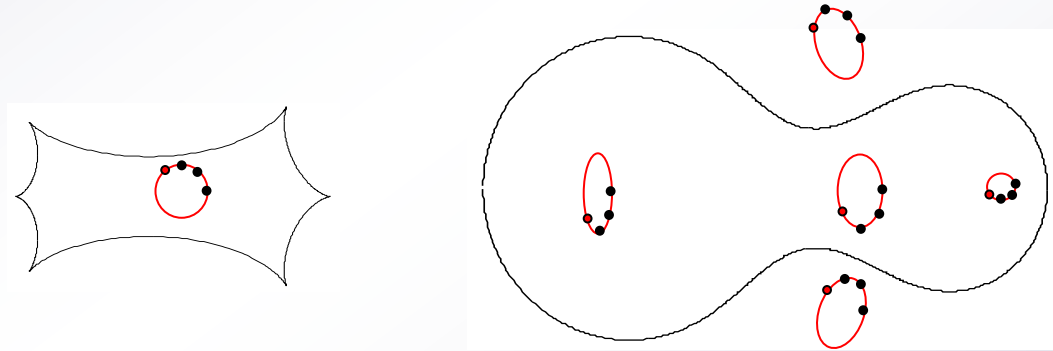


Contour Integration and Downhill Fitting



Valerio Bozza
University of Salerno

Summary

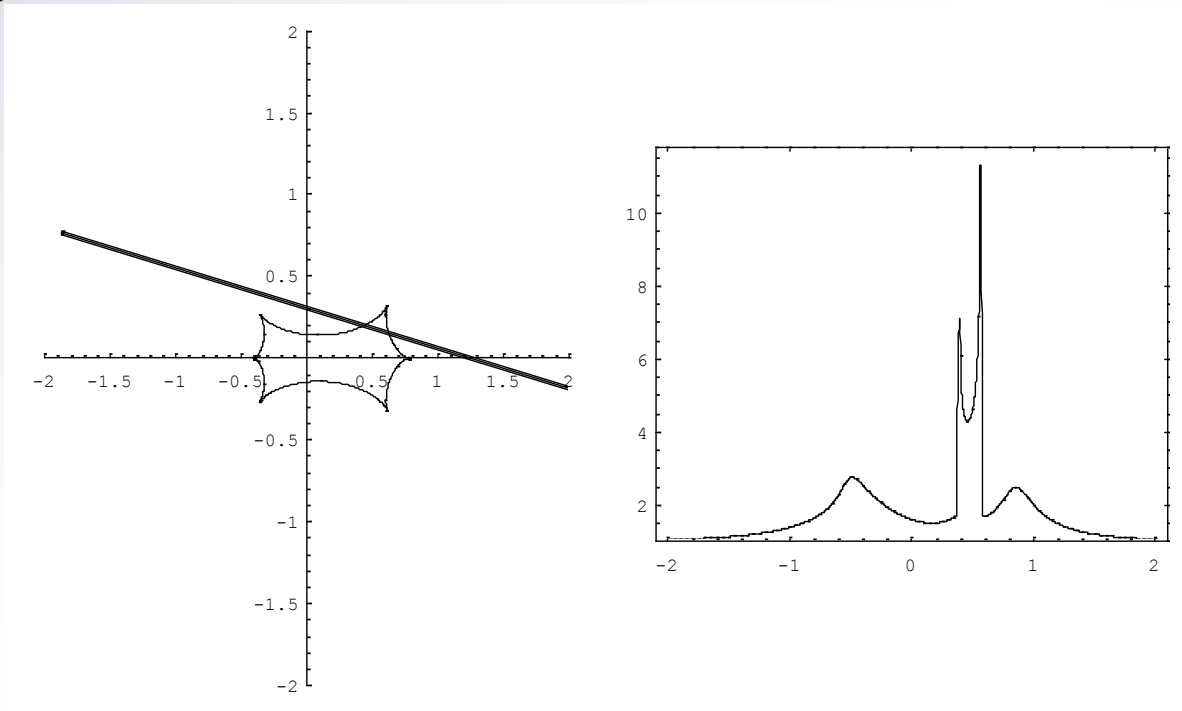
- Contour Integration Concept
 - Green's Theorem
 - History
- Basic Contour Integration
 - Inversion of the lens equation
 - Reconstruction of image boundaries
 - Polygonal approximation
- Advanced Contour Integration
 - Parabolic correction
 - Error control
 - Optimal sampling
 - Limb darkening
- Downhill fitting
 - Levenberg-Marquardt
 - Jumping out of minima

1. Contour Integration Concept

1. Contour Integration Concept

Our goal

- Our purpose is to calculate microlensing light curves



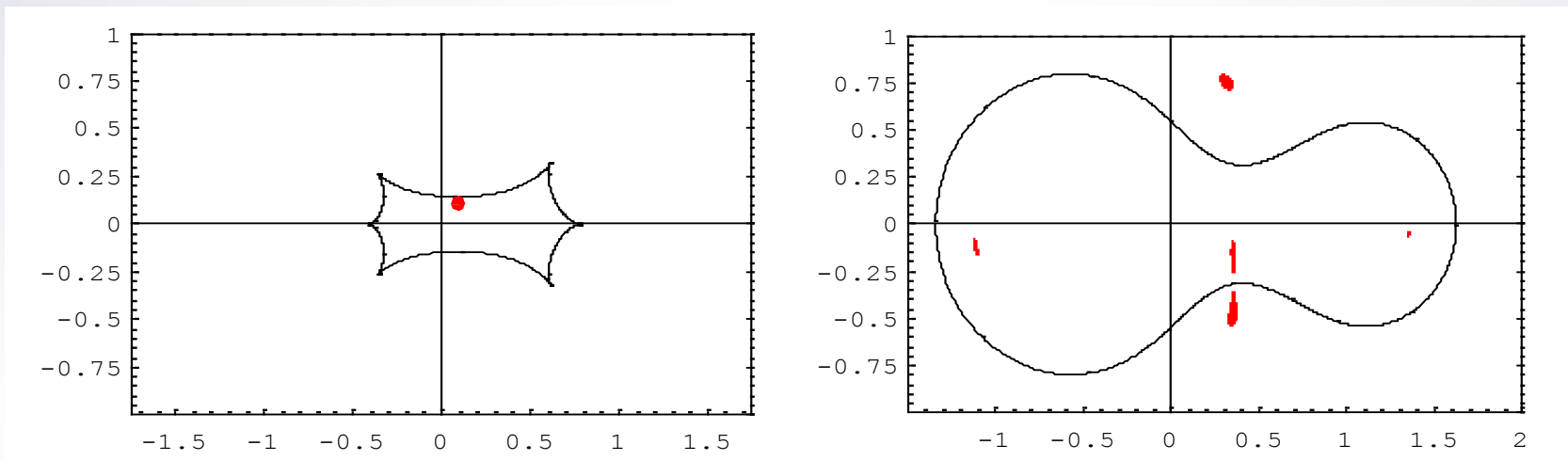
- ... that is the magnification of a given source that transits behind a given binary lens model.

1. Contour Integration Concept

Our goal

- For a given source position and size and for a given lens model, we want to calculate the magnification factor

$$\mu = \frac{\sum A_I}{A_S}$$

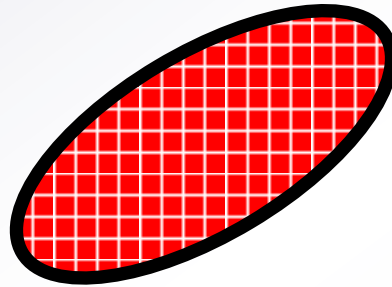


- We must find the images and calculate their area.

1. Contour Integration Concept

Contour Integration Concept

- Contour integration concept:
The area enclosed in a curve is expressed by a simple contour integral on the boundary.



- We only need to find the boundaries of the images
- A surface integral becomes a one-dimensional integral
- In principle this is much faster and very elegant
- In practice, much more work is required to keep everything under control.

1. Contour Integration Concept

Green's Theorem

- Green's theorem:

$$\oint_{\partial I} (L_1 dx_1 + L_2 dx_2) = \iint_I dx_1 dx_2 \left(\frac{\partial L_2}{\partial x_1} - \frac{\partial L_1}{\partial x_2} \right)$$

- ... can be viewed as the two-dimensional version of Stokes' theorem

$$\oint_{\partial I} \mathbf{L} \cdot d\mathbf{x} = \iint_I dS (\nabla \wedge \mathbf{L}) \cdot \mathbf{n}$$

- Note: ∂I is the counterclockwise boundary of I .

1. Contour Integration Concept

Green's Theorem

- If we want the area of the domain I we must choose

$$\left(\frac{\partial L_2}{\partial x_1} - \frac{\partial L_1}{\partial x_2} \right) = 1$$

- Possible choices for (L_1, L_2) are $(-x_2, 0)$, $(0, x_1)$, $(-x_2, x_1)/2$.
- Then the line integral takes the equivalent forms

$$A = -\oint_{\partial I} x_2 dx_1 = \oint_{\partial I} x_1 dx_2 = \frac{1}{2} \oint_{\partial I} \mathbf{x} \wedge d\mathbf{x}$$

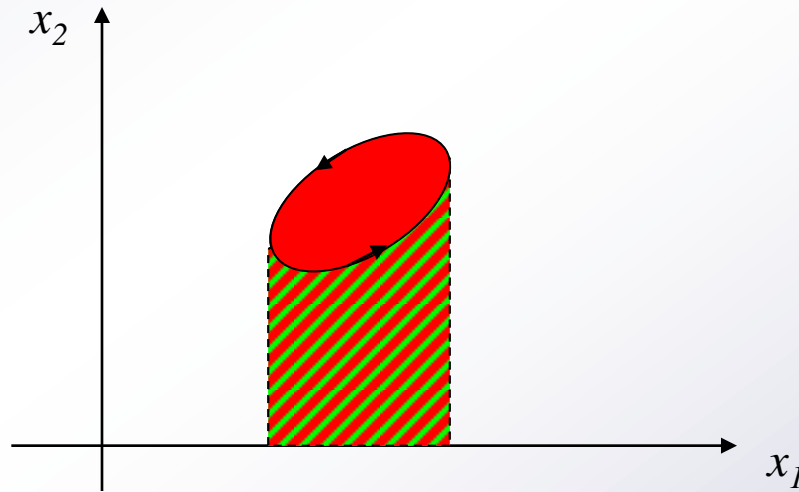
- Note: $\mathbf{x} \wedge d\mathbf{x}$ is a pseudo-scalar in two dimensions.

1. Contour Integration Concept

Green's Theorem

- Green's theorem generalizes the interpretation of Riemann's integral as the area under the graph of a function.

$$A = -\oint_{\partial I} x_2 dx_1$$



Implementation scheme

- Green's theorem allows us to calculate the area of a microlensing image by a contour integral on the image boundary.
- Implementation:
 - a) Sample the source boundary
 - b) Invert the lens equation to find the image boundaries
 - c) Re-order the points in the sample
 - d) Approximate the contour integral using the sample

History

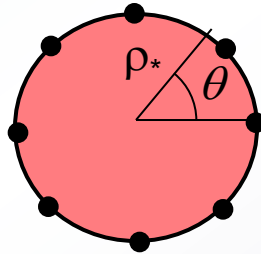
- 1987: Schramm & Kayser, A&A 174, 361
Use of contours in cosmological gravitational lensing
- 1993: Dominik, Diploma thesis
1995: Dominik, A&AS 109, 507
Calculation of the areas by Green's theorem
- 1997: Gould & Gaucherel, ApJ 477, 580
1998: Dominik, A&A 333, L79
Application to microlensing
- 2006: Dong et al., ApJ 642, 842
Hybridization with inverse ray-shooting
- 2007: Dominik, MNRAS 377, 1679
Adaptive grid search
- 2010: Bozza, MNRAS 408, 2188
Advanced contour integration

2. Basic Contour Integration

Source boundary

- Parameterization of the source boundary:

$$\mathbf{y} = \mathbf{y}_0 + \rho_* \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} \iff \zeta = \zeta_0 + \rho_* e^{i\theta}$$



- Generic sampling for θ :

$$\{\theta_0, \theta_1, \dots, \theta_n\} \text{ with } 0 = \theta_0 < \theta_1 < \dots < \theta_n = 2\pi$$

- Sampling for the source boundary:

$$\zeta_i = \zeta_0 + \rho_* e^{i\theta_i}$$

2. Basic Contour Integration

Inversion of the lens equation

- For each point in the source boundary, we must solve the lens equation

$$\zeta = z - \frac{1}{1+q} \left[\frac{1}{\bar{z} + a/2} + \frac{q}{\bar{z} - a/2} \right]$$

- \bar{z} can be eliminated using the conjugate equation.
- We end up with a fifth order polynomial equation

$$\sum_{i=0}^5 c_i z^i = 0$$

- The coefficients c_i are functions of the separation a , the mass ratio q and the source position ζ .

Inversion of the lens equation

- A fifth order polynomial equation can be solved numerically by several methods (e.g. Laguerre's method is implemented in "Numerical Recipes" by Press et al.).

zroots

laguer

- As a result, for each point \mathbf{y}_i in the source boundary we have 5 points $\mathbf{x}_{i,l}$ in the lens plane.

- These 5 roots must be validated by the original lens equation

$$\zeta = z - \frac{1}{1+q} \left[\frac{1}{\bar{z} + a/2} + \frac{q}{\bar{z} - a/2} \right]$$

- If \mathbf{y}_i is inside a caustic, all 5 roots will be acceptable, otherwise, two roots will be rejected.

