

# The SMASH toolbox

## Course overview



Sandia National Labs is managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a subsidiary of Honeywell International, Inc., for the U.S Dept. of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# **Objective:** more powerful and efficient data analysis using the SMASH toolbox

- This course includes
  - Some advanced MATLAB concepts
  - Core toolbox features
  - Short exercises
  - Supervised tinkering

# Why MATLAB?

- Quickest path for technical computing
  - Unified environment: commands, graphics, ...
  - Comprehensive documentation
  - Math is natural, not an afterthought
  
- Most efficient use of your time
  - “Free” alternatives ignore labor cost
  - Subject expertise costs more than software

# What is SMASH?

- Sandia Matlab Analysis Hierarchy
  - Data management (read files, archive results)
  - Numerical computation (smoothing, FFTs, ...)
  - Visualization (1D/2D graphs, ...)
  - User interfaces (Dialog boxes, ...)
  
- Goals
  - Share code, analysis, and results
  - **Don't start from scratch every time**



# What's inside?

- Utilities: toolbox assistance
- Programs: self-contained code, graphical interface
- Packages: general-purpose code grouped by topic
  
- System requirements
  - MATLAB 2017b or later suggested
  - Mac, Windows, or Linux
  - No Mathworks toolboxes are needed\*

# Toolbox installation

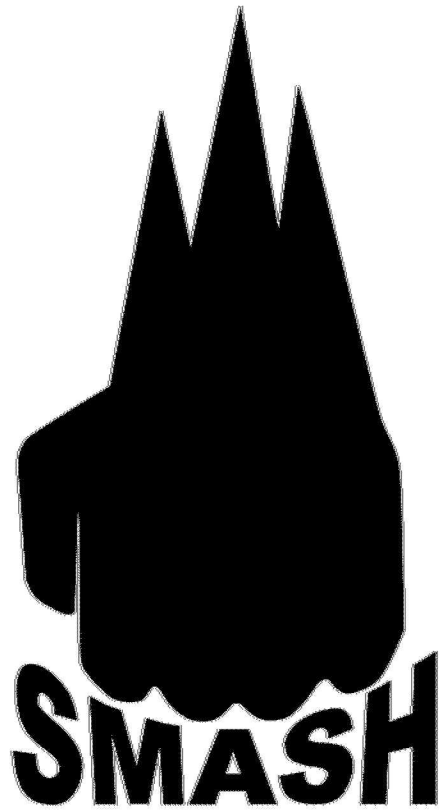
- Internal gitlab (<https://gitlab.sandia.gov>)
  - **Download** ZIP or clone Git repository
  - External GitHub also available
- Navigate MATLAB to toolbox directory
  - Run `install/installSMASH` function
  - Modifies current session path
  - Start up file for future sessions
- Verify install
  - `smashroot` command

# Getting help

- When in doubt, type:
  - `help SMASHtoolbox`
  - `doc SMASHtoolbox`
  - Tab completion is your friend
- Additional resources
  - readme.md file on gitlab/GitHub page
  - Internal wiki (Sandiapedia)
  - Version 1.0 manual (SAND2016-6848)
  - User meetings

# Course Agenda

- Utilities, programs and **packages**
- Arrays, structures, and **objects**
  
- Signal analysis
- File access
- Image analysis
  
- Individual projects
  - Suggested features or things that interest you



# The SMASH toolbox

Utilities, programs, and packages



Sandia National Labs is managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a subsidiary of Honeywell International, Inc., for the U.S Dept. of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# Utilities versus programs

- Utility functions help access the toolbox
  - smashroot
  - SMASHtoolbox
  - loadSMASH
  
- Programs are MATLAB code with:
  - Specific purpose and a graphical interface
  - Formal distinction can be ambiguous, but it's usually obvious in practice

# SMASH programs

- Located in (smashroot)/programs folder
  - Separate folders for each program
  - Usually a main function with a private folder (where most of the code resides)
  - May or may not reference other parts of the toolbox
    - Some predate the toolbox by 5-10 years
    - Many toolbox features drawn from the older programs

## Exercise

- Use the `SMASHtoolbox` utility
- Run the `MCdemo` program to estimate the value of  $\pi$ 
  - Use the start, pause, and reset features
  - How many iterations are needed to ensure at least two-digit accuracy? What about three digits?



