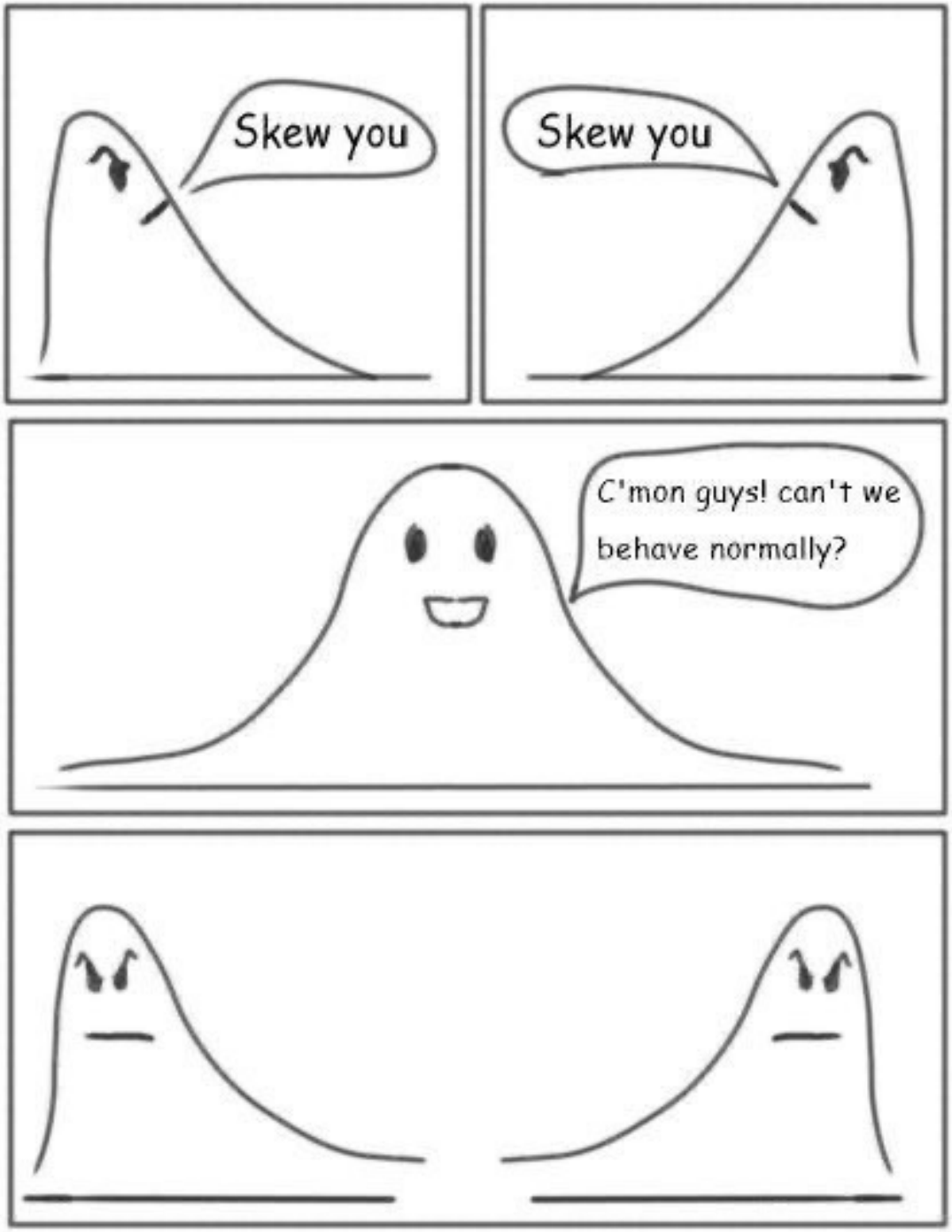
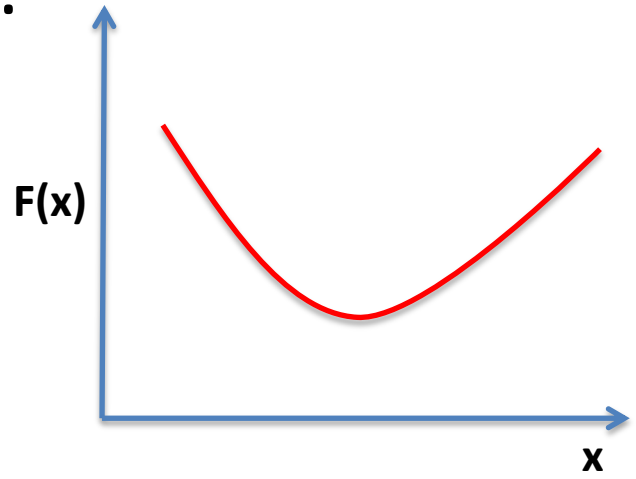

Skew you
Skew you
C'mon guys! can't we behave normally?

Newton's Method of (1-D) Optimization

- We have the 1-D function F(x):



- It is natural to approximate this with a parabola. We thus truncate its Taylor series after the quadratic term, and use the approximation to get the next estimate of the critical point.

Newton's Method of (1-D) Optimization

- $F(x) = F(x_k) + (x - x_k)F'(x_k) + 0.5(x - x_k)^2F''(x_k) + \ldots$
- We seek x^* where $F'(x^*) = 0$
- Thus, $\boxed{x_{k+1} = x_k - F'(x_k)/F''(x_k)}$
- This is virtually identical to root solving via Newton's method.
- In fact it <u>is</u> root solving, just the root of $F'(x)$

