12.010 Computational Methods of Scientific Programming

Lecturers
Thomas A Herring
Chris Hill

Review of last Lecture

- Examined parallel computing and MPI implementation.
- Today: Class project and graphics formats

Class project Groups

Stefan Gimmillaro	The Sodoku Master:
Amanda Levy	Solar Subtense Program
Adrian Melia	Model of an accelerating universe
Eric Quintero	Encryption/decryption algorithm
Karen Sun and Javier Ordonez	Truss Collapse Mechanism
Melissa Tanner and Sean Wahl	Phase diagram generator for binary and ternary solid-state solutions
Celeste Wallace	Adventure Game

Class Project

- Aim of the project is to put together everything you have learned in this course:
  - Algorithm design
  - User interface
  - Numerical methods
  - Graphics output
- We will spend some time over next two lectures introducing some concepts that might assist in these projects.

Ground rules

- Some of you will be working in groups. The role of each person in the group should be explained in the project description.
- There is a written and code components to the projects (see web page of requirements).
- You can solve the problem in any language that you want to use
- For those that choose Fortran, C or C++, they can choose their a graphics program to display results.

Problem statement

- Each group should develop a problem statement that defines the problem they are trying to solve.
- Some projects are driven by differential equations in the acceleration and/or velocity of a body will depend on forces applied to the body
- We will look at the case on a gravity driven orbital problem.

Governing equations for gravity-type problems

- Force = mass*acceleration ($F=ma$)
- Force = GM_1M_2/R^2 where M_1 and M_2 are two masses and R is the distance between them.
- The force is directed along the vector between the bodies
- For multiple bodies, the force acting on each body is the vector sum of the forces from all the other bodies.
- What do you do with these equations?

Solution to equations

- Given the initial positions of all the bodies, the accelerations of each body can be computed.
- Acceleration integrated, gives the velocity change (remember bodies are initially moving)
- Velocity integrated gives position change.
- At the new positions, the forces will be different and therefore there will be different accelerations and velocities.
- How we quantity the changes in positions and velocities?

Integration methods

- Simplest method is Euler's method: Compute accelerations; DV = A*dt;
  P(t+dt) = P(t) + (V(t)+Adt/2)*dt
  V(t+dt) = V(t) + A*dt
- Compute new accelerations at P(t+dt) and continue to iterate.
- This is simplest and most inaccurate method.
- Why is it inaccurate?
- For your projects, write down the basic equations and how these will be solved.

Runge-Kutta integration

- Compare the two versions of the second-order system
- y'' = f(x,y,y')

$$y_{n+1} = y_n + h[y_n' + \frac{1}{6}(k_1 + k_2 + k_3)] + O(h^5)$$

$$y_{n+1}' = y_n' + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

$$k_1 = hf(x_n, y_n, y_n')$$

$$k_2 = hf(x_n + h/2, y_n + hy_n'/2 + hk_1/8, y_n' + k_1/2)$$

$$k_3 = hf(x_n + h/2, y_n + hy_n'/2 + hk_1/8, y_n' + k_2/2)$$

$$k_4 = hf(x_n + h, y_n + hy_n' + hk_3/2, y_n' + k_3)$$

Form with no Velocity dependence

- y'' = f(x,y)

$$y_{n+1} = y_n + h[y_n' + \frac{1}{6}(k_1 + 2k_2)] + O(h^4)$$

$$y_{n+1}' = y_n' + \frac{1}{6}k_1 + \frac{2}{3}k_2 + \frac{1}{6}k_3]$$

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf(x_n + h/2, y_n + hy_n'/2 + hk_1/8)$$

$$k_3 = hf(x_n + h, y_n + hy_n' + hk_2/2)$$

Graphics

- As we have seen some languages (Matlab and mathematica) already have graphics imbedded in them
- Languages such as Fortran, C and C++ do not explicitly contain graphics but graphics routines can be included in programs if a graphics library can be found.
- Often graphically output needs to be included in other documents (such as reports and web pages)
- Quality of the display depends very much on the type of graphics used.
- For graphics there are too many “standards”

Types of graphics files

- Graphics files fall into two basic types:
  - Vector graphics, made of lines and objects (e.g., fonts) that can scaled. The most common of these is Postscript (in its various forms)
  - Bitmapped graphics defined by pixels that have certain characteristics. The most common of these is GIF (Graphics Interchange Format). These types of formats are good for images especially with continuous tone changes.
- Bitmapped graphics can look very bad when re-scaled.
- Some formats are a composite of each type (e.g. Mac PICT format).

Basic Graphic formats

- **Postscript:** Comes in different versions (level 1, 2 and 3) and as encapsulated (EPS).
- EPS files may have a preview of image which is often displayed in WYSIWYG word processors. (The preview image may not always be displayable).
- Postscript is meant for printers and can maintain very high resolution
- Not all printers can print all levels of postscript and not all printers understand all parts of postscript.
- Postscript is a graphics programming language.