## Using programs (continued)

- Programs can be launched directly **if** they are on the MATLAB path
  - `loadSMASH -program (name)`
- Some programs behave like ordinary functions when inputs are passed

## Exercise

- Load MCdemo on the MATLAB path
- Call `MCdemo` from the command window
  - Read the program's help entry
  - Bypass the graphical interface interface to quickly run one million iterations

# Program survey

- Analysis:
  - PointVISAR (VISAR reduction)
  - SIRHEN and THRIVE (PDV/PDI reduction)
- Synthetic data
  - pyrosim (pyrometer simulator)
  - fringen (VISAR/PDV fringe generator)
- Other tools
  - datninja (digitize figure data)
  - SDAbrowser (Sandia Data Archive support)

# MATLAB packages

- Folders begin with a plus sign: `/+mytools`
- Anything inside the package can be accessed with dot notation
  - `[...]=mytools.functionA(...)`
- Packages can be nested within packages
  - `[...]=mytools.subpack.functionA(...)`

## Example: checkDisplay function

- SMASH.Graphics.DisplayTools.**checkDisplay**
  - SMASH package
  - Graphics subpackage
  - DisplayTools subpackage
  - checkDisplay function
- Tab completion is your friend!

# Are all those dots really necessary?

- Import packages to workspace

```
loadSMASH -package Graphics.DisplayTools.*
```

- Import packages as a name space

```
name=loadSMASH('-package',...  
'SMASH.Graphics.DisplayTools.*');  
name.CheckDisplay()
```

- Similar to Python “import as”

- Note: empty parenthesis are important

- MATLAB also has its own import command, but I don't recommend it

# Most toolbox code resides in packages

- +SMASH
  - +SignalAnalysis
  - +FileAccess
  - +ImageAnalysis
  - Much, much more...

## Exercise: the Reference package

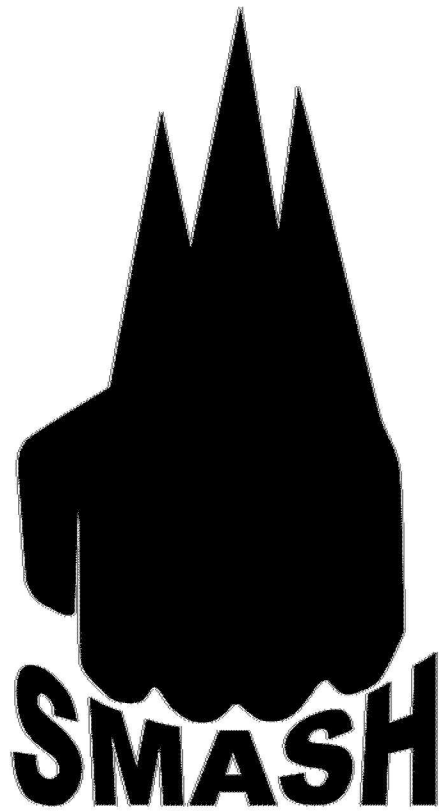
- Locate the Reference package
- Find the PhysicalConstant function
- Look up the value of  $hc$ 
  - Plank's constant times the speed of light
  - Roughly 1240 eV\*nm
  - Convert 532 nm to photon energy

$$E = \frac{hc}{\lambda}$$



# Summary

- Programs are:
  - Self-contained code with graphical interface
  - Added to the path
  
- Packages are:
  - Organized directories starting with a “+” sign
  - Called with dot notation or imported
  
- `loadSMASH` is your friend



# The SMASH toolbox

Array, structures, and objects



Sandia National Labs is managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a subsidiary of Honeywell International, Inc., for the U.S Dept. of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# Numeric arrays are natural in MATLAB

- **Exercise:**
  - `time=linspace(0,1,50);`
  - `signal=cos(2*pi*5*time);`
  - `plot(time,signal)`

# What about metadata?

- Information about the data
  - When/where was it taken?
  - What equipment was used?
  - General notes, ...
  