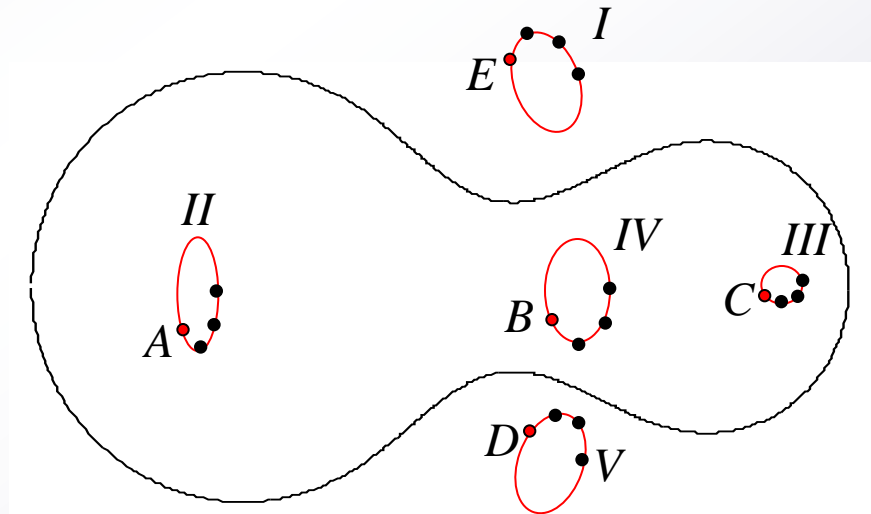
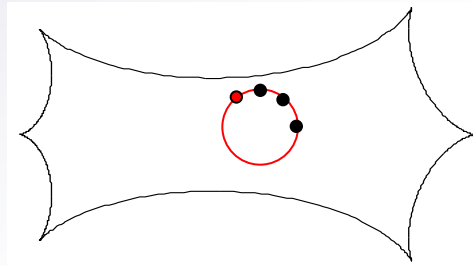
Inversion of the lens equation

- Numerical errors may arise here if the lens parameters are very extreme (e.g. for very low q , and/or $\zeta \approx 0$ or very large).
- Several checks can be performed (consistency of parities, lens equation residual large for more than two images, ...)
- The root finding routine is the slowest step in the whole contour integration method.
- The purpose of all optimizations will be to increase the accuracy with fewer points in the boundaries.

2. Basic Contour Integration

Reconstruction of image boundaries

- The roots $\mathbf{x}_{i,I}$ lie somewhere on the image boundaries. But where?
- We need to associate the roots $\mathbf{x}_{i,I}$ at step i with the roots $\mathbf{x}_{i-1,I}$ of the step $i-1$.

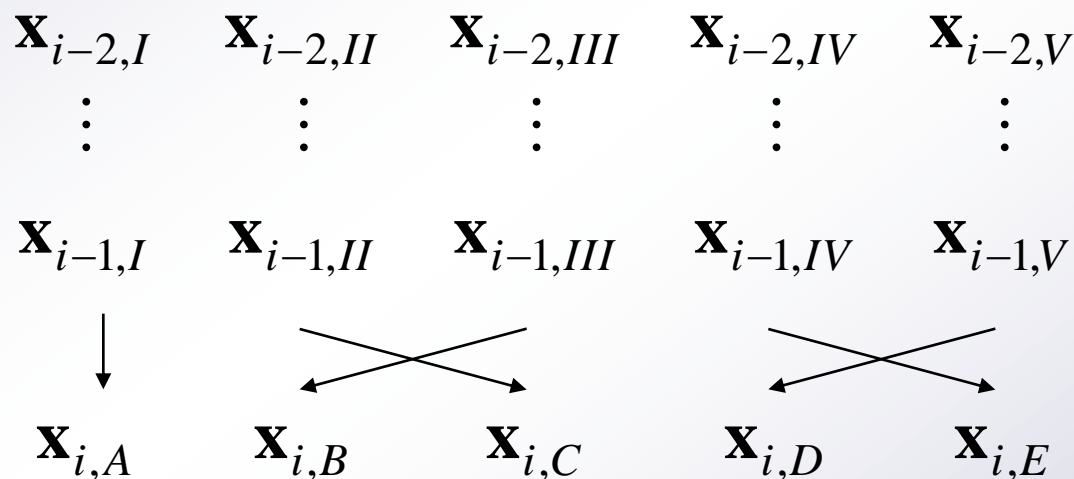


- The simplest way is to use the least distance criterium.
- Only same parity solutions can be associated.

2. Basic Contour Integration

Reconstruction of image boundaries

- The roots $\mathbf{x}_{i,I}$ lie somewhere on the image boundaries. But where?
- We need to associate the roots $\mathbf{x}_{i,I}$ at step i with the roots $\mathbf{x}_{i-1,I}$ of the step $i - 1$.

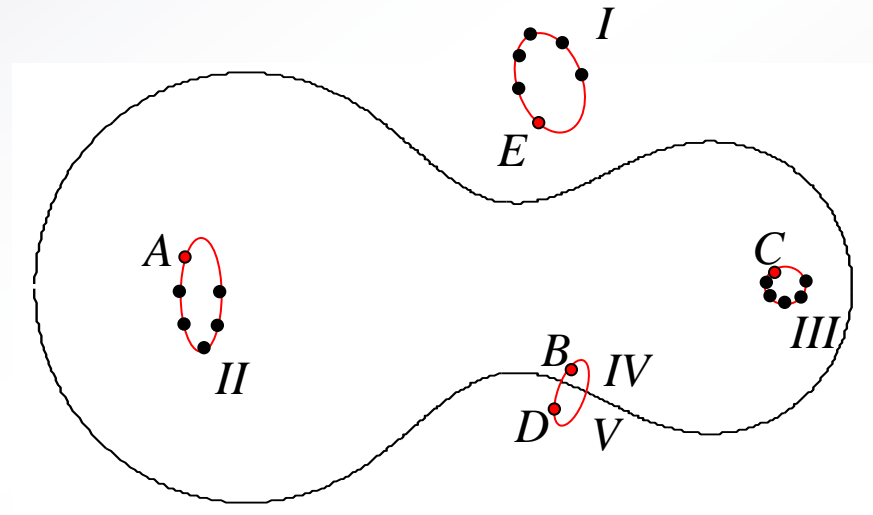
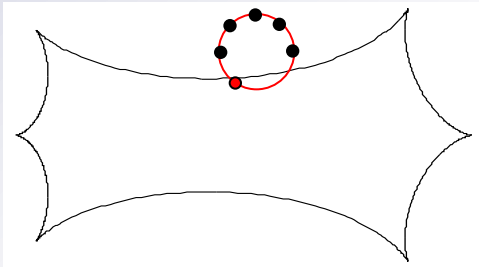


- The simplest way is to use the least distance criterium.
- Only same parity solutions can be associated.

2. Basic Contour Integration

Reconstruction of image boundaries

- If two new images are created at step i , we can recognize them as the last two unmatched roots.

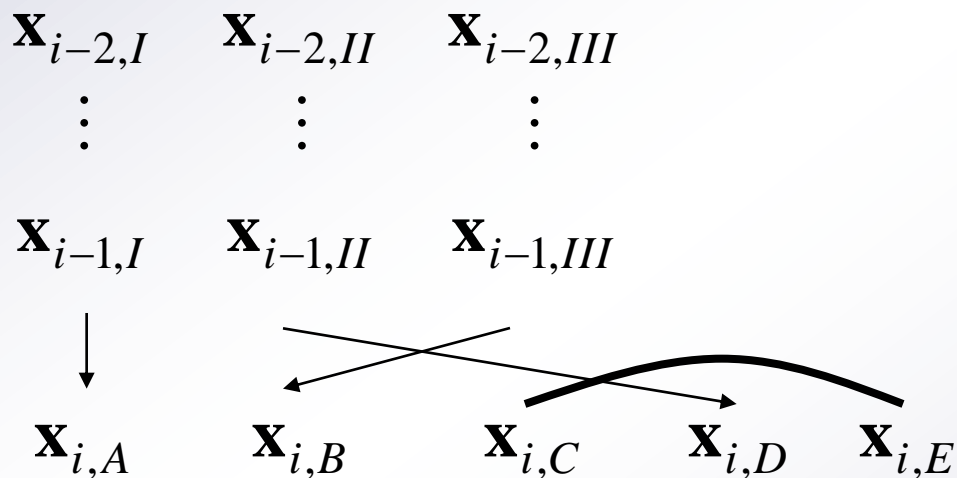


- The same can be done at destruction of two images.
- We must keep track of pairing between image boundaries when they are created or destroyed (see next).

2. Basic Contour Integration

Reconstruction of image boundaries

- If two new images are created at step i , we can recognize them as the last two unmatched roots.



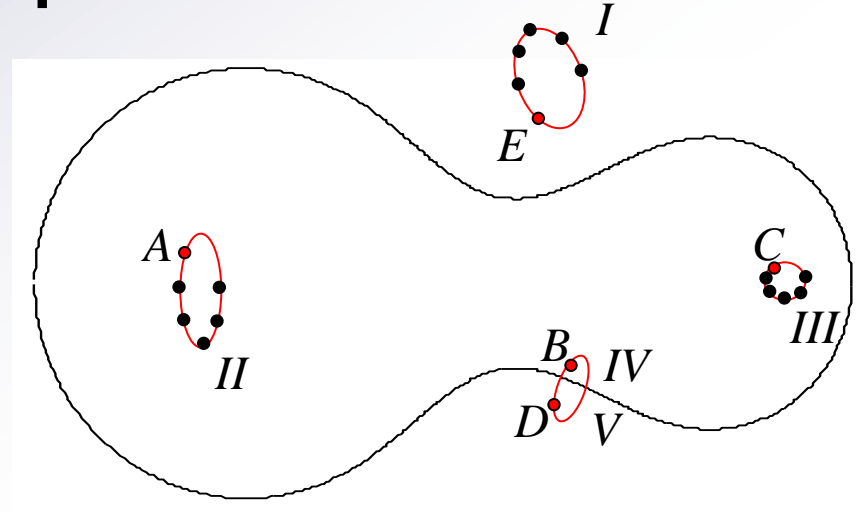
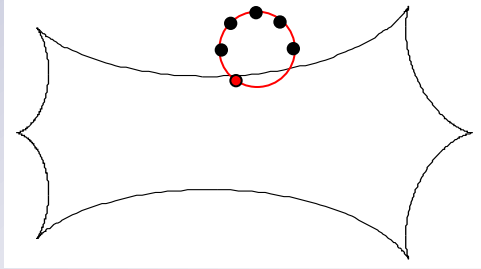
- The same can be done at destruction of two images.
- We must keep track of pairing between image boundaries when they are created or destroyed (see next).

Reconstruction of image boundaries

- There remains a possibility of wrong associations for very stretched images and poor sampling. Error control will take care of this.
- We end up with a collection of image boundaries, made up of a sequence of points.
- Each boundary has a definite parity.
- Some boundaries are complete (having n points as in the source boundary sample).
- Some other boundaries start with a creation episode; some others end with a destruction episode; some start with a creation and end with a destruction.

2. Basic Contour Integration

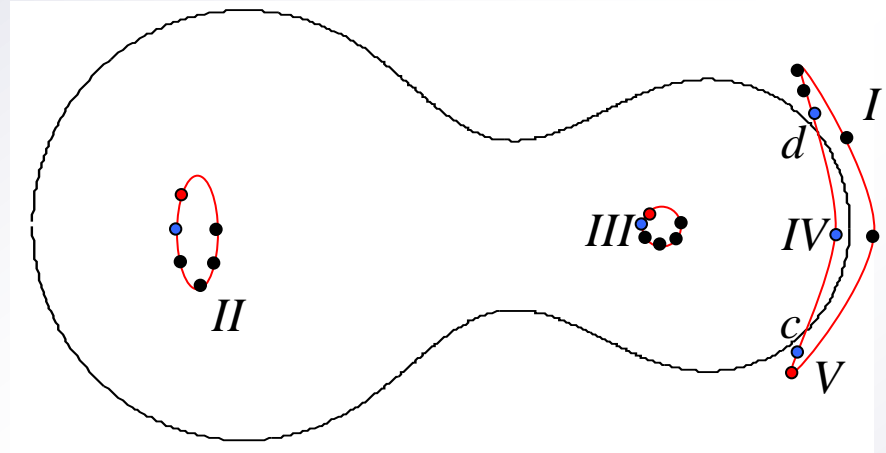
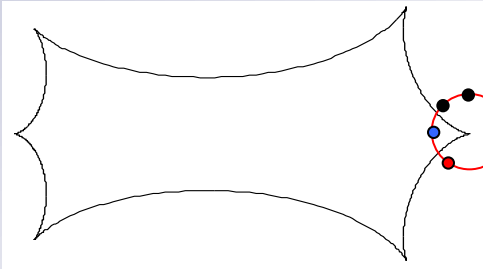
Some examples of boundaries



- Boundaries *I*, *II*, *III* start at $\theta=0$ and end at $\theta=2\pi$
- Boundaries *IV* and *V* start at some θ_c and end at θ_d

2. Basic Contour Integration

Some examples of boundaries



- Boundaries *II*, *III* start at $\theta=0$ and end at $\theta=2\pi$
- Boundary *IV* starts at θ_c and ends at θ_d
- Boundary *I* start at $\theta=0$ and ends at θ_d
- Boundary *V* starts at θ_c and ends at $\theta=2\pi$

2. Basic Contour Integration

Contour integration by polygonal

- As in the trapezium approximation of the Riemann integral, we can approximate the different versions of the contour integral as follows

$$A = -\oint_{\partial I} x_2 dx_1 \cong -\sum_{i=0}^{n-1} \frac{x_{i+1,2} + x_{i,2}}{2} (x_{i+1,1} - x_{i,1})$$

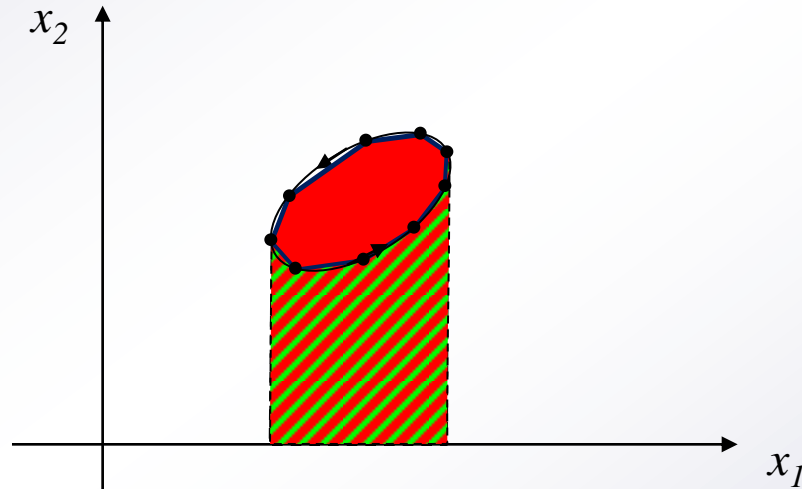
$$A = \oint_{\partial I} x_1 dx_2 \cong \sum_{i=0}^{n-1} \frac{x_{i+1,1} + x_{i,1}}{2} (x_{i+1,2} - x_{i,2})$$

$$A = \frac{1}{2} \oint_{\partial I} \mathbf{x} \wedge d\mathbf{x} \cong \frac{1}{4} \sum_{i=0}^{n-1} (\mathbf{x}_{i+1} + \mathbf{x}_i) \wedge (\mathbf{x}_{i+1} - \mathbf{x}_i) = \frac{1}{2} \sum_{i=0}^{n-1} \mathbf{x}_i \wedge \mathbf{x}_{i+1}$$

2. Basic Contour Integration

Contour integration by polygonal

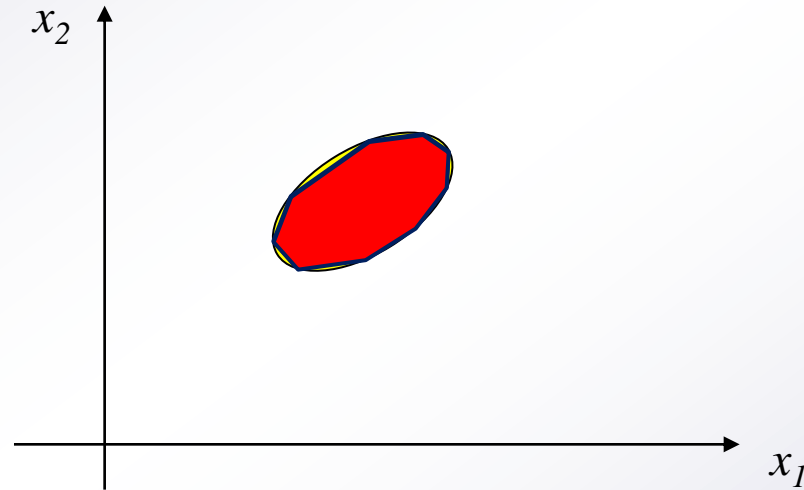
- The trapezium approximation gives the area of the polygonal defined by our image boundary sample



2. Basic Contour Integration

Contour integration by polygonal

- The trapezium approximation gives the area of the polygonal defined by our image boundary sample

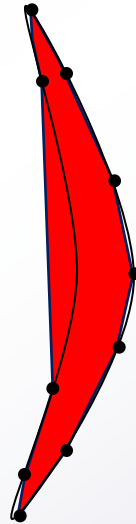


- Typically, the area is underestimated.