Successive Parabolic Interpolation

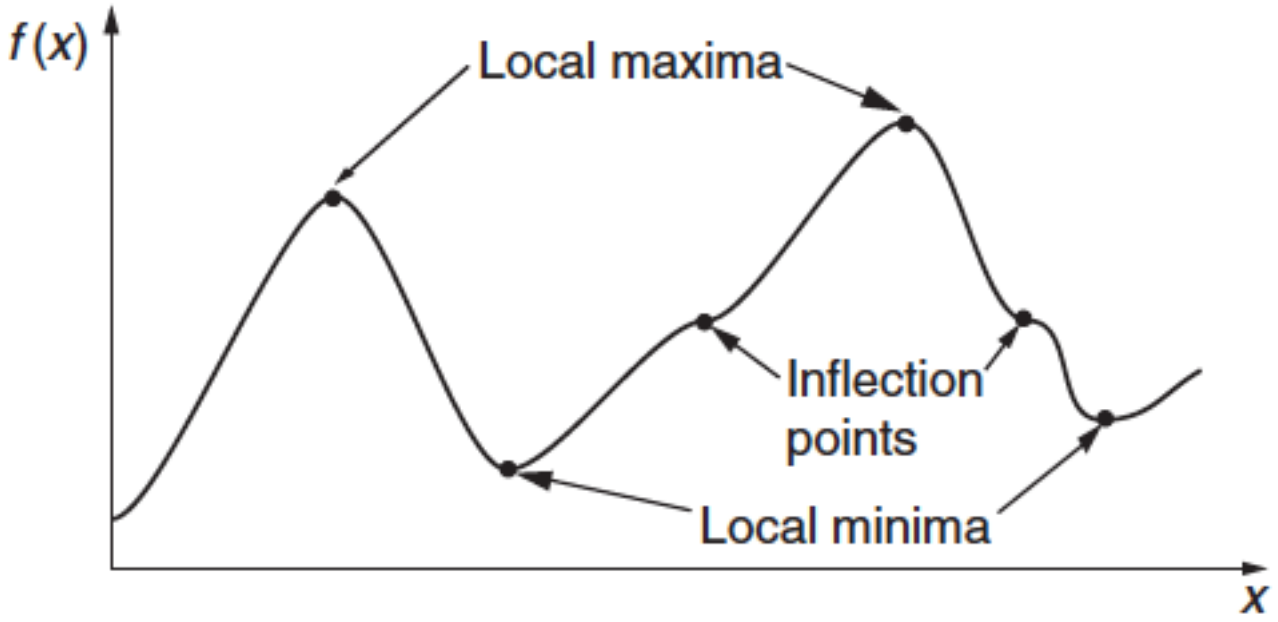
- Successive Parabolic Interpolation is very similar to the secant method for root solving.
- Instead of using two points to get a line, we use three points to get a parabola!
- Thus if we have x_k, x_{k-1}, x_{k-2} we fit $y = ax^2+bx+c$ to the points:

  $(x_k,F(x_k))$, $(x_{k-1},F(x_{k-1}))$, $(x_{k-2},F(x_{k-2}))$
- Once we have a,b and c, we seek the critical point: $dy/dx = 0 = 2ax + b$
- Thus, $x_{k+1} = -b/2a$

Successive Parabolic Interpolation

- We iterate forward, keeping the last three points.
- The rate of convergence is 1.324.
- This is superlinear, but not as fast as Newton's method (r=2).
- Both techniques go to critical points – either maxima or minima – and have convergence problems if you don't start close enough to the correct answer!

Different types of critical points



The Golden Search

- The Golden Search (modified Fibonacci search) is analogous to root finding using bisection.
- In this case rather than knowing some interval [a,b] where f(a)f(b)<0 as in bisection, here we require that the function F(x) be unimodal over [a,b].
- This means that (if we are looking for a minimum):

$$F'(x) = \begin{cases} < 0 & x < x^* \\ > 0 & x > x^* \end{cases}$$

The Golden Search

- Note that F'' may change sign in this interval!
- If F'' = 0, then Newton's method (and successive parabolic interpolation) will fail, but the golden search is unaffected!
- An example of such a function:

$$F(x) = x^2 + a\cos((\omega x)^2)$$

provided $\omega^2 a < 1$ there is only one critical point at $x^* = 0$.

- For large x, however, F'' is both positive and negative.
- Unless Newton's method is started close to the minimum it will not converge!

The Golden Search

- The golden search relies on evaluating the function at points within the interval [a,b] and determining in which part of the sub-interval the minimum lies.
- Suppose we have the interval [0,1] over which the function is unimodal, and which contains the minimum.
- We evaluate the function at two additional points in the interior (1 – r), r such that $r > 0.5$.

Figure 8.5

Choosing a subinterval for a one-dimensional bracketing search method

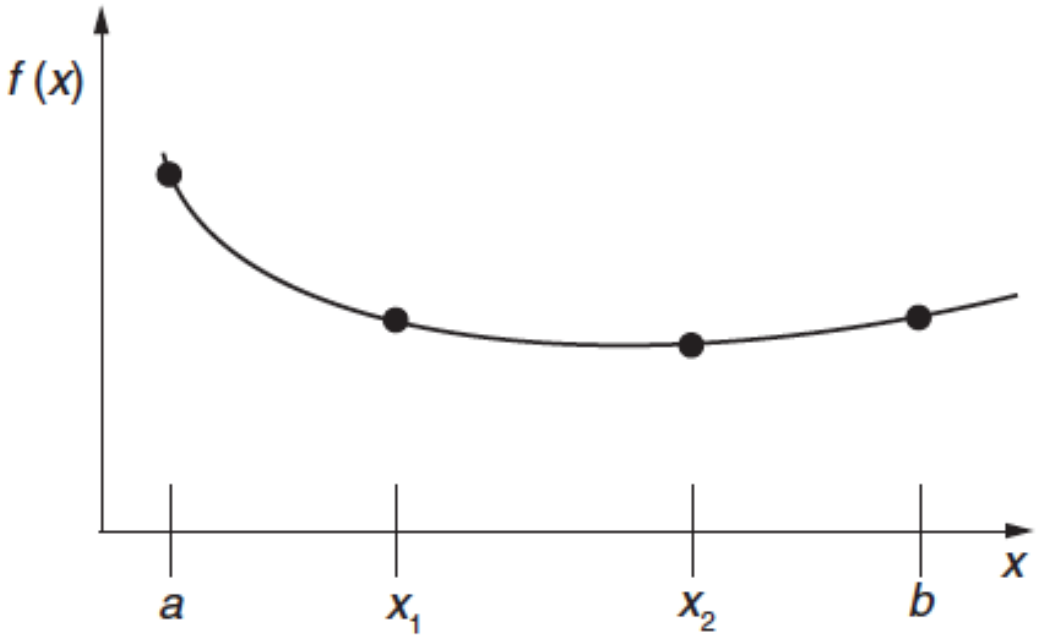
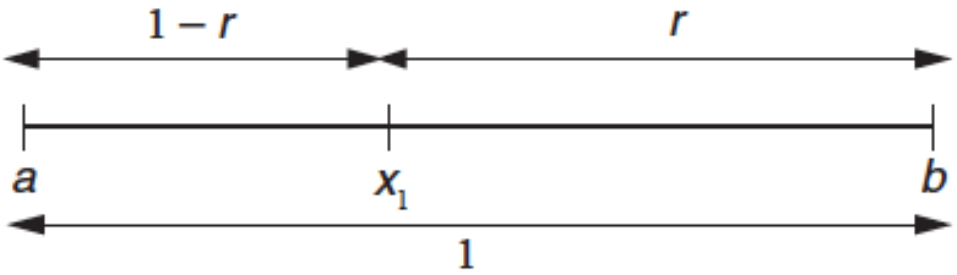