Other vector formats

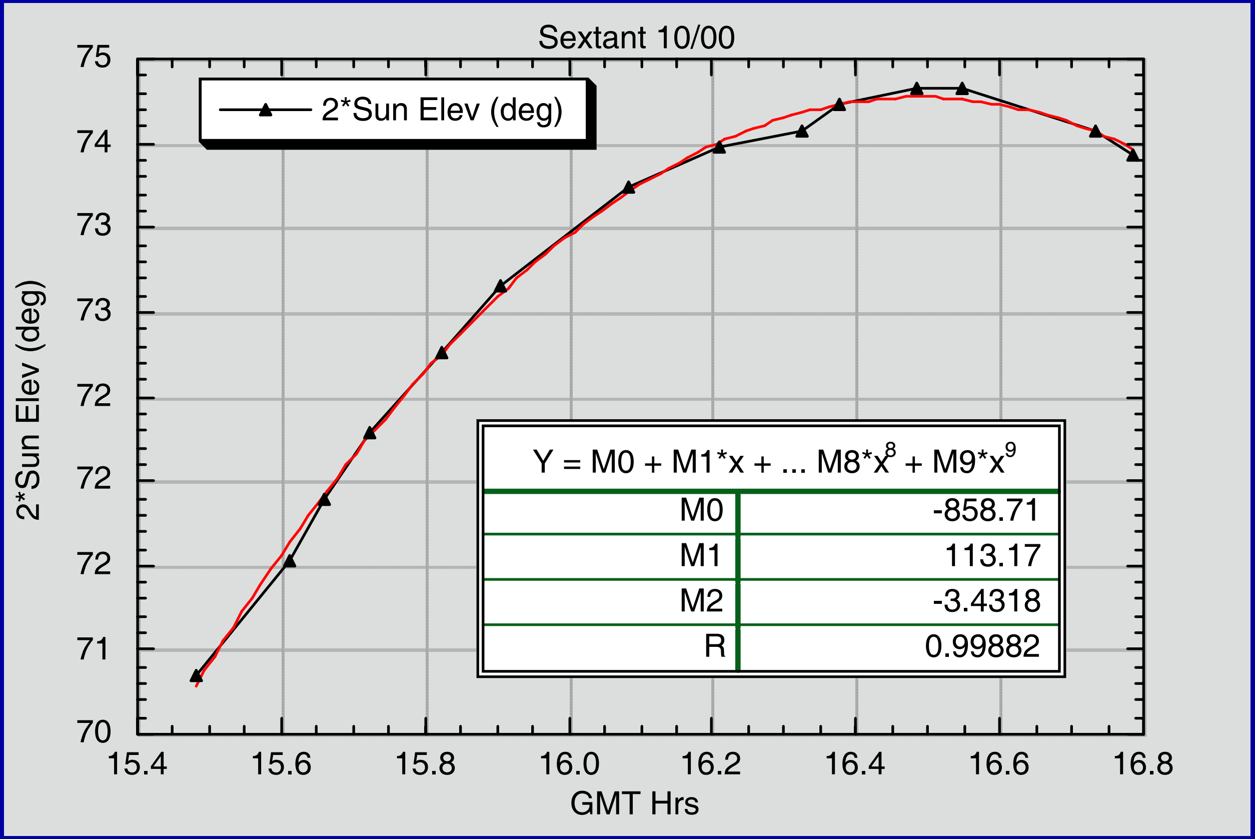
- Often “drawing” program (opposed to “painting” programs) us a vector graphics format that is unique to the program (e.g, Macdraw).
- Advantages:
  - Small file sizes
  - Scalable
  - Good for may scientific graphics plots such as line drawings

Pixel formats

- GIF is the most common but enforcement of the patent on the compression algorithm used has raised concerned
- Network Portable Graphics (PNG) is an alternative but a lot of older software will not handle this.
- JPEG (Joint Photographic Experts Group) also common and good for image data
- These format represent an image as pixels which have attributes such as color. When image is resized, the pixels need to be merged and mapped to the new location.

Examples:

- Output from Kaleidagraph in PICT format

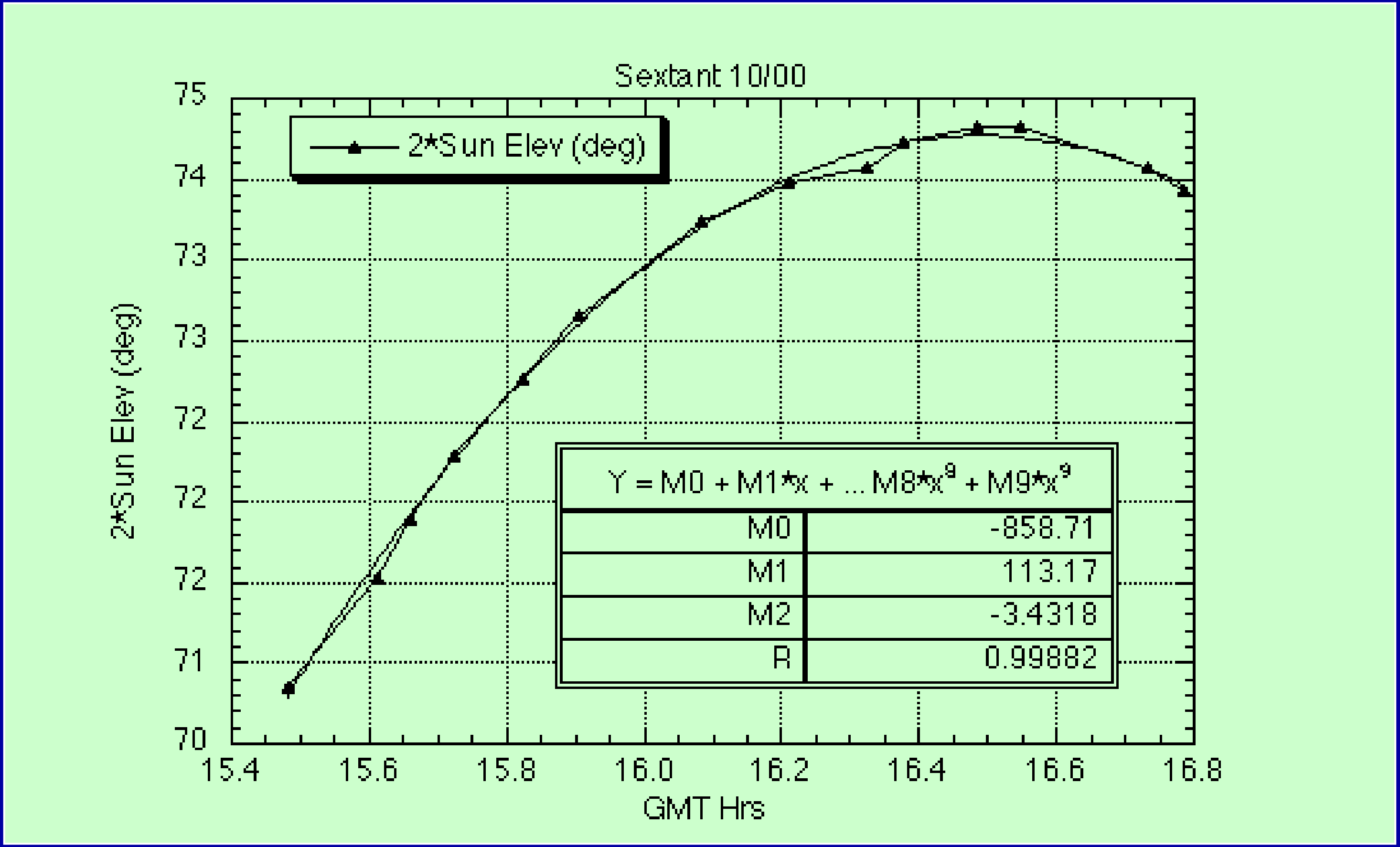


$Y = M0 + M1*x + ... M8*x^8 + M9*x^9$

M0	-858.71
M1	113.17
M2	-3.4318
R	0.99882

Bitmap format

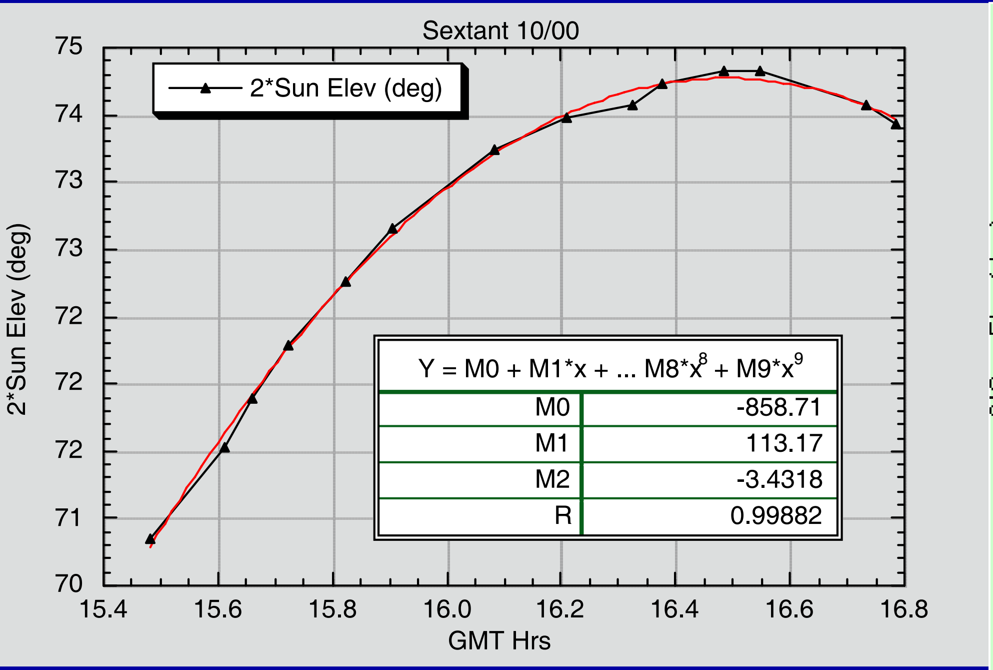
- Basic bit-mapped graphic