2. Basic Contour Integration

Contour integration by polygonal

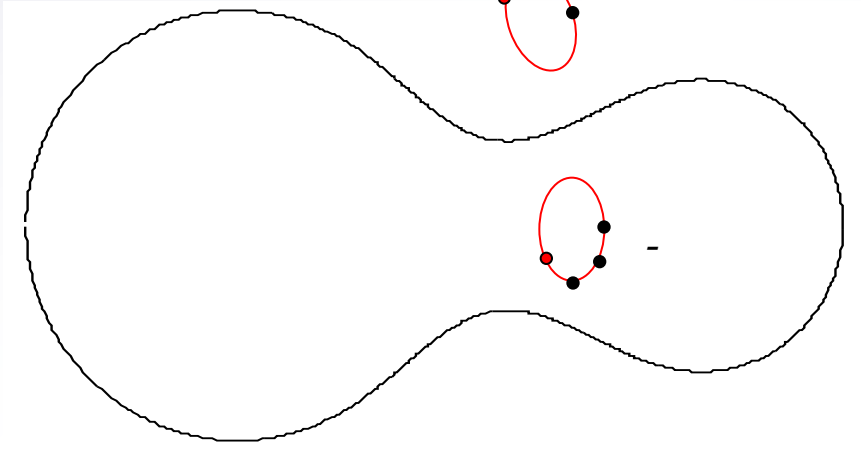
- The residual is positive for convex boundaries.
- It can be negative for boundaries with a concavity.



2. Basic Contour Integration

Contour integration by polygonal

- The contour integral assumes a counterclockwise orientation for the boundaries.
- The source parameterization we used is counterclockwise.
- Positive parity image boundaries preserve the orientation, negative parity boundaries reverse the orientation.
- Therefore, we must multiply the contour integrals by the parities of the boundaries:



$$A_I = \frac{1}{2} p_I \sum_{i=0}^{n-1} \mathbf{x}_i \wedge \mathbf{x}_{i+1}$$

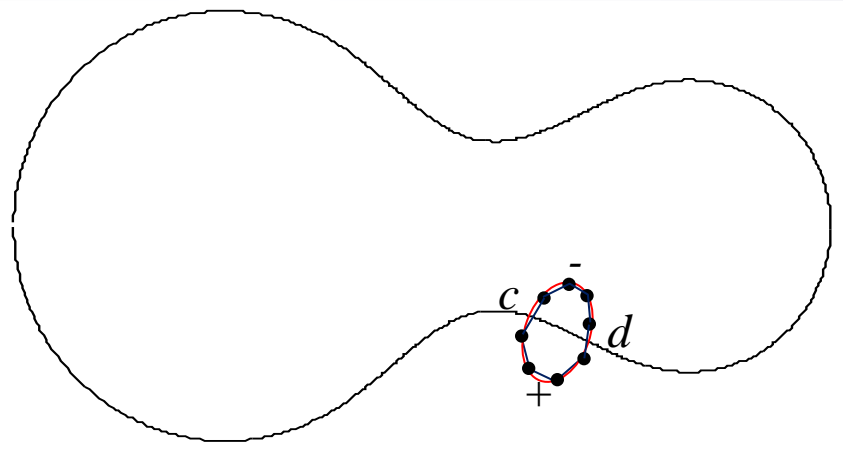
2. Basic Contour Integration

Contour integration by polygonal

- Boundaries starting with a creation or ending in a destruction are treated in the same way.
- We only need to add a connection term between the image pairs.

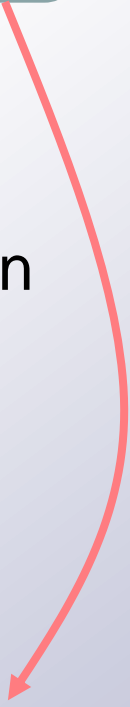
$$\frac{1}{2} \left(\mathbf{x}_{\text{first},-} \wedge \mathbf{x}_{\text{first},+} \right) \quad \text{for creation}$$

$$\frac{1}{2} \left(\mathbf{x}_{\text{last},+} \wedge \mathbf{x}_{\text{last},-} \right) \quad \text{for destruction}$$



Summing up...

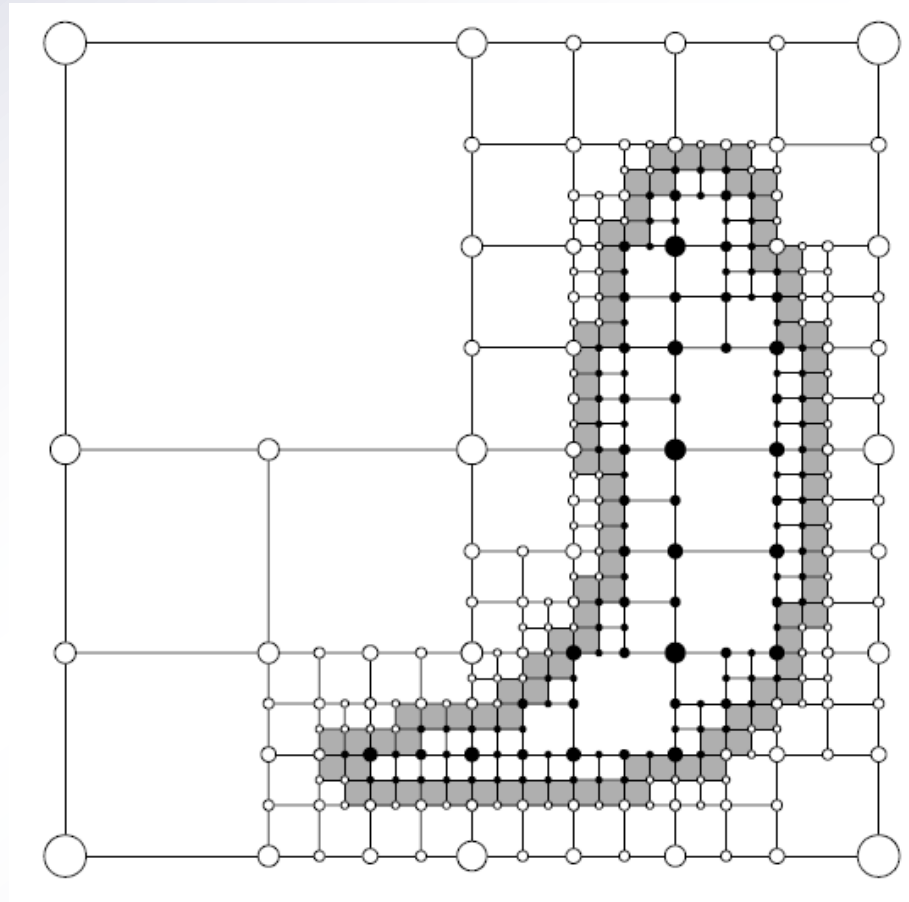
To set up your contour integration routine you must:

- Run a root finder routine for each point in the source boundary.
 - At each step you must put the roots in the correct image boundary (least distance criterium) and keep track of created and destroyed pairs.
 - Calculate the contour integral by polygonal approximation for each boundary.
 - Sum up the contour integrals with the correct parity and add a connection term for each creation/destruction.
- 
- Adaptive grid search is an alternative to root finding.

2. Basic Contour Integration

Adaptive grid search

Start with a big square and then divide it in smaller and smaller squares until you isolate the image boundary.



3. Advanced Contour Integration

3. Advanced Contour Integration

Order of the error

Let us estimate the order of the error

- At each step, the contribution of the interval $\Delta\theta$ to the contour integral is

$$\Delta A_I = \frac{1}{2} \int_{I(\Delta\theta)} \mathbf{x}_I \wedge d\mathbf{x}_I = \frac{1}{2} \int_{\theta_i}^{\theta_i + \Delta\theta} \mathbf{x}_I \wedge \mathbf{x}'_I d\theta$$

- The trapezium approximation is actually

$$\Delta A_I^{(t)} = \frac{1}{2} \mathbf{x}_I(\theta_i) \wedge \mathbf{x}_I(\theta_i + \Delta\theta)$$

- Expanding in powers of $\Delta\theta$, the difference is of third order

$$\Delta A_I^{(t)} - \Delta A_I = O(\Delta\theta^3)$$

3. Advanced Contour Integration

Parabolic correction

Why should we go to higher orders?

- To get an accurate estimate with less points in the sample.
- As the slowest routine is the inversion of the lens equation, by reducing the number of points, we can speed up the code.
- If we add the following correction to the trapezium

$$\Delta A_I^{(p)} = \frac{1}{24} \left[(\mathbf{x}'_I \wedge \mathbf{x}''_I) \Big|_{\theta_i} + (\mathbf{x}'_I \wedge \mathbf{x}''_I) \Big|_{\theta_i + \Delta\theta} \right] \Delta\theta^3$$

... the residual is of fifth order

$$\Delta A_I^{(t)} + \Delta A_I^{(p)} - \Delta A_I = O(\Delta\theta^5)$$

3. Advanced Contour Integration

Parabolic correction

How do we calculate the wedge product $(\mathbf{x}'_I \wedge \mathbf{x}''_I)|_{\theta_i}$?

- Switching to complex notations, the derivatives can be computed in terms of the lens mapping and the source parameterization.
- We have

$$\mathbf{x}' \wedge \mathbf{x}'' = \frac{1}{2i} (z''\bar{z}' - z'\bar{z}'') = \left\{ \rho_*^2 + \text{Im} \left[(z')^2 \zeta' \frac{\partial^2 \bar{\zeta}}{\partial z^2} \right] \right\} J^{-1}$$

$$\frac{\partial \bar{\zeta}}{\partial z} = \frac{1}{1+q} \left[\frac{1}{(z+a/2)^2} + \frac{q}{(z-a/2)^2} \right]$$

$$J = 1 - \left| \frac{\partial \bar{\zeta}}{\partial z} \right|^2$$

$$\frac{\partial^2 \bar{\zeta}}{\partial z^2} = -\frac{2}{1+q} \left[\frac{1}{(z+a/2)^3} + \frac{q}{(z-a/2)^3} \right]$$

$$z' = \left(\zeta' - \frac{\partial \zeta}{\partial \bar{z}} \bar{\zeta}' \right) J^{-1}; \quad \zeta' = i\rho_* e^{i\theta}$$

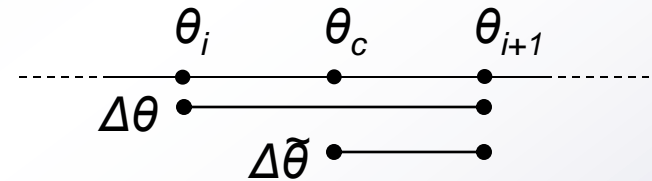
3. Advanced Contour Integration

Parabolic correction for connection terms

- For connection terms between created/destroyed pairs we use

$$\Delta A_C^{(p)} = \frac{1}{24} [(\mathbf{x}'_+ \wedge \mathbf{x}''_+) - (\mathbf{x}'_- \wedge \mathbf{x}''_-)] \Delta \tilde{\theta}^3$$

$$\Delta \tilde{\theta} = \frac{|\mathbf{x}_+ - \mathbf{x}_-|}{\sqrt{|\mathbf{x}_+ \cdot \mathbf{x}_-|}}$$



- Note that in this case the order of the errors is

$$\Delta A_C^{(t)} - \Delta A_C = O(\Delta \tilde{\theta}^{3/2})$$

$$\Delta A_C^{(t)} + \Delta A_C^{(p)} - \Delta A_C = O(\Delta \tilde{\theta}^{5/2})$$

- The errors are larger at caustic crossings.
- Denser sampling needed!

3. Advanced Contour Integration

Error control

- In all numerical computations it is fundamental to have an estimate of the errors.
- We must be able to assess the accuracy of our calculation.
- Conversely, we should know how dense our sampling should be to reach a given accuracy.
- Time is precious: if we control the errors, we can avoid useless computations.
- Error estimators should warn us of possible failures.
- The error estimators must be reliable but also cheap.

3. Advanced Contour Integration

Error estimators

- Here are three proposals

$$E_{I,i,1} = \frac{1}{48} \left| (\mathbf{x}'_I \wedge \mathbf{x}''_I) \Big|_{\theta_i} - (\mathbf{x}'_I \wedge \mathbf{x}''_I) \Big|_{\theta_i + \Delta\theta} \right| \Delta\theta^3$$

$$E_{I,i,2} = \frac{3}{2} \left| \Delta A_I^{(p)} \left(\frac{\Delta \tilde{\theta}^2}{\Delta \theta^2} - 1 \right) \right|$$

$$E_{I,i,3} = \frac{1}{10} \left| \Delta A_I^{(p)} \right| \Delta \theta^2$$

- These work in a complementary way and are combinations of quantities already calculated.