Figure 8.6

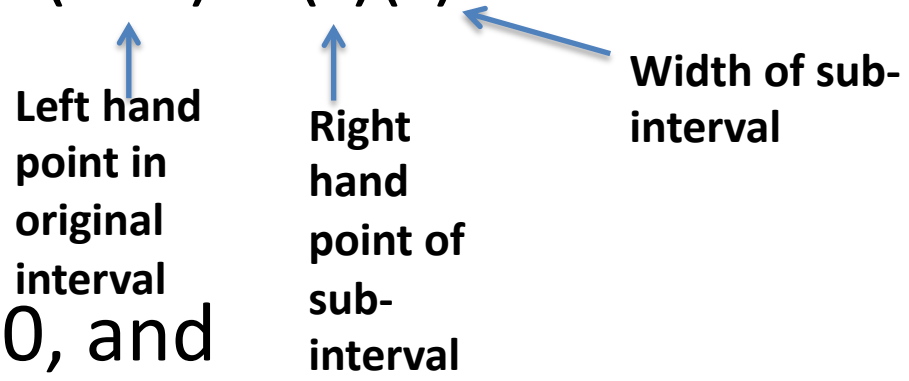
Dividing a line by the golden ratio



- Now if F(r) < F(1–r) then the minimum will lie in the interval [(1–r), 1] and if F(r) > F(1–r) then it will lie in the interval [0,r].
- We discard the balance of the original interval and do it again.
- The trick is to choose r so that in the next step we only need one new evaluation rather than two!

The Golden Search

- This means that we want the point (1–r) on the left side of the original interval to map onto the point r on the right.
- Thus we require: (1–r) = (r)(r)