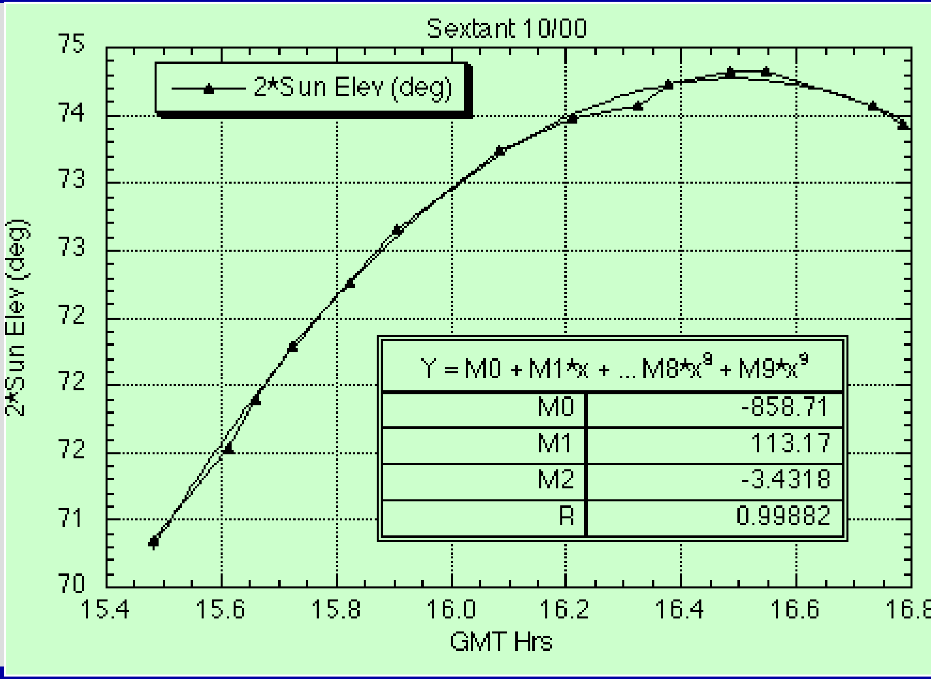
Y = M0 + M1*x + ... M8*x⁹ + M9*x⁹	
M0	-858.71
M1	113.17
M2	-3.4318
R	0.99882

Rescaled versions

- PICT and bitmap



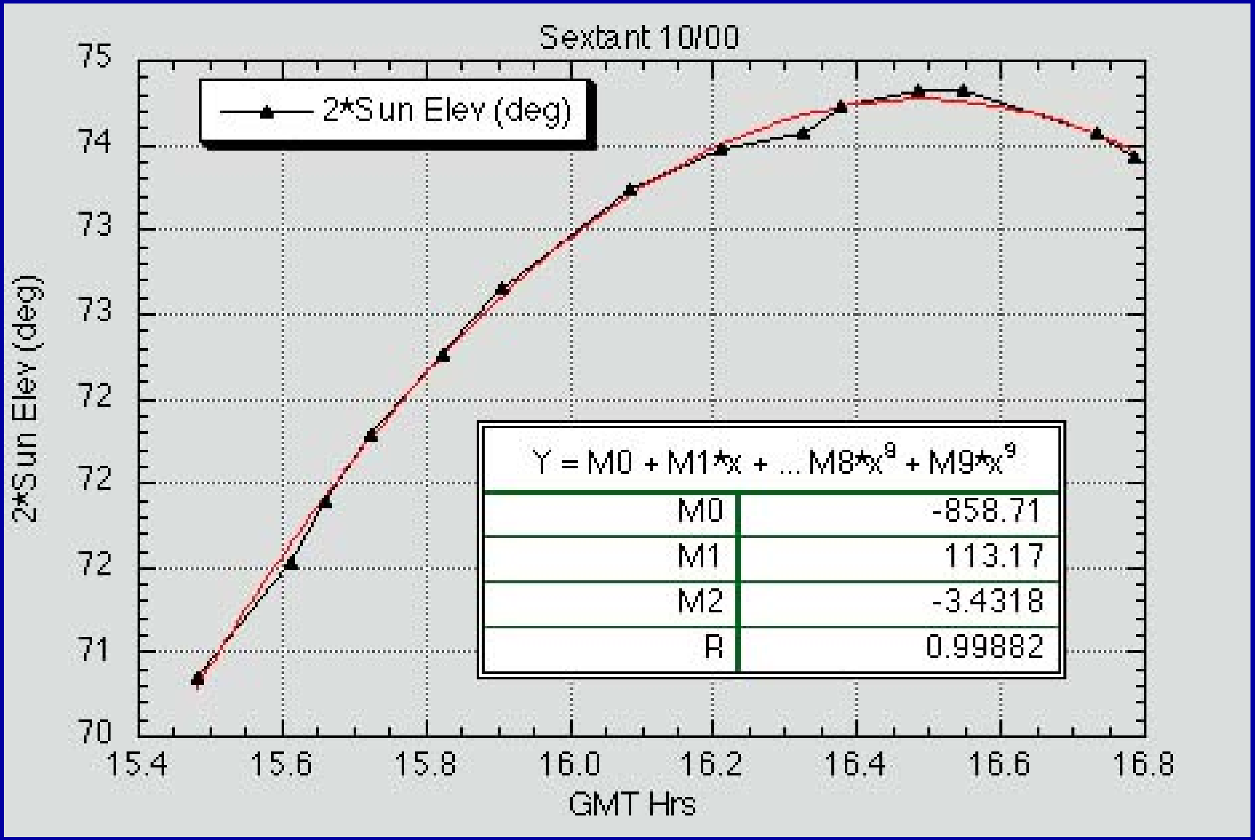
$Y = M0 + M1*x + ... M8*x^8 + M9*x^9$	
M0	-858.71
M1	113.17
M2	-3.4318
R	0.99882