- Information can be stored as separate variables
  - Very fragile

# Compound variables

- Mix data of different type/size in a single variable
- Cell arrays (curly braces)
  - Data indexed by number
  - `data{1}`, `data{2}`, ...
- **Structures**
  - Data indexed by name

## Exercise: structured data

- Store data
  - `data.Time=linspace(0,1,50);`
  - `data.Signal=cos(2*pi*5*time);`
- Store metadata
  - `data.Date='March 29'`
  - `data.Equipment='Fancy digitizer'`
- Structure fields can be used like any other variable
  - `plot(data.Time, data.Signal)`

# Object oriented programming (OOP)

- Custom data type that defines
  - Properties (where information goes)
  - Methods (operations that can be performed)
- What's the difference?
  - Properties are things (**nouns**)
  - Methods are actions (**verbs**)

# Why bother with objects?

- Data can be protected from the user
  - Enforce types (numeric, character, etc.)
  - Controlled names (can't put something in the wrong place)
- Methods know where the data is and are context-aware
  - `view(A)` uses view method for object A
  - `view(B)` uses view method for object B
  - A and B may be different classes altogether...



# Notation

- Properties accessed with dot notation
  - `value=object.Name`
  - `object.Name=value` % (if allowed)
- Two ways for accessing methods
  - `[...]=view(object,...)` % MATLAB style
  - `[...]=object.view(...)` % most languages
  - Generally produce identical results

## Exercise: Signal class

- Class defines object properties and methods
  - Object is an instantiation of the class
  
- Object construction
  - `object=Signal(time,signal)`
  - NOTE: class resides in `SMASH.SignalAnalysis` package, so dot notation or package import is required

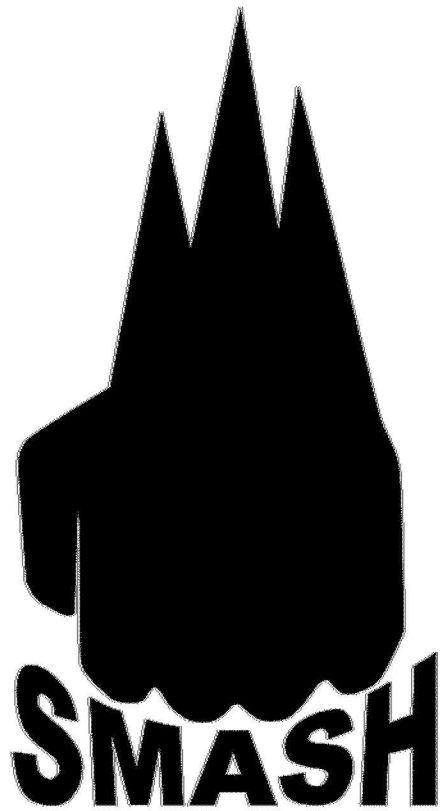
# Learning about an object

- Help/doc statement on class name (including package location)
  - `help(object)` or `doc(object)` also works
- Command window displays a hyperlink whenever object is called without a semicolon
- Browsing properties/methods helps understand what a particular class does

# Summary

- Data can be managed with
  - Numeric arrays (plus extra variables)
  - Compound arrays (structures)
  - Custom objects
  
- Object oriented programming (OOP)
  - Properties store information, enforce limits
  - Methods are context-specific functions
  - Goal: **guide and reign in the end user**

You do not have to be able to write an  
object class to benefit from using  
objects



# The SMASH toolbox

## Signal analysis



Sandia National Labs is managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a subsidiary of Honeywell International, Inc., for the U.S Dept. of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# What is a Signal?

- One-dimensional (scalar) data defined on a one-dimensional grid
  - Digitizer records: grid is time, data is voltage
  - Spectrometers: grid is wavelength, data is power/counts
- Grid may or may not be uniformly spaced
  - Must be monotonically increasing/decreasing
  - Values must be unique

# Learning about the Signal class

- doc SMASH.SignalAnalysis.Signal
  - Class overview
  - Object construction
  - Property descriptions and permissions
  - Method documentation



# Creating a Signal object

- Existing MATLAB variables
  - `object=Signal(grid,data)`
- View method creates a plot
  - `view(object)`

# Exercise

- Create a sinusoid
  - `x=linspace(0,1,80);`
  - `y=cos(2*pi*10*x+2*pi*rand(1));`
  - Use the view command to display plot
- Modify GridLabel and DataLabel properties for the sinusoid object
  - Suggestions: 'Time (ns)' and 'Signal (V)'
  - Call view method