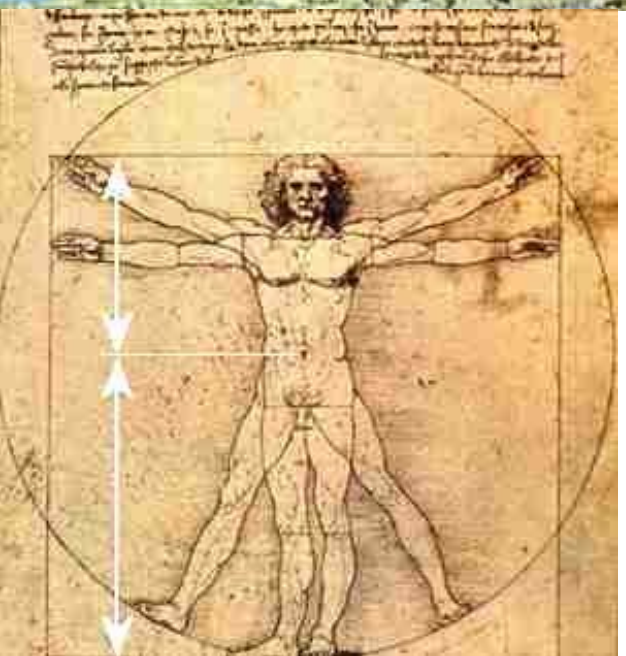


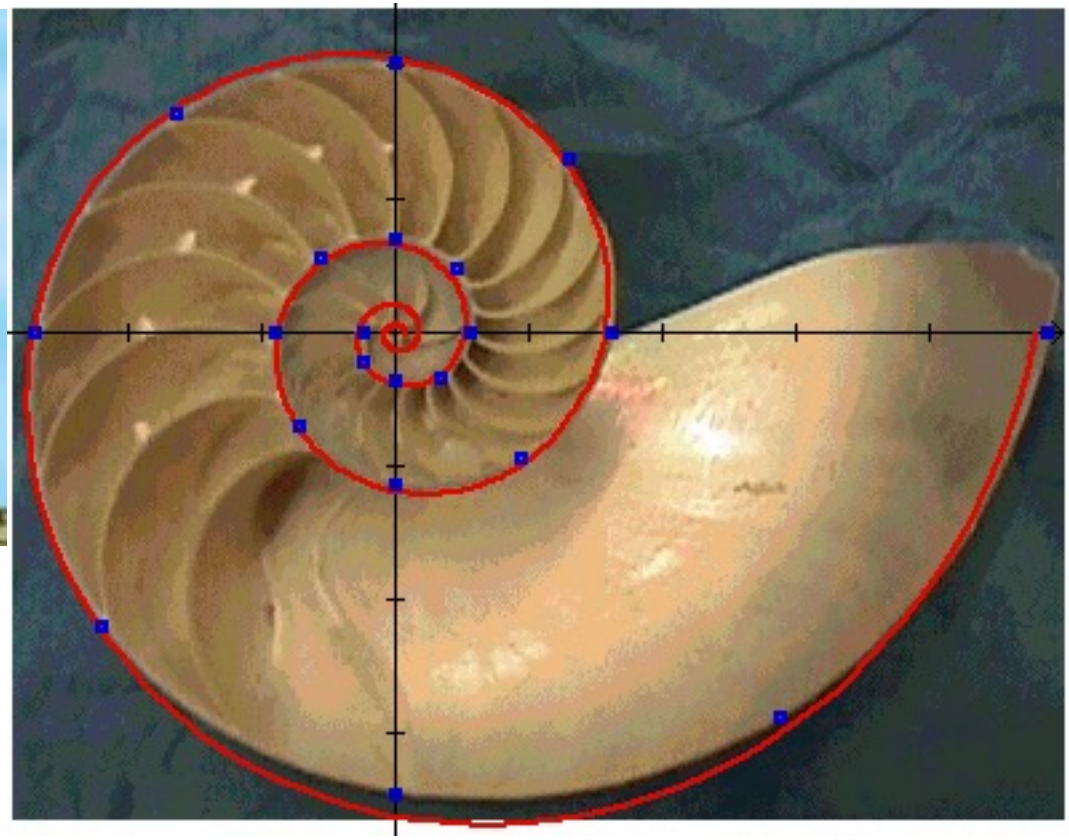
- Thus $r^2 + r - 1 = 0$, and

  r = (sqrt(5) – 1)/2 = 0.6180

Golden Ratio: 0.6180…

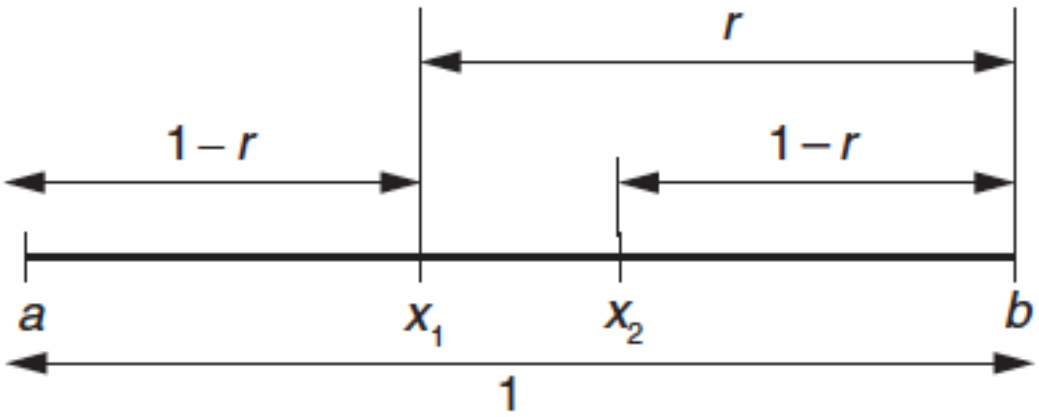
Parthenon in Greece



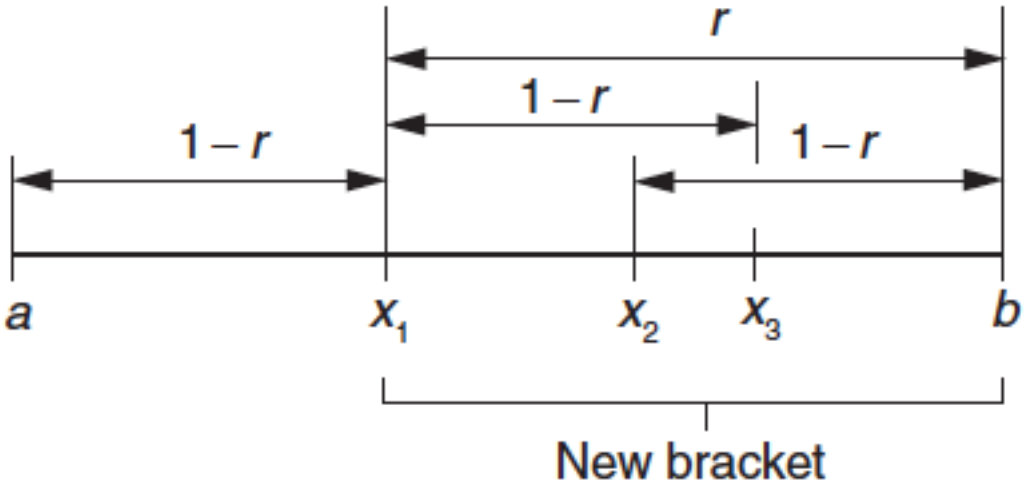


The Golden Search

First iteration of the golden section search method



Second iteration of the golden section search method



The Golden Search

- You get the same value of r if you discard the left side of the interval and map the old r.h.s. point onto the new left hand side point.
- Each iteration discards 38% of the current interval, thus:

$$|e_{i+1}|/|e_i| = 0.618$$

- So we have a linear rate of convergence, but c is higher than in bisection.

Q1: Matlab function bisectionmethod.m

What is the purpose of the highlighted line of the code?

```
% Iterative solution scheme
for i = 1:maxloops
    x = (a+b)/2; % mid-point of interval
    fx = feval(func,x);
    fprintf('%3d %5.4f %5.4f \n',i,x,fx);
    if (i>= minloops && abs(fx) < tolfx)
        break % Jump out of the for loop
    end
    if (fx*fa < 0) % [a x] contains root
        fb = fx; b = x;
    else           % [x b] contains root
        fa = fx; a = x;
    end
end
```

A. Calculate the midpoint of the interval.

B. Calculate the function evaluation at the midpoint.

C. Test whether f(a)*f(m) < 0

D. Choose the first half of the interval.

E. Choose the second half of the interval.

Q2: Matlab function newtonsmethod.m

What is the purpose of the highlighted line of the code?

```
% Iterative solution scheme
for i = 1:maxloops
    x1 = x0 - fx/fxderiv;
    [fx, fxderiv] = feval(func,x1);
    fprintf('%2d %5.4f %7.6f %7.6f \n',i,x1,fx,fxderiv);
    if (abs(x1 - x0) <=tolx && abs(fx) < tolfx)
        break % Jump out of the for loop
    end
    x0 = x1;
end
```