$Y = M0 + M1*x + ... M8*x^8 + M9*x^9$	
M0	-858.71
M1	113.17
M2	-3.4318
R	0.99882

JPEG Version

- Medium Resolution

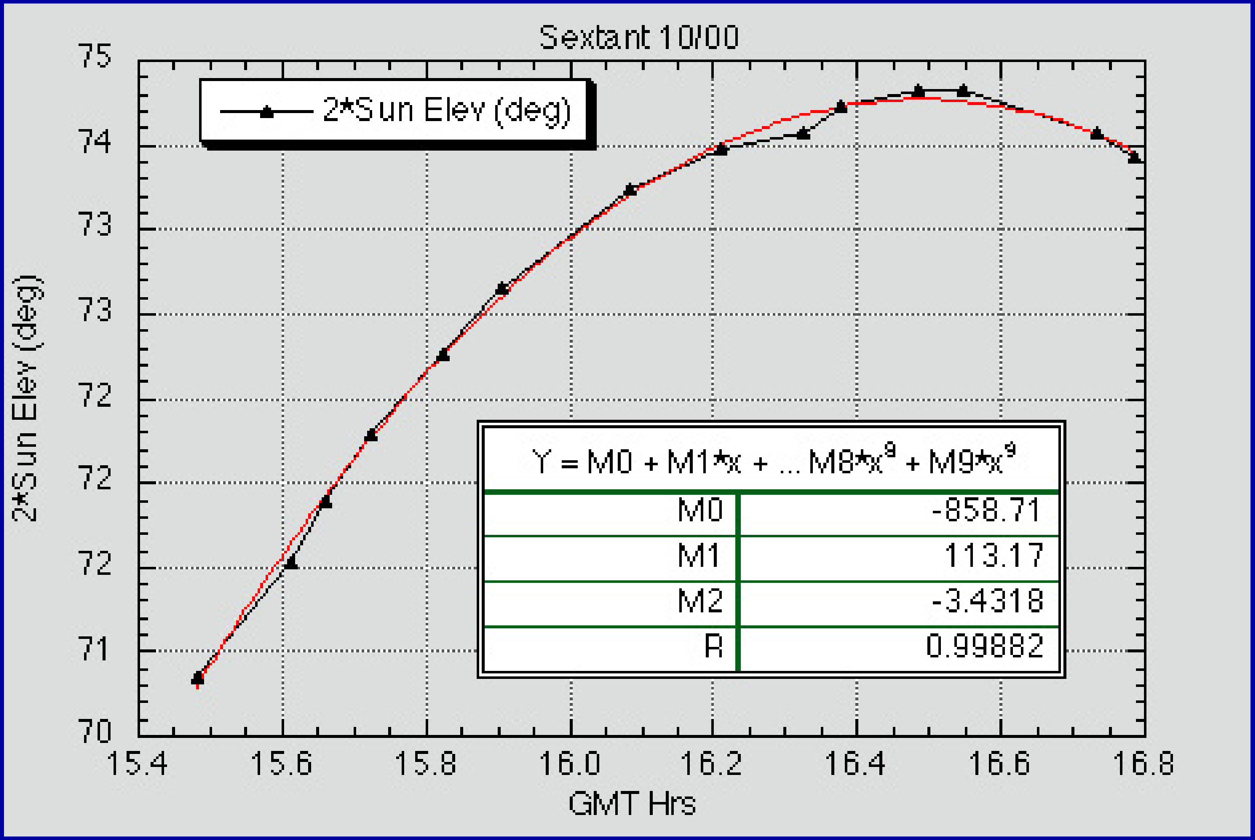


Y = M0 + M1*x + ... M8*x^9 + M9*x^9

M0	-858.71
M1	113.17
M2	-3.4318
R	0.99882

JPEG

- Maximum resolution