# Grid and Data properties

- Read-only access
  - `x=object.Grid` is valid
  - `object.Data=y` is invalid
  
- Reset method allows manual overwrite
  - `object=reset(object,x,y);`
  - Faster than creating a new object

# Grid/Data modifications

- Shift/scale methods modify the grid
  - `object=scale(object,value)`
  - `object=shift(object,value)`
- Object arithmetic automatically applied to Data
  - `object=object+1;`
  - `object=2*object;`

# Exercise

- Convert sinusoid time base from nanoseconds to seconds
  - Scale by  $1\text{e-}9$
- Shift grid by  $42\text{e-}9$  s
- Multiply data by ten
- Plot the results

## Other grid modifications

- Crop removes information outside bounds
  - `object=crop(object,bound); % irreversible`
  
- Limit method focuses inside a bound
  - `object=limit(object,bound); % reversible`
  
- Regrid changes grid and interpolates data
  - `object=regrid(object,new); % irreversible`
  - `object=regrid(object); % uniform grid`

# Fourier transforms

- Convert from time/space to frequency domain
  - `new=fft(object);` % result is a Signal object
- Extra inputs define transform options
  - `new=fft(object,'RemoveDC',true);`

## Exercise

- Calculate the power spectrum of the sinusoid object
  - Where is the peak located?
  
- Increase the number of frequencies for a smoother result
  
- Generate the complex power spectrum
  - How does the view method handle this?

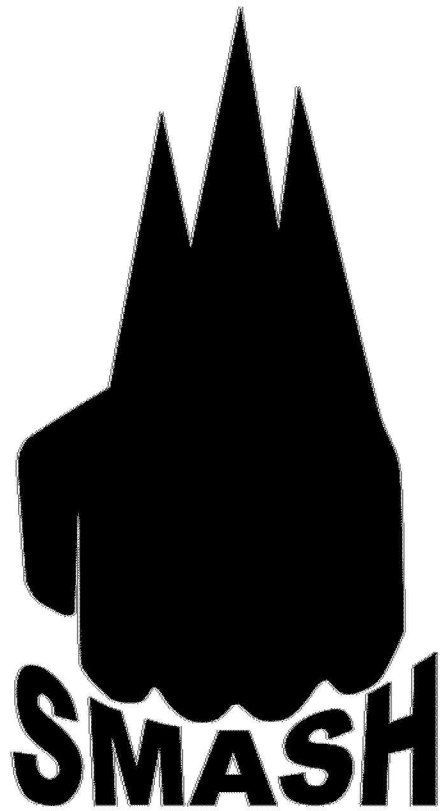


# Documenting your work

- Name property is for short tags
  - Example: 'Experiment TK421'
- Comments property is for verbose description
  - `object=comment(object); % editor window`
- No size restrictions in either case
  - Values must be text

# Summary

- Signal objects describe scalar data on a 1D grid
- Grid/Data properties cannot be set directly
  - Can be modified with scale, shift, reset, ...
  - Arithmetic support for Data
- Methods provide visualization, transforms, and much more



# The SMASH toolbox

## File access



Sandia National Labs is managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a subsidiary of Honeywell International, Inc., for the U.S Dept. of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# Overview

- Getting data into MATLAB
  - FileAccess package
- Storing data
  - Sandia Data Archive (\*.sda) files

# FileAccess package supports many formats

- Text files
  - 'column' : numeric columns with arbitrary header
  - 'block' : numeric column blocks separated by headers
- Digitizer files : 'agilent'/'keysight', 'tektronix', 'lecroy', ...
- Image files
  - 'film' densitometry scans
  - 'standard' graphic formats (\*.jpg, \*.png, ...)
  - Various camera formats ('optronis', 'winspec', ...)
- Laboratory binary files
  - 'dig' digital signals from NTS
  - 'pff' Portable File Format (legacy Sandia format)
  - 'sda' Sandia Data Archives

doc SMASH.FileAccess.SupportedFormats

# Raw data import

- `data=readFile(file,format,record)`
  - Format and record may be optional, depending on file type
    - Some formats have a unique file extension, others do not
  - Returns a structure
- Files can be probed without reading
  - `report=probeFile(file,format)`

## Exercise: tabular data

- Read the file “table.txt”
  - Hint: use ‘column’ format
  - Plot columns 2-3 versus column 1
- Probe the same file
  - Verify the number of header lines and columns