A. Calculate the next iteration using the Newtons Method formula.

B. Calculate the function evaluation at x_i.

C. Calculate the function derivative at x_i.

D. Display the iteration number, current guess, function and derivative to the screen.

E. Test whether the current accuracy is within the specified tolerance.

Q3: Matlab function secantmethod.m

What is the purpose of the highlighted line of the code?

```
% Iterative solution scheme
for i = 1:maxloops
    x2 = x1 - fx1*(x1 - x0)/(fx1 - fx0);
    fx2 = feval(func,x2);
    fprintf('%2d  %5.4f  %5.4f  %5.4f  %7.6f  %7.6f  \n', ...
      i,x0,x1,x2, fx0,fx1);
    if (abs(x2 - x1) <=tolx && abs(fx2) < tolfx)
        break % Jump out of the for loop
    end
    x0 = x1;
    x1 = x2;
    fx0 = fx1;
    fx1 = fx2;
end
```

A. Calculate the next iteration using the Secant Method formula.

B. Calculate the function evaluation at x_i.

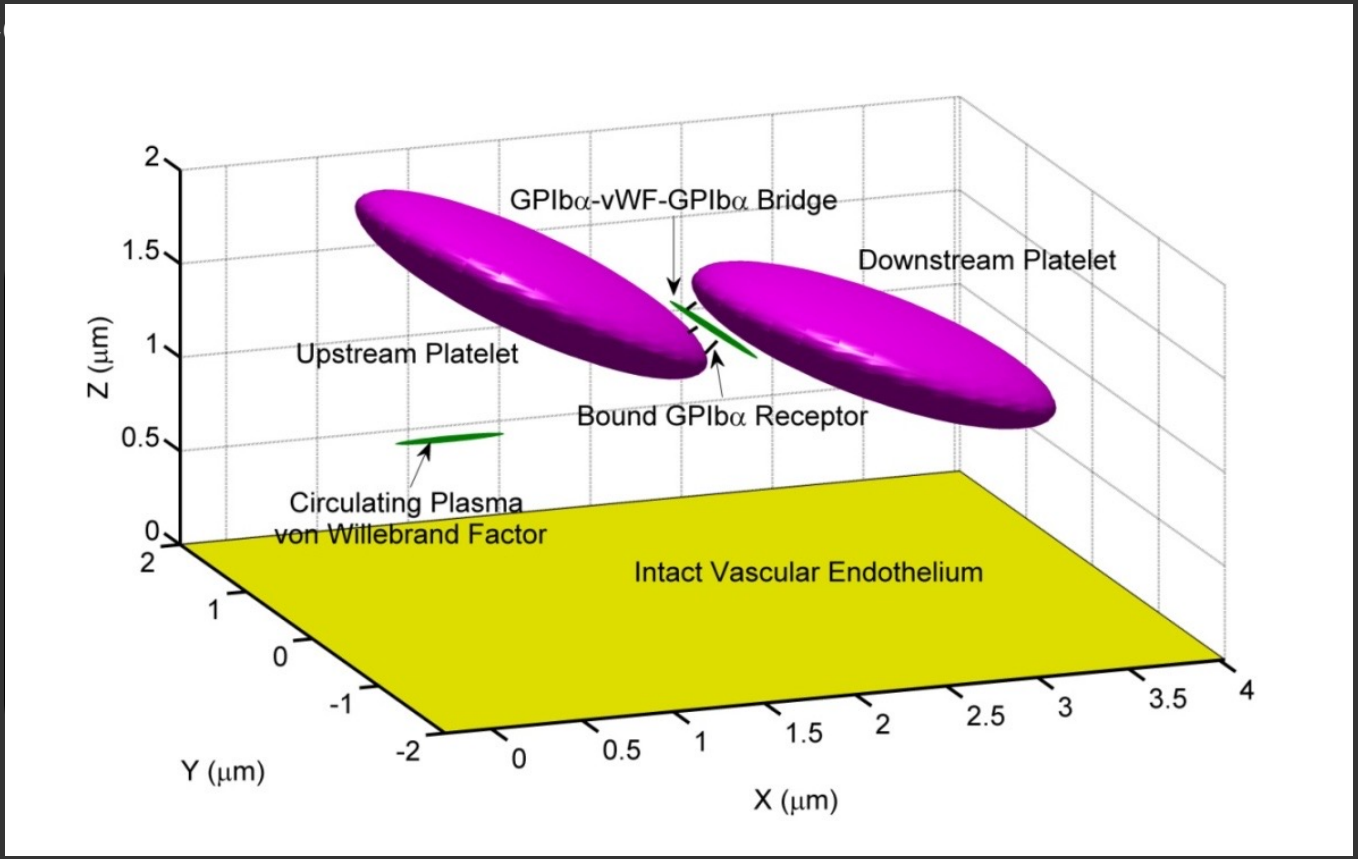
C. Test whether the current accuracy is within the specified tolerance.

D. Save the previous two guesses, as the current two guesses.

E. Save the previous two function evaluations, as the current ones.

Platelet Adhesive Dynamics Model for Platelet – Platelet Bridging

- GPIbα-vWF-GPIbα bonds treated as linear springs with two fixed end-points.
- C

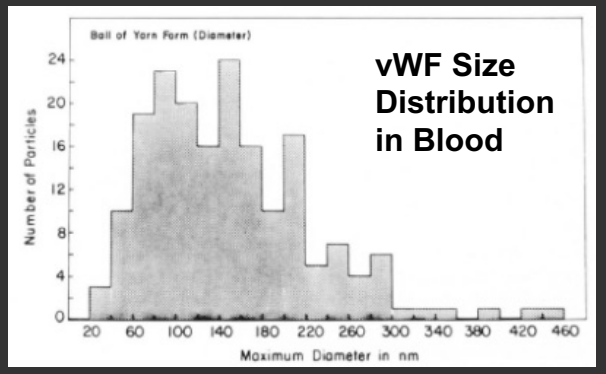


Mody and King, Part II, Biophys. J. 2008

Platelet Adhesive Dynamics (PAD) Model

Two **vWF sizes** modeled:

- n-vWF: 175 x 28 nm (18 A1 binding sites) (Singh et al., 2006; Siedlecki et al., 1996)
- L-vWF: 400 x 28 nm (34 A1 binding sites)



Slayter et al., JBC, 1985

Two **binding kinetics** tested

- Healthy or normal GPIbα-vWF-A1 binding kinetics
- Platelet-type Von Willebrand Disease (VWD): 5-fold higher affinity of A1 for GPIbα (Miura et al., 2000; Doggett et al., 2003).

$$k_{f,2-D} = k^0_{f,2-D} \exp\left(\sigma \left| \mathbf{x}_b - \lambda \right| \frac{\gamma - 0.5 \left| \mathbf{x}_b - \lambda \right|}{k_b T} \right) \text{(Bell et al., 1984)}$$

$$k_r = k_r^o \exp\left(\frac{\gamma F_b}{k_B T} \right); \mathbf{F}_b = \sigma(\mathbf{x}_b - \lambda) \text{(Bell, 1978)}$$

$$k_r^o = 5.47\,\text{s}^{-1}; \gamma = 0.71\,\text{nm (Arya et al., 2005)}$$

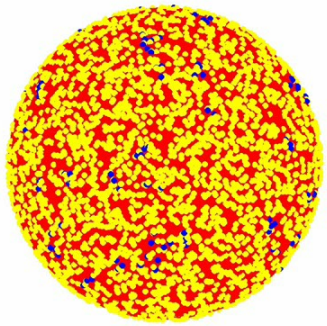
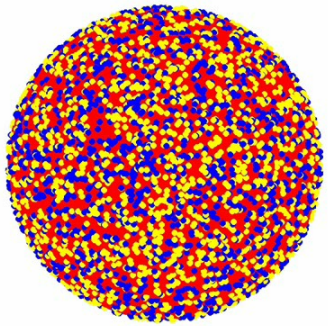
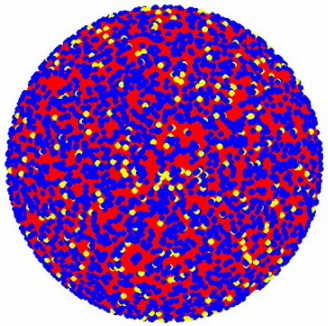
Cornell University

GPIbα-vWF Equilibrium Binding Governs Initial vWF Distribution on Platelet Surface

- Equivalent Site Hypothesis (ESH) Model for Multivalent Ligand/ Monovalent Receptor binding (Perelson, 1981)
- vWF-Platelet equilibrium binding behavior (K_D at 10,800 s^{-1}) governs simulation's initial conditions (Goto et al., 1995)
- Dissociation constant K_D determined to be 7.73×10^{-5} M

$$(1)\ C_{i,eq} = \left[\frac{f!}{i!(f-i)!}\right] K_x^{i-1} \frac{v}{f}\left(\frac{L_o}{K_D}\right) R_{eq}^i \quad (2)\ R_T = R_{eq}\left[1 + v\left(\frac{L_o}{K_D}\right)(1 + K_x R_{eq})^{f-1}\right]$$

●: free GPIbα receptors ●: GPIbα receptors bound to vWF

n-vWF	L-vWF	Platelet-type VWD
		

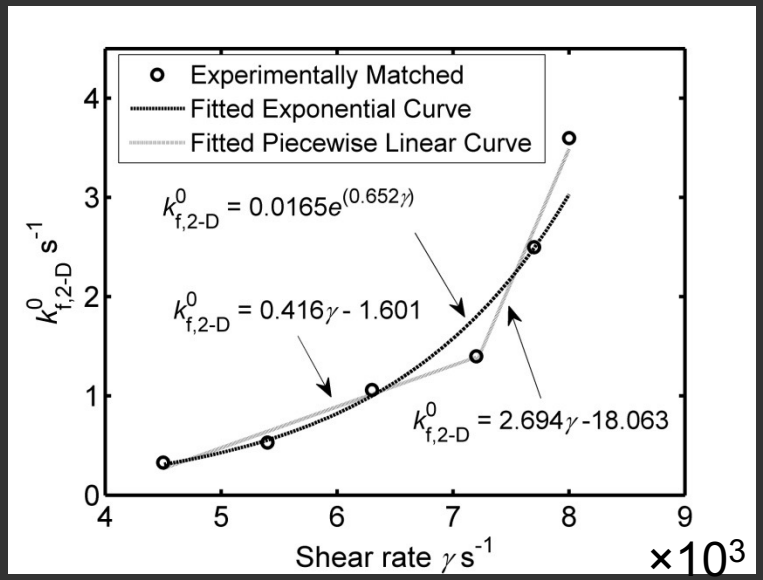
Cornell University

Simulated Platelet Binding Matched to Experiment: Estimation of $k^0_{f,2\text{-}D}$

Matching Simulation to Experiment

Shear Rate γ (s^{-1})	Binding Efficiency η_b (L-vWF) Derived from Experiment (Huang and Hellums, 1993; Konstantopoulos et al., 1997)	Binding Efficiency η_b (L-vWF) Predicted by Platelet Adhesive Dynamics Simulations	$k^0_{f,2\text{-}D}$ (s^{-1})
4500	0.041	0.041	0.33
5400	0.047	0.047	0.53
6300	0.091	0.090	1.06
7200	0.105	0.106	1.4
7700	0.191	0.184	2.5
8000	0.236	0.236	3.6