3. Advanced Contour Integration

Error estimators for connection terms

- Here are the versions for connection terms

$$E_1^{(c)} = \frac{1}{48} |\mathbf{x}'_+ \wedge \mathbf{x}''_+ + \mathbf{x}'_- \wedge \mathbf{x}''_-| \Delta \tilde{\theta}^3$$

$$E_2^{(c)} = \frac{3}{2} |(\mathbf{x}_+ - \mathbf{x}_-) \cdot (\mathbf{x}'_+ - \mathbf{x}'_-) \mp 2|(\mathbf{x}_+ - \mathbf{x}_-)|\sqrt{\mathbf{x}'_+ \cdot \mathbf{x}'_-}| \Delta \tilde{\theta}$$

$$E_3^{(c)} = \frac{1}{10} |\Delta A_C^{(p)}| \Delta \tilde{\theta}^2$$

3. Advanced Contour Integration

Error estimate at step i

- Our error estimate for the step i is thus

$$E_i = \sum_I (E_{I,i,1} + E_{I,i,2} + E_{I,i,3})$$

- If creation/destruction occurs at step i we add

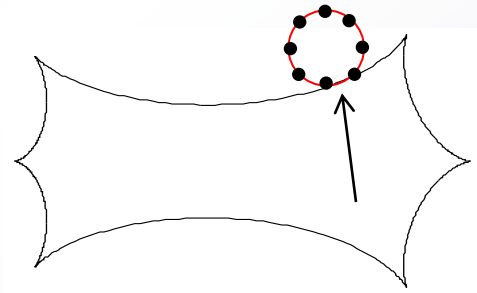
$$E_{i+} = E_1^{(c)} + E_2^{(c)} + E_3^{(c)}$$

3. Advanced Contour Integration

Hidden images

- Finally, it may happen that two images are created and destroyed within steps i and $i + 1$, without leaving traces.
- We then calculate the distance g_i between the two ghost images at each step.
- If g_i reaches a minimum, we check that by linear interpolation we do not go below zero, otherwise we add an error

$$E_{i+} = (g_i - g_{i+1})^2$$



3. Advanced Contour Integration

Optimal sampling

- The total error in the area of all images is

$$E = \sum_{i=0}^{n-1} E_i$$

- At this point we are able to check if we have reached the target accuracy $\delta\mu$ in the magnification:

$$\frac{E}{\pi\rho_*^2} < \delta\mu$$

- If not, we must increase the sampling.
- Can we do it in a clever way?

3. Advanced Contour Integration

Optimal sampling

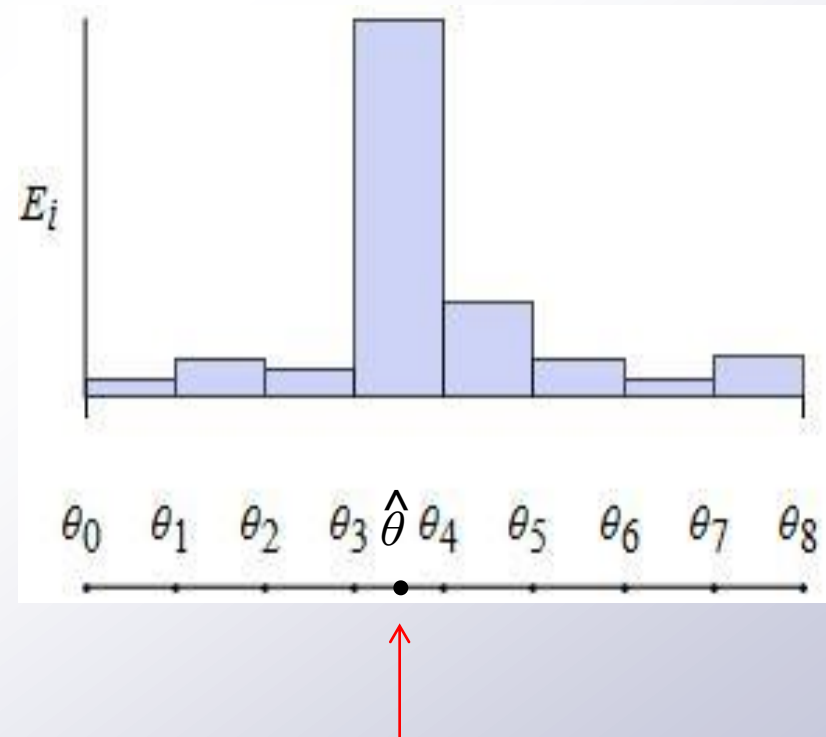
- We can pick the interval with the largest error

$$\text{Let } \hat{i} : E_i \leq E_{\hat{i}} \quad \forall i$$

- ... and add another point in the sample in the middle of this interval:

$$\hat{\theta} = \frac{\theta_{\hat{i}} + \theta_{\hat{i}+1}}{2}$$

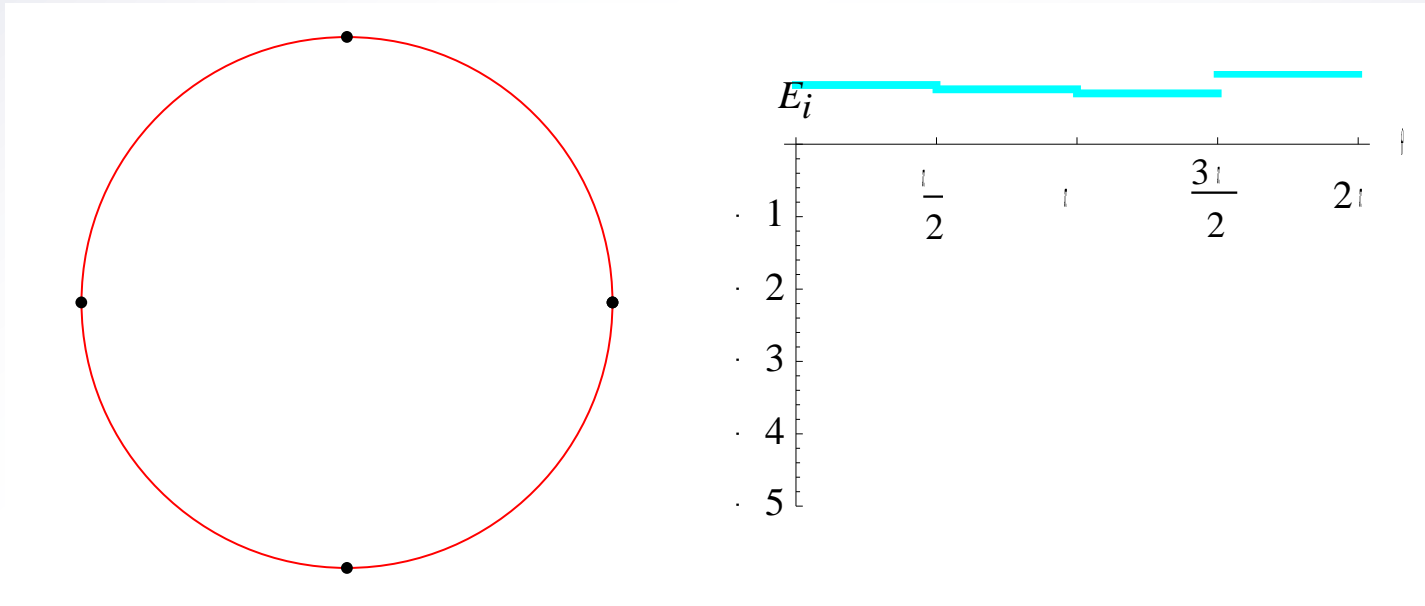
- Then we just need to recalculate the contour integral and the error estimators in the new sub-intervals.
- In this way, sampling is increased only where needed, avoiding useless calculations.



3. Advanced Contour Integration

Optimal sampling

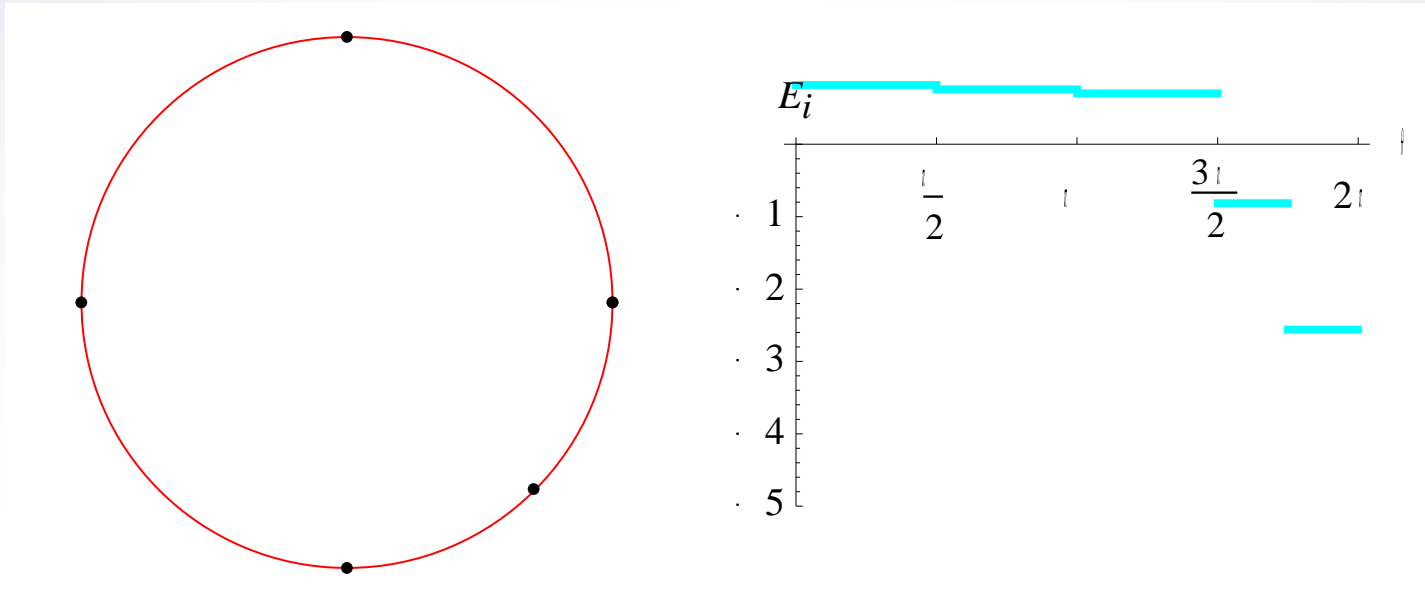
- We can start with a basic four points sample and then add new points according to the rule of largest E_i .



3. Advanced Contour Integration

Optimal sampling

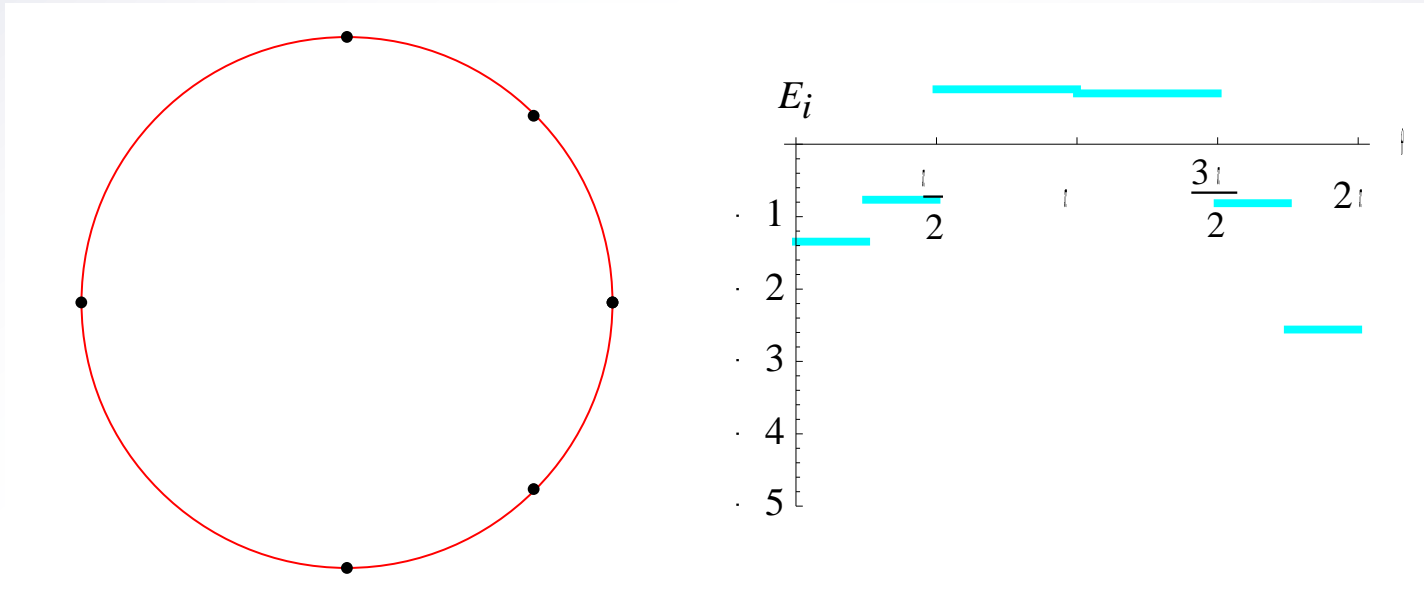
- We can start with a basic four points sample and then add new points according to the rule of largest E_i .