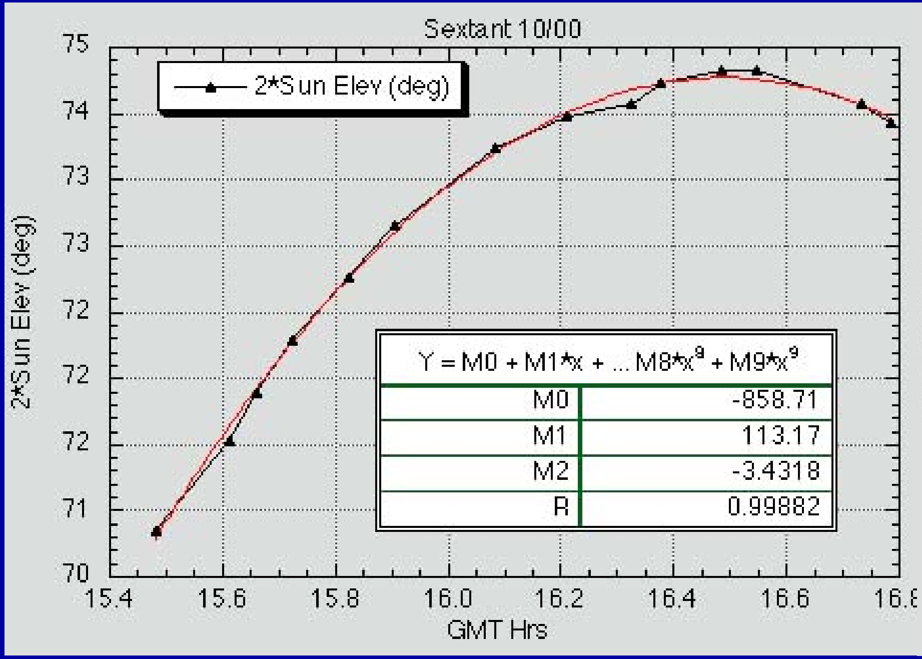


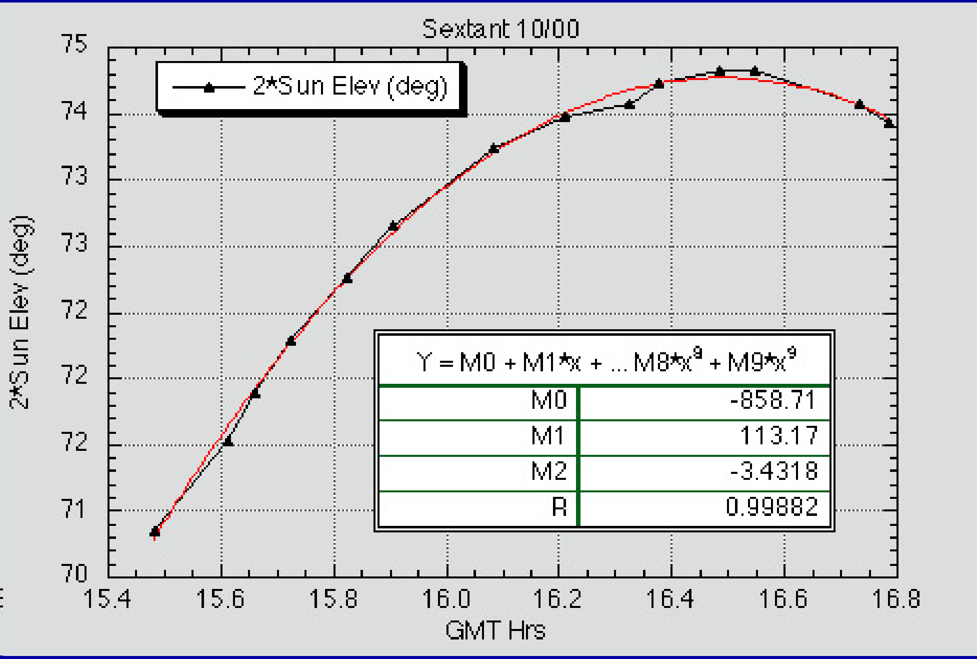
Y = M0 + M1*x + ... M8*x^{9} + M9*x^{9}

M0	-858.71
M1	113.17
M2	-3.4318
R	0.99882

Scaled versions of JPEG

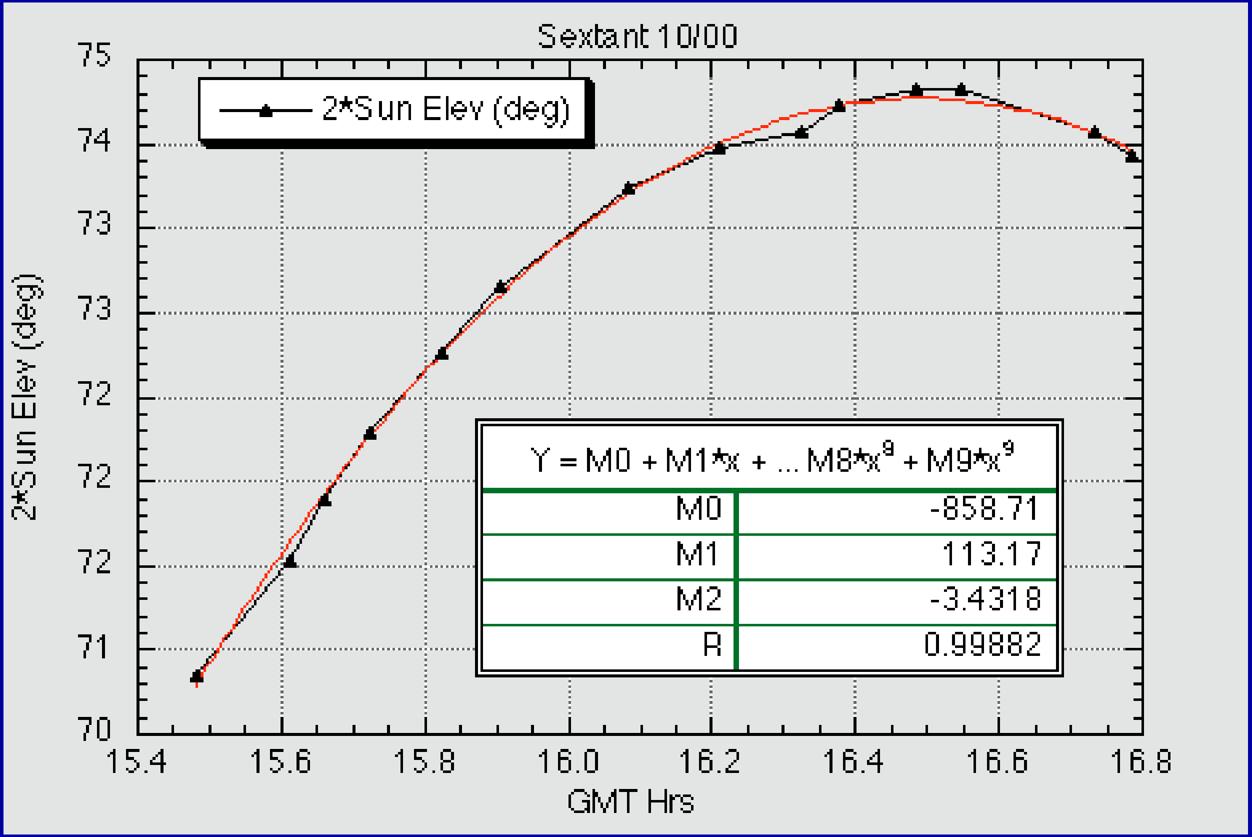
- Effects of scaling graphics





EPS Files (from Photoshop)