Exponential Dependence of $k^0_{f,2\text{-}D}$ on Fluid Shear Rate



Cornell University

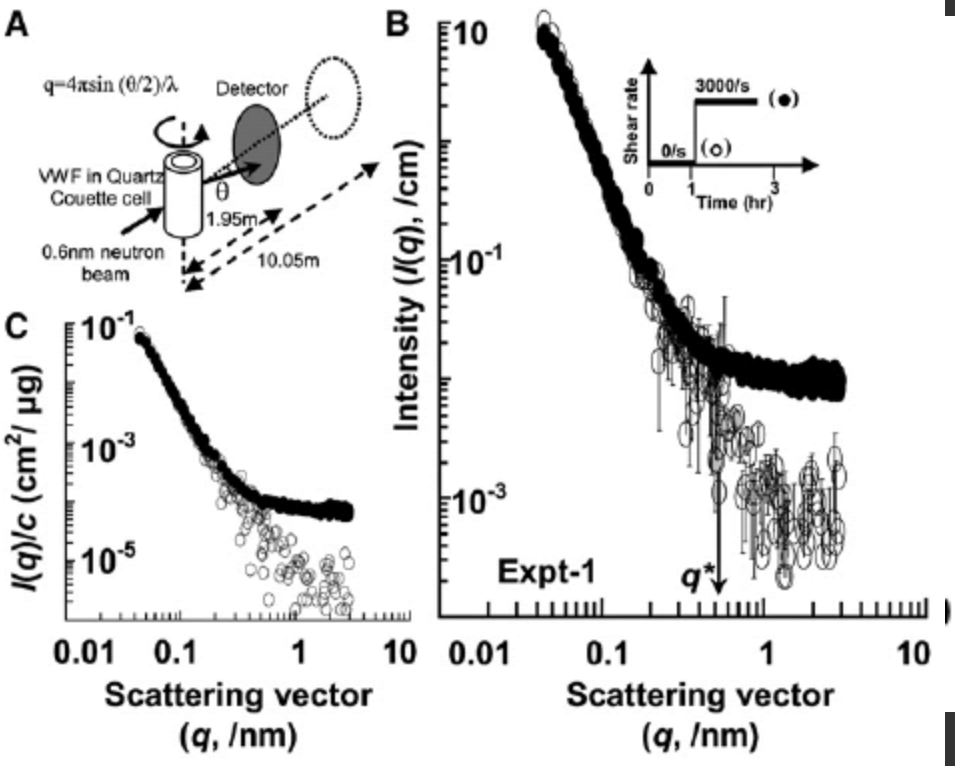
Independent measurement of VWF conformational change under shear

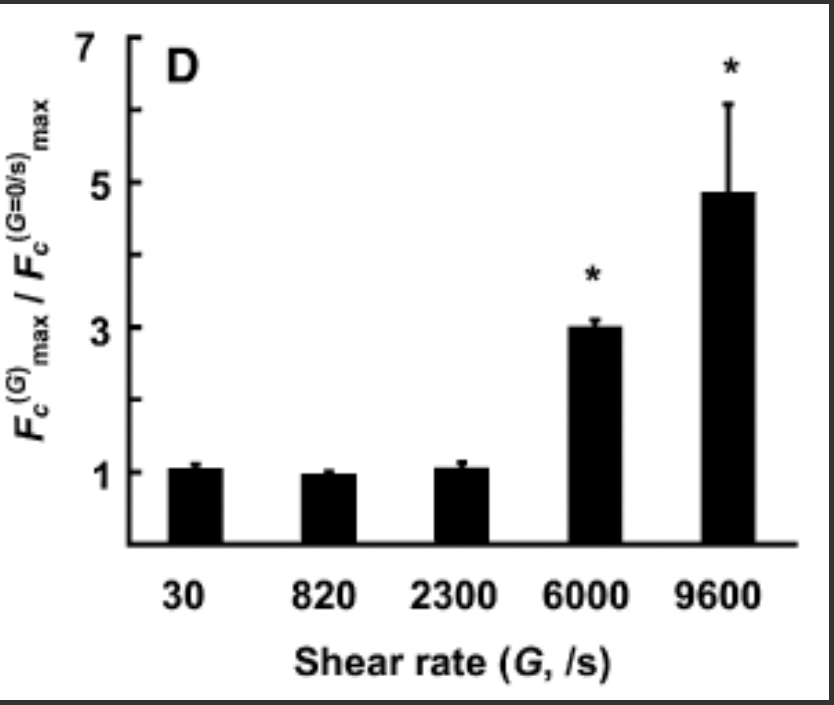
Biophysical Journal Volume 96 March 2009 2313–2320

Fluid Shear Induces Conformation Change in Human Blood Protein von Willebrand Factor in Solution

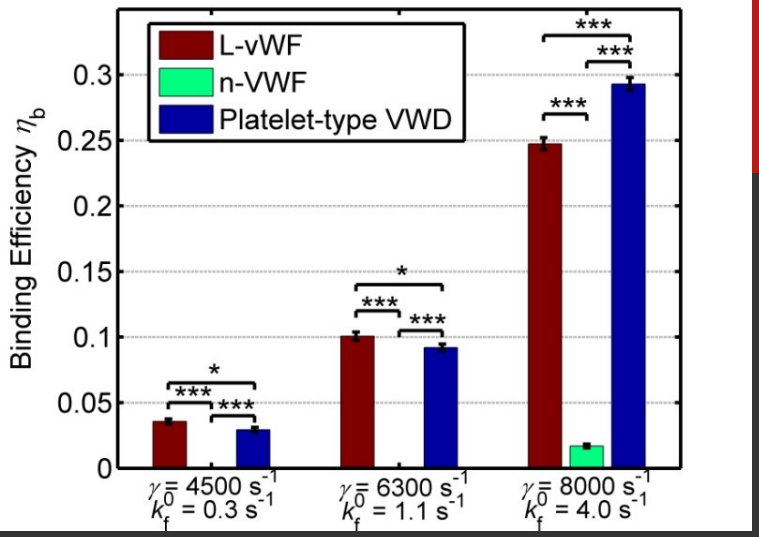
Indrajeet Singh,[†] Efrosyni Themistou,[†] Lionel Porcar,[‡] and Sriram Neelamegham[†*]

[†]Chemical and Biological Engineering, State University of New York, Buffalo, New York, and [‡]Center for Neutron Research, National Institute of Standards and Technology, Gaithersburg, Maryland