## Exercise: binary data from a digitizer

- Read the file “shot\_Ch1.wfm”
  - Hint: file came from a Tektronix digitizer
  - Plot signal
- Probe the file “Z2576\_T10\_SHOT.h5”
  - Hint: file came from an Agilent digitizer
  - How many signals are in this file?
- Read the file “Z2576\_T10\_SHOT.h5”
  - One record at time
  - All at once



# Signal class automatically calls readFile

- `object=Signal(file,format,record)`
- **Exercise:**
  - Create object from “table.txt”
    - Which columns are loaded?
  - Create object from “shot\_Ch1.wfm”
  - Create object form “Z2576\_T10\_SHOT.h5”
    - Which record is loaded?
- Many SMASH classes behave this way...

# Saving data

- Some classes provide an export method
  - Usually a text dump
  - Metadata usually lost
- Sandia Data Archive (\*.sda) files are a more powerful alternative
  - Any MATLAB variable (arrays, structures, objects, ...)
  - External files (documentation, ...)

# SDA overview

- Based on HDF5 standard
  - Portable across platforms/languages
  - SDA-specific Python library available
  - Better documented than MAT files
- Each variable associated with a unique text label and optional description
- Lossless compression available (deflate 0-9)

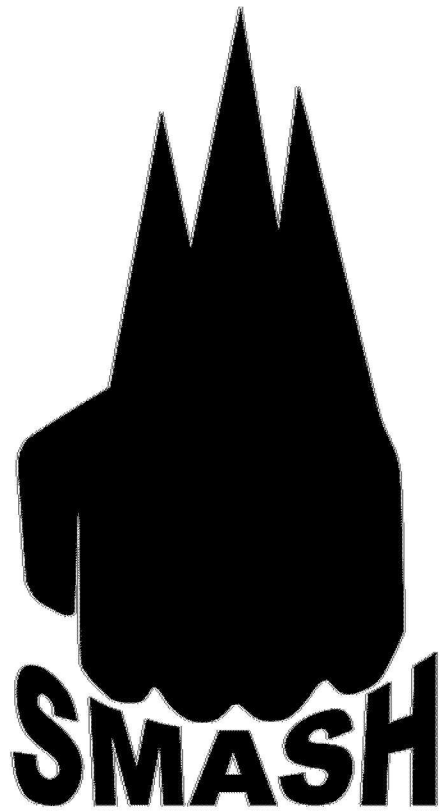
# SDAbrowser program

## ■ Exercise

- Run the SDAbrowser program
- Create a new archive file
- Save Signal object(s) to archive
- Clear workspace
- Restore object(s) to workspace

# Summary

- FileAccess package can read many text and binary formats
  - readFile, probeFile functions
  - Some classes call readFile automatically
- Sandia Data Archives (\*.sda files)
  - Portable format for storing records
  - Unique labels (order doesn't matter)
  - Supports **any** MATLAB variable\*



# The SMASH toolbox

## Image analysis



Sandia National Labs is managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a subsidiary of Honeywell International, Inc., for the U.S Dept. of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# What is an Image?

- One-dimensional (scalar) data on two one-dimensional grids
  - Gray scale image: grid1/grid2 are position, data is counts
  - Spectrograms: grid1 is time, grid2 is frequency, data is power
- Similar conventions as Signal
  - Uniform spacing (or not), monotonic grids, ...

# Learning about the Image class

- doc SMASH.ImageAnalysis.Image
  - Class overview
  - Object construction
  - Property descriptions and permissions
  - Method documentation



# Creating an Image object

- Existing MATLAB variables
  - `object=Image(grid1,grid2,data);`
    - Grids are 1D arrays, data is a 2D array
- Data stored in a file
  - `object=Image(file,[format],[record])`
  - No input launches interactive mode
  - Format/record may be optional, depending on the file type

# Exercise

- Create a 2D Gaussian
  - `x=linspace(-2,2,100); y=x;`
  - `[X,Y]=meshgrid(x,y);`
  - `object=Image(x,y, exp(-X.^2-Y.^2));`
- Use view method to see results
  - `view(object)`

## Exercise

- Create an Image object from Sandia logo
- Use the view method and fix the aspect ratio
  - Hint: look at GraphicOptions property OR MATLAB's "axis" command

# Why are Images displayed upside down?

- MATLAB puts the origin in the upper left corner
  - Common in a lot of image processing
  - Matrix convention
- This can be changed by modifying the `GraphicOptions` property
  - `object.GraphicOptions.YDir='normal';`
  - `view(object)`
- Lots of other graphic options (color map, ...)