3. Advanced Contour Integration

Optimal sampling

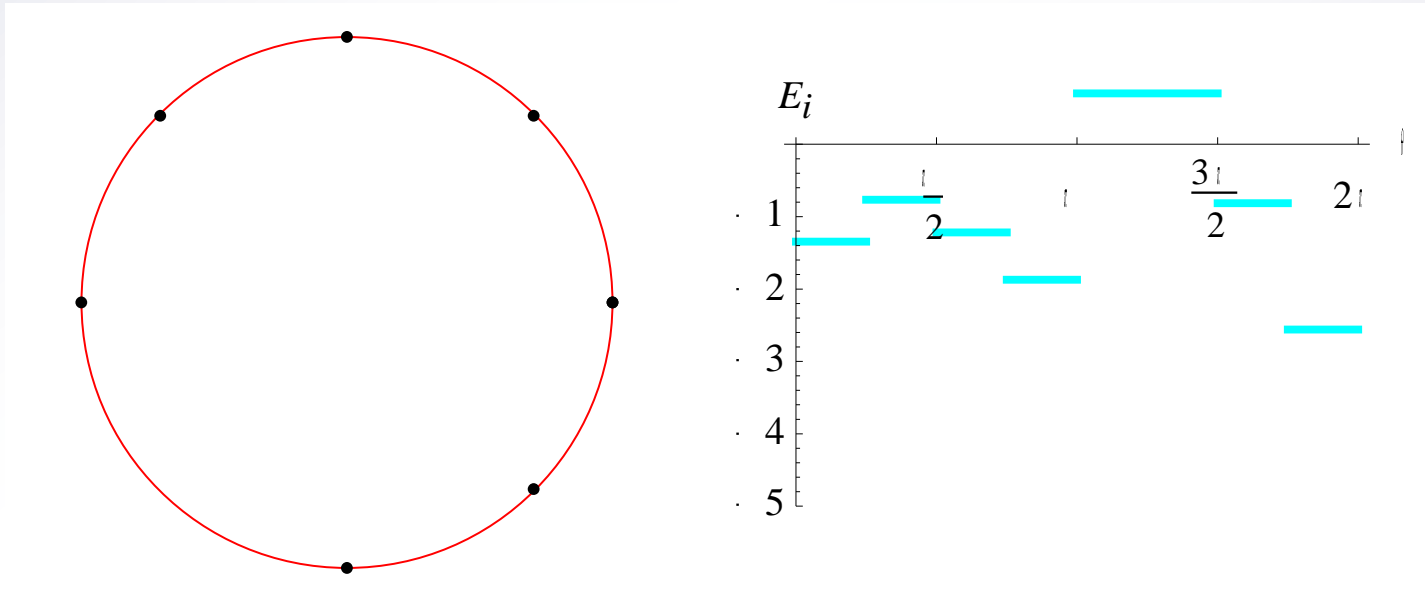
- We can start with a basic four points sample and then add new points according to the rule of largest E_i .



3. Advanced Contour Integration

Optimal sampling

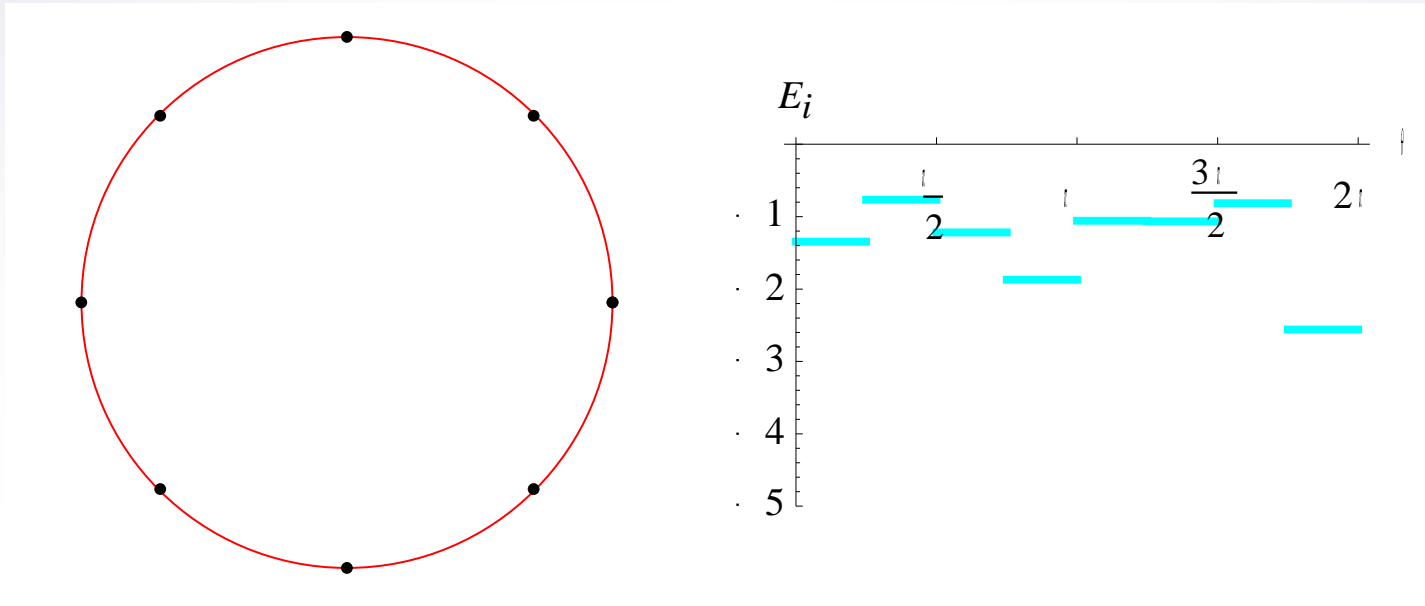
- We can start with a basic four points sample and then add new points according to the rule of largest E_i .



3. Advanced Contour Integration

Optimal sampling

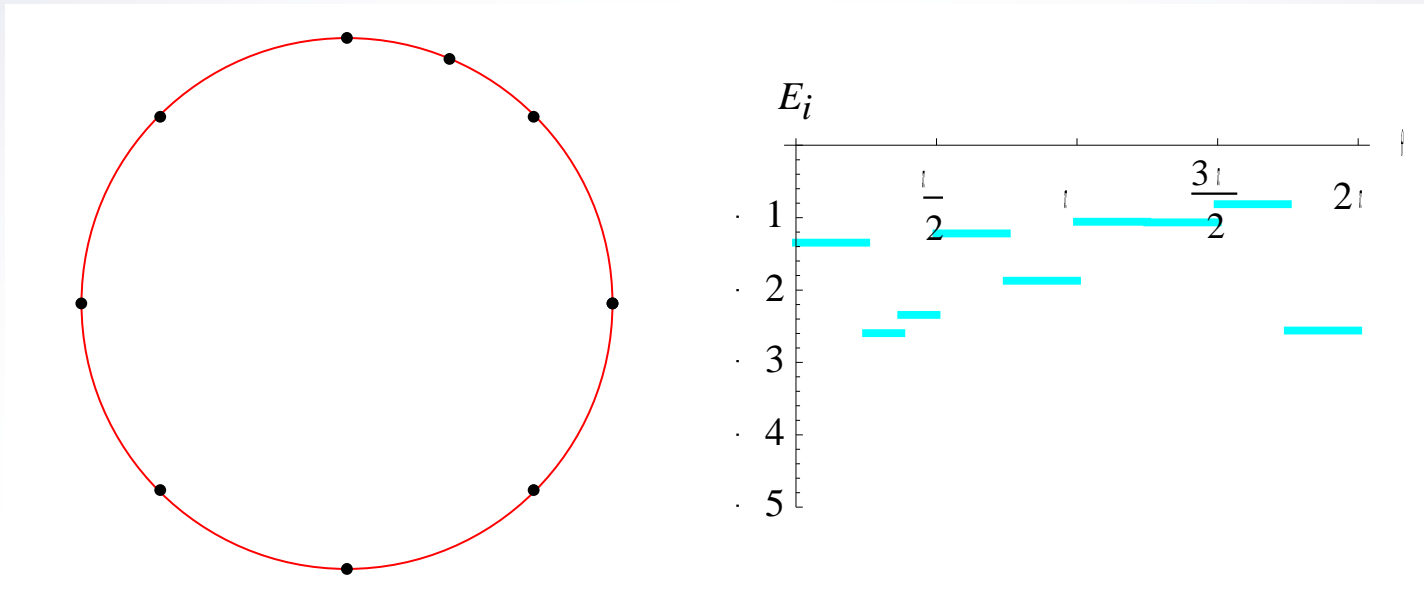
- We can start with a basic four points sample and then add new points according to the rule of largest E_i .



3. Advanced Contour Integration

Optimal sampling

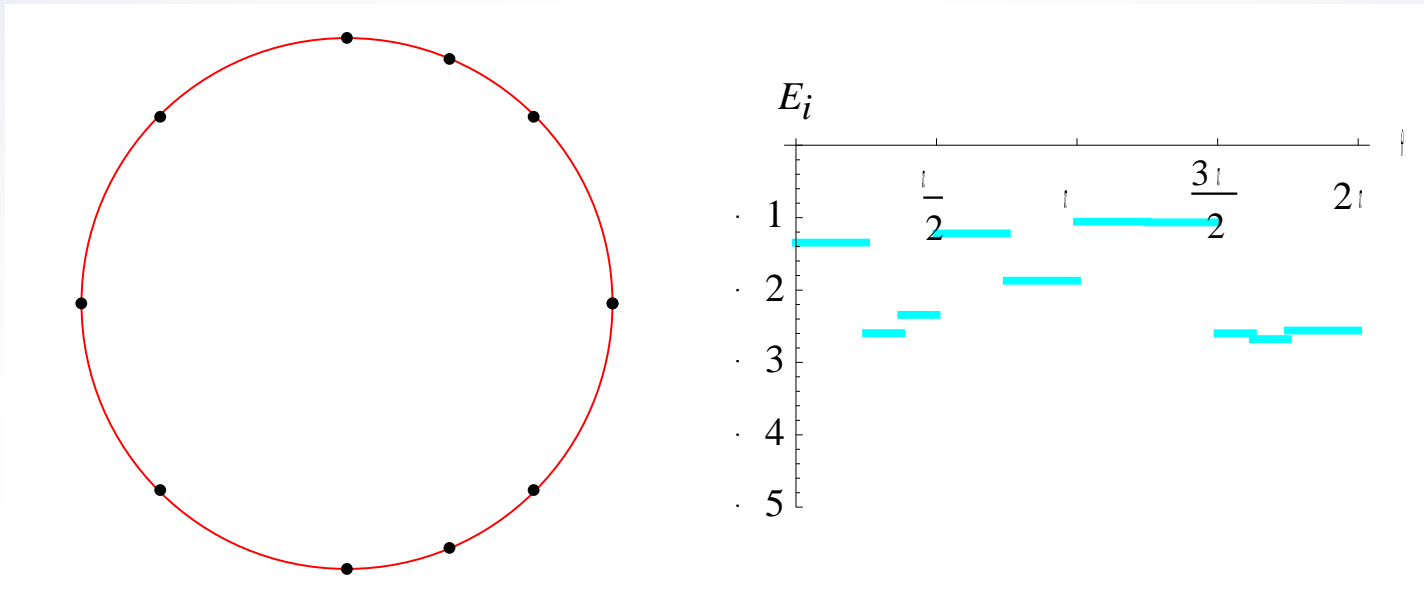
- We can start with a basic four points sample and then add new points according to the rule of largest E_i .



3. Advanced Contour Integration

Optimal sampling

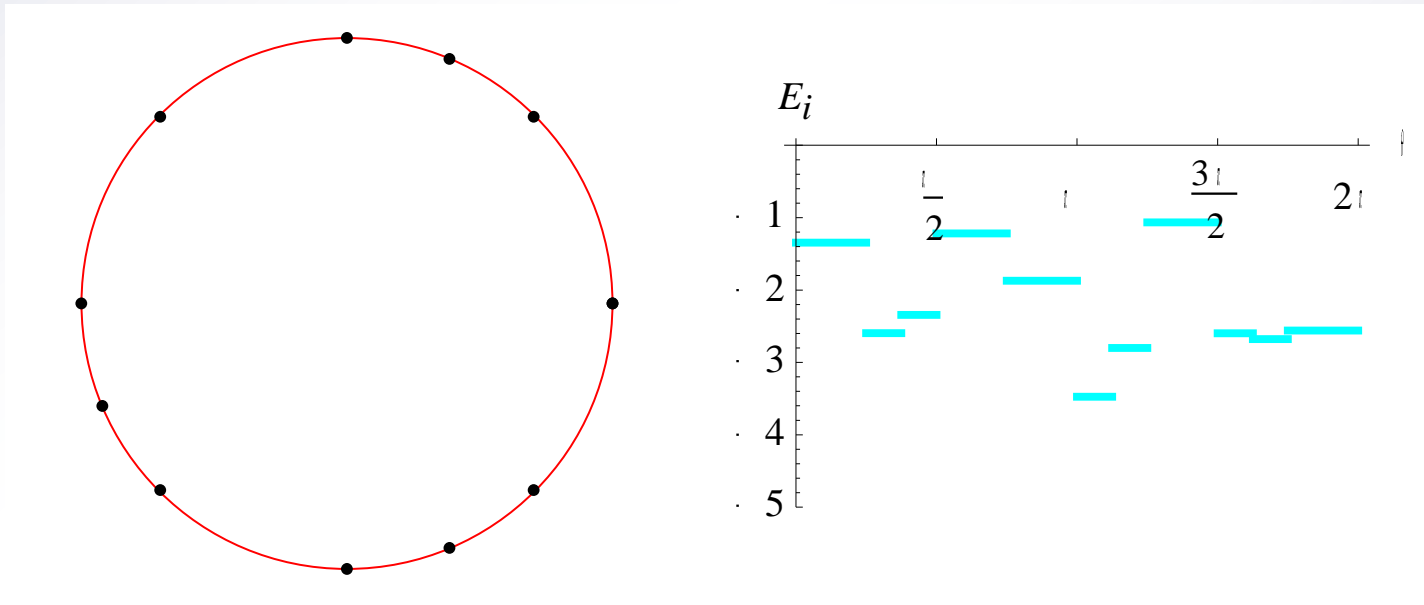
- We can start with a basic four points sample and then add new points according to the rule of largest E_i .



3. Advanced Contour Integration

Optimal sampling

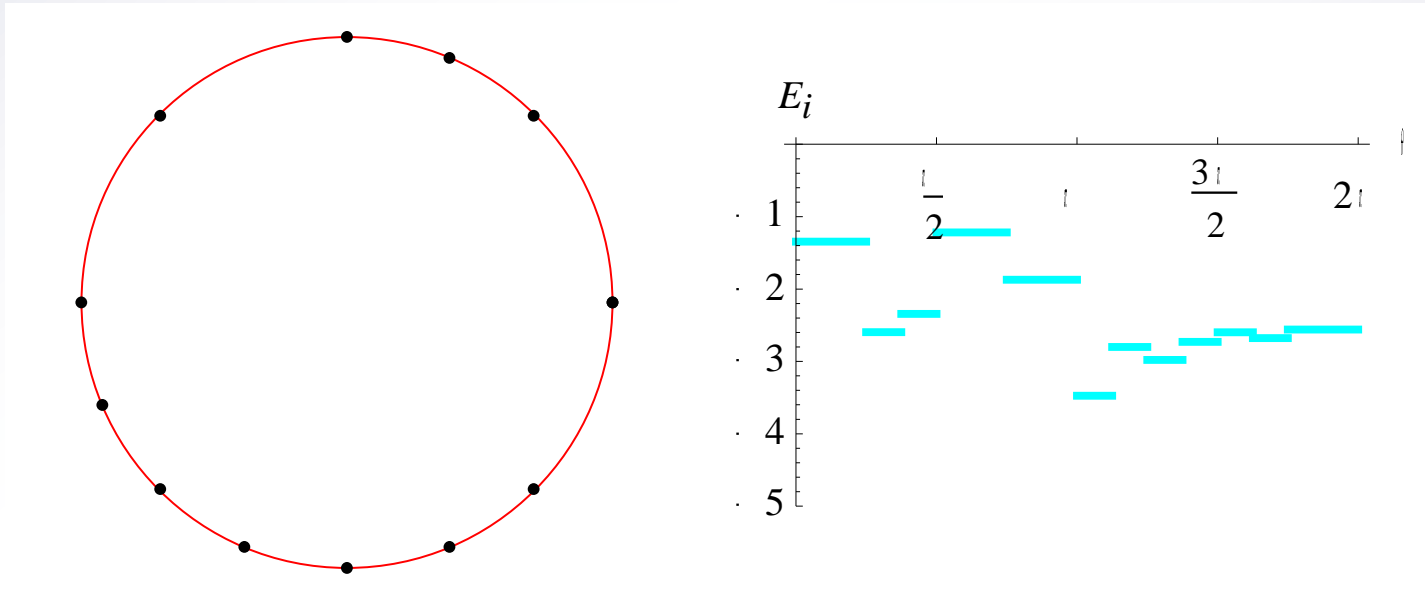
- We can start with a basic four points sample and then add new points according to the rule of largest E_i .