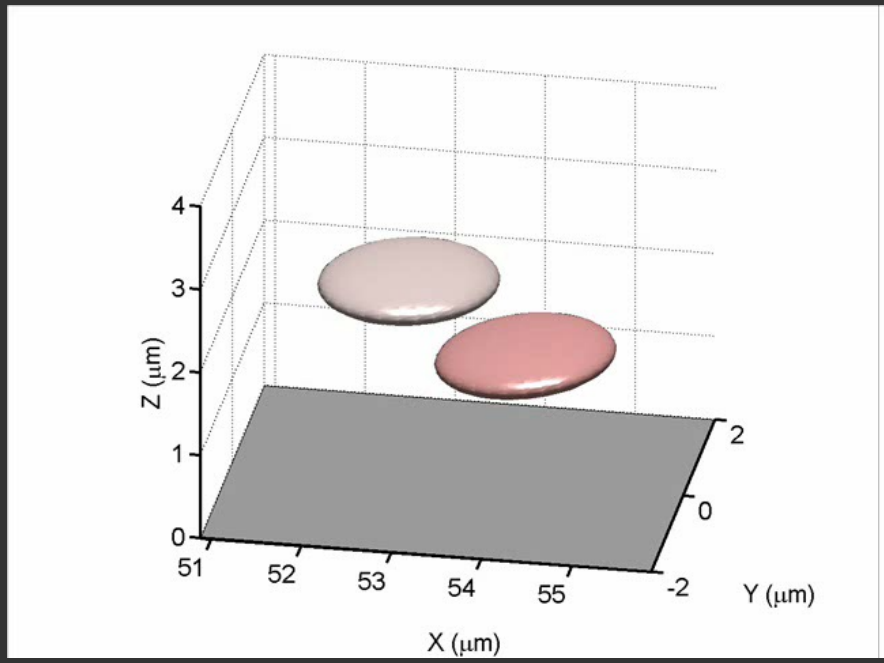


Platelet Aggregation is a Function of Ligand Size and Binding Kinetics

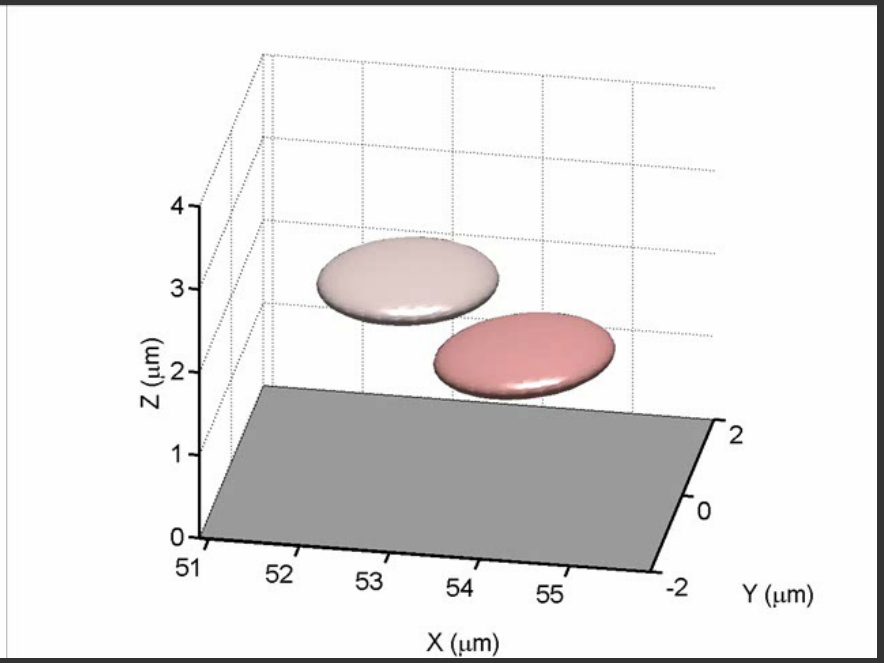


Shear rate = 8000 s^{-1}; $k^0_{f,2\text{-}D}$ (cross-linking bond formation rate constant)= 13 s^{-1}

L-vWF multimers initially bound to downstream platelet



n-vWF multimers initially bound to downstream platelet



Multidimensional Optimization

- Life get much more complex for higher (N>1) dimensional optimization.
- In general, we start from a given point, pick a search direction, and do a 1-D search in that direction.
- Method of steepest descent: If we want to get to a minimum, it makes sense to go downhill.

  Search direction = – grad(F)

Multidimensional Optimization

- Thus we solve the 1-D problem:

$$\min_{\alpha} F\{\mathbf{x}^{(k)} - \alpha\, \mathbf{grad}[F(\mathbf{x}^{(k)})]\}$$

to get:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_{opt}\, \mathbf{grad}\, F(\mathbf{x}^{(k)})$$

where α_{opt} is the desired minimum.

- This method tends to be rather slow. The gradient often does not point towards the minimum!

Multidimensional Optimization

- Let's work an example:

  Let $F(\mathbf{x}) = x_1^2 + 10x_2^2 + 100x_3^2$

- This has a global minimum of $\mathbf{x}^* = \mathbf{0}$

  $(x_1^* = x_2^* = x_3^* = 0)$

- Now grad $F = (2x_1, 20x_2, 200x_3)^T$

- Thus at each iteration we want to solve:

  $\min_\alpha F(\mathbf{x} - \alpha\ \mathbf{grad}\ F) =$

  $\min_\alpha \{(x_1 - 2\alpha x_1)^2 + 10(x_2 - 20\alpha x_2)^2 + 100(x_3 - 200\alpha x_3)^2\}$

Multidimensional Optimization

- We can actually get a linear equation for α for this particular problem.

  $\alpha = (x_1^2 + 10^2 x_2^2 + 10^4 x_3^2)/(2(x_1^2 + 10^3 x_2^2 + 10^6 x_3^2))$

- After 180 iterations we get:

  $x^{(0)} = (1,1,1)$ ← starting point

  $x^{(180)} = (1.07 \times 10^{-4}, 0, 2.01 \times 10^{-6})$

- Why? The problem has different curvature in different directions.
- You can think of the algorithm as wandering back and forth across a narrow river valley and slowly rolling down to the sea.