# Exercise

- Images have three view modes
  - “show” mode used by default
- Apply the other two view modes on the 2D Gaussian image

# Close parallels between Image and Signal

- There are two grids to deal with
  - `object=scale(object,'Grid1',value);`
  - `object=shift(object,'Grid2',value);`
- Arithmetic operations apply to data (2D)
  - `object=2*object+1`
- Some methods accept two grid inputs
  - `object=crop(object,xb,yb);`

# Image orientation

- Flip grid direction
  - `object=flip(object,coordinate);`
  
- Direction or angle rotation
  - `object=rotate(object,'left');` % 90 degrees
    - Reversible
  - `object=rotate(object,value);` % any angle
    - Irreversible

# Exercise

- Use interactive rotation mode on MATLAB's default image
- Interactively crop the result around the face



# Image slicing

- Extract data at fixed grid1/grid2 location(s)
  - `result=slice(object,'Grid1',x); % vertical`
  - `result=slice(object,'Grid',y); % horizontal`
- Slices are SignalGroup objects
  - All class methods (view, etc.) are available

# Exercise

- Slice the Sandia logo at several locations
- Interactively slice that image
  - Hint: read the documentation

## Other Image methods

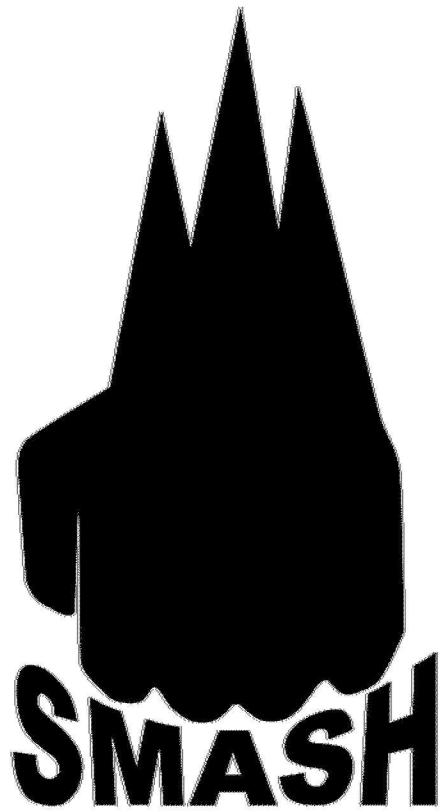
- bin: combine local blocks into super-pixels
- smooth/sharpen: low/high-pass filtering
- center: adjust grid origin
- register: adjust grid/data to master image
- ...and much more!

# Storing Image objects

- Export method dumps [x y z] values to a text file (not recommended)
  - Huge file sizes
  - Metadata lost
  
- Image/ImageGroup objects are fully SDA compatible
  - Retains data and metadata

# Summary

- Images describe scalar data on two 1D grids
  - Camera measurements
    - Spatial grids
    - Time/wavelength grids, ...
- Similar to Signals, with extra features
  - Multiple visualization modes
  - Rotation
  - Slicing



# The SMASH toolbox

## Toolbox summary



Sandia National Labs is managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a subsidiary of Honeywell International, Inc., for the U.S Dept. of Energy's National Nuclear Security Administration under contract DE-NA0003525.

# The SMASH toolbox provides

## ■ Utilities

- Code that makes the toolbox easier to use
- SMASHtoolbox, loadSMASH,...


## ■ Programs

- Self-contained code, graphical interface, specific purpose
- MCdemo, datninja, ...

## ■ Packages

- Everything else

# Core packages

- Signal and image analysis
  - File access
  - Other core packages
    - MUI
    - Graphics
    - System
    - General
- 



## Other general packages

- Arbitrary curve fitting
- Statistics (Monte Carlo and Bayesian)
- Reference (physical parameters, ...)
- Journal (publication figures, tables, ...)
- Region Of Interest selection
- And more...