3. Advanced Contour Integration

Optimal sampling

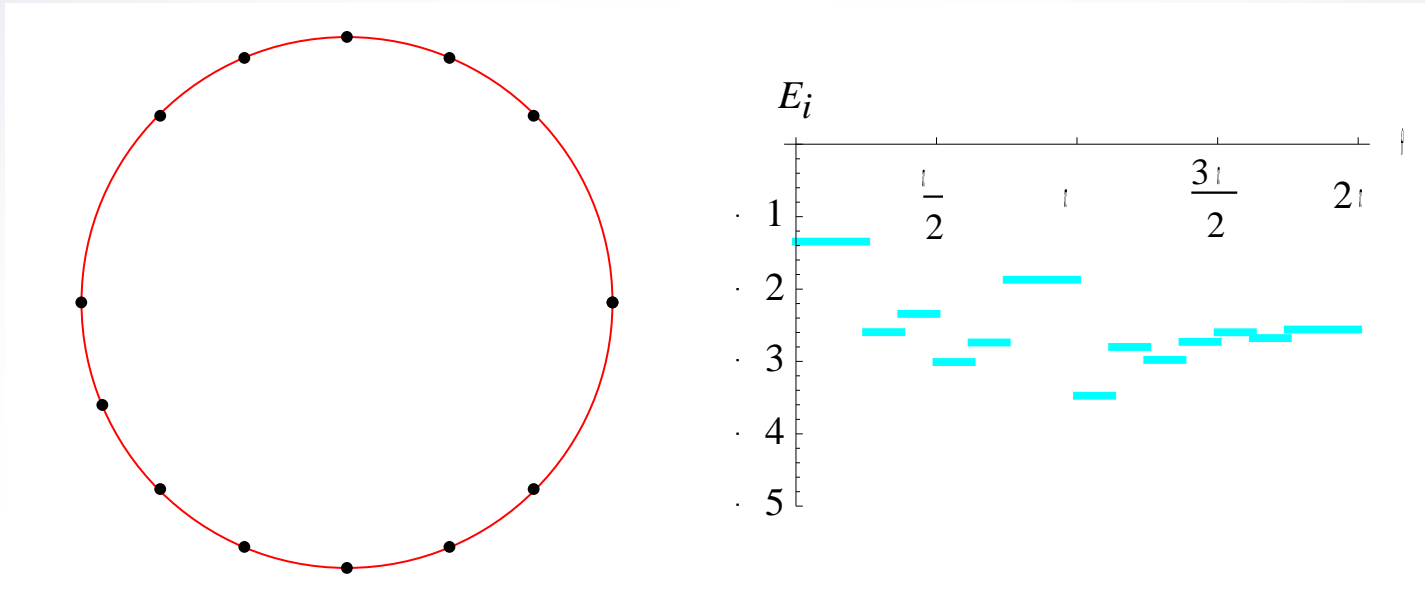
- We can start with a basic four points sample and then add new points according to the rule of largest E_i .



3. Advanced Contour Integration

Optimal sampling

- We can start with a basic four points sample and then add new points according to the rule of largest E_i .



3. Advanced Contour Integration

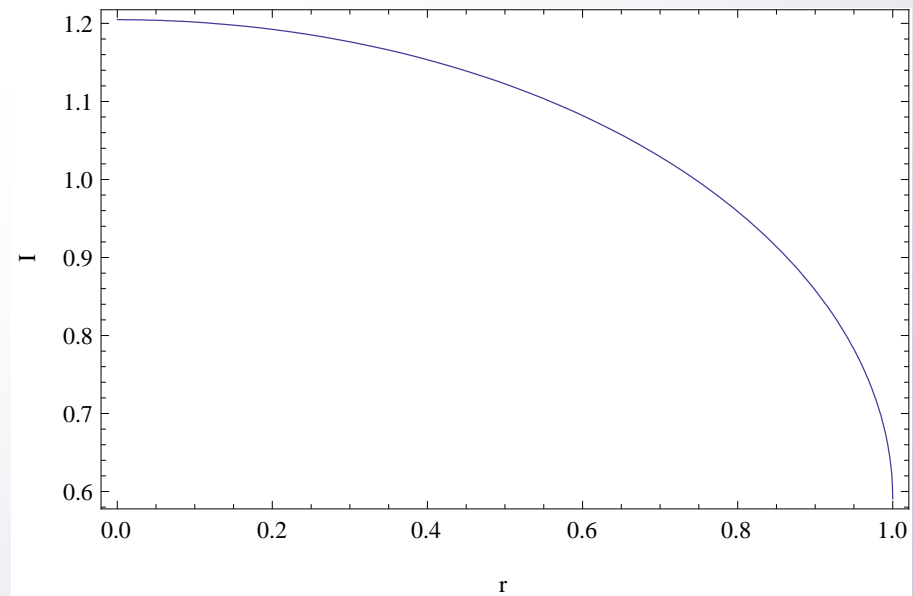
Limb darkening

- Up to now, we have assumed a uniform brightness source.
- However, physical stars have a limb-darkened profile, e.g. Milne's linear law

$$f(r) = \frac{1}{1-a/3} \left[1 - a \left(1 - \sqrt{1-r^2} \right) \right] \quad \text{with} \quad r = \frac{\rho}{\rho_*}$$

- In general, the source profile is a function $f(r)$, normalized in such a way that

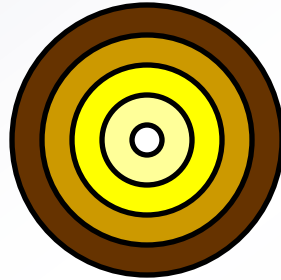
$$\int_0^1 2rf(r)dr = 1$$



3. Advanced Contour Integration

Limb darkening

- The only known way to deal with limb darkening with contour integration is by dividing the source in annuli.



- Each source annulus is magnified by microlensing. The exact contribution to the total amplification is

$$M_i = \frac{1}{\pi} \int_{r_{i-1}}^{r_i} dr 2r f(r) \int_0^{2\pi} \mu(r, \varphi) d\varphi$$

3. Advanced Contour Integration

Limb darkening

- In each annulus we instead use a uniform brightness given by the limb-darkened profile averaged on the annulus

$$f_i = \frac{F(r_i) - F(r_{i-1})}{r_i^2 - r_{i-1}^2} \quad \text{with} \quad F(r) = \int_0^r dr' 2r' f(r')$$

- The approximate contribution to the amplification is

$$\tilde{M}_i = f_i \left[\mu_i r_i^2 - \mu_{i-1} r_{i-1}^2 \right] \quad \text{where}$$

$$\mu_i = \frac{1}{\pi r_i^2} \int_{r_{i-1}}^{r_i} dr 2r \int_0^{2\pi} \mu(r, \varphi) d\varphi$$

- The latter is the magnification factor for a uniform disk of radius r_i , calculable by contour integration.

3. Advanced Contour Integration

Errors in limb darkening

- The difference between the exact and approximate expression is of third order in the annulus thickness.
- The estimators we use are

$$\delta\tilde{M}_i^{(1)} = \frac{1}{4} [r_i^2 - r_{i-1}^2] [f(r_i) - f(r_{i-1})] |\mu_i - \mu_{i-1}|$$

$$\delta\tilde{M}_i^{(2)} = \frac{1}{4} [r_i^2 - r_{i-1}^2] [f(r_i) - f(r_{i-1})] |\mu_{i+1} + \mu_{i-1} - 2\mu_i|$$

$$\delta\tilde{M}_i^{(c)} = \frac{1}{4} [r_i^2 \mu_i - r_{i-1}^2 \mu_{i-1}] [f(r_i) - f(r_{i-1})]$$

3. Advanced Contour Integration

Sampling the source profile

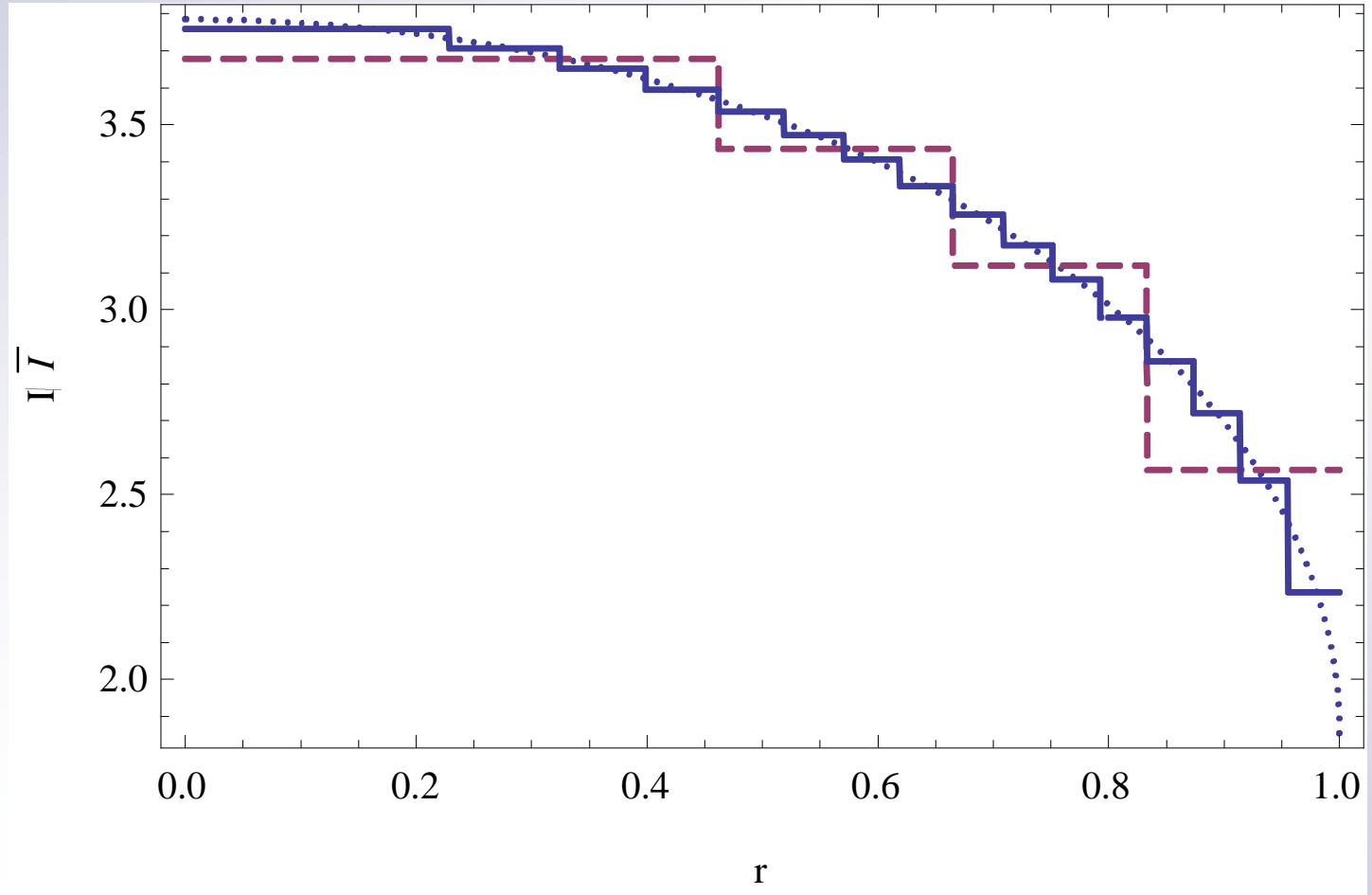
- Given a source sampling $0 = r_0 < r_1 < \dots < r_n = 1$, we split the annulus with the largest error.
- The new circle is put at an intermediate radius \bar{r} so that the two new annuli give the same contribution to the source luminosity:

$$F(r_j) - F(\bar{r}) = F(\bar{r}) - F(r_{j-1})$$

- We start with the two extremal annuli: the boundary ($r=1$) and the center ($r=0$).
- We keep introducing annuli until the total error falls below the target accuracy.