- 1.1 Mbytes (displays badly but will print very well)

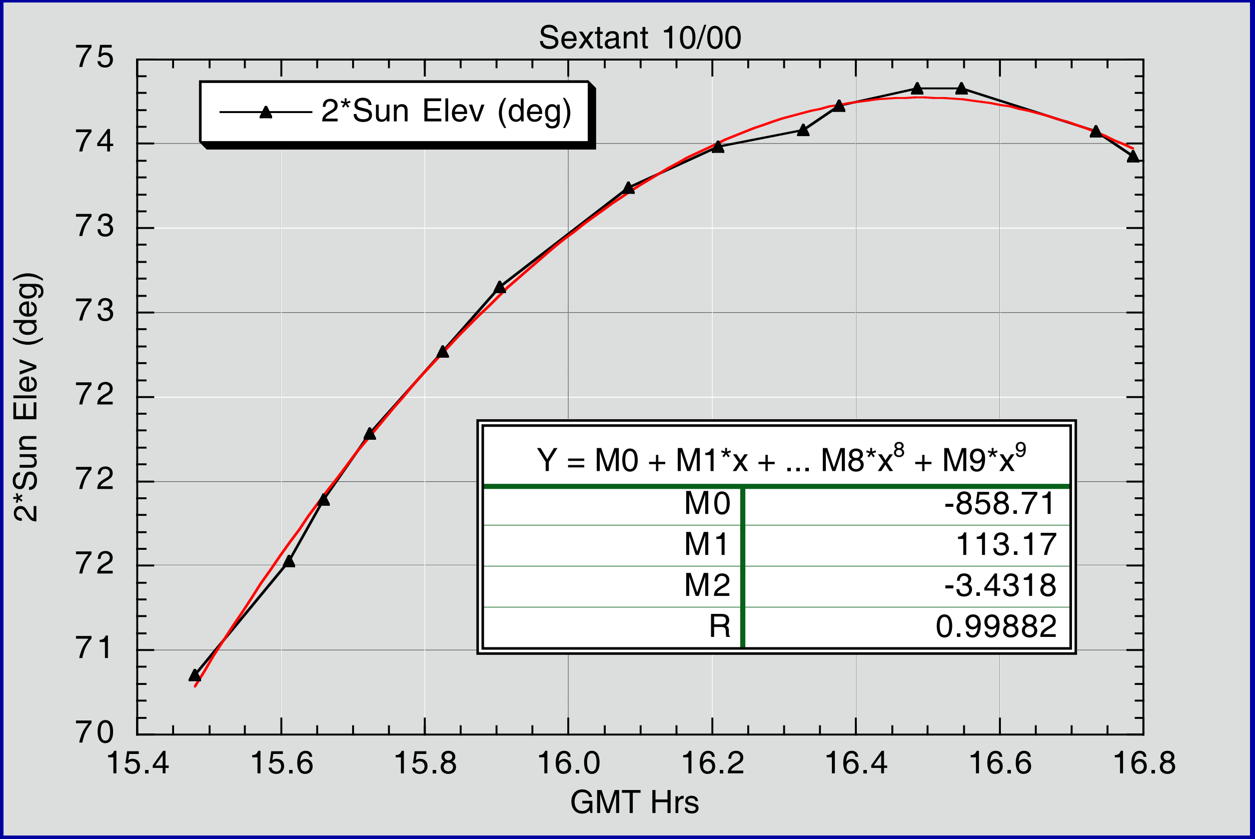


$Y = M0 + M1*x + ... M8*x^9 + M9*x^9$

M0	-858.71
M1	113.17
M2	-3.4318
R	0.99882

EPS Direct from Printing

- (This figure will print with high resolution but displays badly)