# Specialized packages

- Instrument (digitizer control, ....)
- DynamicMaterials (EOS calculations ...)
- Spectroscopy analysis (pyrometry, ...)
- Velocimetry analysis (VISAR, PDV, ...)
- Xray analysis (imaging and diffraction)
- Z data (DAS signals, SVS data, ...)

# Getting involved

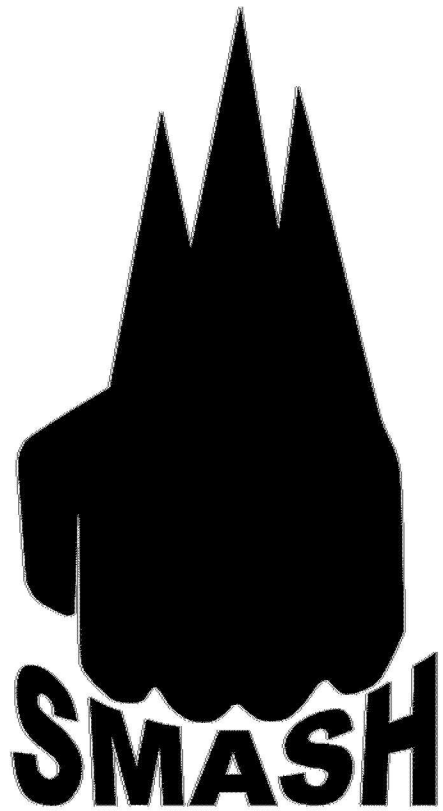
- Step 1: set up Git for version control
- Step 2: find a problem that interests you
- Step 3: figure out how this fits into the toolbox
- Step 4: **work safely**
- Step 5: commit/push/pull often (Git)
- Step 6: Participate in the user/developer meetings

# Work safely

- Don't modify core features (w/o checking)
  - Bug fixes and new features OK
  - Established behavior should be preserved as much as possible
- Use **meaningful** names
  - Properties use camel case: PeakLocation
  - Methods/functions use mixed case: findPeak
- Document your code (how is it supposed to work?)
- Avoid binary files in the repository\*

# We are always open to feedback

- Bug reports and feature requests
  - Submit to GitHub repository (Issues tab)
- Open discussion
  - Community forum on SMASH wiki
- **Limited** consulting for toolbox additions
  - Priority is experimentalist needs in 1600 (dynamic compression, plasma physics, pulsed power, ...)



# The SMASH toolbox

## Individual project suggestions



Sandia National Labs is managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a subsidiary of Honeywell International, Inc., for the U.S Dept. of Energy's National Nuclear Security Administration under contract DE-NA0003525.

## Project idea: Digitize figure data

- Call the `datninja` program
- Digitize the file “ScannedFigure.png”

# Project idea: Monte Carlo analysis

- Use the MonteCarlo.Cloud class to generate two random variables (x,y)
  - Use mean 0, variance 1
  - View cloud (standard and raw mode)
- Transform data to a new cloud
  - $u=x-y$ ;  $v=x+y$
  - $u=x.*$ ;  $v=x.^2 + y.^2$ ;
- Compare transformed clouds to original
  - Correlation
  - Normality



# Project idea: Curve fitting

- Load (x,y) data from the “NoisyPeak.txt” file
  - `data=readFile(filename,'column');`
  - `data=data.Data;`
- Create a Curve object with Gaussian basis function
  - Hint:  $\text{@}(p,x) \exp(-(x-p(1)).^2/(2*p(2)^2))$
- Optimize curve to fit data
  - Plot fit with data
- Analyze parameter uncertainty
  - Determine 90% confidence region

## Project idea: Short-time analysis

- Generate a sinusoid with variable frequency
  - $S(t) = \cos(2\pi \cdot 10 \cdot t + 50\pi \cdot t.^2)$  for  $t = [0 \ 1]$
- Create a STFT object from this data
- Partition the data into 0.2 time durations, advancing 0.01 between each duration
- Use the analyze method to generate a spectrogram
  - View the results

## Project idea: dialog boxes

- Start with MATLAB's `inputdlg` command
  - Ask the user for their name, quest, and favorite color
- Use the `MUI.Dialog` class to mix edit boxes with popup menus
- Add “OK” button that checks answers **before** closing the dialog