3. Advanced Contour Integration

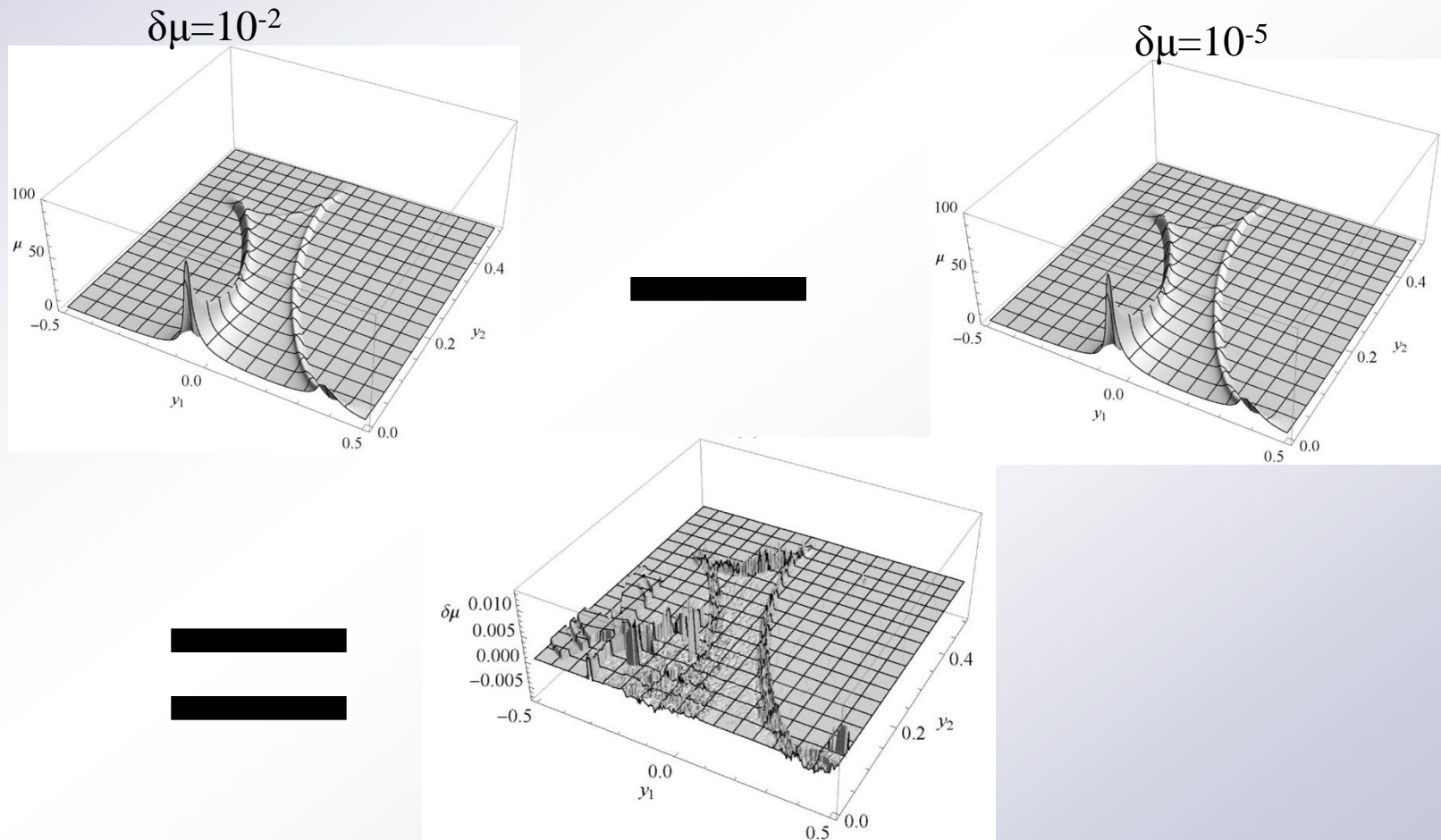
Sampling the source profile



3. Advanced Contour Integration

Testing

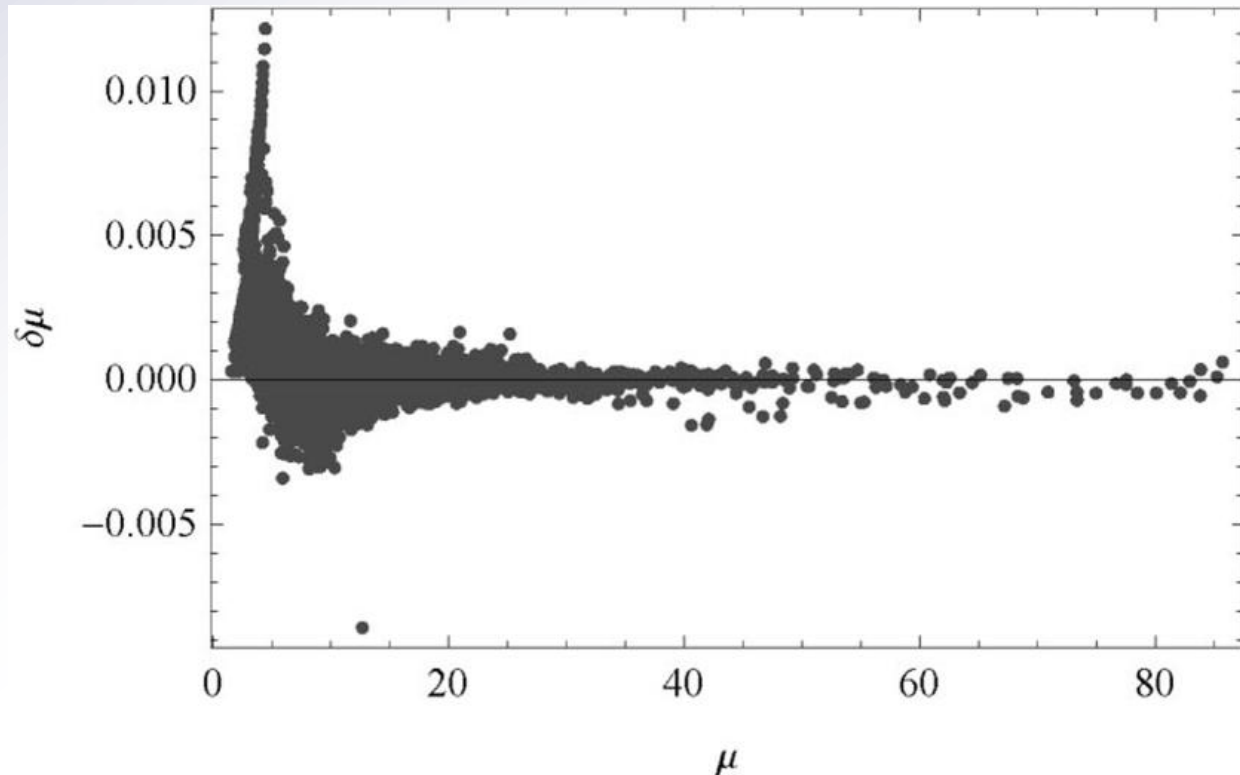
- In order to test our code, we can build maps with different target accuracies and take the difference.



3. Advanced Contour Integration

Testing

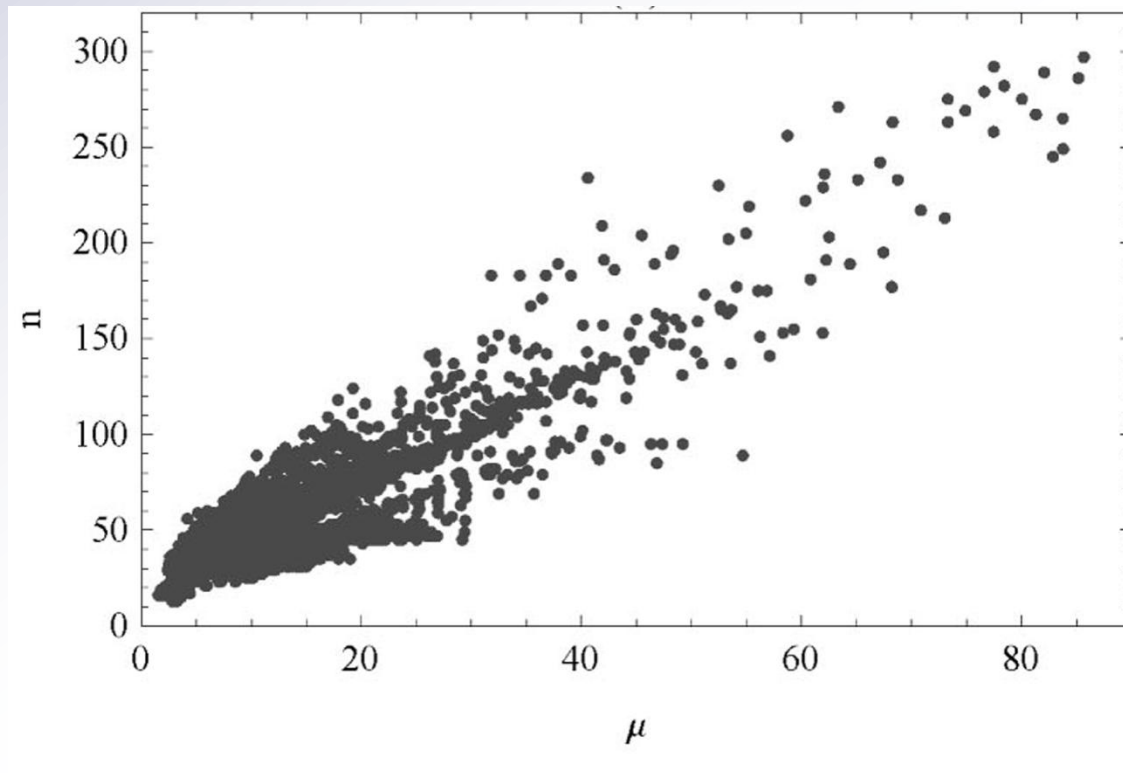
- This is a scatter plot of error vs magnification (target accuracy is 10^{-2}).



3. Advanced Contour Integration

Testing

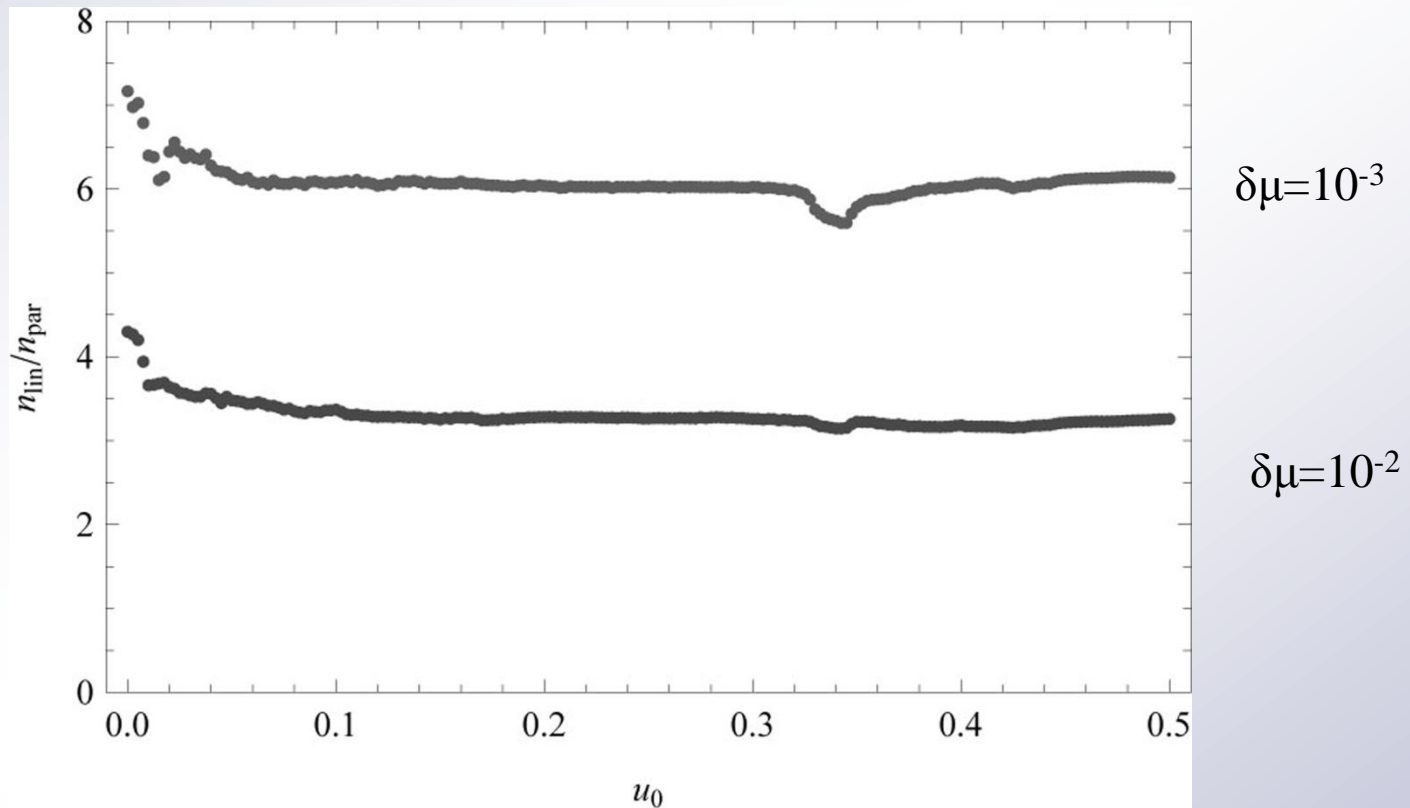
- This is a scatter plot of number of sampling points vs magnification (target accuracy is 10^{-2}).



3. Advanced Contour Integration

Testing

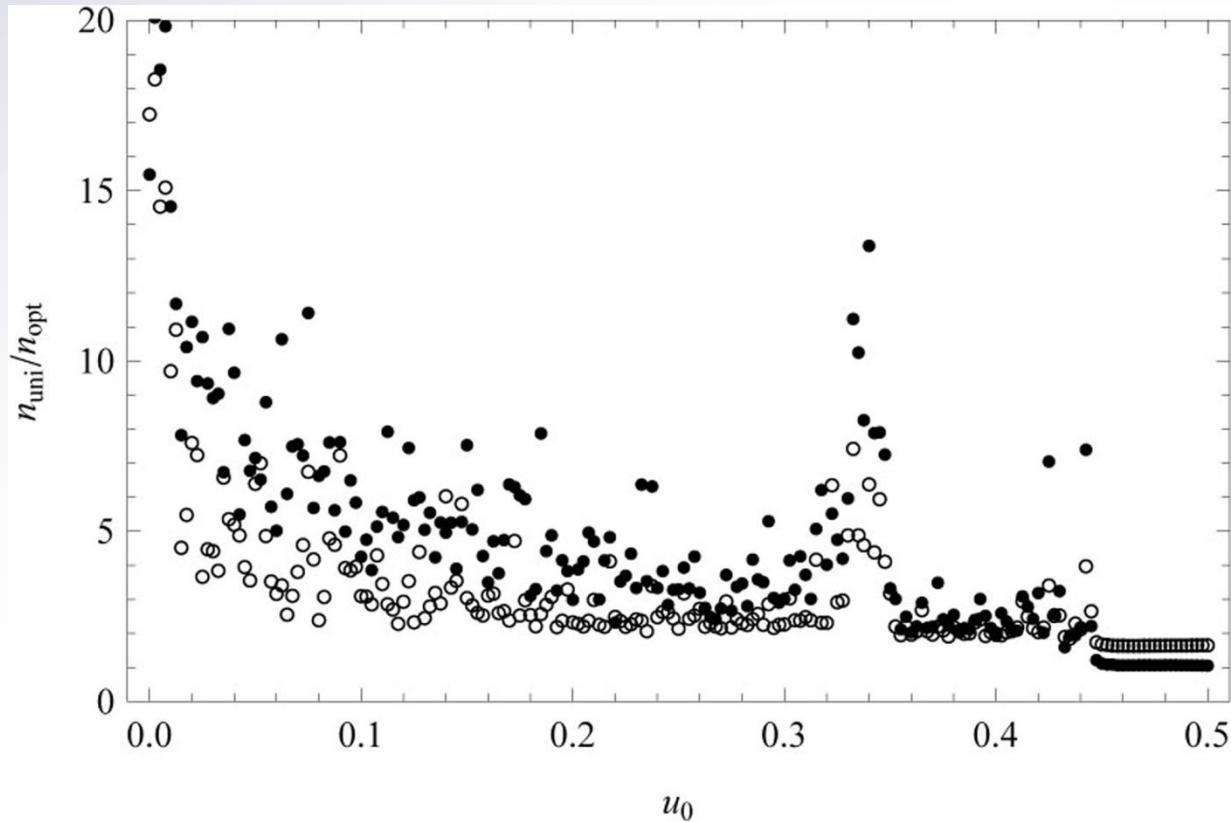
- Gain in the number of sampling points with the parabolic correction.