$Y = M0 + M1*x + ... M8*x^8 + M9*x^9$

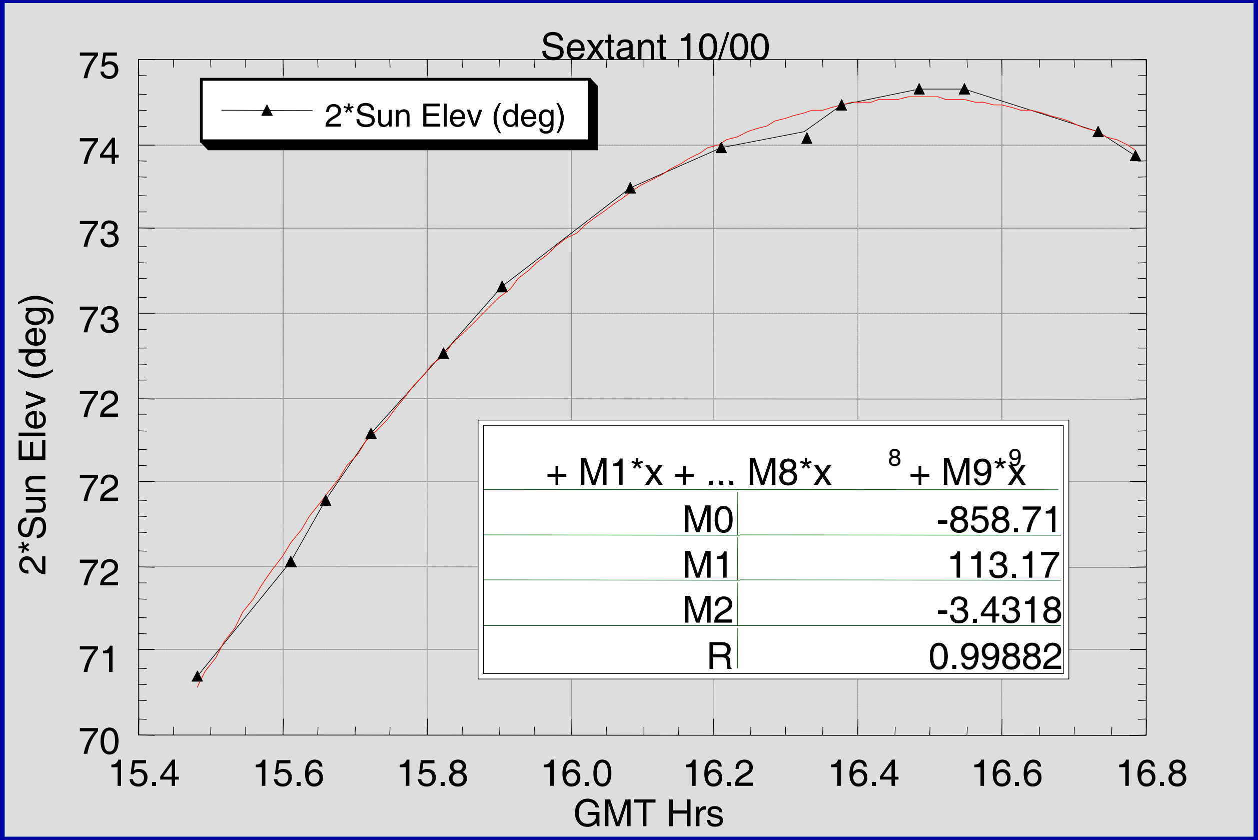
M0	-858.71
M1	113.17
M2	-3.4318
R	0.99882

File sizes

- For the different images we just looked at:

Original KG figure	32K (24K no data)
PICT	8K
Bitmap	12K
JPEG (medium)	44K
JPEG (maximum)	68K
EPS (photoshop)	1.1M
EPS (KG)	40K (no fonts)
EPS (KG+fonts)	256K

Microsoft Graphics (Object based)



+ M1*x + ... M8*x^8 + M9*x^9

M0	-858.71
M1	113.17
M2	-3.4318
R	0.99882

Issue with graphics

- Increasingly documents need to be prepared electronically and generating acceptable graphics is one of the biggest problems still.
- For electronic presentation of images, best if the original graphics is generated at 72 dots-per-inch. If the graphics is not scaled then these can print OK as well.
- For printing, vector graphics are most often the best.
- Display quality: Try antialiasing (process to smooth out edges but some times makes graphics look fuzzy.
- For WYSWYG word processing, encapsulated postscript generates good prints but may not be rendered in electronic versions (such as web pages).
- Currently: Still a lot of testing to see what works best.

Graphics packages in programming

- There are graphics programs that can be purchased or are open source in which data can be supplied and graphics generated
- There are also graphics libraries that can used and incorporated into user developed software.
- Some packages come in both forms
- Example: Generic Mapping Tool (GMT) available from http://gmt.soest.hawaii.edu/
  Includes program that can be scripted and routines that can be used in user developed software
- Caution with these programs is that calls and features can change with versions

Summary of Today's class

- Aim of the project is to put together everything you have learned in this course:
  - Algorithm design
  - User interface
  - Numerical methods
  - Graphics output
- Graphics formats and issues.
- Class evaluation Thursday 12/04
- **Order of presentations will be set on Thursday as well.**