connect to the simulation over the network via CORBA. The simulation also provides a logging facility that outputs simulation data to XML or a database.

3. Controlling the Simulation

The initial configuration of the simulation (number of crew, size of atmosphere, capacity and initial level of stores, number of crops, and many others) is settable from the API before the simulation starts running. This is necessary because the optimal configuration of an ALS system is still an open question.

Our simulation is a discrete event simulation with a fixed time step of an hour; we have abstracted the time step to a simulation **tick**. Each module has a tick method that advances that module's state from t to t+1, i.e., advances its state one hour We have provided a BioDriver class for the convenience of the user which supplies basic control methods over the simulation. BioDriver has a method to advance the entire simulation one tick, i.e., all modules are "ticked" sequentially. While each module is run sequentially, data is cached so that all modules use data generated from the previous tick, which has the effect of making each module appear to run in parallel.

We expect the simulation to be controlled in two ways. First is a dynamic mode in which a controller ticks the simulation once to advance it one hour, then looks at the result, makes any control decisions, advances the simulation another hour and repeats the process. This approach is meant to approximate a real-time controller of an advanced life support system.

Second is a batch mode whereby the simulation is setup by the controller and then told to run for a fixed number of ticks (hours) or until the consumable resources become dangerously low for the crew. The control system can then look at the final states, make a

decision about what initial conditions should change and run the simulation again. We present a simple example of this approach in the next section.

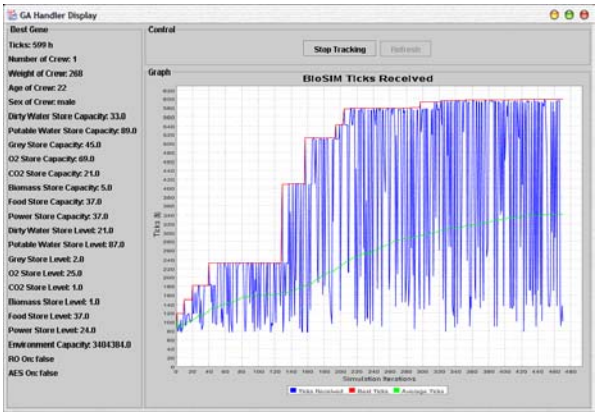

Figure 3 Results of a genetic algorithm optimizing initial conditions of BioSim

4. An Example of Using the Simulation

To explore how BioSim may be used for autonomous control systems research, a simple genetic algorithm (GA) was implemented. The GA consists of a handler and many nodes. Each node consists of a running instance of BioSim. The "gene" of the genetic algorithm is a description of the initial configuration of BioSim (e.g., crew size, store capacities, etc.). The handler sends a gene to a node, which configures BioSim accordingly and runs it until consumable resources become too low and the mission is ended. The node then returns back to the handler the length of time the simulation ran. Thus, the BioSim itself is the fitness function for the GA and the length that the simulation ran is the result of the fitness function. Good genes (i.e., those configurations that resulted in the longest running time for the simulation) are crossed, mutated or inverted and returned to a node to be run on the simulation again. Bad genes are replaced by genes that had a higher fitness. The nodes can be run in parallel to provide faster answers.

Figure 3 shows the results of running the simulation. The X axis are trials using BioSim (about 600), the Y axis are the run lengths of BioSim during each trial. The top red line represents the best gene (i.e., the longest run), which increases and then plateaus as the number of trials increases. The jagged blue line is the run length of each and every trial – it varies significantly as new configurations are tested. The middle green line is a running average of the lengths of all of the simulation runs.

5. Future Work

We plan to continue to improve the accuracy of our simulation. We will add additional crops (rice, soybeans, sweet potatoes, peanuts, lettuce, etc.). This will allow planning and scheduling researchers to investigate crop and menu planning. Also, our current stochastic models are simple Gaussians. We plan to derive specific probabilistic models for each module from test data.

An open research question is evaluation metrics for ALS controllers. We would like to have a measure of mission productivity, which could be used to compare different control strategies. Mission productivity would be computed based on how long and how well the crew members work on pertinent activities. Shortages of O_2, water, food, sleep, exercise, etc. will reduce the mission productivity. Another metric is to measure the peak capacity of stores with the goal of minimizing store sizes (and thus weight). We are working with ALS engineers to define other appropriate metrics.

The simulation infrastructure also needs additional work. A one hour time step was chosen for simplicity of implementation. However, different modules and different control strategies may need different time

steps. We are working on adjustable time steps to give the simulation both fine and coarse grained modes.

6. Conclusion

This simulation is being developed as a challenge to the artificial intelligence community. It presents many opportunities in different areas of research, including planning and scheduling (both crew activities and crop plantings), fault detection and recovery, resource optimization, adaptive control and integrated architectures. While the simulation is evolving as ALS technologies change, the underlying control challenges remain. If you would like to download BioSim go to:

http://www.traclabs.com/biosim

References

[1] Barta, et al., "The Biomass Production System for the Bioregenerative Planetary Life Support Systems Test Complex: Preliminary Designs and Considerations," *SAE Paper 1999-01-2188, Proceedings 29th International Conference on Environmental System (ICES),* 1999.

[2] Bonasso, R. P., Intelligent Control of a NASA Advanced Water Recovery System. In *Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*: A New Space Odyssey. Montreal, 2001.

[3] Edeen, M. et al, "Modeling and Validation of the Ambient and Variable Pressure Growth Chamber Models," *SAE Technical Paper Series 932171*, 1993.

[4] Sara Goudarzi and K.C. Ting, "Top Level Modeling of Crew Component of ALSS," *Proceedings International Conference on Environmental Systems*, 1999.

[5] Debra Schreckenghost, Daniel Ryan, Carroll Thronesbery and R. Peter Bonasso, "Intelligent Control of Life Support Systems for Space Habitats," *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*, 1998.

[6] Tri, T. O., "Bioregenerative Planetary Life Support Systems Test Complex (BIO-Plex): Test Mission Objectives and Facility Development," *SAE Paper 1999-01-2186, 29th International Conference on Environmental Systems*, 1999.