3. Advanced Contour Integration

Testing

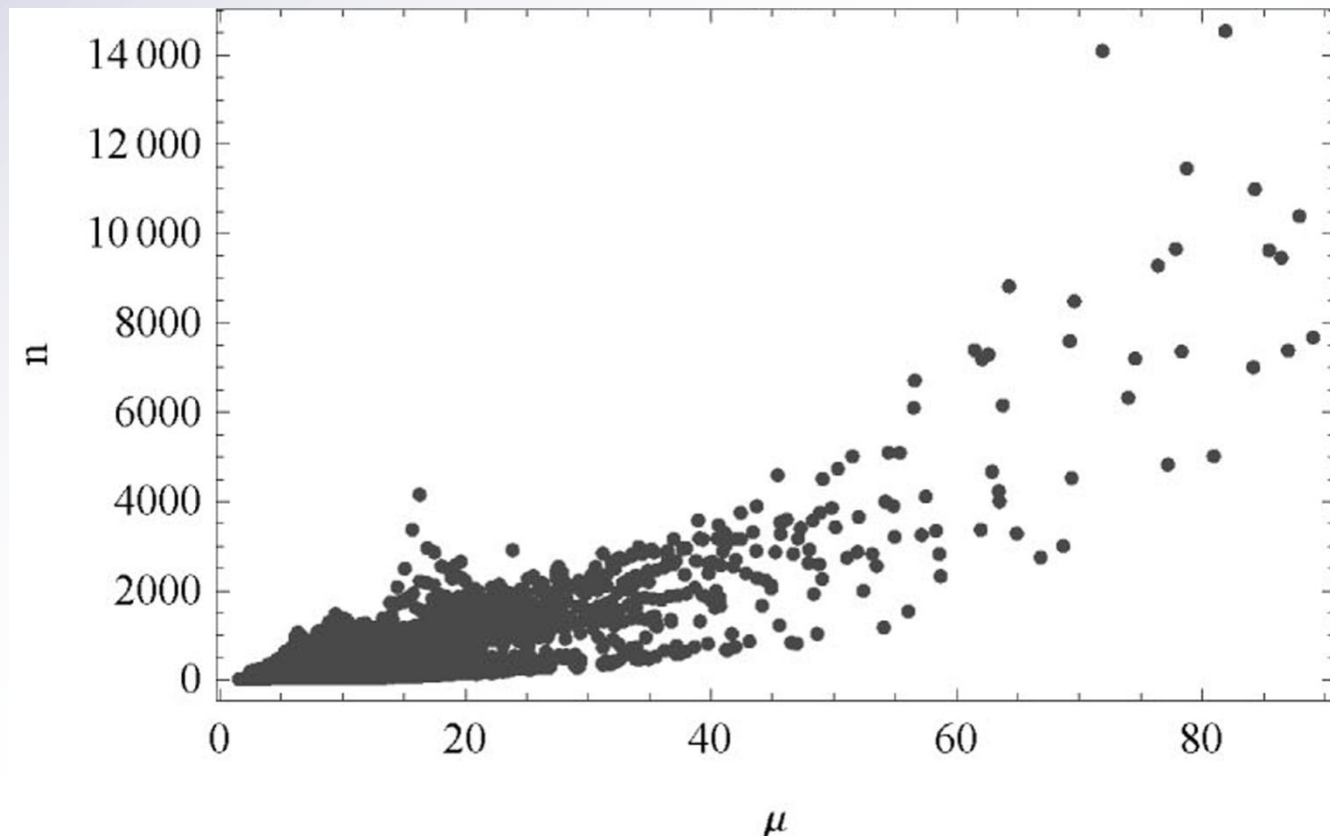
- Gain in the number of sampling points with the optimal sampling.



3. Advanced Contour Integration

Testing

- Number of points vs magnification with limb darkening.



Testing

- Summing up, at $\delta\mu=10^{-2}$ we get
 - a speed-up of 4 thanks to parabolic correction
 - a speed-up ranging from 3 to 20 thanks to optimal sampling
 - a slow-down from 2 to 10 if we include limb darkening
- No redundant calculation thanks to error estimators!

4. Downhill fitting

4. Downhill fitting

Fitting a microlensing event

- Once we are able to calculate the microlensing magnification for a given lens model and source position, we can try to fit microlensing events.
- Binary microlensing events are characterized by a minimum of 7 parameters.
- Note that the calibrations of all datasets (background and source fluxes) can be found analytically by a least-squares fit.

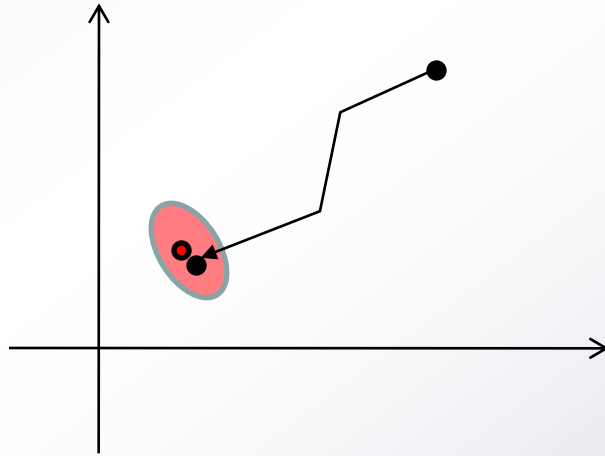
$$y_i = F_* f(t_i, \mathbf{p}) + F_B$$

- How do we find a best fit of the microlensing parameters?

4. Downhill fitting

Fitting a microlensing event

- Make an initial guess (a point in the parameter space)
- Find a minimum in χ^2
(Downhill fit)
- Explore the minimum and refine the best solution
(Markov chain)



Downhill fitting

- Concept: find a direction going down and just follow it...
- Several possibilities: different convergence speeds.
 - Steepest descent
 - Gauss-Newton method
 - Levenberg-Marquardt algorithm

Fitting Problem

- We want to find a minimum in

$$\chi^2(\mathbf{p}) = \sum_{i=1}^n [y_i - f(t_i, \mathbf{p})]^2$$

where \mathbf{p} is the vector of microlensing parameters, (t_i, y_i) are the data points and $f(t_i, \mathbf{p})$ is the model at time t_i according to parameters \mathbf{p} .

- If we are in \mathbf{p}_i , where should we choose \mathbf{p}_{i+1} ?

Steepest descent

- The gradient of χ^2 is just

$$\nabla \chi^2(\mathbf{p}) = -2 \sum_{i=1}^n \mathbf{J}_i [y_i - f(t_i, \mathbf{p})]$$

$$\text{where } \mathbf{J}_i = \left(\frac{\partial f_i}{\partial p_1}, \dots, \frac{\partial f_i}{\partial p_m} \right)$$

- The steepest descent is then implemented by choosing

$$\mathbf{p}_{i+1} = \mathbf{p}_i - \alpha \nabla \chi^2$$

- α is determined by a search along the line.

4. Downhill fitting

Gauss-Newton method

- Let us set $\mathbf{p}_{i+1} = \mathbf{p}_i + \Delta$.
- If Δ is such that \mathbf{p}_{i+1} is a minimum, then

$$0 = \nabla \chi^2(\mathbf{p}_i + \Delta) = -2 \sum_{i=1}^n \mathbf{J}_i [y_i - f(t_i, \mathbf{p}_i + \Delta)] \cong$$

$$\cong -2 \sum_{i=1}^n \mathbf{J}_i [y_i - f(t_i, \mathbf{p}_i) - \mathbf{J}_i \cdot \Delta]$$

- The approximate solution for Δ is obtained by a linear set of equations

$$\sum_{i=1}^n \mathbf{J}_i [\mathbf{J}_i \cdot \Delta] = \sum_{i=1}^n \mathbf{J}_i [y_i - f(t_i, \mathbf{p}_i)]$$

- Convergence is not guaranteed if we are too far from minimum

4. Downhill fitting

Levenberg method

- Interpolates between the two methods, switching from Gauss-Newton to steepest descent when the first fails.
- We modify the normal equations by introducing a parameter λ

$$\sum_{i=1}^n \mathbf{J}_i [\mathbf{J}_i \cdot \Delta] + \lambda \Delta = \sum_{i=1}^n \mathbf{J}_i [y_i - f(t_i, \mathbf{p}_i)]$$

- If λ is small, the normal equations work as in Gauss-Newton.
- If λ is large, the new term dominates and Δ is rotated toward the steepest descent direction.

4. Downhill fitting

Levenberg-Marquardt algorithm

- Steepest descent may be inefficient if there are directions in which χ^2 is very flat.
- The final version of the modified normal equations is

$$\sum_{i=1}^n \left[\mathbf{J}_i (\mathbf{J}_i \cdot \Delta) + \lambda |\mathbf{J}_i|^2 \Delta \right] = \sum_{i=1}^n \mathbf{J}_i [y_i - f(t_i, \mathbf{p}_i)]$$

- In Levenberg-Marquardt algorithm, we start from a value of λ close to 1.
- We calculate Δ ; if $\chi^2(\mathbf{p}_i + \Delta) < \chi^2(\mathbf{p}_i)$, we accept the new point $\mathbf{p}_{i+1} = \mathbf{p}_i + \Delta$ and decrease λ .
- If not, we reject the new point and increase λ .

4. Downhill fitting

Implementation of Levenberg-Marquardt

- We need to calculate the gradient vector $\mathbf{J}_i = \left(\frac{\partial f_i}{\partial p_1}, \dots, \frac{\partial f_i}{\partial p_m} \right)$
- The derivatives require the calculation of magnification at two points spaced by dp_i . This is the slowest step.
- The resolution of normal equations can be done by standard Gauss method, Cholesky decomposition...
- Levenberg-Marquardt algorithm (nearly) always finds a local minimum.
- It is also very very fast.
- It might get stuck at a local minimum.
- How do we find the best minimum?

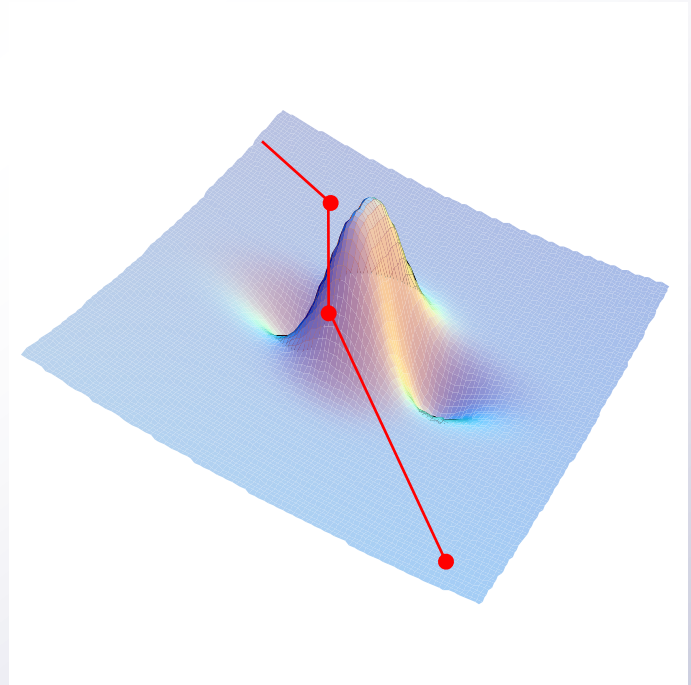
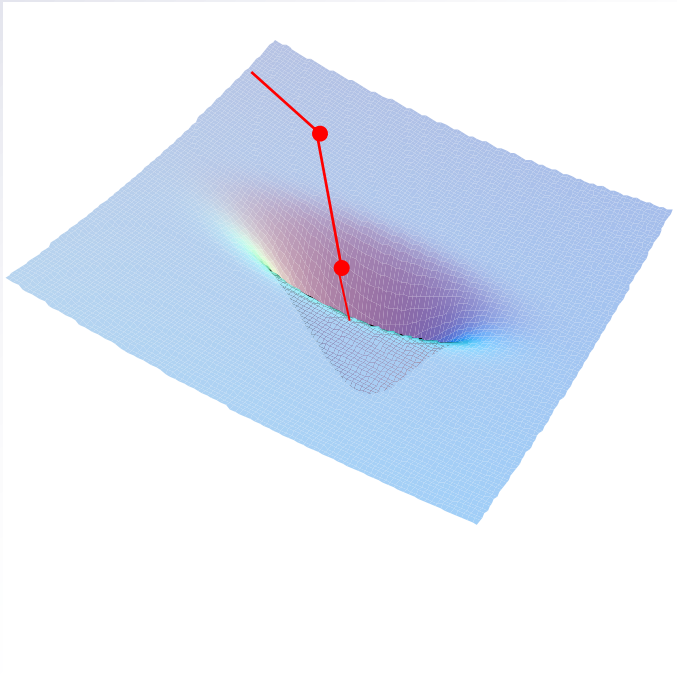
4. Downhill fitting

Jumping out of minima

- One possibility to enlarge our search is to add a penalty on the χ^2 function.
- Once we find the first minimum, we try to fill it with a bumper and run the fit again.
- If the bumper is small, the fit will still remain in the same hole.
- If the bumper is large enough, the fit will jump out of the hole and discover a different minimum.

4. Downhill fitting

Jumping out of minima



Conclusions

- Contour integration is a very elegant way to calculate the microlensing magnification.
- More complicated than ray shooting but very effective.
- Parabolic correction and optimal sampling do boost contour integration code. Error control is essential.
- Limb darkening treatment possible to arbitrary accuracy
- Downhill fitting is the fastest way to find a minimum from a given initial condition.
- Yet any exploration of the parameter space is always partial.
- There is no practical way to check that any given χ^2 -minimum is really the lowest possible one.